



SHORT MANUAL ON THE USE OF ARMD

Rev. 1.0.0 – February 17, 2014

© Digital Technology Art S.r.l.

INDEX

[INTRODUCTION](#)

[INSTALLATION](#)

[PLUG-IN SETTINGS](#)

[CREATING A NEW PROJECT](#)

[CONTEXTUAL MENU](#)

[SETTINGS](#)

[PROJECT SCHEME](#)

[CONFIG SCHEME](#)

[CONFIG MAKE SCHEME](#)

[CONFIG STACK SCHEME](#)

[CONFIG LIBRARY SCHEME](#)

[CONFIG INCLUDE SCHEME](#)

[CONFIG MACRO SCHEME](#)

[HARDWARE SCHEME](#)

[HARDWARE MEMORY SCHEME](#)

[HARDWARE CLOCK SCHEME](#)

[HARDWARE CLOCK PERIPHERALS SCHEME](#)

[HARDWARE PERIPHERALS POWER SCHEME](#)

[HARDWARE PIN FUNCTION SCHEME](#)

[HARDWARE PIN MODE SCHEME](#)

[HARDWARE PIN CONFIG SCHEME](#)

[CONCLUSIONS](#)

INTRODUCTION

mdi has a plug-in called **ARMD** that implements an IDE for the **ARM GNU Toolchain**.

This plug-in is always present in **mdi**; it is meant to develop applications or libraries for ARM microcontrollers that are present in Digital Technology Art devices.

The development environment allows to:

- Create a project file
- Edit C sources
- Compile and link the complete project
- Upload binary in Flash
- Start a console in order to debug the program
- Edit and upload GUI in the expansion Flash

INSTALLATION

All you need to do to install the application is to run the setup program *MDI_Setup.exe* and follow the instructions on the installation pages.

After that you need to apply for a **mdi** license. Without this license you will not be able to save or modify files you created.

In order to obtain the - completely free of charge - license you can send an e-mail to:

info@digitaltechnologyart.com


The e-mail message consists of three lines:

Company/Institute

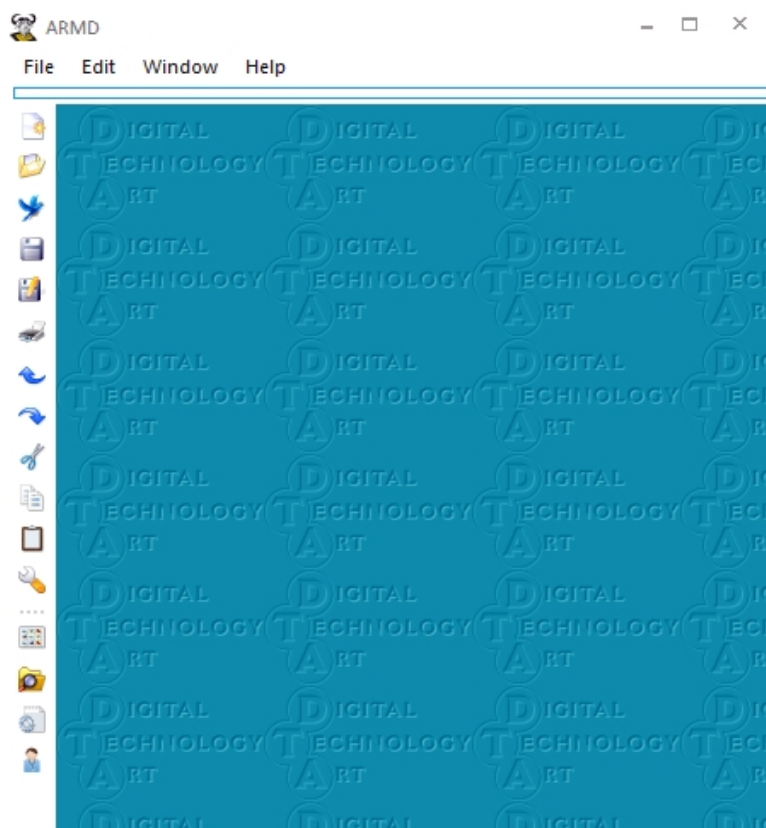
First and last name

E-mail

Within a few hours you will receive an XML file that will overwrite the one present in the CONFIG folder in the BIN directory.

Now you can run **mdi** by double clicking the  icon on the desktop.

The following screen will appear:



If the following icon is shown:



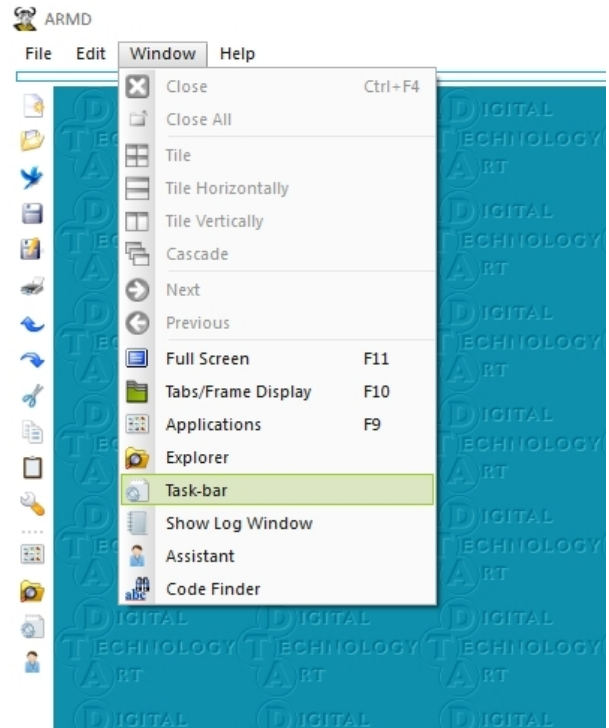
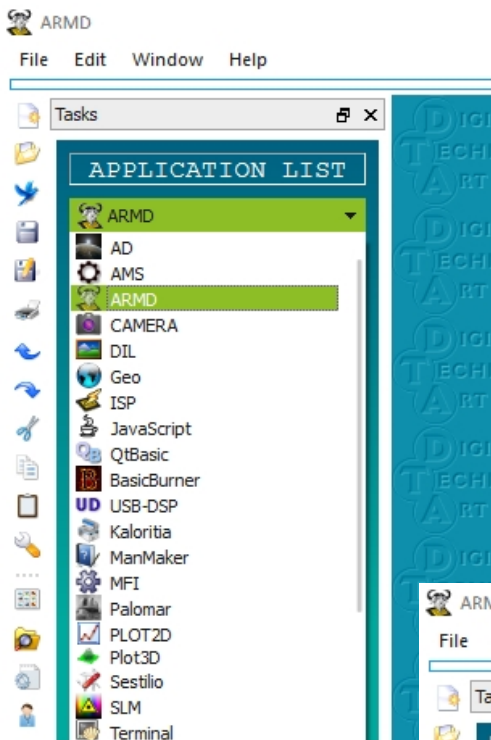
you can skip the following part, as **mdi** is ready for use.

PLUG-IN SETTINGS

We will now set **ARMD** as the initial **mdi** application.

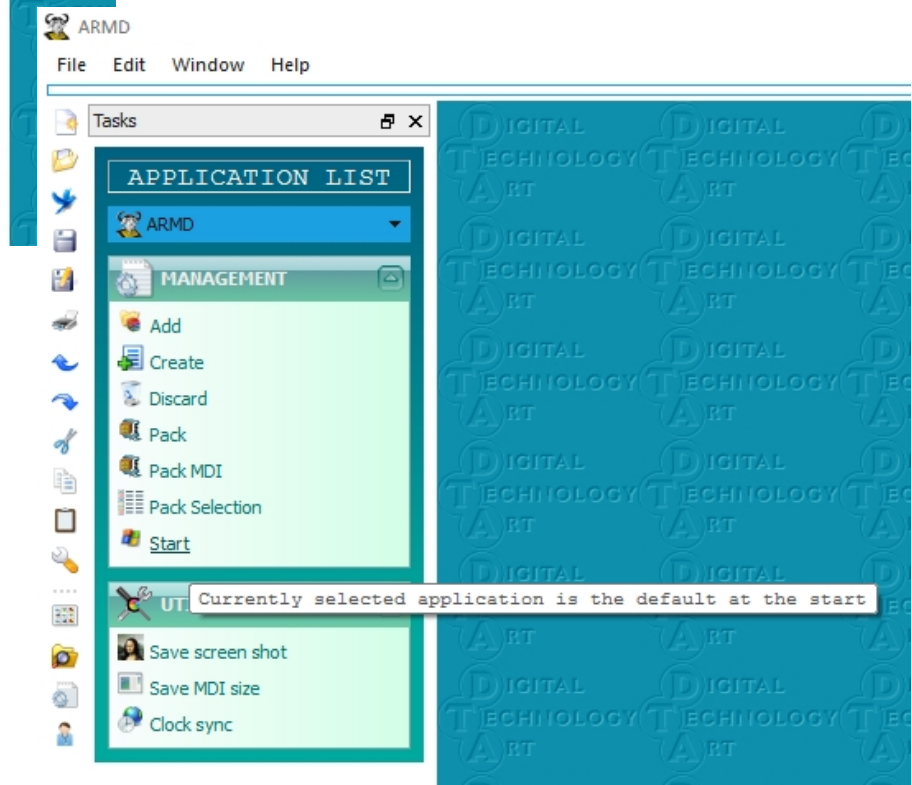
First select the **Task-bar** tab from the **Window** menu:

The Task-bar screen will open and you can select **ARMD** from the combo-box at the top (APPLICATION LIST).




Then select **Start** in the **MANAGEMENT** group.

From now on, **mdi** will always be ready to perform with **ARMD**, unless you decide to change



CREATING A NEW PROJECT

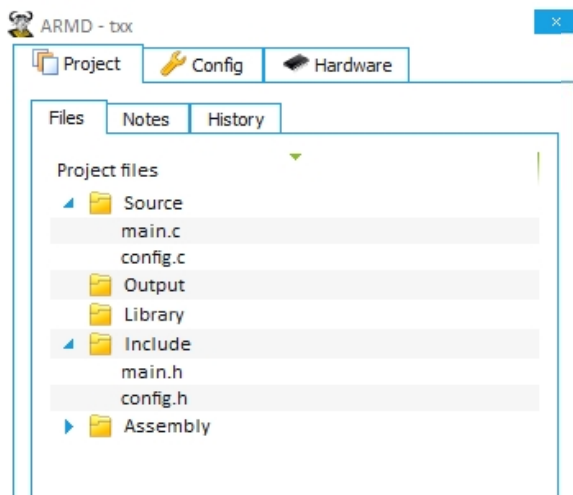
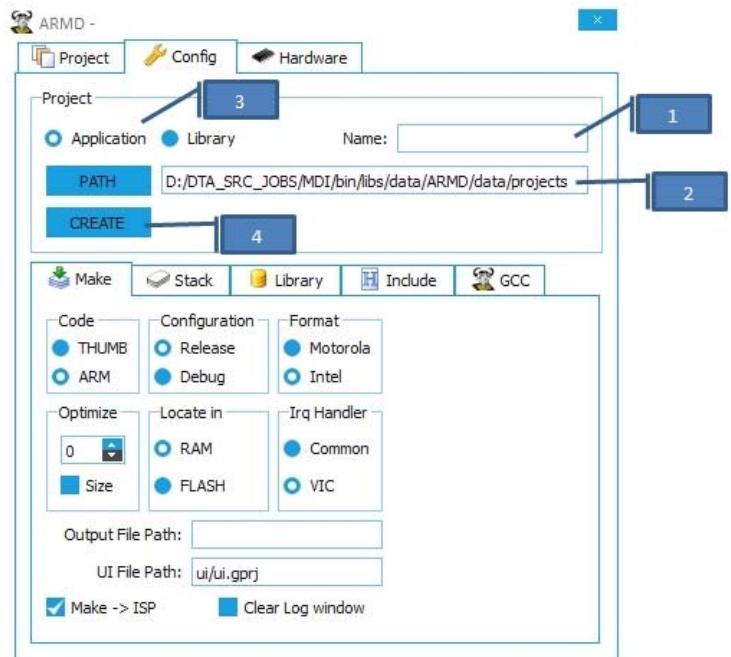
Select the  icon or **File > New**
The following screen will appear:

It is now important to specify:

1. The name you want to give the project you are going to create
2. The path where the project will be created. For now, we advise you to agree with the proposed suggestions.
3. Whether you want to create a library or an application.
4. Finally, click this button to complete the creation of the project.

Subsequently you can modify all remaining controls and settings at any time.

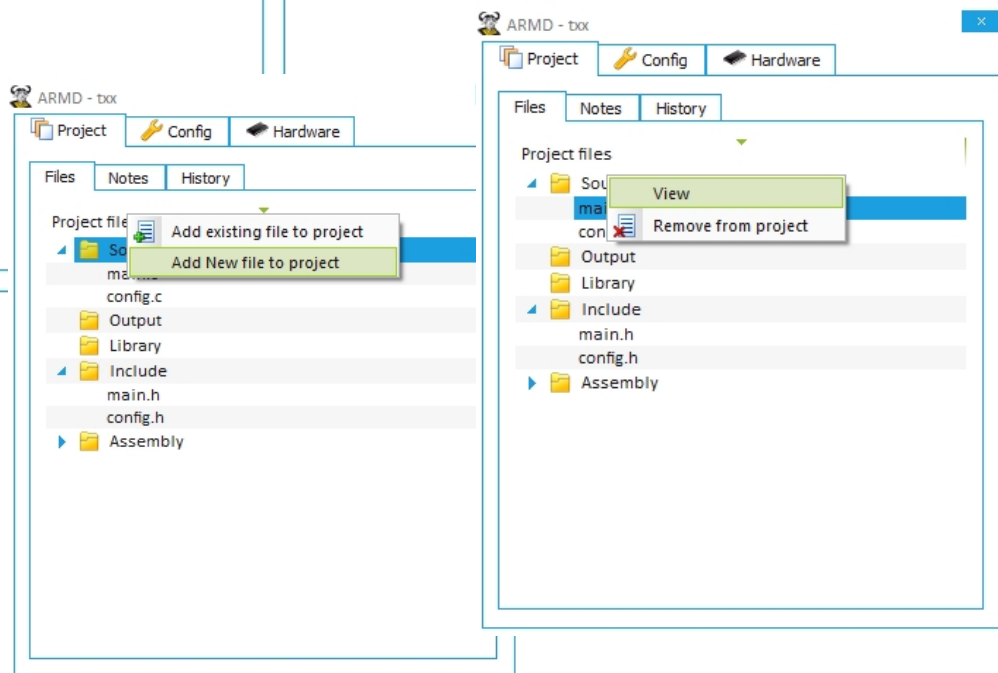
The **ARMD** first page will now appear, where the tree of files that are present in the project will be shown.



CONTEXTUAL MENU

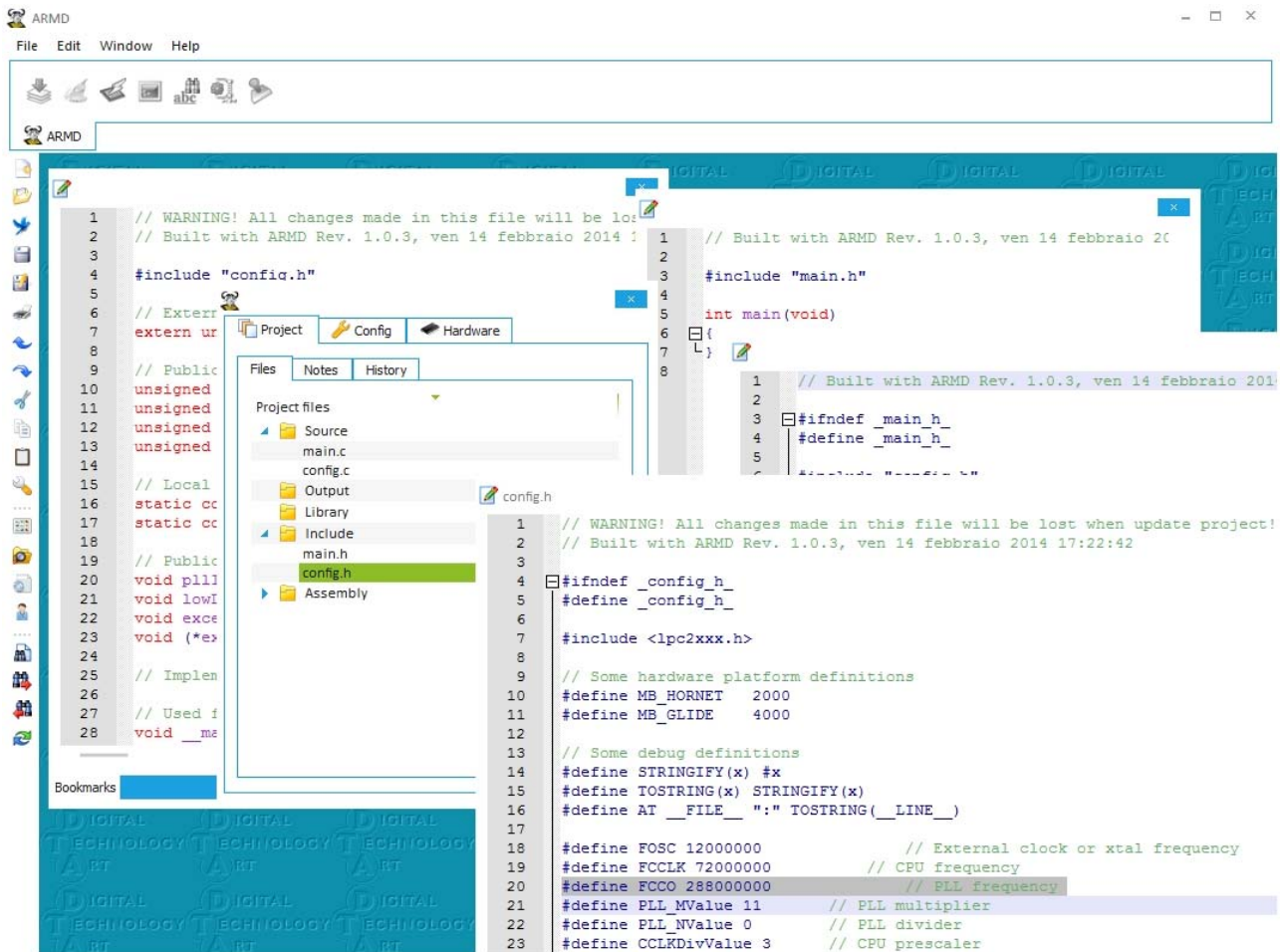
It is possible to interact with the list of files and folders of the project tree through mouse interaction:

1. When double clicking a file, it will be opened in a text editor.
2. When selecting a folder or a file by right-clicking the mouse, a contextual menu will appear.

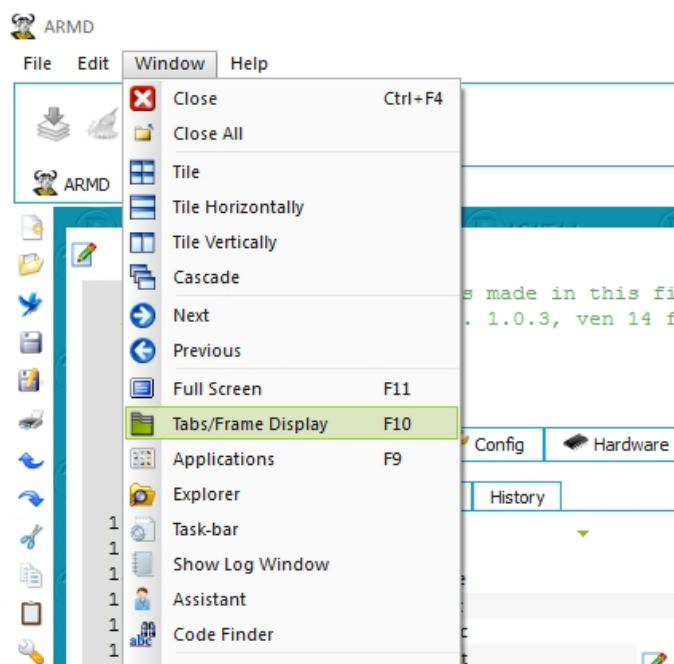




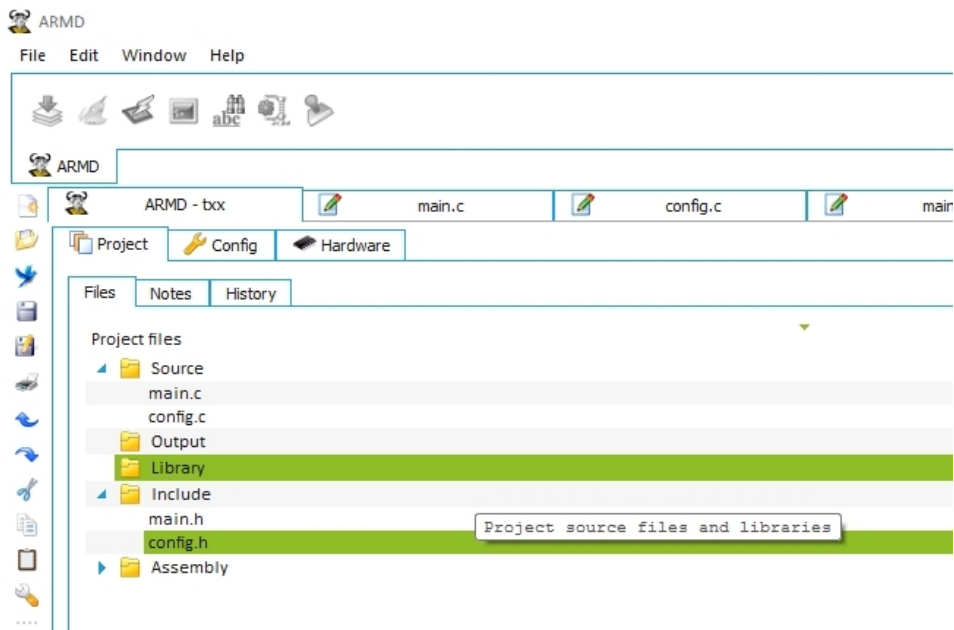
When opening many project files, your desktop could get really crowded, which may slow down the screen display.



We advise you to select the following display mode for a better organized desktop.



When selecting the following display mode, you will obtain ...



From the list of present files various new files will be created automatically:

config.h config.c main.h main.c

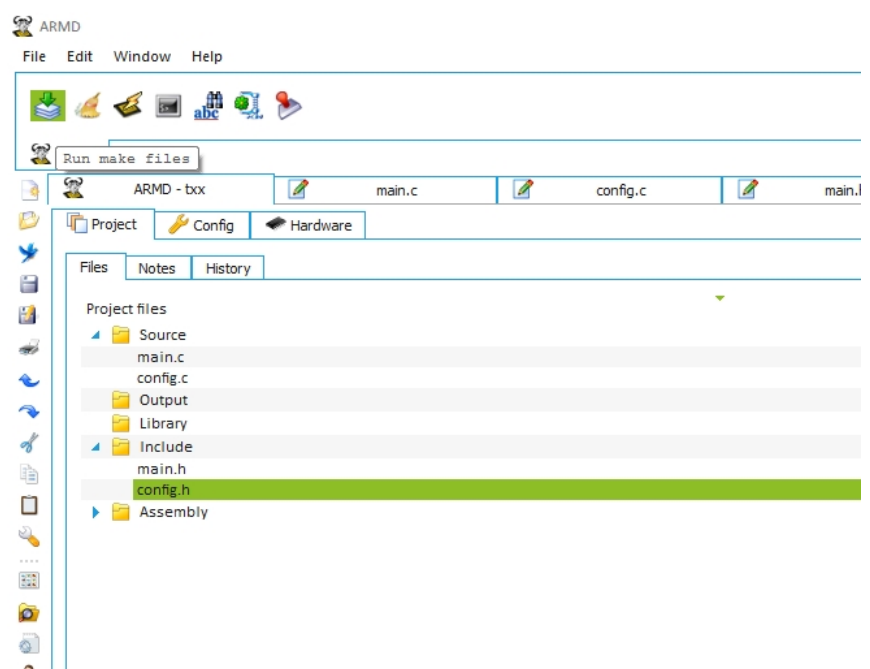
They are the project's base files. One file in particular: **config** contains all hardware settings of the microcontroller and the project related software.

We advise you against modifying the file, as it will be overwritten every time the hardware settings are changed!

Now select the **ARMD** directory and run the **BUILD** command from the icon.

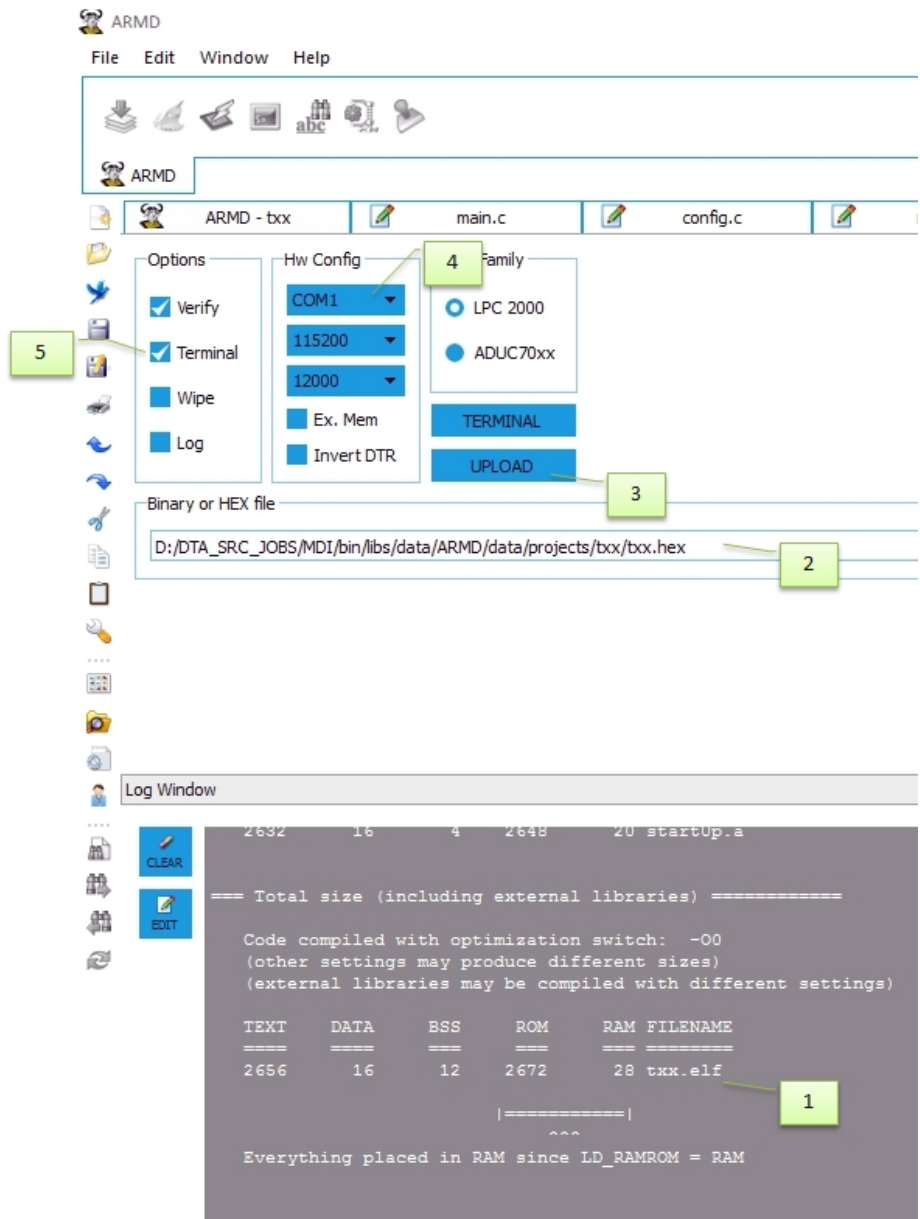
All project files will be compiled and linked. In the end (when no errors occur) a binary file will be created (.HEX)

During compilation a log screen will be activated showing all diagnostic and informative messages of the compiler and linker. At the end of the process a second **mdi** application called **ISP** will be activated automatically. It will allow you to upload instantly the created firmware in FLASH of your hardware.



At the end of the compilation a screen like the following will be shown:

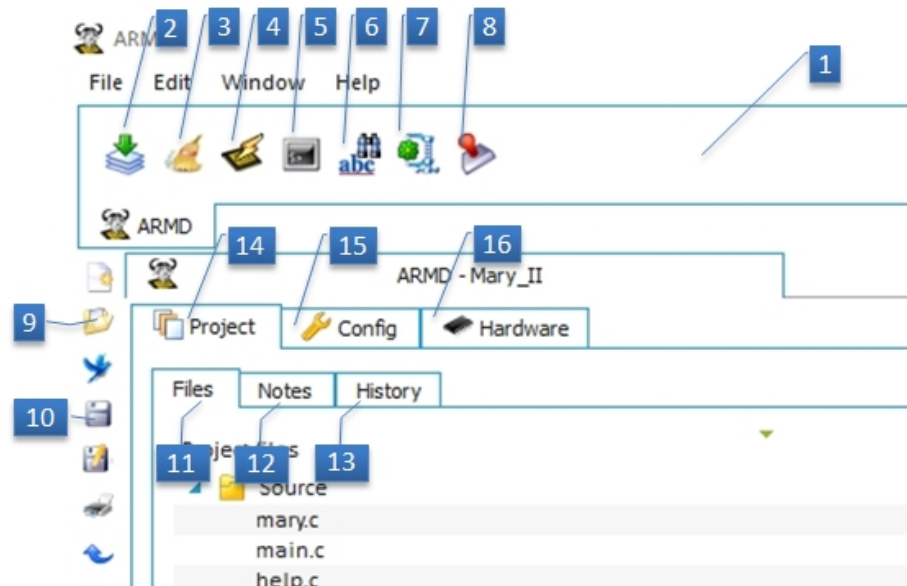
1. Final account of the construction process, the memory used by the product code is shown.
2. Location of the binary product.
3. UPLOAD button. By clicking it the program will search in the selected COM (3) the microcontroller and start uploading the code.
4. COM port where your hardware is connected in order to upload the firmware. Attention: the GLIDE board requires that the appropriate programming cable is plugged into the Y10 connector.
5. If this option is activated at the end of the upload process, an application (TERMINAL) will be opened, allowing you to communicate with the loaded program.



SETTINGS

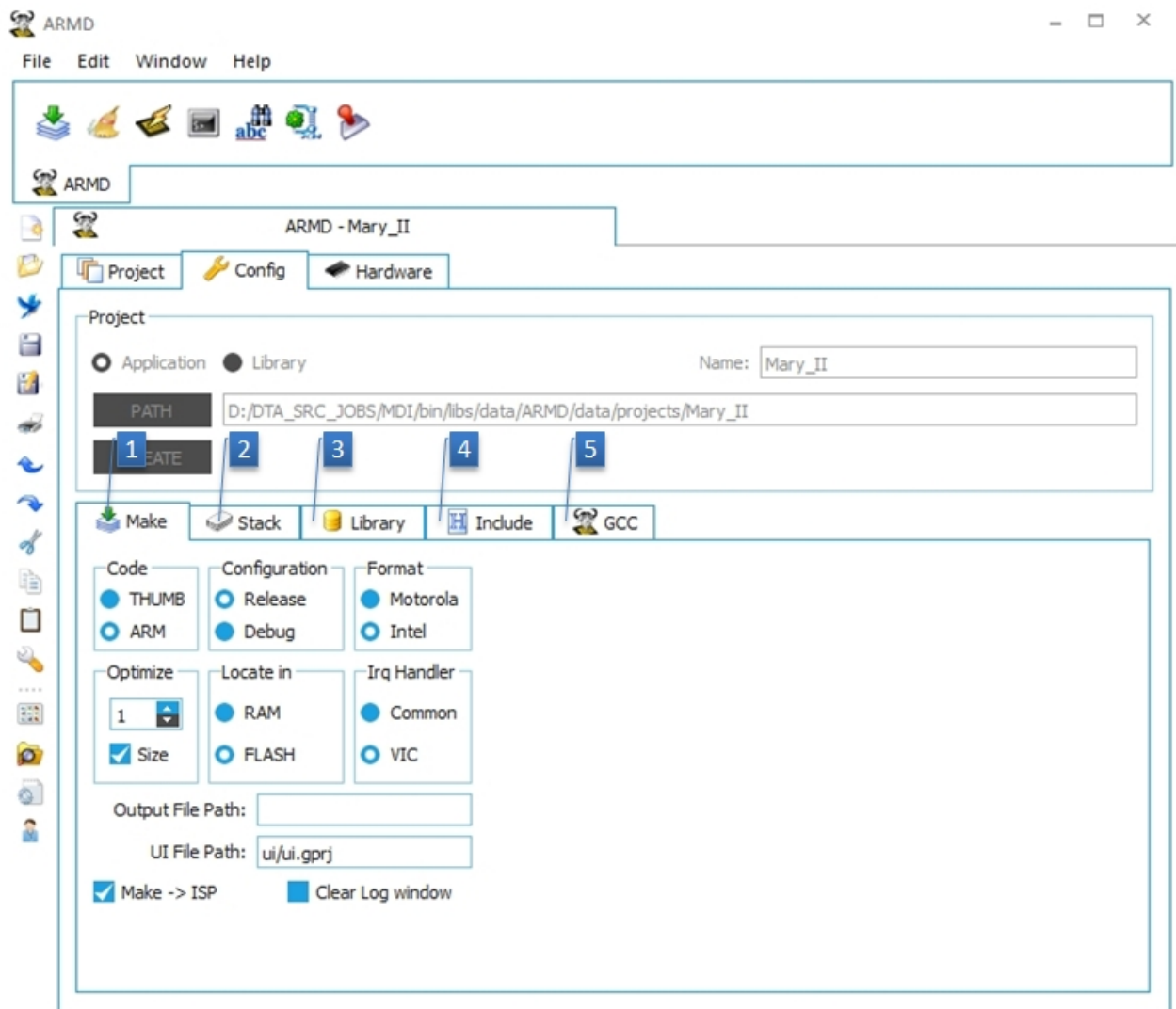
ARMD has various pages that are dedicated to software and hardware configuration. In the following, we will examine them more closely:

PROJECT SCHEME



- 1 **ARMD** Toolbar
- 2 Compiles and links all sources and libraries
- 3 Deletes all compiled object files by compiling all modules
- 4 Starts **ISP**, the application for the upload of a binary in FLASH or to open the terminal
- 5 Uploads **MONITOR** in FLASH. **MONITOR** is a basic management program, required for the functioning of some applications
- 6 **GREP**, a function that searches a text in all source modules
- 7 Starts the execution of **GPACK**, a GUI building tool
- 8 Uploads a created GUI in the expansion FLASH (4Mbyte)
- 9 Opens a project
- 10 Save the project
- 11 Folder that visualizes the project files
- 12 Editor containing text notes regarding the project
- 13 Editor containing information related to the project's revision history
- 14 Folder containing project files
- 15 Folder that enters the code creating settings
- 16 Folder that enters the settings of the microcontroller peripherals

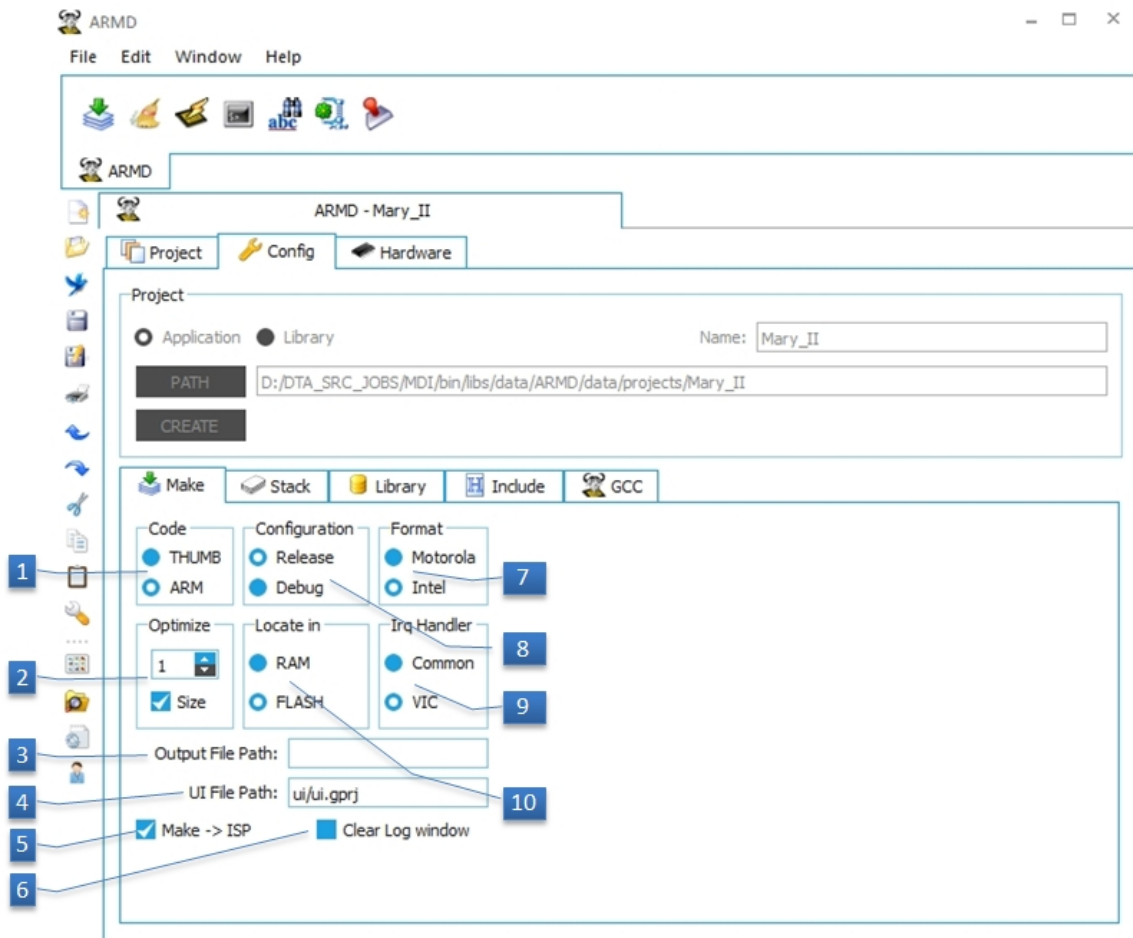
CONFIG SCHEME



This scheme manages the compiling and linking options.

1. Folder for the code creating procedure
2. Stack pointer configuration
3. Applied libraries
4. File Inclusion Search
5. GCC options

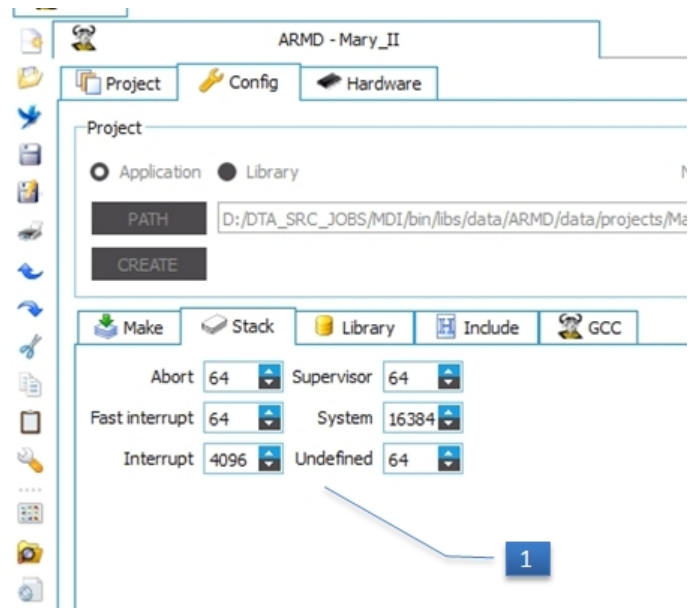
CONFIG MAKE SCHEME



1. OP-CODE size choice, all our libraries are ARM.
2. Code optimizer setting (we advise against using different values).
3. Path (complete with file name extension). This option is particularly for those library projects that, once created, you want to copy to another location.
4. Specifies the project file for the graphical interface.
5. When at the end of the compilation this control is activated and no errors occur, the ISP program will start automatically in order to upload the new code in the FLASH microcontroller.
6. When activating this control, the LOG screen will be automatically cleaned at the start of every compilation.
7. Binary output file size. It needs to be iNTEL at all times.
8. When selecting DEBUG, the required data for the program tracing will also be reported in the binary output file. It is useless if you don't have a JTAG emulator.
9. Setting of interrupt management mode. For reasons of compatibility with our libraries, we advise to always use VIC.
10. Setting of the destination memory location where the product code will be uploaded. Use RAM for small code portions and a fast upload in binary.

CONFIG STACK SCHEME

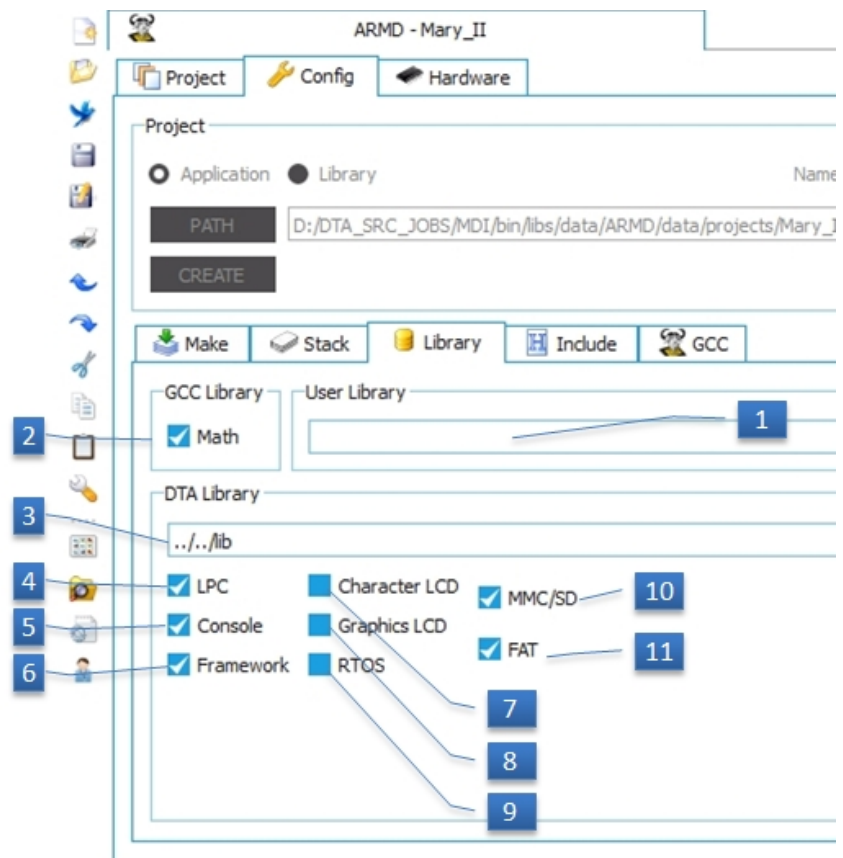
Scheme that allows to set STACK in all conditions provided by ARM



CONFIG LIBRARY SCHEME

Scheme that allows setting of the applied libraries.

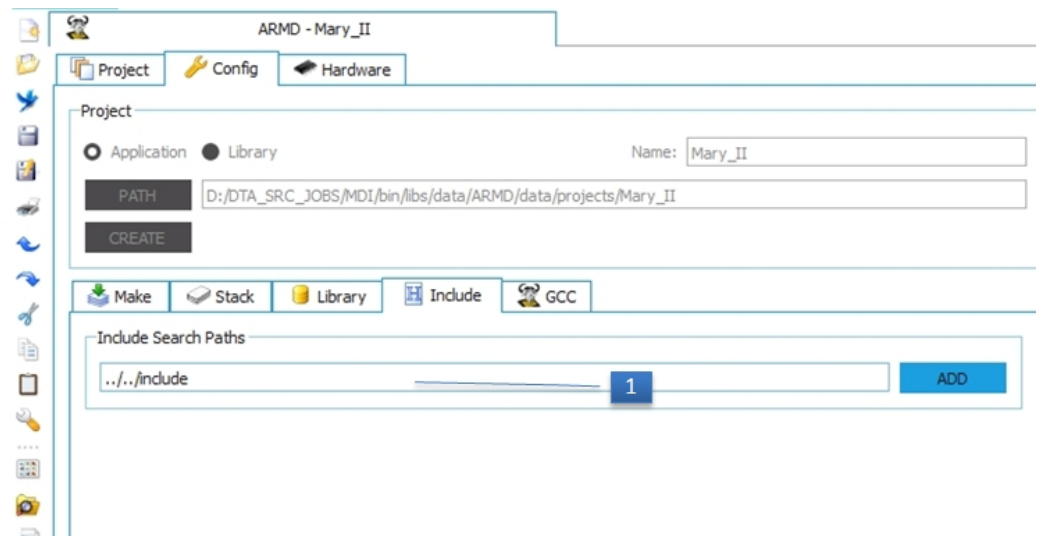
- 1 User library list (incl. extension) divided by a comma
- 2 When activated, this control includes in the connecting phase the GNU floating-point libraries
- 3 Pathname in order to reach our libraries
- 4 Control that activates LPC libraries meant to run the NXP microcontrollers
- 5 Control that activates the libraries for serial console management
- 6 Control that activates a library system
- 7 Control that activates the library for LCD character management
- 8 Control that activates the library for graphic LCD management
- 9 Control that activates the RTOS library
- 10 Control that activates the Secure Digital library
- 11 Control that activates the FAT library



CONFIG INCLUDE SCHEME

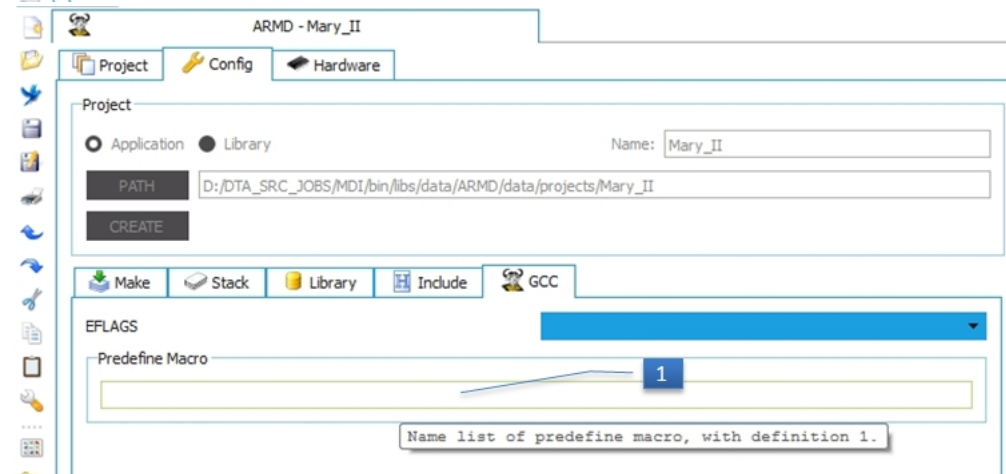
Scheme allowing to set the locations where to search the used include files.

The various locations are divided by a comma.

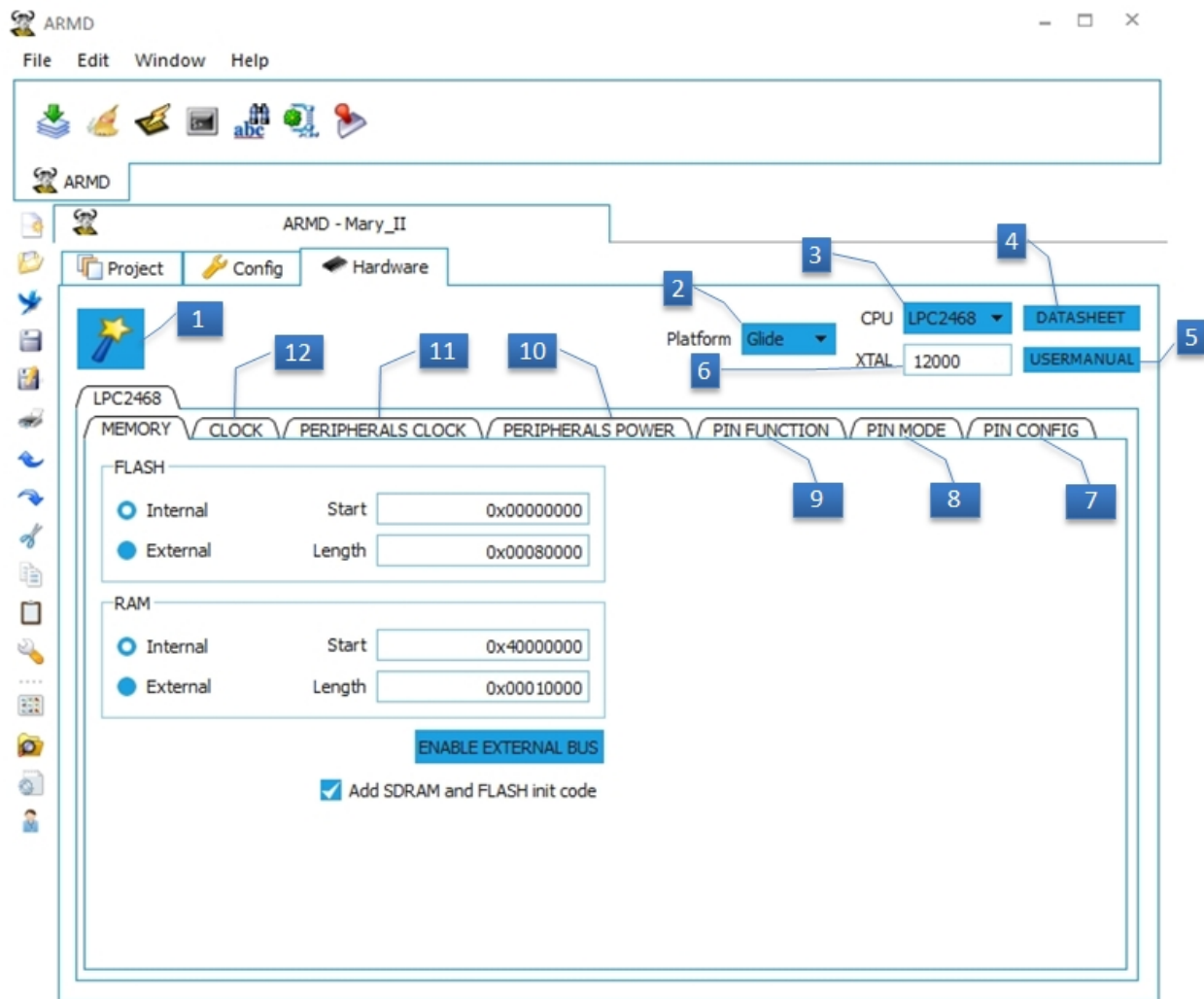


CONFIG MACRO SCHEME

Scheme allowing to define macros used in the source files.



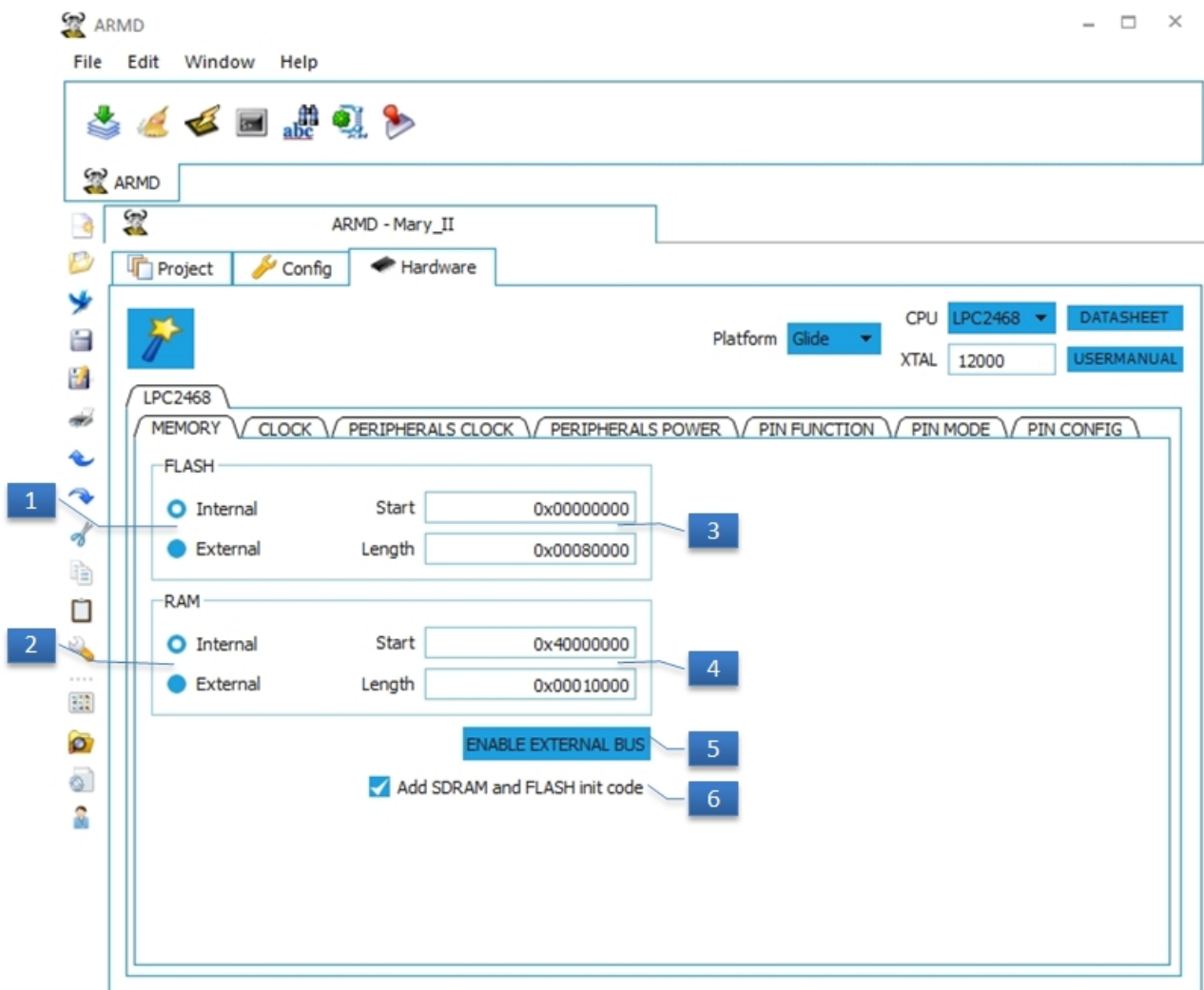
HARDWARE SCHEME



This scheme allows you to configure the microcontroller peripherals and to set some specific parameters of the used hardware. All settings made in these folders will then be converted to C code and saved in the config.c and config.h files.

1. WIZARD button, opens a screen allowing to set the user mode of our hardware in a few steps.
2. Selection of the used hardware platform.
3. Selection of the used microcontroller.
4. Button that allows to visualize the datasheet of the used microcontroller
5. Button that allows to visualize the user manual of the used microcontroller
6. Quartz frequency used by the microcontroller
7. Hardware configuration scheme of the microcontroller pins
8. User mode of the microcontroller pins
9. Selection of function type of the microcontroller pins
10. Activation of the various hardware functions of the microcontroller
11. Operating frequency of the various hardware functions of the microcontroller
12. Operating frequency of the microcontroller

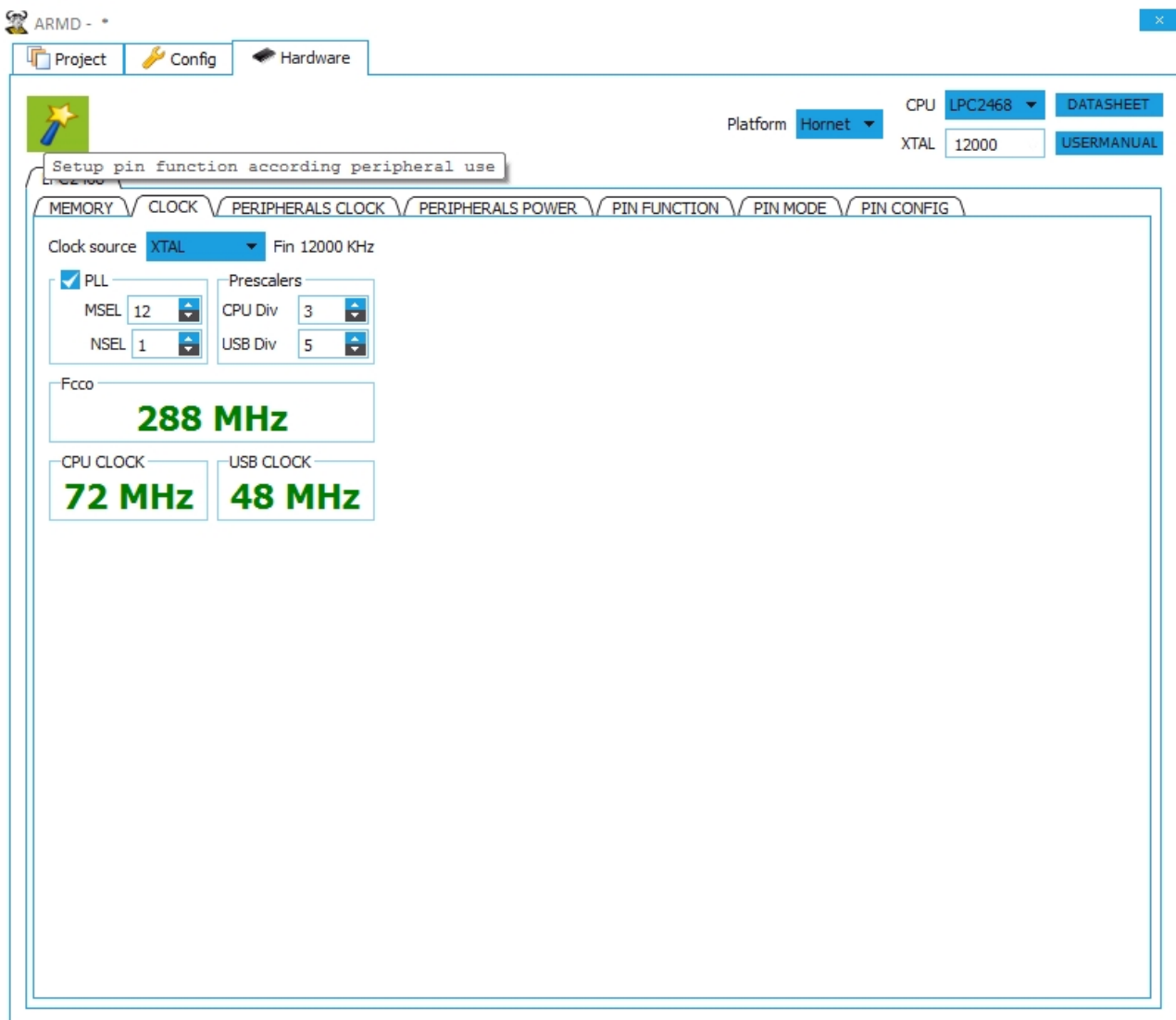
HARDWARE MEMORY SCHEME



This scheme allows you to configure the hardware memory and to indicate its use.

1. Specifies if the product code is uploaded in the microcontrollers internal FLASH memory or in the external expanded memory.
2. Specifies if the RAM memory used by the program is the microcontroller internal memory or the external expanded memory.
3. Specifies the start address and the used FLASH memory size.
4. Specifies the start address and the used RAM memory size.
5. When activated, this button automatically sets the pin functions to activate the external expansion bus. Basically, the external microcontroller bus is activated by setting the required function (data, addresses, strobes, etc.) of the various pins.
6. Adds the start code for the expanded RAM and FLASH.

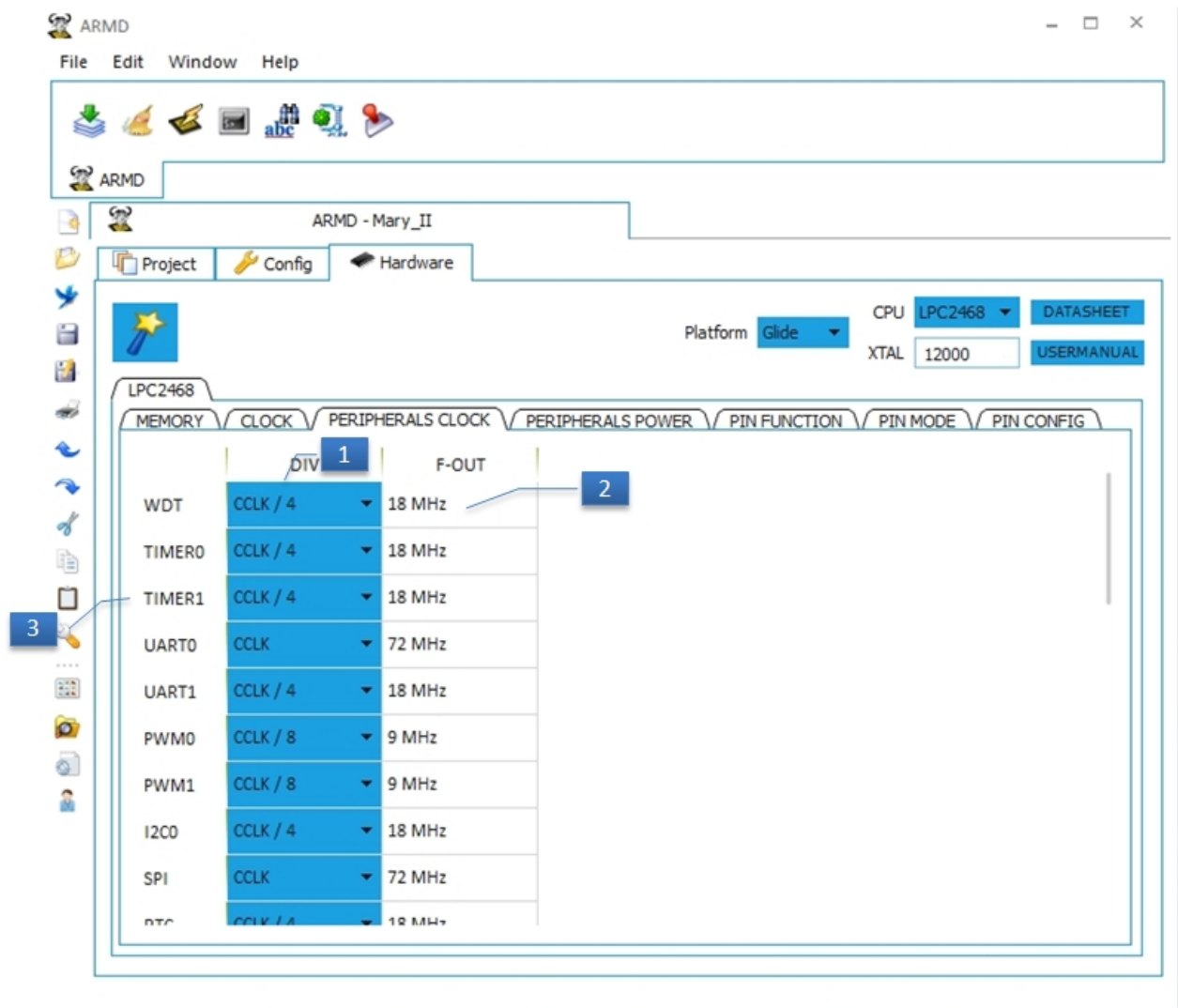
HARDWARE CLOCK SCHEME



Scheme allowing the setting of the microcontroller operating frequency.

1. Time base primary source
2. PLL settings
3. Achieved frequency based on the settings

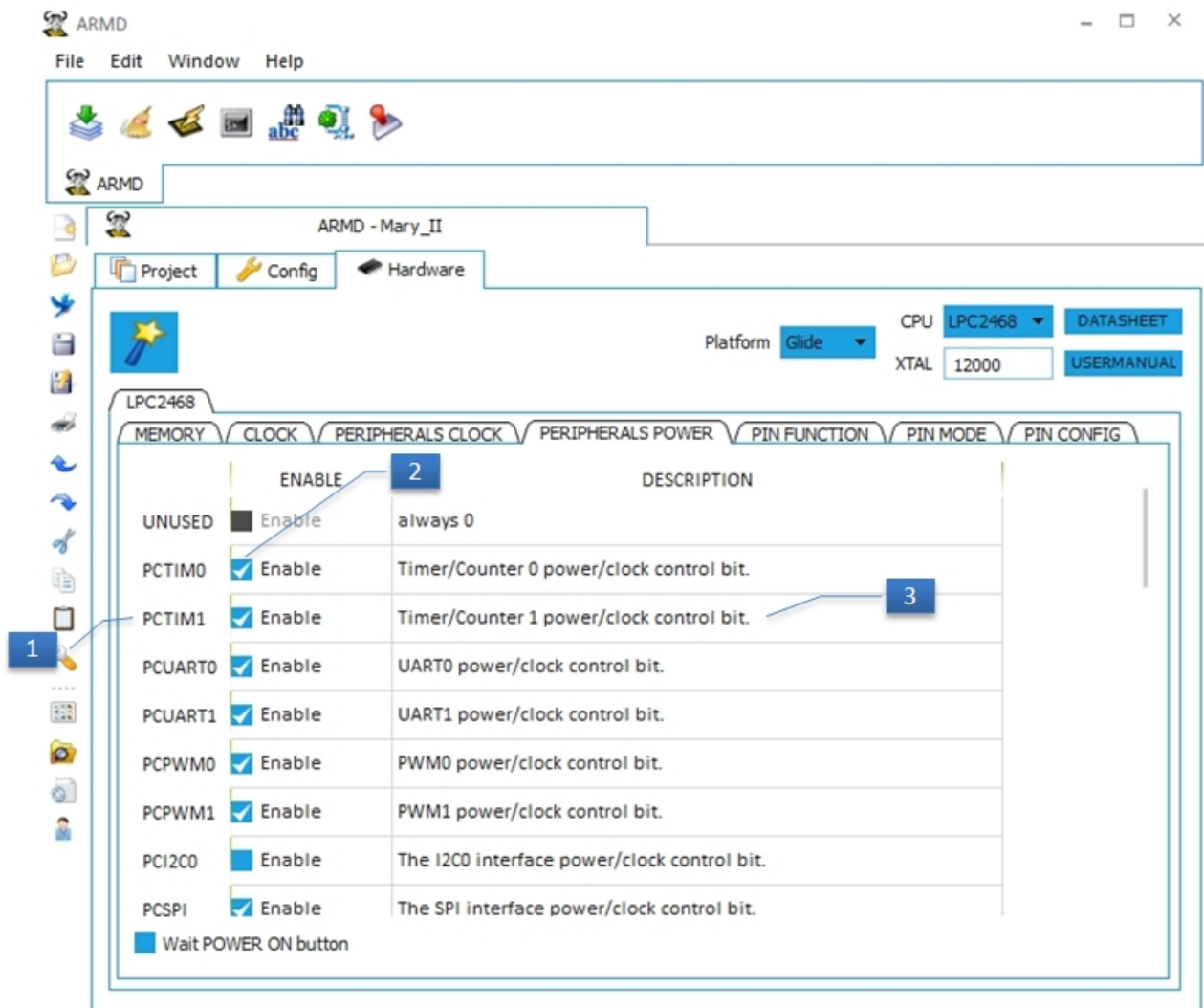
HARDWARE CLOCK PERIPHERALS SCHEME



This scheme allows to set the clock to all microcontroller peripherals.

1. Selection of dividing factor
2. Frequency reaching the peripheral
3. Peripheral

HARDWARE PERIPHERALS POWER SCHEME



This scheme allows to choose which microcontroller's internal peripheral will be power supplied in order to reduce electricity consumption.

1. Name of peripheral
2. Control
3. Description

HARDWARE PIN FUNCTION SCHEME

ARM - *
Project Config Hardware

Platform: Hornet CPU: LPC2468 XTAL: 12000
DATASHEET USERMANUAL

LPC2468
MEMORY CLOCK PERIPHERALS CLOCK PERIPHERALS POWER PIN FUNCTION PIN MODE PIN CONFIG

PORT 0 PORT 1 PORT 2 PORT 3 PORT 4

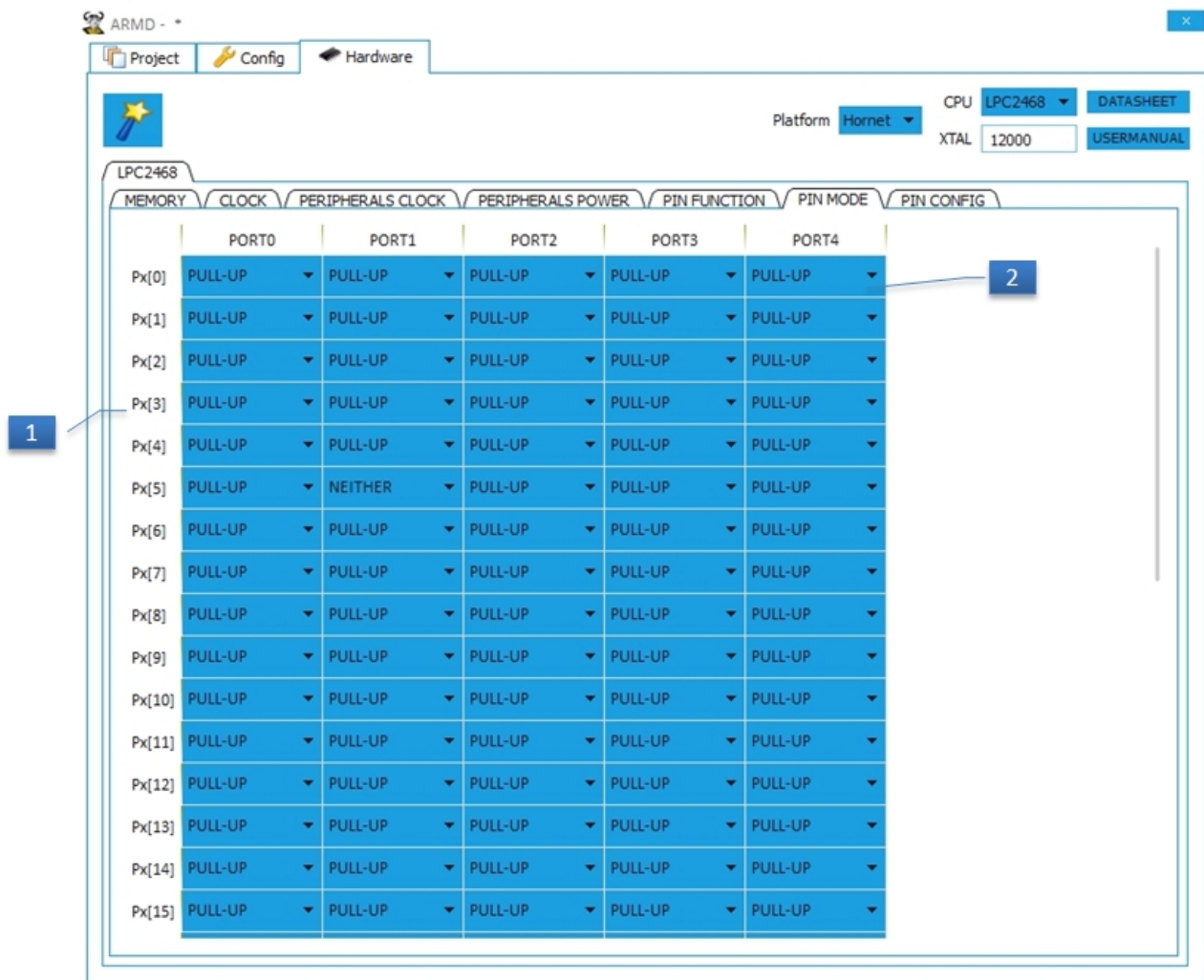
PIN	FUNCTION	LABEL	USAGE
P0[0]	GPIO	H_NAND_FLASH_READY	As input for NAND FLASH READY signal.
P0[1]	GPIO	H_NOR_FLASH_READY	As input for NOR FLASH READY signal.
P0[2]	TXD0	H_TXD0	As output for transmit serial data of main communication serial port.
P0[3]	RXD0	H_RXD0	As input for receive serial data of main communication serial port.
P0[4]	GPIO	H_CTL0	As bidirectional for CTL0 signal of CY7C68013 USB interface.
P0[5]	GPIO	H_CTL1	As bidirectional for CTL1 signal of CY7C68013 USB interface.
P0[6]	GPIO	H_RDY0	As bidirectional for RDY0 signal of CY7C68013 USB interface.
P0[7]	GPIO	H_RDY1	As bidirectional for RDY1 signal of CY7C68013 USB interface.
P0[8]	GPIO		Free for user at pin Y3.58
P0[9]	GPIO		Free for user at pin Y3.57
P0[10]	GPIO		Free for user at pin Y3.60
P0[11]	GPIO		Free for user at pin Y3.59
P0[12]	GPIO	H_LCD_ON	Available for user at Y3.62 but also used, as output, to control LCD back light.
P0[13]	GPIO	H_LCD_CS	As output for the control of CS signal (active low) for graphics LCD.

Scheme allowing to set the function that needs to be applied to all microcontroller pins.

1. Microcontroller PIN
2. List of available functions
3. Label associated with the bit we can refer to (define) in the programming
4. Comment associated with the definition

This scheme can be repeated for all port groups of the microcontroller.

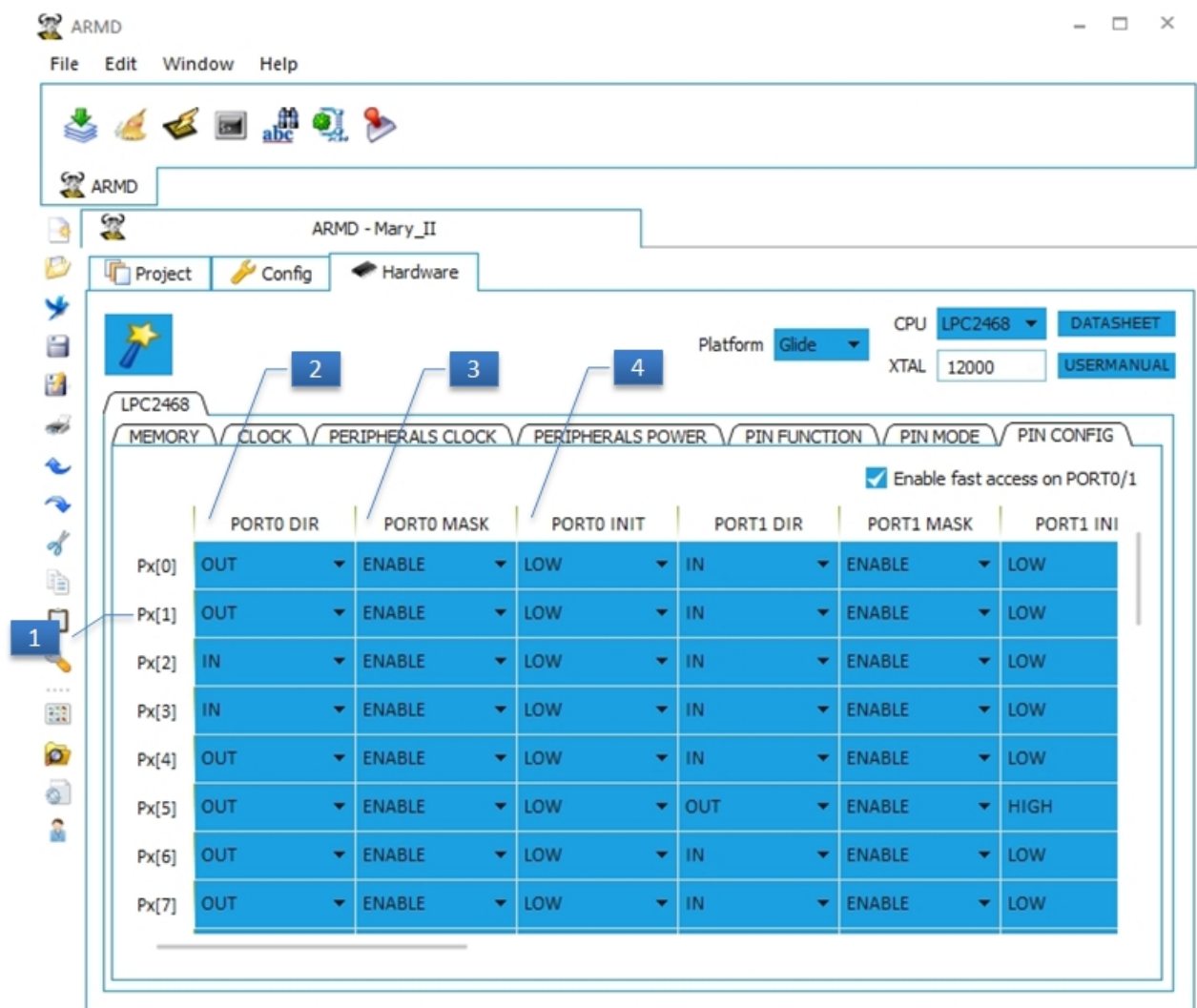
HARDWARE PIN MODE SCHEME



This scheme allows to set whether or not the resistance of the input pins.

1. Pin identification
2. Selection of PULL-UP, PULL-DOWN, NONE

HARDWARE PIN CONFIG SCHEME



Scheme allowing to define whether a pin is set to an input or output, what needs to be its initial state (in case of output) and, if necessary, whether it needs to be masked.

1. Pin identification
2. Direction
3. Mask
4. Initial value

CONCLUSIONS

This is not exactly a thorough manual, but merely a short introduction. For major details we like to refer to the GCC manual as an integrated part of the **mdi** help and to the microcontroller hardware manual reachable through **ARMD**.

Furthermore, we advise to always start with a sample project or with a STARTUP, usually supplied with our products.

Links:

<http://www.gnuarm.com/>

<http://www.yagarto.org/>



Digital Technology Art srl

Via G. Cei 100

56021 Cascina (PI)

Italy

www.digitaltechnologyart.com

info@digitaltechnologyart.com