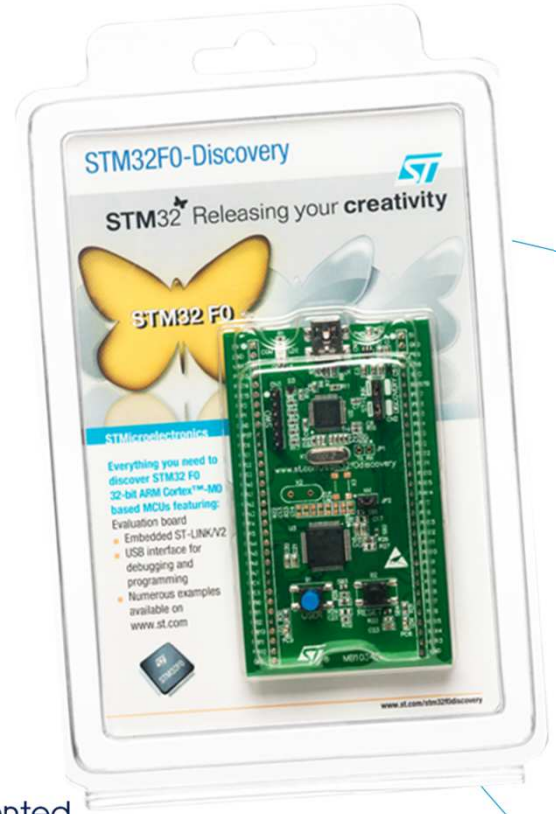
STM32F0-Discovery Training

Hanoi, Vietnam

Nov 2013



life.augmented

Agenda

- Introduction
- Installation
- The STM32 F0 Part 1: Cortex-M0
- The STM32 F0 Part 2: Peripherals
- Hands-on Training
- Review
- Questions and Answers

Objectives

- Hands-on workshop to show you the steps needed to quickly get up and running with the STM32F0-Discovery kit
- An overview of the STM32 F0 Series
- Know where to go for documentation, firmware libraries and application notes and additional ecosystem support
- Know where to obtain additional technical support

Questions?

- The technical team is there to support you
- If you have an issue or a problem, please raise you left hand and make eye contact with one of the available members of the technical team who will assist you
- Feel free to ask you neighbor
- Please keep side conversations to whisper level
- For any product specific questions, please write them down and pass them to one of the members of the technical team.
- We will use the last 20 minutes of our time to answer
  - Written questions
  - Any further questions you may have

Systems Check

- Everyone should have
  - A Windows Laptop (XP, Vista or Win 7)
  - STM32F0DISCOVERY kit
  - USB Cable
  - USB Flash Drive
- Ready to begin?

Step #1 - File Installation

- Insert the USB Flash Drive into your Laptop
- Copy the folder “d:\STM32F0DISCOVERY Kit” on the USB flash drive to your root “c:\” folder
  - C:\STM32F0DISCOVERY Kit\
- Enter this directory. You will find the following:
  - In the \Coocox directory: CoIDE v1.7.5.1 set-up file
  - In the \GCC Compiler: GCC ARM compiler set-up file
  - In the ST-LINK USB driver: ST-Link V2 USB driver set-up file
  - Documentation folder containing all relevant documentation for this training
  - Firmware Library folder

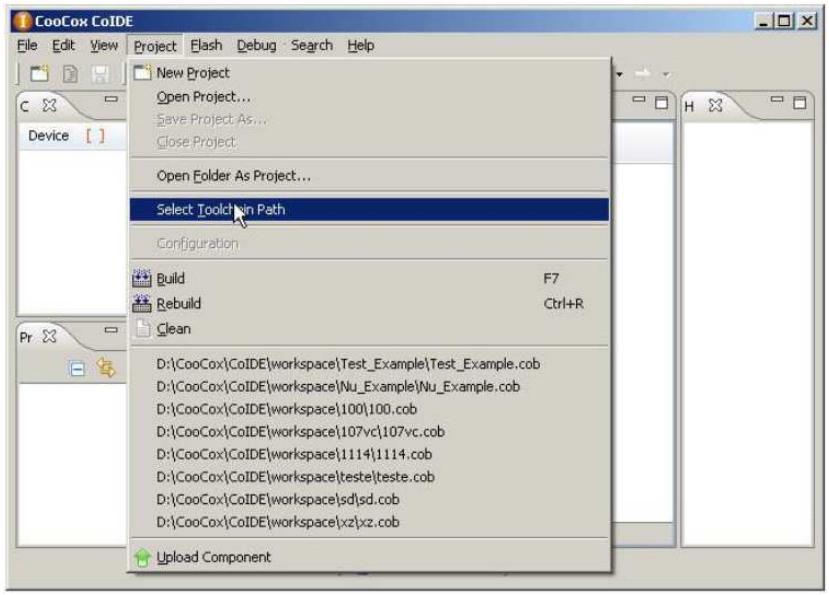
Hardware Preparation

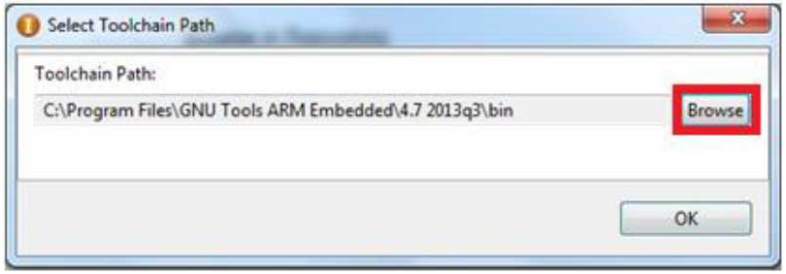
- Download and install the ST-Link/V2 USB driver
  - http://www.st.com/web/en/catalog/tools/PF258167
- Take out the STM32 F0 Value Line Discovery board
- Connect the board to the PC with a Type-A to mini-B USB cable
- Allow Windows OS to detect and install the USB driver for the ST-Link/V2 debugger/programmer

Step #2 - Coocox CoIDE Preparation

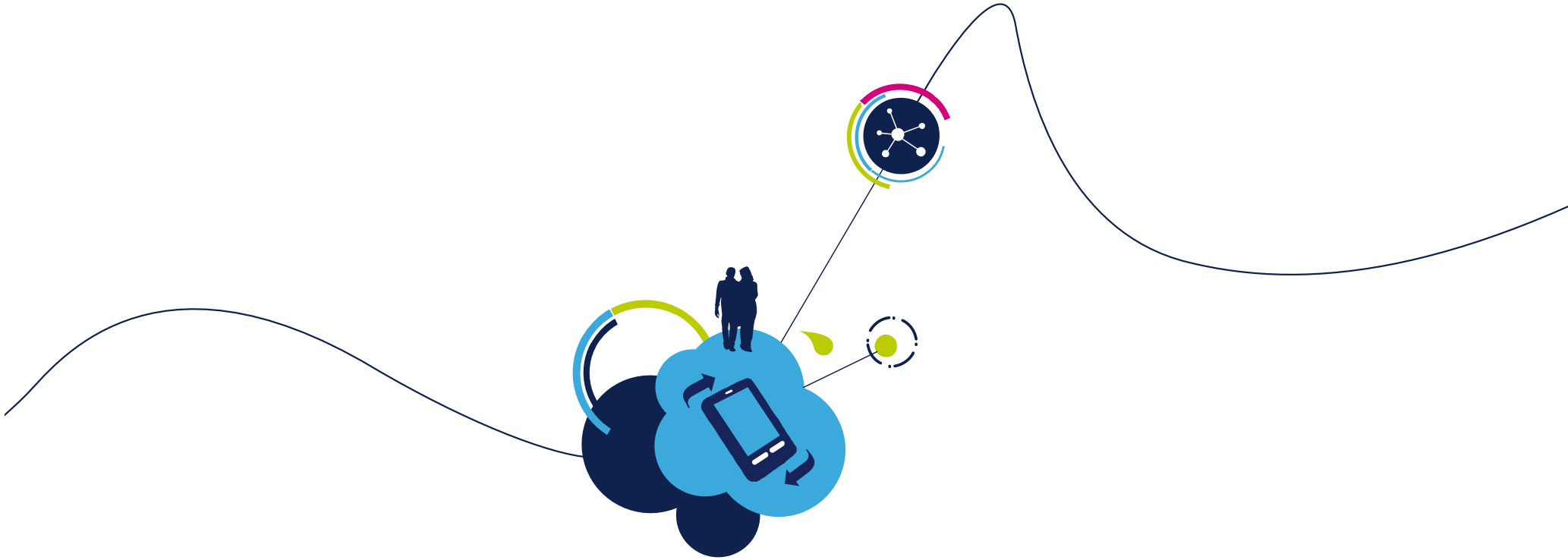
Preparation – Installing CoIDE and Drivers

- Download and install the latest CoIDE software development environment found at http://www.coocox.org/Tools/CoIDE-1.7.5.1.exe
- Configure GCC tool chain for CoIDE.
  - Download and install ARM GCC can be downloaded at
    - https://launchpad.net/gcc-arm-embedded/+download
  - Follow below steps to configure GCC tool chain.
    - After launching CoIDE, click "Select Toolchain Path" under the Project menu.
    - Click the “Browse” button; select the folder that contains the arm-none-eabi-gcc.exe and the other GCC execute files (You can find the folder where you install the ARM GCC tool chain). And Click “OK” button to save the setting.
  - Download and install ST-LINK/V2 USB driver found at http://www.st.com/web/en/catalog/tools/PF258167





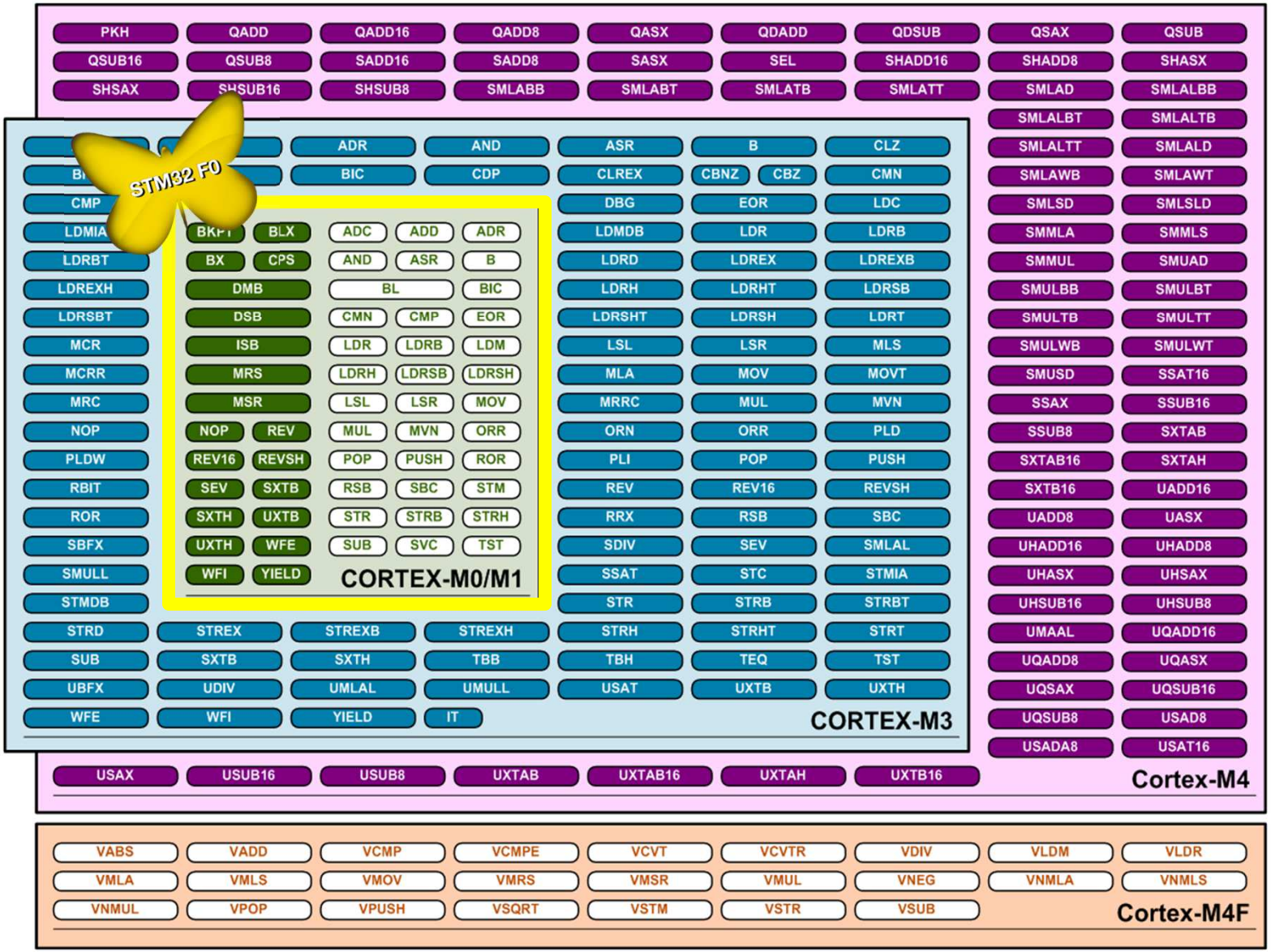
life.augmented



Introducing the ARM Cortex-M0

life.augmented

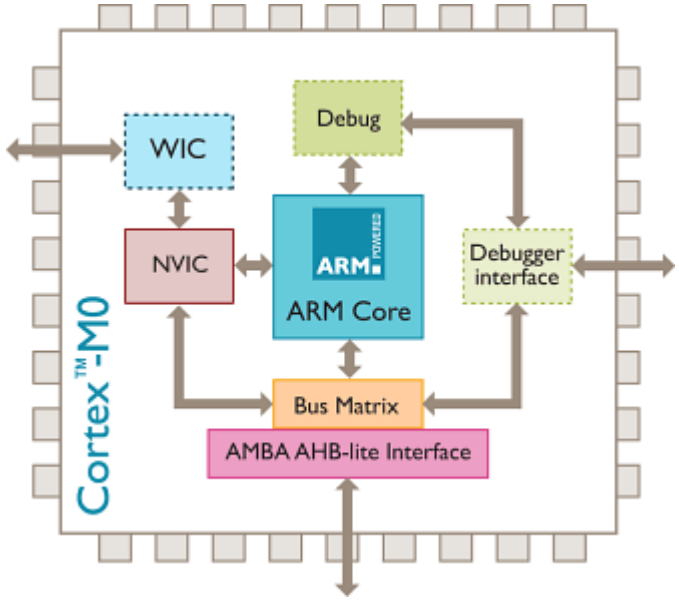
Cortex-M processors binary compatible



life.augmented

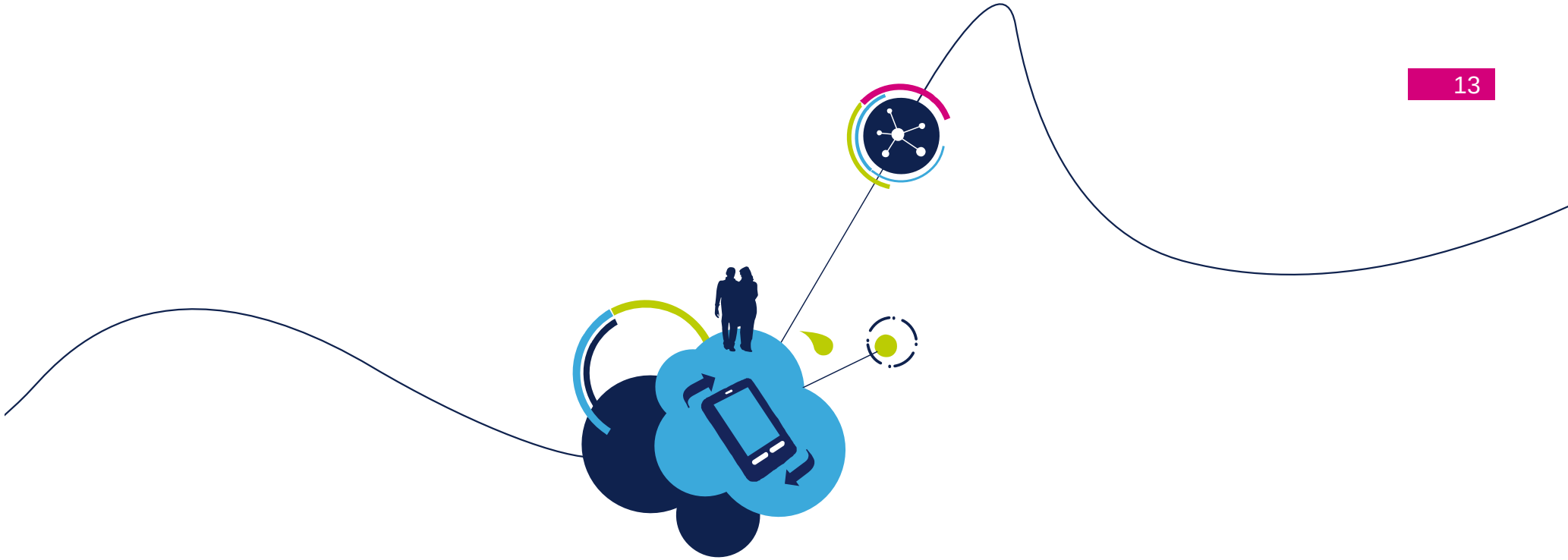
Cortex-M0 processor microarchitecure

- ARMv6M Architecture
  - Thumb-2 Technology
  - Integrated configurable NVIC
  - Compatible with Cortex-M3
- Microarchitecture
  - 3-stage pipeline
  - 1x AHB-Lite Bus Interfaces
- Configurable for ultra low power
  - Deep Sleep Mode, Opt. Wakeup Interrupt Controller
    (Not available for STM32F05)
- Flexible configurations for wider applicability
  - Configurable Interrupt Controller (1-32 Interrupts and Priorities)
  - No Memory Protection Unit
  - Optional Debug & Trace



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



Introducing STM32 F0 Series

STM32 F0 Series: Key Features

Real-time performance

Cortex

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 V or 2…3.6 V supply,
Fast wake-up time

Superior and innovative peripherals

1Mbps I²C Fast mode+
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,
SW library)

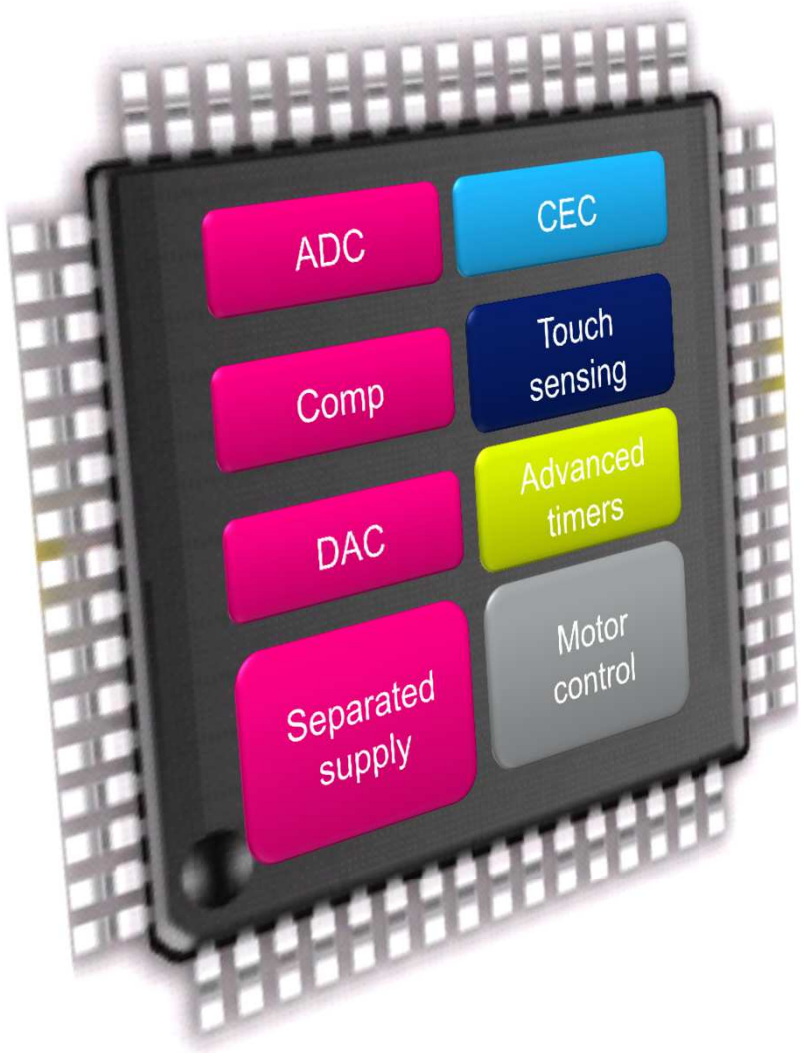
life.augmented

STM32 F0 series

L1, F0, F1, F2, F4 series: seamless migration amongst 300 pin-to-pin compatible part #s

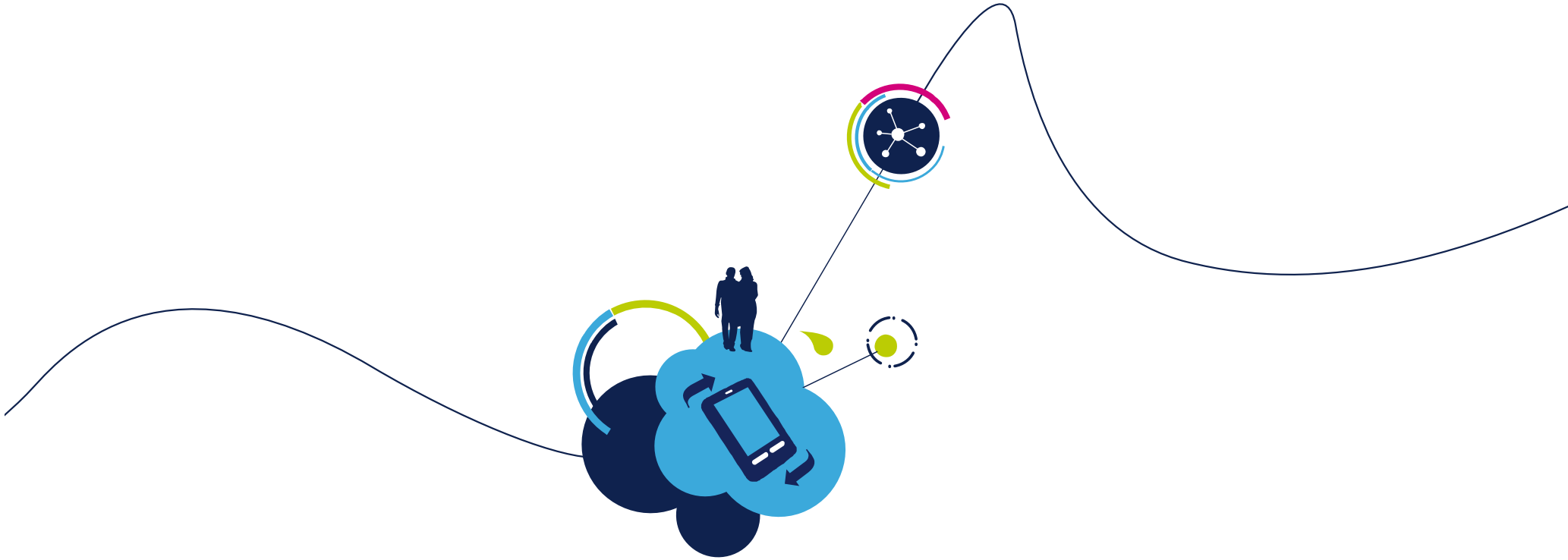
Innovative Peripherals

- Analog
  - 12 bit ADC with 1MSPS
  - 12 bit DAC
  - 2x Comparators
  - Separate supply for improved accura
- HDMI Consumer Electronics Cor (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 mappe on up to 28 pins
- Motor control
  - Permanent Magnet Synchronous Motors (PMSM)



Maximum Integration

- Meets industry safety specifications
  - **Class B-ready** for appliance
  - Hardware **RAM** parity check
  - **Clock Security System** (CSS) for switching to back-up internal RC in case of external clock failure
  - **2x Watchdogs (2x WDG)** 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



Great Fit for Applications

life.augmented

STM32F051/0/2 targeted applications

Industrial

Timers
communication
peripherals

Electricity meters
Home automation, HVAC, sensors

Home appliances

DAC, timers,
I²C FM+,
touch-sensing

Home appliances, motor control,
power tools

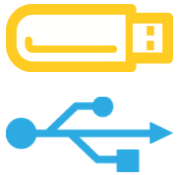
Consumer appliances

USB charging class compatible

A/V receivers, TVs,
Blu-ray disk players

Printers

Gaming



USB RF dongle

Remote control

Cost competitive CEC, DAC

Great Fit for Appliances

Easy communication between front panel and power components with robust **I²C FM+ with 20mA sink** capability and **fast IO toggling** capability (25% faster than STM32F1 @ same frequency)

Advanced DIGITAL & ANALOG IPs

- 3 timers suit **induction cooking apps**
- 1 timer for **motor control** (complete reference designs avail)
- 1µs, 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

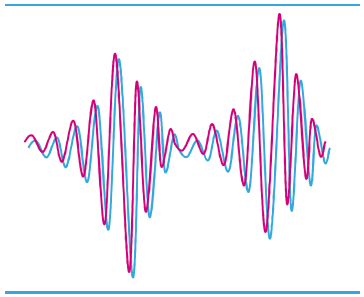
- Real-time hardware **RAM parity check** and 16-bit **CRC** for FLASH-memory integrity checks
- Extended **double watchdog system** with autonomous clock, windowing and **clock security system**

life.augmented

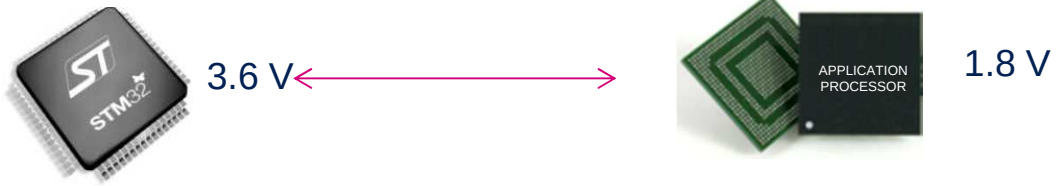
Great Fit for Consumer

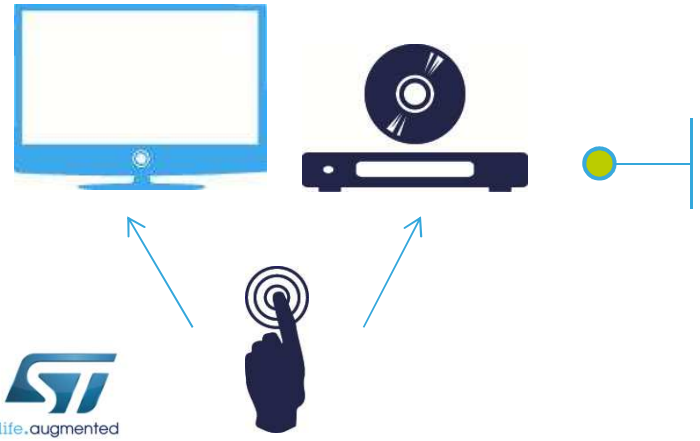
Optimized communication:

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



Easy interface with 1.8 V IC (Application processors, for example): Keeps ADC, DAC and CMP advanced **analog 3.6 V excursion** via dual-voltage domains on STM32F0





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

life.augmented

SW Libraries Speed Time-to-Market

- **Free** STM32 Standard peripheral libraries
  - C source code for easy implementation of all STM32 peripherals in any application

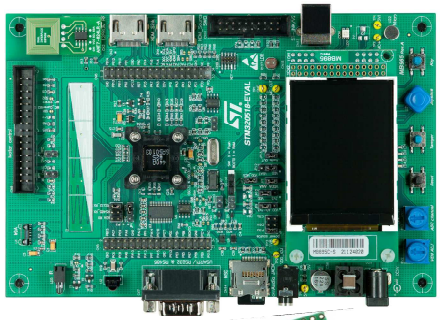
- **Free** STM32 Infra-red software
- **Free** STM32 Motor-control library
- **Free** STM32 CEC software
  - Complete software supported by the **STM320518-EVAL** evaluation board providing an implementation of CEC high-level protocol and full-demonstration software

Visit www.st.com/stm32

life.augmented

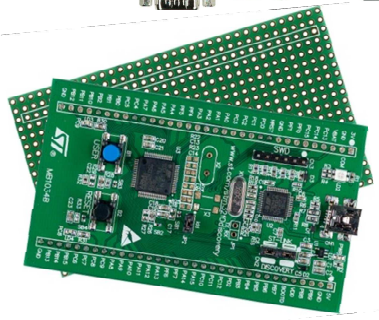
Extensive Tools and SW

- Evaluation board for full product feature evaluation
  - Hardware evaluation platform for all interfaces
  - Possible connection to all I/Os and all peripherals



STM320518-EVAL
$199

- Discovery kit for cost-effective evaluation and prototyping



STM32F0DISCOVERY
$7.99

- Large choice of development IDE solutions from the STM32 and ARM ecosystem

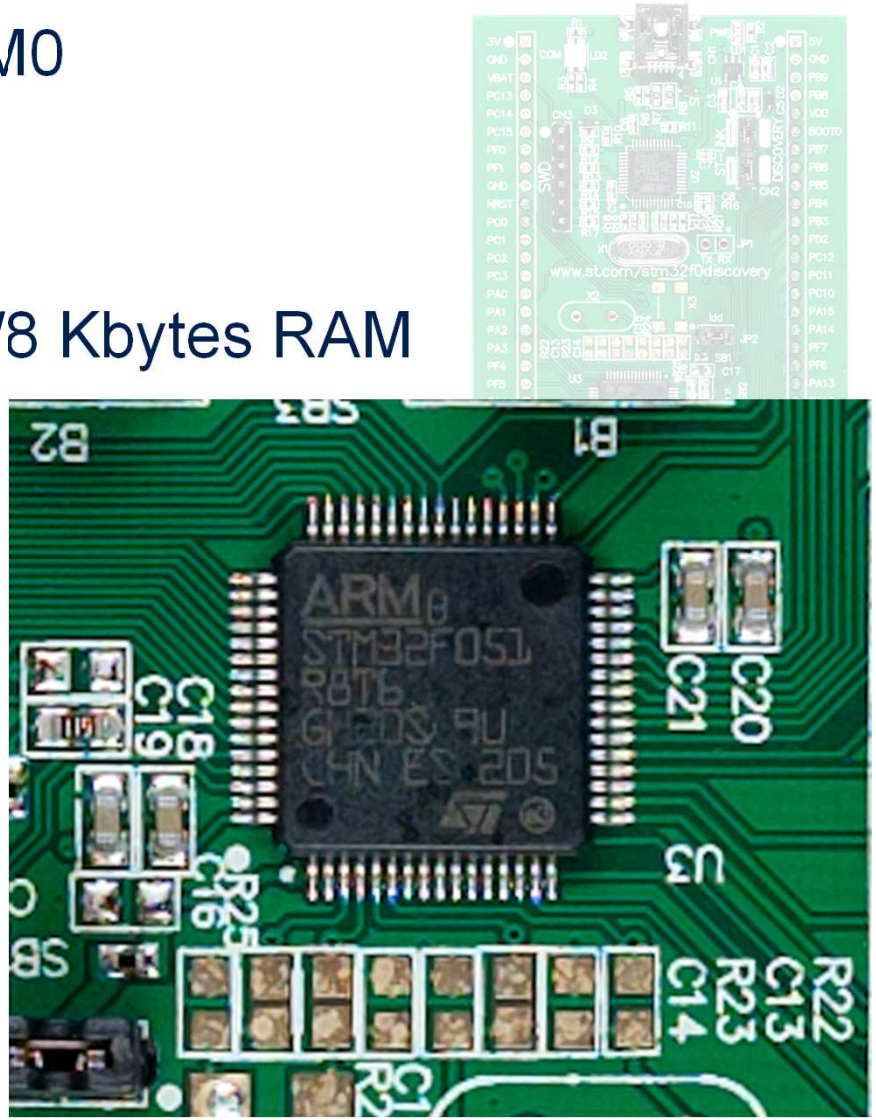
RAISONANCE Embbeded Systems Development Tools
aiji SYSTEM 아이지시스템
hitex DEVELOPMENT TOOLS
IAR SYSTEMS
atollic
iSYSTEM
Green Hills SOFTWARE, INC.
SEGGER
KEIL An ARM® Company
LAUTERBACH
SIGNUM SYSTEMS
TASKING
expresslogic
CMX SYSTEMS

STM32F0-DISCOVERY

life.augmented

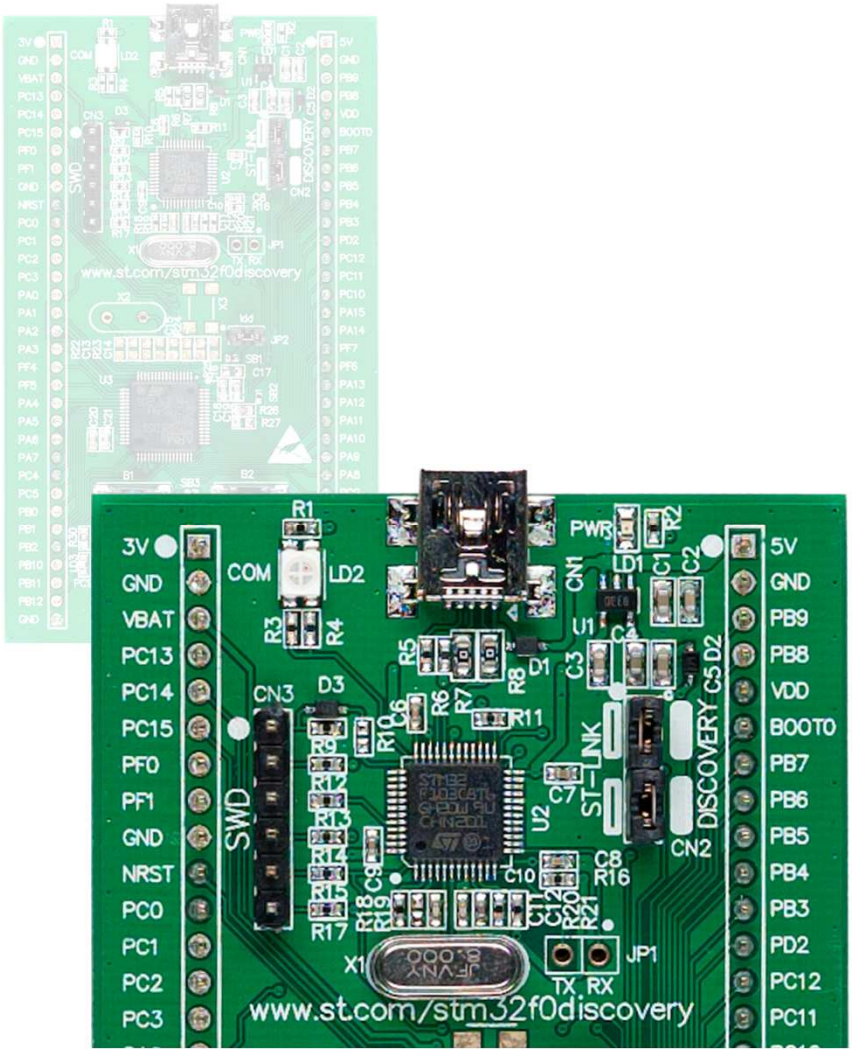
STM32F051R8T6

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



Embedded ST-Link

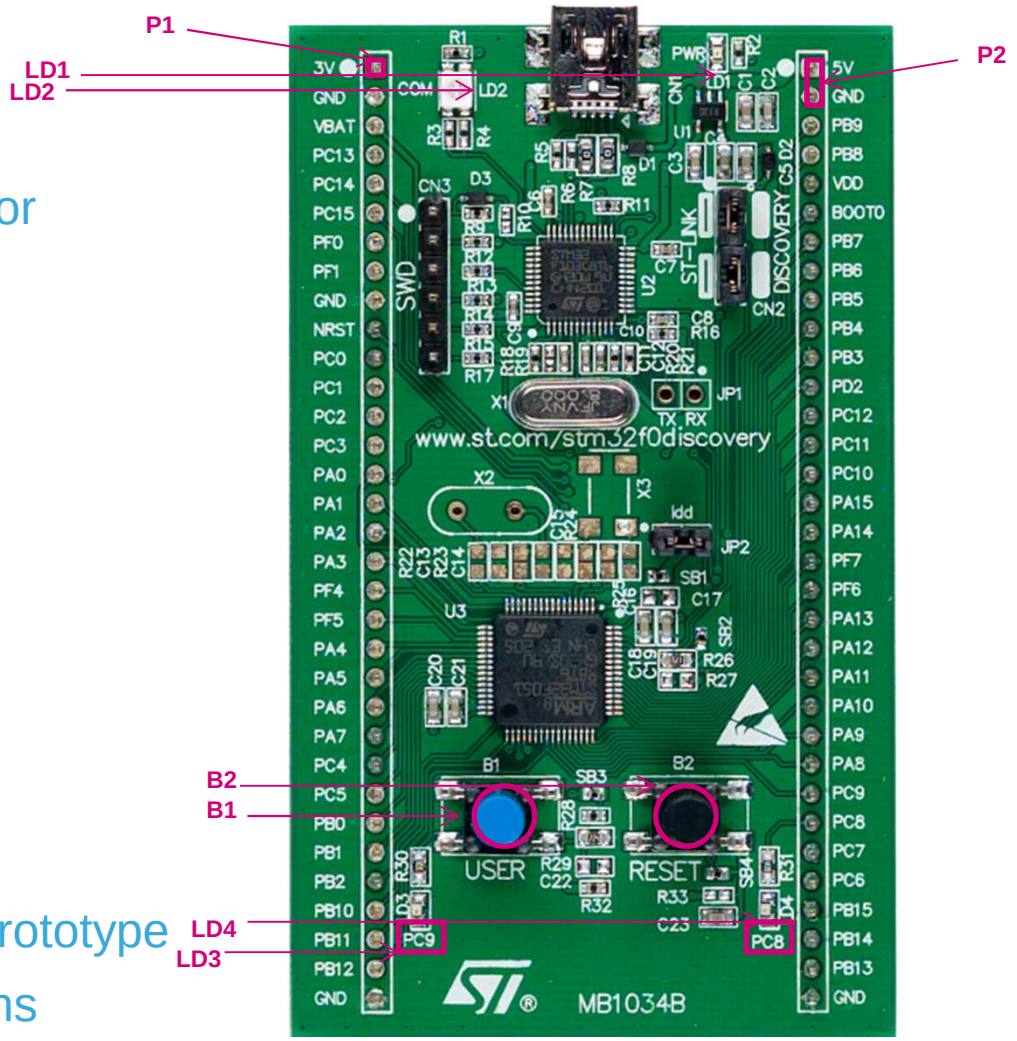
- ST-Link programming and debugging tool integrated on-board the kit (STM32F103C8T6)
- Can be used two different ways
  - Program and debug the MCU on the board
  - Program an MCU on another application board
- Features
  - USB Connector
  - ST-LINK MCU
  - 5 to 3V Power regulator
  - CN2 – MCU Program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



life.augmented

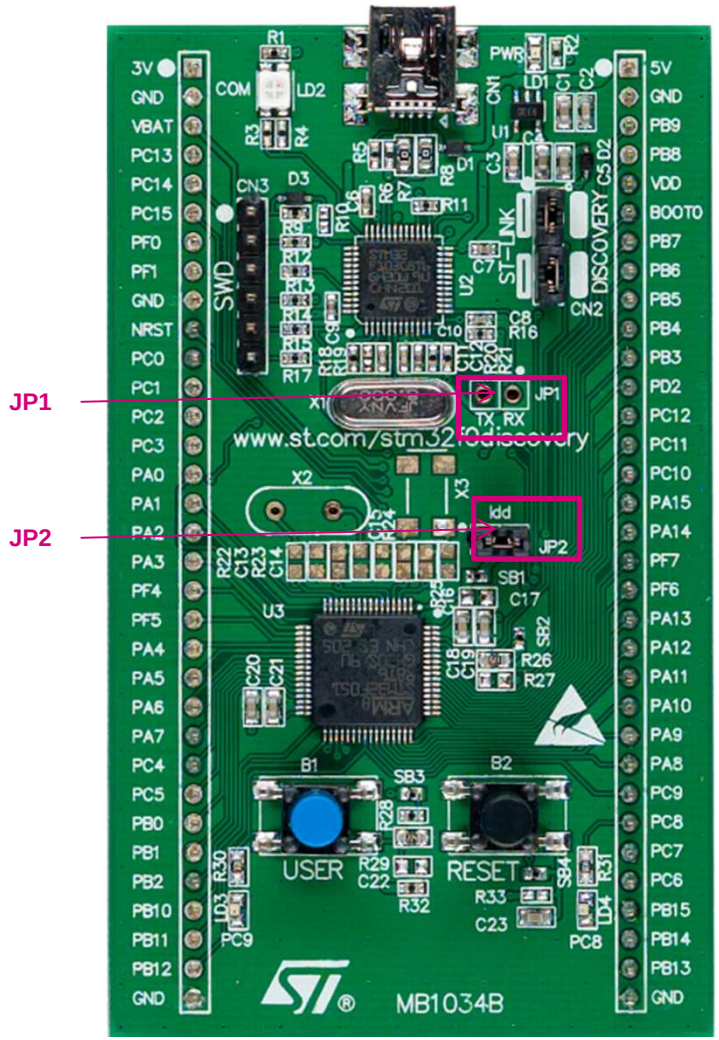
LEDs/Push-Buttons/Extension Connector

- LEDS
  - LD1: Power indicator
  - LD2: ST-LINK Comm 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)
  - JP2: I_{DD} for MCU current measurement (fitted by default)
- Documentation
  - UM1523 Getting started with software and firmware environments
  - UM1525 STM32F0Discovery STM32F0 discovery kit
- Firmware Library
  - Contains STM32F0 Standard Firmware Library
  - Contains example code
  - AN4062 peripheral firmware examples



life.augmented

Step #3 - Install ST-Link Driver

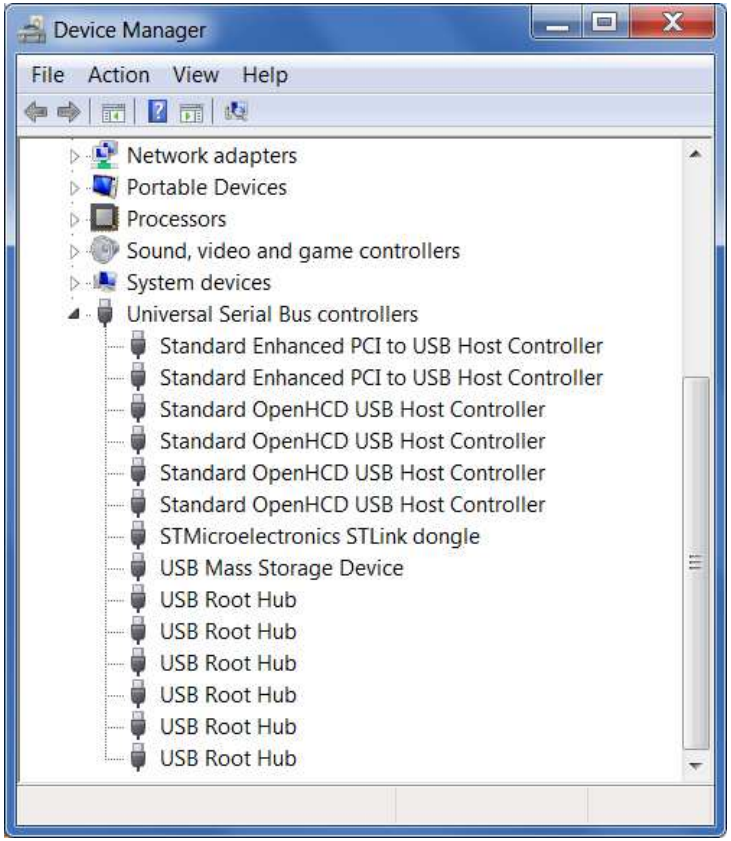
- The STM32F0DISCOVERY board includes and ST-LINK/V2 embedded programming and debug tool
- Download and install the ST-Link/V2 USB driver
  - http://www.st.com/web/en/catalog/tools/PF258167
- Double-click on the file: ST-Link_V2_USBDriver.exe to install
- Click through the installation menu until the driver installation is complete

Step #4: Connect the Discovery Kit/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

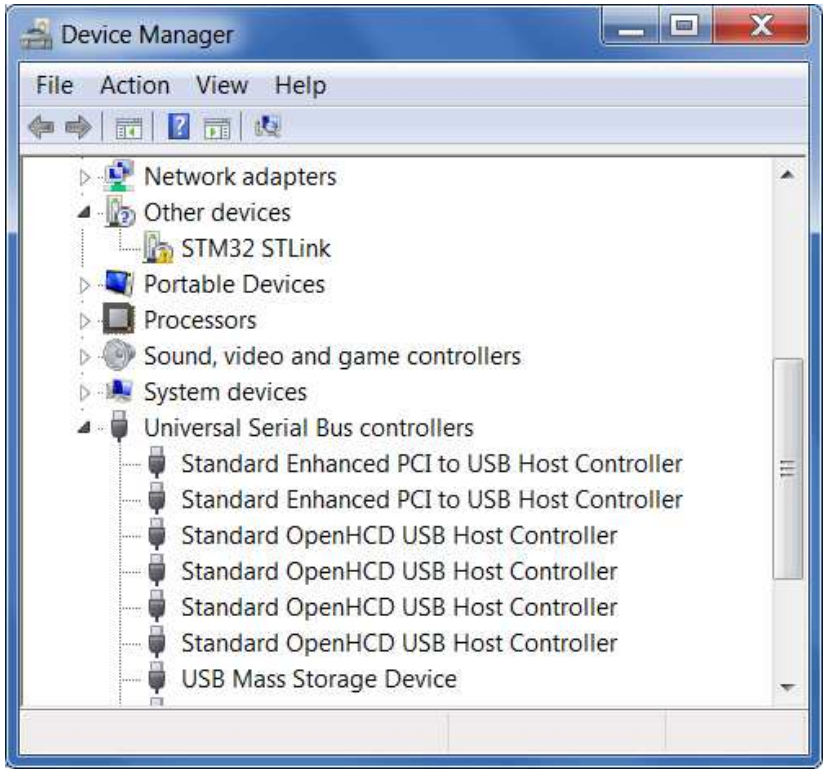
- Wait for Windows to recognize the ST-Link device and follow any step required to install the driver

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

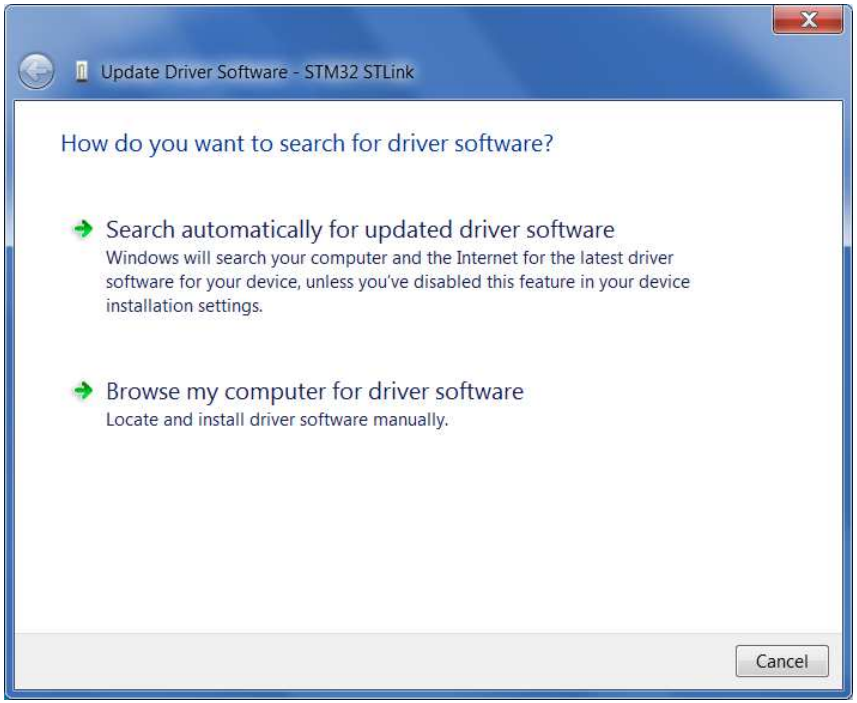


Step #4
ST-Link Driver Trouble Shooting

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”

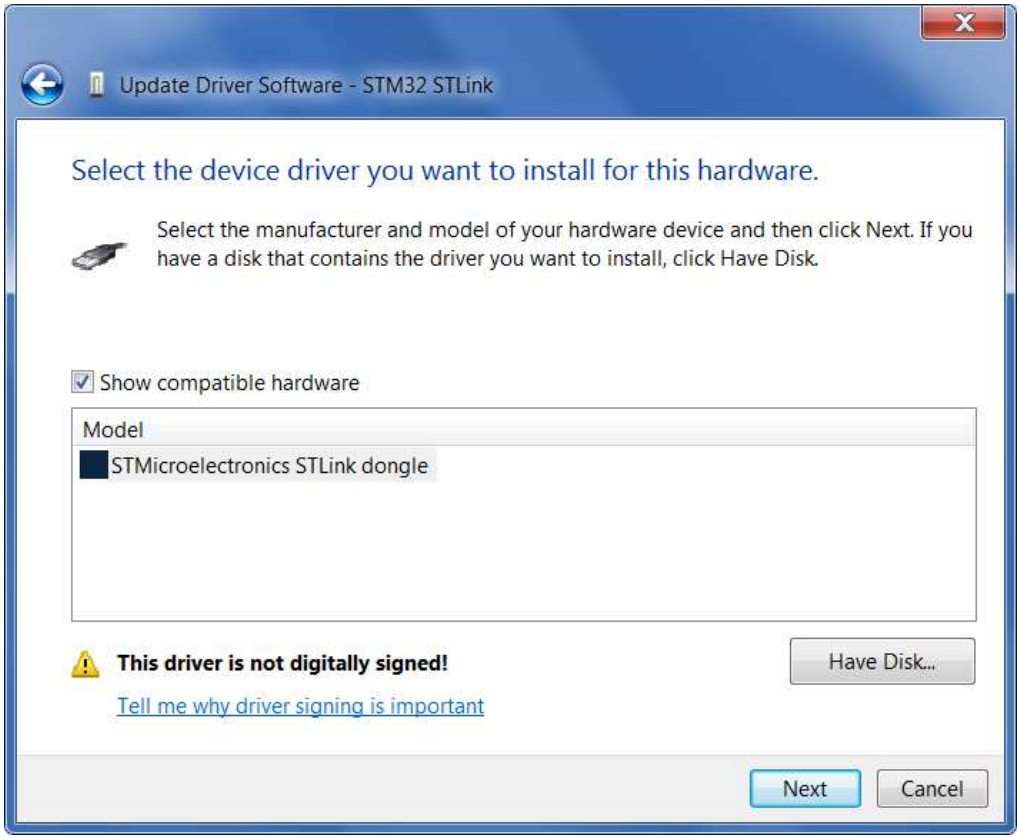
4. Select “Browse my computer for driver software”

life.augmented

Step #4
ST-Link Driver Trouble Shooting

- The “STMicroelectronics ST-Link dongle” should listed

7. Click “Next”



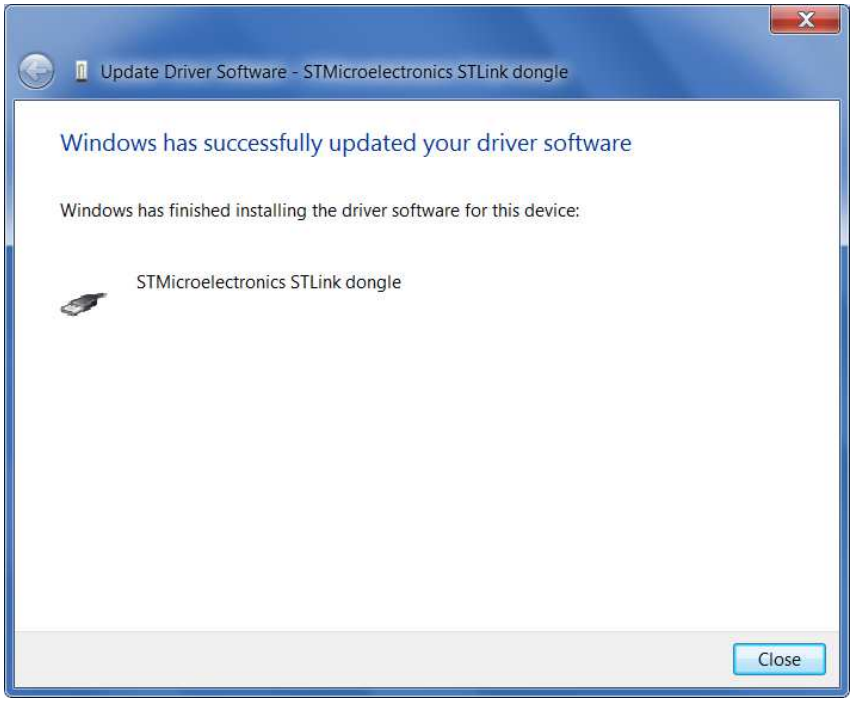
Step #4
ST-Link Driver Trouble Shooting

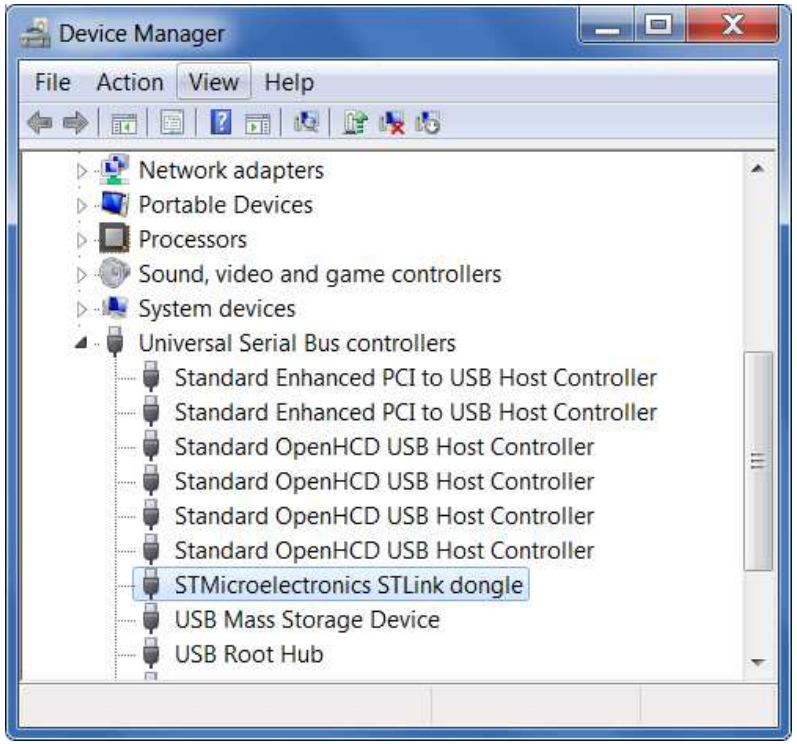
- A warning message may appear

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

life.augmented

STM32 F0 Tools
Documentation overview

life.augmented

Documentation

- Main MCU web page: www.st.com/mcu
- STM32 web page: www.st.com/stm32
- STM32F0 specific web page: www.st.com/stm32f0
- Discovery board web page:
  - From www.st.com/stm32f0 click on Resources -> Hardware -> STM32 MCU Eval Boards. Then select Discovery kit
  - The web page includes:
    - User Manual
    - Getting Started Manual
    - Schematics and Gerber files
    - Discovery Board Firmware Package

Documentation resources

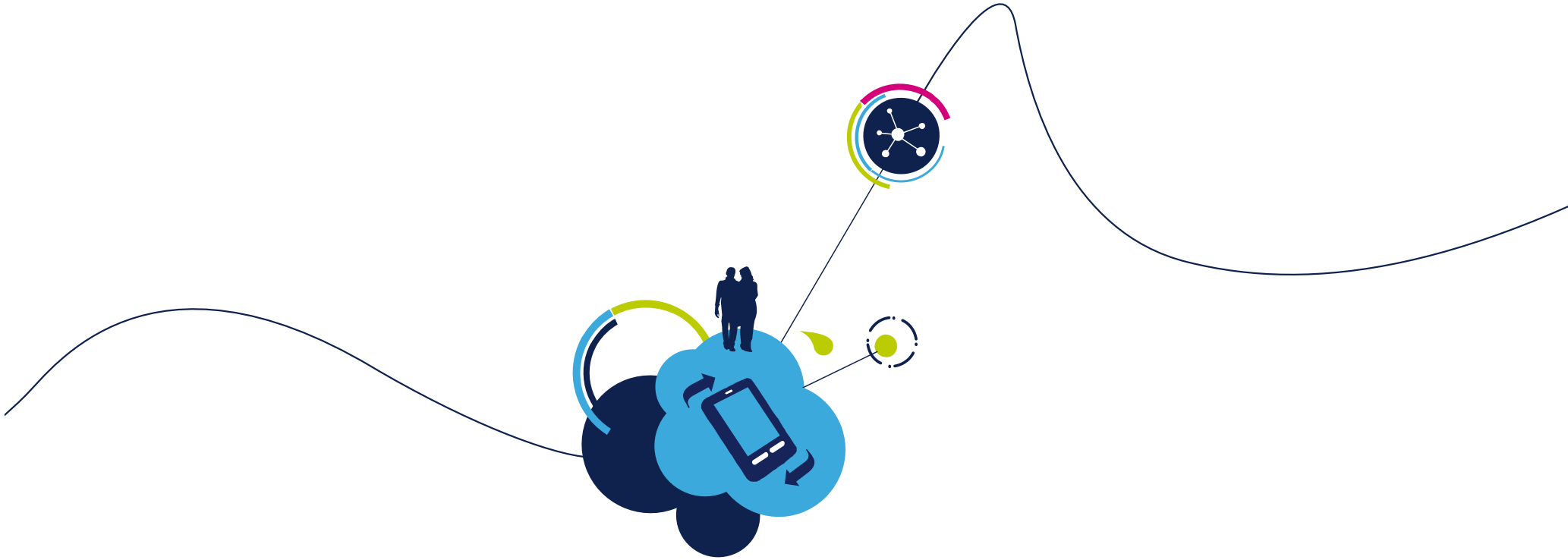
- Main website page for the STM32 F0 Series
  - www.st.com/stm32f0
- You can find
  - Datasheets
  - Applications Notes
  - Errata
  - Technical Notes
  - Programming Manuals
  - Reference Manual
  - User Manuals
  - Firmware
- For all STM32 related products: www.st.com/stm32

Support resources

- Submit technical questions to ST Online Support:
  - Located on main www.st.com page under the Support tab – Online Support
- ST Public Forums:
  - Located on main www.st.com page under Support tab – ST e2e Communities

Process check

- ST-Link is recognized by your system
- LD1 and LD2 should on ON (indicating board power available and 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 slowly
  - Press the USER Button again
    - LD4 (Blue) should flash once
    - LD3 (Green) will shut off



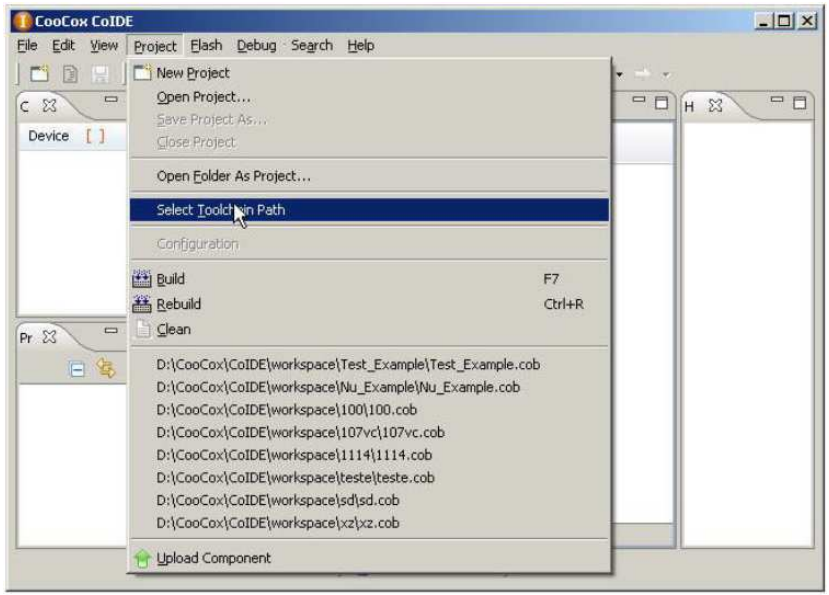
Compile, Debug and Run

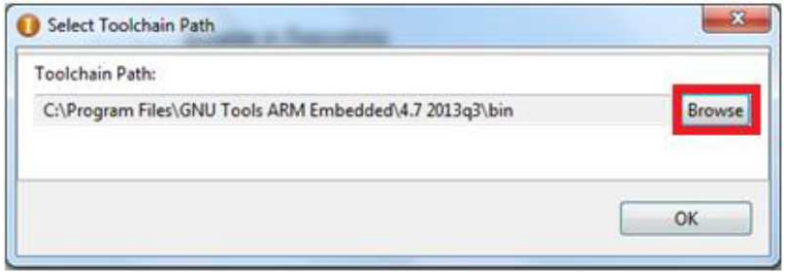
life.augmented

Coocox CoIDE Preparation

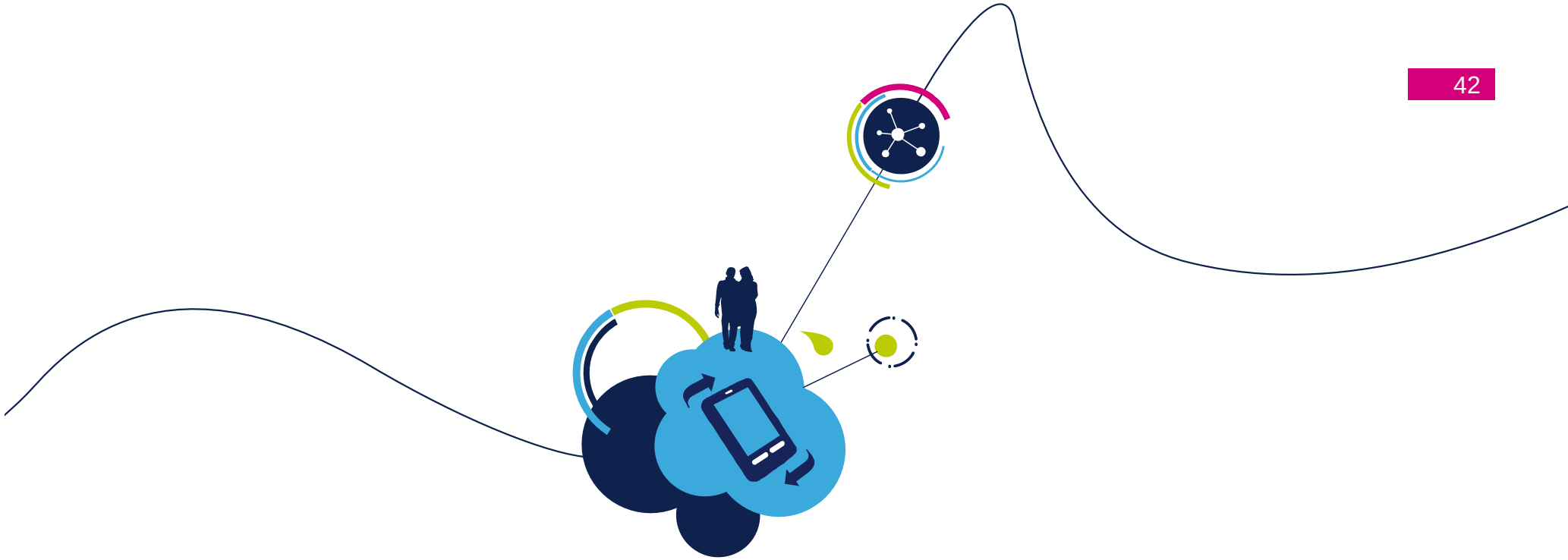
Preparation – Installing CoIDE and Drivers

- Download and install the latest CoIDE software development environment found at http://www.coocox.org/Tools/CoIDE-1.7.5.1.exe
- Configure GCC tool chain for CoIDE.
  - Download and install ARM GCC can be downloaded at
    - https://launchpad.net/gcc-arm-embedded/+download
  - Follow below steps to configure GCC tool chain.
    - After launching CoIDE, click "Select Toolchain Path" under the Project menu.
    - Click the “Browse” button; select the folder that contains the arm-none-eabi-gcc.exe and the other GCC execute files (You can find the folder where you install the ARM GCC tool chain). And Click “OK” button to save the setting.
  - Download and install ST-LINK/V2 USB driver found at http://www.st.com/web/en/catalog/tools/PF258167

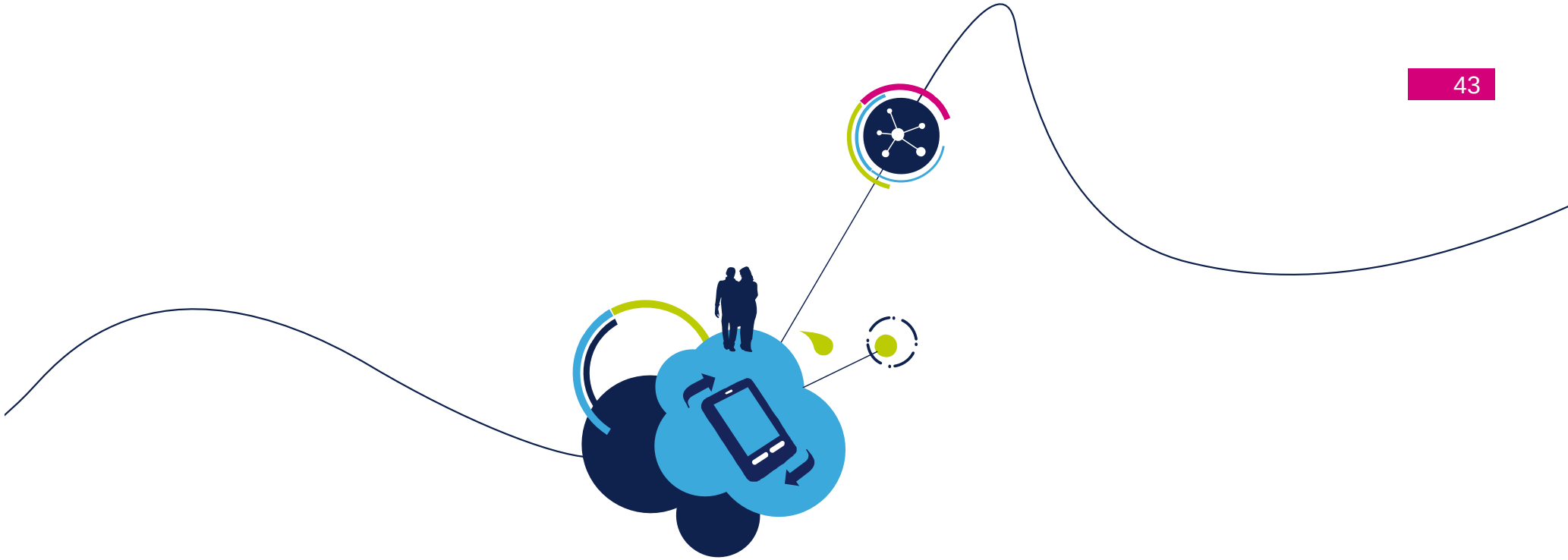




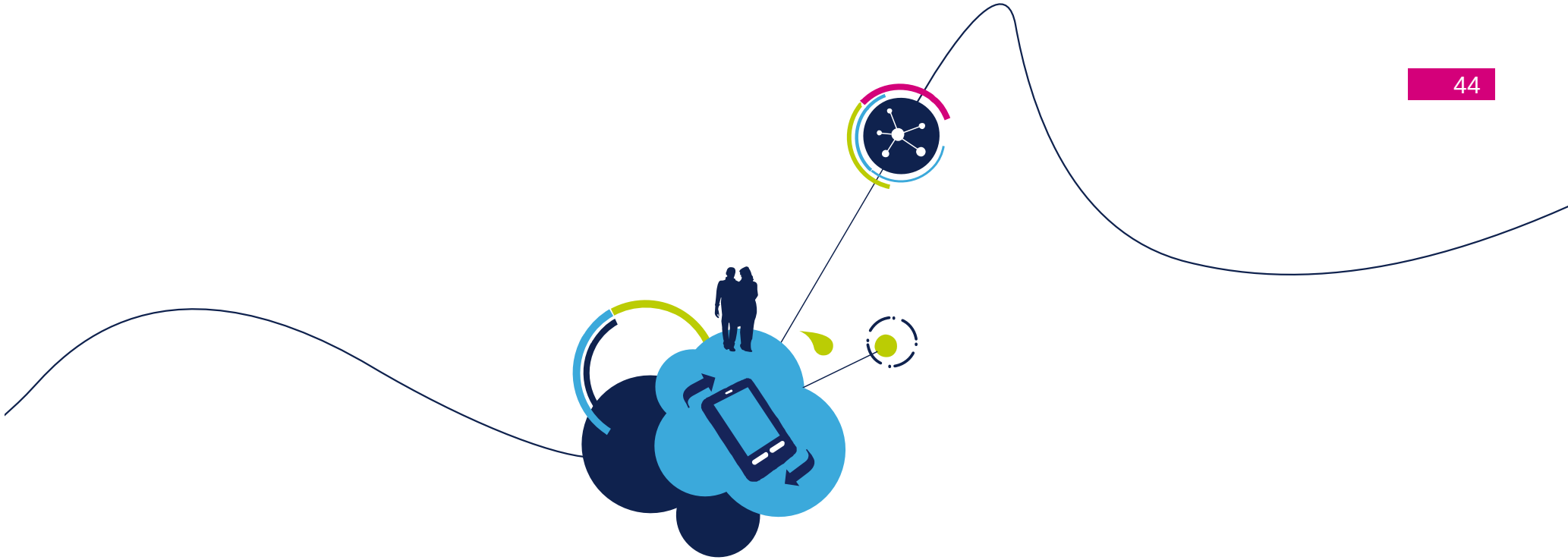
life.augmented



Demo to “Create a CoIDE Project by Chip Mode”



Demo to “Create a CoIDE Project by Board Mode”

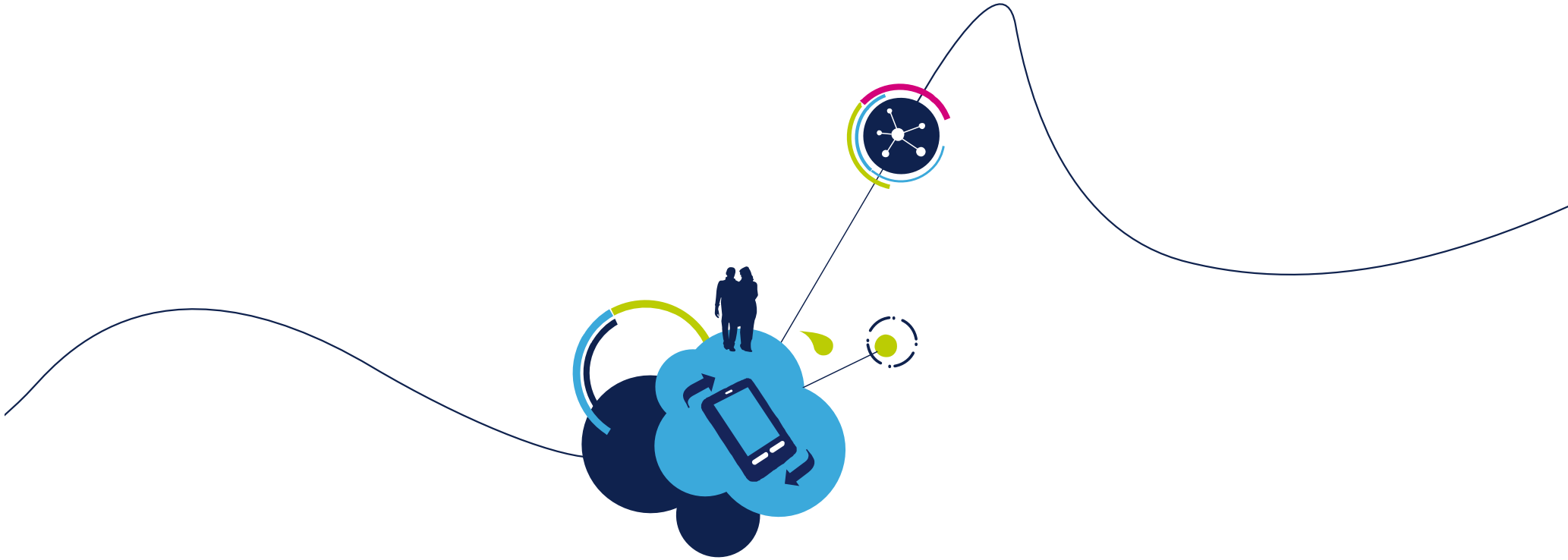


Demo to “Create a CoIDE Project using Discovery kit firmware package” and debug

life.augmented

Create Project using ST Firmware Package

- Objective:
  - Run codes on the STM32F0 Discovery board using sample codes included in the STM32F0 Discovery firmware package
- Procedure:
  - Refer to CoIDE Demo document
  - Go to the STM32F0 Discovery website
    - http://www.st.com/web/catalog/tools/FM116/SC959/SS1532/LN1199/PF252994
  - Download the STM32F0 discovery firmware package and unzip the files
  - Choose a particular example you want to try out from "***\STM32F0-Discovery_FW_V1.0.0\Project\Peripheral_Examples***" folder.
    - For example "***Systick***"
  - Read the readme file for details regarding the example.
  - Launch Coocox IDE and create and configure a workspace
  - Add the following files into the environment.
    - Example files (main.c, stm32f0xx._it.c, system_stm32f0xx.c, etc)
    - Required Library files
    - Utility files from "***\STM32F0-Discovery_FW_V1.0.\Utilities\STM32F0-Discovery***" folder (stm32f0_discovery.c, stm32f0_discovery.h)

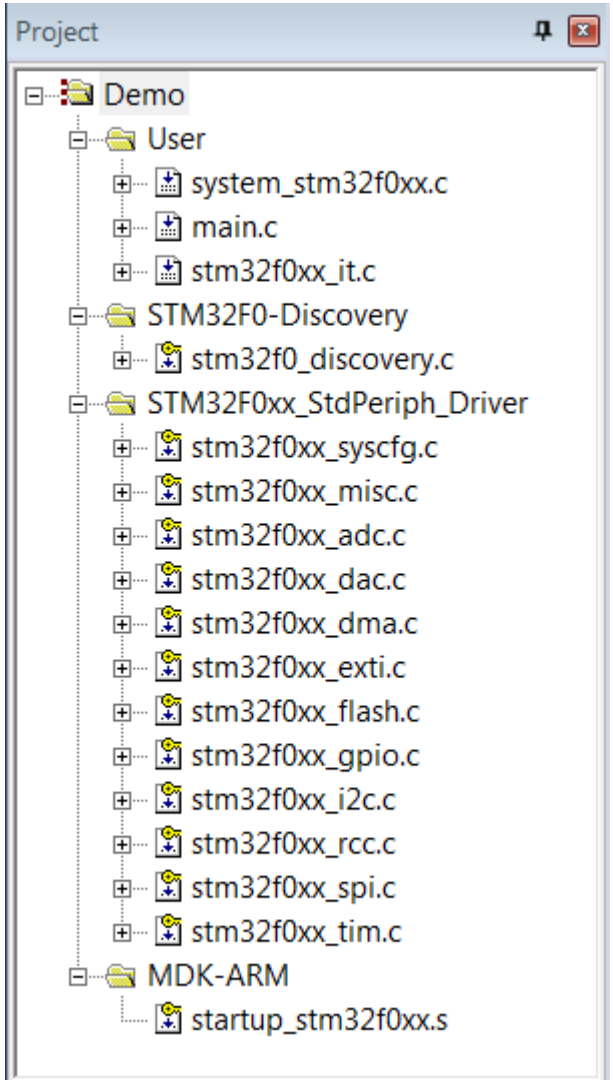


Firmware Project Overview

life.augmented

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) {};

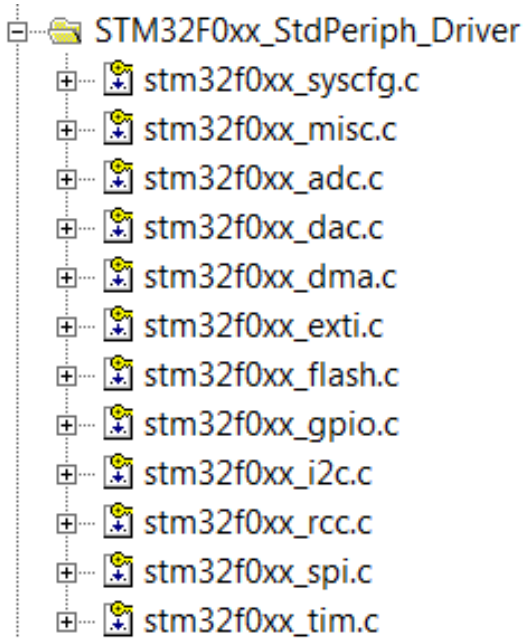
stm32f0-discovery.c

- 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

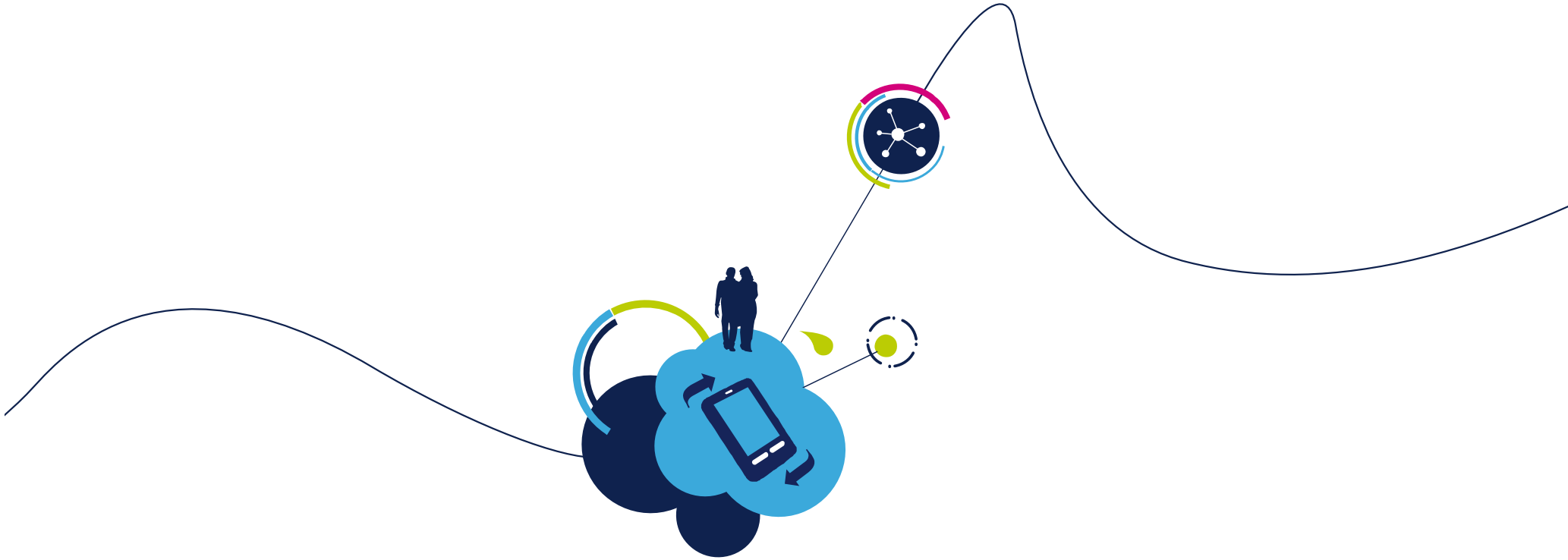
- Each file contains library functions that can be used for each peripheral
- Abstracts register manipulation and gives a standard API for access to peripheral functions
- Example:




```
/**
  * @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;
}
```

life.augmented

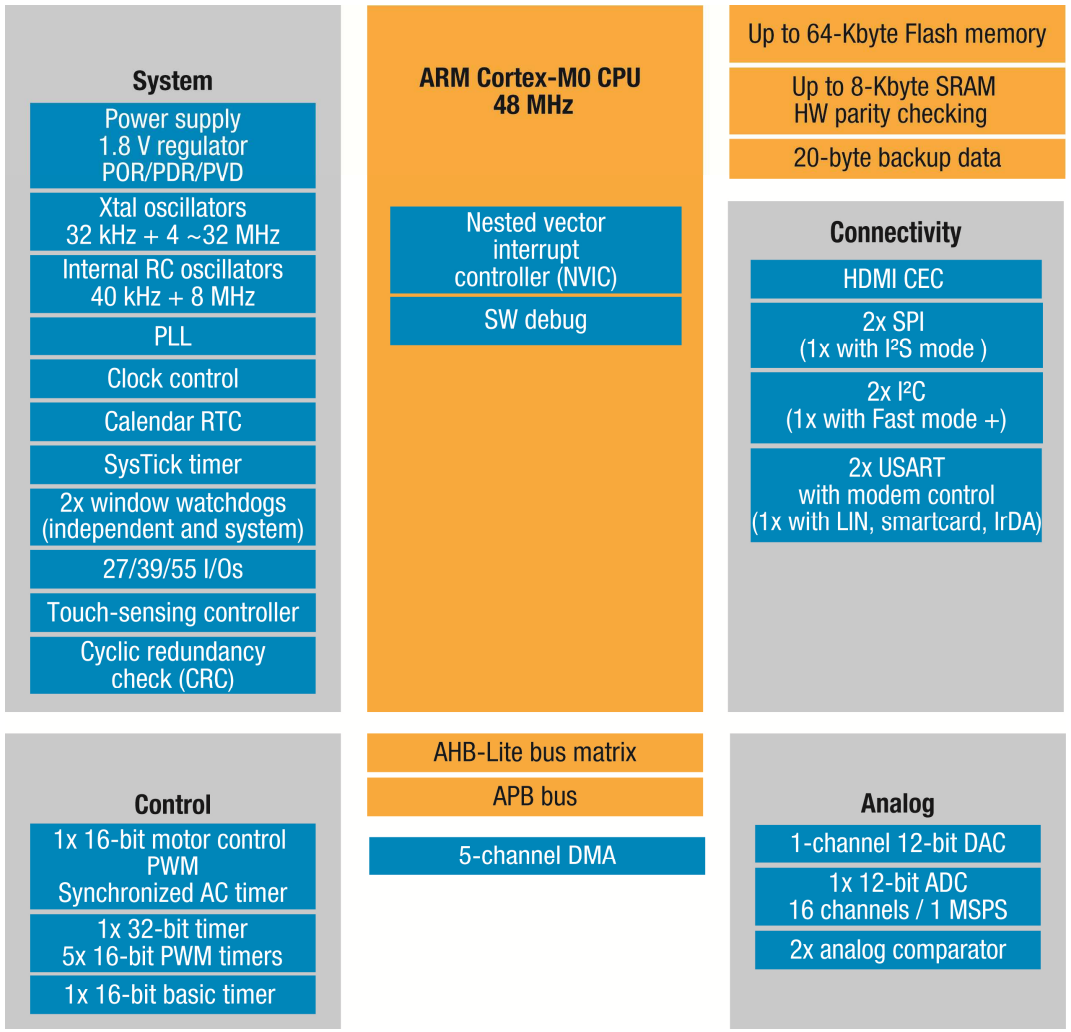


The STM32 F0 Series in Detail

life.augmented

STM32 F0 Series 64KB STM32F051

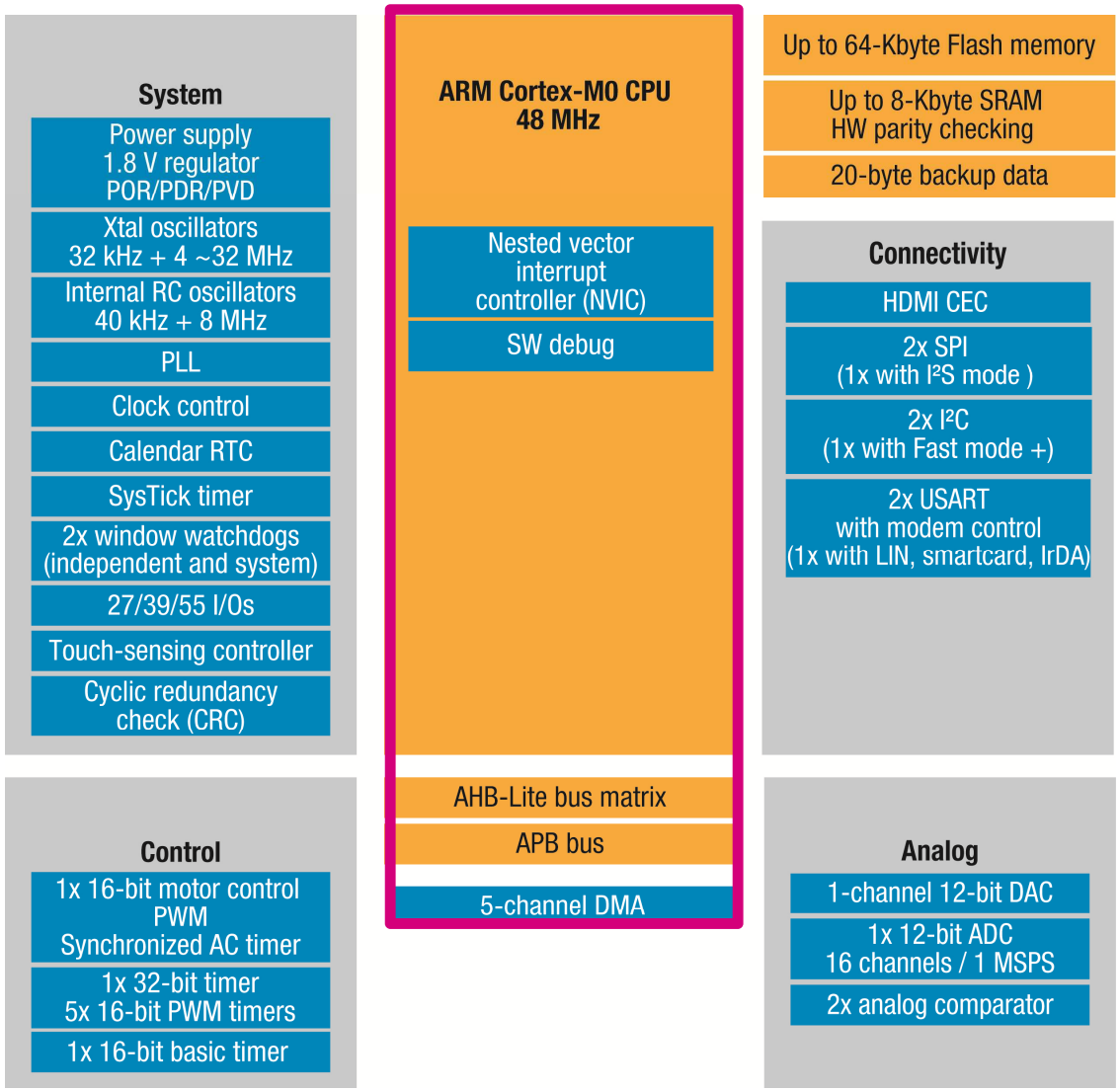
- **ARM 32-bit Cortex-M0 CPU**
- **Operating Voltage:**
  - VDD = 2.0 V to 3.6 V
  - VBAT = 1.6 V to 3.6 V
- **Safe Reset System (Integrated Power On Reset (POR)/Power Down Reset (PDR) + Programmable voltage detector (PVD))**
- **Embedded Memories:**
  - FLASH: up 64 Kbytes
  - SRAM: up 8 Kbytes
- **CRC calculation unit**
- **5 Channels DMA**
- **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

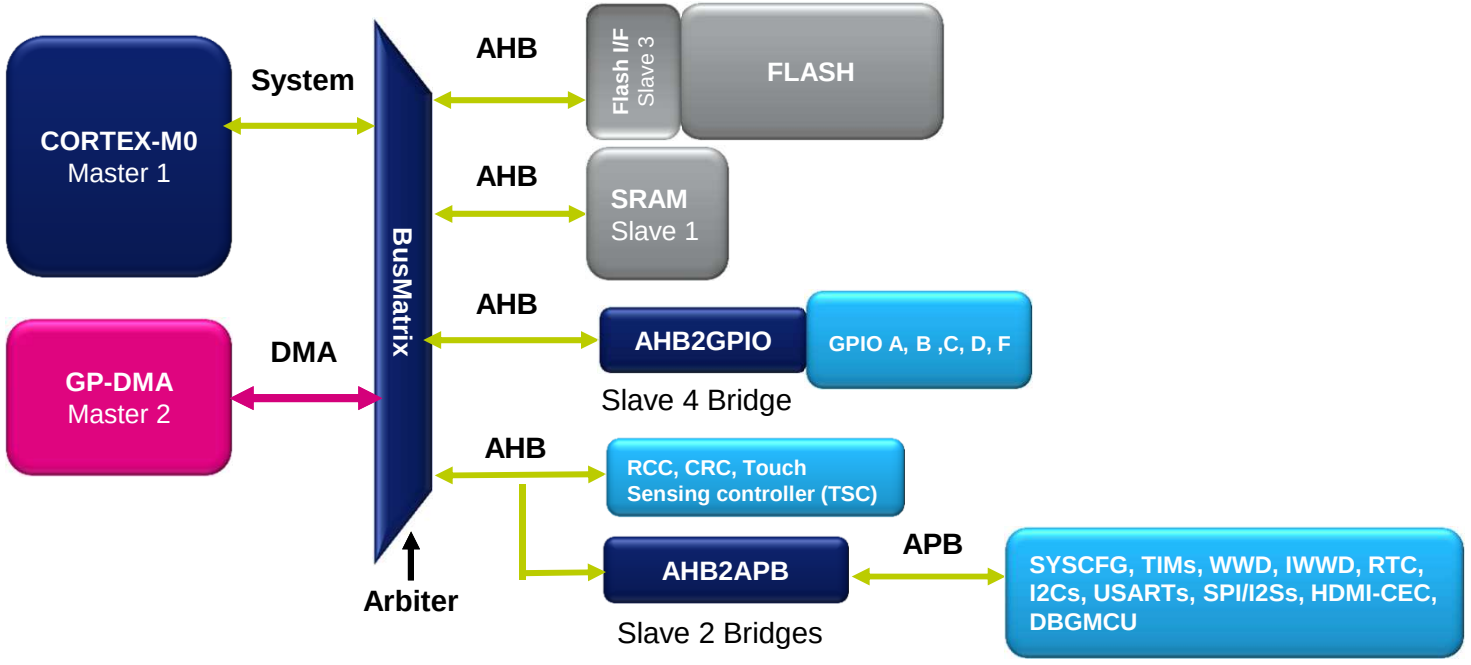
- **ARM 32-bit Cortex-M0 CPU**
- **Nested Vector Interrupt Controller (NVIC)**
- **SWD**
- **Bus Matrix**
- **5 Channels DMA**



life.augmented

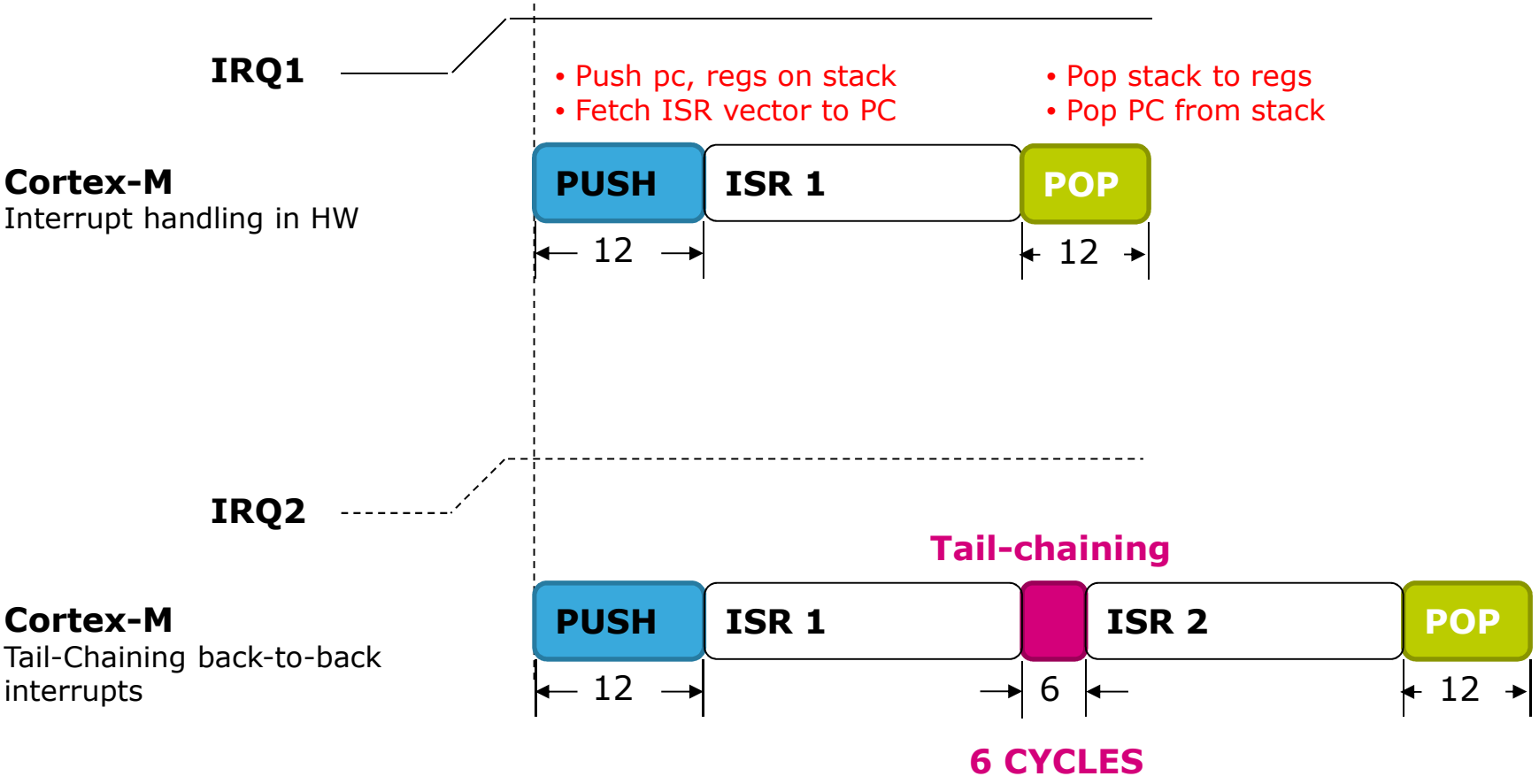
System architecture/DMA

- Cortex-M0 and DMA share the AHB bus matrix to allow parallel access
  - Multiply 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



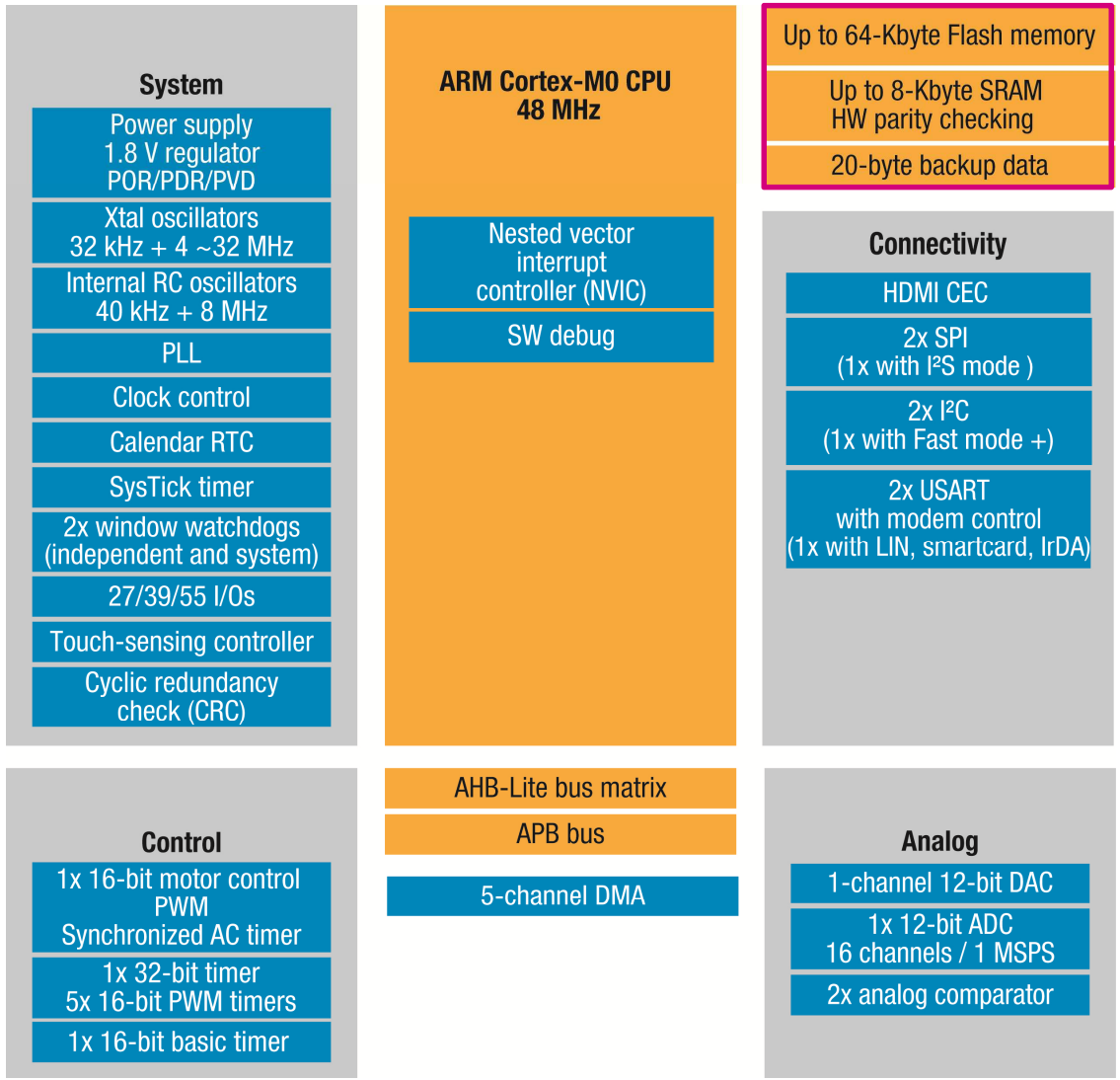
Cortex-M0 Exception Types

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

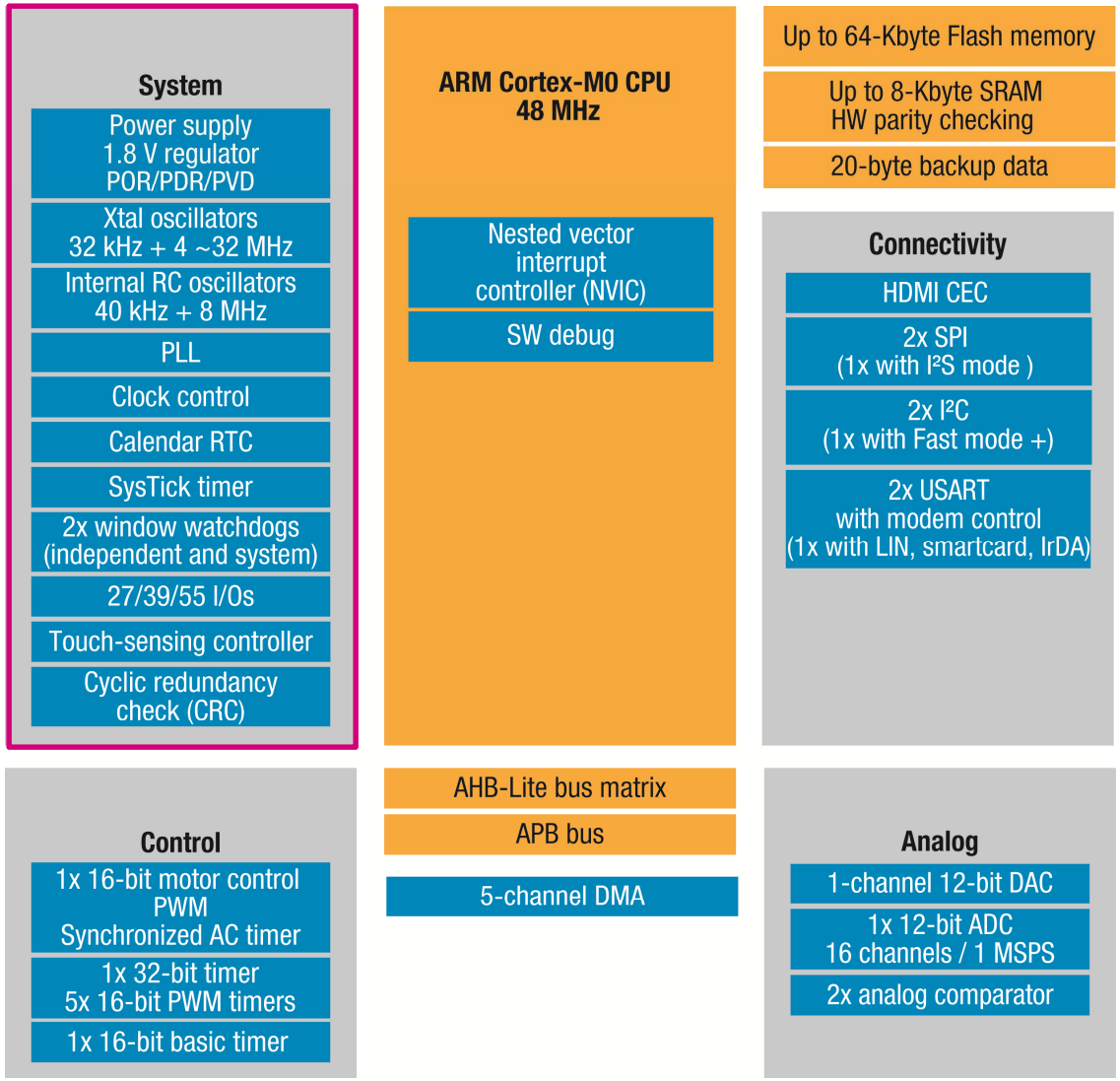
Memories: Flash, SRAM, Back-up Data Registers

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



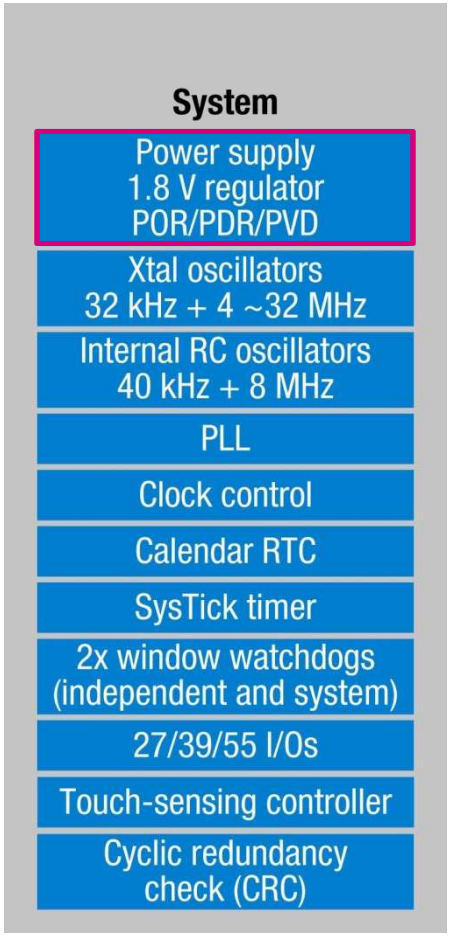
life.augmented

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:**
  - VDD = 2.0 V to 3.6 V
  - VDDA = 2.0 V to 3.6 V
  - VBAT = 1.6 V 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

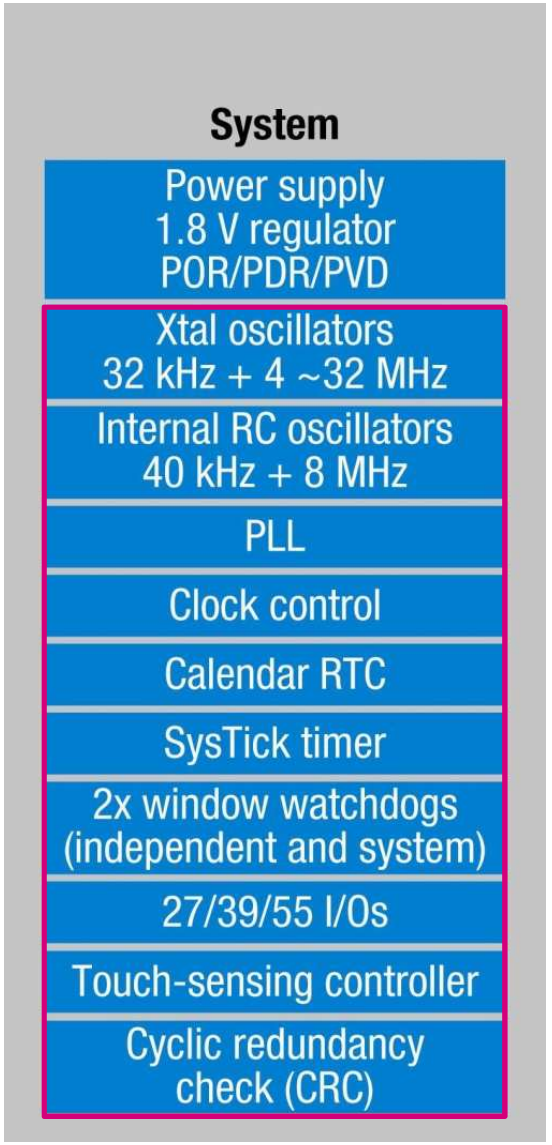
- STM32F05x Low Power modes: uses Cortex-M0 Sleep modes
  - SLEEP, STOP and STANDBY

Feature	STM32F05x typ IDD/IDDA (*)
RUN mode w/ execute from **Flash** on **48MHz** (HSE bypass 8MHz x 6 PLL = 48MHz) All peripherals clock ON	**22.9 / 0.166 (mA)**
RUN mode w/ execute from **Flash** on **24MHz** (HSE bypass 8MHz x 3 PLL = 24MHz) All peripherals clock ON	**11.7 / 0.088 (mA)**
RUN mode w/ execute from **Flash** on **8MHz** (HSI) All peripherals clock ON	**4.15 / 0.079 (mA)**
Sleep mode w/ execute from **Flash** at **48MHz** (HSI 8MHz / 2 x 12 PLL = 48MHz) All peripherals clock ON	**12.9 / 0.243 (mA)**
STOP w/ Voltage Regulator in low power All oscillators OFF, PDR on VDDA is OFF	**3.6 / 1.34 (µA)**
STANDBY w/ LSI and IWDG OFF PDR on VDDA is OFF	**1.1 / 1.21 (µA)**

Typical values are measured at TA = 25 °C, V_{DD} =3.3 V V_{DDA}= 3.3 V.

On-chip oscillators

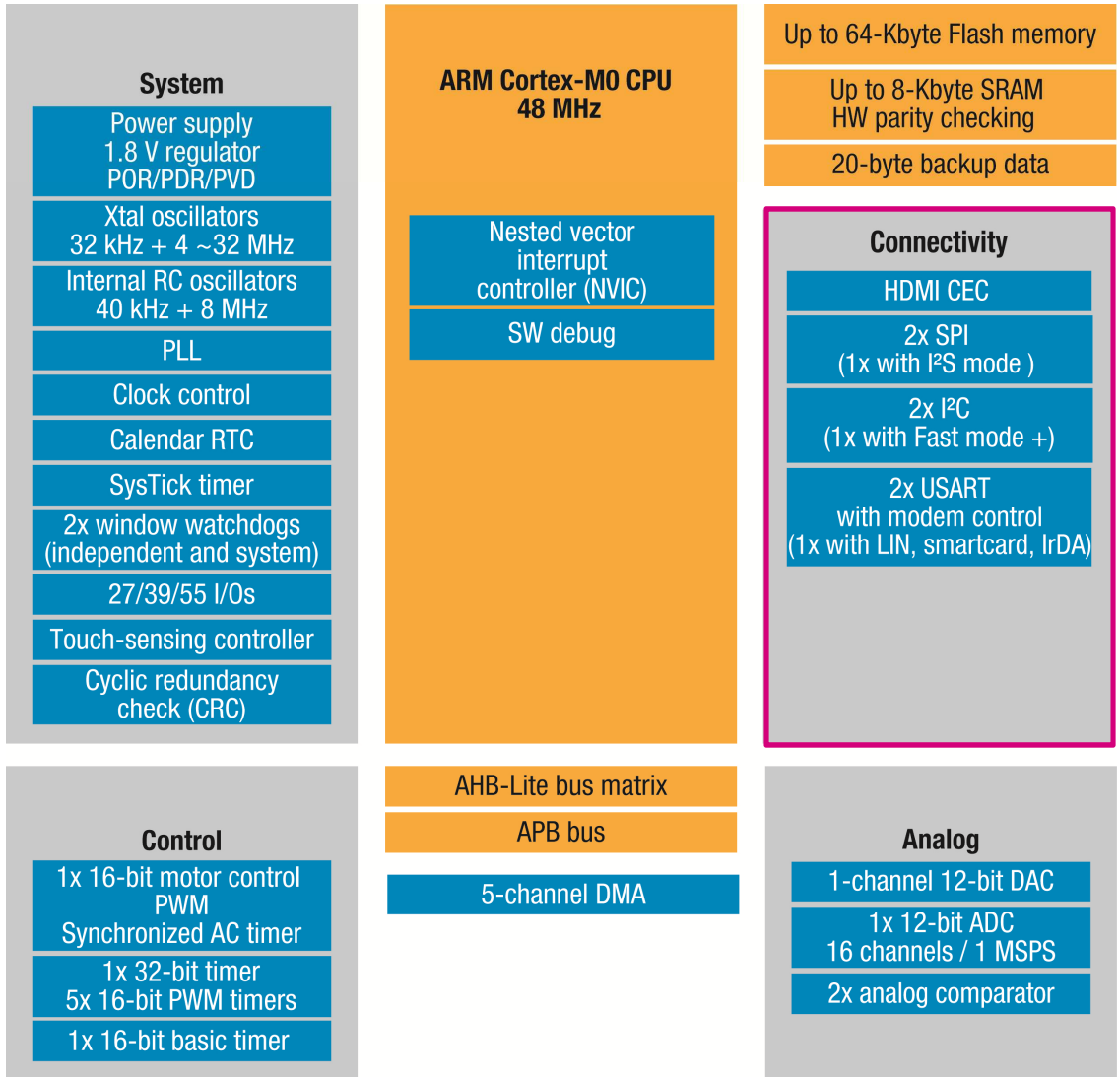
- Up to 48 MHz frequency managed & monitored by the Clock Control w/ Clock Security System
  - External Oscillators 32kHz (RTC) and 4-32MHz (System Clock)
  - Internal RC Oscillators 40kHz (IWDG) and 8MHz
  - PLL
- Low power calendar RTC
- Dual Watchdog Architecture
- Up to 55 fast I/Os all mapable on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



life.augmented

Connectivity

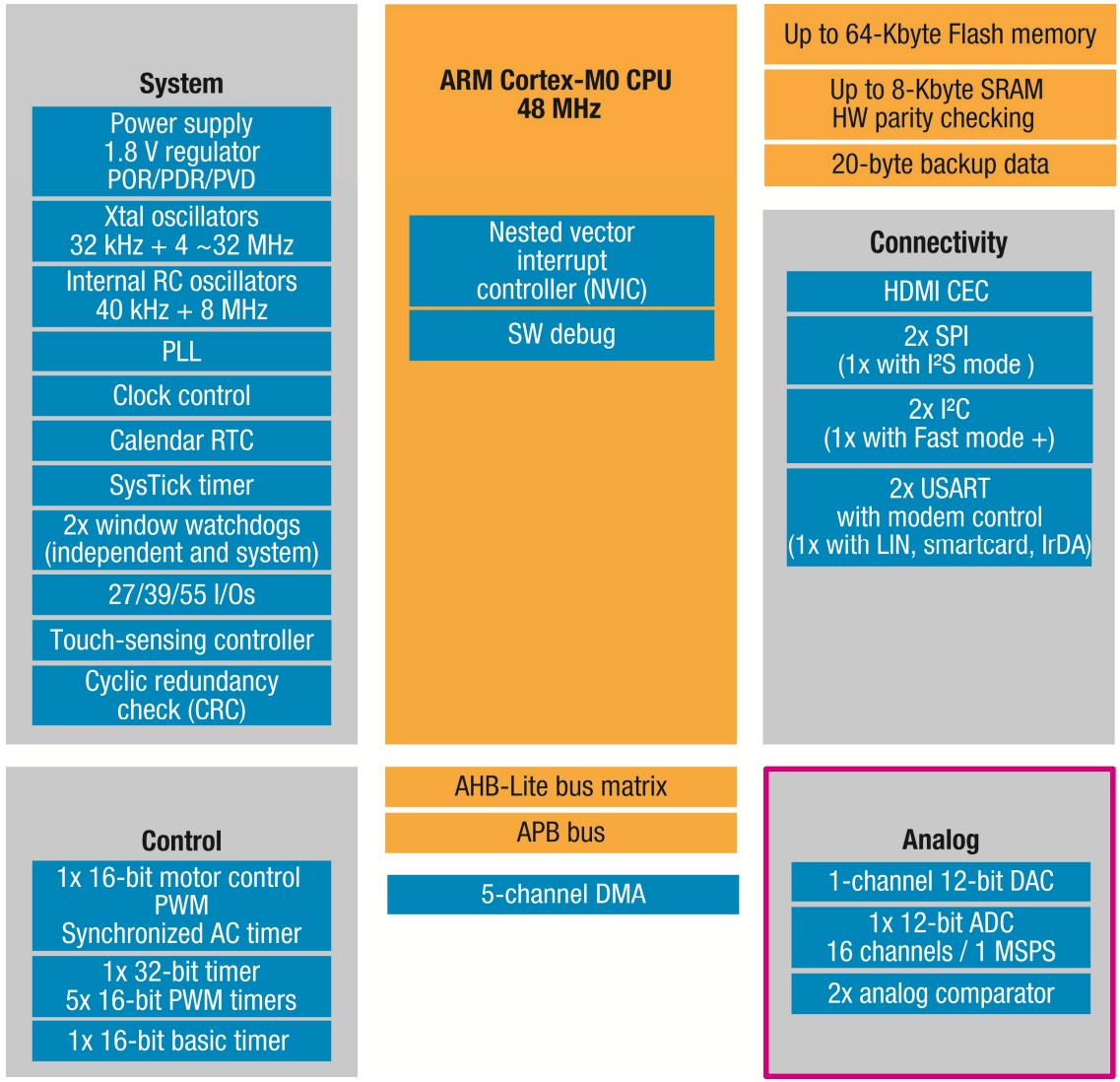
- 7 communications Interfaces



life.augmented

Analog

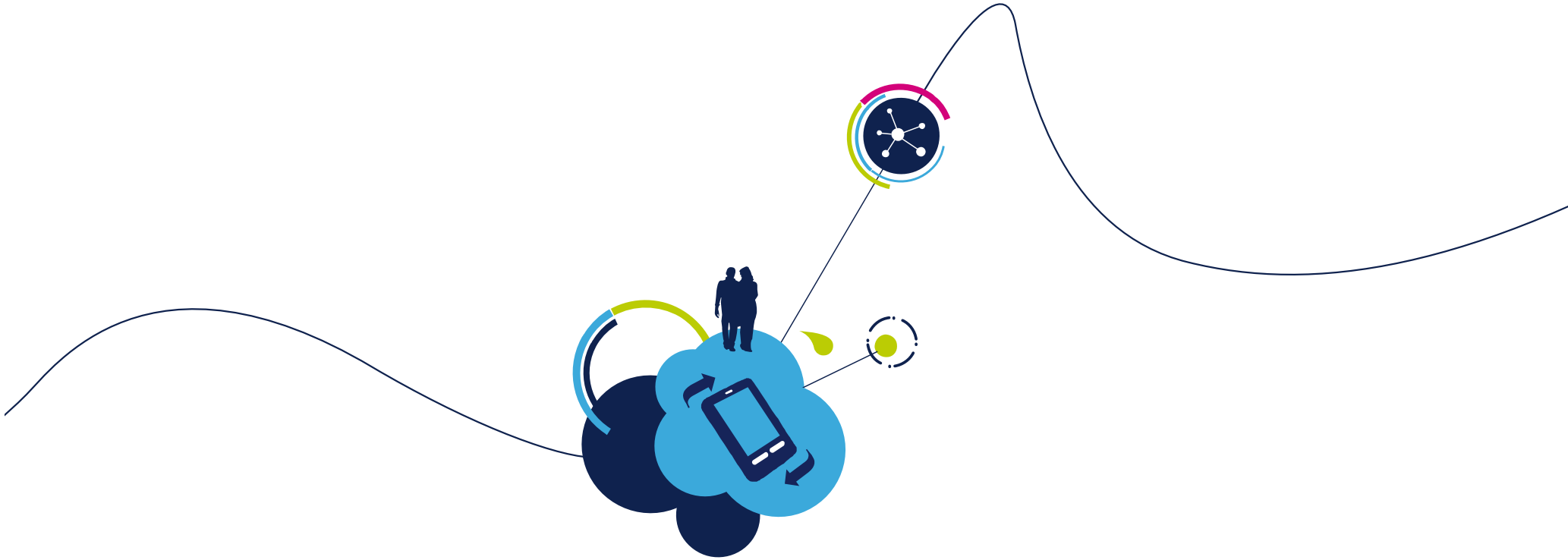
- 1 × 12-bit DAC with output buffer
- 1x12-bits 1Msps ADC w/ up to 16 external channels + Temperature sensor/ voltage reference/VBAT measurement
- 2 low power comparators (Window mode and wakeup)



life.augmented

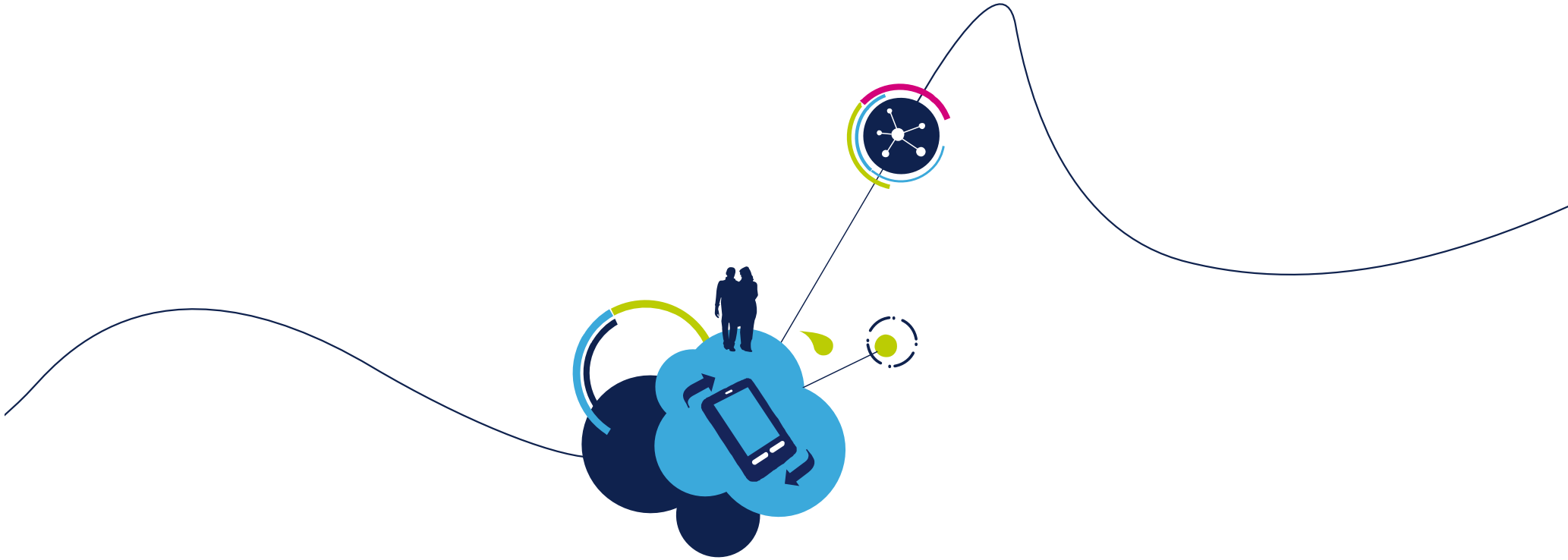
Recap of action steps

1. Obtain an STM32F0-Discovery kit (or other)
2. Download documentation and firmware library from www.st.com
3. Select, download, and install an IDE
4. Install the ST-Link Driver
5. Connect your discovery kit to your PC with USB cable
6. Select and open a firmware library project
7. Compile
8. Debug
9. Run



Questions and Answers

life.augmented

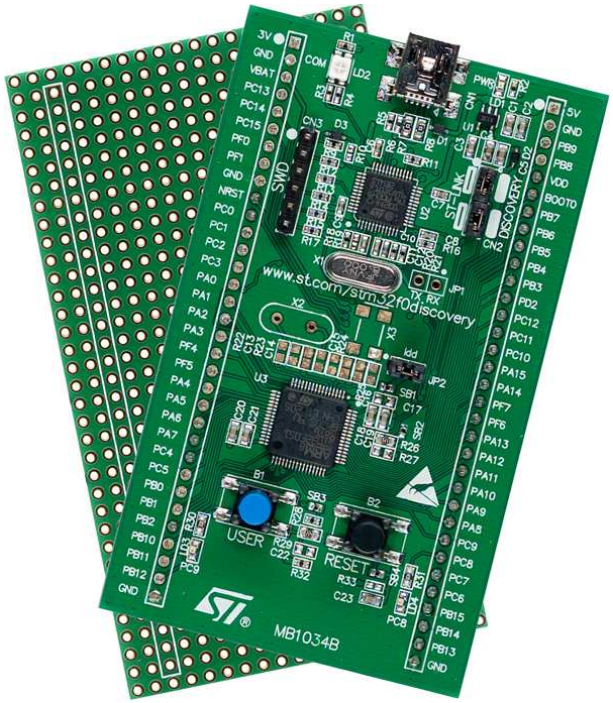


Tips and Tricks:
Using the Firmware Library

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) and/or Remap(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



Thank you

www.st.com/stm32f0discovery

life.augmented