UM1718
User manual

STM32CubeMX for STM32 configuration and initialization C code generation

Introduction

STM32CubeMX is a graphical tool for 32-bit ARM® Cortex® STM32 microcontrollers. It is part of STMCube™ initiative (see *Section 1*) and is available either as a standalone application or as an Eclipse plug-in for integration in Integrated Development Environments (IDEs).

STM32CubeMX has the following key features:

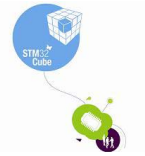
- **Easy microcontroller selection covering whole STM32 portfolio.**
- **Board selection from a list of STMicroelectronics boards.**
- **Easy microcontroller configuration** (pins, clock tree, peripherals, middleware) and generation of the corresponding initialization C code.
- **Easy switching to another microcontroller belonging to the same series** by importing a previously-saved configuration to a new MCU project.
- **Generation of configuration reports**.
- **Generation of IDE ready projects** for a selection of integrated development environment tool chains. STM32CubeMX projects include the generated initialization C code, STM32 HAL drivers, the middleware stacks required for the user configuration, and all the relevant files needed to open and build the project in the selected IDE.
- **Power consumption calculation** for a user-defined application sequence**.**
- **Self-updates** allowing the user to keep the STM32CubeMX up-to-date.
- Download and update of STM32Cube™ embedded software required for user application development (see *Appendix E: STM32Cube embedded software packages* for details on STM32Cube embedded software offer).

Although STM32CubeMX offers a user interface and generates a C code compliant with STM32 MCU design and firmware solutions, it is recommended to refer to the product technical documentation for details on actual implementation of microcontroller peripherals and firmware.

Reference documents

The following documents are available from *http://www.st.com*:

- STM32 microcontroller reference manuals
- STM32 microcontroller datasheets
- *STM32Cube HAL driver user manuals for STM32F0 (UM1785), STM32F1 (UM1850), STM32F2 (UM1940), STM32F3 (UM1786), STM32F4 (UM1725), STM32F7 (UM1905), STM32L0 (UM1749), STM32L1 (UM1816) and STM32L4 (UM1884).*



Contents

1 **STM32Cube overview 13**

2 **Getting started with STM32CubeMX 14**

2.1 Principles 14

2.2 Key features 16

2.3 Rules and limitations 17

3 **Installing and running STM32CubeMX 18**

3.1 System requirements 18

3.1.1 Supported operating systems and architectures 18

3.1.2 Memory prerequisites 18

3.1.3 Software requirements 18

3.2 Installing/uninstalling STM32CubeMX standalone version 18

3.2.1 Installing STM32CubeMX standalone version 18

3.2.2 Installing STM32CubeMX from command line 19

3.2.3 Uninstalling STM32CubeMX standalone version 22

3.3 Installing STM32CubeMX plug-in version 22

3.3.1 Downloading STM32CubeMX plug-in installation package 22

3.3.2 Installing STM32CubeMX as an Eclipse IDE plug-in 23

3.3.3 Uninstalling STM32CubeMX as Eclipse IDE plug-in 24

3.4 Launching STM32CubeMX 26

3.4.1 Running STM32CubeMX as standalone application 26

3.4.2 Running STM32CubeMX in command-line mode 26

3.4.3 Running STM32CubeMX plug-in from Eclipse IDE 28

3.5 Getting STM32Cube updates 30

3.5.1 Updater configuration 31

3.5.2 Downloading new libraries 34

3.5.3 Removing libraries 36

3.5.4 Checking for updates 37

4 **STM32CubeMX User Interface 38**

4.1 Welcome page 38

4.2 New project window 39

4.3 Main window . . . 42
4.4 Toolbar and menus . . . 45
4.4.1 File menu . . . 45
4.4.2 Project menu . . . 46
4.4.3 Pinout menu . . . 46
4.4.4 Window menu . . . 48
4.4.5 Help menu . . . 48
4.5 Output windows . . . 49
4.5.1 MCUs selection pane . . . 49
4.5.2 Output pane . . . 49
4.6 Import Project window . . . 50
4.7 Set unused / Reset used GPIOs windows . . . 56
4.8 Project Settings window . . . 58
4.8.1 Project tab . . . 60
4.8.2 Code Generator tab . . . 61
4.8.3 Advanced Settings tab . . . 65
4.9 Update Manager windows . . . 66
4.10 About window . . . 67
4.11 Pinout view . . . 67
4.11.1 IP tree pane . . . 69
4.11.2 Chip view . . . 70
4.11.3 Chip view advanced actions . . . 73
4.11.4 Keep Current Signals Placement . . . 75
4.11.5 Pinning and labeling signals on pins . . . 76
4.11.6 Setting HAL timebase source . . . 77
4.12 Configuration view . . . 83
4.12.1 IP and Middleware Configuration window . . . 85
4.12.2 User Constants configuration window . . . 87
4.12.3 GPIO Configuration window . . . 92
4.12.4 DMA Configuration window . . . 95
4.12.5 NVIC Configuration window . . . 97
4.12.6 FreeRTOS middleware configuration view . . . 105
4.13 Clock tree configuration view . . . 111
4.13.1 Clock tree configuration functions . . . 111
4.13.2 Recommendations . . . 116
4.13.3 STM32F43x/42x power-over drive feature . . . 117

4.13.4 Clock tree glossary 119
4.14 Power Consumption Calculator (PCC) view 119
4.14.1 Building a power consumption sequence 120
4.14.2 Configuring a step in the power sequence 127
4.14.3 Managing user-defined power sequence and reviewing results 131
4.14.4 Power sequence step parameters glossary 134
4.14.5 Battery glossary 137

5 STM32CubeMX C Code generation overview 138
5.1 Standard STM32Cube code generation 138
5.2 Custom code generation 141
5.2.1 STM32CubeMX data model for FreeMarker user templates 141
5.2.2 Saving and selecting user templates 141
5.2.3 Custom code generation 142

6 Tutorial 1: From pinout to project C code generation using an STM32F4 MCU 145
6.1 Creating a new STM32CubeMX Project 145
6.2 Configuring the MCU pinout 148
6.3 Saving the project 149
6.4 Generating the report 150
6.5 Configuring the MCU Clock tree 150
6.6 Configuring the MCU initialization parameters 153
6.6.1 Initial conditions 153
6.6.2 Configuring the peripherals 154
6.6.3 Configuring the GPIOs 157
6.6.4 Configuring the DMAs 158
6.6.5 Configuring the middleware 159
6.7 Generating a complete C project 162
6.7.1 Setting project options 162
6.7.2 Downloading firmware package and generating the C code 164
6.8 Building and updating the C code project 169
6.9 Switching to another MCU 174

7 Tutorial 2 - Example of FatFs on an SD card using STM32429I-EVAL evaluation board 176

8 Tutorial 3- Using PCC to optimize the embedded application power consumption and more 183

8.1 Tutorial overview 183

8.2 Application example description 184

8.3 Using the Power Consumption Calculator 184

8.3.1 Creating a PCC sequence 184

8.3.2 Optimizing application power consumption 187

9 FAQ 195

9.1 On the Pinout configuration pane, why does STM32CubeMX move some functions when I add a new peripheral mode? 195

9.2 How can I manually force a function remapping? 195

9.3 Why are some pins highlighted in yellow or in light green in the Chip view? Why cannot I change the function of some pins (when I click some pins, nothing happens)? 195

9.4 Why do I get the error “Java 7 update 45’ when installing ‘Java 7 update 45’ or a more recent version of the JRE? 195

9.5 Why does the RTC multiplexer remain inactive on the Clock tree view? 196

9.6 How can I select LSE and HSE as clock source and change the frequency? 197

9.7 Why STM32CubeMX does not allow me to configure PC13, PC14, PC15 and PI8 as outputs when one of them is already configured as an output? 197

Appendix A STM32CubeMX pin assignment rules 198

A.1 Block consistency 198

A.2 Block inter-dependency 202

A.3 One block = one peripheral mode 205

A.4 Block remapping (STM32F10x only) 205

A.5 Function remapping 206

A.6 Block shifting (only for STM32F10x and when “Keep Current Signals placement” is unchecked) 207

A.7 Setting and clearing a peripheral mode 208

A.8 Mapping a function individually 208

A.9 GPIO signals mapping 208

Appendix B STM32CubeMX C code generation design choices and limitations 209

B.1 STM32CubeMX generated C code and user sections 209

B.2 STM32CubeMX design choices for peripheral initialization 209

B.3 STM32CubeMX design choices and limitations for middleware initialization 210

B.3.1 Overview 210

B.3.2 USB Host 211

B.3.3 USB Device 211

B.3.4 FatFs 211

B.3.5 FreeRTOS 212

B.3.6 LwIP 213

Appendix C STM32 microcontrollers naming conventions 215

Appendix D STM32 microcontrollers power consumption parameters 217

D.1 Power modes 217

D.1.1 STM32L1 series 217

D.1.2 STM32F4 series 218

D.1.3 STM32L0 series 219

D.2 Power consumption ranges 220

D.2.1 STM32L1 series feature 3 VCORE ranges 220

D.2.2 STM32F4 series feature several VCORE scales 221

D.2.3 STM32L0 series feature 3 VCORE ranges 221

Appendix E STM32Cube embedded software packages 222

10 Revision history 223

List of tables

Table 1. Command line summary . . . 27
Table 2. Welcome page shortcuts . . . 39
Table 3. File menu functions . . . 45
Table 4. Project menu . . . 46
Table 5. Pinout menu . . . 47
Table 6. Window menu . . . 48
Table 7. Help menu . . . 48
Table 8. IP tree pane - icons and color scheme . . . 69
Table 9. STM32CubeMX Chip view - Icons and color scheme . . . 71
Table 10. IP configuration buttons . . . 84
Table 11. IP Configuration window buttons and tooltips . . . 86
Table 12. Clock tree view widget . . . 115
Table 13. Voltage scaling versus power over-drive and HCLK frequency . . . 118
Table 14. Relations between power over-drive and HCLK frequency . . . 118
Table 15. Glossary . . . 119
Table 16. Document revision history . . . 223

List of figures

Figure 1. Overview of STM32CubeMX C code generation flow. . . . 15
Figure 2. Example of STM32CubeMX installation in interactive mode 20
Figure 3. STM32Cube Installation Wizard 21
Figure 4. Auto-install command line. . . . 22
Figure 5. Adding STM32CubeMX plug-in archive 23
Figure 6. Installing STM32CubeMX plug-in 24
Figure 7. Closing STM32CubeMX perspective 24
Figure 8. Uninstalling STM32CubeMX plug-in 25
Figure 9. Opening Eclipse plug-in 29
Figure 10. STM32CubeMX perspective 29
Figure 11. Displaying Windows default proxy settings. . . . 30
Figure 12. Updater Settings window 31
Figure 13. Connection Parameters tab - No proxy 32
Figure 14. Connection Parameters tab - Use System proxy parameters. . . . 33
Figure 15. Connection Parameters tab - Manual Configuration of Proxy Server 34
Figure 16. New library Manager window 35
Figure 17. Removing libraries 36
Figure 18. Removing library confirmation message. . . . 37
Figure 19. Library deletion progress window 37
Figure 20. STM32CubeMX Welcome page 38
Figure 21. New Project window - MCU selector. . . . 40
Figure 22. New Project window - board selector 41
Figure 23. STM32CubeMX Main window upon MCU selection 42
Figure 24. STM32CubeMX Main window upon board selection (Peripheral default option unchecked) 43
Figure 25. STM32CubeMX Main window upon board selection (Peripheral default option checked) 44
Figure 26. Pinout menus (Pinout tab selected) 46
Figure 27. Pinout menus (Pinout tab not selected) 47
Figure 28. MCU selection menu 49
Figure 29. Output pane 50
Figure 30. Error message obtained when importing from different series 50
Figure 31. Automatic project import 51
Figure 32. Manual project import 52
Figure 33. Import Project menu - Try import with errors 54
Figure 34. Import Project menu - Successful import after adjustments 55
Figure 35. Set unused pins window 56
Figure 36. Reset used pins window 56
Figure 37. Set unused GPIO pins with Keep Current Signals Placement checked 57
Figure 38. Set unused GPIO pins with Keep Current Signals Placement unchecked 58
Figure 39. Project Settings window 59
Figure 40. Project folder. . . . 60
Figure 41. Project Settings Code Generator 63
Figure 42. Template Settings window 64
Figure 43. Generated project template 65
Figure 44. Advanced Settings window. . . . 66
Figure 45. About window 67
Figure 46. STM32CubeMX Pinout view. . . . 68

Figure 47. Chip view . . . 70
Figure 48. Red highlights and tooltip example: no mode configuration available . . . 72
Figure 49. Orange highlight and tooltip example: some configurations unavailable . . . 73
Figure 50. Tooltip example: all configurations unavailable . . . 73
Figure 51. Modifying pin assignments from the Chip view . . . 73
Figure 52. Example of remapping in case of block of pins consistency . . . 74
Figure 53. Pins/Signals Options window . . . 77
Figure 54. Selecting a HAL timebase source (STM32F407 example) . . . 78
Figure 55. TIM2 selected as HAL timebase source . . . 78
Figure 56. NVIC settings when using systick as HAL timebase, no FreeRTOS . . . 79
Figure 57. NVIC settings when using FreeRTOS and SysTick as HAL timebase . . . 80
Figure 58. NVIC settings when using freeRTOS and TIM2 as HAL timebase . . . 82
Figure 59. STM32CubeMX Configuration view . . . 83
Figure 60. Configuration window tabs for GPIO, DMA and NVIC settings (STM32F4 series) . . . 84
Figure 61. IP Configuration window (STM32F4 series) . . . 85
Figure 62. User Constants window . . . 87
Figure 63. Extract of the generated mxconstants.h file . . . 87
Figure 64. Using constants for peripheral parameter settings . . . 88
Figure 65. Specifying user constant value and name . . . 89
Figure 66. Deleting user constant not allowed when constant already used for another constant definition . . . 89
Figure 67. Deleting a user constant used for parameter configuration- Confirmation request . . . 90
Figure 68. Deleting a user constant used for peripheral configuration - Consequence on peripheral configuration . . . 90
Figure 69. Searching user constants list for name . . . 91
Figure 70. Searching user constants list for value . . . 91
Figure 71. GPIO Configuration window - GPIO selection . . . 92
Figure 72. GPIO Configuration window - displaying GPIO settings . . . 93
Figure 73. GPIO configuration grouped by IP . . . 94
Figure 74. Multiple Pins Configuration . . . 94
Figure 75. Adding a new DMA request . . . 95
Figure 76. DMA Configuration . . . 96
Figure 77. DMA MemToMem configuration . . . 97
Figure 78. NVIC Configuration tab - FreeRTOS disabled . . . 98
Figure 79. NVIC Configuration tab - FreeRTOS enabled . . . 99
Figure 80. I2C NVIC Configuration window . . . 99
Figure 81. NVIC Code generation – All interrupts enabled . . . 101
Figure 82. NVIC Code generation – Interrupt initialization sequence configuration . . . 103
Figure 83. NVIC Code generation – IRQ Handler generation . . . 104
Figure 84. FreeRTOS configuration view . . . 105
Figure 85. FreeRTOS: configuring tasks and queues . . . 106
Figure 86. FreeRTOS: creating a new task . . . 107
Figure 87. FreeRTOS - Configuring timers, mutexes and semaphores . . . 108
Figure 88. FreeRTOS Heap usage . . . 110
Figure 89. STM32F429xx Clock Tree configuration view . . . 114
Figure 90. Clock Tree configuration view with errors . . . 114
Figure 91. Clock tree configuration: enabling RTC, RCC Clock source and outputs from Pinout view . . . 116
Figure 92. Clock tree configuration: RCC Peripheral Advanced parameters . . . 117
Figure 93. Power Consumption Calculator default view . . . 120
Figure 94. Battery selection . . . 121

Figure 95. Building a power consumption sequence . . . 122
Figure 96. Step management functions . . . 122
Figure 97. Power consumption sequence: new step default view . . . 123
Figure 98. Edit Step window . . . 124
Figure 99. Enabling the transition checker option on an already configured sequence - all transitions valid . . . 125
Figure 100. Enabling the transition checker option on an already configured sequence - at least one transition invalid . . . 125
Figure 101. Transition checker option -show log . . . 126
Figure 102. Interpolated Power Consumption . . . 128
Figure 103. ADC selected in Pinout view . . . 129
Figure 104. PCC Step configuration window: ADC enabled using import pinout . . . 130
Figure 105. Power Consumption Calculator view after sequence building . . . 131
Figure 106. Sequence table management functions . . . 132
Figure 107. Power Consumption: Peripherals Consumption Chart . . . 133
Figure 108. Description of the Results area . . . 134
Figure 109. Peripheral power consumption tooltip . . . 136
Figure 110. Labels for pins generating define statements . . . 139
Figure 111. User constant generating define statements . . . 139
Figure 112. Duplicate labels . . . 139
Figure 113. extra_templates folder – default content . . . 142
Figure 114. extra_templates folder with user templates . . . 143
Figure 115. Project root folder with corresponding custom generated files . . . 143
Figure 116. User custom folder for templates . . . 144
Figure 117. Custom folder with corresponding custom generated files . . . 144
Figure 118. MCU selection . . . 145
Figure 119. Pinout view with MCUs selection . . . 146
Figure 120. Pinout view without MCUs selection window . . . 147
Figure 121. GPIO pin configuration . . . 148
Figure 122. Timer configuration . . . 148
Figure 123. Simple pinout configuration . . . 149
Figure 124. Save Project As window . . . 149
Figure 125. Generate Project Report - New project creation . . . 150
Figure 126. Generate Project Report - Project successfully created . . . 150
Figure 127. Clock tree view . . . 151
Figure 128. HSI clock enabled . . . 152
Figure 129. HSE clock source disabled . . . 152
Figure 130. HSE clock source enabled . . . 152
Figure 131. External PLL clock source enabled . . . 152
Figure 132. Configuration view . . . 154
Figure 133. Case of IP without configuration parameters . . . 154
Figure 134. Timer 3 configuration window . . . 155
Figure 135. Timer 3 configuration . . . 156
Figure 136. Enabling Timer 3 interrupt . . . 157
Figure 137. GPIO configuration color scheme and tooltip . . . 157
Figure 138. GPIO mode configuration . . . 158
Figure 139. DMA Parameters configuration window . . . 159
Figure 140. FatFs disabled . . . 159
Figure 141. USB Host configuration . . . 160
Figure 142. FatFs over USB mode enabled . . . 160
Figure 143. Configuration view with FatFs and USB enabled . . . 160
Figure 144. FatFs IP instances . . . 161

Figure 145. FatFs define statements . . . 161
Figure 146. Project Settings and toolchain choice . . . 162
Figure 147. Project Settings menu - Code Generator tab . . . 163
Figure 148. Missing firmware package warning message . . . 164
Figure 149. Error during download . . . 164
Figure 150. Updater settings for download . . . 165
Figure 151. Updater settings with connection . . . 166
Figure 152. Downloading the firmware package . . . 166
Figure 153. Unzipping the firmware package . . . 167
Figure 154. C code generation completion message . . . 167
Figure 155. C code generation output folder . . . 168
Figure 156. C code generation output: Projects folder . . . 169
Figure 157. C code generation for EWARM . . . 170
Figure 158. STM32CubeMX generated project open in IAR IDE . . . 171
Figure 159. IAR options . . . 172
Figure 160. SWD connection . . . 172
Figure 161. Project building log . . . 173
Figure 162. User Section 2 . . . 173
Figure 163. User Section 4 . . . 173
Figure 164. Import Project menu . . . 175
Figure 165. Project Import status . . . 175
Figure 166. Board selection . . . 176
Figure 167. SDIO IP configuration . . . 177
Figure 168. FatFs mode configuration . . . 177
Figure 169. RCC peripheral configuration . . . 177
Figure 170. Clock tree view . . . 178
Figure 171. Project Settings menu - Code Generator tab . . . 178
Figure 172. C code generation completion message . . . 179
Figure 173. IDE workspace . . . 179
Figure 174. Power Consumption Calculation example . . . 185
Figure 175. PCC VDD and battery selection menu . . . 186
Figure 176. PCC Sequence table . . . 186
Figure 177. PCC sequence results before optimization . . . 187
Figure 178. Step 1 optimization . . . 188
Figure 179. Step 5 optimization . . . 189
Figure 180. Step 6 optimization . . . 190
Figure 181. Step 7 optimization . . . 191
Figure 182. Step 8 optimization . . . 192
Figure 183. Step 10 optimization . . . 193
Figure 184. PCC Sequence results after optimizations . . . 194
Figure 185. Java Control Panel . . . 196
Figure 186. Pinout view - Enabling the RTC . . . 196
Figure 187. Pinout view - Enabling LSE and HSE clocks . . . 197
Figure 188. Pinout view - Setting LSE/HSE clock frequency . . . 197
Figure 189. Block mapping . . . 199
Figure 190. Block remapping . . . 200
Figure 191. Block remapping - example 1 . . . 201
Figure 192. Block remapping - example 2 . . . 202
Figure 193. Block inter-dependency - SPI signals assigned to PB3/4/5 . . . 203
Figure 194. Block inter-dependency - SPI1_MISO function assigned to PA6 . . . 204
Figure 195. One block = one peripheral mode - I2C1_SMBA function assigned to PB5 . . . 205
Figure 196. Block remapping - example 2 . . . 206

Figure 197. Function remapping example 206
Figure 198. Block shifting not applied 207
Figure 199. Block shifting applied 208
Figure 200. FreeRTOS HOOK functions to be completed by user 212
Figure 201. LwIP 1.4.1 configuration 213
Figure 202. LwIP 1.5 configuration 214
Figure 203. STM32 microcontroller part numbering scheme 216
Figure 204. STM32Cube Embedded Software package 222

1 STM32Cube overview

STMCube™ is an STMicroelectronics original initiative to ease developers life by reducing development efforts, time and cost. STM32Cube covers STM32 portfolio.

STM32Cube includes:

- The STM32CubeMX, a graphical software configuration tool that allows to generate C initialization C code using graphical wizards.
- A comprehensive embedded software platform, delivered per series (such as STM32CubeF2 for STM32F2 series and STM32CubeF4 for STM32F4 series)
  - The STM32Cube HAL, an STM32 abstraction layer embedded software, ensuring maximized portability across STM32 portfolio
  - A consistent set of middleware components such as RTOS, USB, TCP/IP, Graphics
  - All embedded software utilities coming with a full set of examples.

2 Getting started with STM32CubeMX

2.1 Principles

Customers need to quickly identify the MCU that best meets their requirements (core architecture, features, memory size, performance…). While board designers main concerns are to optimize the microcontroller pin configuration for their board layout and to fulfill the application requirements (choice of peripherals operating modes), embedded system developers are more interested in developing new applications for a specific target device, and migrating existing designs to different microcontrollers.

The time taken to migrate to new platforms and update the C code to new firmware drivers adds unnecessary delays to the project. STM32CubeMX was developed within STM32Cube initiative which purpose is to meet customer key requirements to maximize software reuse and minimize the time to create the target system:

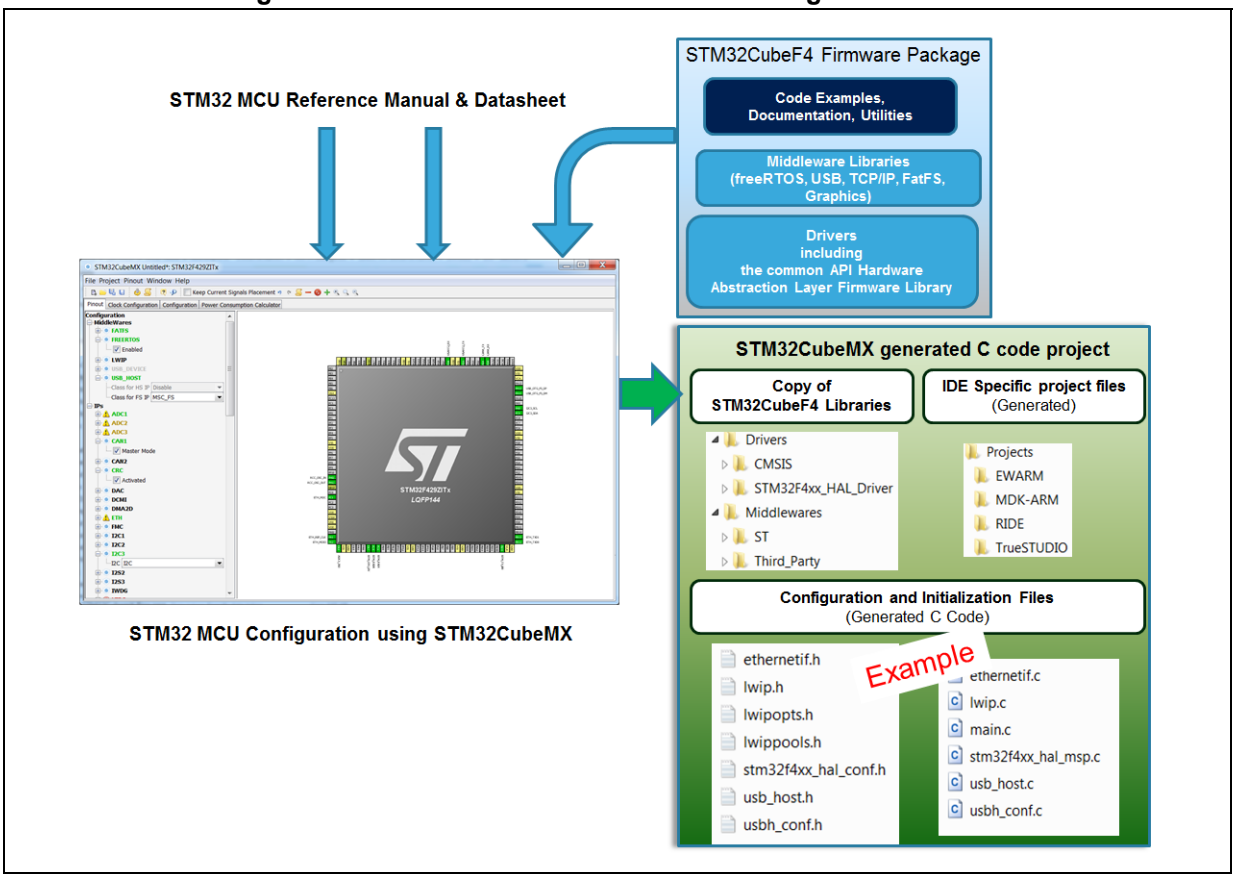
- Software reuse and application design portability are achieved through STM32Cube firmware solution proposing a common Hardware Abstraction Layer API across STM32 portfolio.
- Optimized migration time is achieved thanks to STM32CubeMX built-in knowledge of STM32 microcontrollers, peripherals and middleware (LwIP and USB communication protocol stacks, FatFs file system for small embedded systems, FreeRTOS).

STM32CubeMX graphical interface performs the following functions:

- Fast and easy configuration of the MCU pins, clock tree and operating modes for the selected peripherals and middleware
- Generation of pin configuration report for board designers
- Generation of a complete project with all the necessary libraries and initialization C code to set up the device in the user defined operating mode. The project can be directly open in the selected application development environment (for a selection of supported IDEs) to proceed with application development (see *Figure 1*).

During the configuration process, STM32CubeMX detects conflicts and invalid settings and highlights them through meaningful icons and useful tool tips.

Figure 1. Overview of STM32CubeMX C code generation flow



2.2 Key features

STM32CubeMX comes with the following features:

- **Project management**

  STM32CubeMX allows creating, saving and loading previously saved projects:

  – When STM32CubeMX is launched, the user can choose to create a new project or to load a previously saved project.

  – Saving the project saves user settings and configuration performed within the project in an .ioc file that will be used the next time the project will be loaded in STM32CubeMX.

  STM32CubeMX also allows importing previously saved projects in new projects.

  STM32CubeMX projects come in two flavors:

  – MCU configuration only: .ioc file are saved anywhere, next to other .ioc files.

  – MCU configuration with C code generation: in this case .ioc files are saved in a dedicated project folder along with the generated source C code. There can be only one .ioc file per project.

- **Easy MCU and STMicroelectronics board selection**

  When starting a new project, a dedicated window opens to select either a microcontroller or an STMicroelectronics board from STM32 portfolio. Different filtering options are available to ease the MCU and board selection.

- **Easy pinout configuration**

  – From the **Pinout** view, the user can select the peripherals from a list and configure the peripheral modes required for the application. STM32CubeMX assigns and configures the pins accordingly.

  – For more advanced users, it is also possible to directly map a peripheral function to a physical pin using the **Chip** view. The signals can be locked on pins to prevent STM32CubeMX conflict solver from moving the signal to another pin.

  – Pinout configuration can be exported as a .csv file.

- **Complete project generation**

  The project generation includes pinout, firmware and middleware initialization C code for a set of IDEs. It is based on STM32Cube embedded software libraries. The following actions can be performed:

  – Starting from the previously defined pinout, the user can proceed with the configuration of middleware, clock tree, services (RNG, CRC, etc...) and IP peripheral parameters. STM32CubeMX generates the corresponding initialization C code. The result is a project directory including generated main.c file and C header files for configuration and initialization, plus a copy of the necessary HAL and middleware libraries as well as specific files for the selected IDE.

  – The user can modify the generated source files by adding user-defined C code in user dedicated sections. STM32CubeMX ensures that the user C code is preserved upon next C code generation (the user C code is commented if it is no longer relevant for the current configuration).

  – STM32CubeMX can generate user files by using user-defined freemarker .ftl template files.

  – From the Project settings menu, the user can select the development tool chain (IDE) for which the C code has to be generated. STM32CubeMX ensures that the IDE relevant project files are added to the project folder so that the project can be

directly imported as a new project within third party IDE (IAR™ EWARM, Keil™ MDK-ARM, Atollic® TrueSTUDIO and AC6 System Workbench for STM32).

- **Power consumption calculation**

  Starting with the selection of a microcontroller part number and a battery type, the user can define a sequence of steps representing the application life cycle and parameters (choice of frequencies, enabled peripherals, step duration). STM32CubeMX Power Consumption Calculator returns the corresponding power consumption and battery life estimates.

- **Clock tree configuration**

  STM32CubeMX offers a graphical representation of the clock tree as it can be found in the device reference manual. The user can change the default settings (clock sources, prescaler and frequency values). The clock tree is then updated accordingly. Invalid settings and limitations are highlighted and documented with tool tips. Clock tree configuration conflicts can be solved by using the solver feature. When no exact match is found for a given user configuration, STM32CubeMX proposes the closest solution.

- **Automatic updates of STM32CubeMX and STM32Cube firmware packages**

  STM32CubeMX comes with an updater mechanism that can be configured for automatic or on-demand check for updates. It supports STM32CubeMX self-updates as well as STM32Cube firmware library package updates. The updater mechanism also allows deleting previously installed packages.

- **Report generation**

  .pdf and .csv reports can be generated to document user configuration work.

2.3 Rules and limitations

- C code generation covers only peripheral and middleware initialization. It is based on STM32Cube HAL firmware libraries.
- STM32CubeMX C code generation covers only initialization code for peripherals and middlewares that use the drivers included in STM32Cube embedded software packages. The code generation of some peripherals and middlewares, such as cryptographic IPs and StemWin graphic library, is not yet supported.
- Refer to *Appendix A* for a description of pin assignment rules.
- Refer to *Appendix B* for a description of STM32CubeMX C code generation design choices and limitations.

3 Installing and running STM32CubeMX

3.1 System requirements

3.1.1 Supported operating systems and architectures

- Windows® XP: 32-bit (x86)
- Windows® 7: 32-bit (x86), 64-bit (x64)
- Windows® 8: 32-bit (x86), 64-bit (x64)
- Linux®: 32-bit (x86) and 64-bit (x64) (tested on RedHat, Ubuntu and Fedora)
- MacOS: 64-bit (x64) (tested on OS X Yosemite)

3.1.2 Memory prerequisites

- Recommended minimum RAM: 2 Gbytes.

3.1.3 Software requirements

The following software must be installed:

- Java Run Time Environment for 1.7.0_45

  If Java is not installed on your computer or if you have an old version, STM32CubeMX installer will open the Java download web page and stop.
- For Eclipse plug-in installation only, install one of the following IDE:
  - Eclipse IDE Juno (4.2)
  - Eclipse Luna (4.4)
  - Eclipse Kepler (4.3)
  - Eclipse Mars (4.5)

3.2 Installing/uninstalling STM32CubeMX standalone version

3.2.1 Installing STM32CubeMX standalone version

To install STM32CubeMX, follow the steps below:

1. Download STM32CubeMX installation package from www.st.com/stm32cubemx.
2. Extract (unzip) stm32cubemx.zip whole package into the same directory.
3. Check your access rights and launch the installation wizard:

   On windows:

   a) Make sure you have administrators rights.

   b) Double click the SetupSTM32CubeMX-VERSION.exe file to launch the installation wizard.

   On Linux:

   a) Make sure you have access rights to the target installation director. You can run the installation as root (or sudo) to install STM32CubeMX in shared directories.

b) Double click (or launch from the console window) on the SetupSTM32CubeMX-VERSION.linux file.

On MacOS:

a) Make sure you have administrators rights.

b) Double click SetupSTM32CubeMX application file to launch the installation wizard.

4. Upon successful installation of STM32CubeMX on Windows, STM32CubeMX icon is displayed on your desktop and STM32CubeMX application is available from the Program menu. STM32CubeMX .ioc files are displayed with a cube icon. Double-click them to open up them using STM32CubeMX.
5. Delete the content of the zip from your disk.

Note: If the proper version of the Java Runtime Environment (version 1.7_45 or newer) is not installed, the wizard will propose to download it and stop. Restart STM32CubeMX installation once Java installation is complete. Refer to Section 9: FAQ for issues when installing the JRE.

When working on Windows, only the latest installation of STM32CubeMX will be enabled in the program menu. Previous versions can be kept on your PC (not recommended) when different installation folders have been specified. Otherwise, the new installation overwrites the previous ones.

3.2.2 Installing STM32CubeMX from command line

There are 2 ways to launch an installation from a console window: either in console interactive mode or via a script.

Interactive mode

To perform interactive installation, type the following command:

```
java -jar SetupSTM32CubeMX-4.14.0.exe -console
```

At each installation step, an answer is requested (see *Figure 2* below).

Figure 2. Example of STM32CubeMX installation in interactive mode