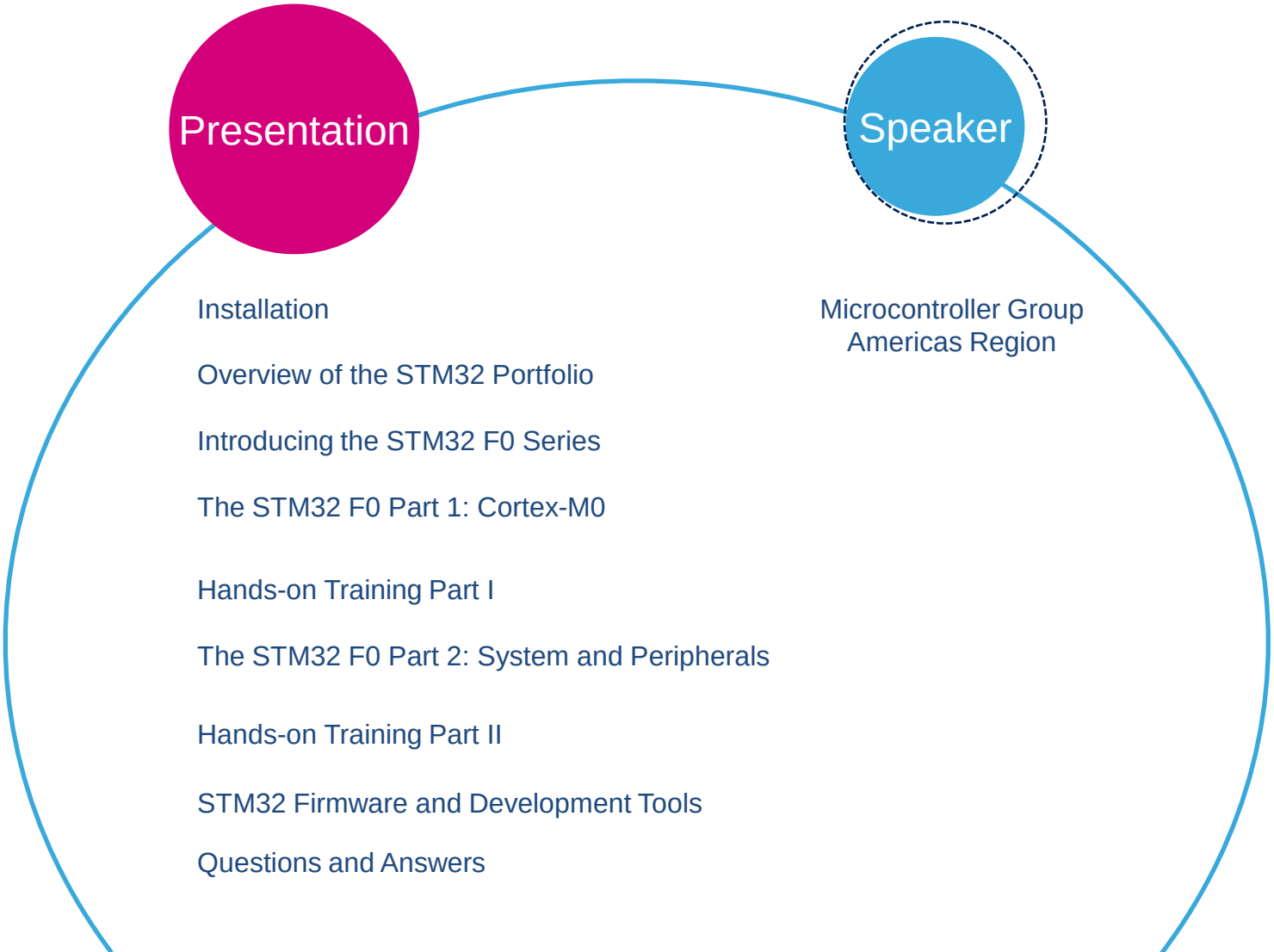
Hands-On Workshop Highlights ease of developing with Cortex-M0 MCU

life.augmented

Objectives

- Introduction to the ARM Cortex-M microcontroller
- Learn how to build, program, and debug an embedded system
- Free STM32F0 Discovery Kit for each participant

Agenda



Installation

Overview of the STM32 Portfolio

Introducing the STM32 F0 Series

The STM32 F0 Part 1: Cortex-M0

Hands-on Training Part I

The STM32 F0 Part 2: System and Peripherals

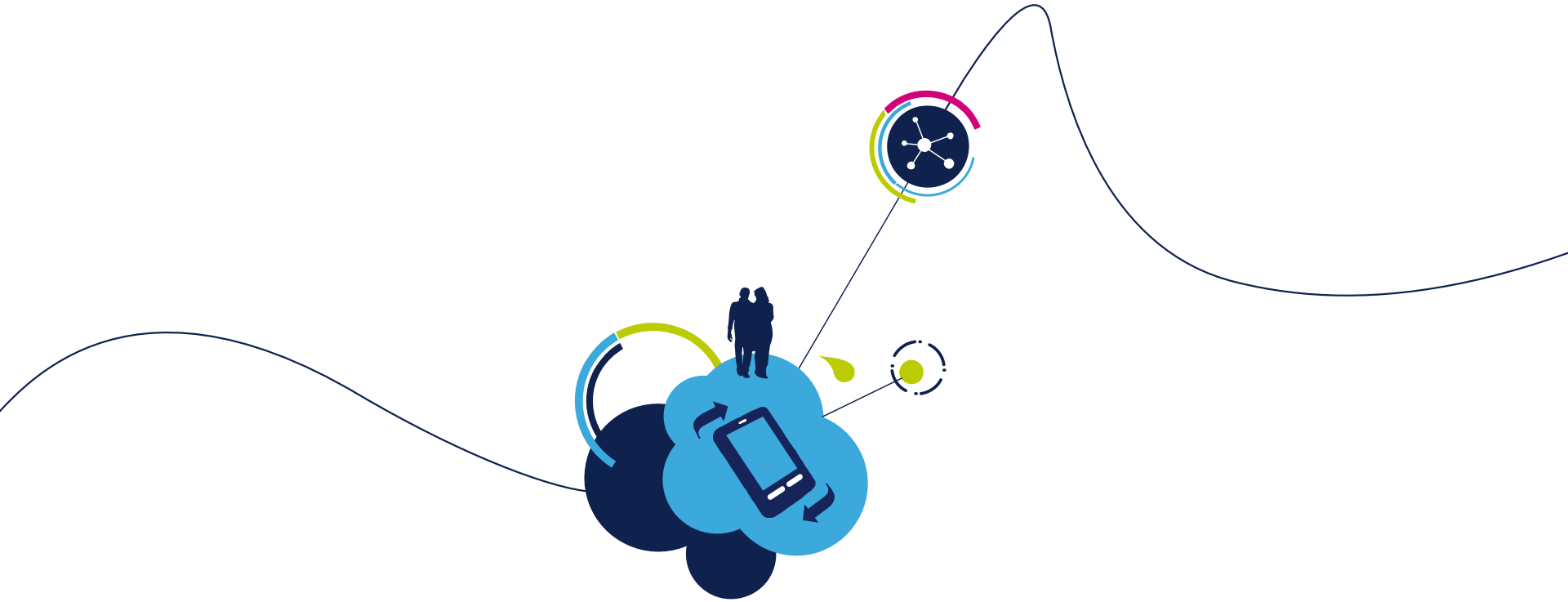
Hands-on Training Part II

STM32 Firmware and Development Tools

Questions and Answers

Microcontroller Group
Americas Region

life.augmented



Tool Installation

Systems Check

- Everyone should have
  - A Windows Laptop (XP, Vista or Windows 7)
  - USB Flash Drive
  - The following will be provided after software installation:
    - USB Cable
    - STM32F0DISCOVERY kit

- Ready to begin?

- **Note: Please do not attempt to plug in the STM32 F0 Discovery Kit into your laptop until instructed to do so.**

Step #1 - File Installation

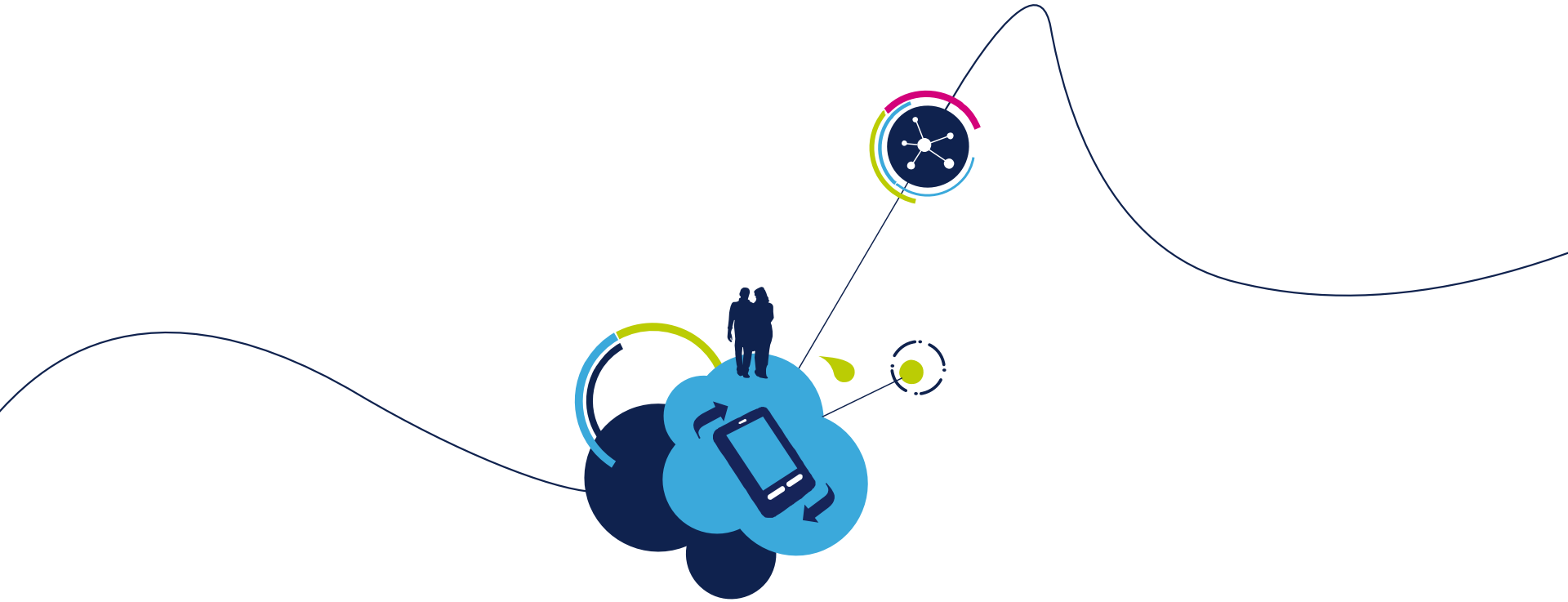
- Insert the USB Flash Drive into your Laptop
- Copy the folder “d:\STM32F0-Discovery_Kit” on the USB flash drive to your root “c:\” folder
  - C:\STM32F0-Discovery_Kit\
- Enter this directory. You will find the following:
  - In the \ARM_Keil_MDK directory: Keil µVision set-up file
  - Documentation folder containing all relevant documentation for this training
  - Discovery Kit Firmware folder

Keil µVision Version

- For this workshop, we will be using the evaluation version of the Microcontroller Development Kit from ARM. Some restrictions apply:
  - Program and debug up to 32 Kbytes of code
  - No disassembly listing
  - Some restriction on linkage usage
  - Limited base address usage

Step #2 - Install Keil µVision

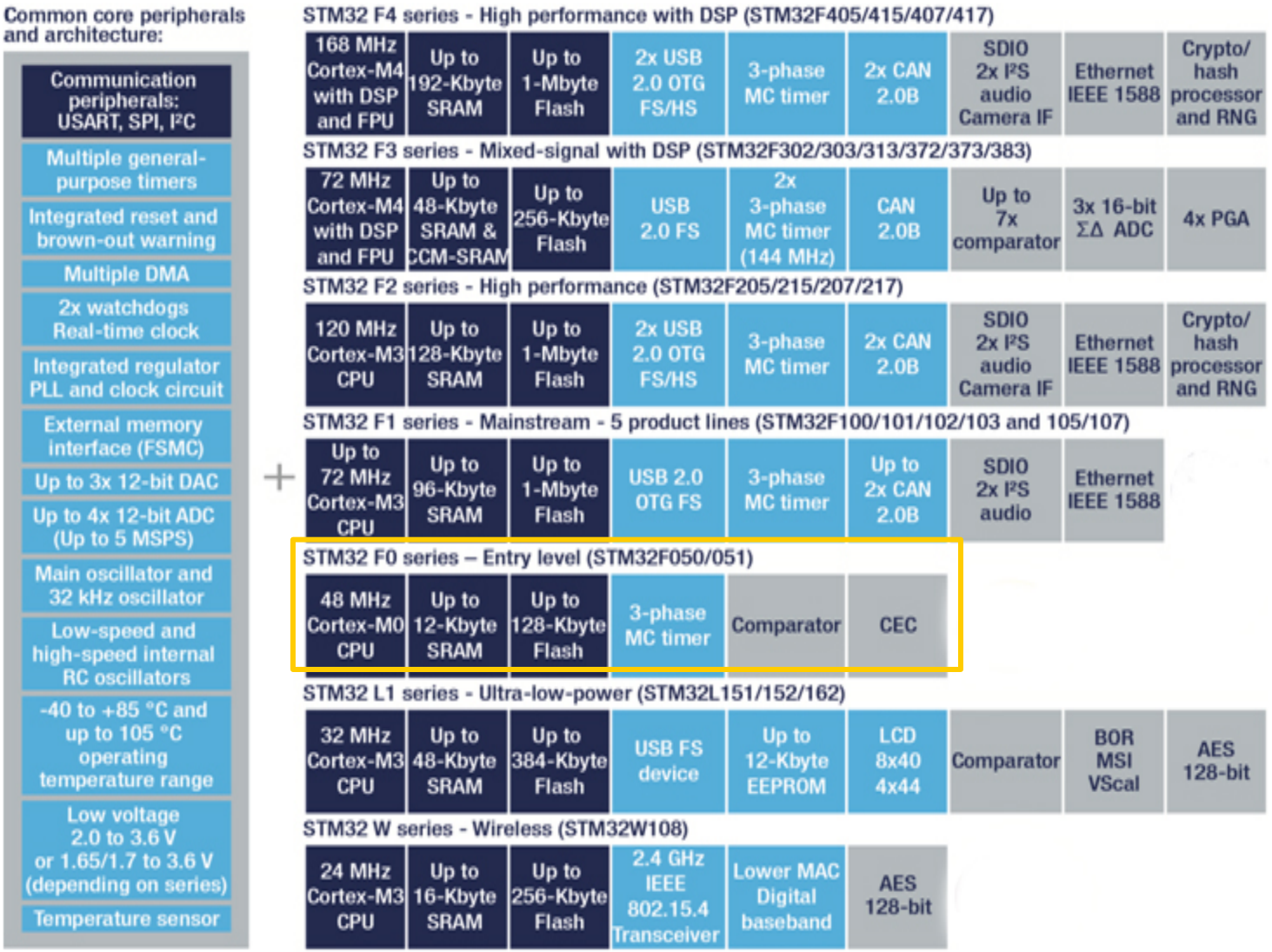
- Double-click on the file ***mdkxxx.exe*** to begin installation (***xxx***=version number). Please click-through the default options and accept the license agreement
- At the end of installation you will be asked if you want to add example projects to the recently used project list. Nothing specific is required for the hands-on exercises.
- The final dialog box:
  - Don't select – *Launch Driver Installation: "**ULINK Pro Driver V1.0**"*
    - We will be using the embedded ST-Link on the Discovery Kit
  - Optional – *Show Release Notes*
- Ask for assistance if you have an question or issue



Overview of the STM32 Portfolio

life.augmented

STM32 – 7 product series



STM32 applications

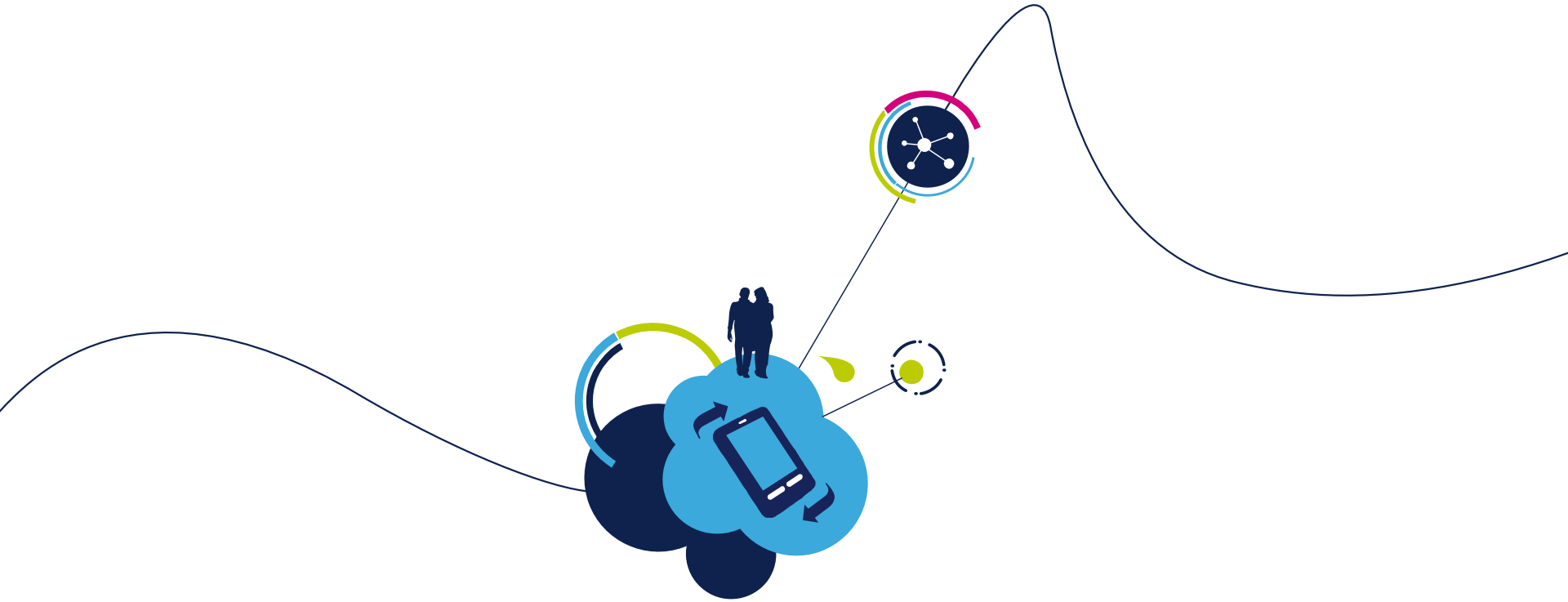
- Industrial
  - PLC
  - Inverters
  - Printers, scanners
  - Industrial networking
  - Solar inverters

- Medical
  - Glucose meters
  - Portable medical care
  - VPAP, CPAP
  - Patient monitoring

- Meets Building and security
  - Alarm systems
  - Access control
  - HVAC
  - Power meters

- Appliances
  - 3-phase motor drive
  - Application control
  - User interfaces
  - Induction cooking

- Consumer
  - Home audio
  - Gaming
  - PC peripherals
  - Digital cameras, GPS



Introducing STM32 F0 Series

life.augmented

STM32 F0 series: key features

Real-time performance

Cortex Intelligent Processors by ARM

48 MHz/38 DMIPS,
5 channels DMA
mapped on 11 IPs +
bus matrix allows Flash
execution in parallel with
DMA transfer

Power efficiency

5 µA in Stop mode
2 µA in standby mode
0.4 µA Vbat with RTC,
1.8 or 2 V to 3.6 V supply,
Fast wake-up time

Superior and innovative peripherals

1 Mbit/s I²C Fast mode+
SPI with
4- to 16-bit data frame,
HDMI CEC,
16-bit 3-phase MC timer

Maximum integration

Calendar RTC with
independent supply,
battery backed RAM,
separate analog supply,
safety

Extensive tools and software

ARM + ST ecosytem
(eval board,
discovery kit,
SW library)

STM32 F0

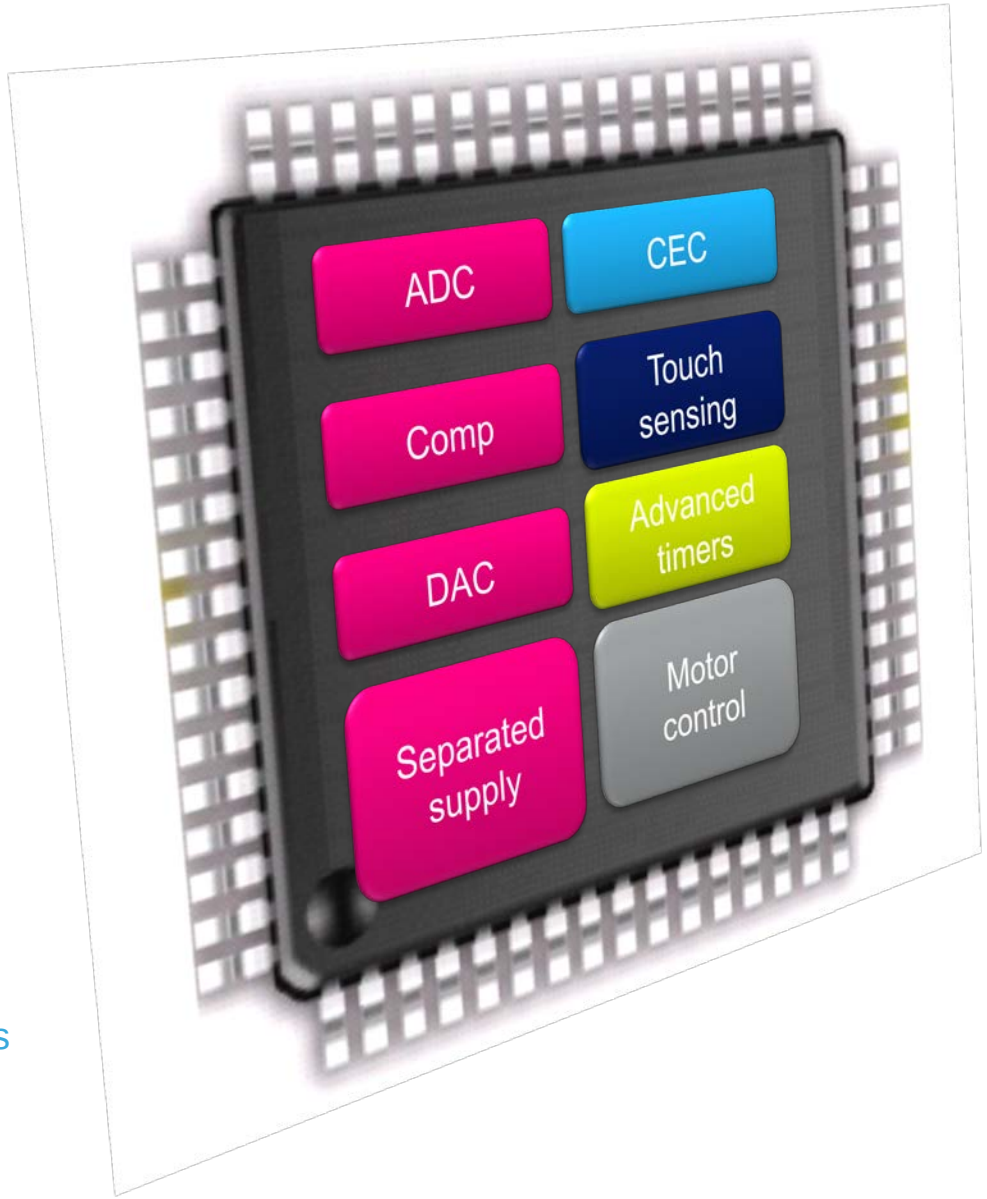
STM32 F0 series

W, L1, F0, F1, F2, F3, F4 series: seamless migration amongst 350 pin-to-pin compatible devices

life.augmented

Innovative peripheral mix

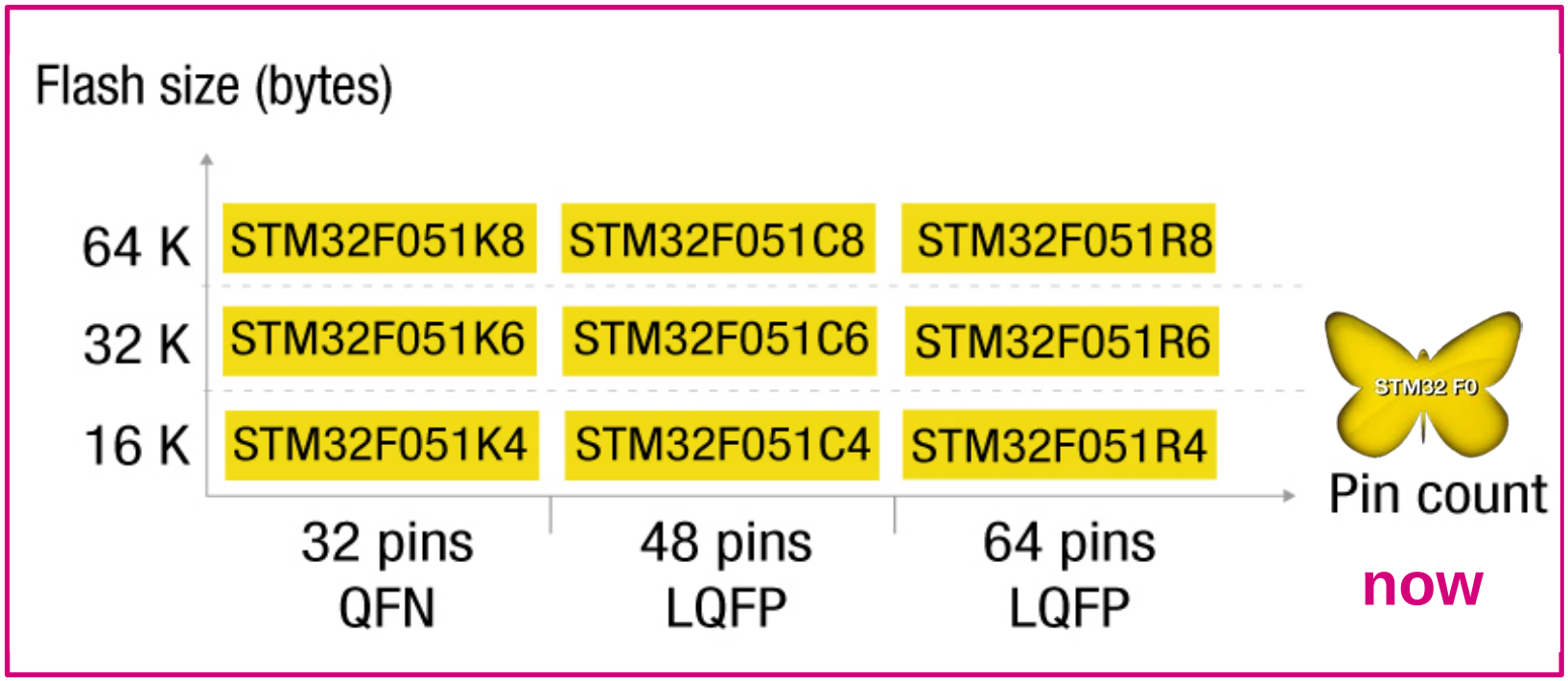
- Analog
  - 12-bit ADC with 1 MSPS
  - 12-bit DAC
  - 2x comparators
  - Separate supply for improved accuracy
- HDMI consumer electronics control (CEC)
- Touch-sensing
  - Up to18 keys
  - Key, slider and wheel
- Advanced timers
  - 32-bit and 16-bit PWM timers with 17 capture/compare input/outputs mapped on up to 28 pins
- Motor control
  - Permanent magnet synchronous motors (PMSM)

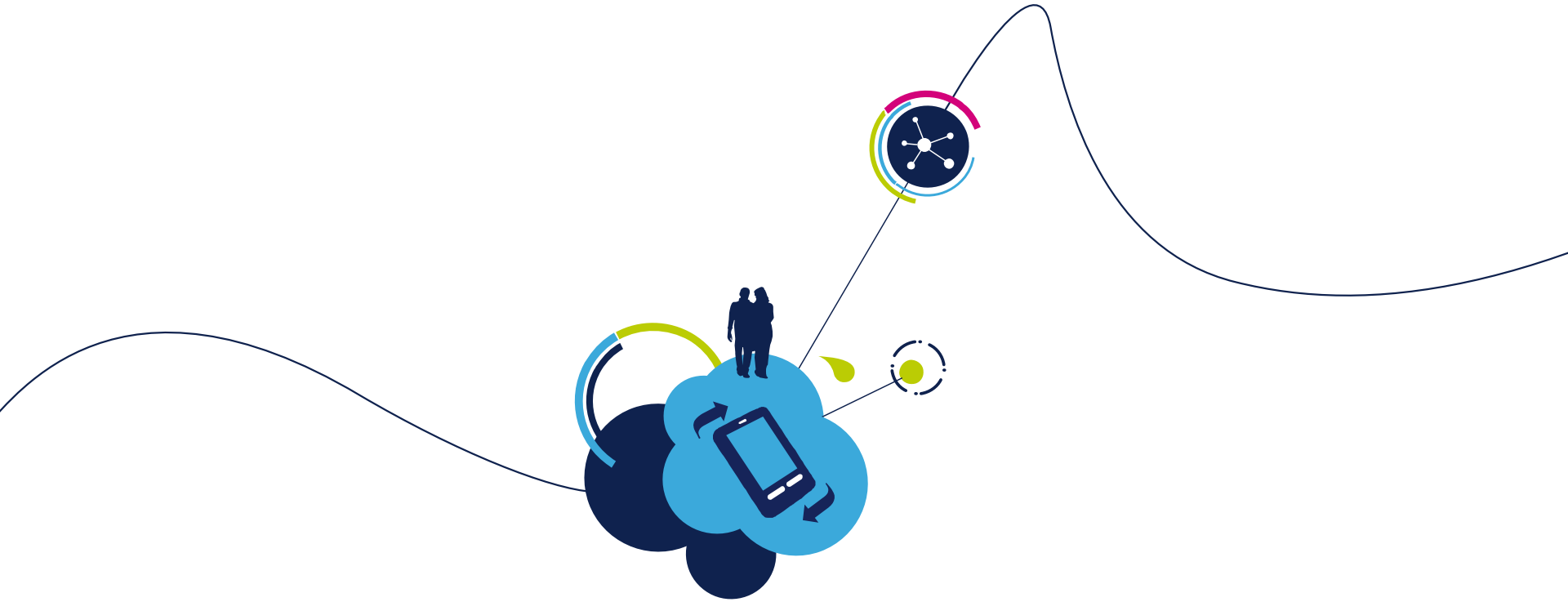


Maximum integration

- Meets industry safety specifications
  - **Class B-ready** for appliances
  - Hardware **RAM** parity check
  - **Clock security system** (CSS) to switch to internal backup RC in case of external clock failure
  - **2x watchdogs** capable of real-time code execution monitoring and ensuring the application integrity independently from system clock
  - **Cyclic redundancy check** (CRC) with DMA support for embedded Flash-memory content-integrity checking

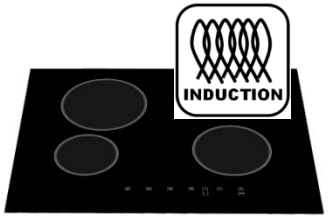
STM32 F0 portfolio





Great Fit for Applications

Great fit for appliances



Easy communication between front panel and power components with robust **I²C FM+ with 20 mA sink** capability and **fast IO toggling** capability (25% faster than STM32 F1 at same frequency)

Advanced digital and analog IPs

- 3 timers suit **induction cooking apps**
- 1 timer for **motor control** (complete reference designs available)
- 1 µs, 12-bit ADC with 12 channels for **efficient sensors**

Safety ready: optimized self-test routines for **EN/IEC 60335-1 Class B**
Advanced system and peripheral set

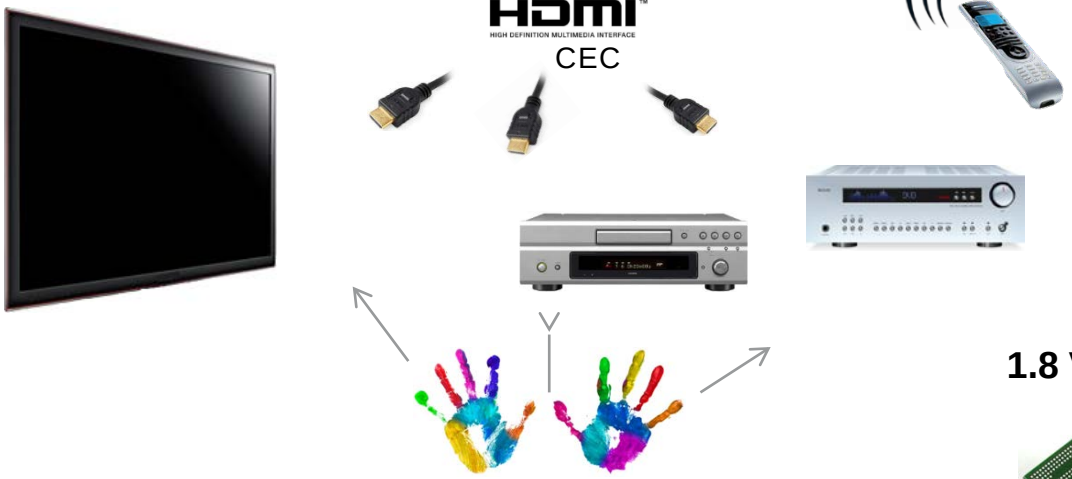
- Real-time hardware **RAM parity check** and 16-bit **CRC** for Flash-memory integrity checks
- Extended **double watchdog system** with autonomous clock, windowing and **clock security system**

life.augmented

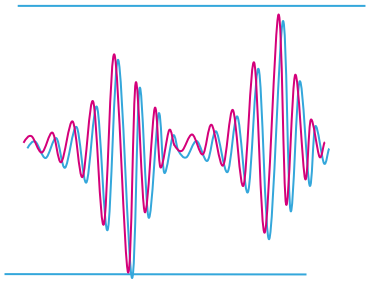
Great fit for consumer electronics

Optimized communication:

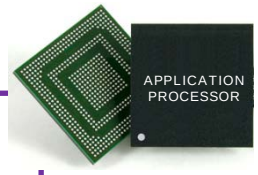
- **CEC** with dual clock domain allows flexible wake-up and synchronization
- Infrared **remote-control** decoder/encoder firmware libraries with optimum hardware implementation



3.6 V

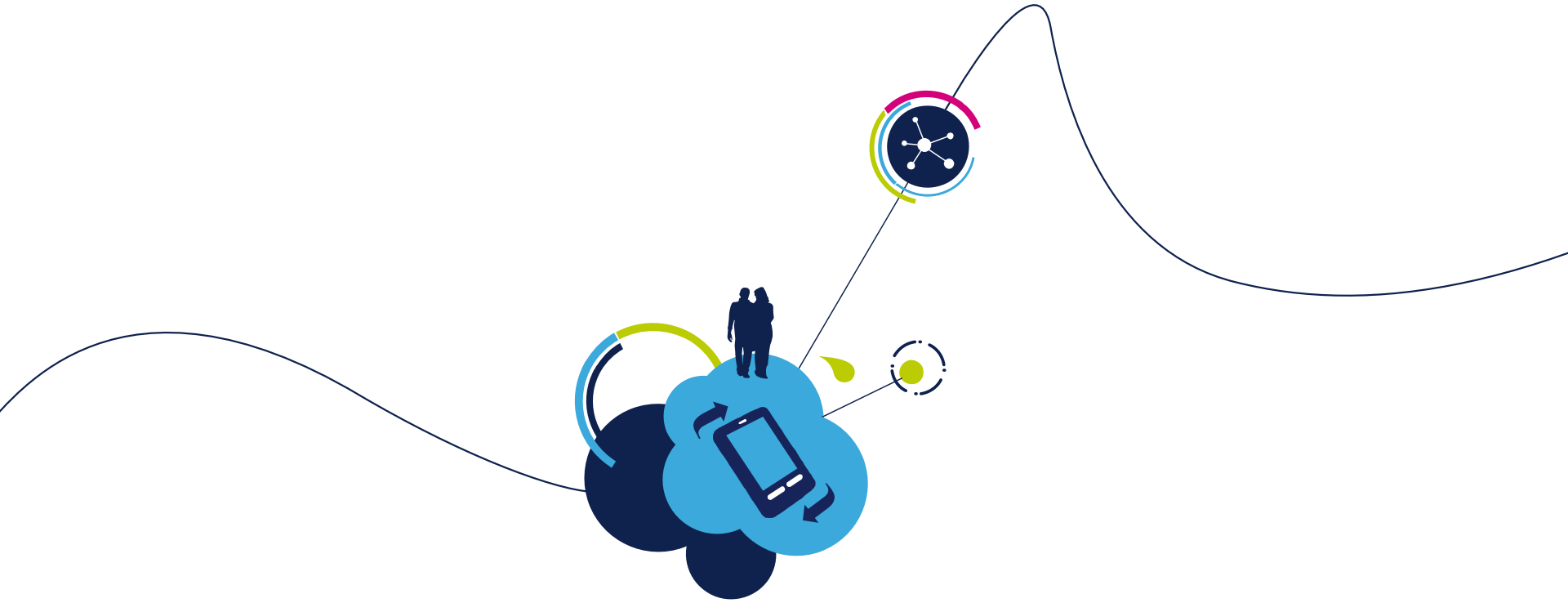


1.8 V



Easy interface with 1.8 V ICs (i.e. application processors) Separate power supply domains allows for a wider dynamic range for the ADC, DAC and Comparator (up to **3.6 V)** when Vdd is **1.8V.**

Capacitive touch sensing: Touch-controller IP allows zero CPU load with charge transfer method Supporting up to **18 keys and slider/wheel** capability

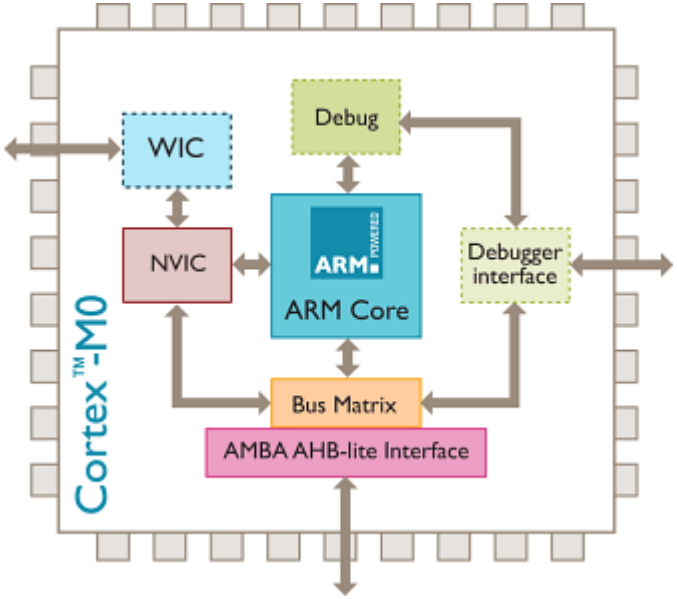


Introducing the ARM Cortex-M0

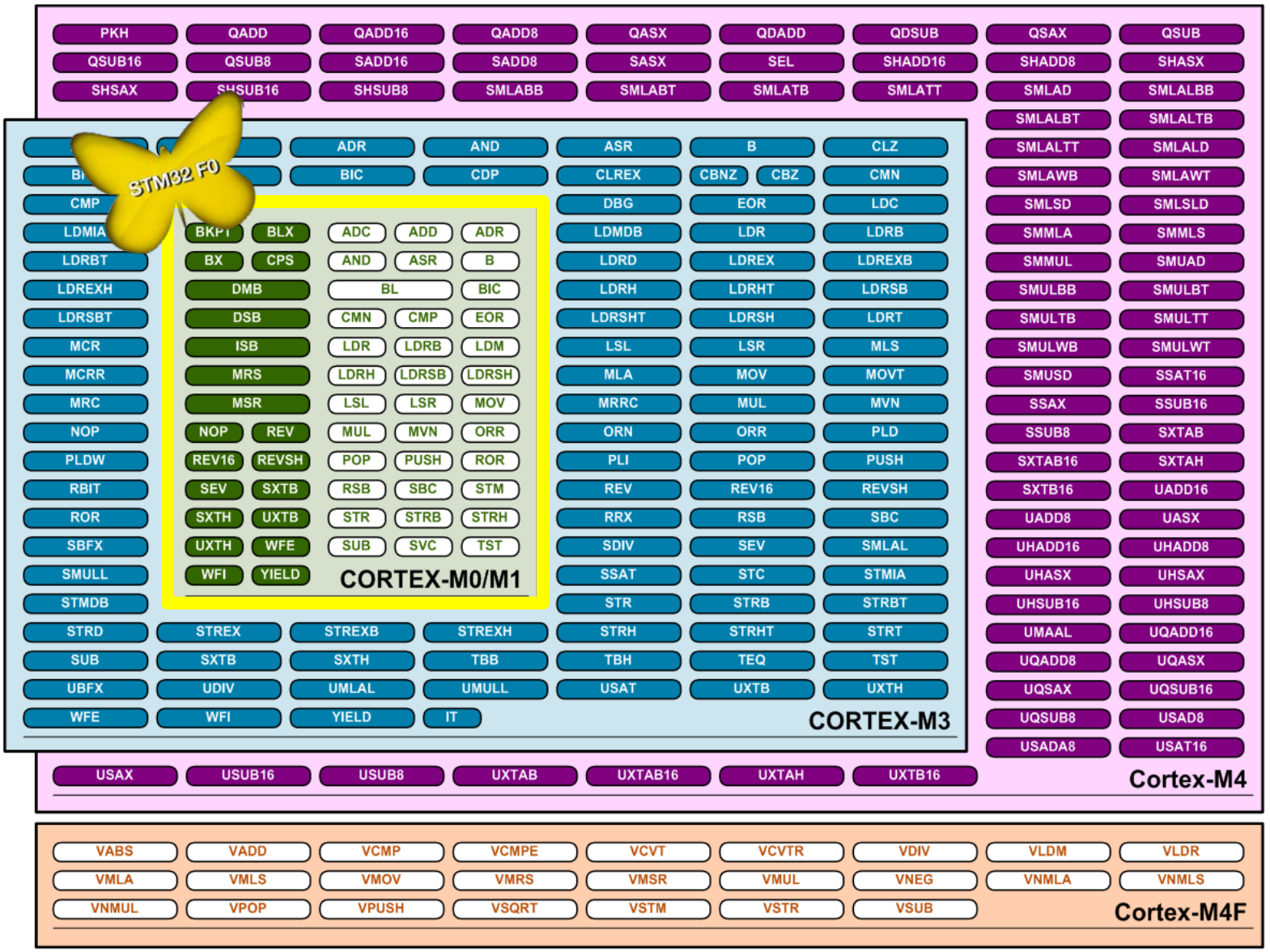
life.augmented

Cortex-M0 processor architecture

- ARMv6M Architecture
  - Thumb-2 Technology
  - Integrated configurable NVIC
  - Compatible with Cortex-M3/M4
- Microarchitecture
  - 3-stage pipeline
  - 1x AHB-Lite Bus Interfaces
- Configurable for ultra low power
  - Deep Sleep Mode, Opt. Wakeup Interrupt Controller
- Flexible configurations for wider applicability
  - Configurable Interrupt Controller (1-32 Interrupts and Priorities)
  - No Memory Protection Unit
  - Optional Debug & Trace



Cortex-M processors binary compatible



Cortex-M feature set comparison

	Cortex-M0	Cortex-M3	Cortex-M4
Architecture Version	**V6M**	v7M	v7ME
Instruction set architecture	**Thumb, Thumb-2 System Instructions**	Thumb + Thumb-2	Thumb + Thumb-2, DSP, SIMD, FP
DMIPS/MHz	**0.9**	1.25	1.25
Bus interfaces	**1**	3	3
Integrated NVIC	Yes	Yes	Yes
Number interrupts available	**1-32 + NMI**	1-240 + NMI	1-240 + NMI
Interrupt priorities available	**4**	8-256	8-256
Breakpoints, Watch points	**4/2/0, 2/1/0**	8/4/0, 2/1/0	8/4/0, 2/1/0
Memory Protection Unit (MPU)	**No**	Yes (Option)	Yes (Option)
Integrated trace option (ETM)	**No**	Yes (Option)	Yes (Option)
Fault Robust Interface	**No**	Yes (Option)	No
Single Cycle Multiply	Yes (Option)	Yes	Yes
Hardware Divide	**No**	Yes	Yes
WIC Support	Yes (Option)	Yes	Yes
Bit banding support	**No**	Yes	Yes
Single cycle DSP/SIMD	**No**	No	Yes
Floating point hardware	**No**	No	Yes (Option)
Bus protocol	**AHB Lite**	AHB Lite, APB	AHB Lite, APB
CMSIS Support	Yes	Yes	Yes

Cortex-M – firmware compatibility

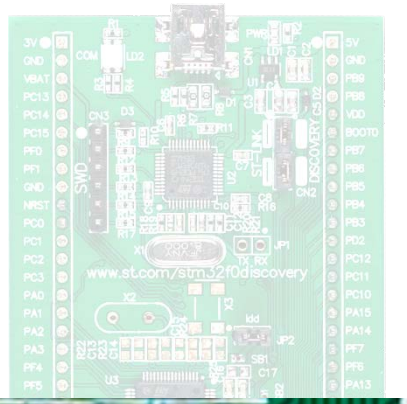
- Cortex M processors are FW and binary compatible
  - Re-compilation of the code is recommended between cores
  - When moving from M0/M3 to M4, some parts of the firmware may be re-coded to take advantage of the advanced DSP/SIMD instructions
  - When moving backwards (from M3/M4 -> M0), the code must be recompiled to only use M0 instructions
  - For a given STM32 family, the compatibility between the full peripheral set provides a simple migration path
- Code density is equivalent on the different Cortex-M implementations
  - Code sizes are within a few percent of each other in typical cases when the same compiler optimizations are used

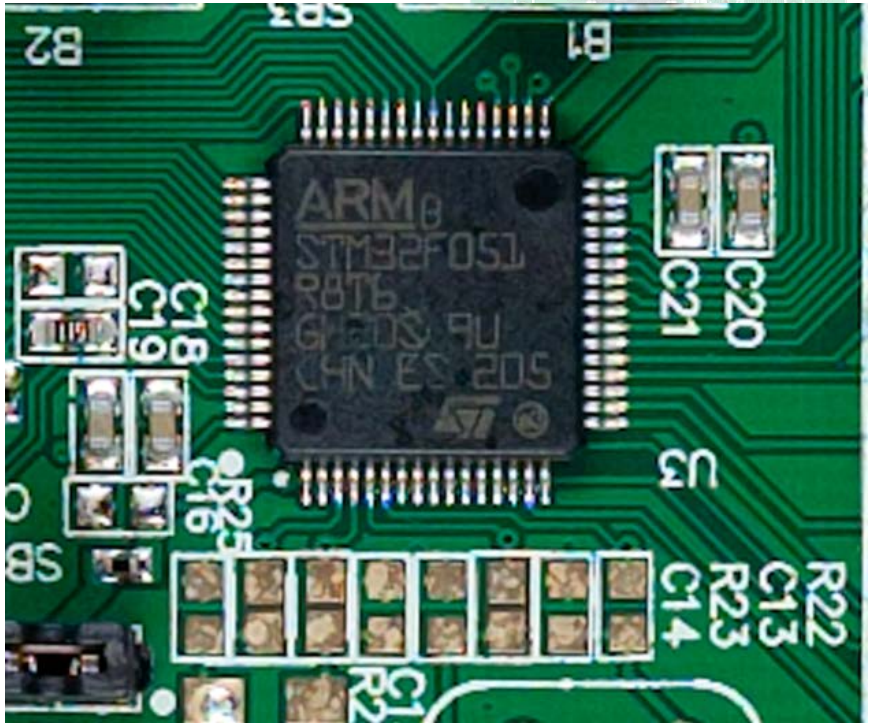
STM32F0-DISCOVERY

life.augmented

STM32F051R8T6

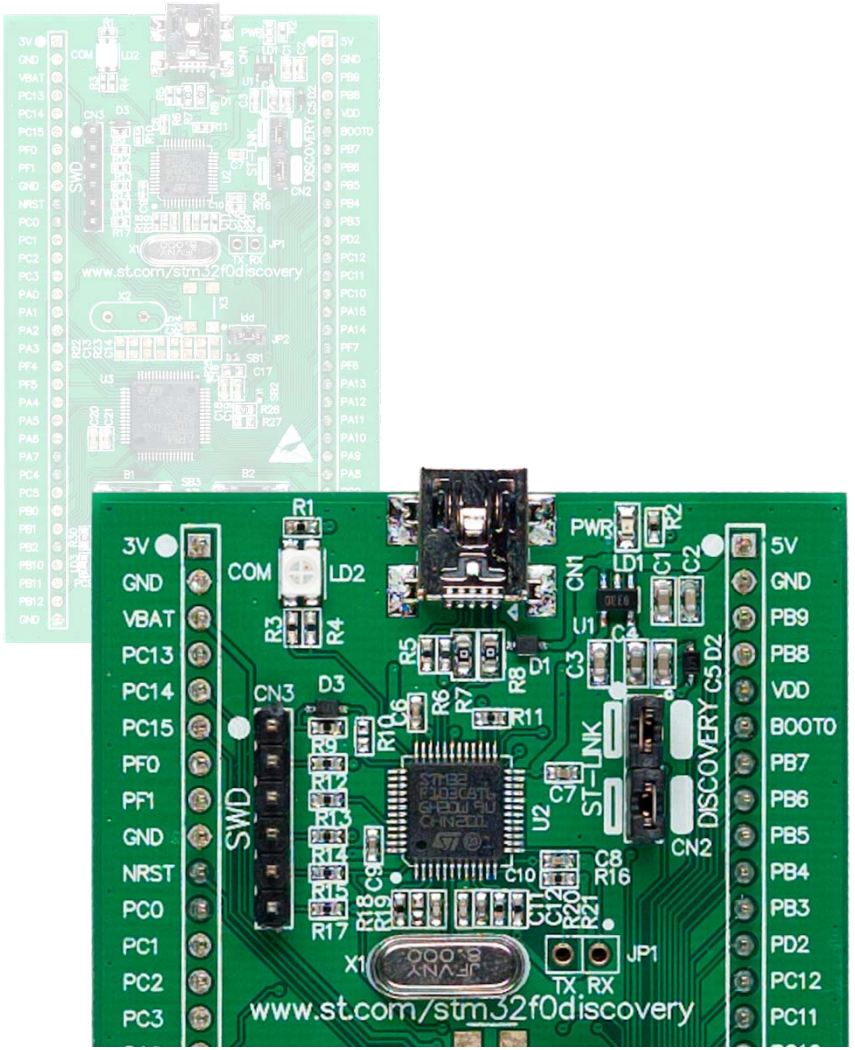
- 48 MHz Cortex-M0
- 64-pin LQFP
- 64 Kbytes Flash/8 Kbytes RAM





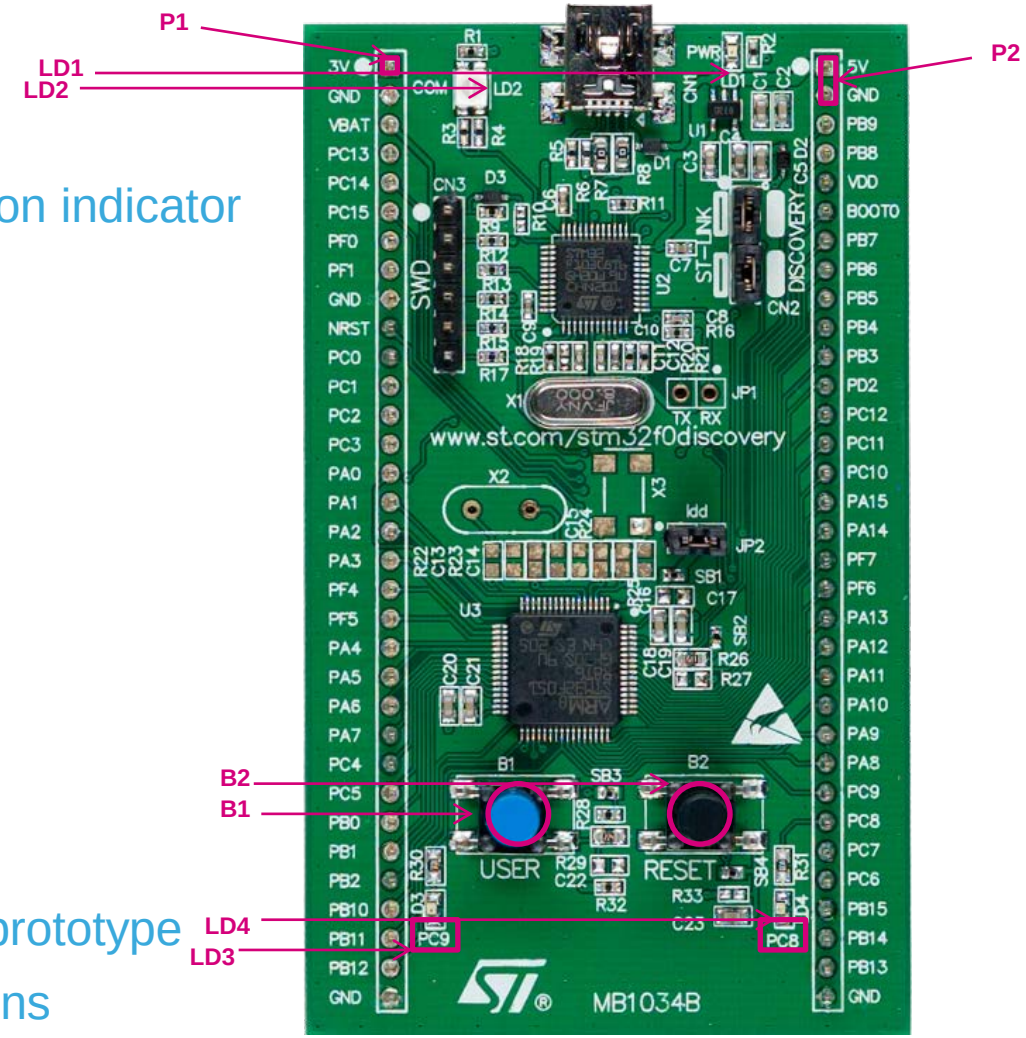
Embedded ST-Link

- ST-Link programming and debugging tool integrated on-board the kit (STM32F103C8T6)
- Can be used two different ways
  - Program and debug the MCU on the board
  - Program an MCU on another application board
- Features
  - USB Connector
  - ST-LINK MCU
  - 5V to 3V Voltage regulator
  - CN2 – MCU Program Jumper
  - CN3 – Application SWD connector



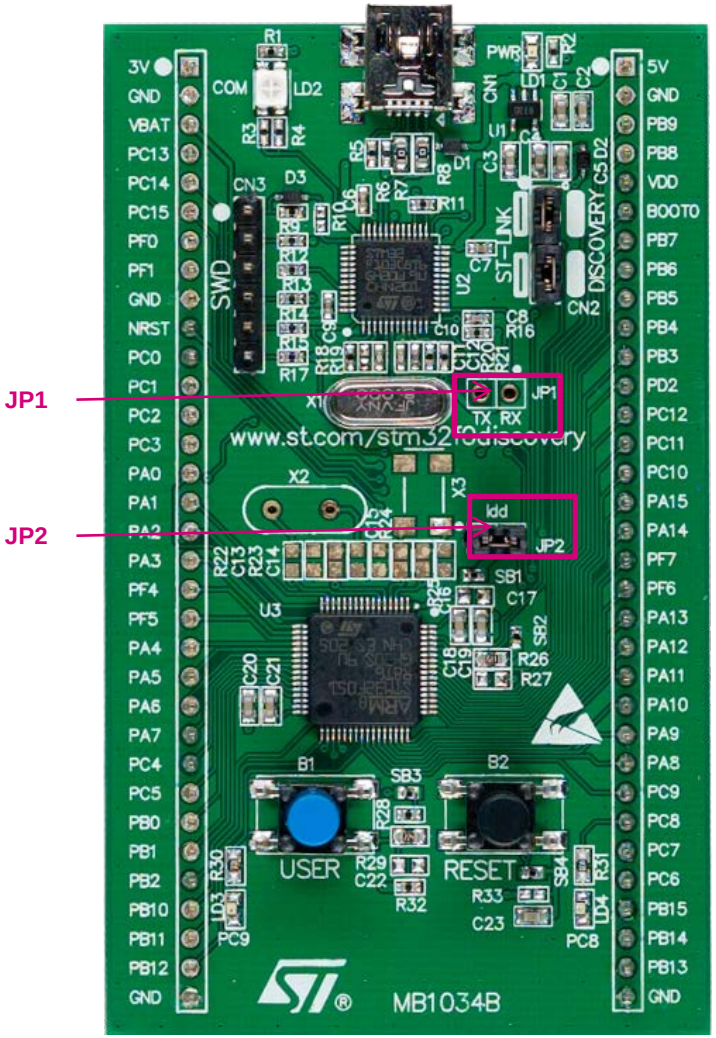
LEDs/Push-Buttons/Extension Connector

- LEDS
  - LD1: Power indicator
  - LD2: ST-LINK Communication indicator
  - LD3: User LED (PC9)
  - LD4: User LED (PC8)
- Push-Buttons
  - B1: User/Wake-up (PA0)
  - B2: Reset (NRST)
- Extension Connector
  - P1 and P2
  - All GPIOs are available for prototype
  - Includes 5V, 3V and GND pins



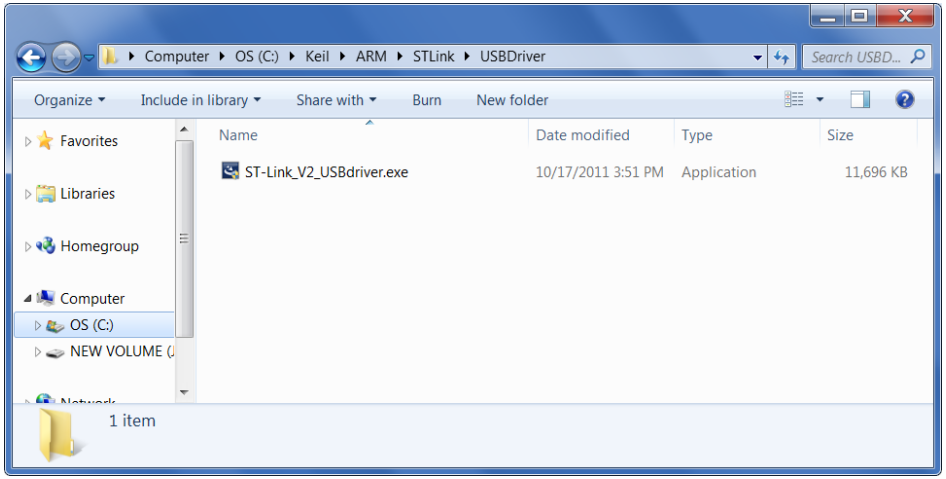
Jumpers/User Manual/Firmware Library

- Jumpers
  - JP1: USART1 TX and RX (not fitted, reserved function)
  - JP2: I_{DD} for MCU current measurement (fitted by default)
- Documentation
  - UM1523 STM32F0DISCOVERY Kit
- Firmware Library
  - Contains STM32F0 Standard Firmware Library
  - Contains example code
  - AN4062 peripheral firmware examples



Step #3 - Install ST-Link Driver

- The STM32F0DISCOVERY board includes an ST-LINK/V2 embedded programming and debug tool
- The driver for ST-Link is contained in the Keil uVision toolchain and located in this directory:
  - C:\Keil\ARM\STLink\USBDriver
- Double-click on the file: ST-Link_V2_USBDriver.exe to install
- Click through the installation menu until the driver installation is complete

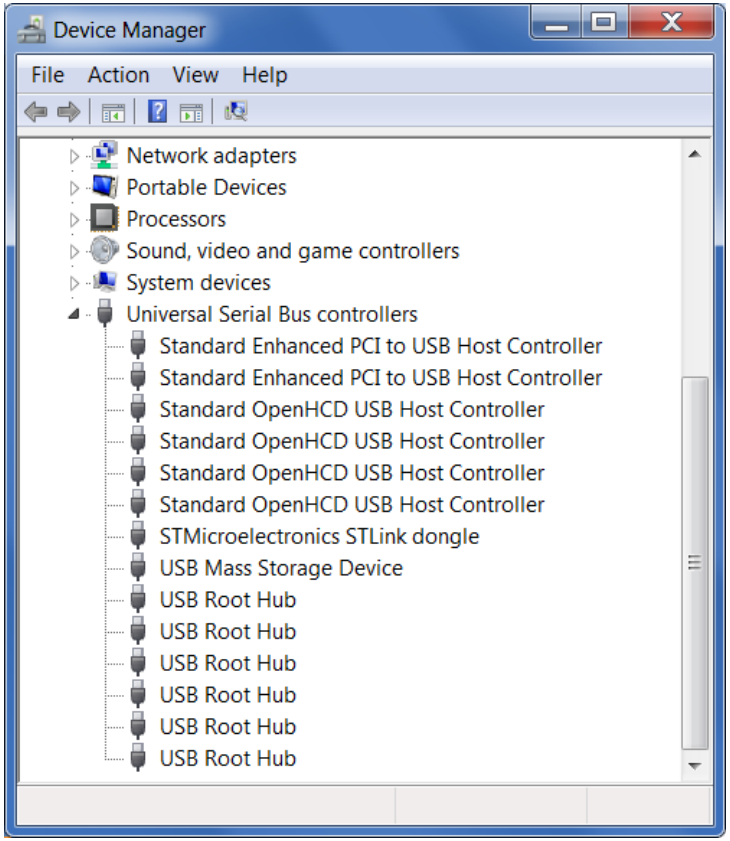


Step #4: Connect the DiscoveryKit/ Enable ST-Link

- Using the USB cable, connect the mini-B male connector into the STM32F0DISCOVERY USB port and connect the A male connector into your Laptop

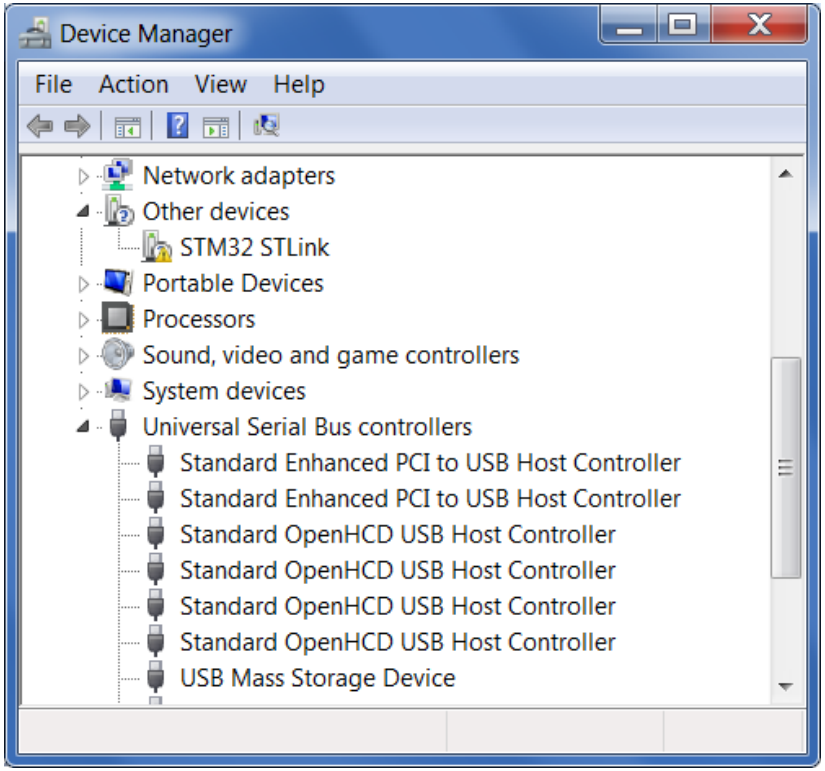
- Wait for Windows to recognize the ST-Link device and follow any step required to install the driver

- Upon successful driver recognition, the ST-Link device should be fully enumerated in Windows Device Manager as show:

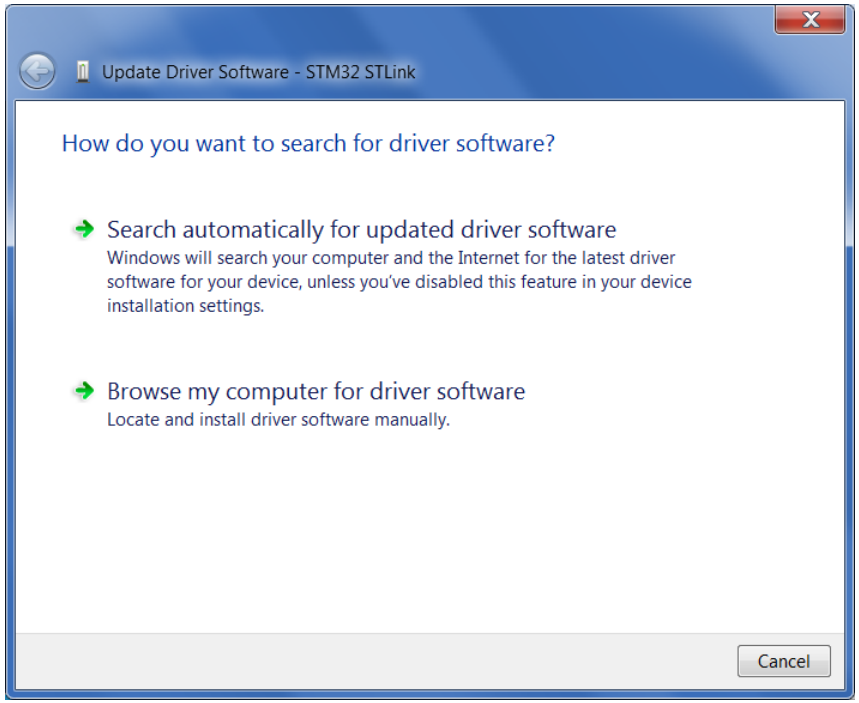


Step #4
ST-Link Driver Trouble Shooting

1. Open Device Manager
2. Right-click on the STM32 ST-Link Driver icon
3. Select “Update Driver Software”



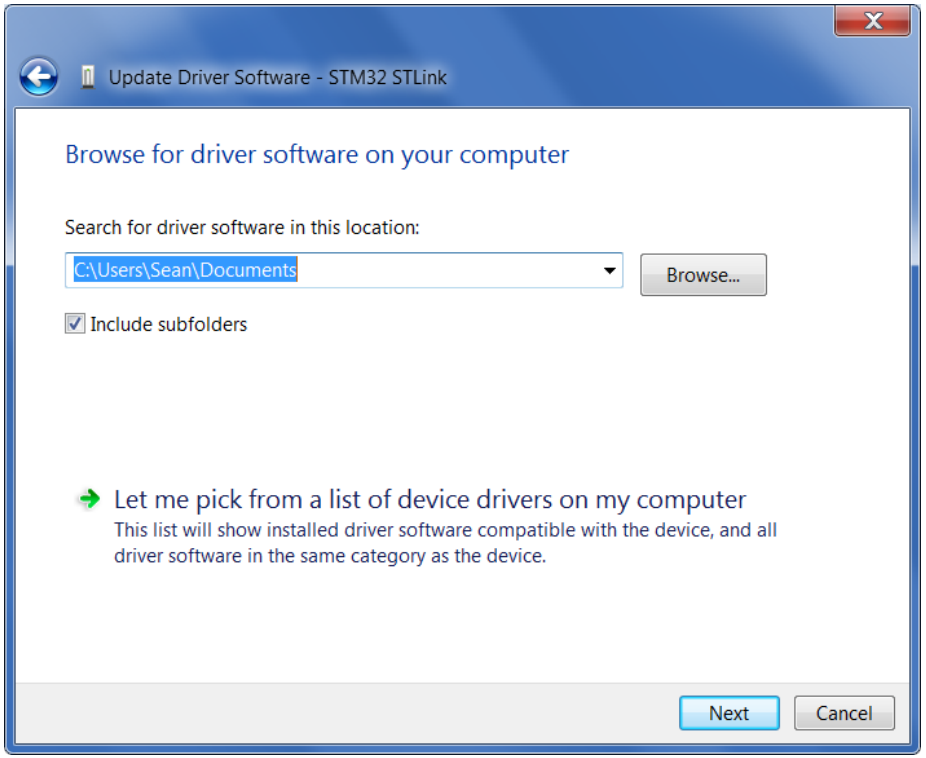
Step #4
ST-Link Driver Trouble Shooting



5. Select “Let me pick from a list of device drivers of my computer”

6. Click “Next”

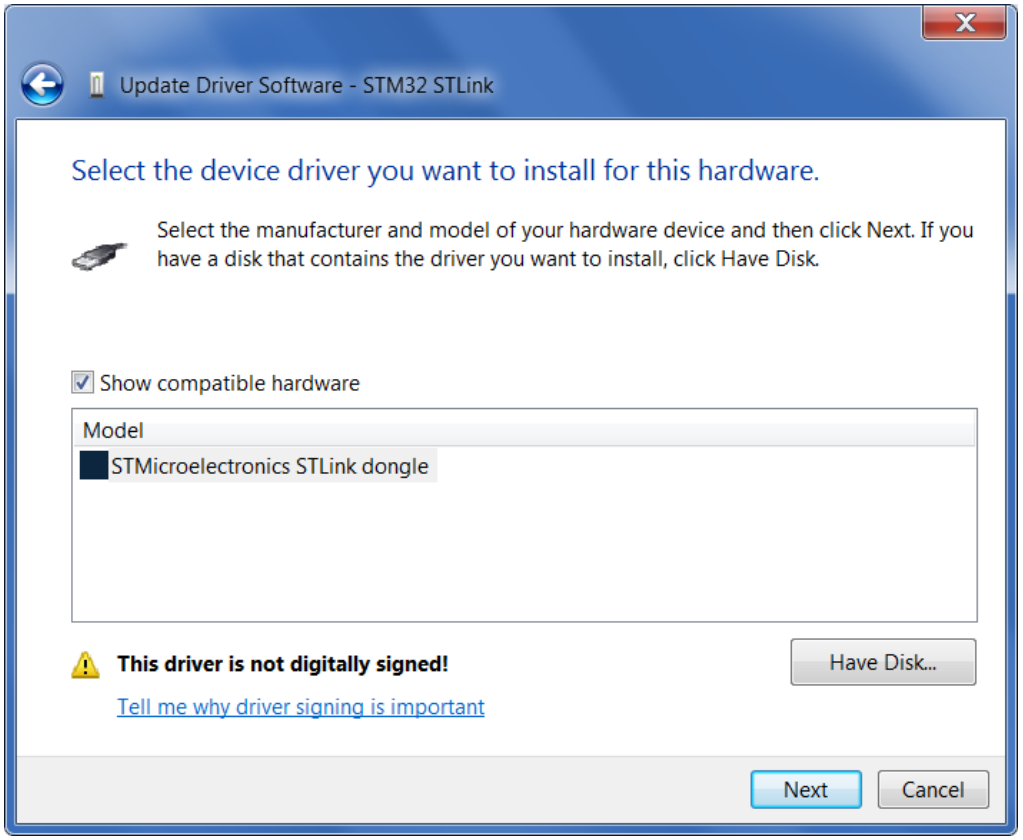
4. Select “Browse my computer for driver software”



Step #4
ST-Link Driver Trouble Shooting

- The “STMicroelectronics ST-Link dongle” should be listed

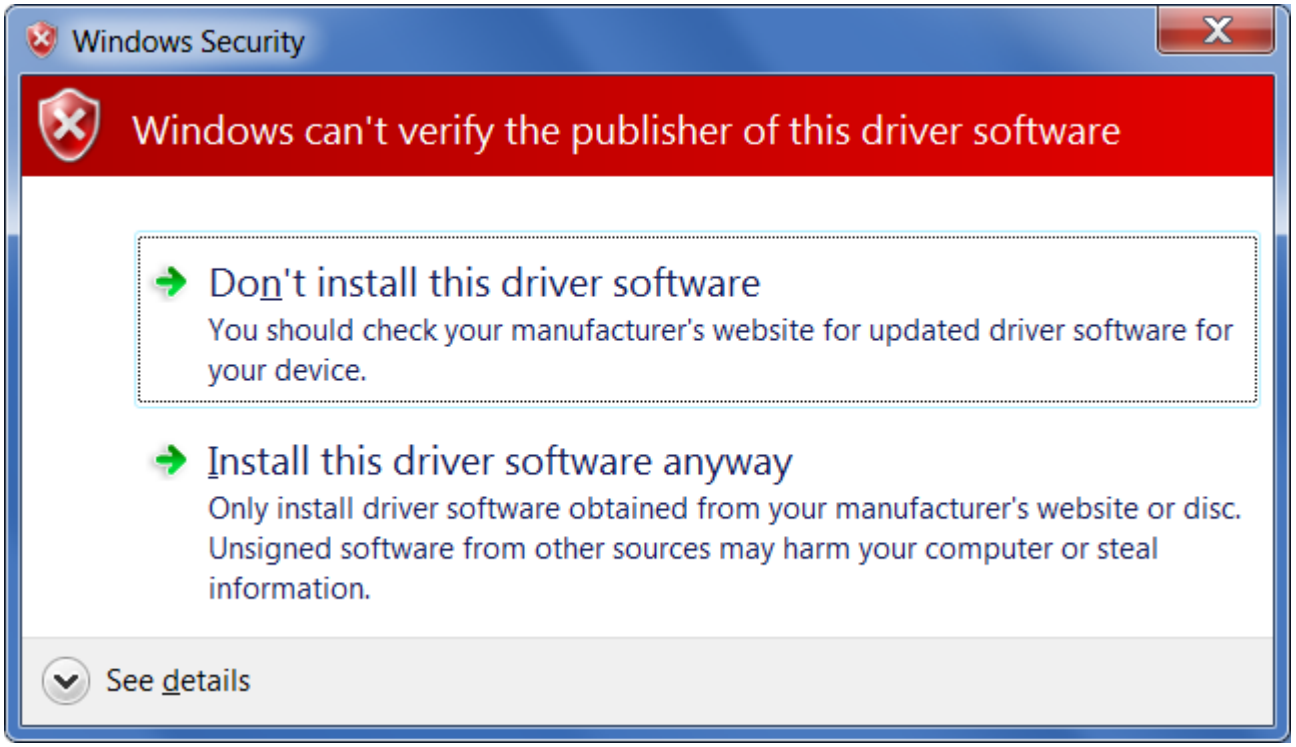
7. Click “Next”



Step #4
ST-Link Driver Trouble Shooting

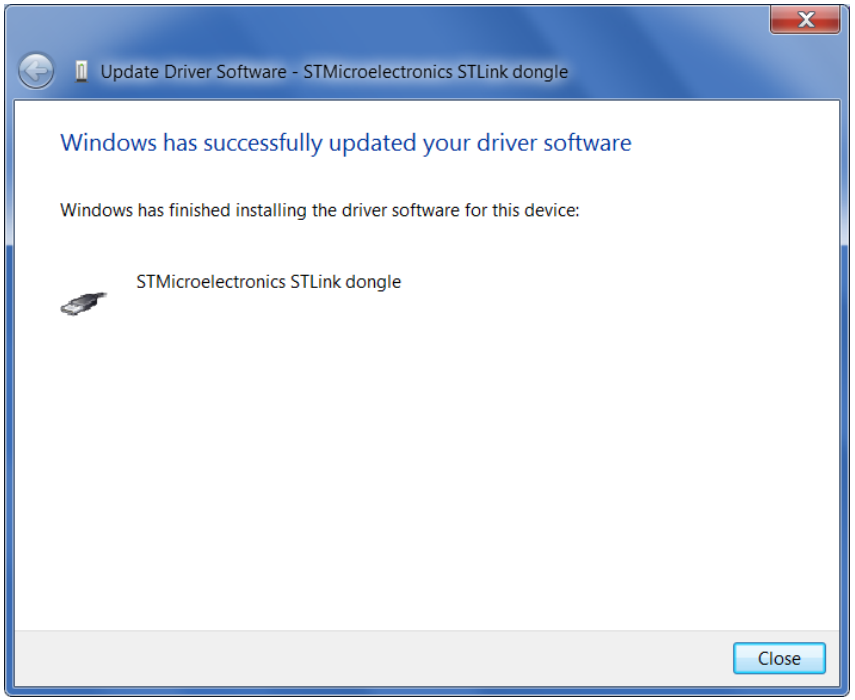
- A warning message may appear

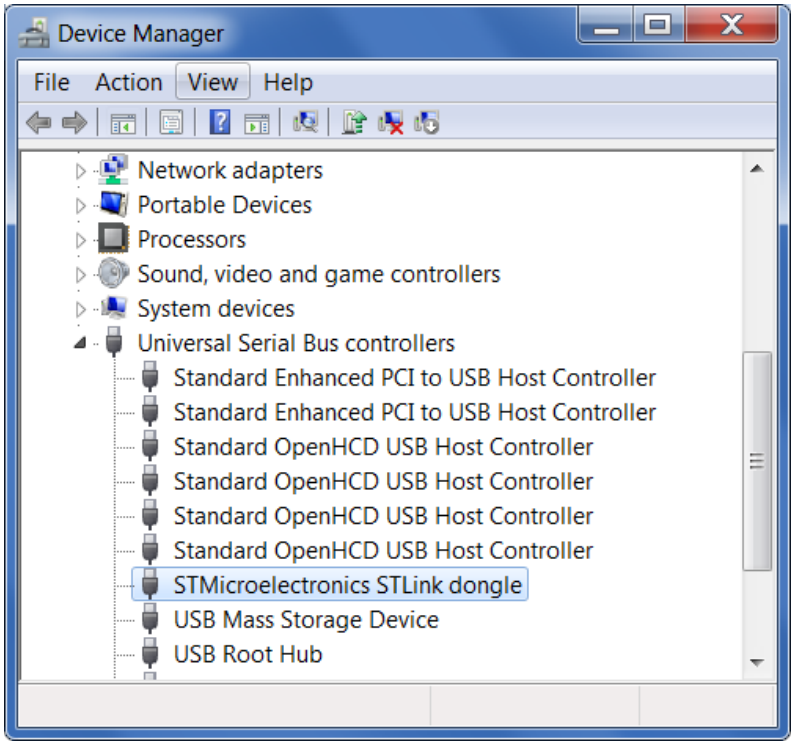
8. Select “Install this driver software anyway”



Step #4
ST-Link Driver Trouble Shooting

- You should receive a message: “Windows has successfully updated your driver software”





- Re-check device manager to ensure STMicroelectronics ST-Link dongle is functioning normally

STM32 F0 Tools
Documentation overview

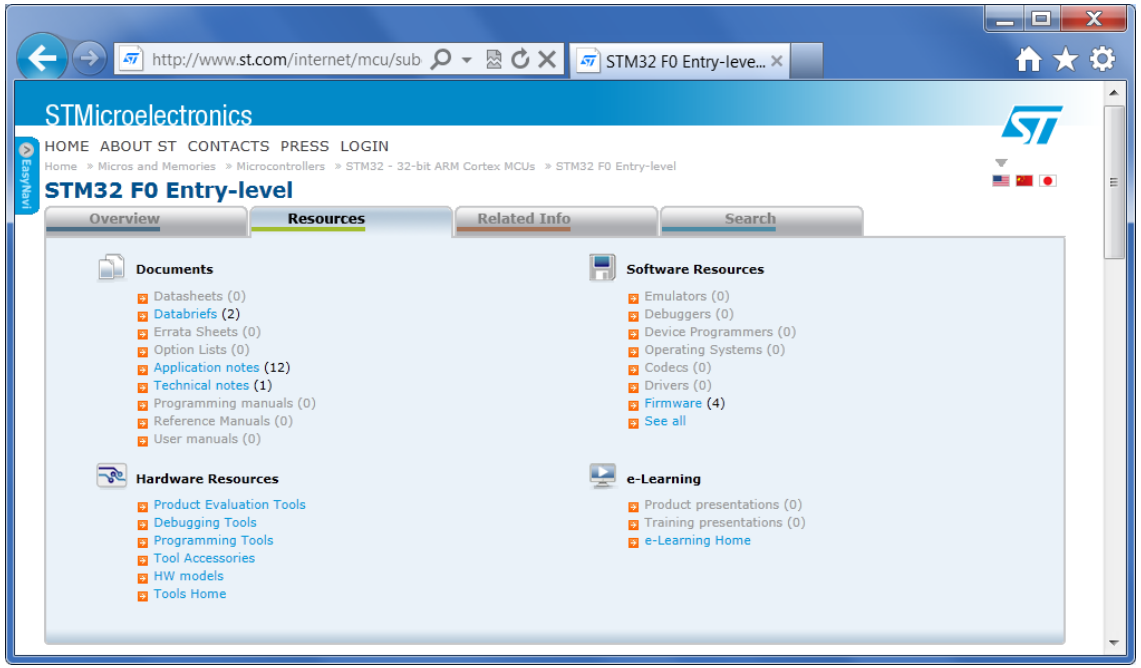
life.augmented

Documentation resources

- Discovery Kit related documentation can be found @ www.st.com/stm32f0discovery under the “Design support” tab
- In the directory:
  - C:\STM32F0-Discovery_Kit\Documentation\
- You will find:
  - STM32F051x Datasheet
  - STM32F051x Errata (ES0202)
  - STM32F051x Reference Manual (RM0091)
  - STM32F0xxx Cortex-M0 programming manual (PM0215)
  - STM32F0DISCOVERY peripheral firmware examples (AN4062)
  - Getting started with software and firmware environments for the STM32F0DISCOVERY kit (UM1523)
  - STM32F0DISCOVERY kit user manual (UM1525)

Documentation resources

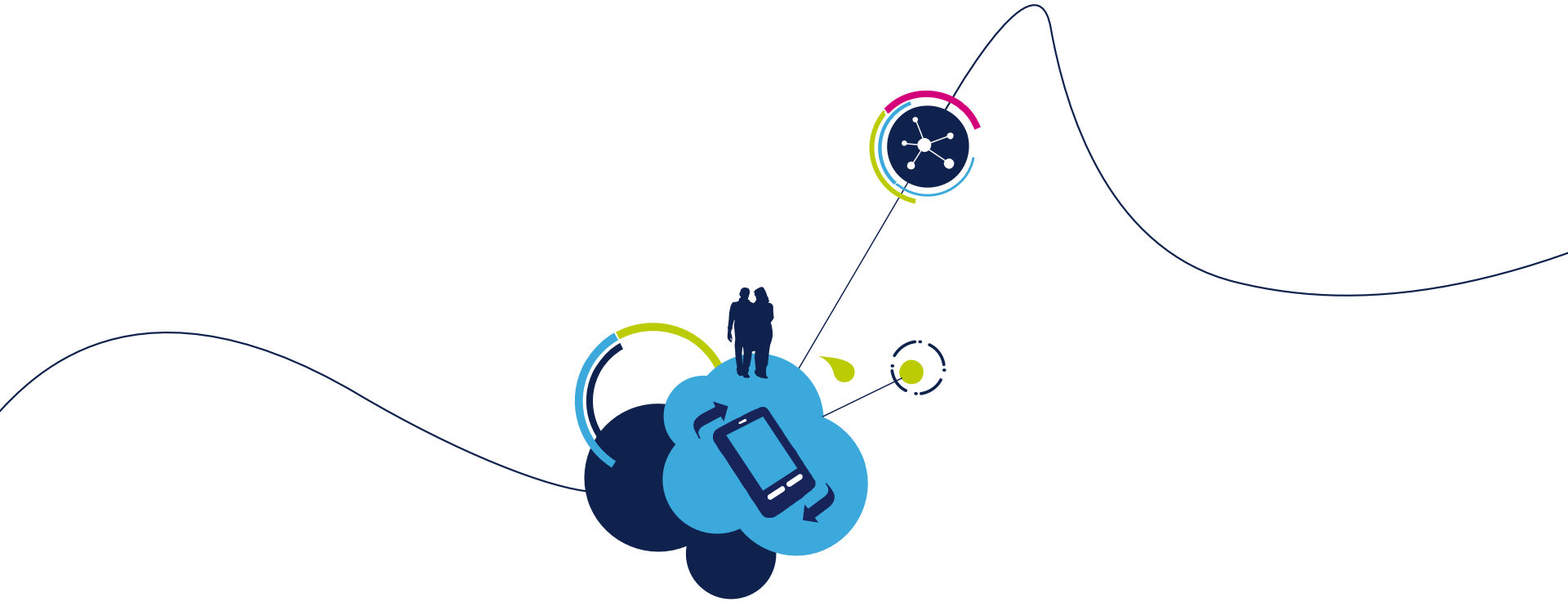
- Main website page for the STM32 F0 Series
  - www.st.com/stm32f0
- You can find
  - Datasheets
  - Applications Notes
  - Errata
  - Technical Notes
  - Programming Manuals
  - Reference Manual
  - User Manuals
  - Firmware



- For all STM32 related products: www.st.com/stm32

Support resources

- Technically trained distributors
  - Distributors listed on CONTACTS page, www.st.com/contactus
- ST Public Forums:
  - Located on main www.st.com page under Support tab – ST e2e Communities
- Submit technical questions to ST Online Support:
  - Located on main www.st.com page under the Support tab – Online Support



Compile, Debug and Run

First, a process check…

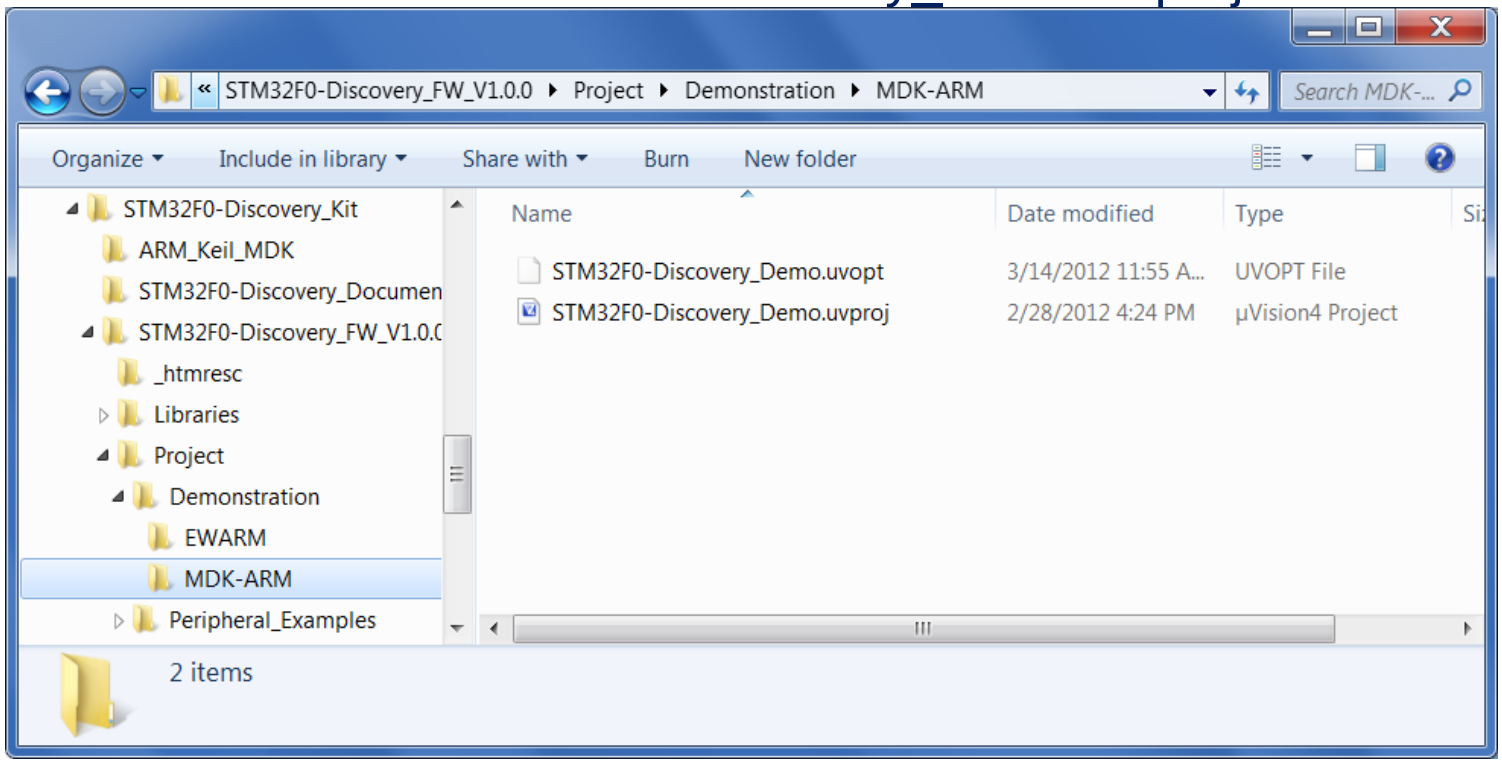
- ST-Link is recognized by your system
- LD1 and LD2 should be ON (indicating the board is powered and the ST-Link is functional)
- LD3 (Green) should be flashing
- A brief test of the board
  - Press the USER Button
    - LD4 (Blue) should flash once
    - LD3 (Green) will blink rapidly
  - Press the USER Button again
    - LD4 (Blue) should flash once
    - LD3 (Green) will shut off

Step #5
Open FW demo project with Keil uVision

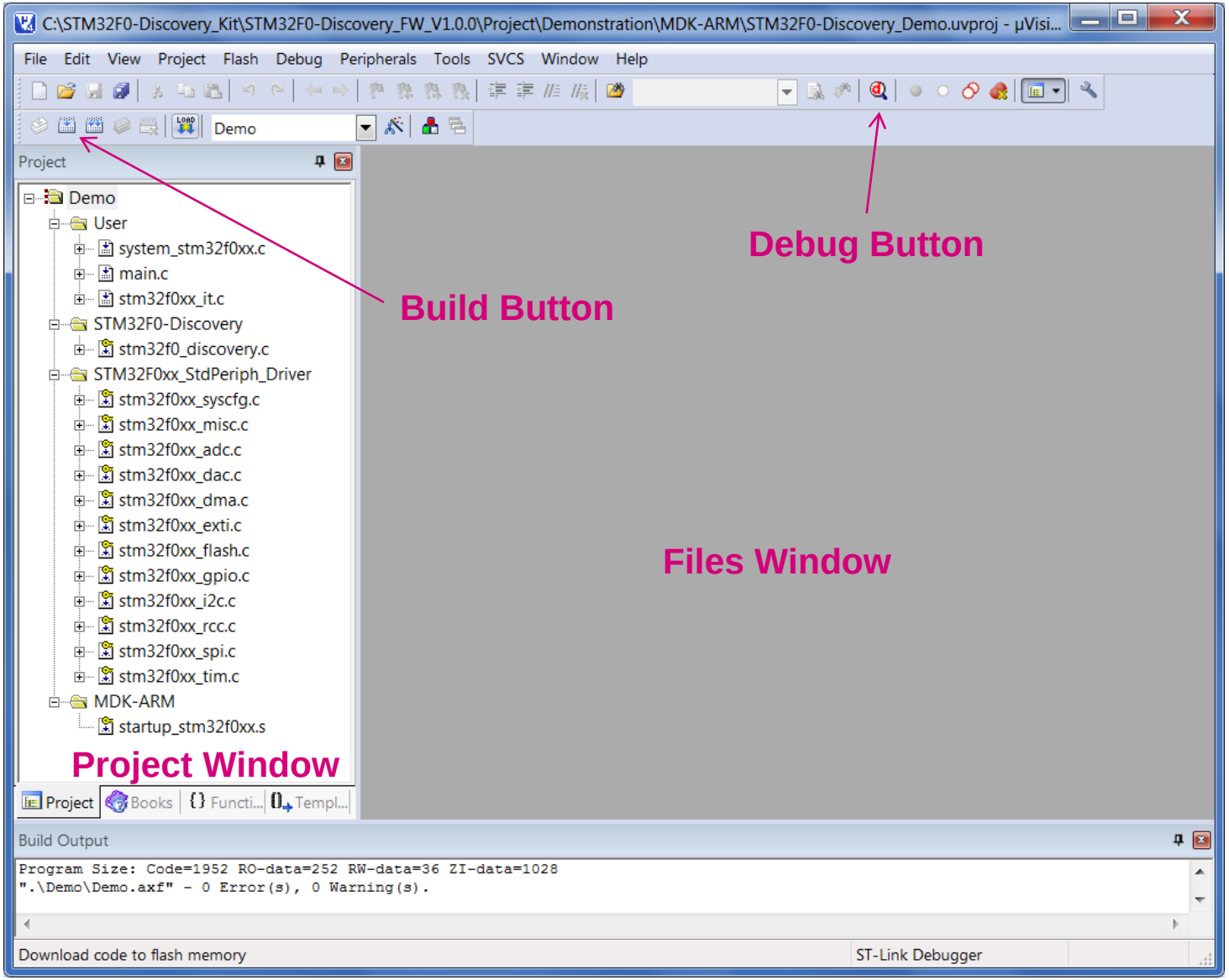
- Using explorer, go to the directory:

  C:\STM32F0-Discovery_Kit\STM32F0-Discovery_FW_V1.0.0\
  Project\Demonstration\MDK-ARM

- Double-click on the STM32F0-Discovery_Demo.uvproj file



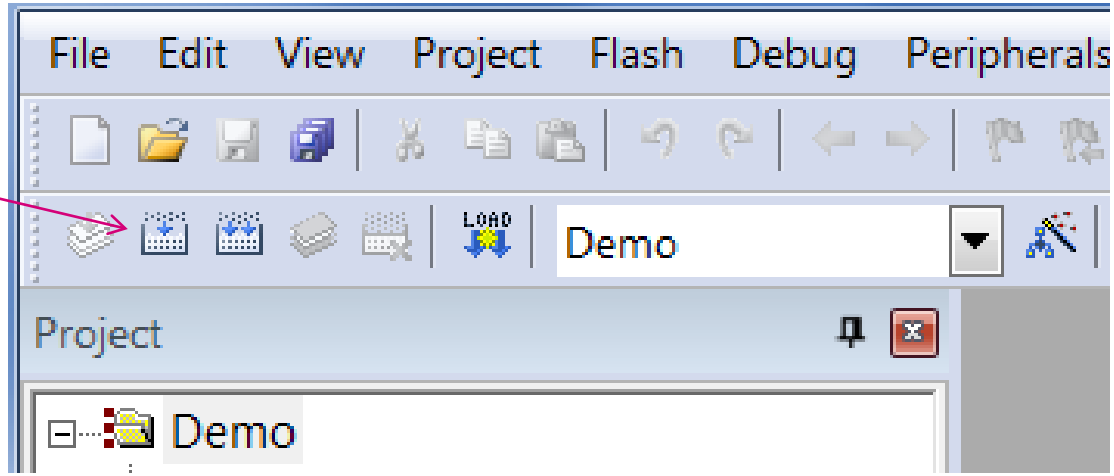
Step #5 - Inside Keil uVision



Step #6 - Compile

- Click on the Build button or Menu::Project::Build Target

Build Button



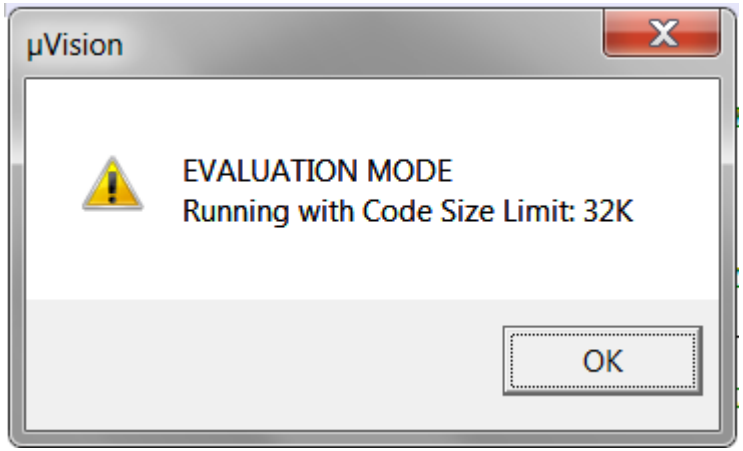
- The project should compile without errors

```
Build Output
compiling stm32f0xx_rcc.c...
compiling stm32f0xx_spi.c...
compiling stm32f0xx_tim.c...
assembling startup_stm32f0xx.s...
linking...
Program Size: Code=1952 RO-data=252 RW-data=36 ZI-data=1028
".\Demo\Demo.axf" - 0 Error(s), 0 Warning(s).
```

Step #7 - Debug

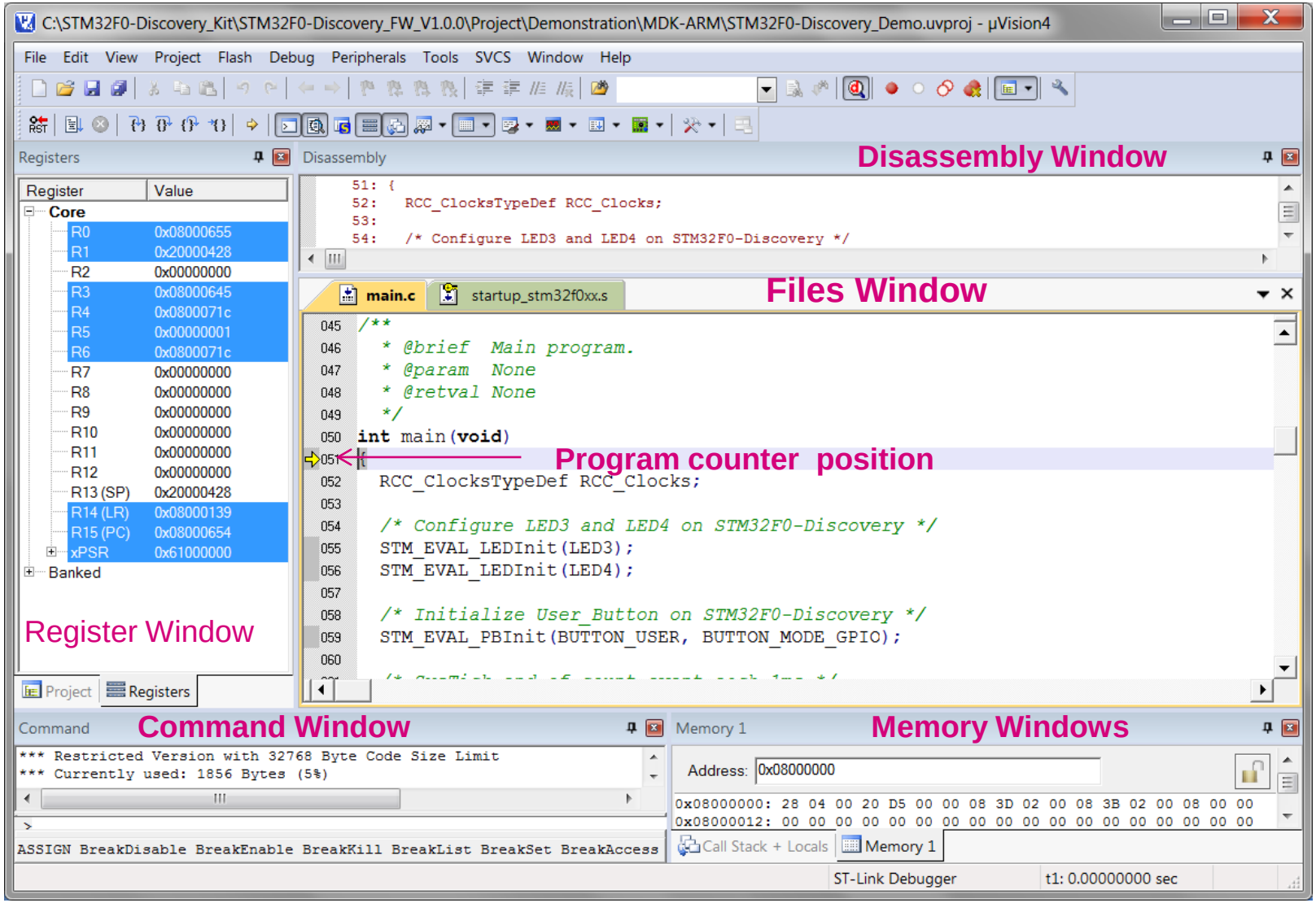
- Click on the Start/Stop Debug Session button or Menu: Start/Stop Debug Session

Debug Button



- You should receive a warning message. Click “OK”

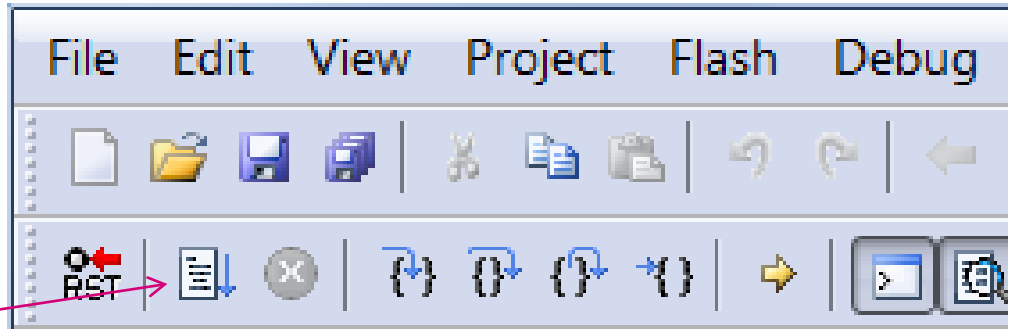
Step #7
The MDK-ARM IDE Debugger



Step #8 - Run

- Click on the Run button to start the program

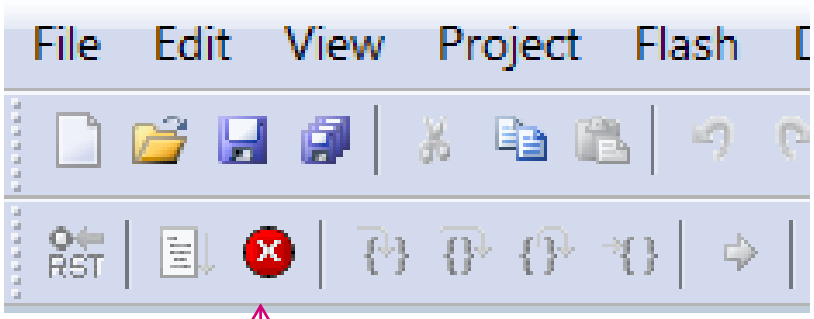
Run Button



- Your STM32F0DISCOVERY board LD3 should begin flashing
  - Note: LD2 (ST-Link Status) Should be flashing

Step #8 - Run

- Mission Accomplished
- Please click on the Stop button.
- You code will stop anywhere within the program flow
- Click on the Debug button to exit from the debugger



Stop Button

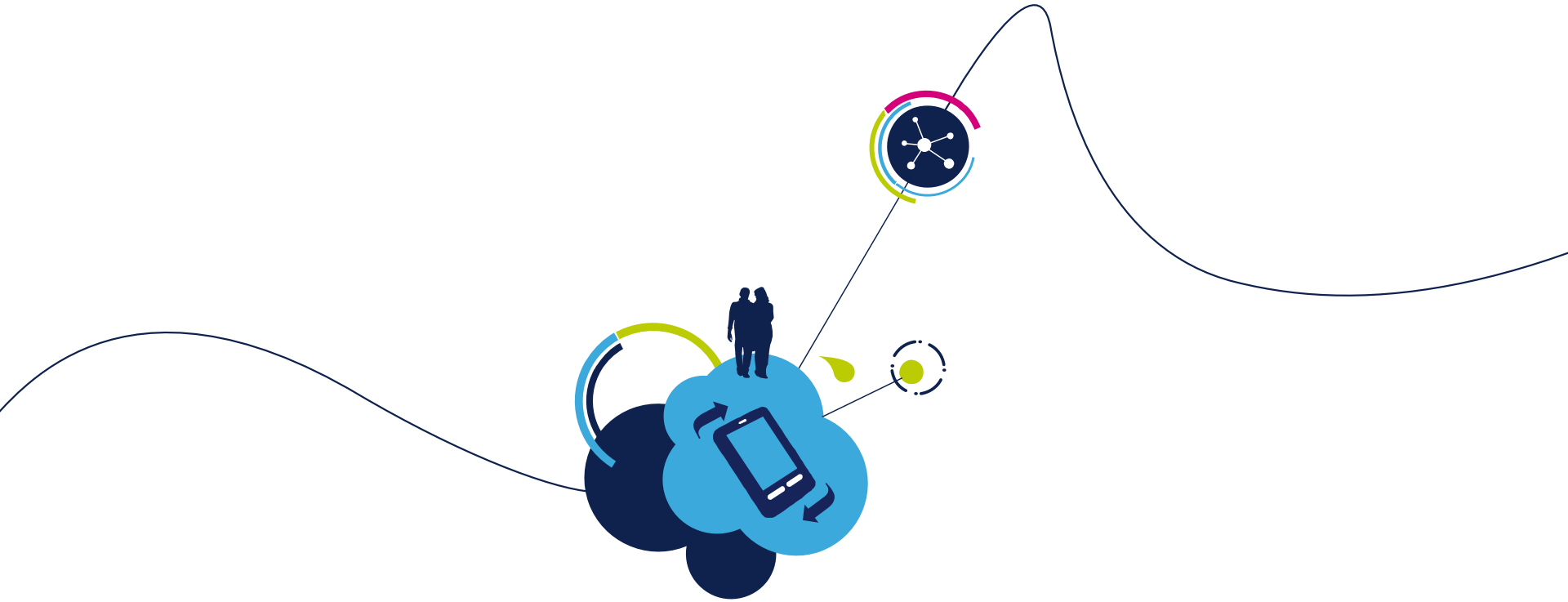
Debug Button

Let’s make a change

- Double-click to open the main.c file
- Scroll down to line 100

```
092
093     /* Test on blink speed */
094     if(BlinkSpeed == 2)
095     {
096       /* LED3 toggles each 100 ms */
097       STM_EVAL_LEDToggle(LED3);
098
099       /* maintain LED3 status for 100ms */
100       Delay(100); // Your number here [10,500]
101     }
102     else if(BlinkSpeed == 1)
103     {
```

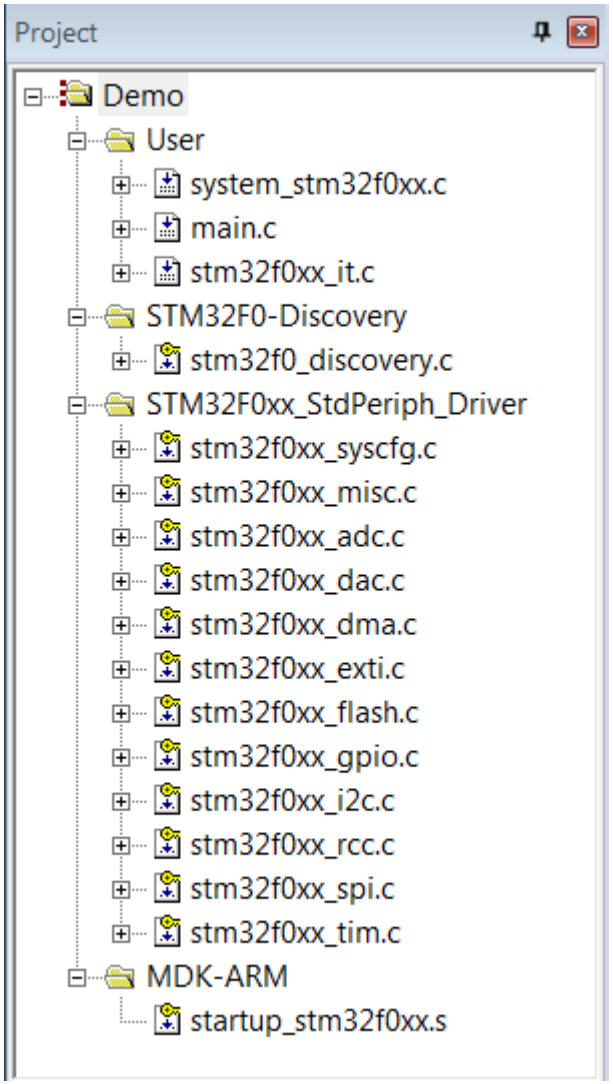
- Enter a number from 10 to 500 and place in the Delay(xxx) statement
  - Note: a larger number results in a slower blink rate after the first press of the button
- Compile, Debug, and Run
- Press the User button. Validate! Did it work?
- Stop debug and exit the debugger



Firmware Project Overview

Project Files

- User files
  - main.c (program entry point)
  - system_stm32f0xx.c (initial system configuration)
  - stm32f0xx_it.c (ISR's)
- stm32f0-discovery.c
  - Board specific functions
- STM32F0xx_StdPeriph_Driver
  - Contains peripheral library functions
- startup_stm32f0xx.s
  - System initialization, vector table, reset and branch to main()



startup_stm32f0xx.s

- Main Characteristics
  - Initializes stack pointer

```
Stack_Size      EQU     0x00000400

                AREA    STACK, NOINIT, READWRITE, ALIGN=3
Stack_Mem       SPACE   Stack_Size
__initial_sp
```

  - Contains the vector table for the part

```
; Vector Table Mapped to Address 0 at Reset
                AREA    RESET, DATA, READONLY
                EXPORT  __Vectors
                EXPORT  __Vectors_End
                EXPORT  __Vectors_Size

__Vectors       DCD     __initial_sp              ; Top of Stack
                DCD     Reset_Handler             ; Reset Handler
                DCD     NMI_Handler               ; NMI Handler
                DCD     HardFault_Handler         ; Hard Fault Handler
```

  - Contains Reset handler - called on system reset
    - Calls SystemInit() function
    - Branches to main()

```
; Reset handler routine
Reset_Handler    PROC
                 EXPORT  Reset_Handler                 [WEAK]
        IMPORT   __main
        IMPORT   SystemInit
                 LDR     R0, =SystemInit
                 BLX     R0
                 LDR     R0, =__main
                 BX      R0
                 ENDP
```

system_stm32f0xx.c

- SystemInit()
  - This function is called at startup just after reset and before branch to main program. This call is made inside the "startup_stm32f0xx.s" file.
  - Setups the system clock (System clock source, PLL Multiplier and Divider factors, AHB/APBx prescalers and Flash settings)

Define PLL source

```
103 #define PLL_SOURCE_HSI        // HSI (~8MHz) used to clock the PLL, and the PLL is used as system clock source
104 //#define PLL_SOURCE_HSE       // HSE (8MHz) used to clock the PLL, and the PLL is used as system clock source
105 //#define PLL_SOURCE_HSE_BYPASS // HSE bypassed with an external clock (8MHz, coming from ST-Link) used to clock
106                                 // the PLL, and the PLL is used as system clock source
```

SystemInit()

```
151 void SystemInit (void)
152 {
153   /* Set HSION bit */
154   RCC->CR |= (uint32_t)0x00000001;
155
156   /* Reset SW[1:0], HPRE[3:0], PPRE[2:0], ADCPRE and MCOSEL[2:0] bits */
157   RCC->CFGR &= (uint32_t)0xF8FFB80C;
```

.
.
.

Call SetSysClock()

```
180   /* Configure the System clock frequency, AHB/APBx prescalers and Flash settings */
181   SetSysClock();
```

Configure clock tree

```
271 static void SetSysClock(void)
272 {
273   __IO uint32_t StartUpCounter = 0, HSEStatus = 0;
274
275   /* SYSCLK, HCLK, PCLK configuration ----------------------------------------*/
276 #if defined (PLL_SOURCE_HSI)
277   /* At this stage the HSI is already enabled */
278
279   /* Enable Prefetch Buffer and set Flash Latency */
280   FLASH->ACR = FLASH_ACR_PRFTBE | FLASH_ACR_LATENCY;
281
282   /* HCLK = SYSCLK */
283   RCC->CFGR |= (uint32_t)RCC_CFGR_HPRE_DIV1;
284
285   /* PCLK = HCLK */
286   RCC->CFGR |= (uint32_t)RCC_CFGR_PPRE_DIV1;
287
288   /* PLL configuration = (HSI/2) * 12 = ~48 MHz */
289   RCC->CFGR &= (uint32_t)((uint32_t)~(RCC_CFGR_PLLSRC | RCC_CFGR_PLLXTPRE | RCC_CFGR_PLLMULL));
290   RCC->CFGR |= (uint32_t)(RCC_CFGR_PLLSRC_HSI_Div2 | RCC_CFGR_PLLXTPRE_PREDIV1 | RCC_CFGR_PLLMULL12);
```

main.c

- Example main()
  - Standard C main() function entry
  - Start of application program

```
045 /**
046   * @brief  Main program.
047   * @param  None
048   * @retval None
049   */
050 int main(void)
051 {
052   RCC_ClocksTypeDef RCC_Clocks;
053
054   /* Configure LED3 and LED4 on STM32F0-Discovery */
055   STM_EVAL_LEDInit(LED3);
056   STM_EVAL_LEDInit(LED4);
057
058   /* Initialize User_Button on STM32F0-Discovery */
059   STM_EVAL_PBInit(BUTTON_USER, BUTTON_MODE_GPIO);
060
```

stm32f0xx_it.c

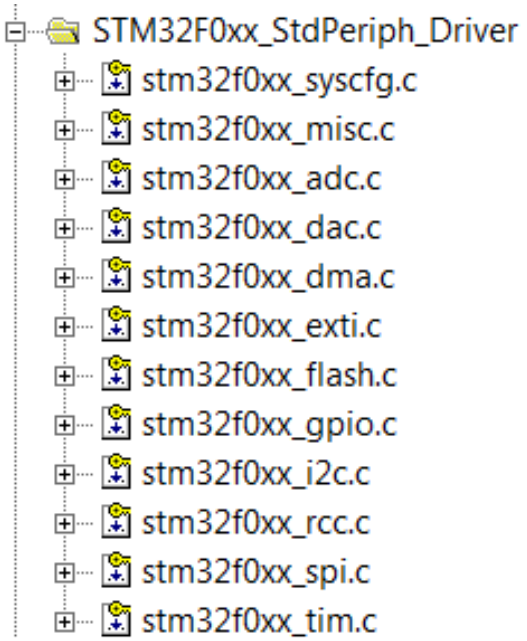
- Contains Cortex-M0 Processor Exception Handlers (ISRs)
  - void NMI_Handler(void);
  - void HardFault_Handler(void);
  - void SVC_Handler(void);
  - void PendSV_Handler(void);
  - void SysTick_Handler(void);
- Contains the STM32F0xx Peripherals Interrupt Handlers (default is empty)
- Add the Interrupt Handler for the used peripheral(s) (PPP), for the available peripheral interrupt handler's name please refer to the startup file: startup_stm32f0xx.s
  - void PPP_IRQHandler(void) {};

stm32f0-discovery.c

- Contains board specific function and definition
- Defines Push-button and LED GPIO definitions
- Contains board specific funcitons
  - void STM_EVAL_LEDInit(Led_TypeDef Led);
  - void STM_EVAL_LEDOn(Led_TypeDef Led);
  - void STM_EVAL_LEDOff(Led_TypeDef Led);
  - void STM_EVAL_LEDToggle(Led_TypeDef Led);
  - void STM_EVAL_PBInit(Button_TypeDef Button, ButtonMode_TypeDef Button_Mode);
  - uint32_t STM_EVAL_PBGetState(Button_TypeDef Button);

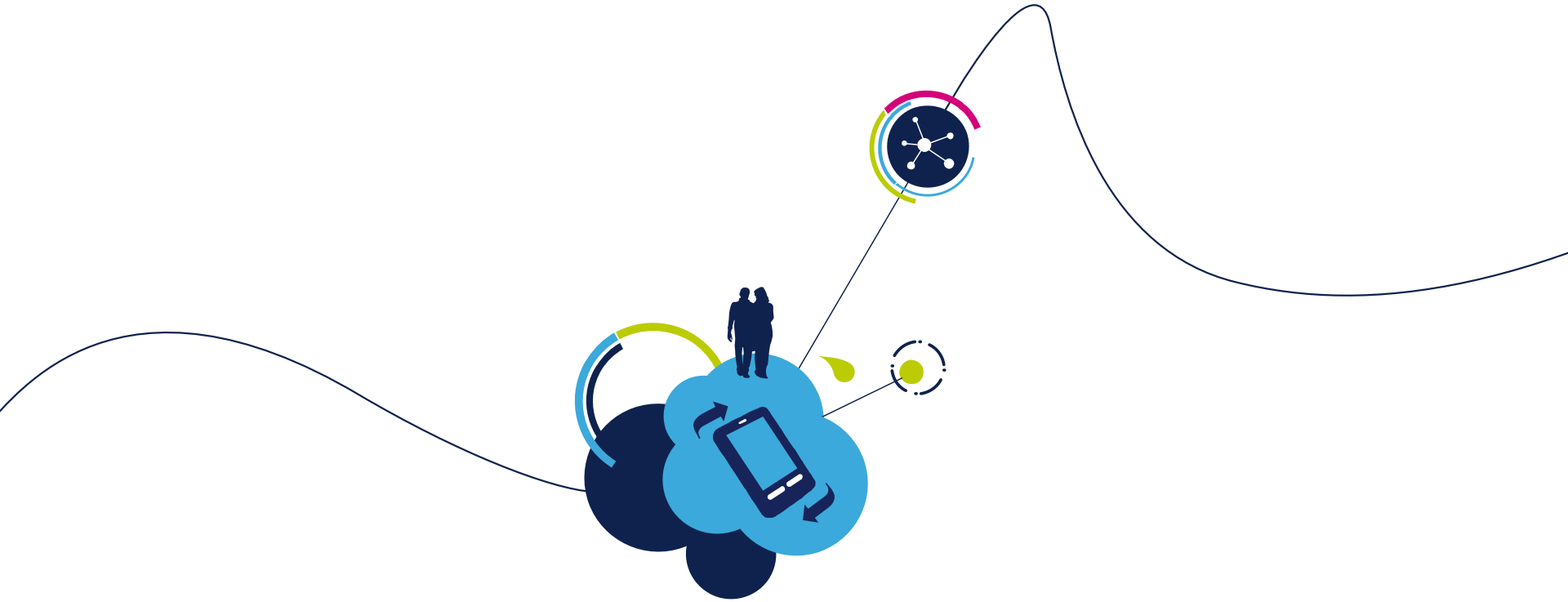
STM32F0xx_StdPeriph_Driver

- Each file contains library functions that can be used for each peripheral
- Abstracts register manipulation and gives a standard API for access to peripheral functions
- Example:




```
/**
  * @brief  Sets the selected data port bits.
  * @param  GPIOx: where x can be (A, B, C, D or F) to select the GPIO peripheral.
  * @param  GPIO_Pin: specifies the port bits to be written.
  * @note   This parameter can be GPIO_Pin_x where x can be:(0..15) for GPIOA,
  *         GPIOB or GPIOC,(0..2) for GPIOD and(0..3) for GPIOF.
  * @retval None
  */
void GPIO_SetBits(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin)
{
  /* Check the parameters */
  assert_param(IS_GPIO_ALL_PERIPH(GPIOx));
  assert_param(IS_GPIO_PIN(GPIO_Pin));

  GPIOx->BSRR = GPIO_Pin;
}
```

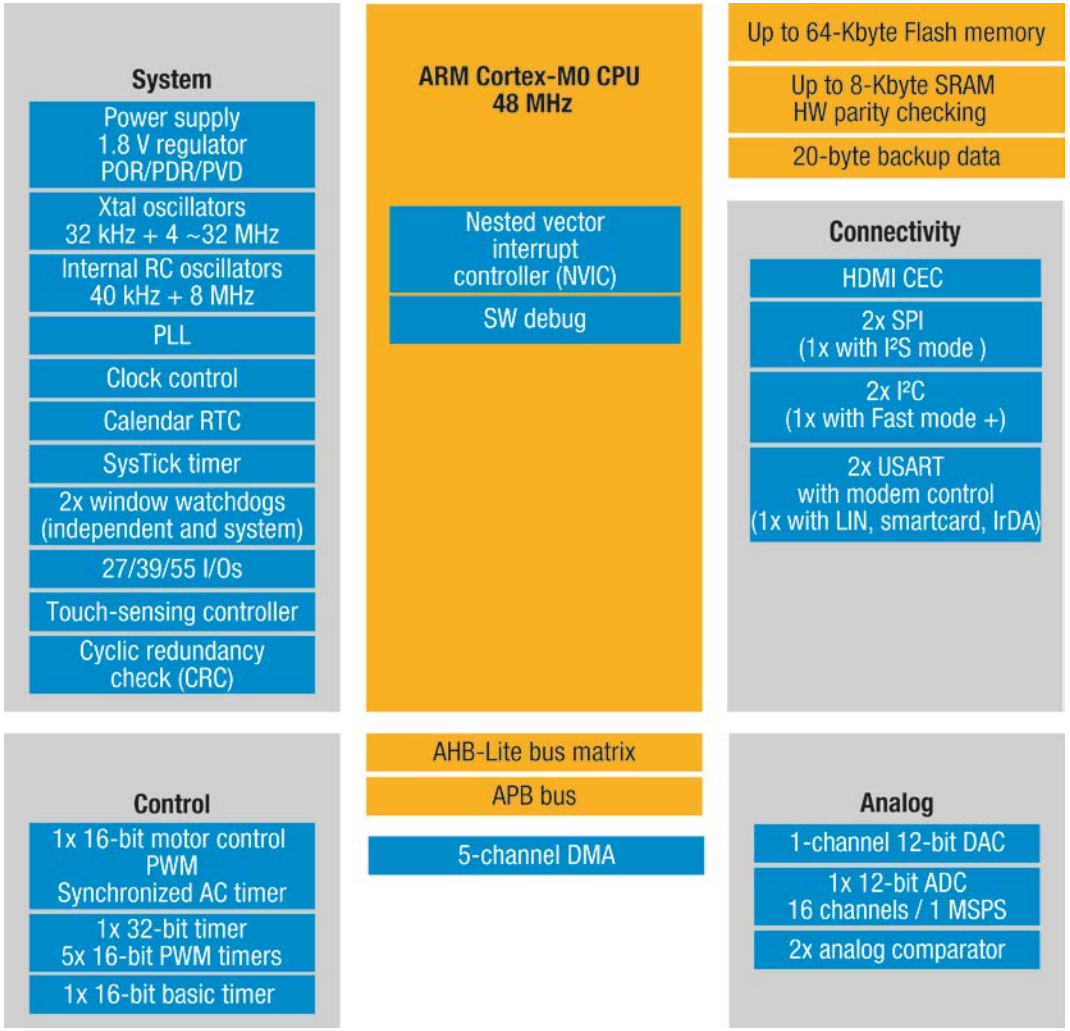


The STM32 F0 Series in Detail

life.augmented

STM32 F0 Series 64KB STM32F051

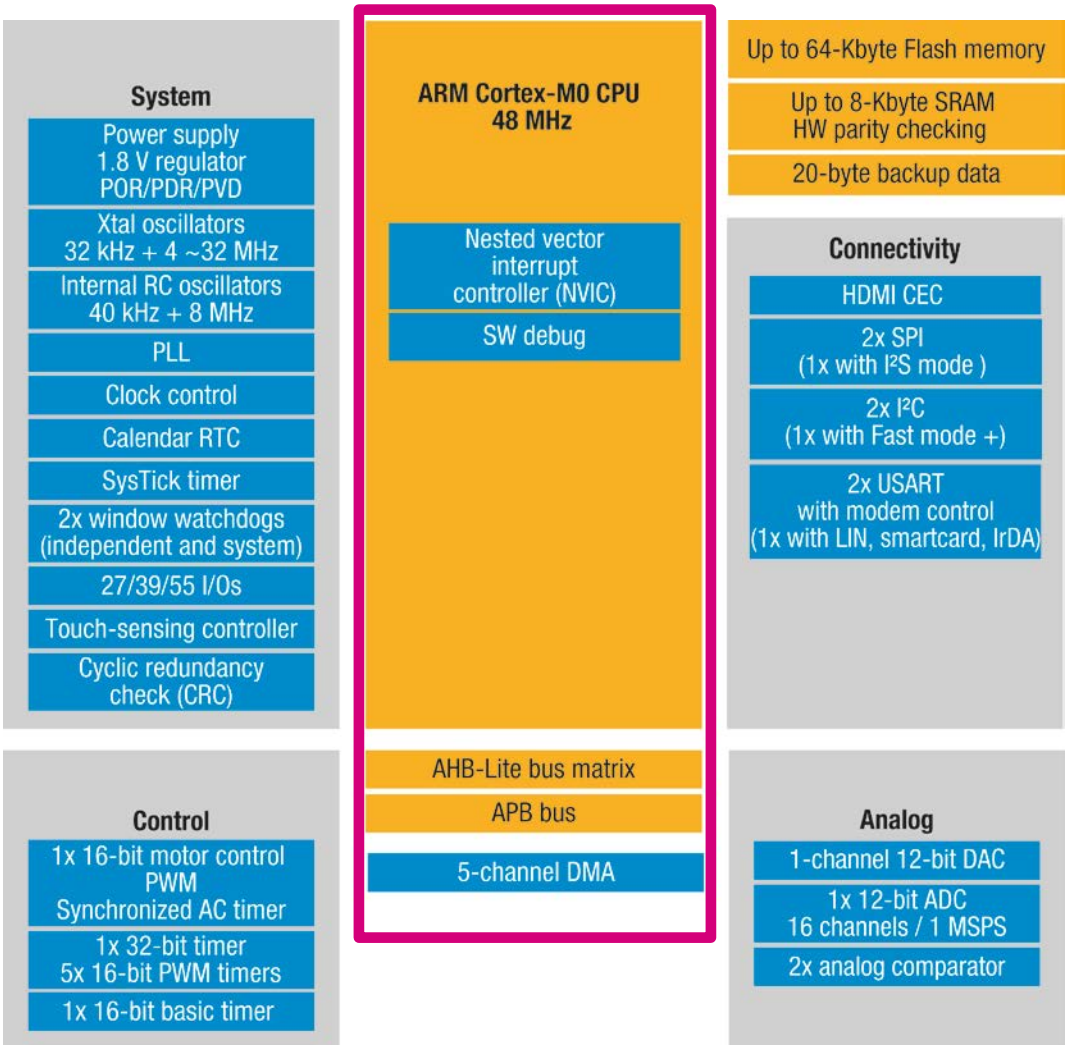
- **ARM 32-bit Cortex-M0 CPU**
- **Operating Voltage:**
  - V_{DD} = 2.0 to 3.6 V
  - V_{BAT} = 1.8 to 3.6 V
- **Safe Reset System (Integrated Power On Reset (POR)/Power Down Reset (PDR) + Programmable voltage detector (PVD))**
- **Embedded Memories:**
  - FLASH: up 64 Kbytes
  - SRAM: up 8 Kbytes
- **CRC calculation unit**
- **5 DMA Channels**
- **Power Supply with software configurable internal regulator and low power modes.**
- **Low Power Modes with Auto Wake-up**
- **Low power calendar RTC with 20 bytes of backup registers**
- **Up to 48 MHz frequency managed & monitored by the Clock Control w/ Clock Security System**
- **Rich set of peripherals & IOs**
  - 1 × 12-bit DAC with output buffer
  - 2 low power comparators (Window mode and wakeup)
  - Dual Watchdog Architecture
  - 11 Timers w/ advanced control features (including Cortex SysTick and WDGs)
  - 7 communications Interfaces
  - Up to 55 fast I/Os all mapable on external interrupts/event
  - 1x12-bits 1Msps ADC w/ up to 16 external channels + Temperature sensor/ voltage reference/VBAT measurement



life.augmented

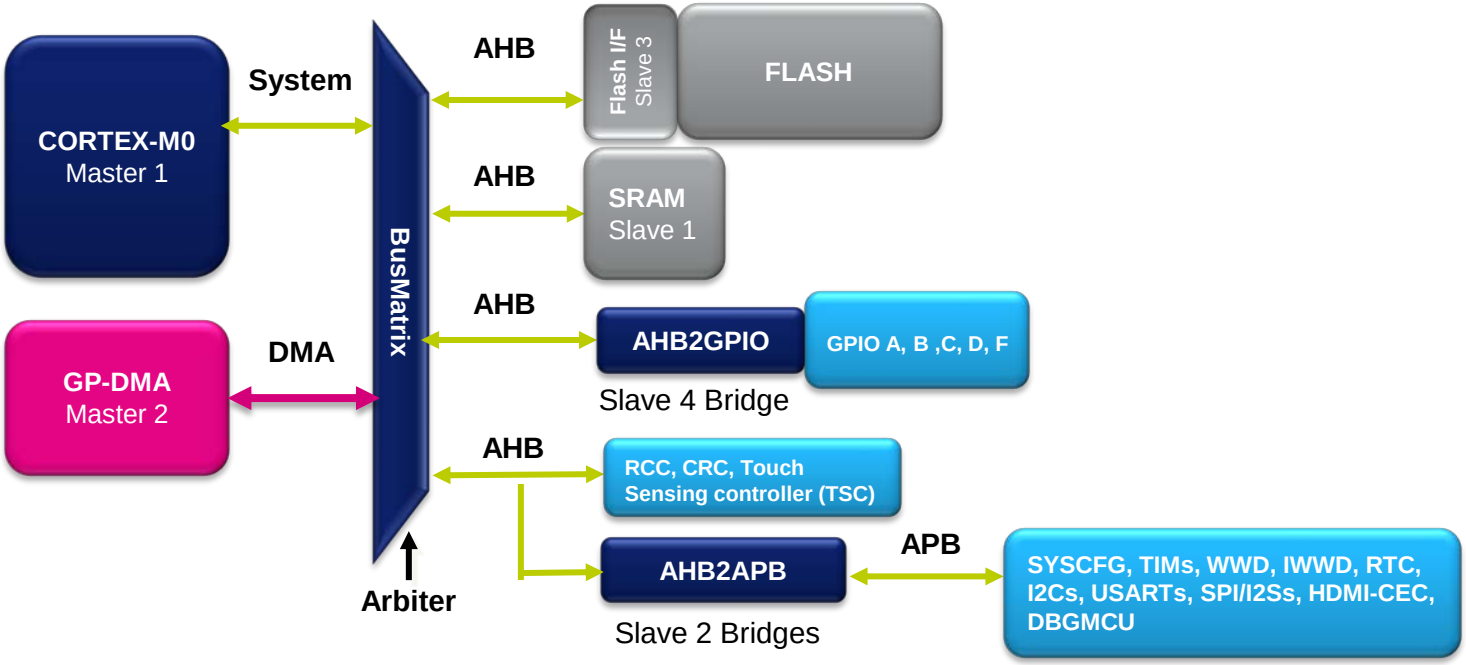
System Architecture

- ARM 32-bit Cortex-M0 CPU
- Nested Vector Interrupt Controller (NVIC)
- SWD
- Bus Matrix
- 5 DMA Channels



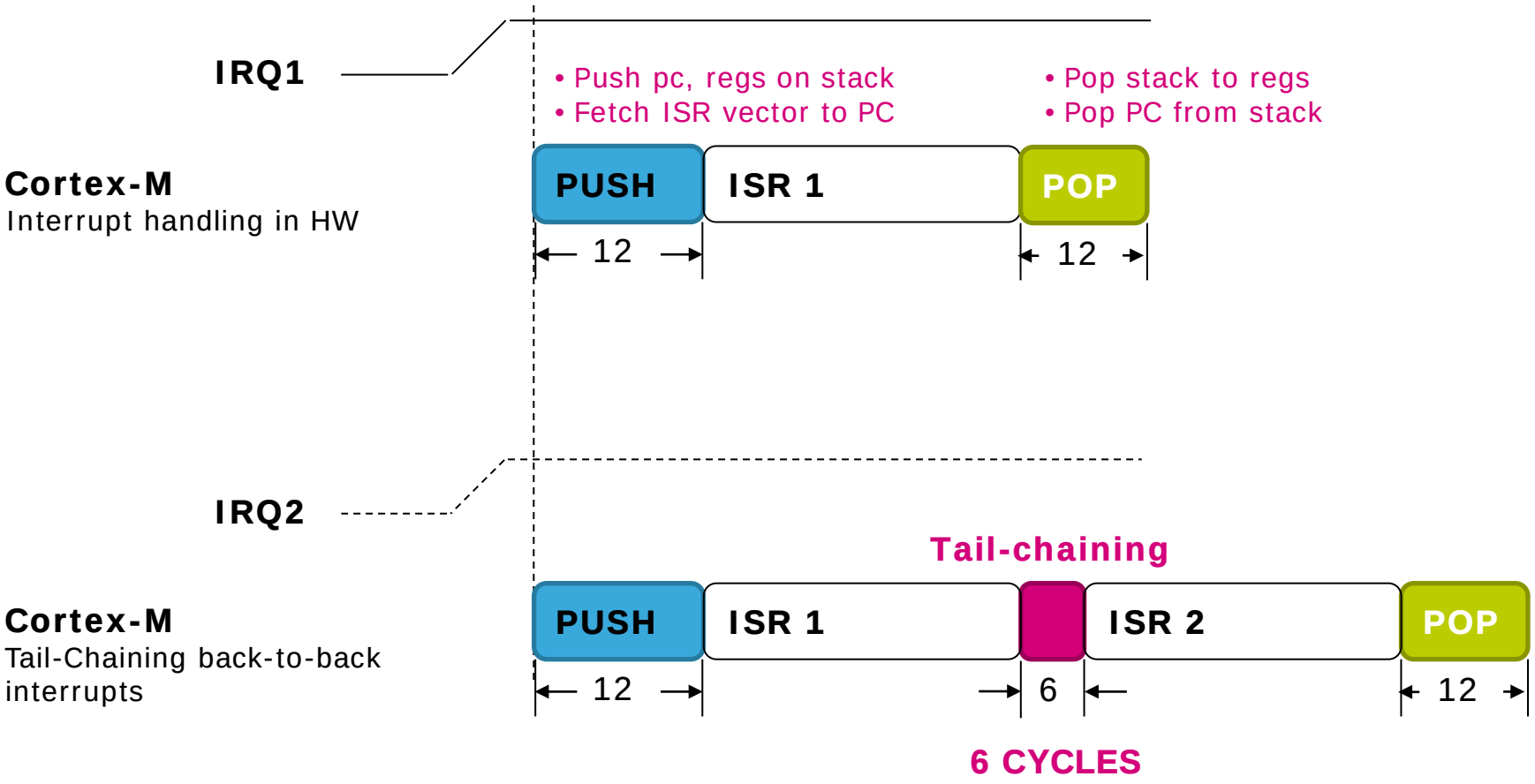
System architecture/DMA

- Cortex-M0 and DMA share the AHB bus matrix to allow parallel access
  - Multiple possibilities of bus access to SRAM, Flash, Peripheral, DMA
  - Von Neumann + Bus Matrix allows Flash execution in parallel with DMA transfer
  - Buses are not overloaded with data movement tasks
- Increase peripheral speed for better performance
  - Advanced Peripheral Bus (APB) architecture up to 48 MHZ



Cortex-M NVIC (Nested Vector Interrupt Controller)

- Interrupts are Fast AND Deterministic



Cortex-M0 Exception Types

No.	Exception Type	Priority	Type of Priority	Descriptions
1	Reset	-3 (Highest)	fixed	Reset
2	NMI	-2	fixed	Non-Maskable Interrupt
3	Hard Fault	-1	fixed	Default fault if other hander not implemented
4-10	Reserved	N.A.	N.A.	
11	SVCall	Programmable	settable	System Service call
12-13	Reserved	N.A.	N.A.	
14	PendSV	Programmable	settable	Pendable request for System Device
15	SYSTICK	Programmable	settable	System Tick Timer
16	Interrupt #0	Programmable	settable	External Interrupt #0
……	……	……	settable	……
47	Interrupt#31	Programmable	settable	External Interrupt #31

The NVIC supports up to 32 dynamically re-prioritizable interrupts each with 4 levels of priority

Memories: Flash, SRAM, Back-up Data Registers

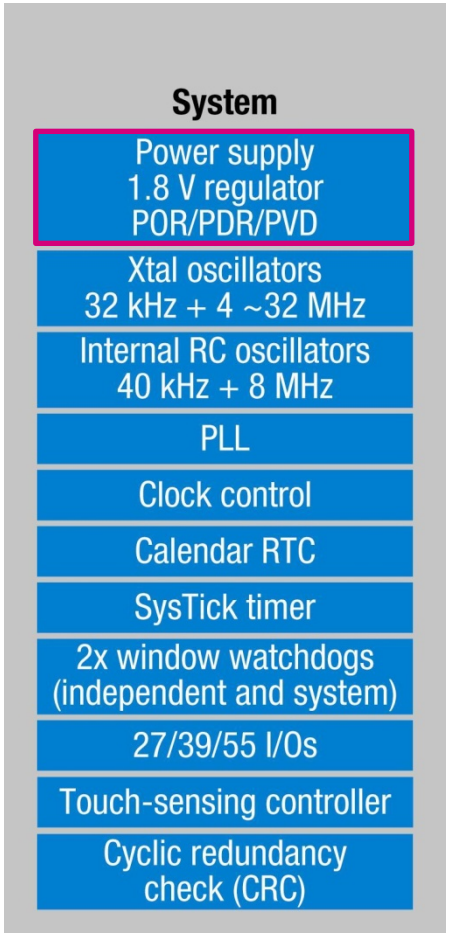
- Embedded Memories:
  - FLASH: up 64 Kbytes
  - SRAM: up 8 Kbytes
  - 20 bytes of backup registers

System


System
Power supply
1.8 V regulator
POR/PDR/PVD
Xtal oscillators
32 kHz + 4 ~32 MHz
Internal RC oscillators
40 kHz + 8 MHz
PLL
Clock control
Calendar RTC
SysTick timer
2x window watchdogs
(independent and system)
27/39/55 I/Os
Touch-sensing controller
Cyclic redundancy
check (CRC)
ARM Cortex-M0 CPU
48 MHz
Nested vector
interrupt
controller (NVIC)
SW debug
Up to 64-Kbyte Flash memory
Up to 8-Kbyte SRAM
HW parity checking
20-byte backup data
Connectivity
HDMI CEC
2x SPI
(1x with I²S mode)
2x I²C
(1x with Fast mode +)
2x USART
with modem control
(1x with LIN, smartcard, IrDA)
Control
1x 16-bit motor control
PWM
Synchronized AC timer
1x 32-bit timer
5x 16-bit PWM timers
1x 16-bit basic timer
AHB-Lite bus matrix
APB bus
5-channel DMA
Analog
1-channel 12-bit DAC
1x 12-bit ADC
16 channels / 1 MSPS
2x analog comparator

System: power supply

- Operating Voltage:
  - V_{DD} = 2.0 to 3.6 V
    - Note: VDD must be <= VDDA
  - V_{DDA} = 2.0 to 3.6 V (VDDA >= VDD)
  - V_{BAT} = 1.8 to 3.6 V
- Safe Reset System (Integrated Power On Reset (POR)/Power Down Reset (PDR) + Programmable voltage detector (PVD))
- Power Supply with software configurable internal regulator and low power modes.
- Low Power Modes with Auto Wake-up



Low Power Modes

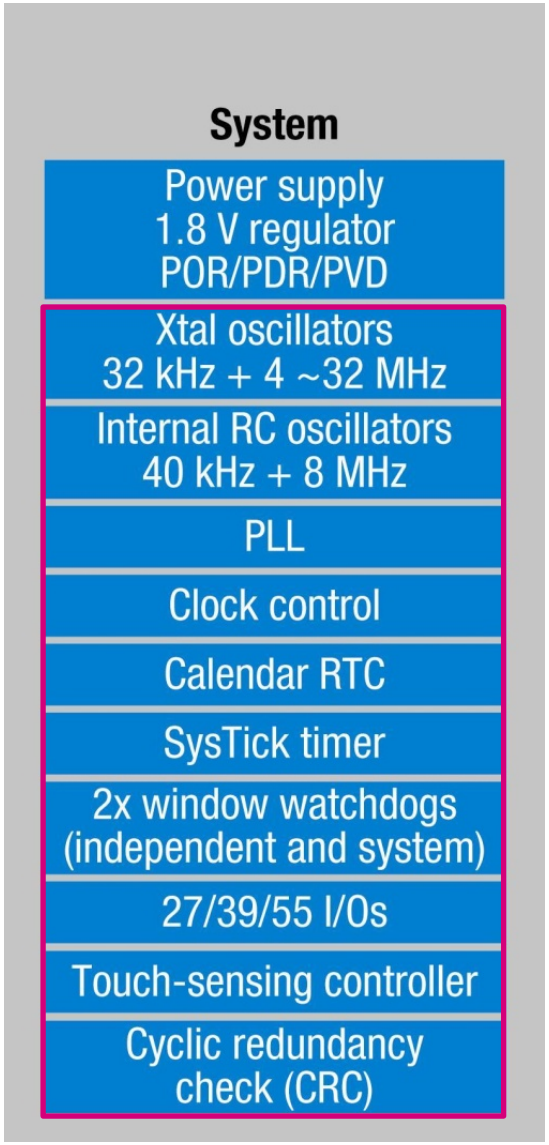
- STM32F05x Low Power modes: uses Cortex-M0 Sleep modes
  - SLEEP, STOP and STANDBY

Feature	STM32F05x typ IDD / IDDA (*)
RUN mode w/ execute from **Flash** on **48MHz** (HSE bypass 8MHz x 6 PLL = 48MHz) All peripherals clock ON	**22 / 0.260 (mA)**
RUN mode w/ execute from **Flash** on **24MHz** (HSE bypass 8MHz x 3 PLL = 24MHz) All peripherals clock ON	**12.2 / 0.185 (mA)**
RUN mode w/ execute from **Flash** on **8MHz** (HSI) All peripherals clock ON	**4.4 / 0.210 (mA)**
Sleep mode w/ execute from **Flash** at **48MHz** (HSI 8MHz / 2 x 12 PLL = 48MHz) All peripherals clock ON	**14 / 0.340 (mA)**
STOP w/ Voltage Regulator in low power All oscillators OFF, PDR on VDDA is OFF	**3.7 / 1.34 (µA)**
STANDBY w/ LSI and IWDG OFF PDR on VDDA is OFF	**1.3 / 1.21 (µA)**

Typical values are measured at TA = 25 °C, V_{DD} =3.6 V V_{DDA}= 3.6 V.

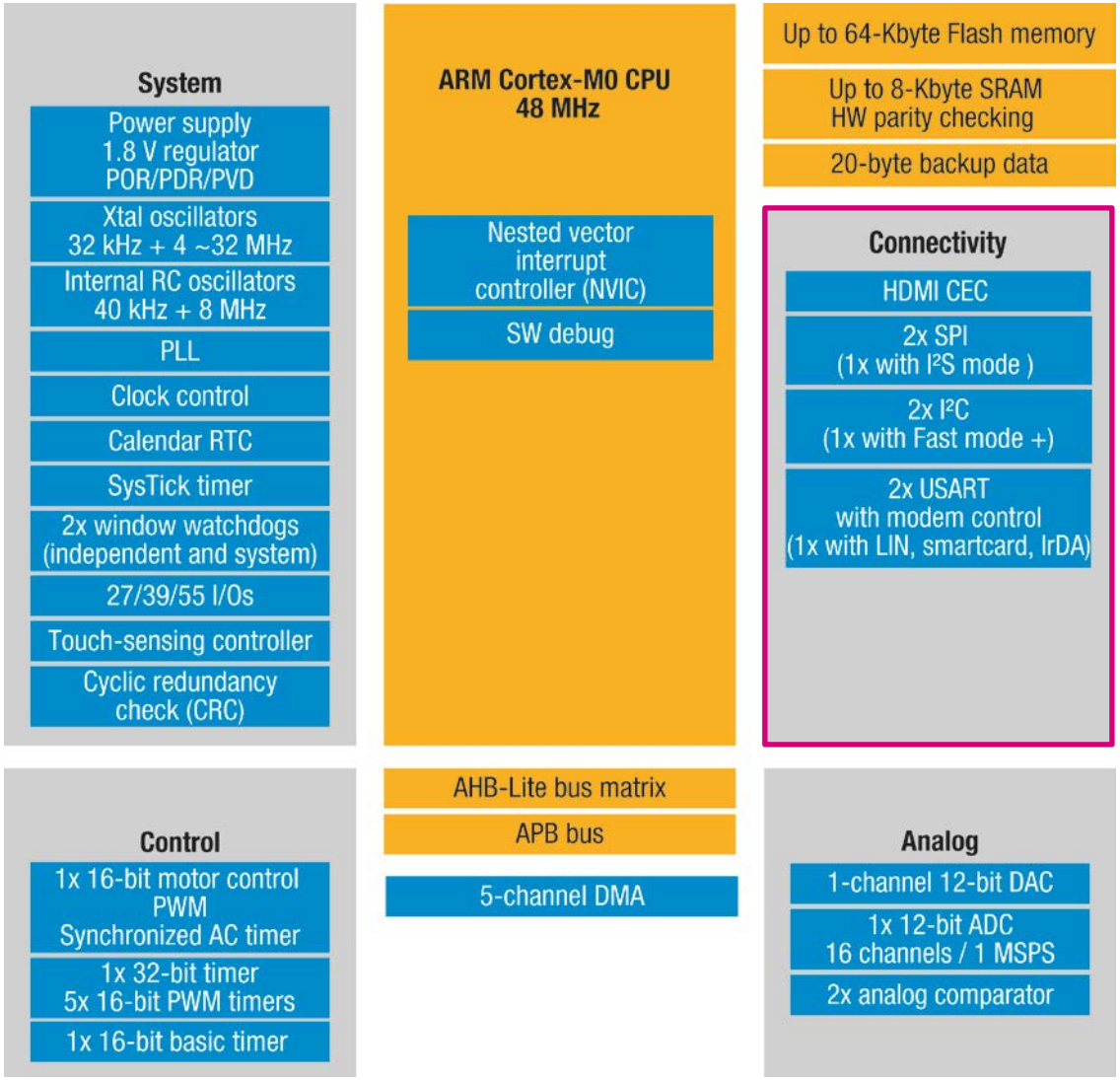
On-chip oscillators

- Up to 48 MHz frequency managed & monitored by the Clock Control w/ Clock Security System
  - External Oscillators 32kHz (RTC) and 4-32MHz (System Clock)
  - Internal RC Oscillators 40kHz (IWDG) and 8MHz
  - PLL
- Low power calendar RTC
- Dual Watchdog Architecture
- Up to 55 fast I/Os all mapable on external interrupts/event
- Touch Sensing Controller with up to 18 touch sensing electrodes
- CRC calculation unit



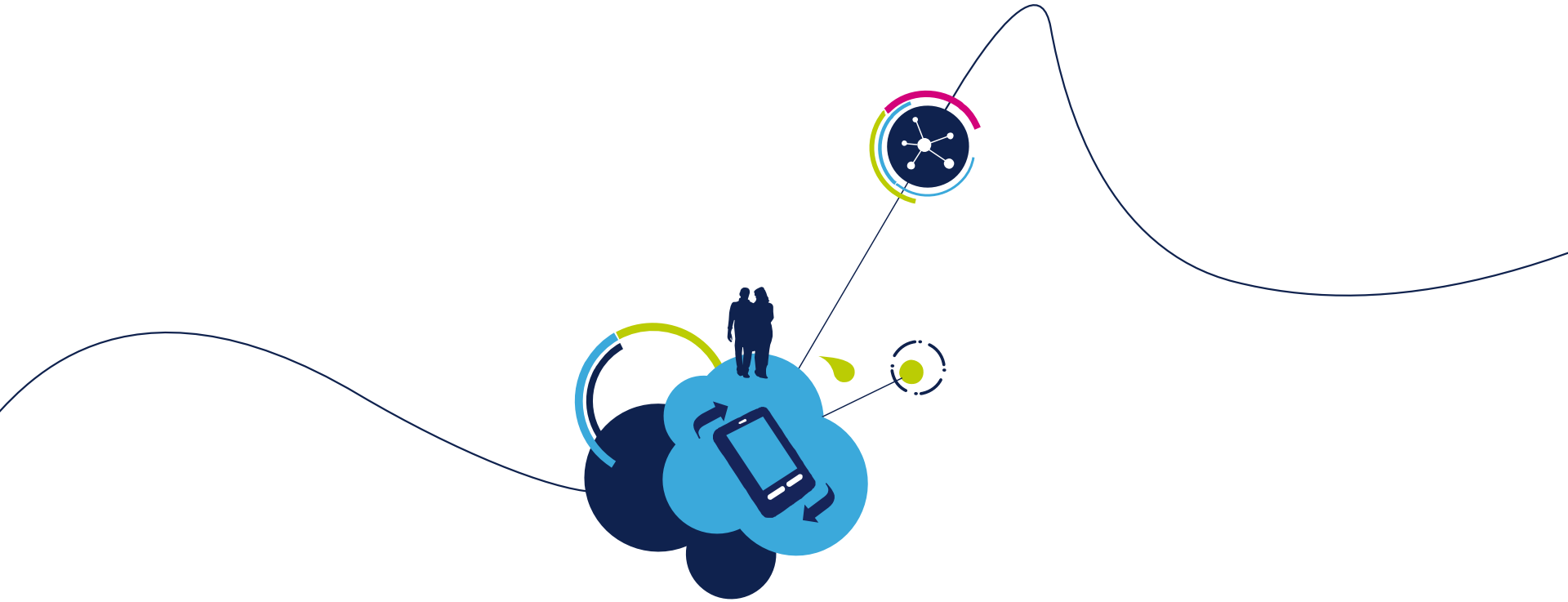
Connectivity

- 7 communications Interfaces



Analog

- 1 × 12-bit DAC with output buffer
- 1x12-bits 1Msps ADC w/ up to 16 external channels + Temperature sensor/ voltage reference/VBAT measurement
- 2 low power comparators (Window mode and wakeup)



Hands-on #2 NVIC_WFI_Mode

life.augmented

NVIC_WFI_Mode Example

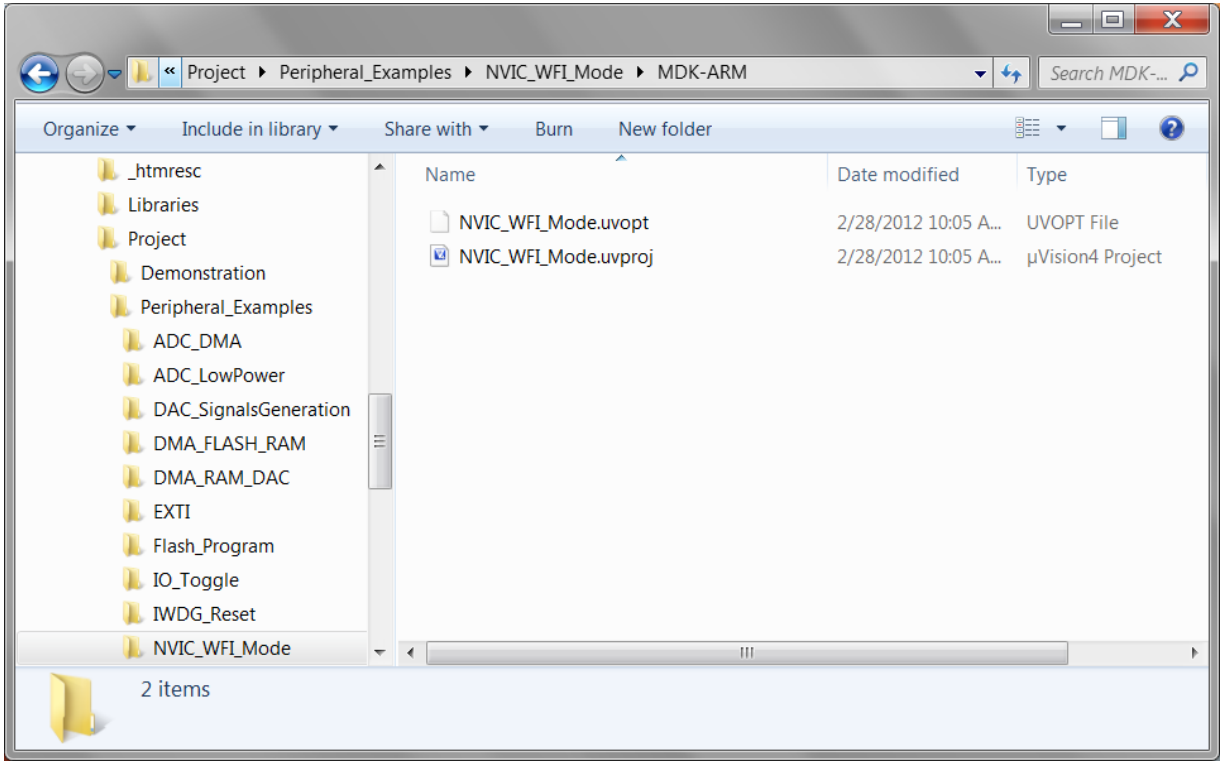
- NVIC (Nested Vector Interrupt Controller) using EXTI interrupt event
- WFI (Wait For Interrupt) places CPU to sleep (reduces power)
- Action Steps
  - Exit Debugger
  - Close Keil uVision

Open NVIC_WFI_Mode Project

- Go to the directory:

C:\STM32F0-Discovery_Kit\STM32F0-Discovery_FW_V1.0.0\Project\Peripheral_Examples\NVIC_WFI_Mode\MDK-ARM

- Double-click on NVIC_WFI_Mode.uvproj to open the project



Run Example

- Action Steps
  - Compile
  - Debug
  - Run
  - Validate!
- What happens when you press the USER button? Why?
- Continue
  - Stop; Reset
  - Place Breakpoint @ stm32f0xx_it.c line 117 (EXTI0_1_IRQHandler)
  - Run
  - Press USER pushbutton

main.c

1. Configure board specific functionality

- Init LED
- Init pushbutton

```
054 int main(void)
055 {
056   /*!< At this stage the microcontroller clock setting is already configured,
057        this is done through SystemInit() function which is called from startup
058        file (startup_stm32f0xx.s) before to branch to application main.
059        To reconfigure the default setting of SystemInit() function, refer to
060        system_stm32f0xx.c file
061     */
062
063   /* Initialize Led and USER Button mounted on STM32F0-Discovery Kit */
064   STM_EVAL_LEDInit(LED3);
065
066   STM_EVAL_PBInit(BUTTON_USER, BUTTON_MODE_EXTI);
```

stm32f0_discovery.c: GPIO & SYSCFG

- Enable PCLKs for GPIO and SYSCFG
- Configure GPIO pin

```
183 void STM_EVAL_PBInit(Button_TypeDef Button, ButtonMode_TypeDef Button_Mode)
184 {
185   GPIO_InitTypeDef GPIO_InitStructure;
186   EXTI_InitTypeDef EXTI_InitStructure;
187   NVIC_InitTypeDef NVIC_InitStructure;
188
189   /* Enable the BUTTON Clock */
190   RCC_AHBPeriphClockCmd(BUTTON_CLK[Button], ENABLE);
191   RCC_APB2PeriphClockCmd(RCC_APB2Periph_SYSCFG, ENABLE);
192
193   /* Configure Button pin as input */
194   GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN;
195   GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
196   GPIO_InitStructure.GPIO_Pin = BUTTON_PIN[Button];
197   GPIO_Init(BUTTON_PORT[Button], &GPIO_InitStructure);
```

stm32f0_discovery.c: EXTI & NVIC config

- Configure EXTI
- Configure NVIC

```
199  if (Button_Mode == BUTTON_MODE_EXTI)
200  {
201    /* Connect Button EXTI Line to Button GPIO Pin */
202    SYSCFG_EXTILineConfig(BUTTON_PORT_SOURCE[Button], BUTTON_PIN_SOURCE[Button]);
203
204    /* Configure Button EXTI line */
205    EXTI_InitStructure.EXTI_Line = BUTTON_EXTI_LINE[Button];
206    EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;
207    if (Button != BUTTON_USER)
208    {
209      EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Rising;
210    }
211    else
212    {
213      EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Falling;
214    }
215    EXTI_InitStructure.EXTI_LineCmd = ENABLE;
216    EXTI_Init(&EXTI_InitStructure);
217
218    /* Enable and set Button EXTI Interrupt to the lowest priority */
219    NVIC_InitStructure.NVIC_IRQChannel = BUTTON_IRQn[Button];
220    NVIC_InitStructure.NVIC_IRQChannelPriority = 0x03;
221    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
222
223    NVIC_Init(&NVIC_InitStructure);
224  }
```

stm32f0xx_it.c: set EXTI0 IRQ handler

- Place logic to handle EXTI exception
  1. Check interrupt status. Determine which interrupt triggered the exception
  2. Clear EXTI pending interrupt bit
  3. Perform action. (Set LowPowerMode = 1)

```
111 /**
112   * @brief  This function handles External lines 0 to 1 interrupt request.
113   * @param  None
114   * @retval None
115   */
116 void EXTI0_1_IRQHandler(void)
117 {
118   if(EXTI_GetITStatus(USER_BUTTON_EXTI_LINE) != RESET)
119   {
120     EXTI_ClearITPendingBit(USER_BUTTON_EXTI_LINE);
121
122     LowPowerMode = 1;
123   }
124 }
```

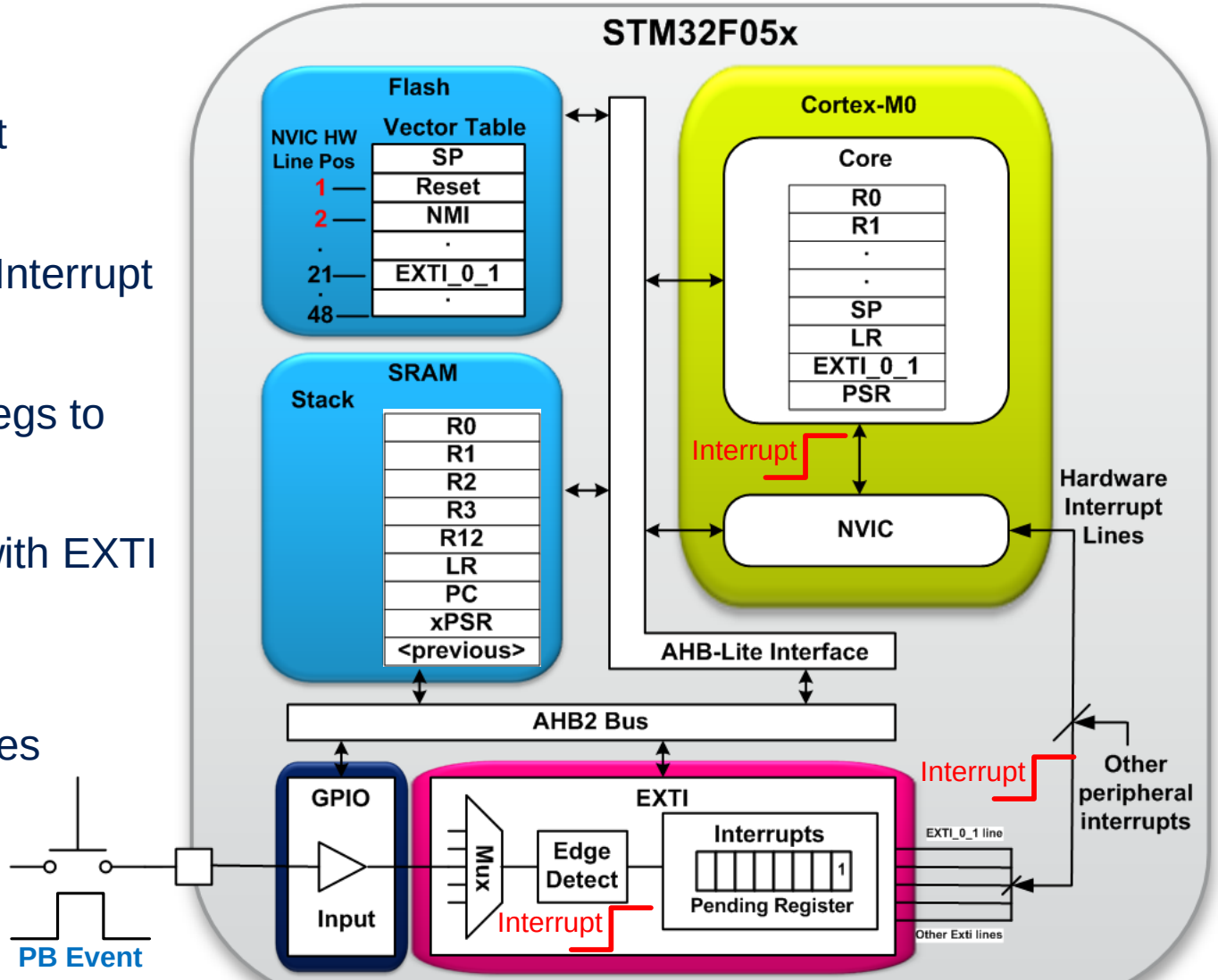
main.c: WFI

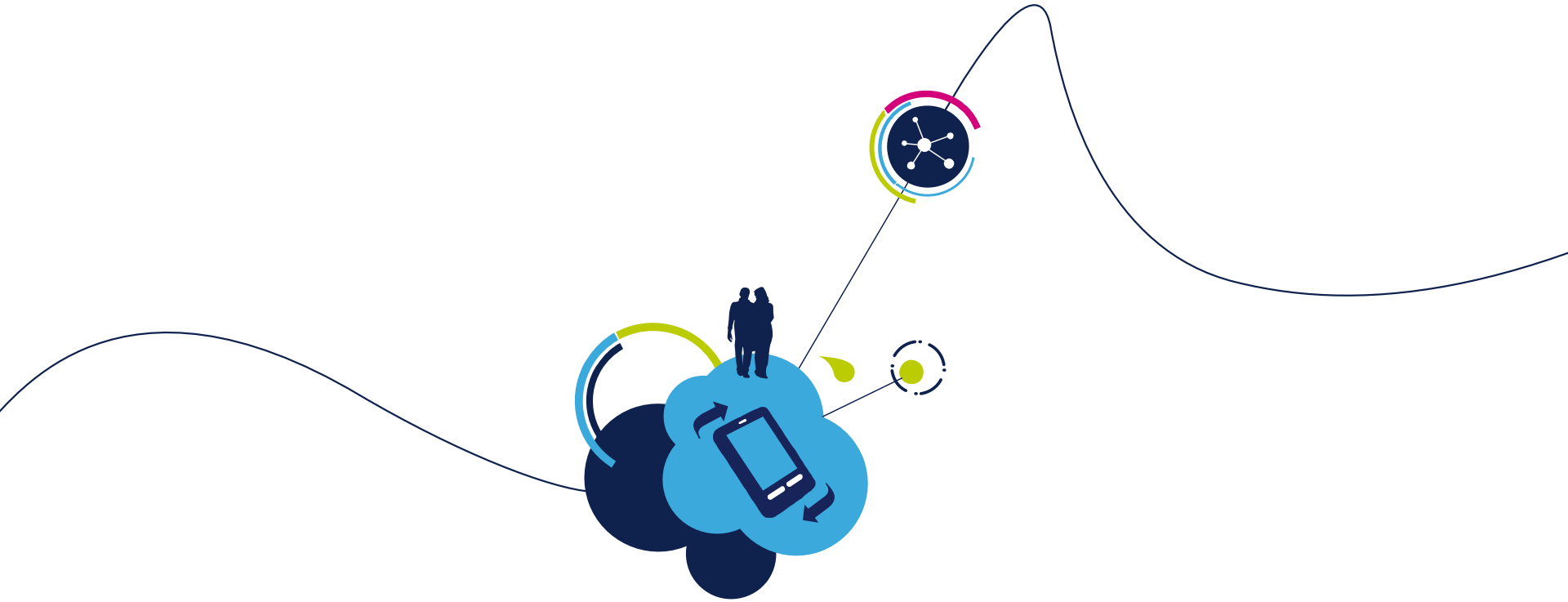
- The main control loop
  - EXTI interrupt controls the state variable LowPowerMode
  - Either flash LED or
  - Turn LED off and sleep using the Cortex M command Wait For Interrupt (WFI)

```
068   while (1)
069   {
070     if(LowPowerMode == 1)
071     {
072       /* Turn Off LED3 */
073       STM_EVAL_LEDOff(LED3);
074
075       /* Request to enter WFI mode */
076       __WFI();
077       LowPowerMode = 0;
078     }
079
080     Delay(0xFFFFF);
081     STM_EVAL_LEDToggle(LED3);
082   }
083 }
```

NVIC Linkage

1. PB triggers Event
2. EXTI Generates Interrupt
3. NVIC pushes 8 regs to stack
4. NVIC loads PC with EXTI vector
5. EXTI ISR Executes





Tips and Tricks:
Using the Firmware Library

life.augmented

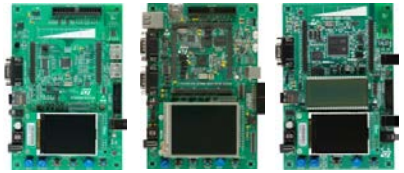
Tips and Tricks

- Start with the firmware library examples
- Basic peripheral configuration
  - Configure your clock tree (typically done with SystemInit() in system_stm32f0xx.c)
  - For each peripheral you plan on using:
    - Enable the PCLK to that peripheral (RCC_APB1PeriphClockCmd(…))
    - Determine GPIO usage
    - Enable GPIO PCLK(s); Alternate Function PCLK (if used)
    - Configure Alternate Function(s) as needed
    - Configure GPIO(s) pins
    - Configure the peripheral
    - Enable the peripheral (if needed)
- Interrupt usage and mapping
  - Add ISR in stm32f0xx_it.c (copy name from startup_stm32f0xx.s)
  - Enable interrupt in NVIC
  - Enable interrupt source in the peripheral

STM32 tools

Starter and promotion kits numerous boards

Motor Control kit



Evaluation boards:

STM320518-EVAL, STM3240G-EVAL and STM32L152D-EVAL

STM32 W evaluation kit

STM32W-SK

STM32 promotion kits

EvoPrimer

STM32-ComStick

4 Discovery kits

More than **15** different development IDE solutions

IAR SYSTEMS, KEIL An ARM® Company, atollic, iSYSTEM, RAISONANCE, TASKING, Green Hills SOFTWARE, SIGNUM SYSTEMS, RA, SEGGER, hitex DEVELOPMENT TOOLS, LAUTERBACH, Altium, ajji SYSTEM, mbest

More than **25** different RTOS and stack solution providers

HighIntegritySystems, interniche technologies, inc., KEIL Tools by ARM, SEGGER, Quadros, eCosCentric, CMX SYSTEMS, VDE, Green Hills SOFTWARE, freeRTOS, port, IXXAT, MESCO Engineering, embeX, ANDREA, expresslogic, ALPWISE, µd Micro Digital, embedded solutions, Micrium, MicroControl, embeddedlabs, RoweBots, JUNGO, ARC CORE, AVIX-RT, HCC

life.augmented

ST standard peripheral lib

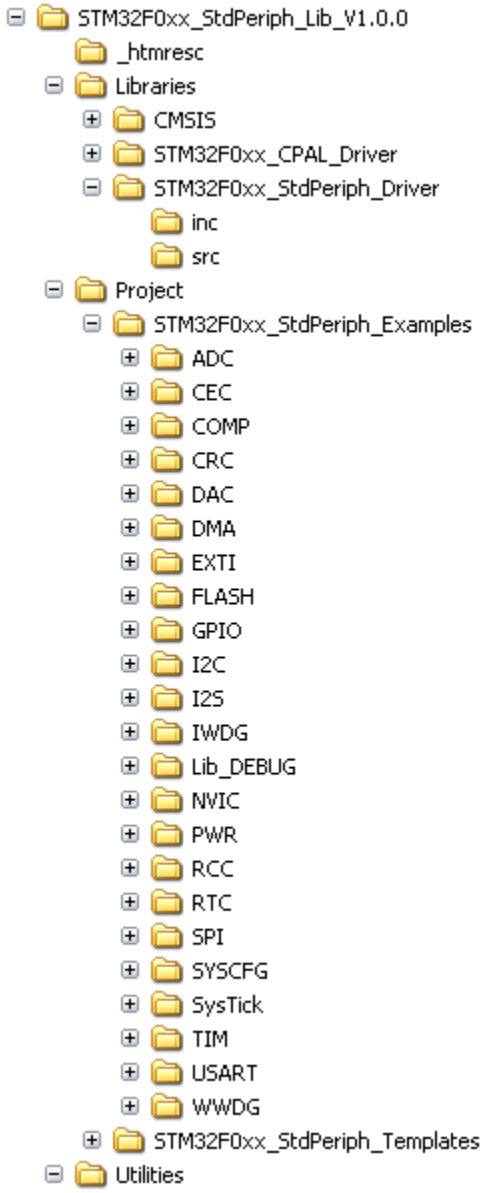
Hardware abstraction layer fully covering the microcontroller, STM32 or STM8

Compliant with standards

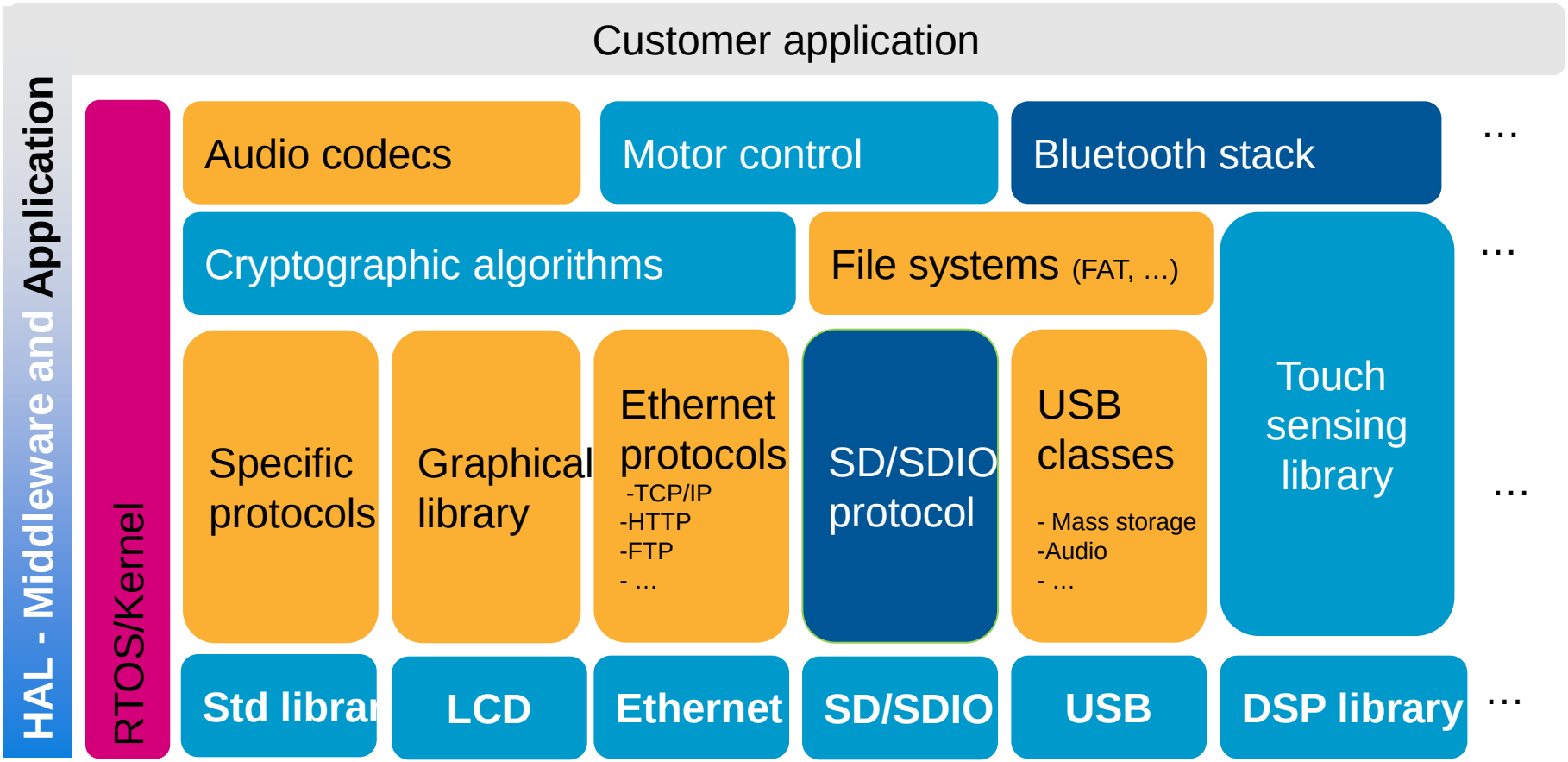
- ANSI-C source code
- Misra and ST coding rules
- ARM-CMSIS compliant for STM32

As real help for developers

- Comes with a multitude of examples demonstrating usage



STM32 – firmware solutions



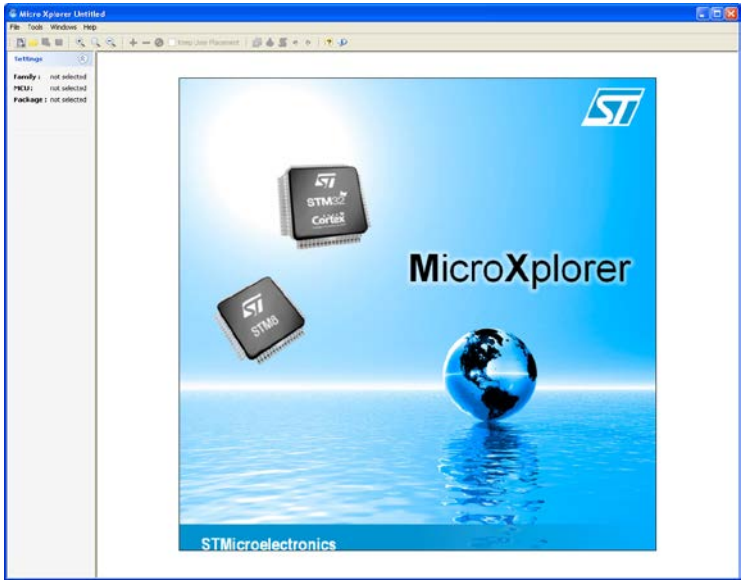
Free (ST or open source)
3rd parties
Both

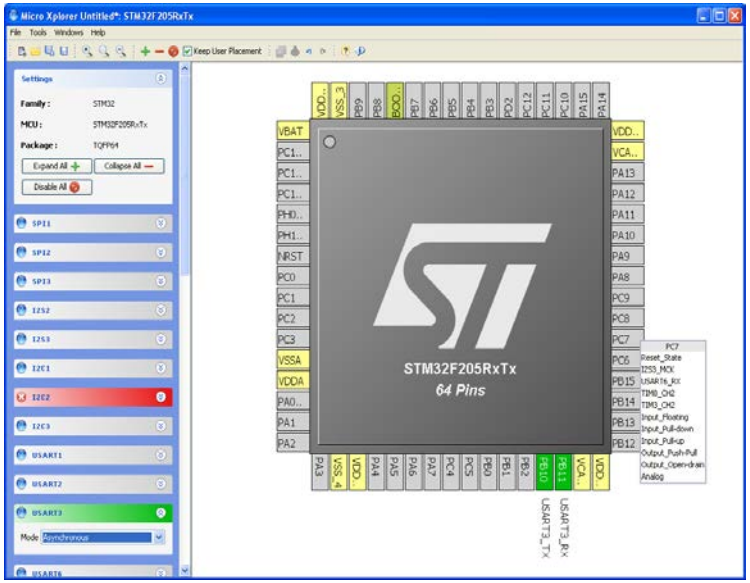
→ We provide more than just silicon

life.augmented

MicroXplorer: STM32 configuration tool

- Product Selection
- Peripherals configuration
- Pinout configuration
- Conflicts resolution

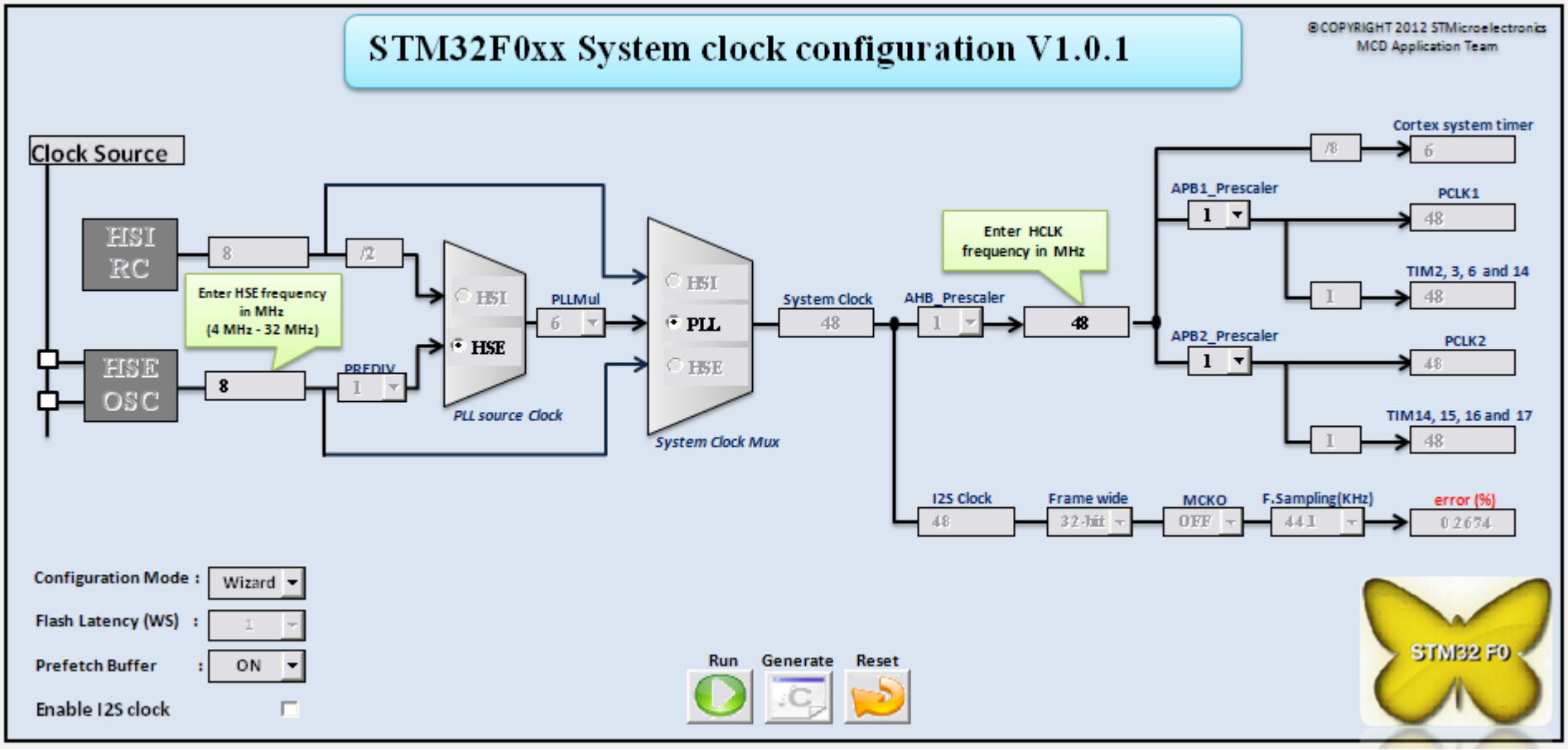




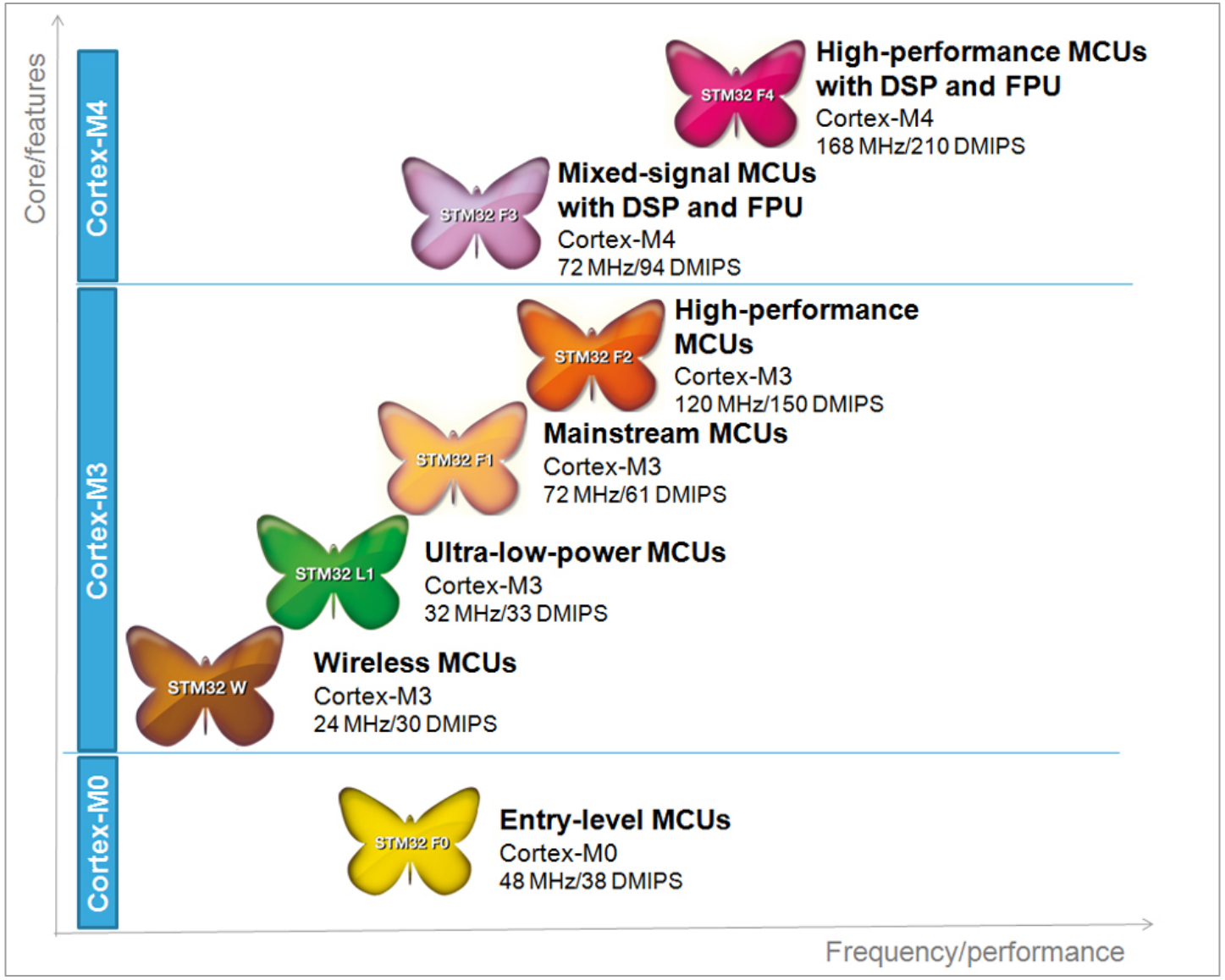
www.st.com/microxplorer

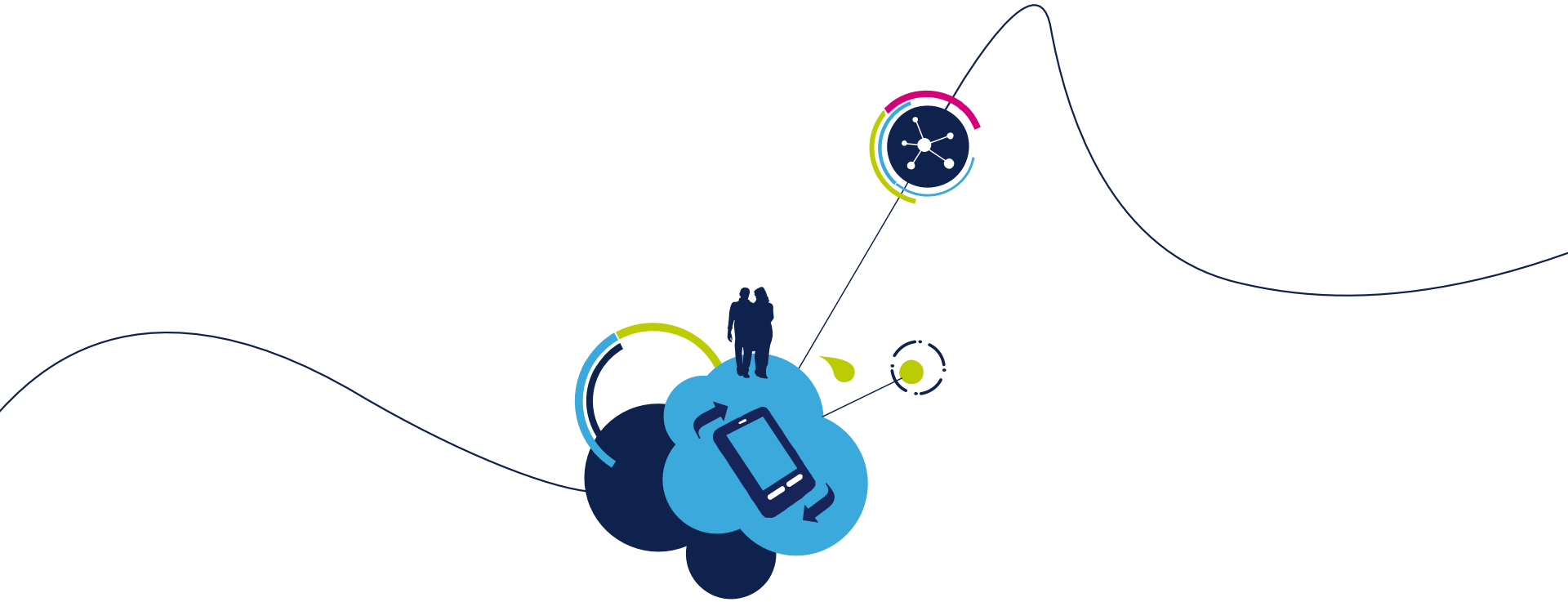
Clock Configuration Utility

- Helps configure the microcontroller clocks
- Generates the *system_stm32f0xx.c file*

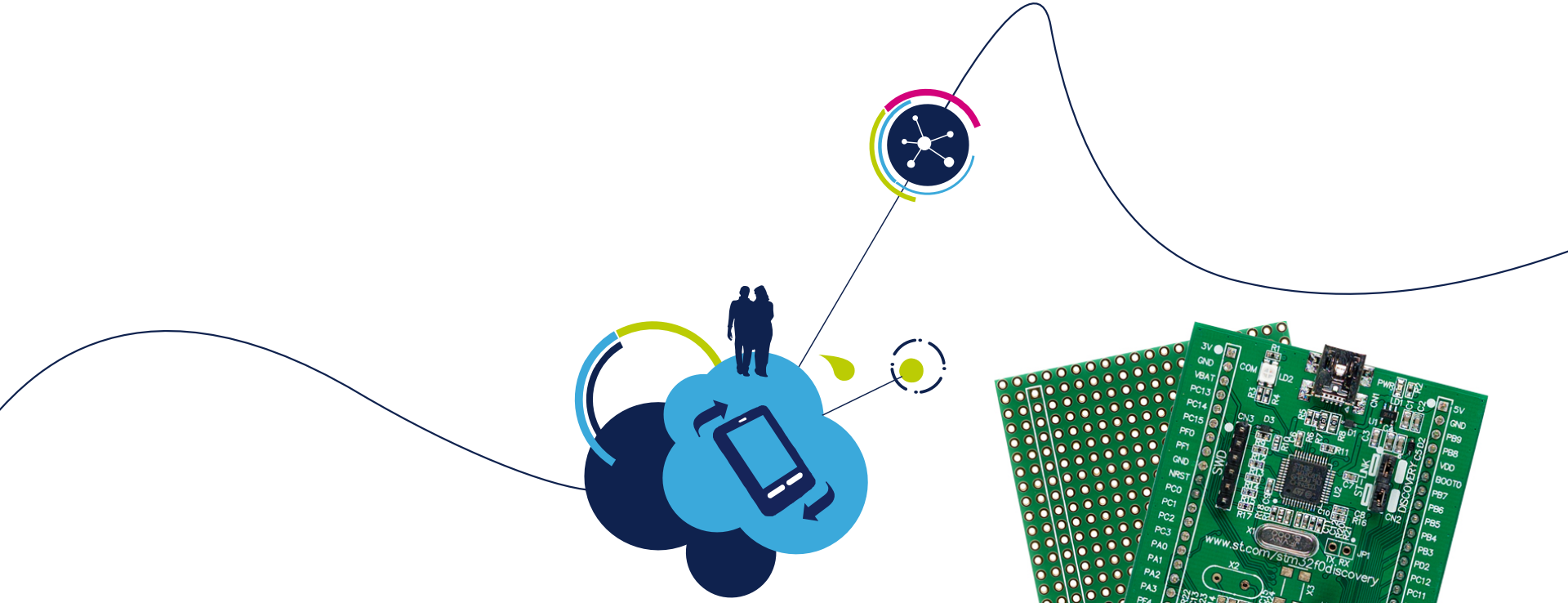


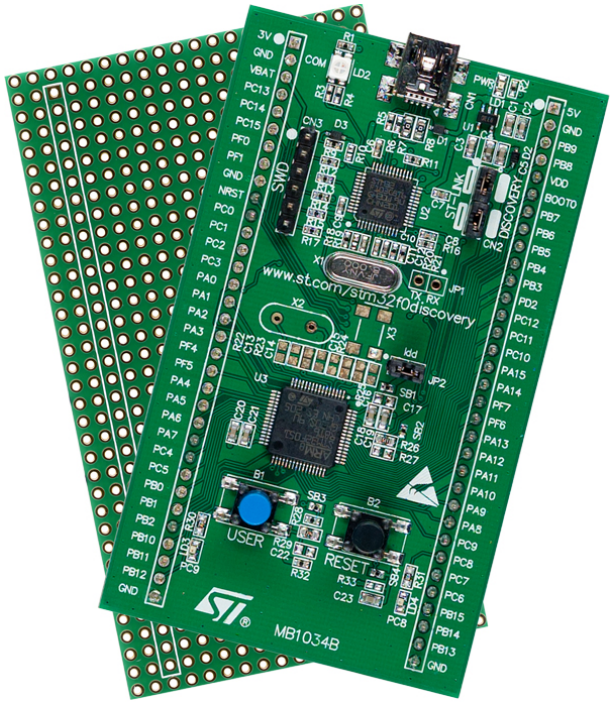
STM32 portfolio overview





Questions and Answers





Thank you

www.st.com/stm32f0discovery

life.augmented