# AN2361
# Application note

## Interfacing with the STR91x software library using Configuration and Programming Software (CAPS)

## Introduction

STR91x microcontrollers offer extremely flexible configuration of I/O routing and clock structures to optimize the use of each I/O pin and provide fine control of system clocks. The contents of a number of STR91x control registers are initialized by firmware at runtime to configure the I/O switch matrix on I/O Ports 0 through 9, and to set the initial values of system clock divisors.

STMicroelectronics provides the Configuration and Programming Software (CAPS) utility to graphically assist users in the selection of pin functions and clock divisors, and to generate a C header file with all the correct settings to integrate into the STR91x standard software library. CAPS makes it very easy to visualize the optimum mapping of communication interfaces on the I/O pins, reduces the chance of error and reduces the need to study the datasheet definitions of these control registers in detail.

CAPS is also an in-system programming tool that, via an external JTAG adapter, allows fast programming of STR9 in both development and production environments. Although this application note does not explore this topic, it describes the fundamental steps needed to interface with the STR91x Standard Library (STR91x StdLib). For more details about CAPS software functionality, please refer to *UM0225 - CAPS User Manual.*

The first part of this document focuses on how to use CAPS with the STR91x StdLib. The second part provides an example of how to change the clock frequencies and the GPIO pin configurations.

The source file, 91x_caps.c, which allows the interface between CAPS and the STR91x StdLib is provided with this application note.

## Glossary

This section gives a brief definition of acronyms and abbreviations used in this document:

***CAPS_project directory***: refers to CAPS project directory. By default it is located at: [CAPS Install directory\CAPS\Projects]

**PPP** or **ppp**: in the notation for file names, is used as a placeholder to stand for any peripheral acronym, such as ADC, CAN, DMA, etc.

***StdLib_project directory***: is the project directory in the STR91x StdLib package

***user_project directory***: refers to user project directory

## Related documents

The following topic related documents are available at www.st.com/mcu:

- UM0225 - CAPS User Manual for STR9
- UM0216 - STR91xF Reference Manual

This is only a select list of supporting documents. Refer to www.st.com/mcu for a complete list of STR91xF application notes.

# Contents

# 1 Interfacing CAPS with the STR91x StdLib

This section provides an overview of the tools you will need and the procedure for interfacing CAPS with the STR91xStdLib. Complete installation and user instructions for all software are found in their respective user manuals. All software, files and documentation are available for free download at *www.st.com/mcu*.

You will need to have downloaded and installed the following:

- **STR91x StdLib** of the STR91x Standard Software Library
- **CAPS** - Configuration and Programming Software (version 2.0 or later)
- **91x_caps.c** provided with this application note. For installation, see *Section 1.2*.

## 1.1 STR91x StdLib

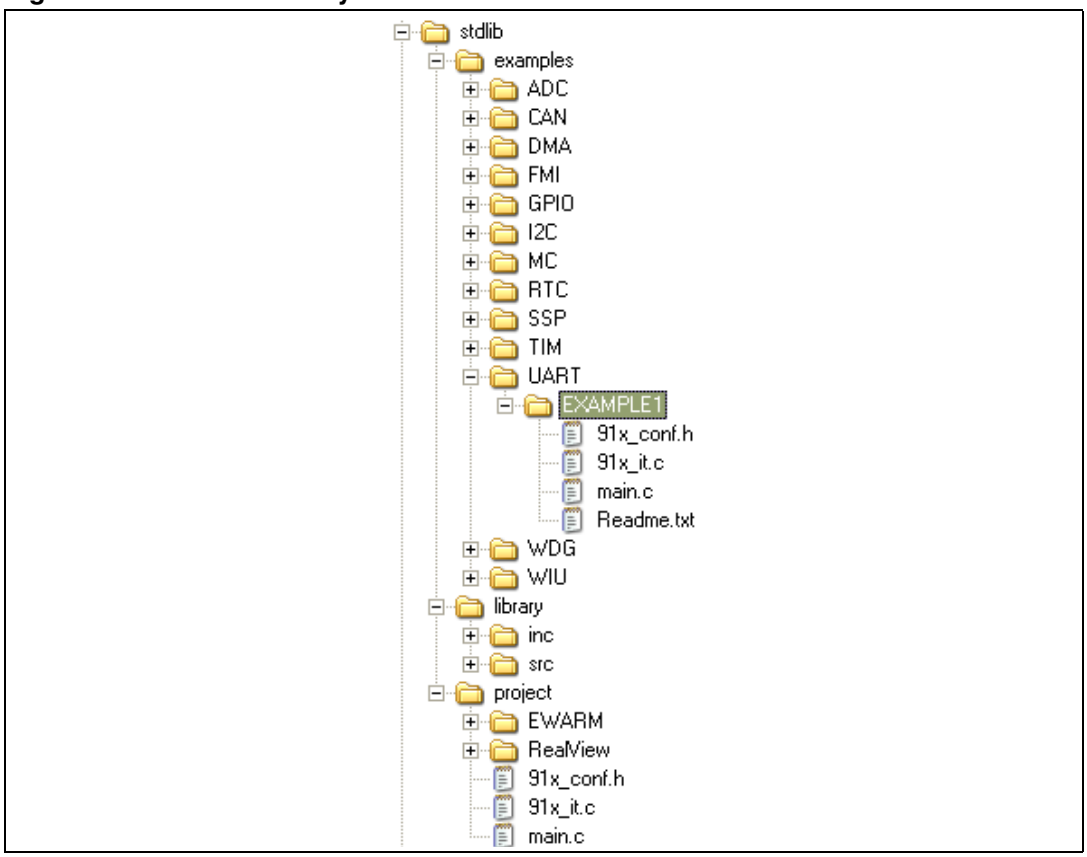
The STR91x software library software package is composed of three different libraries:

1. STR91x StdLib consisting of device drivers for all standard STR91x peripherals
2. STR91x USB library consisting of USB driver and sample application
3. STR91x Ethernet library consisting of Ethernet driver and sample application

This application note deals only with the STR91x StdLib.

To clearly understand the use of CAPS with the STR91x StdLib, it is important to understand the library architecture. The STR91x StdLib contains the subdirectories shown in *Figure 1*.

**Figure 1. StdLib directory structure**



The ***example*** directory has a sub-directory for each peripheral containing the minimum set of files needed to run a typical example on how to use the peripheral. These files are:

- ***Readme.txt***: a brief text file describing the example and how to make it work
- ***91x_conf.h****:* the header file to configure the used peripherals and miscellaneous defines
- ***91x_it.c***: the source file containing the interrupt handlers (the function bodies may be empty if not used)
- ***main.c***: the sample program

All examples are independent of any software toolchain.

*Note: In the notation for file names, "ppp" is used as a placeholder for any peripheral acronym, such as ADC, CAN, DMA, etc.*

The **Library** directory contains all the subdirectories and files that form the core of the library:

- ***inc*** sub-directory contains the software library header files that do not need to be modified by the user:
  - ***91x_type.h***: contains the common data types and enumeration used in all other files
  - ***91x_map.h***: contains the peripheral memory mapping and register data structures
  - ***91x_lib.h***: is the main header file including all other headers
  - ***91x_ppp.h*** (one header file per peripheral): contains the function prototypes, data structures and enumeration

***src*** sub-directory contains the software library source files that do not need to be modified by the user:

- ***91x_ppp.c*** (one source file per peripheral): contains the function bodies of each peripheral.

All library files are coded in Strict ANSI-C and are independent from any software toolchain

The ***project*** directory contains a standard template project program for each supported software toolchain, which compiles all library files and also all the user modifiable files needed to create a new project:

- ***91x_conf.h***: the configuration header file with all peripherals defined by default
- ***91x_it.c***: the source file containing the interrupt handlers (the function bodies are empty in this template)
- ***main.c***: the main program body

For more details, please refer to UM0233 -*Standard Library User Manual*

## 1.2 Using the STR91x StdLib with CAPS

### 1.2.1 Source file definitions

This section introduces the source files that are referred to in the procedure in *Section 1.2.2 on page 6*.

CAPS interfaces with the STR91x StdLib using the 91x_caps.c source file and the header file generated by CAPS containing the entries specified by the user. The 91x_caps.c source file is provided with this application note. The following table describes these files.

| File | Description |
|---|---|
| C header file generated by CAPS | CAPS generates a C header file with all the correct settings to integrate into the STR91x standard library. By default, the filename is that of the CAPS project. For efficiency, this filename name should be changed to 91x_conf.h. |
| 91x_caps.c | The 91x_caps.c is a configuration file. **It should be included in the StdLib\library\src subdirectory to be linked with the application source files**. It describes:<br>1. The clock settings: clock sources and dividers (AHB, APB, EMI, FMI)<br>2. The GPIO matrix configuration<br>3. EMI wait state configuration<br>4. The clocks enabled for the peripherals used in the application |

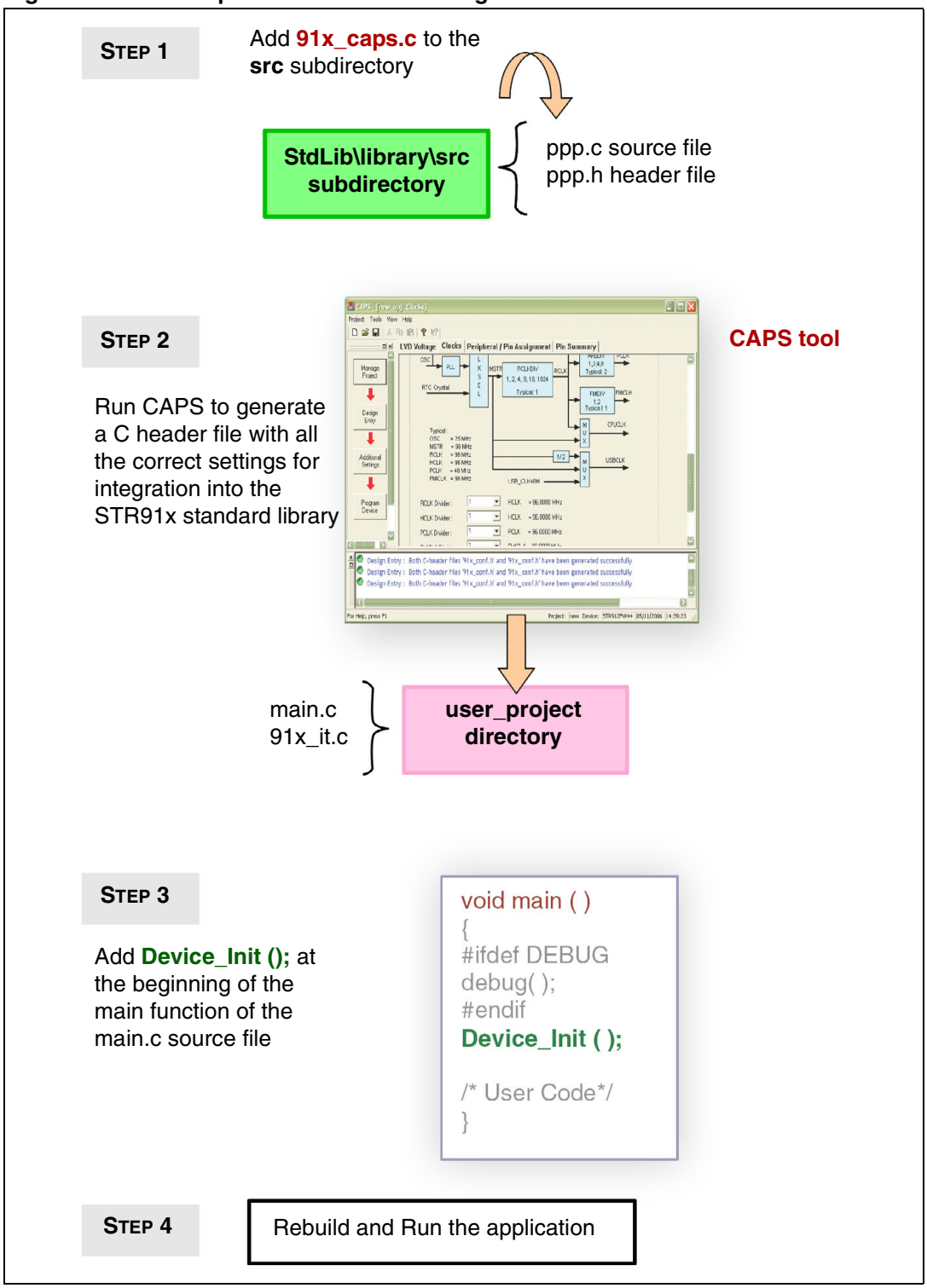
### 1.2.2 General procedure for Interfacing with the STR91x StdLib

1. Add the 91x_caps.c source file to the ***StdLib\library\src*** subdirectory to be linked with the final project, as shown in *Figure 2*, step 1.
2. Run the CAPS utility and choose/change the clock and/or the GPIO settings and assignment. In the **Select folder and file** field choose the ***user_project directory*** and name the C header file to 91x_conf.h, as shown in *Figure 2*, step 2 (For a detailed example, refer to *Section 2*).
3. Add "Device_Init()" function, defined in 91x_caps.c, in the main.c source file at the beginning of the main function, as shown in *Figure 2*, step 3.
4. Include all C source files in the StdLib_project\library\src subdirectory and C header files in the StdLib_project\library\inc subdirectory to your project.

*Note: Including all C source and header files to the user application reduces the risk of error. Nevertheless, for optimization reasons, at least the following source and header files should be compiled:*

* 91x_emi.c & *9 1x_emi.h*
* 91x_fmi.c & *9 1x_fmi.h*
* 91x_gpio.c & *9 1x_gpio.h*
* 91x_dma.c & *9 1x_dma.h*.
* 91x_scu.c & *9 1x_scu.h*.

**Figure 2. General procedure for interfacing with the STR91x StdLib**



# 2 Example

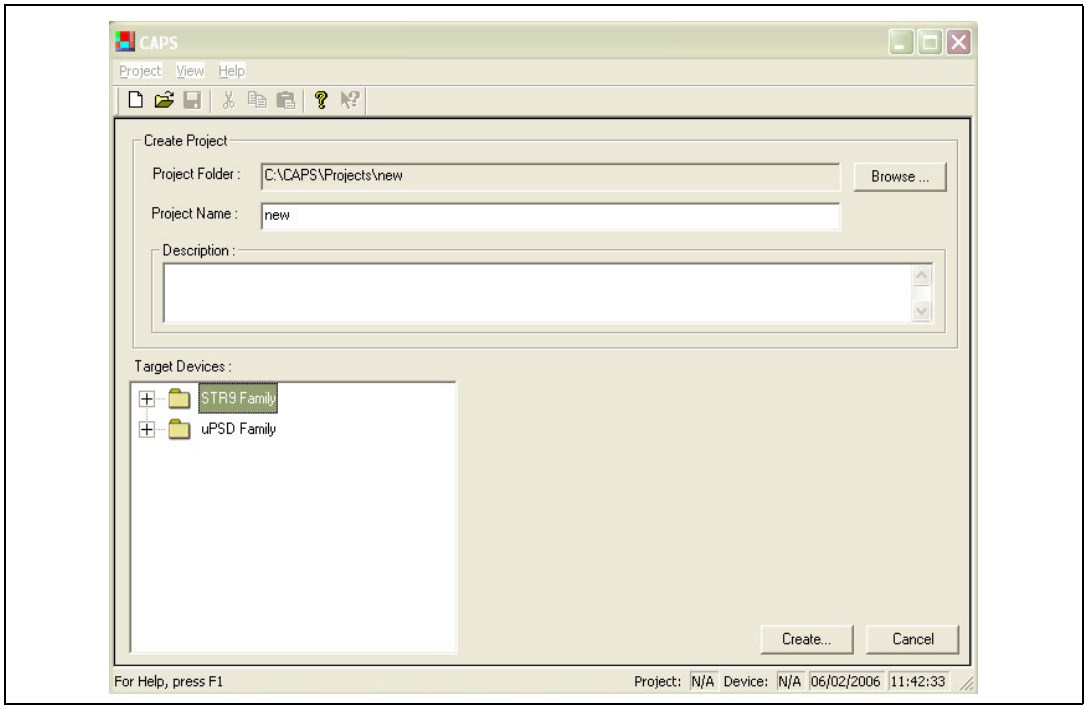
This section provides a step-by-step example of how to use CAPS to select pin functions and clock divisors, and to generate a C header file with all the correct settings to integrate into the STR91x standard software library.

To run CAPS tool:

- Select ***Start>Programs>STMicroelectronics-CAPS>CAPS***.

CAPS starts and the following window appears:

**Figure 3. CAPS project window**

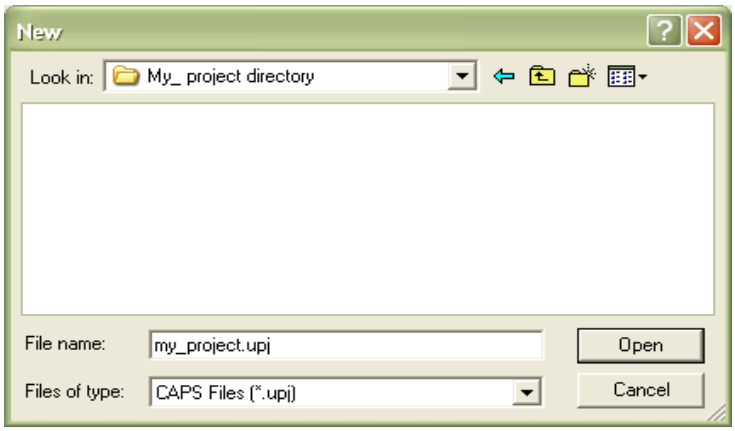



By default the Project folder is ***[CAPS Install directory]\projects\new*** and the defaultproject name is **new**.

To create new project:

- Use the **Browse** button to select a working directory, this opens a dialog box that asks you for the new project file name (see *Figure 4*).
- Enter *project _name* as the project space name in the **File name** field.

**Figure 4. Specifying the project folder and name**

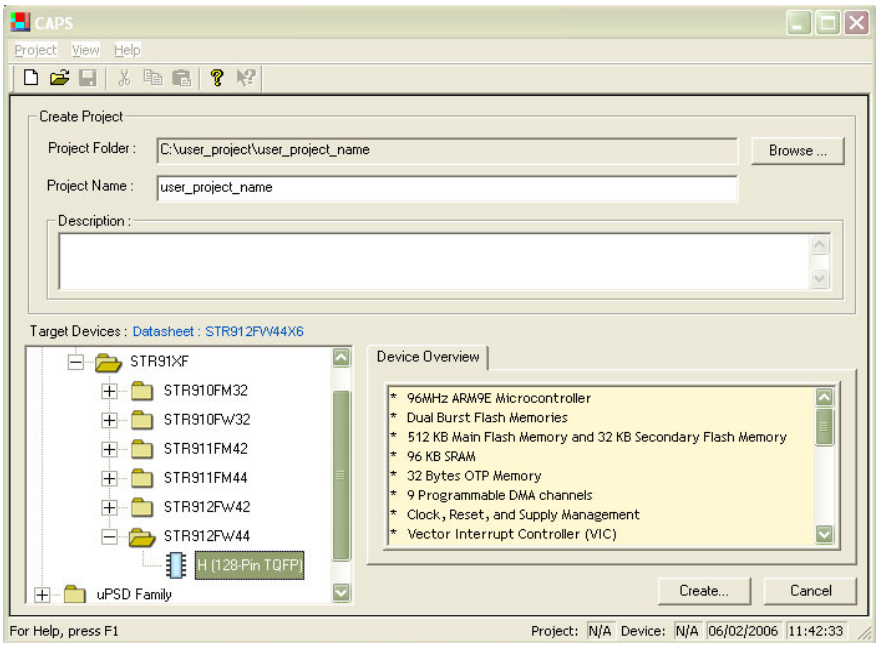



- Click on the **Open** button to validate your entries.

The **Target Device for Target** window allows you to select the microcontroller that you will use in your application. This window shows the CAPS device database for the STR9 and µPSD families.

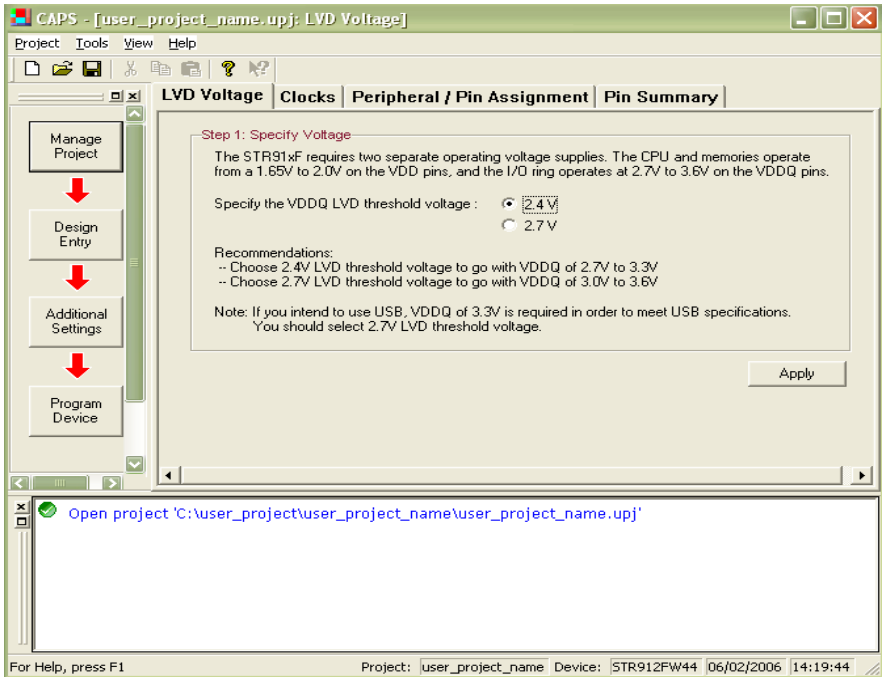
- Select the STR9 MCU you plan to use. In this example, the STR912FW44 microcontroller is used.
- Click to expand the STR912FW44 device and choose the package. The device overview will appear in the right panel (see *Figure 5*).

**Figure 5. Specifying the target device**



- Click on **Create** to confirm your entry. The following dialog appears:

**Figure 6. LVD Voltage window**



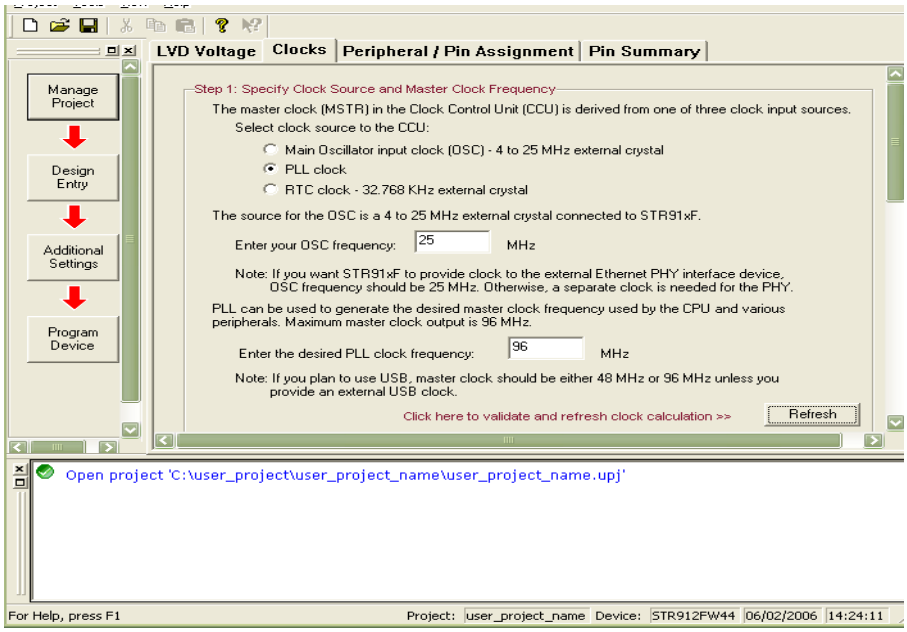

This dialog contains several tabs for specifying the desired configurations. Only the use of the **Clocks tab** and the **Peripheral/Pin Assignment** tab are described in this example.

## 2.1 Clock setting

In the **Clocks** tab, you can specify clock settings in two steps.

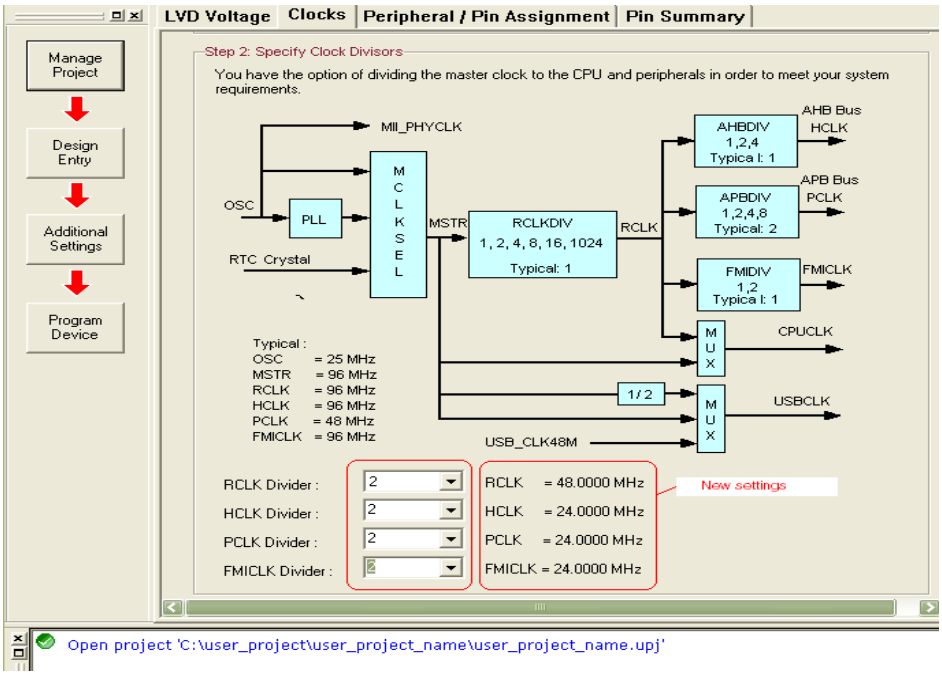
The first step allows the specification of the clock source, the master clock frequency, the OSC frequency and the PLL value with just a few simple and point-and-click operations (see *Figure 7*).

**Figure 7. Clock source Master clock frequency**




The second part shows the clock control diagram and lets users choose the clock divider that is used to obtain the desired frequency from the selection lists.

**Figure 8. Clock divisors**

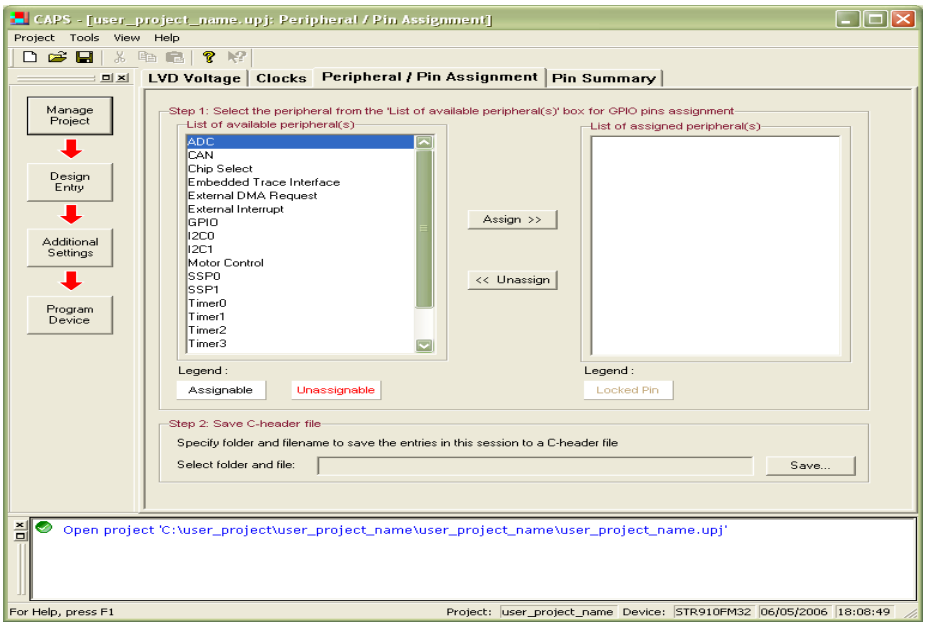


## 2.2 Pin assignments

On the **Peripheral/Pin Assignment** tab, you need to:

1. Specify peripherals fin the **List of available peripheral(s)** pane for GPIO assignment.
2. Specify the folder and file name in order to save the entries to a C header file.

**Figure 9. Peripheral / Pin assignment tab**

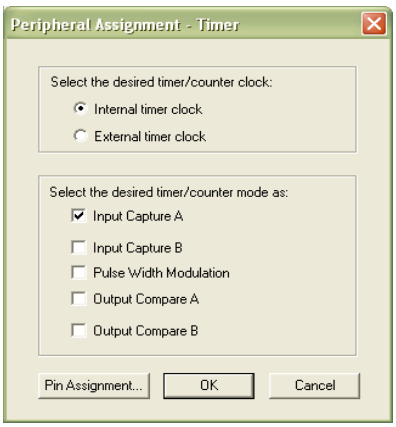



In this example, configuration of the Timer 0 (TIM0) and I2C0 peripherals is described.

### To configure Timer 0 (TIM0)

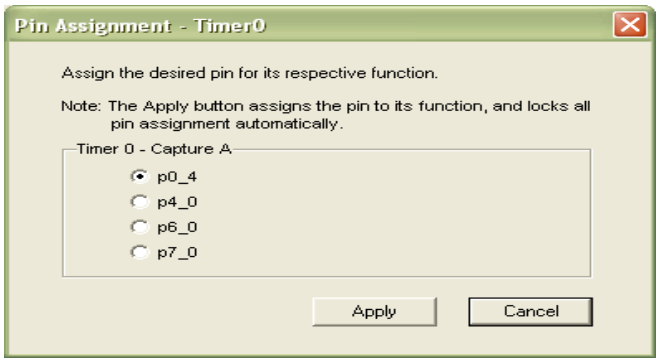
- Select the Timer 0 (TIM0) item from the **List of the available peripheral(s)** pane.
- Click on the **Assign** button. The **Peripheral Assignment-Timer** window appears automatically, allowing you to select the desired timer/counter clock and mode. In this example, the internal timer clock is chosen as timer clock and the input capture A is the selected mode. (see *Figure 10*).

**Figure 10. Peripheral assignment window**



- Click on the **Pin Assignment** button. The **Pin Assignment-Timer0** window appears automatically allowing you to assign the desired pin to its respective function. In this example, the port 0 pin 4 (P0_4) is used (see *Figure 11*).

**Figure 11. Pin Assignment - Timer0 window**



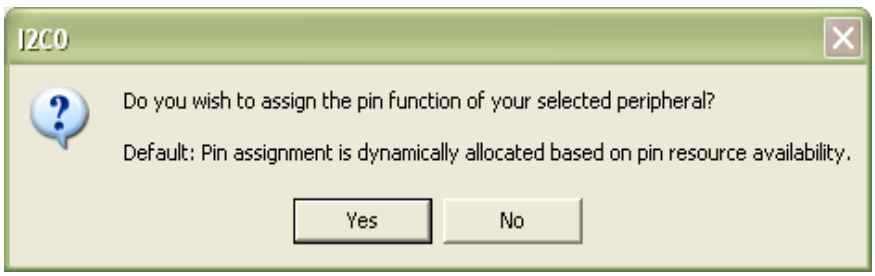

- Click on **Apply** to confirm your entry.

The TIM0 is automatically added to the **List of assigned peripheral(s)** pane.

### To configure I2C0

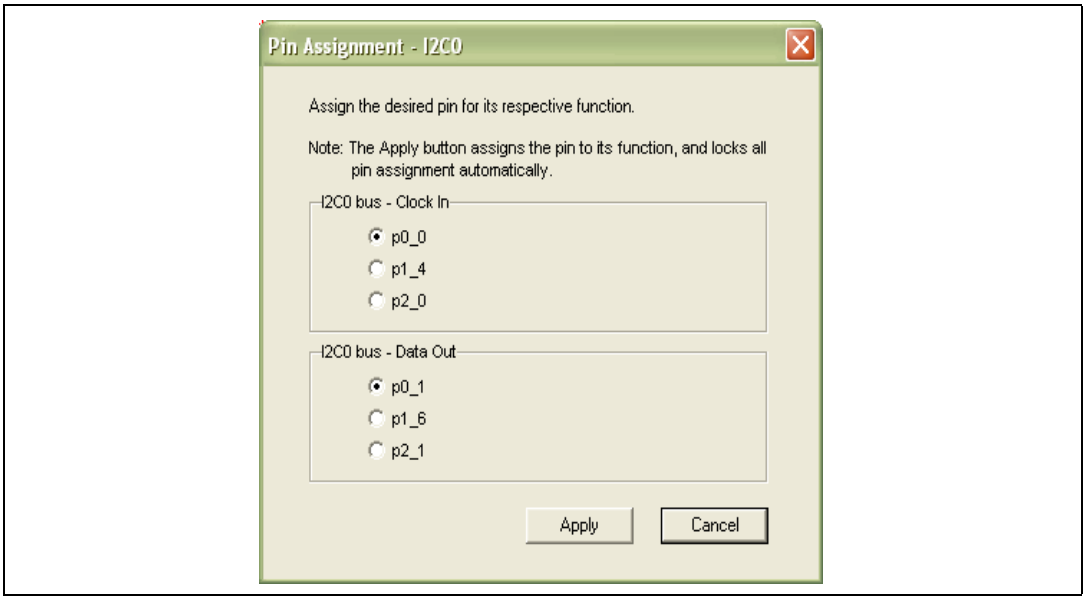
- Select the I2C0 peripheral from the **List of the available peripheral(s)** pane
- Click on the **Assign** button. The following window appears:

**Figure 12. Prompt window**




- Click on **Yes** to confirm. The **Pin Assignment-I2C0** dialog box in *Figure 13* appears:

**Figure 13. Pin Assignment I2C0 window**




The **Pin Assignment-I2C0** window allows you to assign the desired pin to its respective function. In this example, the port 0 pin 0 (p0_0) is used for clock in and port 0-pin1 (p0_1) is used for the data out.

- Click on **Apply** to confirm your entry.

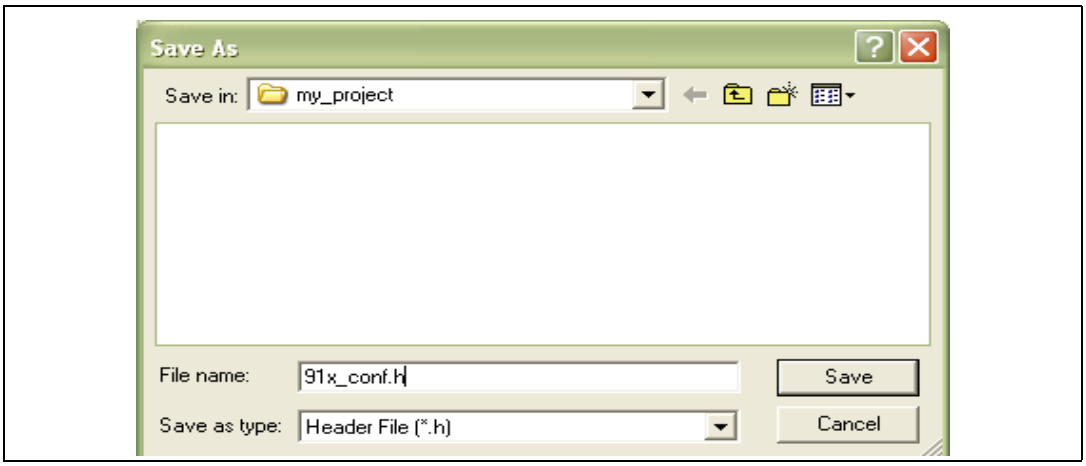
The I2C0 is automatically added to the **List of assigned peripheral(s)** pane.

## Save entries to a C header file

To save the entries in this session to a C header file, you have to specify the folder and file name.

- Click on the **Save** button in the **Peripheral/Pin Assignment** tab. This opens a **Save As** window that allows you to specify the C header file name and directory.
- Choose the ***user_project directory*** and enter **91x_conf.h** in the File name field.

**Figure 14. Save As window**



- Click on **Save** to confirm your entry.

A message indicating that the save was successful appears in the Output window.

Now that you have output the header file from CAPS, follow the remaining two steps described in *Section 1.2.2* and run your application successfully.

# 3 Revision history

| Date | Revision | Changes |
|---|---|---|
| 21-June-2006 | 1 | Initial release. |

**Please Read Carefully:**

Information in this document is provided solely in connection with ST products. STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.

All ST products are sold pursuant to ST’s terms and conditions of sale.

Purchasers are solely responsible for the choice, selection and use of the ST products and services described herein, and ST assumes no liability whatsoever relating to the choice, selection or use of the ST products and services described herein.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by ST for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

**UNLESS OTHERWISE SET FORTH IN ST’S TERMS AND CONDITIONS OF SALE ST DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF ST PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.**

**UNLESS EXPRESSLY APPROVED IN WRITING BY AN AUTHORIZED ST REPRESENTATIVE, ST PRODUCTS ARE NOT RECOMMENDED, AUTHORIZED OR WARRANTED FOR USE IN MILITARY, AIR CRAFT, SPACE, LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS, NOR IN PRODUCTS OR SYSTEMS WHERE FAILURE OR MALFUNCTION MAY RESULT IN PERSONAL INJURY, DEATH, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE. ST PRODUCTS WHICH ARE NOT SPECIFIED AS "AUTOMOTIVE GRADE" MAY ONLY BE USED IN AUTOMOTIVE APPLICATIONS AT USER’S OWN RISK.**

Resale of ST products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by ST for the ST product or service described herein and shall not create or extend in any manner whatsoever, any liability of ST.

ST and the ST logo are trademarks or registered trademarks of ST in various countries.

Information in this document supersedes and replaces all information previously supplied.

The ST logo is a registered trademark of STMicroelectronics. All other names are the property of their respective owners.



STMicroelectronics group of companies

Australia - Belgium - Brazil - Canada - China - Czech Republic - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan - Malaysia - Malta - Morocco - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States of America

**www.st.com**