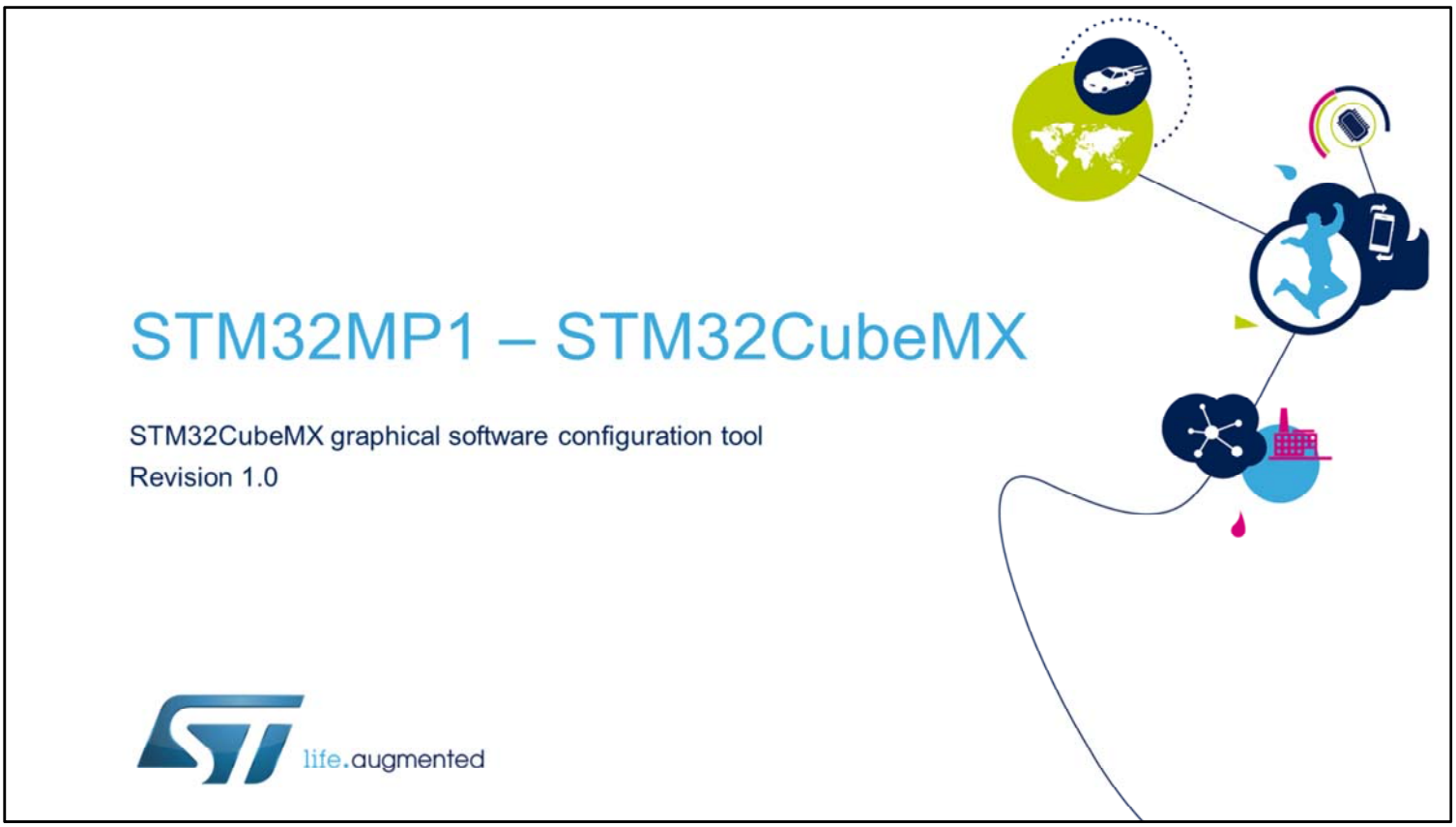
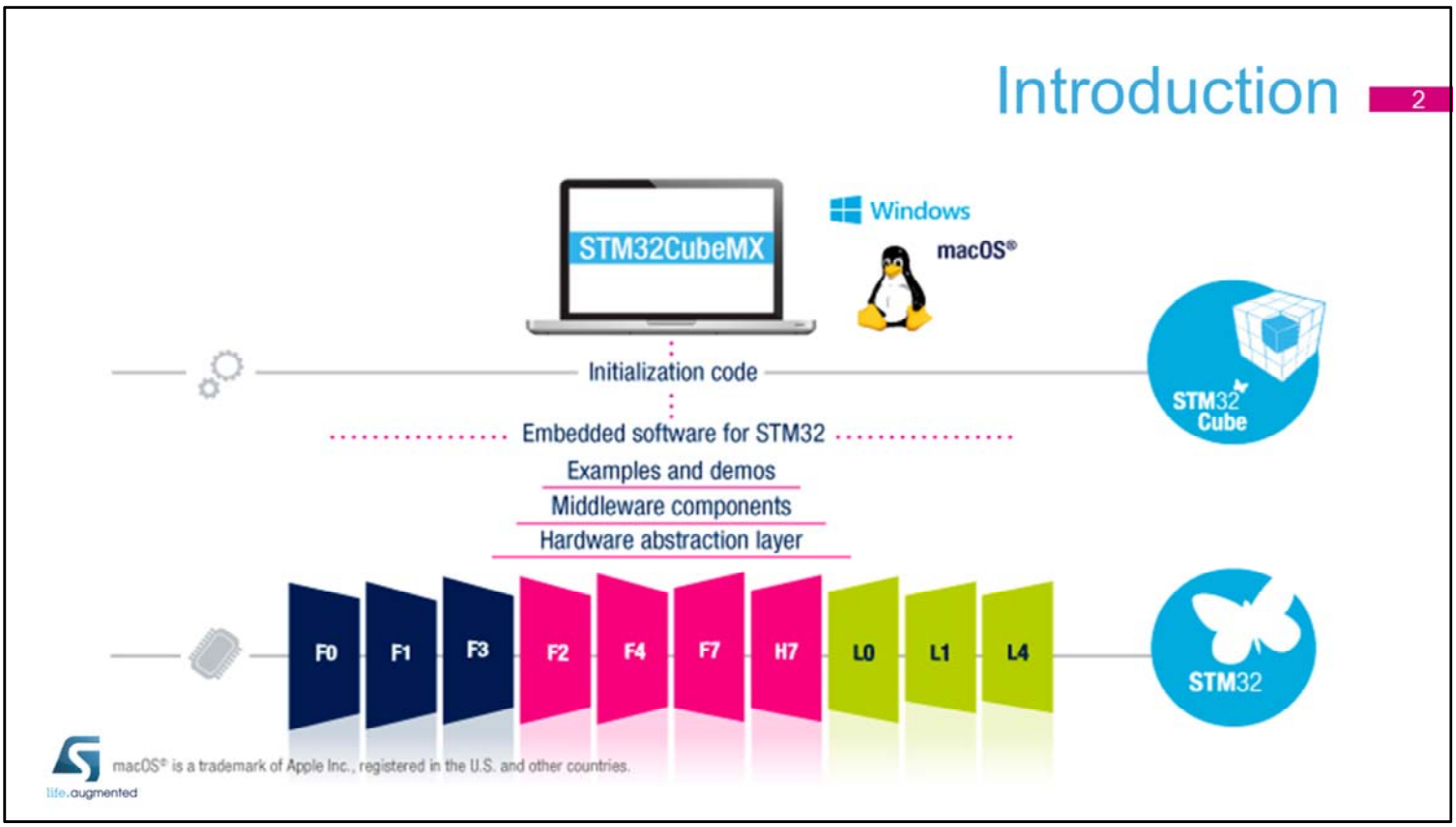


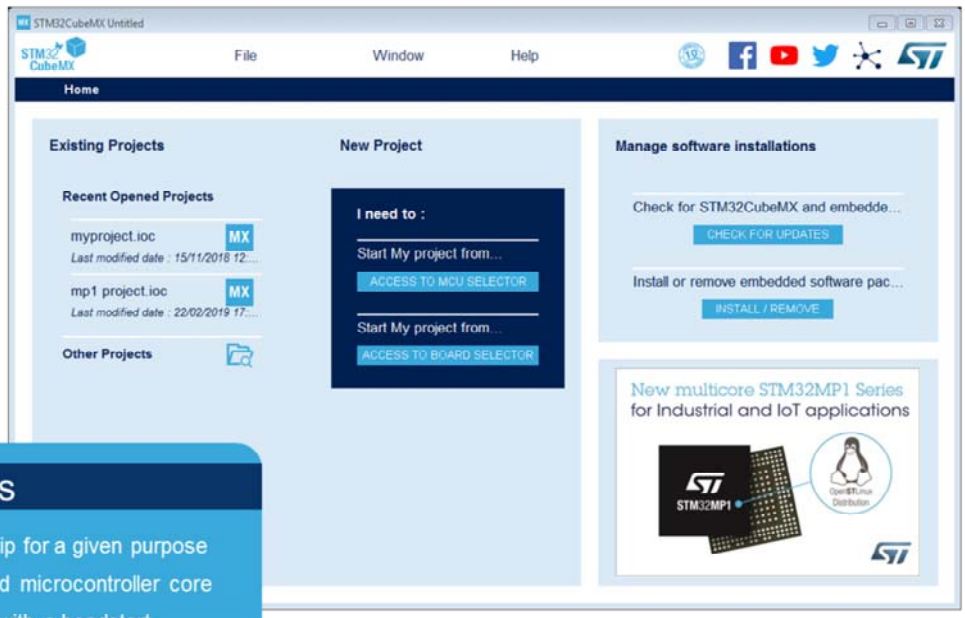
Hello, and welcome to this presentation of the STM32CubeMX Code Generation Tool. It covers the main features of this tool, which is used to configure and generate code, compile and debug, and estimate power consumption for the STM32 family of microcontrollers.



While this presentation is specifically about the STM32MP1 microprocessor, STM32CubeMX is a common platform to the whole STM32 family. However the feature set available for MPU class device is different.

Overview

- Choose MCU/MPU and configure:
  - Pinouts
  - Clock tree setup
  - MCU peripherals
  - MPU device tree source files generating
  - Middleware
  - Memory



Application benefits

- Helps choose the correct chip for a given purpose
- Generate code for embedded microcontroller core
- Boosts development speed with a headstart

life.augmented

The STM32CubeMX application helps developers using STM32 microcontrollers through a user interface that guides the initial configuration of a firmware project.
It provides the means to configure pin assignments, the clock tree, integrated peripherals, and simulate the power consumption of the resulting project. It uses a rich library of data from the STM32 microcontroller portfolio.
The application is intended to ease the initial phase of development, by helping developers select the best product with regards to features and power.

Key features

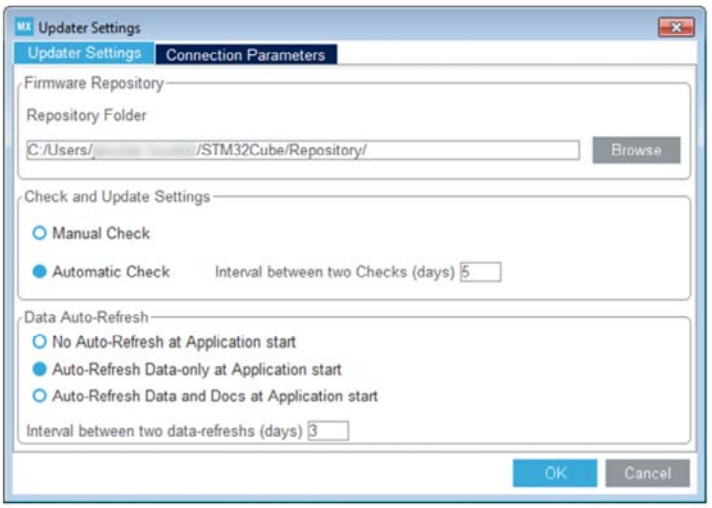
- Peripheral and middleware parameters and assignment to core context
- Power consumption calculator
- Code generation
  - Possible to re-generate HAL code for M4 while keeping user code intact.
  - Device tree sources generation for the application core (Linux)
- DDR Test Tool for testing and tuning
- MCU selector
  - Filter by family, package, peripherals or memory sizes
  - Search for similar product
- Pinout configuration
  - Choose peripherals to use and assign GPIO and alternate functions to pins.
- Configure NVIC and DMA
- Clock tree initialization
  - Choose oscillator and set PLL and clock dividers.

life.augmented

The user interface is built around a natural workflow of choosing a suitable MCU, selecting the required peripherals and assigning pin configurations.
The power consumption calculator aids in designing an efficient system.
Finally, the project initialization code can be generated and, potentially, re-generated while keeping the user code intact.

Prerequisites and settings

- STM32CubeMX needs Java RE
  - Check release notes of the particular version for additional requirements.
  - Multiplatform tool runs on Windows, Linux and macOS
- After installation, hit Alt+S to configure the updater – not only for the GUI but also for Cube FW libraries.
- Select SW library placement.



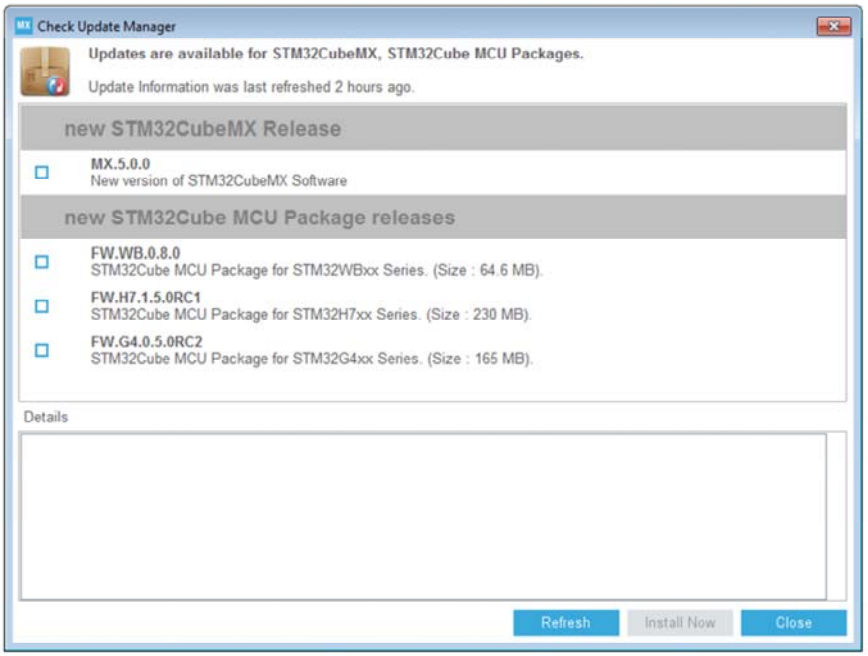
Download the STM32CubeMX installer for free from the ST website and install it.
Then, set your preferences in the Settings menu:

- one menu for the updater and library download (Alt+S),
- the other menu for code generation and integration with development toolchains (Alt+P).

Once this setup is completed, a new project can be created.

Updater

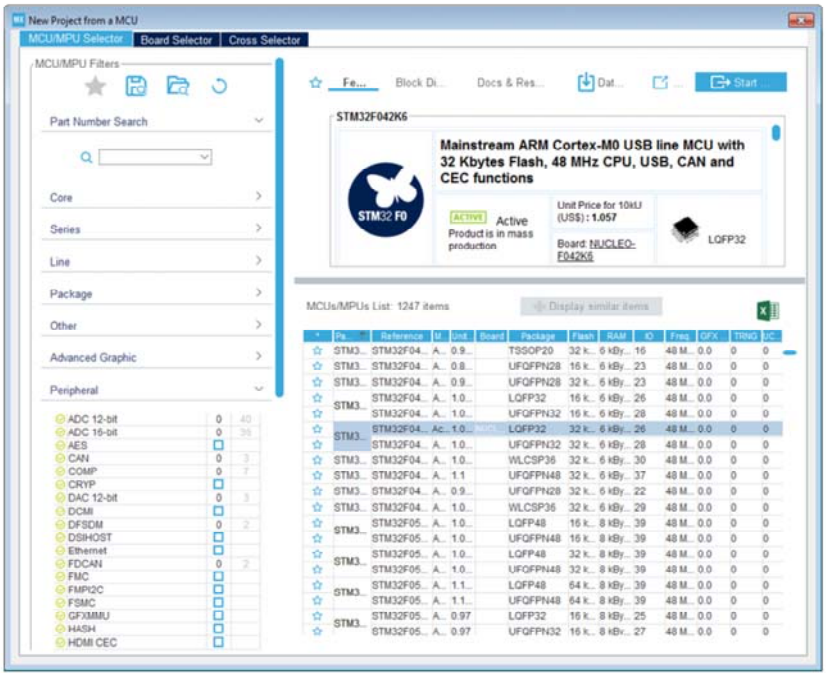
- Updates are accessible from the Help menu
- The tool updater can detect new releases of the tool and the associated Cube library.
- Use the libraries manager to download new library packages.



If the internet connection is configured correctly, the tool can update itself as well as the code libraries used for generating the project workspaces.
Use the “Install new libraries” option (Alt+U) to download additional STM32Cube libraries, or retrieve older versions for interoperability reasons.
However, note that the STM32CubeMX tool is not tested with all historical library releases, and, new library releases may not work correctly with old tool versions.

MCU/MPU selector

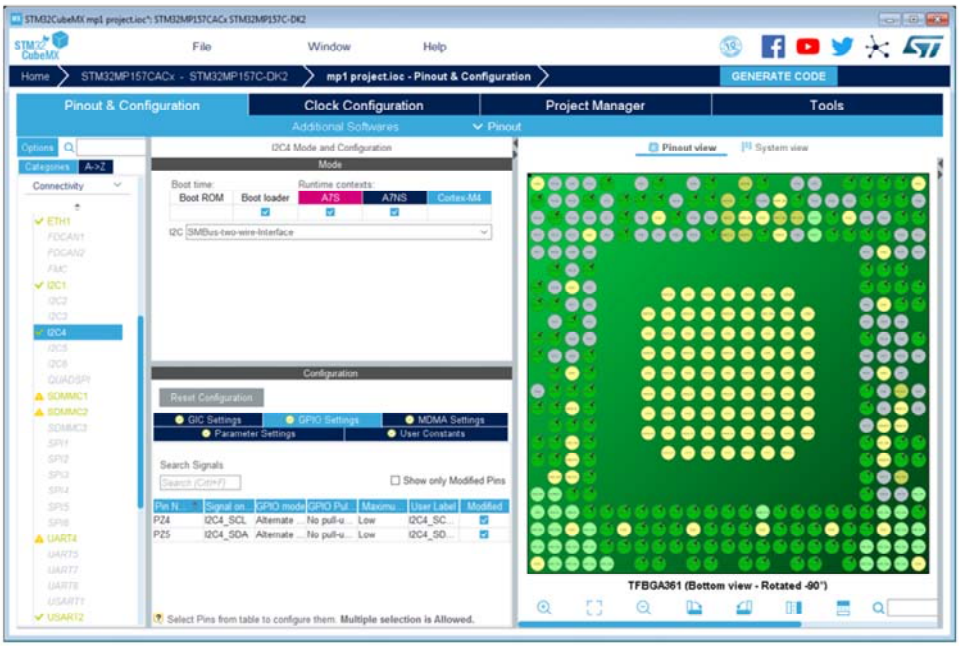
- Find MCU by name …
  - Quickly locate by Series and Lines
- … or application needs
  - Package (pin count)
  - RAM size
  - NV memory requirements
  - Embedded peripherals
  - Number and type of interfaces
  - Core and frequency
  - Price
- Convenient links to documentation
- Export table to Excel file



The MCU selector window will come up after selecting the “New Project” option. If the user knows which MCU to use, it can be found quickly.
If not, the available products can be filtered based on the specific requirements.

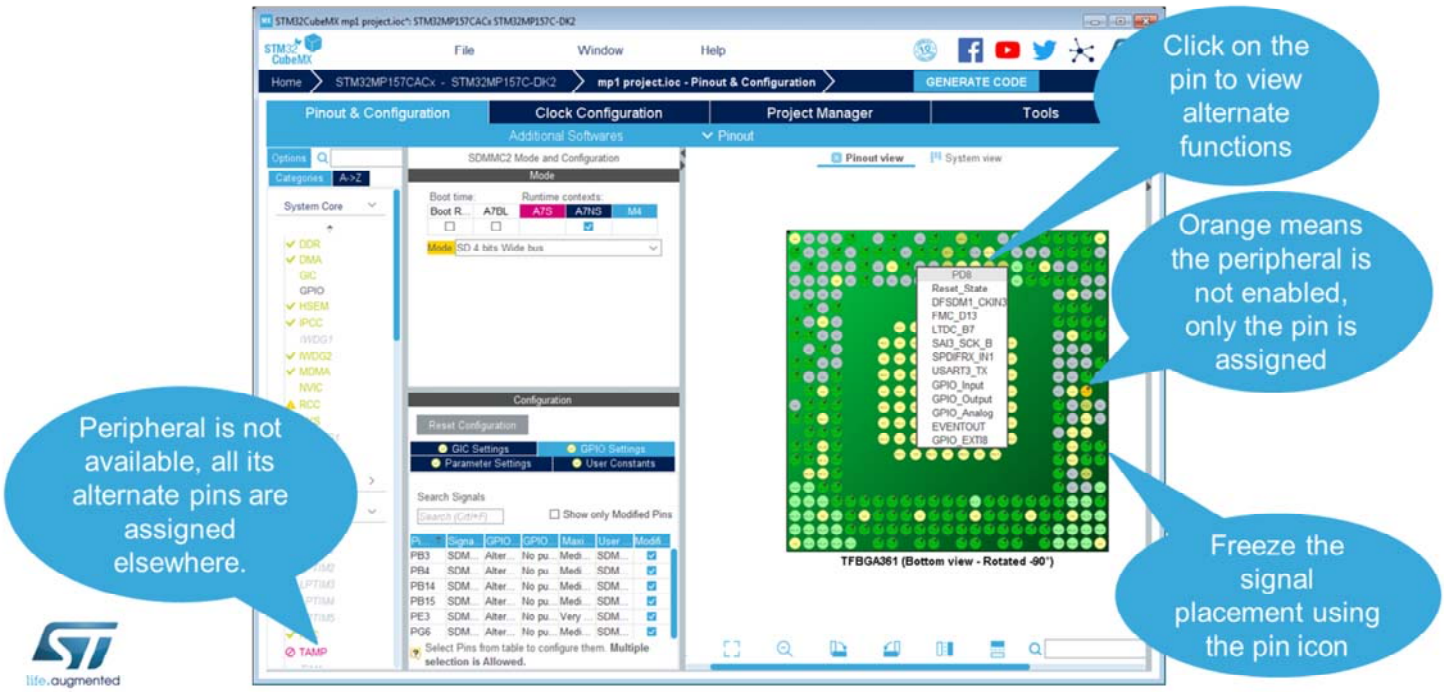
Pin assignment

- Pinout from:
  - Peripheral tree
  - Manually
- Automatic signal remapping
- Management of dependencies between peripherals and/or middleware (FatFS, USB …)

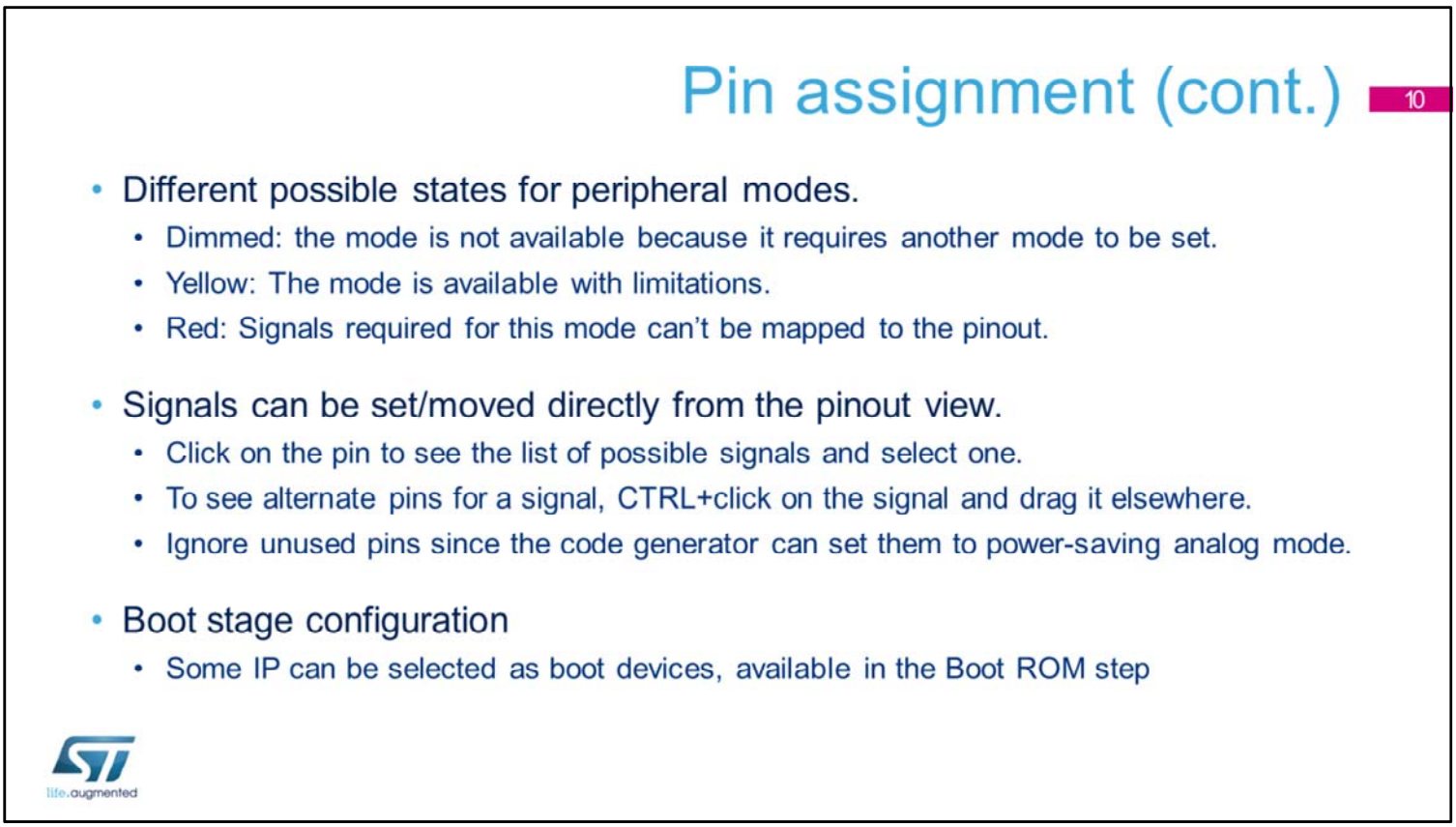


The next step is to select the peripherals to be used and, where applicable, assign pins to their inputs and outputs. Independent GPIOs can also be configured.
Signals are assigned to default pins, but they can be transferred to alternate locations, which are displayed by CTRL-clicking on the pin.
For example, when the I2C peripheral is enabled, the tool automatically assigns it to the default pins.
The tool automatically takes into account most bonds between the peripherals and software components it manages.

Pin assignment continued



As more pins are reserved for their alternate functions, the choice of remaining configurations for other peripherals decreases.
The limitations are indicated by icon changes on other peripheral nodes.
Left-click on the pin to display its alternate functions.
Right-click on the pin to name or select the pin assignment.
If a pinout is selected without a particular peripheral enabled or if there is any other problem with the pinout, the pin turns orange instead of green.



There are different possible states for peripheral modes:

- Dimmed: The mode is not available because it requires another mode to be set. Place the mouse pointer over the dimmed mode to see the reason – it may require a disabled clock source or have other peripheral dependencies.
- Yellow: The mode is available with limitations because some options are blocked by conflicts. For example, the USART may not be configured to synchronous mode because all selectable clock pins are taken.
- Red: Signals required for this mode cannot be mapped to the pinout. This may occur, for example, if a crucial signal has all its alternate pins used by other peripherals.

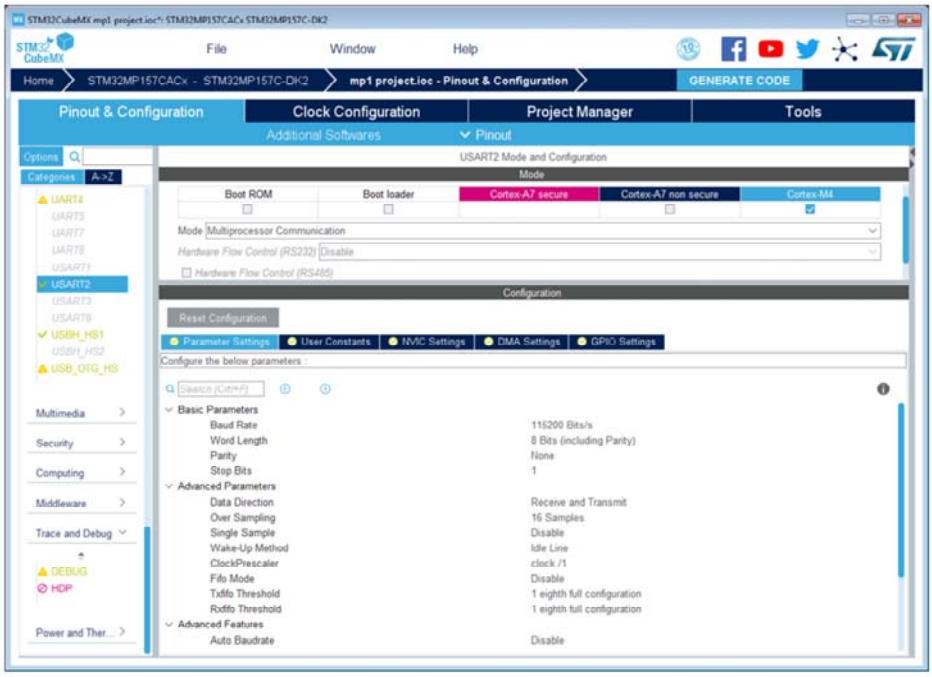
Signals can be set/moved directly from the pinout view.

- Click on the pin to display the list of possible signals and select one. This works for GPIOs which have no peripherals assigned.

- To see alternate pins for a signal, CTRL+click on the signal. You can then drag and drop the signal to the new pin (while holding the CTRL key).
- It is not necessary to manually set all unused pins to analog. There is a semi-automated step that does this.

Peripheral configuration

- Assign each peripheral to a core/step in Mode section
- All available initialization parameters are presented with short description and options.
- Interrupt may be assigned to peripherals.
- DMA may be associated, where applicable.
- GPIO settings for peripherals with input and/or output.



When configuring a peripheral, most important is to decide what boot step and ARM core it is assigned to. That determine what code is generated for it. Not all IP are available for each mode.

The dialogue window shows basic parameters, dependencies and constraints. Simple drop-down menus are used when applicable.

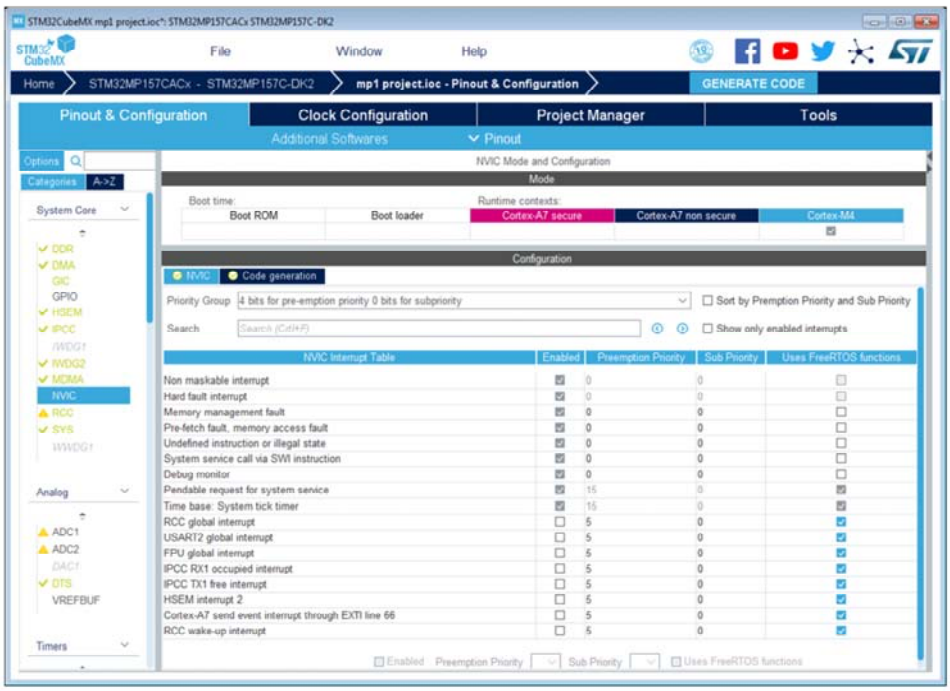
Interrupts priorities can only be set in the “NVIC settings” tab. The peripheral window can only be used to enable or disable each interrupt.

The DMA settings tab contains all the parameters for DMA requests relevant for initialization, but run-time parameters (start address, …) are not managed here.

The GPIO settings tab is used to define GPIO parameters and features, pin filtering and the possibility to label each signal for easy identification.

NVIC configuration panel

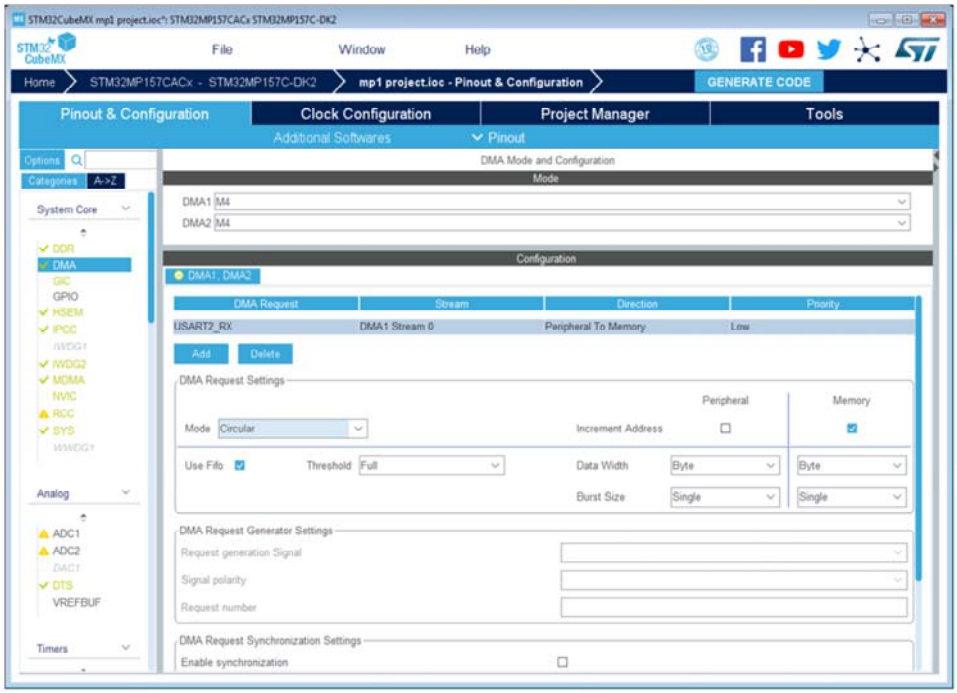
- Single control panel for all interrupts.
- Manage priorities and sub-priorities.
- Searching, filtering and sorting interrupts in the list.
- Code generation tab allows to customize interrupt initialization.



A central location with easy-to-understand overview of available and enabled interrupts, along with their priorities, is another advantage of STM32CubeMX. This window is used to enable interrupts for selected peripherals and to configure the priorities.

DMA configuration panel

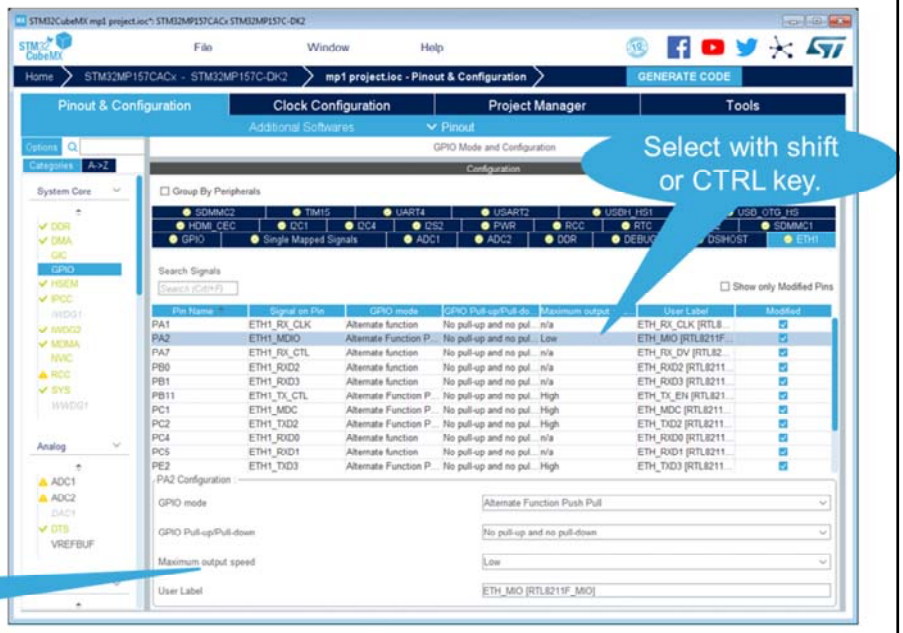
- Manages all DMA requests including memory to memory.
- Configure direction, priority and other settings.



Select the tab for the corresponding DMA channel and click the “Add” button to add a DMA request for the specified peripherals. Verify all configuration options. Note that this configures a DMA channel, but does not fully describe a DMA transfer. This must be done in the application code.

GPIO settings panel

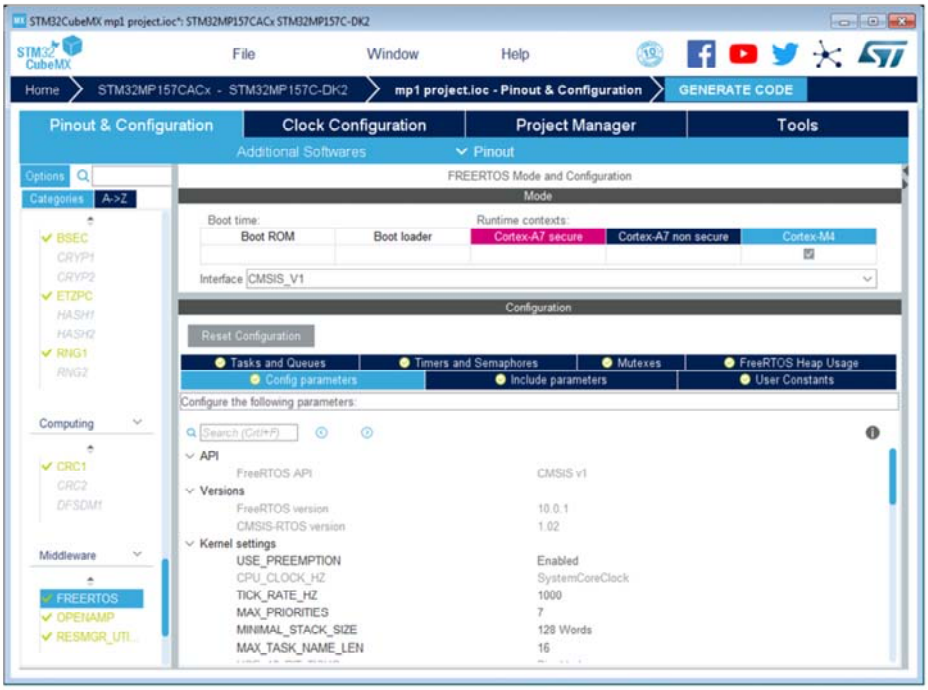
- The application attempts to set sensible default values for most GPIO parameters.
- Default values are chosen, as low-speed and no pull-up.
- Multiple pins can be selected to set them to the same configuration.



The GPIO tab in the Pin configuration window facilitates the configuration and initialization settings for each pin.
Each pin is listed in table format which provides an overview of the pin configuration along with its user label.
Sort, search, and apply modifications to selected pins using drop-down menus.
Default values assigned by the tool are safe but not may not work with certain peripheral configurations.
Check that the GPIO speed selected by the tool is sufficient for the peripheral communication speed, and that an internal pull-up is selected where needed.
To assign the settings faster, try selecting groups of pins rather than configuring pins individually. Use tabs to get pin groups dedicated to specific peripherals.
Note that settings applied during initialization can be modified during runtime, but that is outside the scope of the STM32CubeMX tool.

Middleware configuration

- Presents options specific to each supported software component.
- All settings are organized in logical groups.
- Description and constraints are available for quick reference.



Each middleware software component has options that are different, but they are all presented in a similar fashion, giving easy access to initialization options and providing informative descriptions.

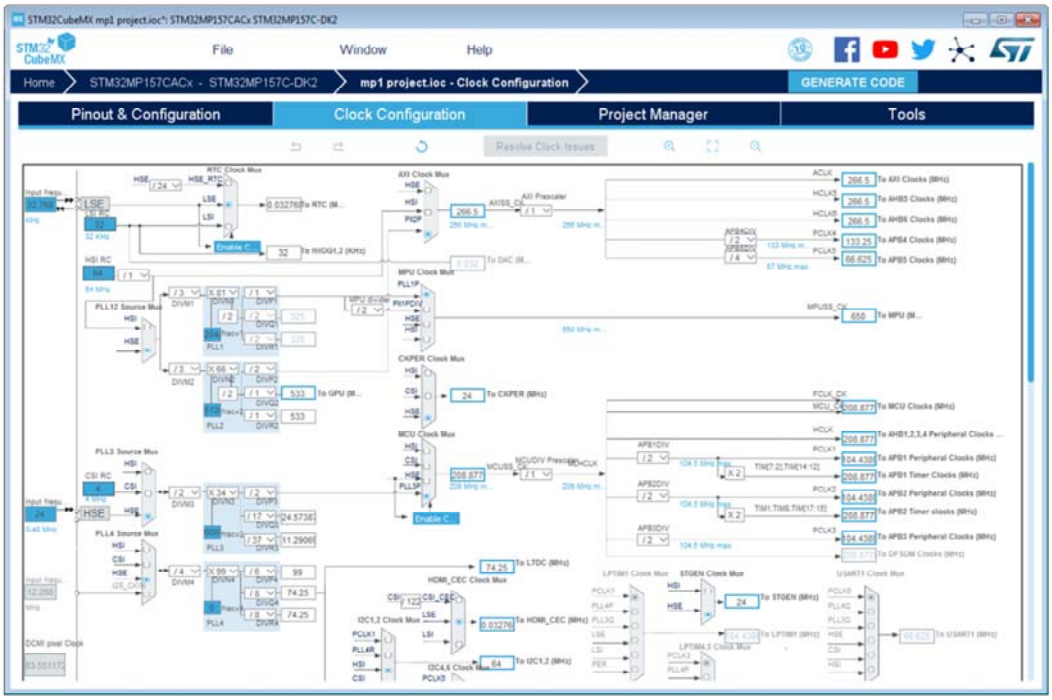
Peripheral and Middleware configuration

- Global view of used peripherals and middleware.
- Highlight of configuration state
  - Not configured
  - OK
  - Non-blocking problem
  - Error

The Configuration tab of the main window provides an overview of all the configurable hardware and software components that STM32CubeMX can help set up.
Each button with access to configuration options is displayed with a small icon indicating the configuration state.
The default state is not configured. Clicking on a button for a peripheral or middleware displays its configuration options.
Even when configured correctly, further modifications are possible.
Warning signs provide notifications about incorrect configurations, and the peripheral will not work if code is generated in this state.
Critical errors are represented by a red “X“ and the configuration must be modified to continue.
To add more peripherals and components, return to the Pinout tab.

Clock configuration

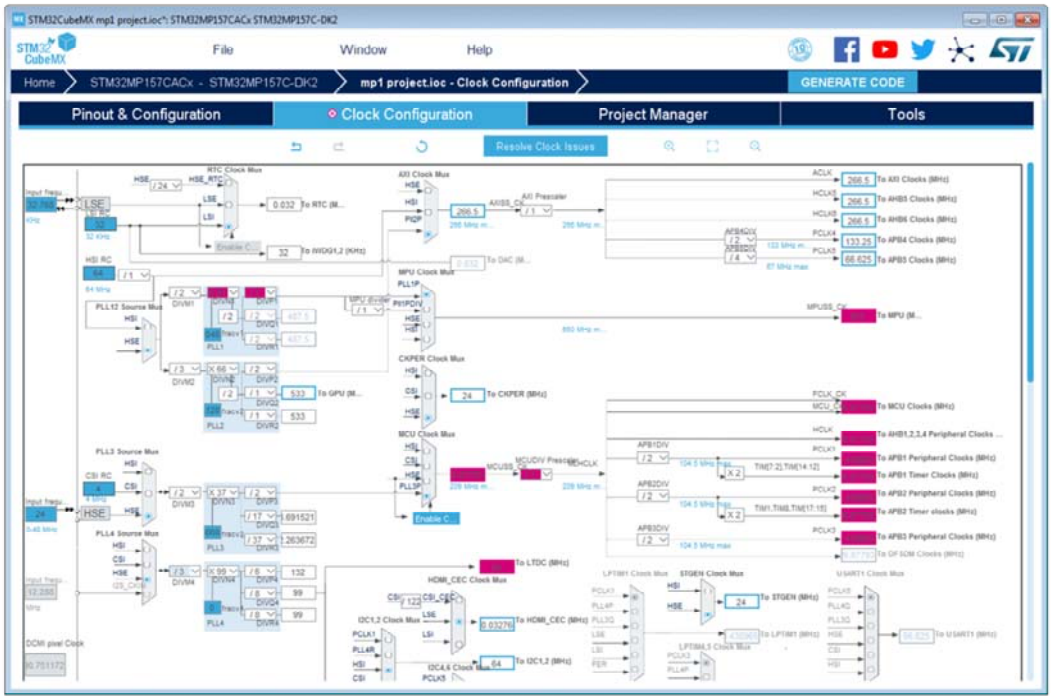
- Immediate display of all clock values.
- Active and inactive clock paths are differentiated.
- Management of clock constraints and features.



The clock configuration tab provides a schematic overview of the clock paths, along with all clock sources, dividers, and multipliers. Actual clock speeds are visible.
Active and enabled clock signals are highlighted in blue.
Drop-down menus and buttons serve to modify the actual clock configuration.

Clock configuration (cont.)

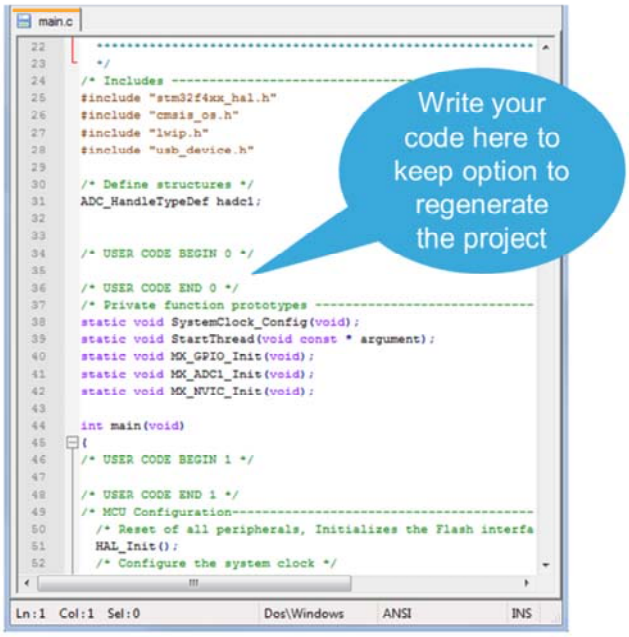
- Highlight of errors – instantly turns red.
- Enter the value in the blue frame and let the tool adjust the dividers and multipliers.
- Lock a value to prevent the tool from modifying it.



If a configured value is out of bounds, it immediately turns red to highlight a problem.
It also works the other way; enter the required clock speed in a blue frame and the software will attempt to reconfigure multipliers and dividers to provide the requested value.
Right-click on a clock value in blue to lock it to prevent modifications.

Code generation

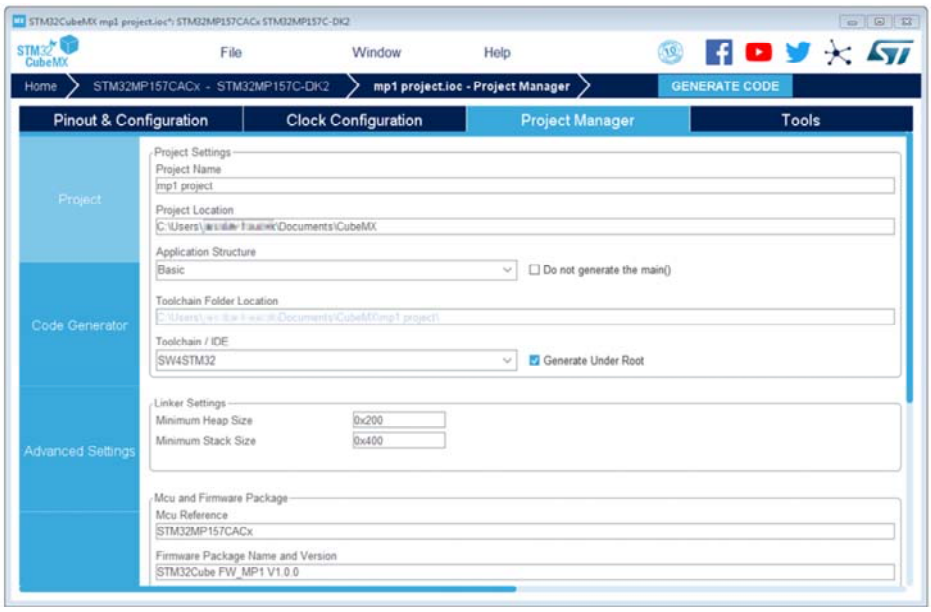
- Generates STM32HAL based C code to initialize MCU peripherals.
- Generated device tree sources for the application cores serve as an aid for U-Boot and Linux kernel configuration.
- Generates project file for any supported development toolchain.
- User code can be added in dedicated sections and will be kept upon regeneration.
- Option to use the latest library version or keep the same even if re-generating.



When all inputs, outputs, and peripherals are configured, the code is ready to be generated.
First, check the settings in the Project menu of the main window.
STM32Cube HAL initialization code is generated for the ARM M core, DTS for the ARM application core.
One of the several supported development tools can be selected to take over the generated project, including toolchains from Keil, IAR, and Atollic.
User code must be kept between the constraints of the “USER CODE“ comment blocks in order for the initialization settings to be modified using STM32Cube MX without affecting the custom code.

Code generation project settings

- Name your project when saving
- Browse for project location
- Pick the preferred toolchain
- Review the exact MCU/MPU type and library version

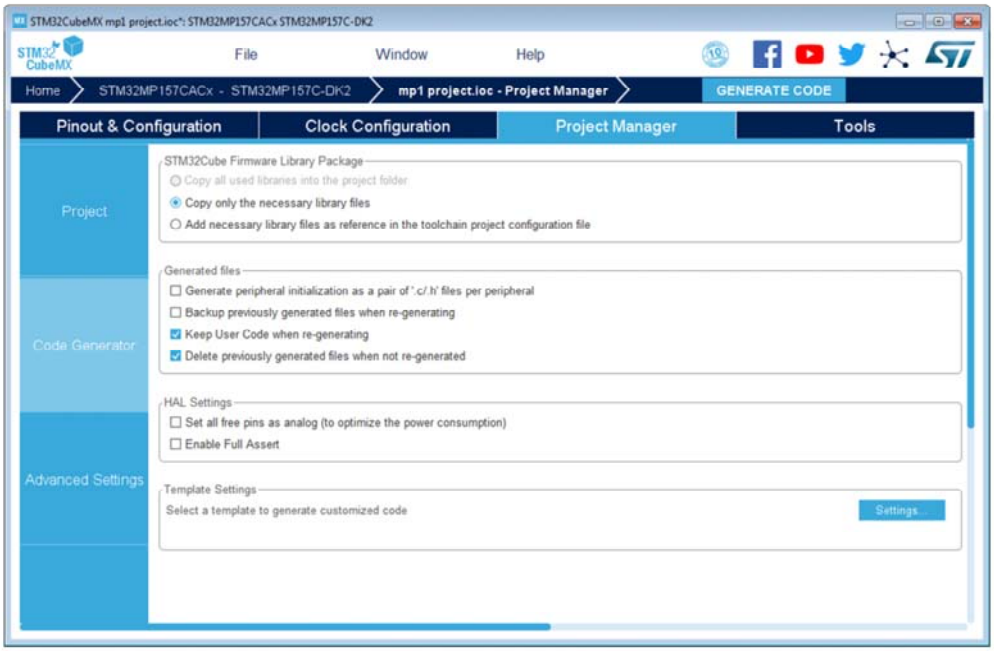


life.augmented

This window is available when saving the project (Save as...). The toolchain folder refers to where the workspace for the toolchain will be placed, not the actual toolchain application location. A limited version of this dialog window is also available using the Alt+P shortcut to display Project settings.

Code generation options

- Library package
  - Whole library or the necessary part may be copied to the generated project folder.
  - Or keep the library in original place and refer to it from all projects.
- Generated files
  - Each peripheral initialized in separate file or in common source file.
  - Options for working with old files.
  - **The option to keep user code intact is here.**
- HAL settings
  - Setting available pins to analog reduces power consumption, but be careful to **explicitly select SWD/JTAG in pinout.**
  - Full assert is useful for debugging.



The STM32Cube HAL library may be associated with the project in various ways. Select the Copy option if the project should be migrated as a compact package or if there is need to customize the library code. Keeping the library in the original location makes it easier to share the latest version of the library among several projects.

It can also generate the initialization code for all peripherals together in the stm32mp1xx_hal_msp.c file, or generate one file per peripheral.

Options to back up or delete old files are a matter of the preferred workflow. Keep in mind that the options are tied to the re-generation function. This is also where the “keep user code when re-generating” option is enabled.

The “Set all free pins as analog” setting helps lower power consumption, but if the SWD/JTAG interface is not specifically selected in the Pinout tab, this option will disable the debug interface.

“Full assert” enables checking the parameters passed to the

HAL functions, and may help reveal some bugs in the user code without an excessive debugging effort.

Warning and disclaimer

22

- Universal dedication toward the whole STM32 family range at times prevents the tool from focusing on specific features of a particular product.
- **The STM32CubeMX GUI tool is not a replacement for the reference manual or datasheet**
  - Always refer to the technical documentation for further information!
  - Important features are often available on the product or in the HAL but not in the GUI.
- The GUI helps start a project and initialize a working configuration – but the configuration can be dynamically changed at runtime (i.e. GPIO, NVIC priority or clock settings).

The user interface is a great tool, it is a universal assistant for all STM32 microcontrollers. However it cannot tackle all the details of each product while providing a useful overview of the diversified STM32 portfolio.
In case of doubt, please refer to the reference manual or datasheet for more detailed and accurate information. Do not hesitate to read the application notes and examples to learn more. It is common practice to start an application with STM32CubeMX to quickly get a prototype working, and then, modify the code when dynamic changes are needed (typically to support a different clock or a new GPIO configuration in the same application).
If the user wrote the code within the user areas defined by the STM32CubeMX generator, it is possible to revert to the initial STM32CubeMX setup, when some modifications are needed at the top level of user interface. This typically involves adding GPIO pin configurations, selecting another clock, or changing the NVIC Priority, for example.

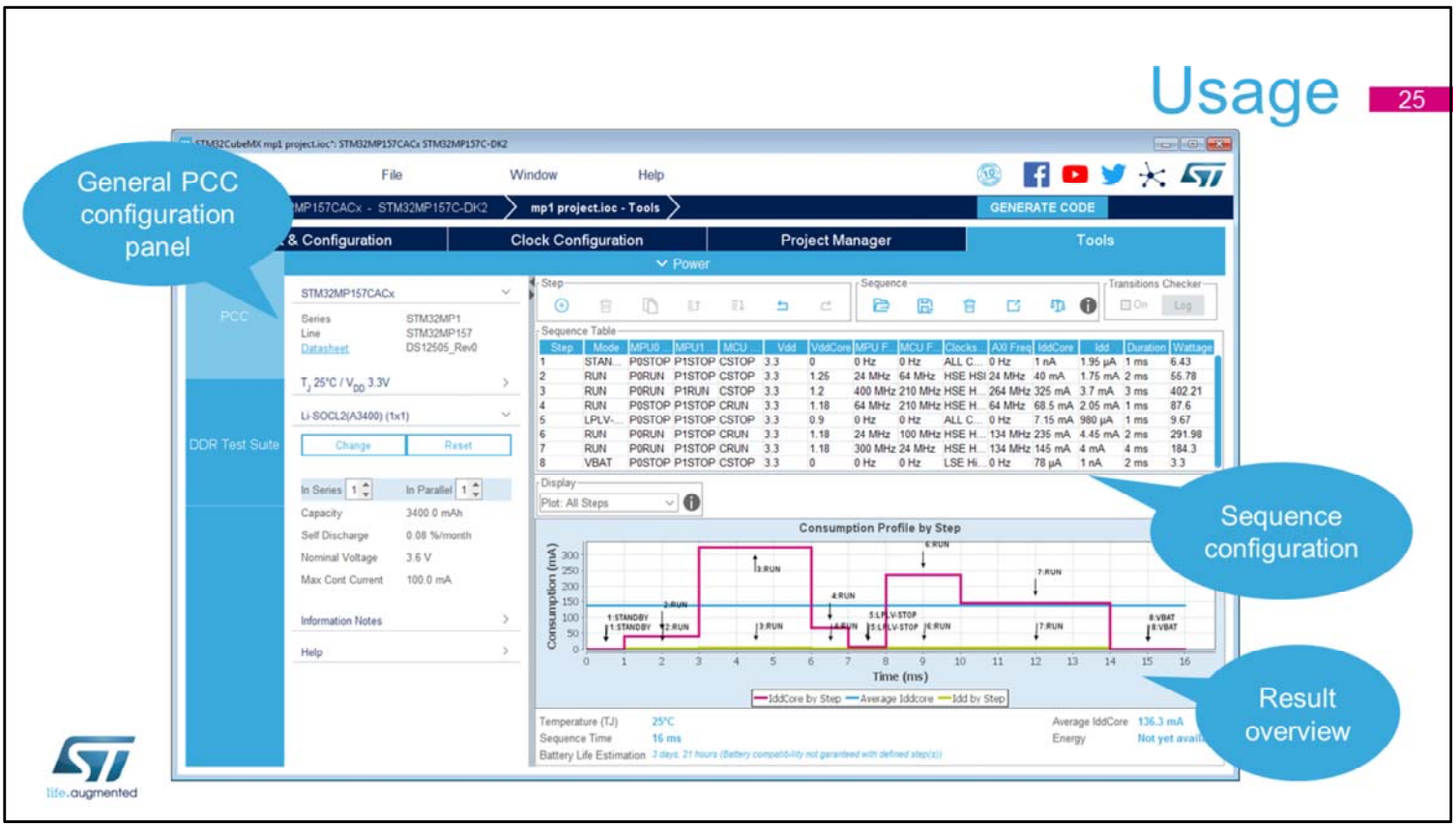
When developing embedded applications, low power consumption is often the primary design goal. Extracting power consumption levels from datasheets is a time-consuming and tedious job. The power consumption calculator attempts to simplify this task by exporting referenced values from the datasheet to a smart graphic tool, producing informative estimates from configurable use cases. For external memory timing settings, an automated DDR memory tester is also available.

Introduction

- The Power Consumption Calculator (PCC) uses a database of typical values to estimate the power consumption, DMIPS, and battery life of STM32 MCUs.
- Graphic tool integrated into the STM32CubeMX suite
- Highly configurable use cases with validity check
- Use the battery selector or define a custom battery
- Facilitates comparison with other MCUs/MPUs or other power options
- Import, export and generate reports

life.augmented

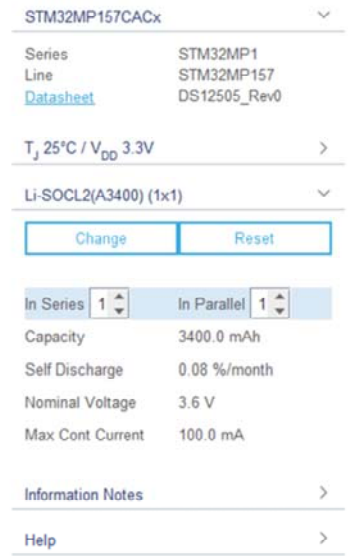
The Power Consumption Calculator can estimate the battery lifetime used as either main or supplementary power supplies. Sequences can easily be imported and exported. Illegal state transitions are detected too. It is even possible to compare sequence executions of two different MCUs or MPUs and generate a report.



The Power Consumption Calculator is the fourth tab in the STM32CubeMX main window. The window is divided into several panes.
The general configuration pane summarizes the typical operating conditions and the MCU type currently selected.
The second pane displays the simulation sequence and its controls.
There is no button to execute the simulation; the results are available instantly.

General PCC parameters

- MCU selection taken from STM32CubeMX
  - Use the direct link to the datasheet to get more detailed information.
- Parameter selection
  - Temperature and voltage choice may be limited, depending on the selected MCU.
- Battery selection – select existing or define your own
  - Battery is defined by capacity, voltage, self discharge and current limitations.
- Information notes
  - Purpose is to warn about estimation limitations.

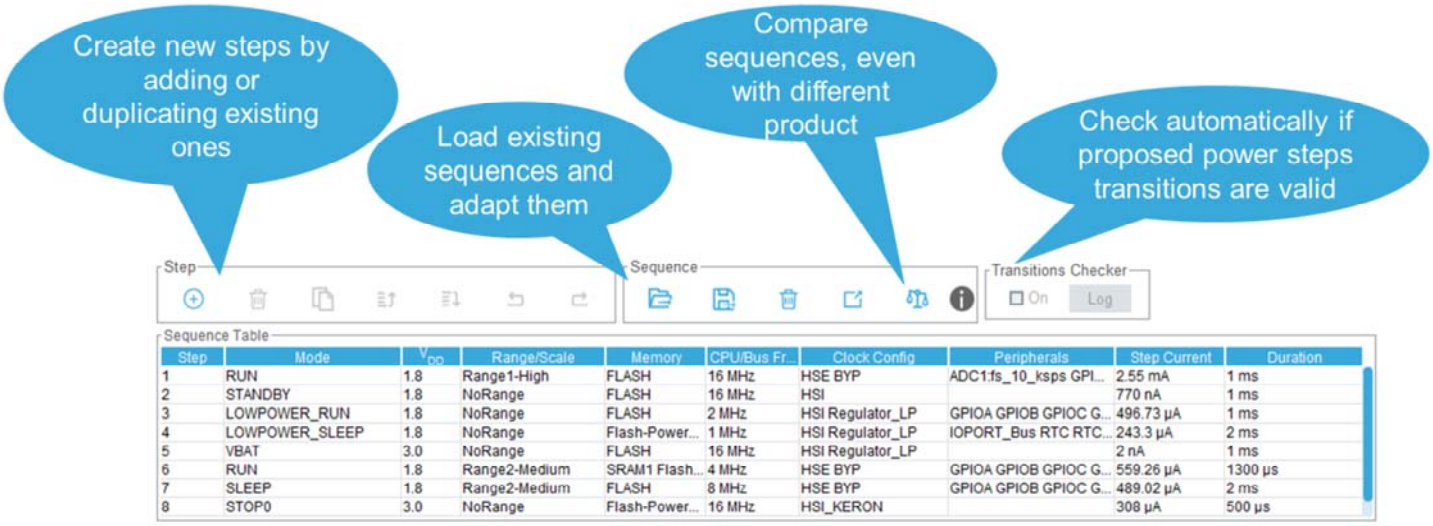


life.augmented

The general PCC configuration pane is mostly informative, summarizing the selected MCU and the default power source.
Parameters such as temperature and voltage may even be defined, depending on the MCU selected and the available power consumption data.
The Battery selection pane is used to select or define a battery type. The battery source is optional and, if defined, may be used in only selected sequence steps, simulating a device that works both independently and connected to an external power source.
Information and help sections include useful notes for the user.

Building a sequence

- Sequence is a set of ordered steps



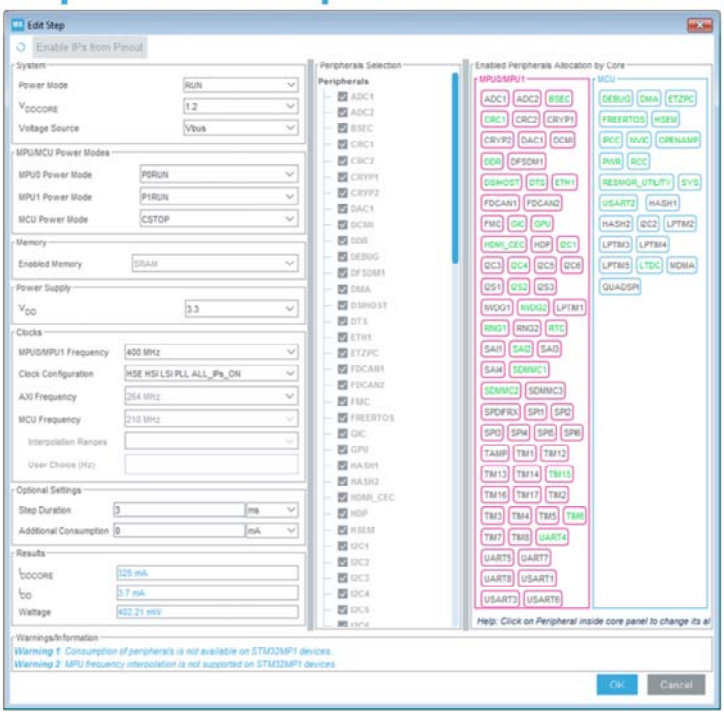
Step	Mode	V_{DD}	Range/Scale	Memory	CPU/Bus Fr...	Clock Config	Peripherals	Step Current	Duration
1	RUN	1.8	Range1-High	FLASH	16 MHz	HSE BYP	ADC1:fs_10_ksps GPI...	2.55 mA	1 ms
2	STANDBY	1.8	NoRange	FLASH	16 MHz	HSI		770 nA	1 ms
3	LOWPOWER_RUN	1.8	NoRange	FLASH	2 MHz	HSI Regulator_LP	GPIOA GPIOB GPIOC G...	496.73 µA	1 ms
4	LOWPOWER_SLEEP	1.8	NoRange	Flash-Power...	1 MHz	HSI Regulator_LP	IOPORT_Bus RTC RTC...	243.3 µA	2 ms
5	VBAT	3.0	NoRange	FLASH	16 MHz	HSI Regulator_LP		2 nA	1 ms
6	RUN	1.8	Range2-Medium	SRAM1 Flash...	4 MHz	HSE BYP	GPIOA GPIOB GPIOC G...	559.26 µA	1300 µs
7	SLEEP	1.8	Range2-Medium	FLASH	8 MHz	HSE BYP	GPIOA GPIOB GPIOC G...	489.02 µA	2 ms
8	STOP0	3.0	NoRange	Flash-Power...	16 MHz	HSI_KERON		308 µA	500 µs

life.augmented

The “Sequence Table” defines a series of steps, with different durations and configurations. Its length is virtually unlimited.
Sequences can be loaded, modified, and reused.
Individual steps can be duplicated and repositioned within the sequence using the User Interface.
If enabled, all state transitions are checked against basic validity rules to prevent illegal jumps in frequency or power ranges. Problematic steps are instantly highlighted in the sequence table.
Click on the “Show log” button to display a detailed explanation.
The “Compare” feature displays a comparison of the power and performance in the current scenario with a saved sequence. Different configurations, including different MCUs, can be evaluated against each other.

Power consumption step definition

1. Clock setting balances performance and consumption.
2. Select the memory from which the code is executed.
3. Vdd – several representative settings available.
4. Choose which core is active in the step.

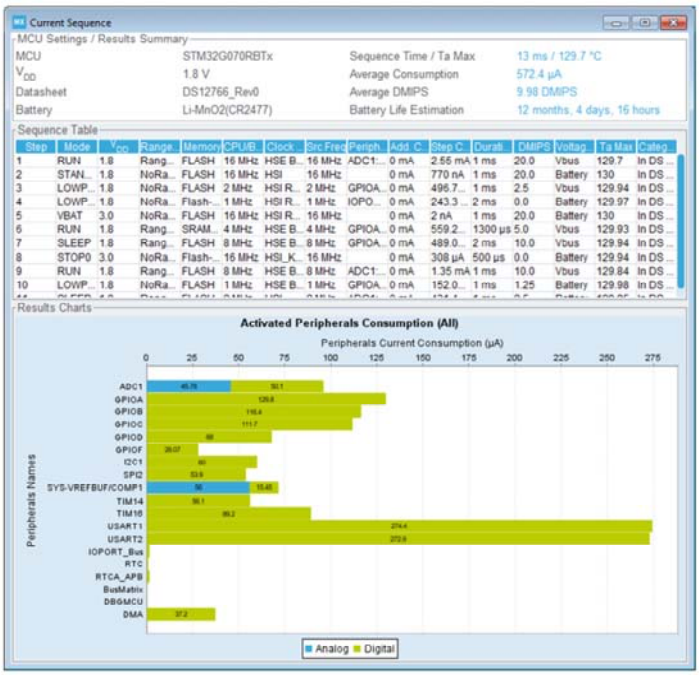


A power step can be added or edited in this dialog window. If the transition checker is enabled, it will preset the new step with allowed values.
The power step is determined by several characteristics, with the power mode being the most important. The availability and characteristics of each power mode are described in the specific reference manual or datasheet.
Power mode selection has the most significant impact on the availability of other settings, interfaces and power/performance balance.
The voltage regulator sets the core voltage. At lower voltages, the system clock frequency is limited, but the power consumption is often drastically reduced. Refer to the datasheet for more details.
The address from which the instruction is fetched and the related settings can also influence the power consumption and available clock speeds.
The supply voltage for which power consumption is

calculated. Use the nearest possible value if the actual voltage is not available.
The last option is present to exclude cases when the device is, for example, connected to the USB in battery drain model.
To learn more about power modes, refer to the system power control module training presentation.

Sequence consumption profile display

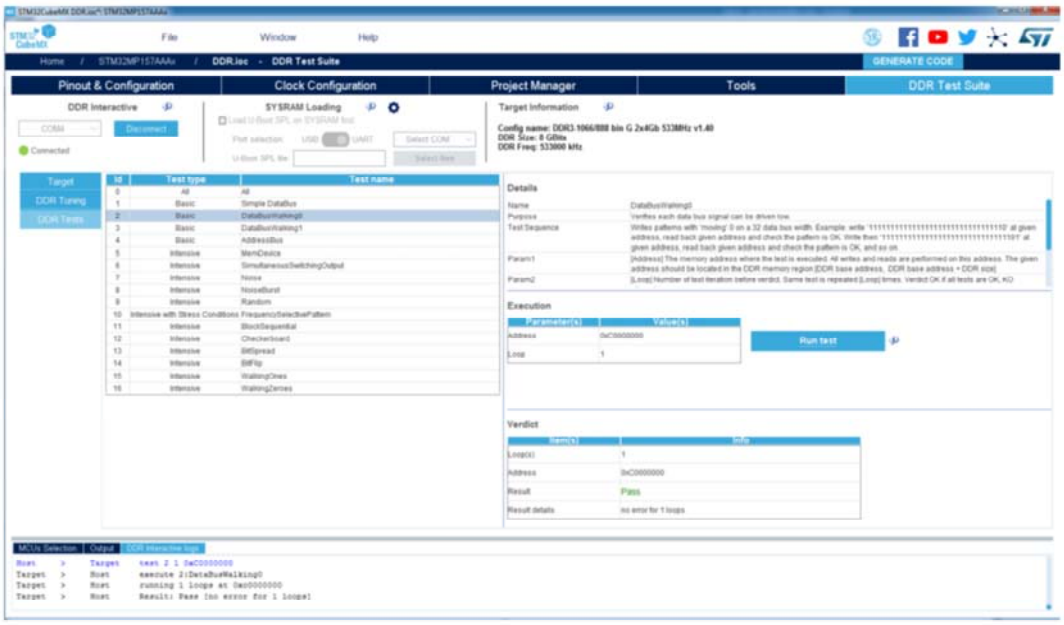
- It's possible to detach the charts to external display for presentation purposes
- Several different views selectable
  - Plot current vs time
  - Pie chart
  - Consumption of peripherals



The power consumption calculator features powerful presentation tools. Click on "Ext. Display" button to display the report in a separate window. There are many different ways available to plot the current consumption estimates in graphical form. The default method is based on the power step sequence and the consumption over time.
Alternatively, the percentage of energy spent in different modes can be charted. The pie chart may show the share of each mode, or split to only display Run and Low-power modes.
It is also possible to split the power consumption of peripherals, and plot their power requirements in a graph. You can plot digital peripherals only, analog peripherals only, or a mixed view with both.

DDR Test Suite

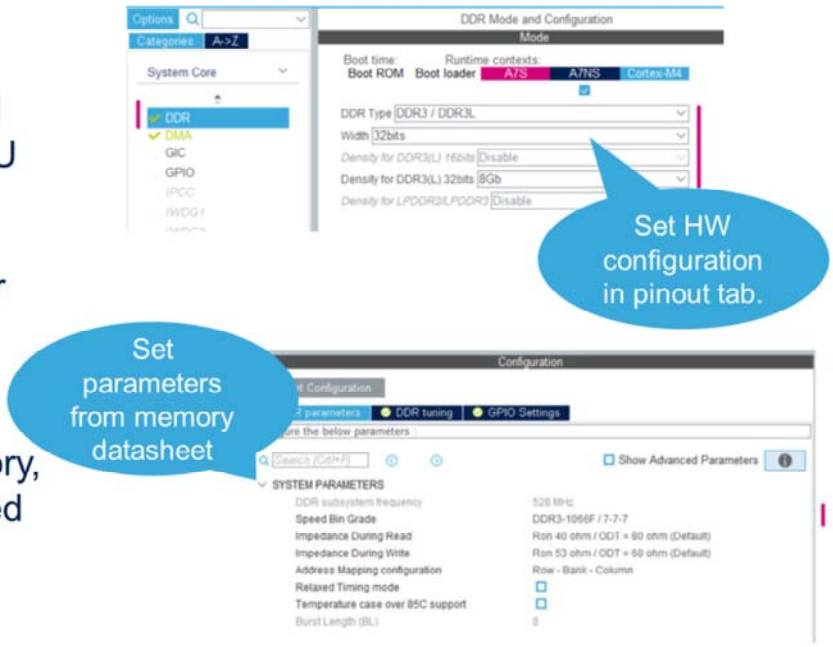
- Configuration
- Testing
- Tuning



Following three slides are a brief introduction to a new STM32CubeMX feature, so far specific to the STM32MP1 MPU. It is the ability to configure, test and fine-tune a connection to a standard external DDR memory.

DDR Test Suite pre-requisites

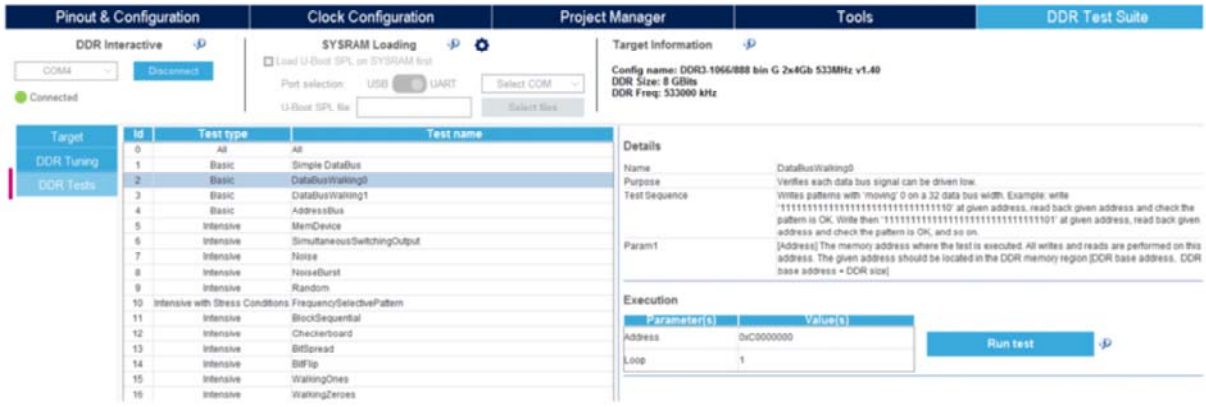
- It is necessary to connect a board populated with a STM32MP1 MPU (connection through UART port)
- U-Boot secondary program loader (SPL) binary to load (either to external flash or SYSRAM)
- If not booting from external memory, STM32cubeProgrammer is needed to load U-Boot SPL in SYSRAM



Unlike other function of the STM32CubeMX, this one requires a physical board connection and a binary to load. Basic U-Boot SPL binary image is available in the starter package. The connection may be a simple STLink Virtual Com Port (VCP) when a Discovery board is used. To test a user specific board, the default port used by the U-Boot SPL for connection is the UART4 port.

DDR Test

- Tests are executed by the U-Boot SPL
- STM32CubeMX tool only configures testing and presents the results



This function provides easy means to execute various tests to detect and identify possible failures in the dynamic memory. Both basis and stress tests are available in the U-Boot SPL and executable from the STM32CubeMX suite.

DDR Tuning

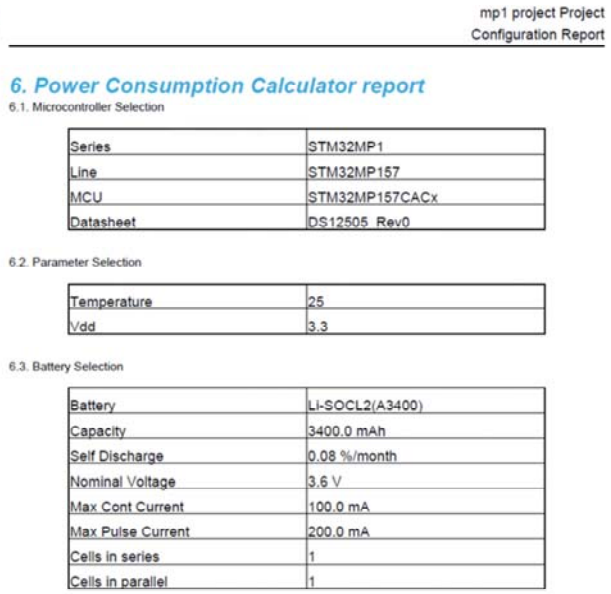
- The goal is to compensate for minor imperfections in the target HW, especially PCB routing.
- The tuning can set optimal values for Bit Deskew, DQS Gating and Eye centering.
- Refer to application notes AN5168 and AN5122 for further details

life.augmented

The DDR Tuning is a semi-automated process used to determine fine settings of the DDR memory interface and compensate for unequal route length and other factors. See application notes AN5168 and AN5122 for further details. A more detailed description is also available in the STM32CubeMX documentation.

Output and generating report

- An optional step is to generate a PDF report
- The PDF report is also available without PCC.
- Complete saved project work includes:
  - Project.ioc
  - Project.pcs
  - Project.pdf
  - Project.txt
  - Project.jpg
  - … and the generated project for a supported development environment.



mp1 project Project Configuration Report

6. Power Consumption Calculator report

6.1. Microcontroller Selection

Series	STM32MP1
Line	STM32MP157
MCU	STM32MP157CACx
Datasheet	DS12505_Rev0

6.2. Parameter Selection

Temperature	25
Vdd	3.3

6.3. Battery Selection

Battery	LI-SOCL2(A3400)
Capacity	3400.0 mAh
Self Discharge	0.08 %/month
Nominal Voltage	3.6 V
Max Cont Current	100.0 mA
Max Pulse Current	200.0 mA
Cells in series	1
Cells in parallel	1

A file with the extension .ioc contains the static initialization settings. The power sequence is saved using the .pcs extension. A PDF report is generated, along with simplified text and a separate JPG image file with pinout.

References

- For more details, please refer to the following resources
  - UM1718 – User manual
  - DB2163 – Product specifications
  - TN0072 – Product technical note
  - RN0094 – Product release note
  - Download the tool from ST website www.st.com

life.augmented

For more information about using the STM32CubeMX Code Generation Tool, the documents listed in this slide are available for download on www.st.com.