UM0225
User manual

Configuration and programming software (CAPS) tool for STR9 families

Introduction

Configuration and Programming Software (CAPS) is the configuration software for the STR9 family microcontroller. The CAPS configuration tool allows you to easily configure the STR9 using simple drag-and-drop and point-and-click operations. CAPS also supports In-System-Programming through an external JTAG adapter, allowing fast In-System-Programming of the STR9 in both development and production environments.

This is the CAPS user manual, describing CAPS software functionality. When working with the CAPS tool, you are also encouraged to download the datasheet associated with your particular device; the datasheet may provide the only source of important configuration information needed for your design.

Getting started gives an introduction to CAPS installation procedures and hardware requirements. Although installation may seem trivial, it is highly recommended that you carefully follow the instructions because many problems are often caused by incorrectly installing CAPS.

This is followed by *Introduction to the CAPS user interface*, which describes general CAPS usability and the design process.

The *Designing with CAPS* section gives detailed information about each CAPS feature.

The appendices provide reference material useful for design and analysis.

1 Getting started

Before using CAPS, install the software on your PC. Connect your target device, if you plan to program the device.

This section discusses the following topics needed to begin using the CAPS software.

- CAPS software installation.
- Setting up the target hardware.
- The recommended approach to using this manual.
- Complementary documentation considered to be useful when using the CAPS software.

1.1 Installation

This section describes the requirements and procedures needed to install the CAPS software.

1.1.1 System requirements

The CAPS PC configuration minimally requires:

- PC with an Intel Pentium processor running a 32-bit Microsoft operating system:
  - Microsoft Windows XP
  - Windows 2000
  - Windows 98
  - Windows ME
  - Windows NT with Service Pack 6
- 32 MB RAM
- 25 MB hard disk space available

Note: To use RLINK-ST, a USB port is required with a USB-supporting Windows operating system; e.g., Win98SE, Win2000, Me and XP. (Note that Win95, Win98 First Edition and NT4.0 do NOT support USB).

Caution: FlashLINK Cable: JTD driver (OD) is NOT supported on dual-processor systems or hyper-threading enabled systems. Refer to *Appendix D: FlashLINK Cable – Install fast JTAG driver*, for workaround options for both dual-processor and hyperthreading systems.

1.1.2 Installing CAPS

Follow these procedures and the on-screen instructions to install CAPS.

1. Download the compressed CAPS software from the website.
2. Extract the contents of the *.zip* file into a temporary directory.
3. Double-click the extracted executable, *setup.exe*, to initiate the installation, and follow the on-screen prompts to install CAPS in the development environment. This executable installs all the necessary files and configures the PC environment for running CAPS. You may be prompted to restart your PC before running CAPS for the first time following the installation.

CAPS installation includes a number of utilities. Documentation for the utilities is located in the subdirectory *\Docs* where CAPS is installed.

Table 1. CAPS utility programs

Utility Executable	Description
ObjFileEditor.exe	Programming data file (.OBJ) editor.
uFLink.exe	Standalone JTAG/ISP programming utility.
uMerge.exe	Merge firmware utility.
uObjOsf.exe	Program data file conversion utility. Convert obj-to-osf and osf-to-obj files.

Subdirectory *Projects* is also created, and is the default location for storing your CAPS project files. For example, if CAPS is installed in base directory *C:\CAPS*, the *Projects* directory is located at *C:\CAPS\Projects*.

1.1.3 Uninstalling CAPS

To uninstall CAPS, select **Start | Programs | STMicroelectronics - CAPS | Uninstall CAPS**. This removes all CAPS executable software and desktop references.

Note: Any project files and environment files are preserved so they are available following a CAPS software upgrade. However, it is a safe practice to backup your project files before uninstalling and reinstalling the software.

1.2 Setting up the target hardware

If you are using CAPS features that interact with the target hardware, such as programming the flash, configure and power the target hardware before starting a CAPS session. (Refer to the the User Guide for your particular target hardware, found at http://www.st.com/mcu/).

Note: 1 When using FlashLINK, the 20-pin to 14-pin adapter (provided with the EVAL Kit or a customer board) is required.

2 The device under test must be the same as the target device you select when you created your project.

3 Follow the instructions below referring to the device-specific quick start guide or design guide as needed.

1. Connect either the RLINK-ST USB cable or the FlashLINK parallel cable to your PC, and connect the other end of the JTAG interface to the target board.
2. Configure jumpers according to the quick start or design guide documentation for the target board.
3. Attach the power plug to the power jack of the target board.
4. Switch ON the target board.

1.3 How to use this manual

Use these recommendations as a guide to learning the CAPS software.

- Read *Getting started* to learn what CAPS is and to install the software for the first time.
- Read *Introduction to the CAPS user interface* for a basic understanding of the CAPS user interface. More advanced users may skip this section.
- For a detailed discussion of how to use CAPS features, read *Designing with CAPS*.

1.4 Recommended reading

You are encouraged to download the datasheet associated with a particular device. The datasheet may be the only source of important configuration information needed for your design.

2 Introduction to the CAPS user interface

This section introduces you to the following CAPS topics:

- Project development steps and considerations.
- A reference for the CAPS user interface.
- Beginning steps needed to start a project.

2.1 Project development

The CAPS software guides the designer through the process of configuring a target device for a particular application, using the following project development steps.

1. Create a unique project for each device under test/application combination.
2. Configure voltage, clock source, peripheral and GPIO pins, and hardware and firmware parameters.
3. Merge the design with your Intel hex-format firmware file.
4. Program the resulting programming data file into your device.
5. Save your project to a file for later use.

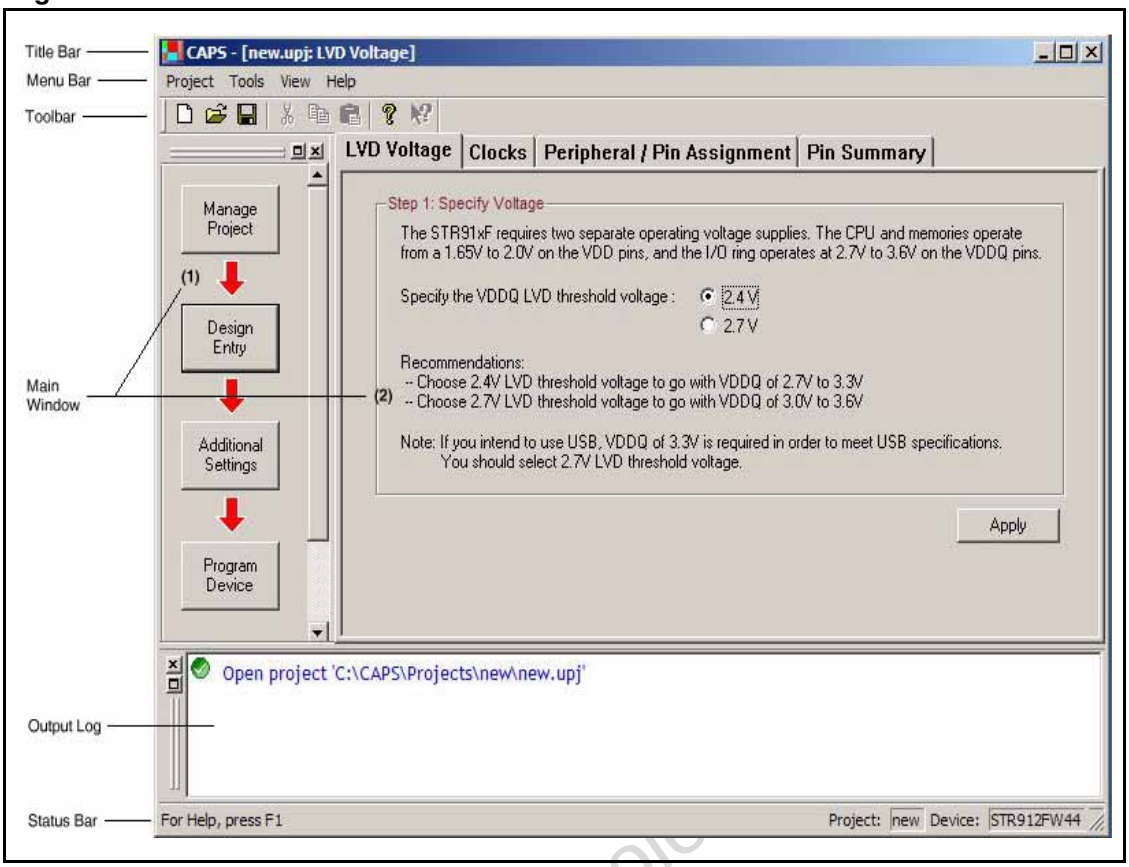
2.2 The user interface

This section is a reference for the CAPS user interface. To learn how to begin using the interface to work with a project, see *Starting a project*.

Figure 1: CAPS user interface shows the basic CAPS user interface components, followed by a brief description of each component. Following sections describes the menu bar in more detail. Specific functionality is described in *Designing with CAPS*.

The CAPS software is a Windows-based program. As such, the user interface implements basic interface conventions commonly found in Windows programs.

Figure 1. CAPS user interface



Menu bar

Use the menu bar to access these CAPS design functions: *The project menu*, *The tools menu*, *The view menu* and *The help menu*. (These functions are described in more detail beginning with *2.2.1: The project menu*.

Toolbar

The toolbar provides quick access to common menu bar functions, including:

- Create a new project.
- Open an existing project.
- Save a project.
- File editing functions: cut, copy and paste.
- Help using CAPS.

Main window

The main window displays the CAPS design entry forms. The window may consist of multiple panes, depending on the current design function.

Figure 1: CAPS user interface shows an example of a typical design window with two panes.

1. A navigation pane appears on the left.
2. A current work pane appears on the right.

Some modes of operation may also have function tabs across the top of the work window, as shown in *Figure 1.*

Output log window

The output log window echoes all commands executed by CAPS along with informational and progress messages.

Note: This window is made visible by checking the Output Log option in the View menu.

Status bar

The status bar displays:

- Current project name.
- Target device.
- Today's date. (MM/DD/YYYY)
- Current time (HH:MM:SS)

The CAPS user interface provides the following additional aids to using the software.

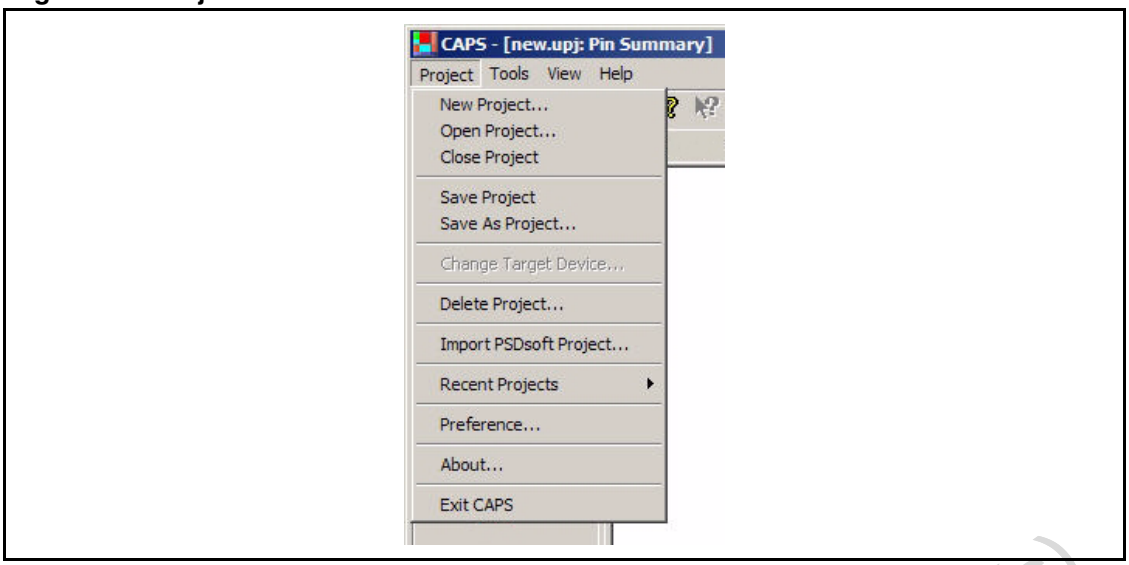
- Descriptive pop-up error messages.
- Explicit design flow sequencing that models the actual design process.
- Detailed directions on most forms that describe input field formats and how to use the form.

2.2.1 The project menu

The project menu allows users to manage the project life cycle, set project preferences and exit the CAPS program.

This section describes the operations available in the project menu.

Figure 2. Project menu



New Project ... Creates a new project.

Open Project ... Opens an existing project.

Close Project Closes the currently active project. Other projects previously opened within the same CAPS session remain open.

Save Project Saves the currently active project.

Save as Project ... Saves the currently active project to a different file name. Follow these steps in the displayed dialog box:

1. Enter the new project name (see *2.3.1: Creating a new project* for project name constraints).
2. Optionally, enter or modify the project description
3. Click the **Save** button.

Change Target Device ... Selects another target device ONLY within the same product family, from the expandable device tree list.

Changing the package type may invalidate the pin function re-assignment done in the **Peripheral/Pin Assignment** window.

Delete Project ... Deletes a project and all files associated with the project.

Import PSDsoft Project ... Imports an existing project created using the PSDsoft tool.

Note: This option is only applicable to µPSD projects.

Recent Projects Lists the most recently opened projects. A project may be opened by double-clicking on a list entry.

Preference ... Sets the CAPS design environment. Select either “Single device view” or “Multiple Device view” for the JTAG/ISP property. Single and multiple device operations are discussed in *3.7: Validating and programming the target device*.

About ... Displays details about a project:

- Project name.
- Project folder.
- Device family.
- Part number.
- Voltage.
- Project description.

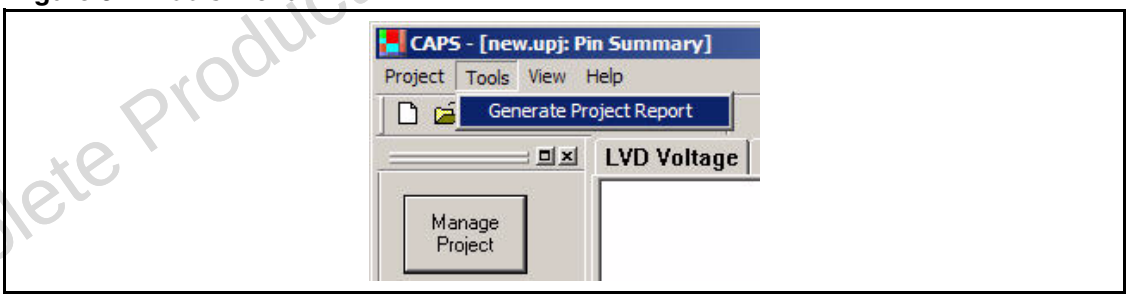
Note: The project must already be open.

Exit CAPS Exit the CAPS program.

2.2.2 The tools menu

The tools menu provides the ability to produce a project report.

Figure 3. Tools menu



Generate Project Report Produces a text file report of your design.

The result shows detailed peripheral and pin function assignments.

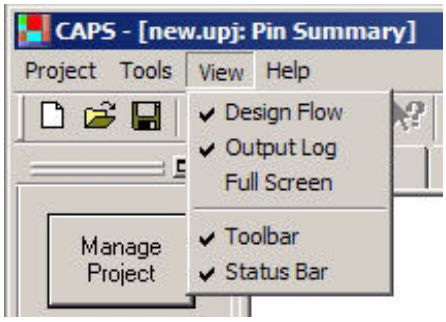
See *Project Report* for an example.

2.2.3 The view menu

The view menu selection allows the user to toggle the user interface panes to be displayed while working on a project. Check the menu item to enable the display.

This section describes the operations available in the view menu.

Figure 4. View menu



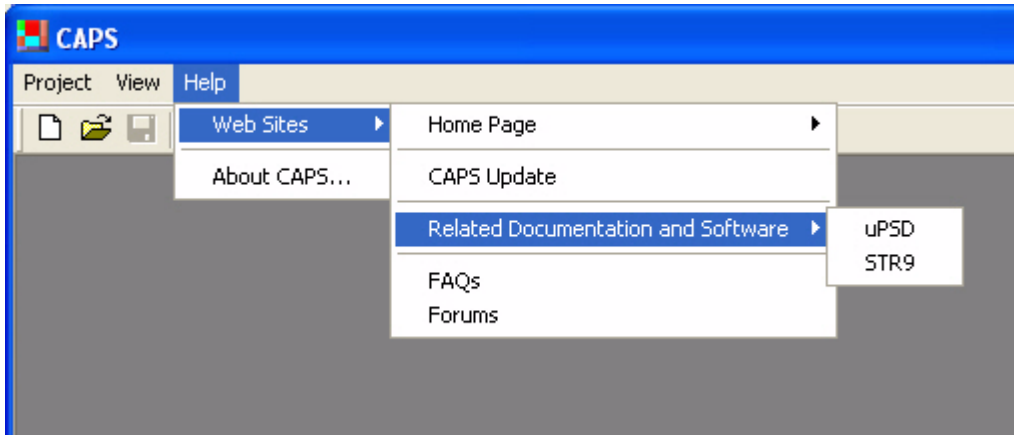
Design Flow	Display the CAPS design flow pane.
Output Log	Display the output log pane that shows commands executed by CAPS, and informational and status messages (See *Figure 1: CAPS user interface*).
Full Screen	Maximize the CAPS display interface on the screen.
Toolbar	Display the toolbar (See *Figure 1: CAPS user interface*).
Status Bar	Display the status bar (See *Figure 1: CAPS user interface*).

2.2.4 The help menu

The help menu provides access to various links to obtain technical information about ST Microelectronics microcontroller products and to report questions or issues related to CAPS.

This section describes the operations available in the help menu.

Figure 5. Help menu



Web Sites Displays links to obtain technical information about ST Microelectronics microcontroller products and to report questions or issues regarding CAPS. A link to Frequently Asked Questions (FAQs) is also provided. (See *Figure 5.: Help menu* for a list of the links provided).

About CAPS ... Displays CAPS software version, copyright, contact and licensing information.

2.3 Starting a project

Every CAPS session begins by opening a project. You may either create a new project or open an existing project.

This section describes the basic steps common to all projects.

2.3.1 Creating a new project

Follow these steps to create a new project.

1. Use either the **New** icon in the toolbar or the **Project** | **New Project ...** menu item (*Figure 2.: Project menu*) to create a new project.
2. Enter an optional project description in the *Description* text box.
3. In the *Create Project* dialog, use the **Browse** button to specify the name of your project and directory where the project files are to be created. (The default project name is *new.upj* and the default directory is *c:\CAPS\Projects\new*). The project name is the same as the directory name).
4. To accept the default project name and directory, click **Open** in the file browser dialog. To choose another directory, use the file browser to navigate to the desired directory. Then,

either use the default *Project Name* or enter a new name for your project. After a new directory and project name are entered, click **Open**.

*Note: A project is saved to disk only after choosing one of the **Save** project menu options.*

A valid project name,

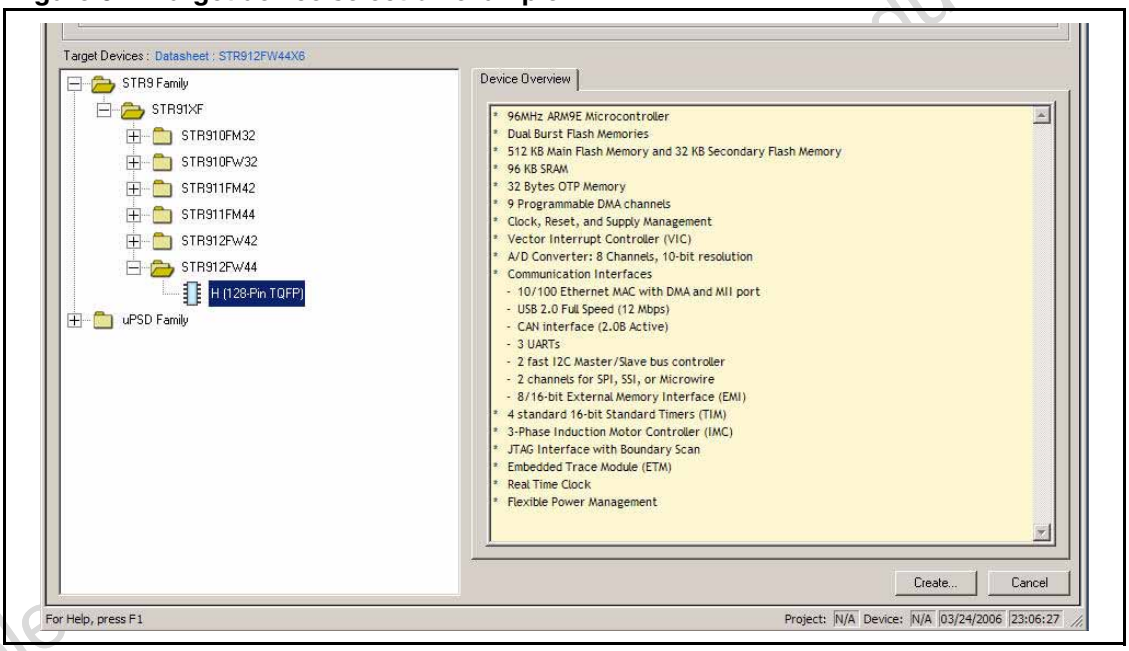
– Can have a maximum number of 45 characters.
– Must begin with an alphanumeric character, an underscore (_) or a tilde (~). Names are not case-sensitive.
– Cannot include symbols or punctuation marks.

5. For new projects, a target device must also be specified. Using the device list tree in the *Target Devices* dialog, expand the tree until your target device appears, then click on the device icon. A description of the device appears in the *Device Overview* window to the right, and the name of the datasheet associated with the device appears in the *Target Devices* line, above the device selection tree.

   Available devices include all devices currently supported by CAPS software.

   Figure 6.: Target device selection example shows an example in which the 128-pin STR912FW44 device is selected.

Figure 6. Target device selection example

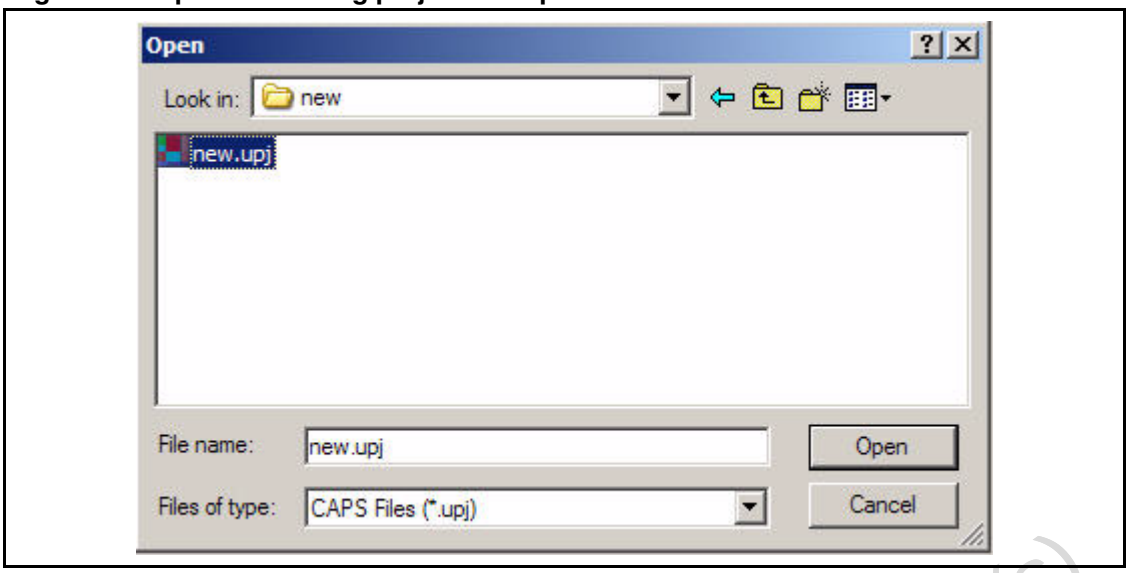


2.3.2 Opening an existing project

Follow these steps to open an existing project.

1. Use either the **Open** icon in the toolbar or the **Project** | **Open Project ...** menu item (*Figure 2.: Project menu*) to open an existing project.
2. In the Open Project dialog, use the **Browse** button to use the file browser to locate the directory where your existing project file resides. The dialog displays the project file names with extension *.upj*, as shown in *Figure 7.: Open an existing project example*. Click on the desired file name, then click **Open** to open the file to display your previously saved project.

Figure 7. Open an existing project example



Note: You may edit the project description, however, notice that the target device may not be changed.

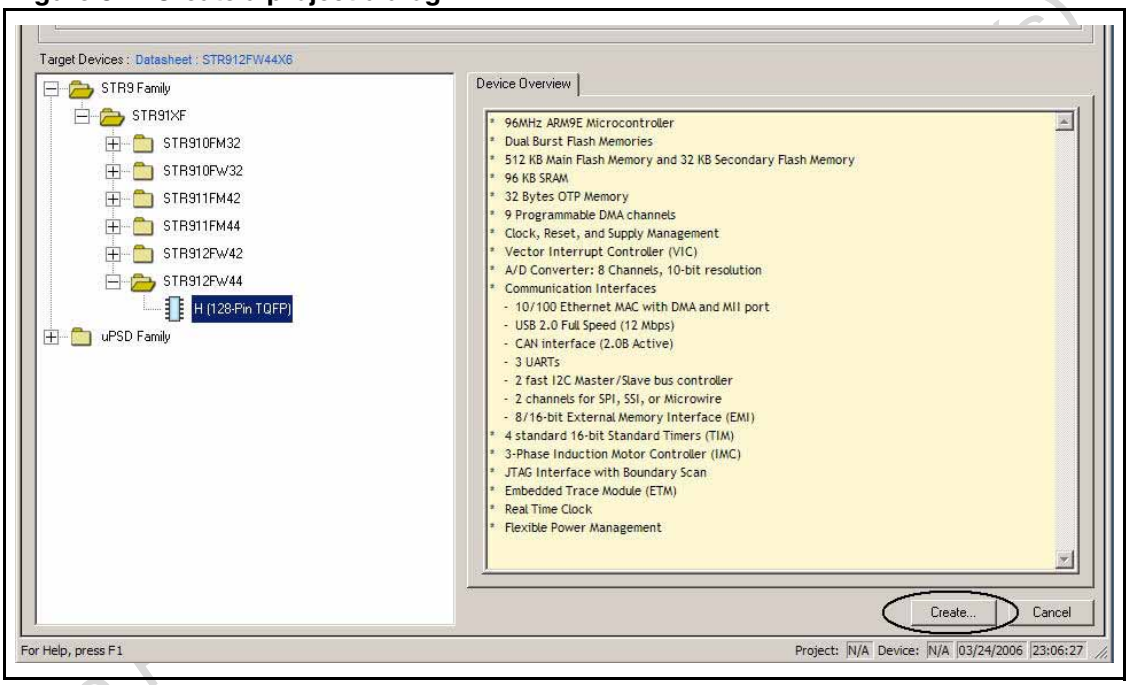
3 Designing with CAPS

This section provides a detailed description of how to use CAPS features, including:

- The design flow model that guides you through the design process.
- Firmware placement.
- Specifying peripheral and GPIO pin function assignment.
- Setting security.
- Setting JTAG/ISP parameters.
- Setting sector protection.
- Validating and programming the target device.

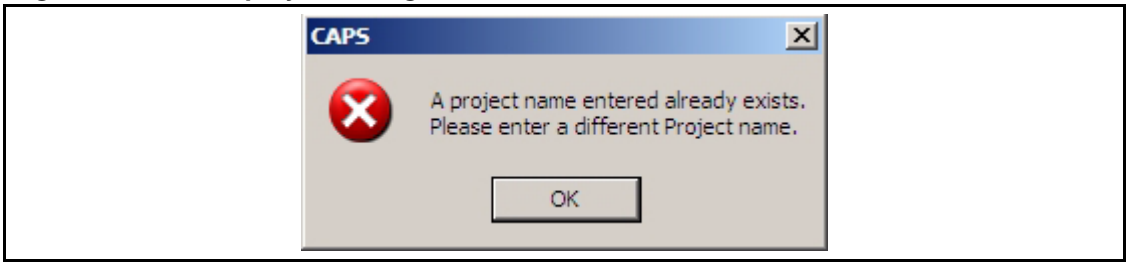
From the window used to create a new project, click the **Create** button to begin designing a project, as shown in *Figure 8.: Create a project dialog*.

Figure 8. Create a project dialog



Project names must be unique. Otherwise, CAPS displays the error message shown in *Figure 9.: Create project dialog error*.

Figure 9. Create project dialog error

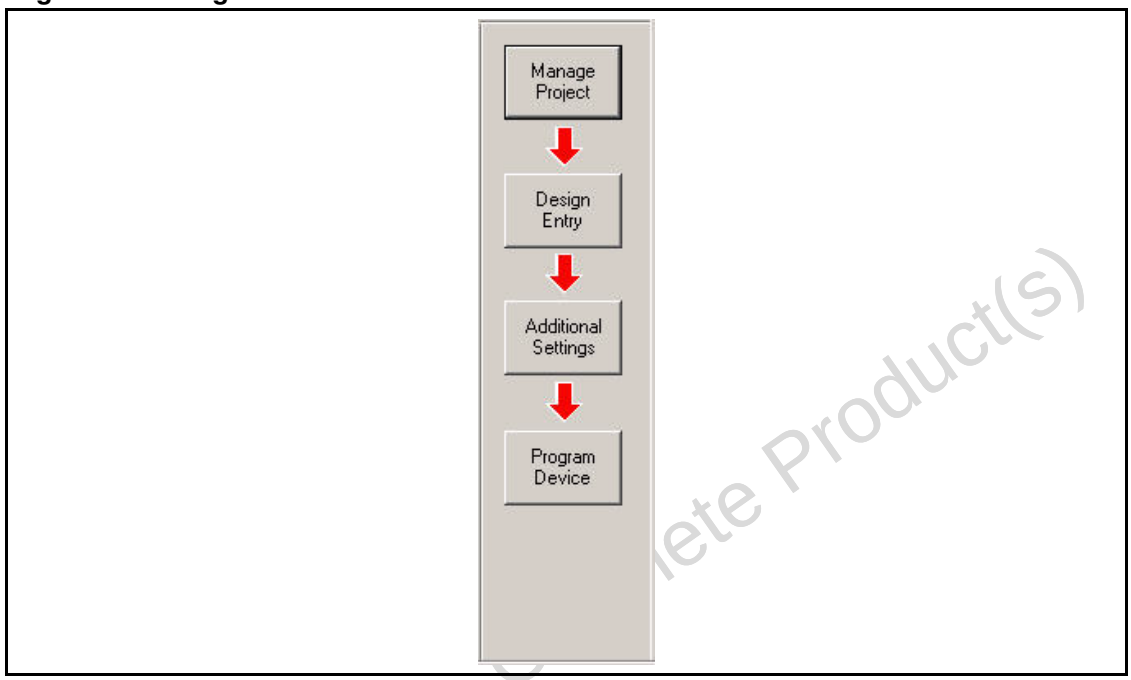


Click **OK** and enter a different project name to continue.

3.1 Design flow

The CAPS design flow interface models the typical steps used in the design process. These steps have corresponding navigation buttons, shown in the left frame of the main window, as shown in *Figure 10: Design flow interface buttons*.

Figure 10. Design flow interface buttons



The red arrow indicates the next action to be performed in the design process.

Click the button corresponding to the desired design and programming operation.

- Manage Project
- Design Entry
- Additional Settings
- Program Device

3.1.1 Manage Project dialog

Click the **Manage Project** button to access the project management functions listed below. Invoke the function by clicking the radio button associated with the function, then click **OK**.

Create a new project	Select this option to create a new project for your design.
Open an existing project	Select this option to open a project that you have previously created.

Save current opened project to a different name	Select this option to rename the current project. The project with the new name becomes the currently active project.
Delete an existing project	Select this option to delete a project and all of its associated files.
Close current opened project	Select this option to close the currently active project.

3.1.2 Design entry forms

Use one or more of the design entry forms for CAPS design. These forms are accessed by clicking the **Design Entry** button, then selecting the appropriate tab:

LVD Voltage	Use this form to choose the desired voltage specification option. See *Specifying the LVD voltage* for a detailed discussion on using this feature.
Clocks	Use this form to choose the desired clock source option. See *Specifying the clock source and frequency* for a detailed discussion on using this feature.
Peripheral/Pin Assignment	Use this form to select the desired peripheral for your design and its pin function assignment. See *Specifying peripheral and GPIO pin assignment* for a detailed discussion on using this feature.
Pin Summary	Use this form to graphically display the results of your **Peripheral/Pin Assignment** operations.

3.1.3 Additional settings form

Optional configuration choices are available by clicking on the design flow **Additional Settings** button, then clicking the **Configuration** tab.

Configuration	Use this form for additional design settings, including setting security, selecting the power-up boot flash, specifying a user code, setting the internal memory sector protection, specifying OTP memory programming, and your firmware placement information.

3.1.4 Program device form

Click the **Program Device** design flow button to program your design into the target device. Options are also available to calculate the file checksum, generate an SVF or JAM file, and validate the target device. See *Validating and programming the target device* for a detailed description on using this feature.

3.2 Specifying the LVD voltage

Click the **LVD Voltage** tab to select the LVD threshold voltage.

The STR91x requires separate operating voltage supplies. The CPU and memory operate from at 1.65 to 2.0 volts on the VDD pins, and the I/O ring operates at 2.7 to 3.6 volts on the VDDQ pins

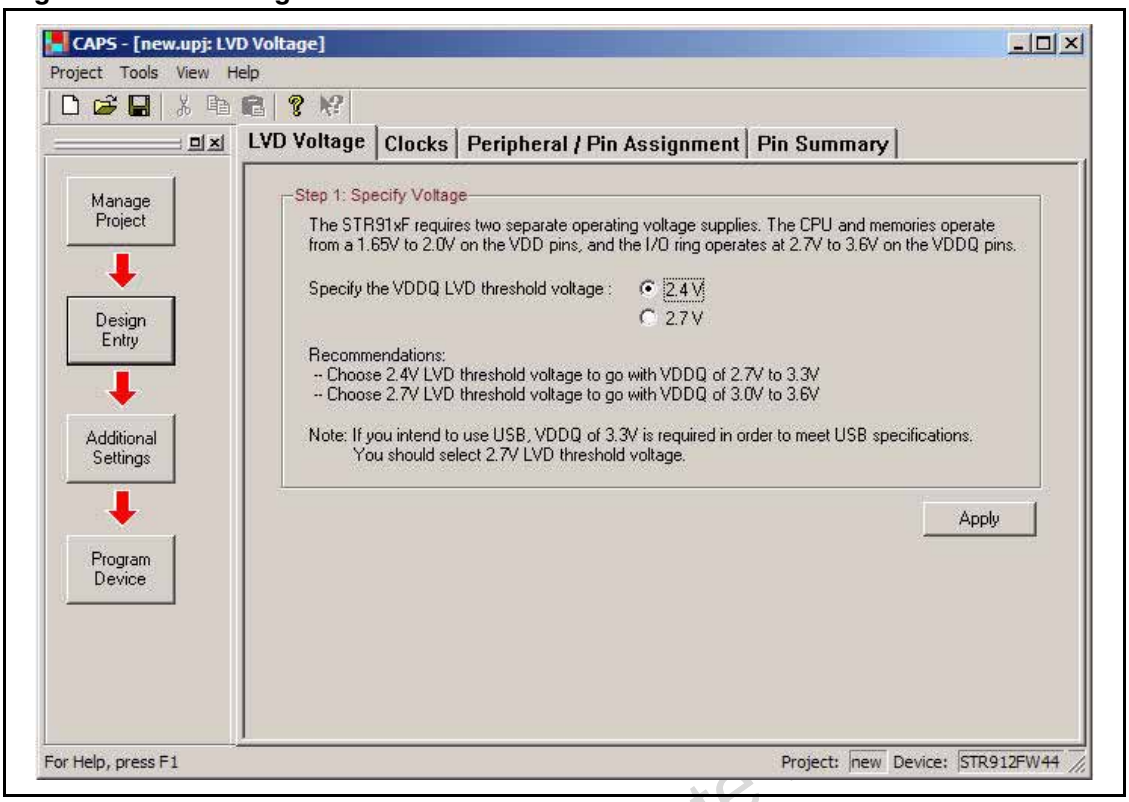
Note: 1 Choose 2.4 volts LVD threshold voltage to go with a VDDQ of 2.7 to 3.3 volts.

2 Choose 2.7 volts LVD threshold voltage to go with a VDDQ of 3.0 to 3.6 volts.

3 If you to use USB, VDDQ of 3.3 volts is required to meet USB specifications. Select the 2.7 volt LVD threshold voltage.

The dialog shown in *Figure 11.: LVD Voltage tab*, is used to select the desired LVD threshold voltage.

Figure 11. LVD Voltage tab



Click either the *2.4V* or *2.7V* LVD threshold radio button, as applicable.

3.3 Specifying the clock source and frequency

Click the **Clocks** tab for the following clocking options.

- Specify the clock source and master clock frequency.
- Specify the clock divisors.
- Save clock settings in a C-header file.

Note: The clock settings affect the Hardware Abstraction Layer (HAL) library so the capability is provided to produce a C-header file to be included in your source code.

The following subsections discuss the clock configuration options in more detail.

3.3.1 Selecting the clock source and frequency

There are three clock source options available.

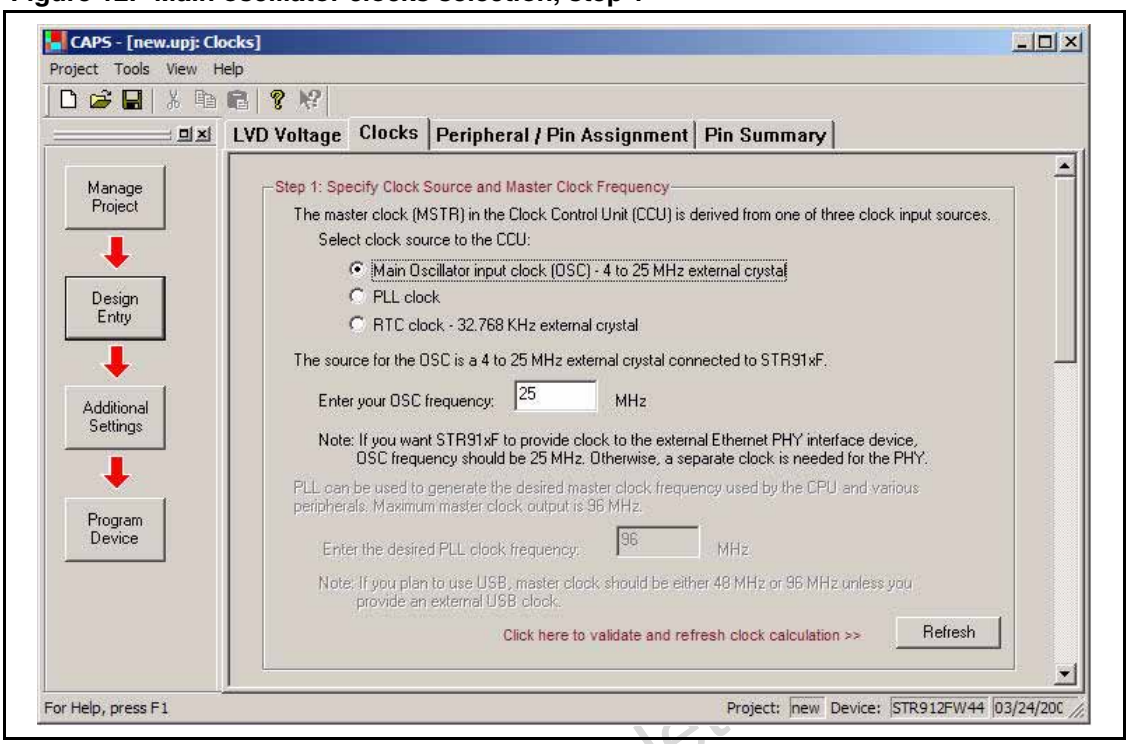
- Main oscillator input clock (OSC)
- PLL clock
- RTC clock

Which clock source is selected affects which other clock configuration options are available. Clicking the radio button associated with the desired clock source automatically displays the configurable parameters, as describe below.

Note: Typical system values are displayed as default values in the following dialogs.

Selecting the *Main Oscillator input clock - 4 to 25 MHz external crystal*, requires you to enter the external crystal frequency, as shown in *Figure 12.: Main oscillator clocks selection, step 1*.

Figure 12. Main oscillator clocks selection, step 1



The frequency must be in the range from 4 to 25 MHz.

Note: If the STR91x provides clock to the external Ethernet PHY interface device, the frequency must be 25 MHz. Otherwise, a separate clock is needed for the PHY.

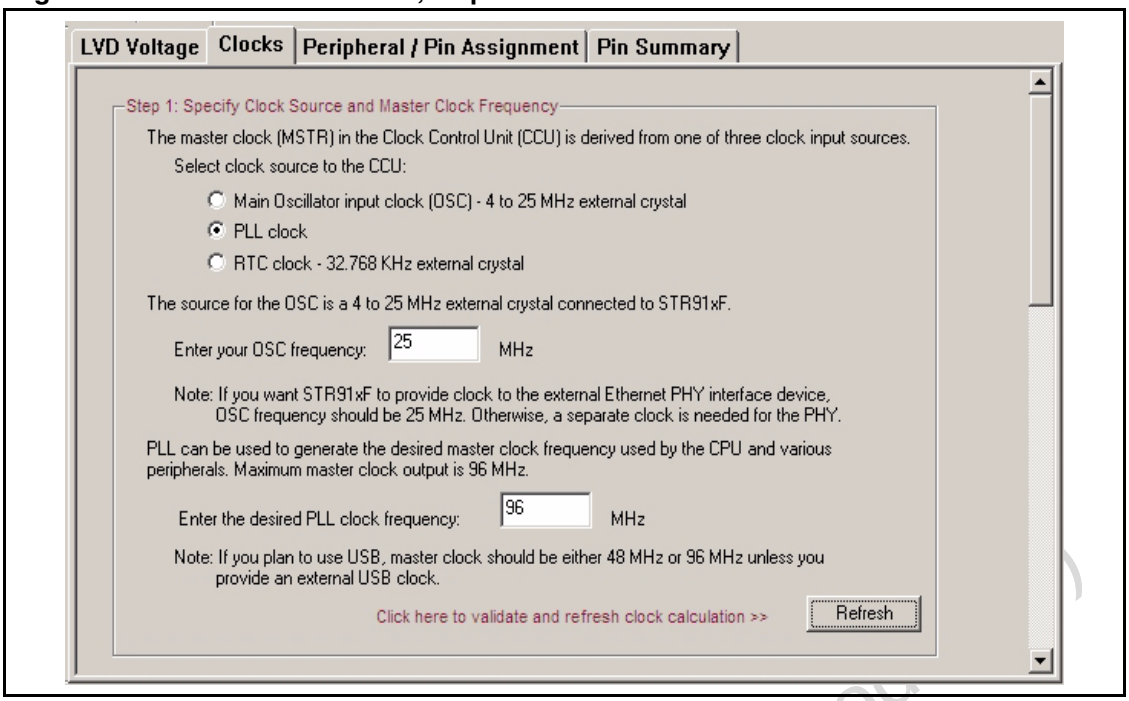
The PLL can be used to generate the desired master clock frequency used by the CPU and various peripherals.

Note: 1 The maximum master clock output is 96 MHz.

2 When using USB, the master clock frequency should be either 48 MHz or 96 MHz, unless an external USB clock is provided.

Selecting the *PLL clock*, requires you to enter the external crystal frequency and the PLL frequency, as shown in *Figure 13.: PLL clocks selection, step 1*.

Figure 13. PLL clocks selection, step 1

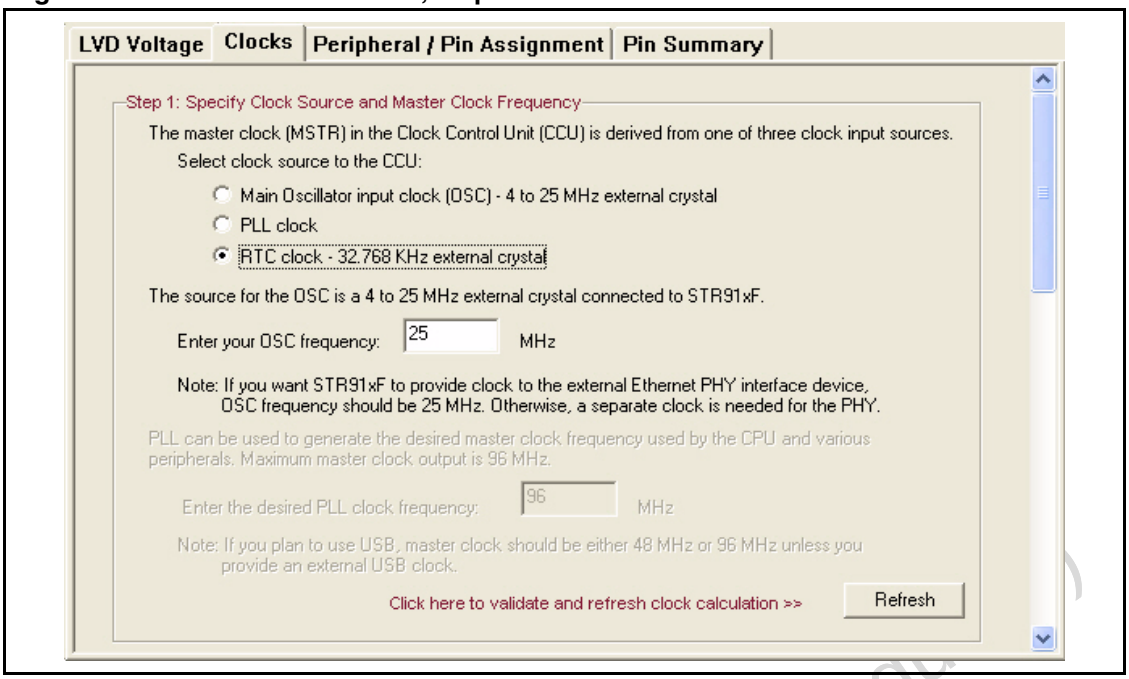


Enter the external crystal frequency, between 4 and 25 MHz, and with the same constraints as described, above.

Enter the PLL frequency.

Selecting the *RTC clock - 32.768 KHz external crystal* , requires you to enter the external crystal frequency, as shown in *Figure 14.: RTC clocks selection, step 1*.

Figure 14. RTC clocks selection, step 1



Enter the external crystal frequency, between 4 and 25 MHz, and with the same constraints as described, above.

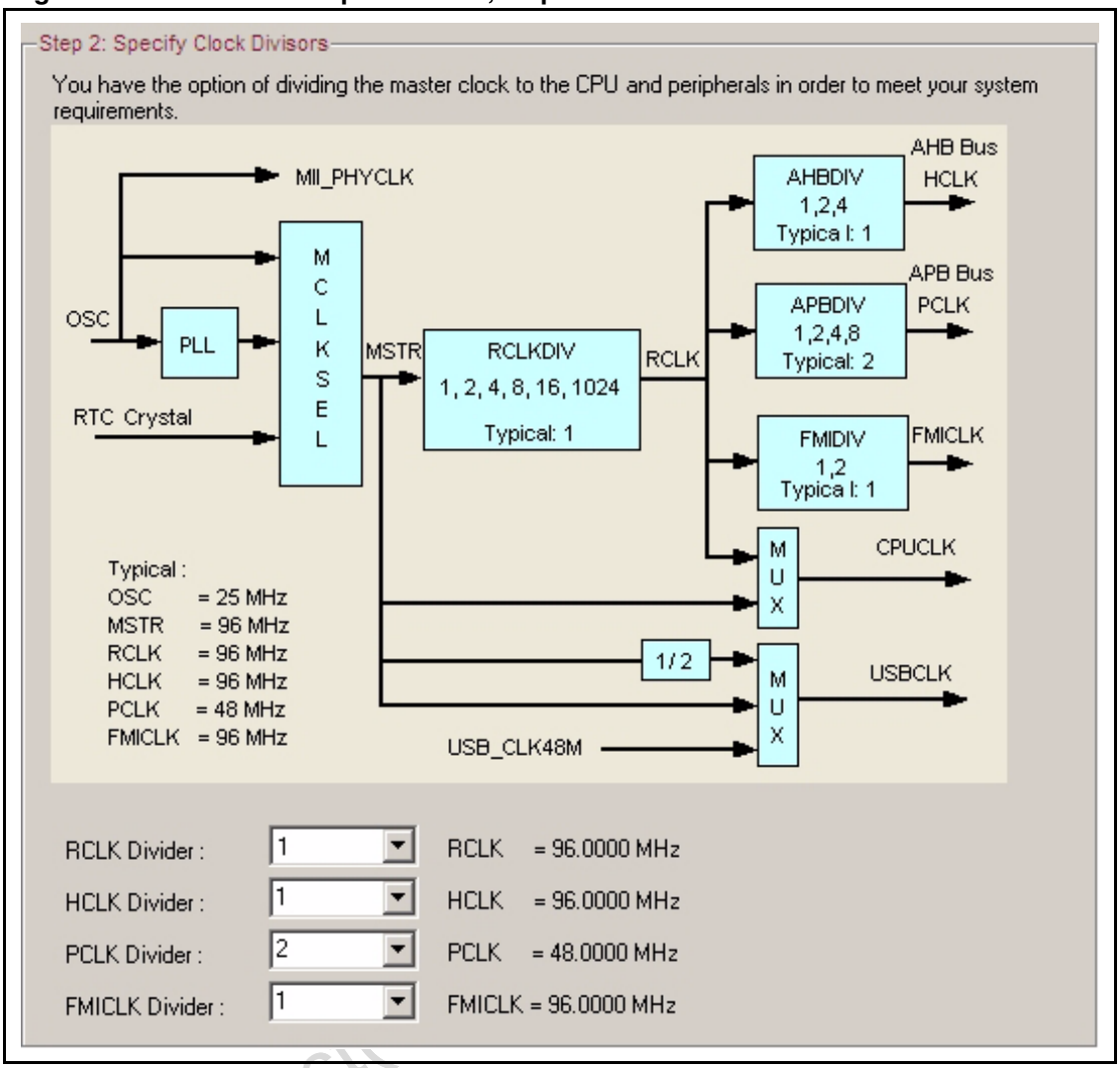
After selecting the clock source and specifying the frequencies, click **Refresh** to validate he input and refresh the clock calculation.

3.3.2 Specifying the clock divisor

CAPS provides the facility for dividing the master clock to the CPU and peripherals to meet particular system requirements; *Step 2: Specify Clock Divisors* provides the following divider options.

- RCLK Divider
- HCLK Divider
- PCLK Divider
- FMICLK Divider

Figure 15. Clock divisor specification, step 2

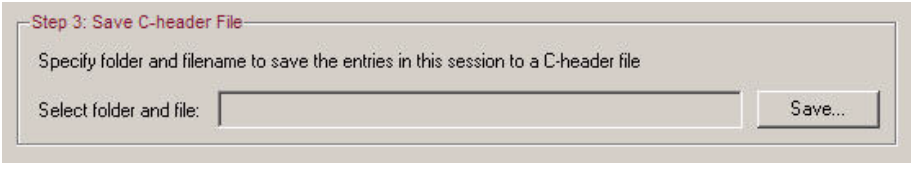


From the drop-down menu for each clock signal, select the desired divider value.

3.3.3 Saving the clock settings

It is recommended that you continue to step 3 of this dialog to create a C-header file. The header file *91x_conf.H* is automatically created for your project along with *Project.H* file. The header file *91x_conf.H* is used by the STR9 HAL library to initialize clock source and peripheral usage. (See *Figure 16.: Clock C-header create, step 3*).

Figure 16. Clock C-header create, step 3



Click the **Save...** button to save the C-header file.

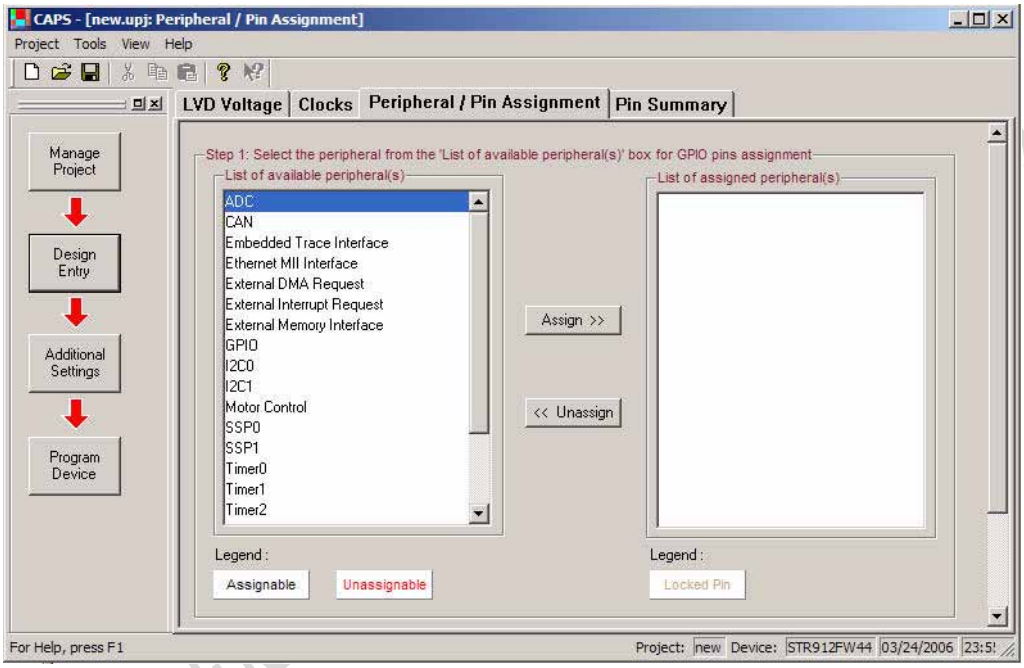
3.4 Specifying peripheral and GPIO pin assignment

Click the **Peripheral/Pin Assignment** tab to select the peripherals for your design. Pin resources are dynamically allocated and de-allocated based on peripheral selection. Therefore, the peripheral selection sequence directly affects pin resource assignment.

Note: The selected peripherals affect the initialization module in the HAL library. A default C-header file, 91x_conf.H, is automatically generated to be included with the HAL library source code. It is recommended to generate a C-header file to be included in your source code, as described in Section 3.3.3, to facilitate integration of the CAPS design and initialization.

Figure 17.: Peripheral/Pin Assignment tab shows a list of peripherals available for the target device.

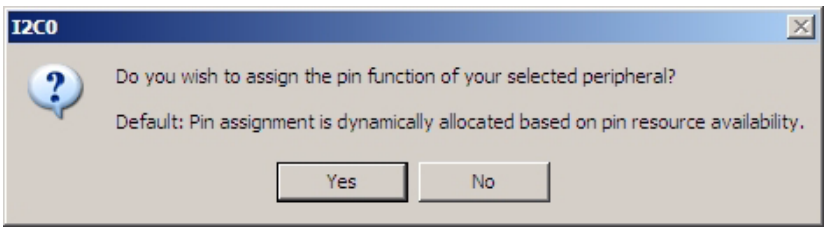
Figure 17. Peripheral/Pin Assignment tab



Select a peripheral from the list and click the *Assign>>* button to assign the pin, moving it to the list in the right window.

As peripherals are assigned, a dialog window displays configurable peripheral parameters. Enter the desired information and values to configure the peripheral. For peripherals that have predefined pin assignments, the following prompt is displayed to either accept the default assignment or enter a different pin assignment.

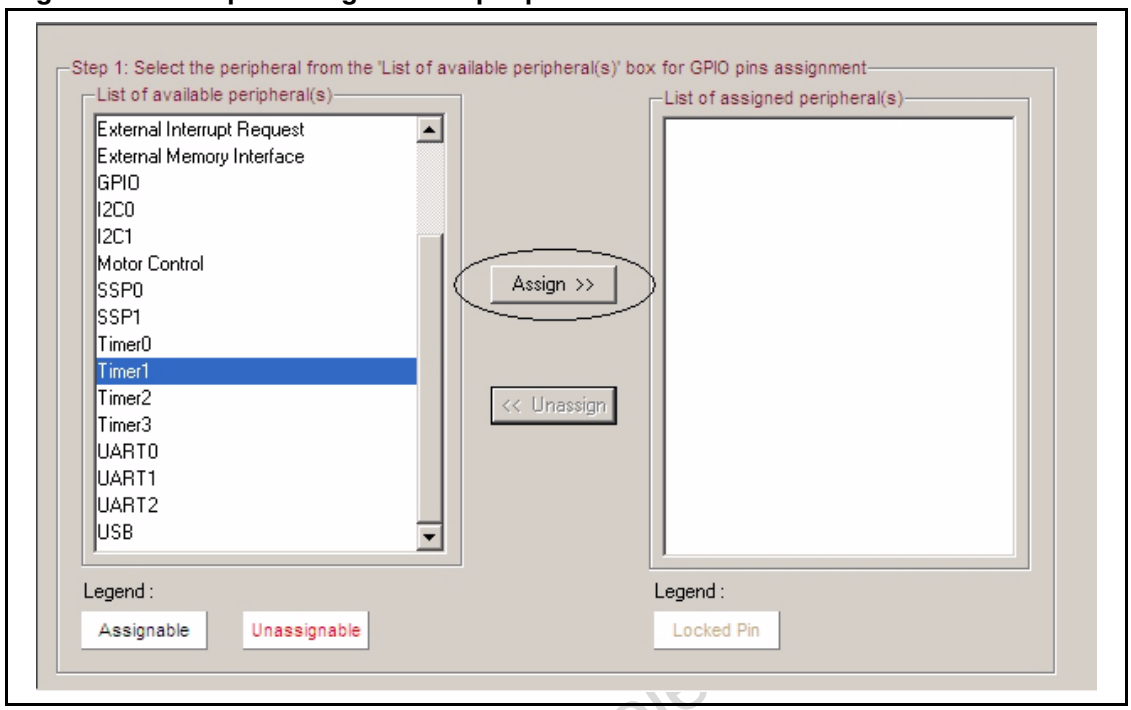
Figure 18. Accept default pin assignment prompt



Click **Yes** to define new pin assignments. Click **No** to accept the default pin assignments for the peripheral.

The following is a peripheral configuration example, using the Timer1 peripheral.

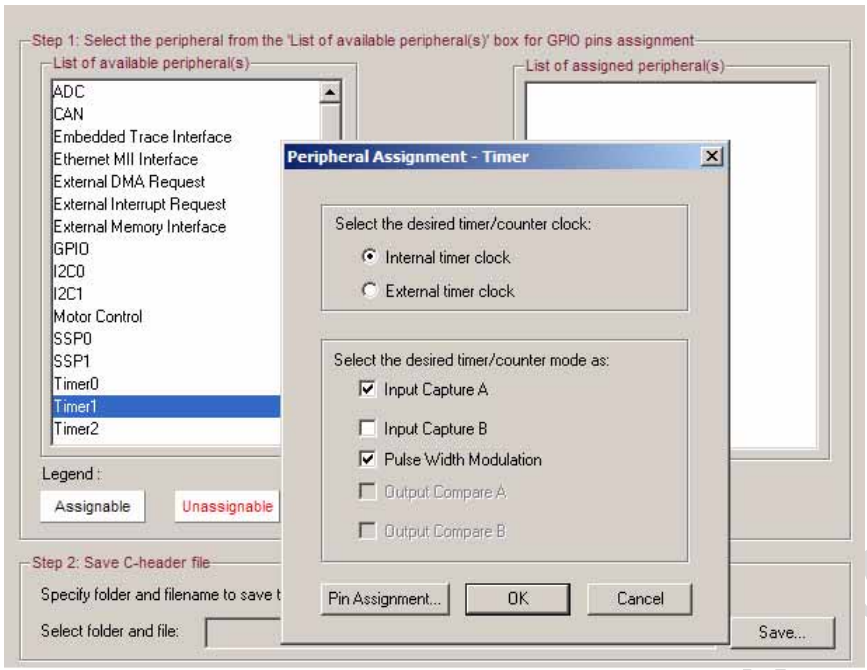
Figure 19. Example - assign Timer1 peripheral



To select the *Timer1* peripheral, left-click Timer1 in the list of available peripherals, then click the *Assign >>* button.

For configurable peripherals, such as Timer1, a dialog window appears with the configuration options, as shown in *Figure 20.*

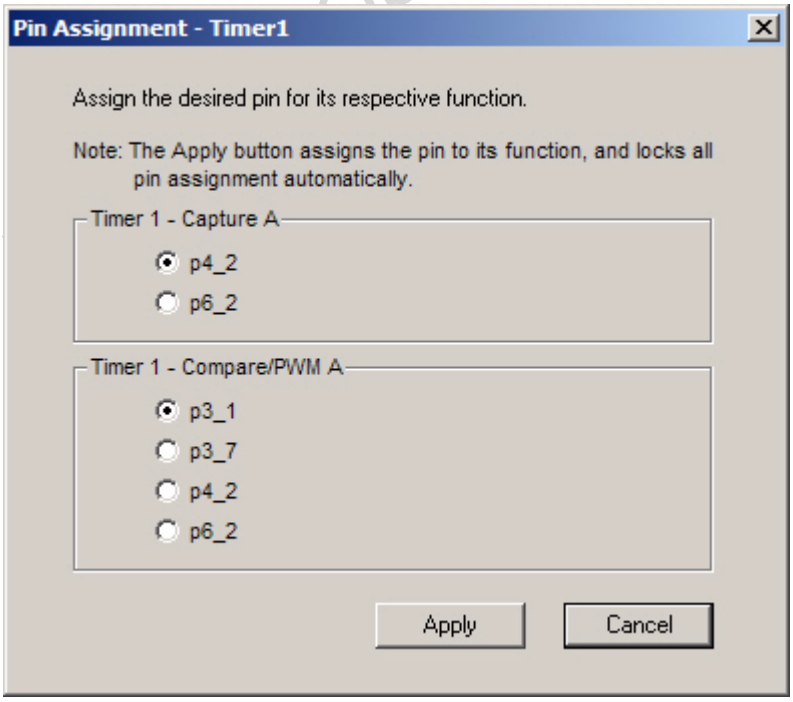
Figure 20. Example - configure Timer1 peripheral



This example specifies the *Internal timer clock* (default) as the clock source, and *Input Capture A* and *Pulse Width Modulation* as the desired counter/timer modes.

Click **Pin Assignment...** to select the pin assignments for the peripheral function. The program displays the dialog shown in *Figure 21*.

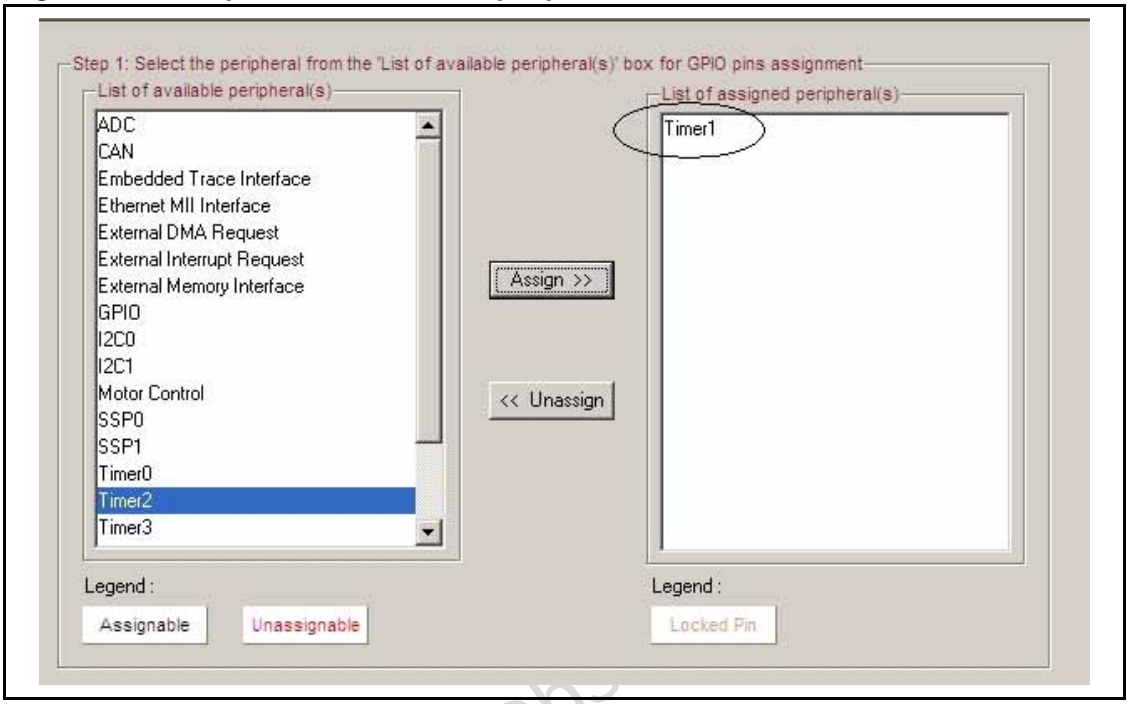
Figure 21. Peripheral pin assignment



Click the desired radio buttons to assign the pins to the respective function. Click the **Apply** button to save the assignment and to automatically lock the pin to the function.

When the dialog shown in *Figure 20* is displayed, click **OK** to accept the configuration and continue. The Timer1 peripheral is selected and displayed in the *List of assigned peripheral(s)*, as shown in *Figure 22.*

Figure 22. Example - selected Timer1 peripheral

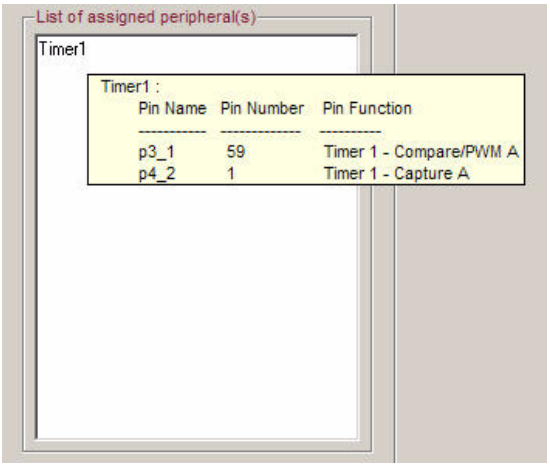


Mouse-over the *Timer1* peripheral in the *List of assigned peripheral(s)* to view pin resource utilization, as shown in *Figure 23.*

The example shows the functions dynamically assigned to pins 1 (P4_2) and 50 (P3_1).

*Note: As an alternative, you may also view resource utilization using the **Tools**, **Generate Project Report** menu, or use **Pin Summary** for a graphic display of pin utilization.*

Figure 23. Example - Timer1 pin assignment display

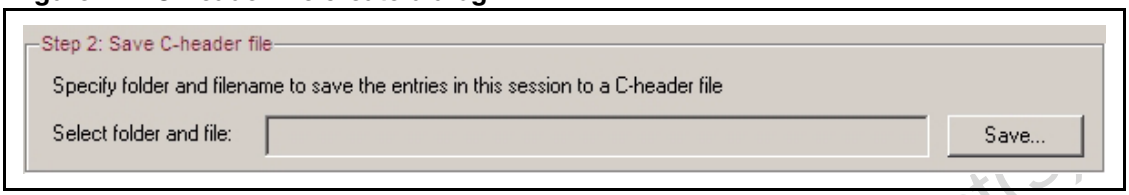


Because of STR91x pin dependencies, some peripherals may become unavailable for assignment as pins are assigned. CAPS, dynamically, makes the necessary pin assignments and reassignments. Peripherals no longer available for assignment, due to GPIO resource limitations, are highlighted in red.

Note: As a design aid, during pin assignment, you may always choose the Pin Summary tab to graphically show the current pin assignments. (See Figure 26.: Pin Summary tab).

The header file *91x_conf.H* is automatically created for your project. This default header file is used by the STR9 HAL library to initialize clock source and peripheral usage. You may, optionally, continue to step 2 of the dialog to create a C-header file. It is recommended that you create the separate C-header file to be included in your source code to reference the pins and their respective functions. (See *Figure 24.: C-header file create dialog*).

Figure 24. C-header file create dialog



Click the **Save** button to save the C-header file.

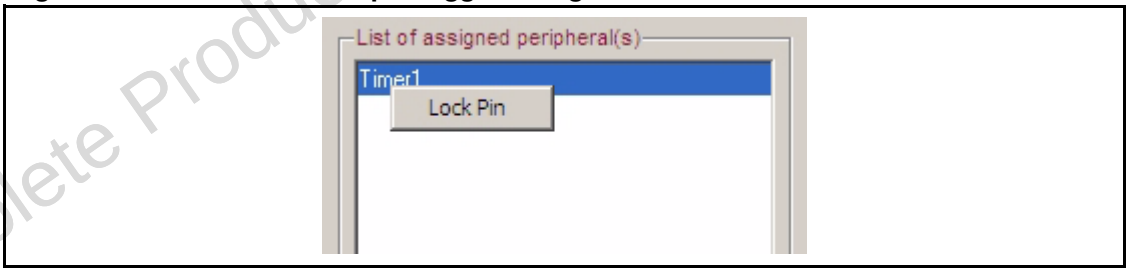
To address incremental design and PCB constraints there are two options for fixing pin assignment.

- Lock-pin feature
- Selective pin function assignment

The *lock-pin feature* gives the user the option to fix a set of assigned pin(s) for the selected peripheral by preventing CAPS from de-allocating those pins during subsequent peripheral selection. The benefit of the lock-pin feature, a toggle feature, is to allow the user to lock the optimal pin grouping assigned by CAPS.

To lock the pin, preventing CAPS pin reassignment, right-click on the desired peripheral in the assigned list and click the **Lock Pin** pop-up button, as shown in *Figure 25.*

Figure 25. Lock and unlock pin toggle dialog



The peripheral mnemonic is grayed when the pin is locked. This is a toggle operation so use the same mechanism, right-click and click the **Unlock Pin** pop-up button, to unlock the pin.

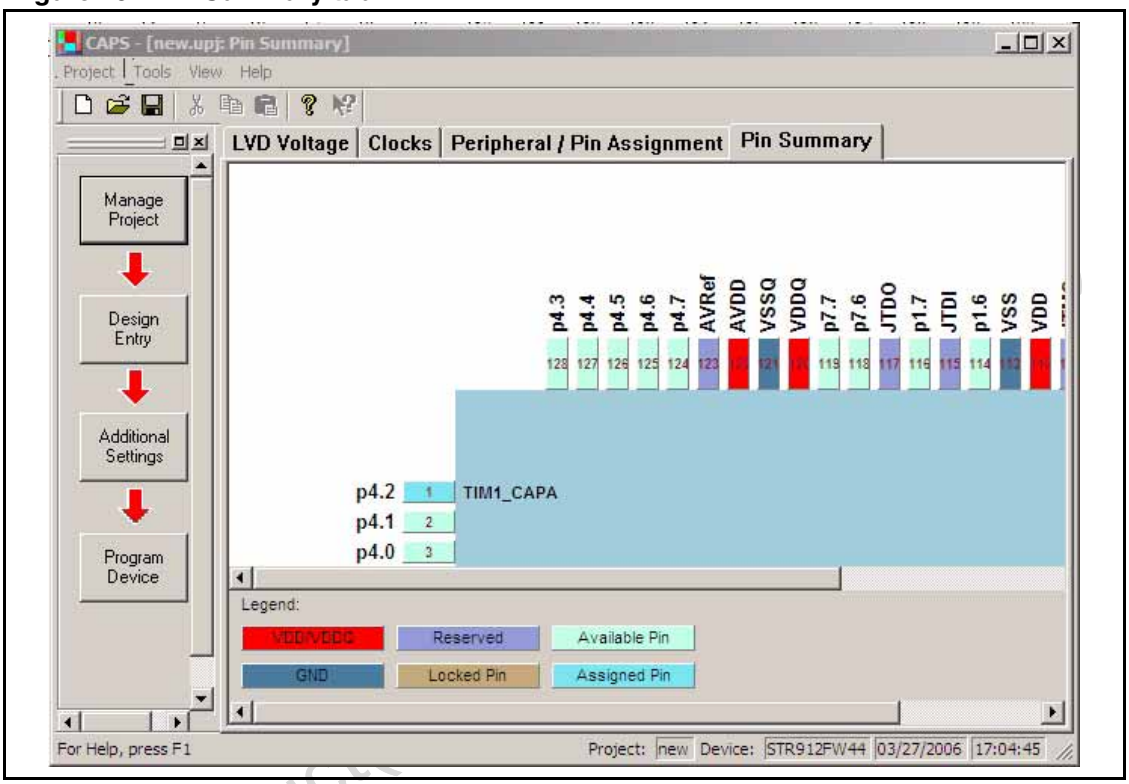
The *selective pin function assignment* option allows advanced users to have full control in selectively assigning the pin functions of the selected peripheral. This option prevents the pin function from being re-routed, fixing the pin for PCB layout.

3.5 View the GPIO pin assignment summary

The pin assignment summary dialog displays the current pin assignments as the assignments are made, following the steps in *Section 3.4: Specifying peripheral and GPIO pin assignment*. The dialog shows the default and user-defined pin assignments, which were assigned according to resource availability.

Click the **Pin Summary** tab to graphically view the current pin assignments. *Figure 26.: Pin Summary tab* shows the **Pin Summary** dialog.

Figure 26. Pin Summary tab



Use the horizontal and vertical scroll bars to view pins outside the display window.

Mnemonic pin names are shown for pins assigned in the **Peripheral/Pin Assignment** step. Pins are color-coded according to the following legend.

Color	Description
Red	Voltage (VCC/VDD)
Dark blue	Ground (GND)
Light blue	Reserved; should not be used.
Light brown	Locked; cannot be assigned or unassigned.
Light cyan	Available pin; pin is available for assignment.
Dark cyan	Assigned pin; pin is used but can be unassigned.

Figure 27. Pin summary with unassigned pins

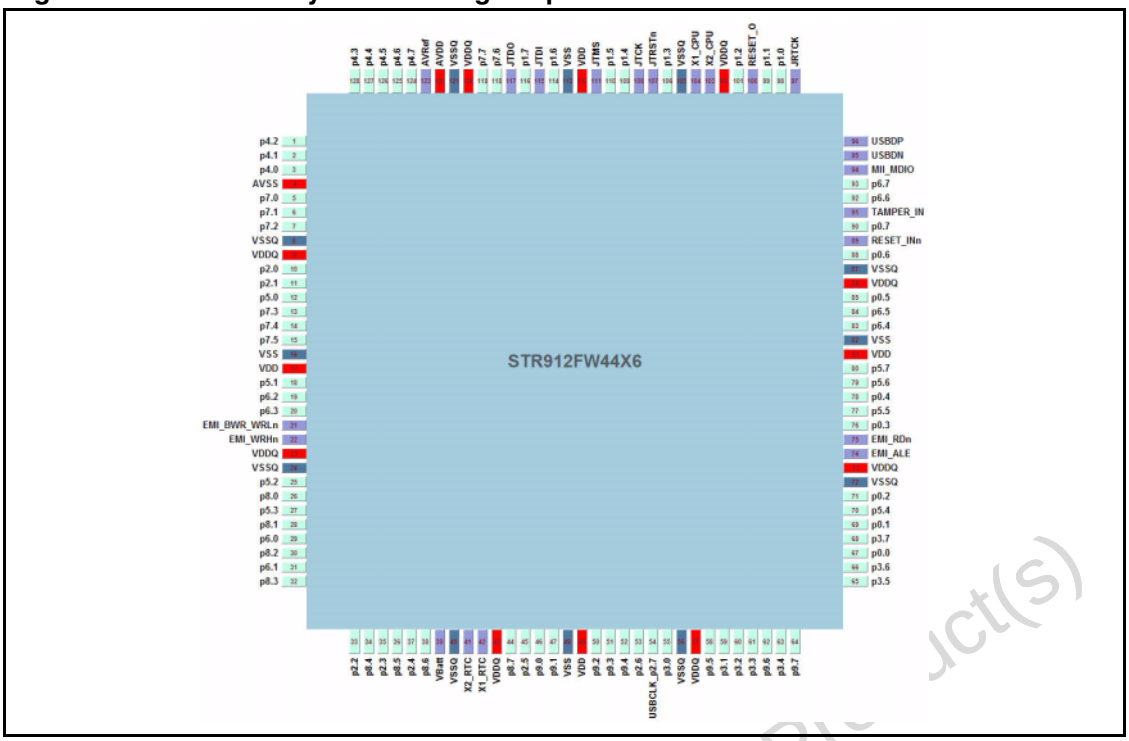
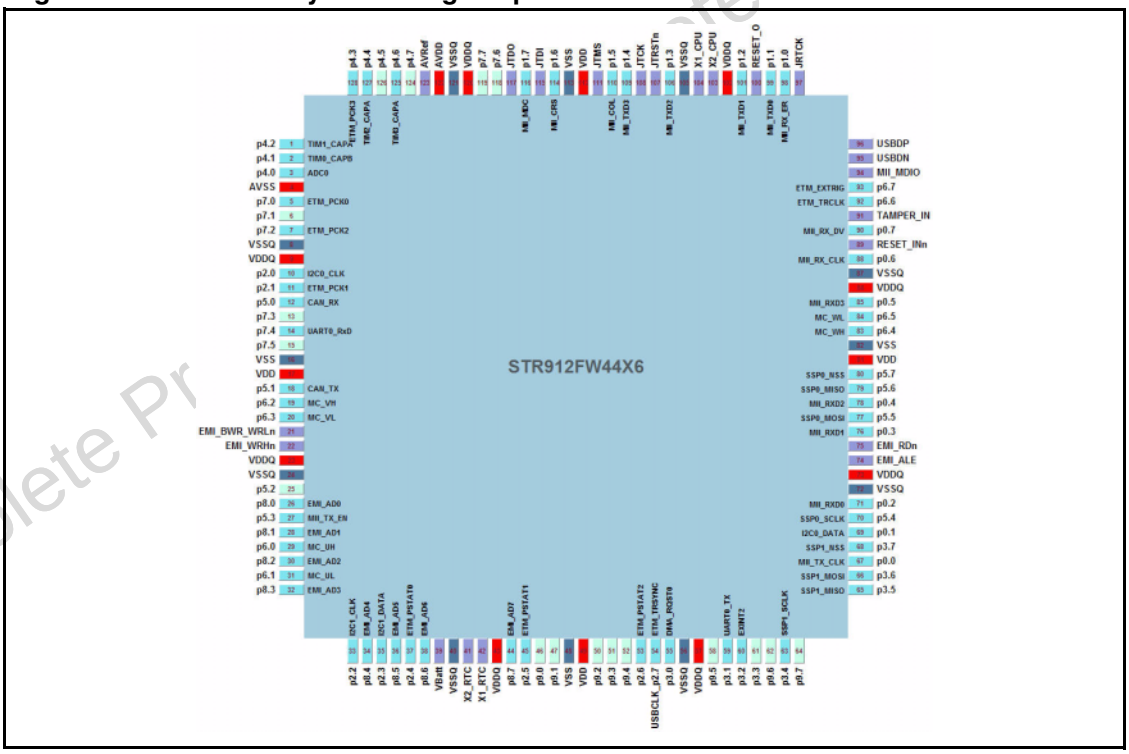


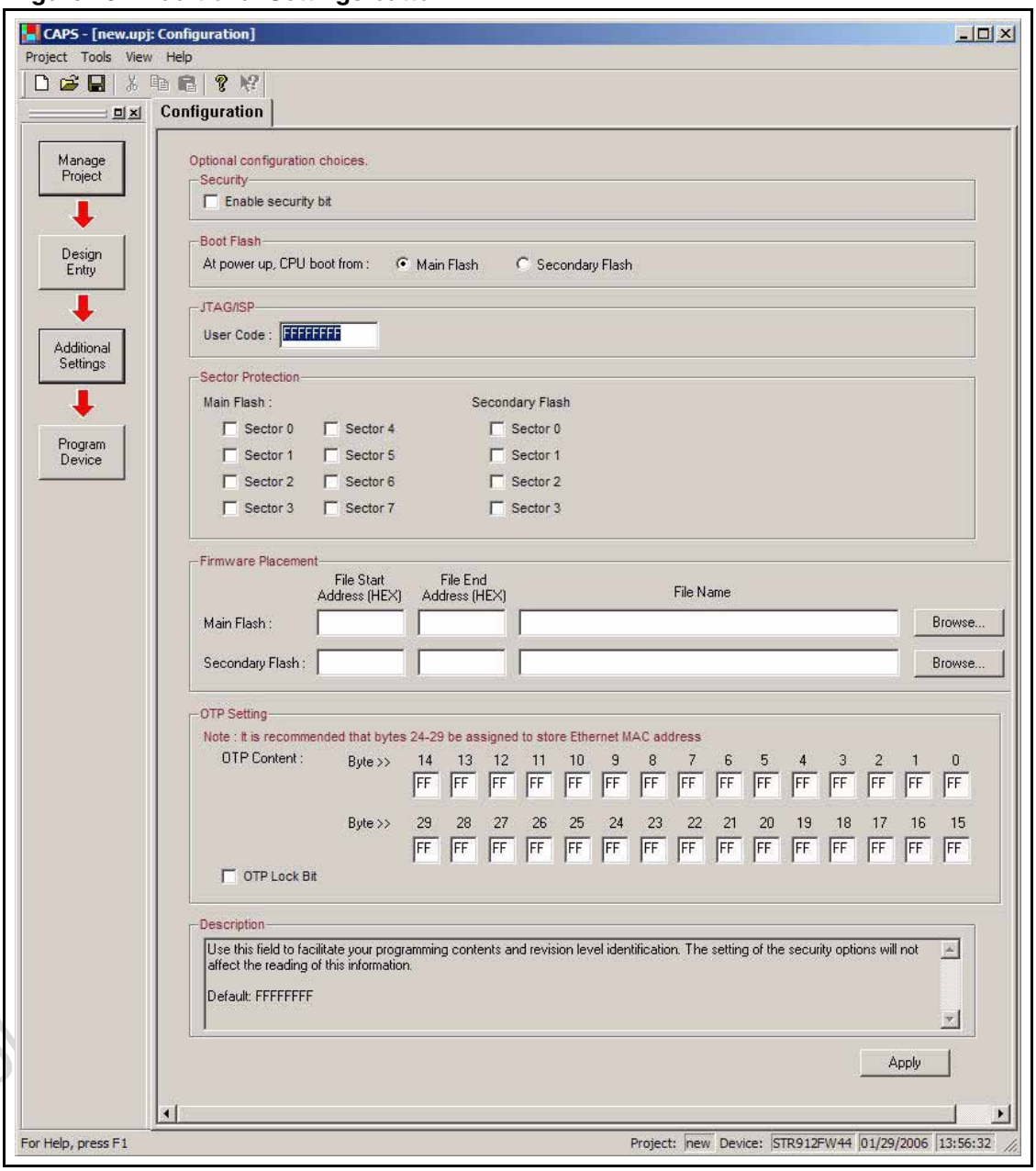
Figure 28. Pin summary with assigned pins



3.6 Configuring optional device parameters

Click the **Additional Settings** button to use the **Configuration** form. (See *Figure 29.: Additional Settings button*).

Figure 29. Additional Settings button



The form provides the interface for the following configuration options.

- Security
- Boot Flash
- JTAG/ISP
- Sector Protection

- Firmware Placement
- OTP Setting

These are discussed in detail in the following subsections.

At the bottom of the form is a **Description** window. This window displays useful information corresponding to the current cursor location in any data entry window on the form. Click the cursor in a data entry window to see the description for that window.

After completing all desired **Configuration** form entries, click the **Apply** button at the bottom of the form to regenerate a new programming data file for the design.

3.6.1 Setting security

Setting the security bit blocks all access to the content of the device by means of JTAG or a conventional programmer. Once the security bit is set, reading or copying the configuration or memory contents of the device is disabled.

In the **Security** dialog, click the *Enable security bit* checkbox to enable security. Uncheck the box do disable security.

Note: The only way to override the security bit is to erase the entire device.

3.6.2 Power-up boot flash selection

This option, in the *Boot Flash* dialog, allows you to select the flash from which the CPU boots at power up. Click the desired radio button to select either main flash or secondary flash.

3.6.3 Setting a JTAG/ISP user code

This option, in the *JTAG/ISP* dialog, allows you to enter a 32-bit code, which can be used for various functions, such as programming-contents and revision-level identification.

Enter the *JTAG/ISP User Code* on the *Configuration* form, entering any 32-bit hexadecimal value. The default value is FFFFFFFF.

3.6.4 Setting sector protection

Individual NVM segments within the device may be statically write-protected to prevent accidental data loss.

Click the checkbox for the sector you wish to write-protect. Uncheck the box to disable protection. Protection granularity is eight sectors in *Main Flash* and four sectors in *Secondary Flash*.

3.6.5 Firmware placement

The firmware file may be located in main or secondary flash memory.

In the *Firmware Placement* dialog, enter the start and end hexadecimal addresses and the firmware filename for the desired flash area.

3.6.6 Setting OTP programmable memory bytes

The *OTP Settings* dialog allows you to write values to and lock programmable memory.

Thirty-two bytes of OTP memory can be programmed one time, only, through either the JTAG programmer or by the CPU.

Note: These bytes can never be altered once they are programmed.

Enter the hexadecimal value in the numbered byte location. *Table 2.: Reserved OTP memory bytes* shows the reserved and recommended OTP memory usage.

Table 2. Reserved OTP memory bytes

'Byte Number	Reserved/Recommended Usage
24-29	Ethernet MAC address
31	Mask identifier
32	Family identifier

Setting the OTP lock bit disables any further writing to OTP memory. The lock bit, itself, is also one-time programmable. If the OTP array is unlocked, it is always possible to write to an OTP byte location that has not been written. It is never possible to change the value of an OTP byte location if any one bit of the byte location was previously written.

Check the **OTP Bit Lock** checkbox to disable writes to OTP memory.

3.7 Validating and programming the target device

After completing the design of your project, you are ready to program the firmware into your target device.

Before programming the device, perform the following steps.

1. Set your default programming to use either *Single device view* or *Multiple device view,* using the **Project** | **Preferences** menu selection shown in *Figure 30.: Project preferences dialog*.

Figure 30. Project preferences dialog

Use single device view if there is only one device on your board. Use multiple device view if multiple devices are daisy-chained. Click the radio button corresponding to the desired view.

A JTAG chain is defined to be two or more JTAG-compliant, IEEE 1149.1 standard devices connected together in a chain. Each device in the chain must support the four basic JTAG signals: TDI, TDO, TCK and TMS.

The following JTAG chaining rules apply.

– Different brands of chips must be set in "ByPass" operation and will require different programming software.

– Only one device in a chain will be programmed at a time, not concurrently.
– Before any JTAG operation can begin, the chain order must be defined and the instruction register length for each device must be known by the software.

*Note: The order of the chain is important when programming the devices. You must set up the chain in your programming software such that it matches physical ordering of the devices on your board. The multiple device view, in **Project Preferences...** allows you to build the JTAG-ISP chain.Next, setup and power your target device as described in 1.2: Setting up the target hardware.*

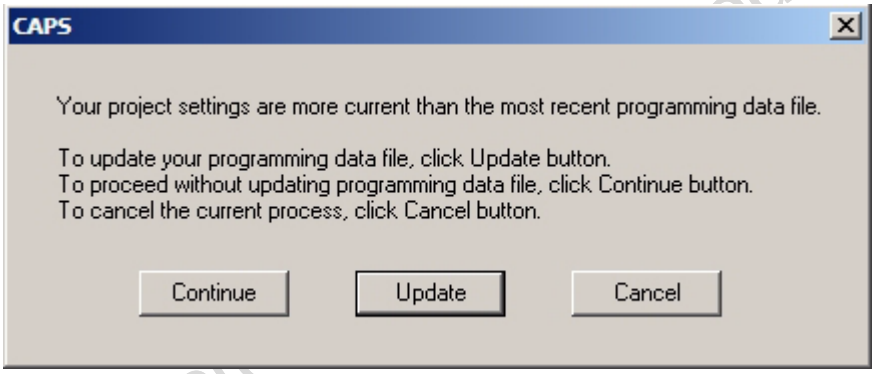
2. Setup and power your target device as described in *1.2: Setting up the target hardware*.

CAPS setup and the hardware are now ready for programming the device.

Click the **Program Device** design flow button to access the device verification and programming dialog.

CAPS verifies the changes made in the previous design steps were applied, by clicking the **Apply** button, before continuing with device programming. If the current open project settings differ from the saved data file for the project, the dialog shown in *Figure 31.: Project file synchronization dialog* gives the option to synchronize with the saved project file.

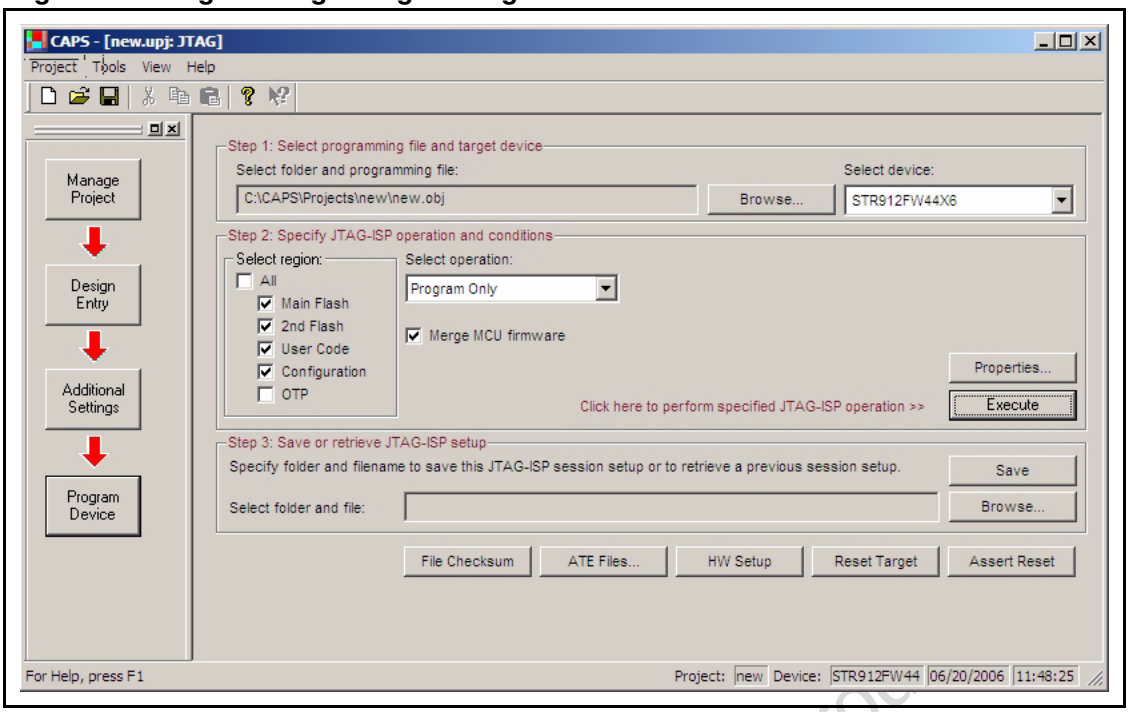
Figure 31. Project file synchronization dialog



Click **Continue** or **Cancel** to proceed without updating the programming data file. Click **Update** to write the current project settings to the saved file.

If *Single device view* is selected in the **Project Preferences...** menu, the program displays *Figure 32.: Programming dialog for single device*. (All discussion for the single device dialog also applies to the multiple device dialog. The multiple device dialog provides additional chaining options, which are discussed in *3.7.5: Chaining multiple devices*).

Figure 32. Programming dialog for single device



The dialog guides you through the following programming steps.

1. *Select programming file and target device*

   By default, the programming data file for the currently active session is displayed. Use the **Browse** button to open another, previously saved programming data file.

   The type of target device associated with the programming data file is shown in the *Select device* box. Use the drop-down menu to change the device type.

Note: All other file-related operations within this dialog window operate on the current programming data file.

2. *Specify JTAG-ISP operation and conditions*

   Choose one of the available JTAG-ISP operations using the *Select operation* drop-down menu. Refer to *3.7.1: JTAG-ISP operations* for a more detailed discussion of the programming options.

Note: These operations should only be performed on a target device in a known, operational state.

The JTAG-ISP operation may be applied to the following device memory regions.

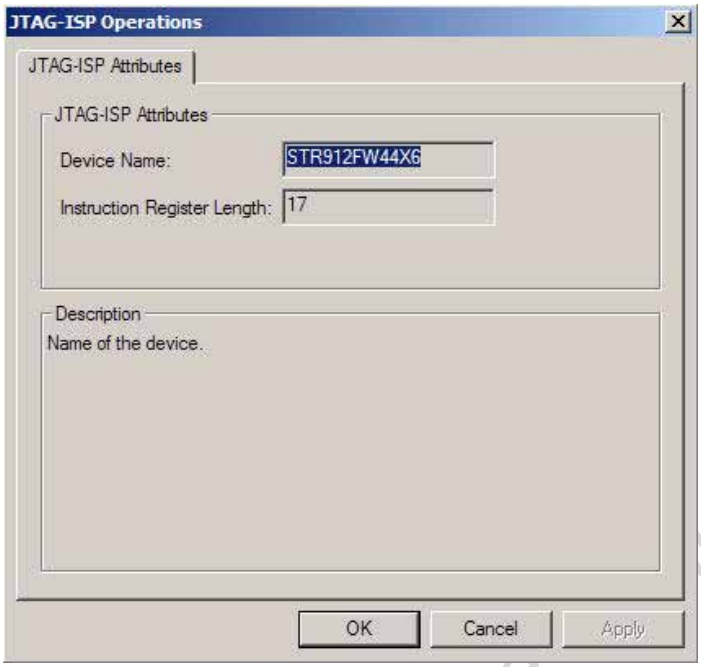
- Main Flash
- Secondary Flash
- User Code
- Configuration
- OTP

One or more regions may be selected simultaneously by clicking the desired checkbox(es). As a convenience, click the *All* checkbox to select all regions for the JTAG-ISP operation.

To automatically merge the MCU firmware, click the *Merge MCU firmware* checkbox.

Click the **Properties...** button to view the JTAG-ISP attributes. This displays the configuration dialog shown in *Figure 33.: JTAG-ISP attributes dialog*.

Figure 33. JTAG-ISP attributes dialog



3. Save or retrieve JTAG-ISP setup

To save the JTAG/ISP settings entered in the preceding steps, click the **Save** button. Use the browser dialog to enter the file name and directory location.

This saves a JTAG Chain File with extension *.jcf*. Later, you may use the **Browse** button to open the file to restore your current settings.

4. Additional functions

If this is the first time programming the device, click the **HW Setup** button to set the target device to a known state by select and validate the communication link. This operation is discussed in detail in section *3.7.4: Target hardware verification*.

Click the **Checksum** button to generate NVM checksums. This operation is discussed in detail in section *3.7.2: Checksum the programming file*.

Click the **ATE file...** button to generate ATE files. This operation is discussed in detail in section *3.7.3: Generate ATE file*.

Click **Reset Target** to reset the target device.

Click **Assert Reset** to hold the reset signal low. This toggle option provides a way to hold the reset signal low, at the user's discretion. When the option is selected, the reset signal is continuously asserted. Click **De-Assert Reset** to release the reset signal.

3.7.1 JTAG-ISP operations

The following steps provide complete JTAG-ISP programming functionality.

1. Use the *Select operation* drop-down menu to choose from the following programming operations.

Blank Test	Determine if any region of the device has been programmed.
Erase	Erase one or more regions of the device.
Program/Verify	Write code, configuration or data to the device based on the contents of the programming data (*.obj*) file. Following programming, compare the actual contents of the device with the contents of the current programming data (*.obj*) file.
Program Only	Write code, configuration or data to the device based on the contents of the programming data (*.obj*) file.
Verify	Compare the actual contents of the device with the contents of the current programming data (*.obj*) file.
Upload	Read the contents of the device and save the contents to a programming data (*.obj*) file.
ByPass	Place the device in bypass mode. This option has limited use in single device view mode, other than to verify that the JTAG interface works correctly.

2. Use the *Select region* checkbox to select the device memory region for the programming operation.

All	The entire device
Main Flash	The main flash region only
2nd Flash	The secondary flash region only
User Code	The user code region only
Configuration	Configuration data region only

OTP	One-time-programmable region only

3. Click the **Properties...** button to view the JTAG-ISP Attributes.
   This shows the device name and its instruction register length. This information may be useful to setup the JTAG chain for 3rd party programming tools that do not have an auto-detect function.
4. Click the **Execute** button to perform the programming operation selected, above.
   If you attempt to program a device that is not blank, a warning message verifies that you "want to do the full chip erase" before continuing. Click **Yes** to continue.

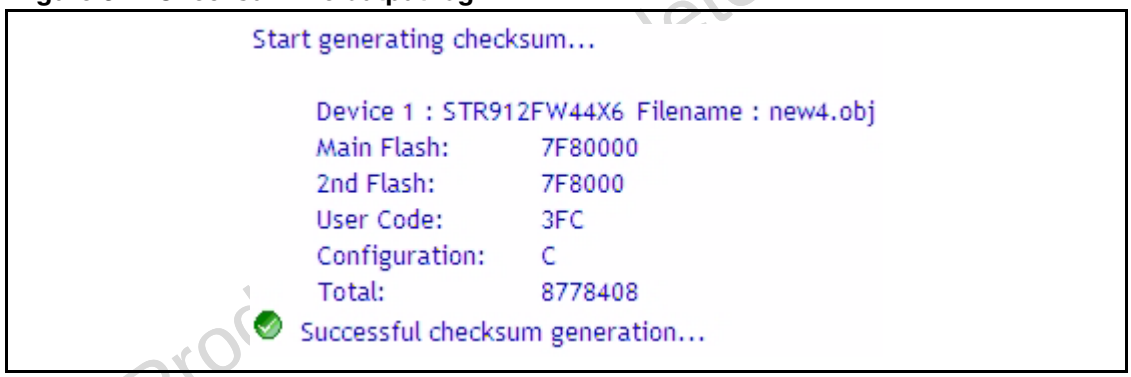
3.7.2 Checksum the programming file

Click the **File Checksum** button to generate checksums for the current programming file. The following checksums are generated.

- Main flash
- Secondary flash
- User code
- Configuration data area, including OTP memory
- The entire file

The output log, if enabled, shows the results of the operation, as shown in *Figure 34.: Checksum file output log*.

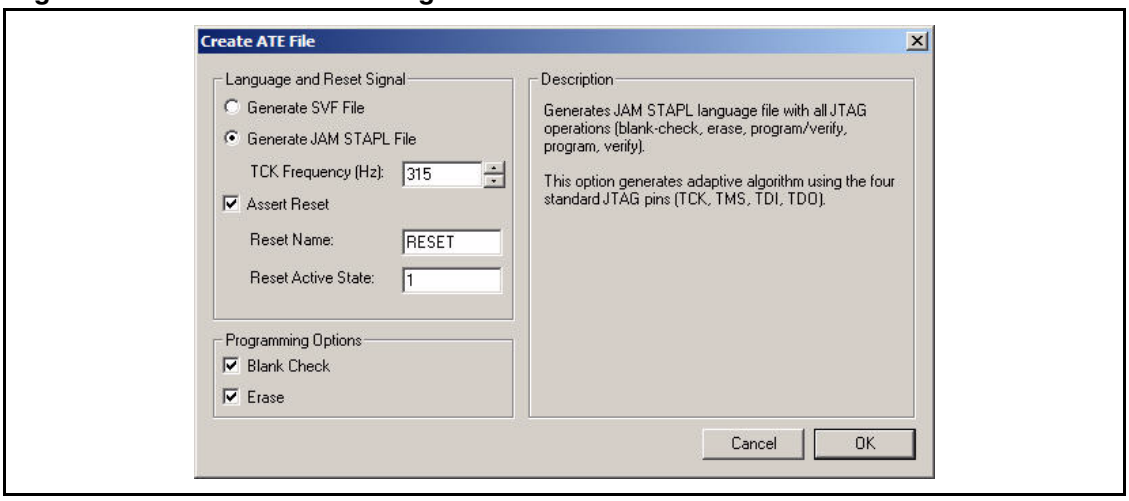
Figure 34. Checksum file output log



3.7.3 Generate ATE file

For STR9 devices, the Serial Vector Format (SVF) or JEDEC Standard Programming Language (STAPL) format file is not supported. The **ATE Files...** feature should not be used.

Figure 35. Create ATE File dialog



The description frame on the right of the dialog provides a context-sensitive help for the various selection options.

1. Click the **OK** button to accept the specified file format. A browser dialog window appears, allowing you to enter the file name and directory location to save the file.
2. Click the **Save** button. The file is saved to disk and “Successful ATE file creation ...” displays in the output log.

3.7.4 Target hardware verification

Use the programming dialog to perform the following functions

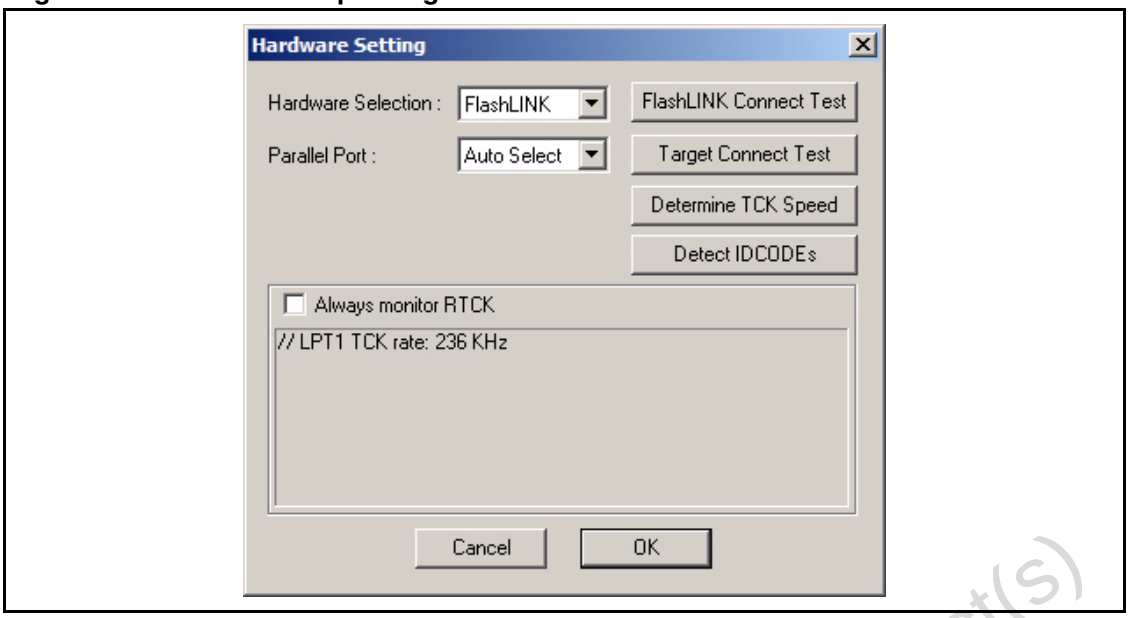
- Set the target hardware to an initial state.
- Specify the type of communication link.
- Perform loopback tests on the link and validate target board connectivity.
- List the IDCODEs of all devices in the JTAG chain.

Click the **HW Setup** button to display the hardware setup dialog shown in *Figure 36.: Hardware setup dialog*.

Note: 1 The target device and communication links must first be connected and powered, as described in 1.2: Setting up the target hardware.

*2 It is recommended that you reset the target device before performing these operations by clicking the **Reset Target** button.*

Figure 36. Hardware setup dialog



1. Use the drop-down menu to choose the type of communication link (*RLink* or *FlashLINK*) connected between the PC and target board.

RLink	Communication using a USB port.
FlashLINK	Communication using a parallel port.

   If *FlashLINK* is selected, also click *Parallel Port* and choose *Auto Select* or the LPT port from the drop-down menu.

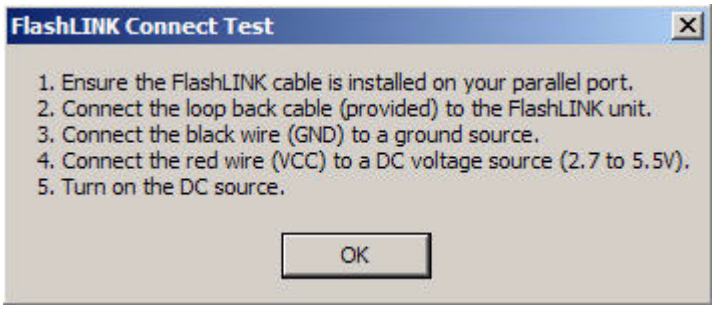
2. Check the *Always monitor RTCK* checkbox to use the RTCK to pace the TCK clock frequency, from the external JTAG test equipment.

Note: Using RTCK requires another connector pin.

3. Click the **FlashLINK Connect Test** button or the **RLink Connect Test** button, as applicable, to confirm connectivity between your PC and the target board.

   The dialog shown in *Figure 37.: Link connectivity test dialog* appears, showing the preliminary steps required before continuing.

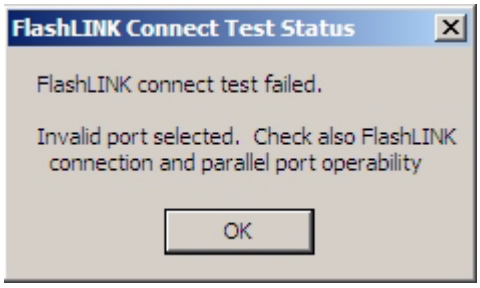
Figure 37. Link connectivity test dialog



When you are ready, click **OK** to continue. If the test passes, a success message is displayed. Click **OK** to continue.

If the test fails, the dialog shown in *Figure 38.: Link connect test error dialog* displays. (A similar dialog displays for the RLink option).

Figure 38. Link connect test error dialog



Press **OK** to continue, and check the physical connections between your PC and target board by performing a loopback test.

4. Click the **Target Connect Test** button to test communication from your PC through the JTAG chain on the target board.

Note: It is recommended that link and connectivity tests be run before attempting any JTAG operations.

The dialog shown in *Figure 39.: Target connectivity test dialog* displays, showing the preliminary steps required for a successful test.

Figure 39. Target connectivity test dialog

If the JTAG chain on the target board successfully responded, a message displays indicating the test passed. Press **OK** to continue.

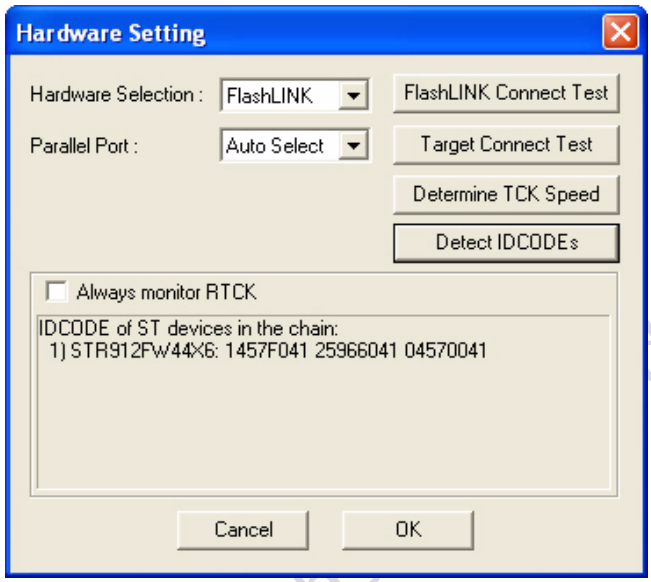
If the test fails, the error message shown in *Figure 39.: Target connectivity test dialog* displays.

Figure 40. Target connect test error dialog

Check the JTAG chain connectivity on the target board and that the target board is powered on. Click **OK** to continue.

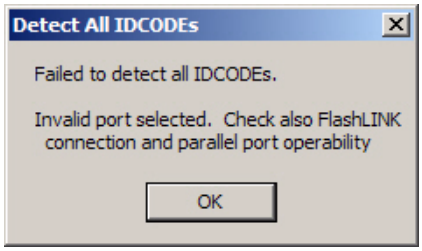
5. Click **Determine TCK Speed** button to determine the maximum attainable TCK rate, in KHz.
6. Click **Detect IDCODEs** to display the IDCODEs of CAPS-supported devices in the JTAG chain. For unsupported devices, the string “Non-ST device” is displayed. If the detection passes, the detected IDCODEs are listed in the dialog shown in *Figure 41.: Detected JTAG IDCODEs dialog*.

Figure 41. Detected JTAG IDCODEs dialog



If the detection fails, the dialog shown in *Figure 42.: JTAG IDCODEs detection error dialog* displays. Check the JTAG chain connectivity on the target board and that the target board is powered on. Click **OK** to continue.

Figure 42. JTAG IDCODEs detection error dialog



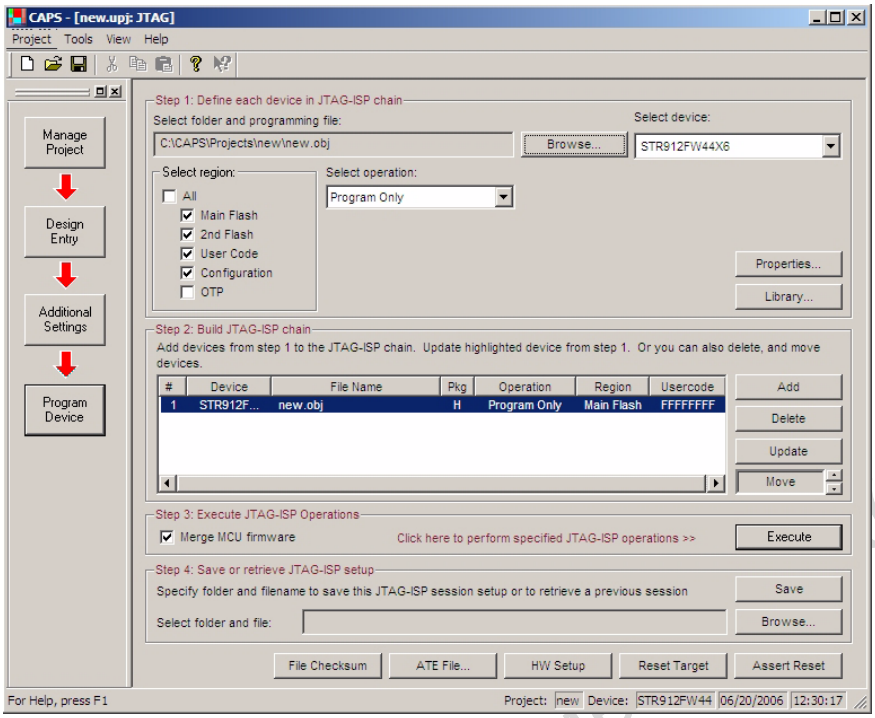
3.7.5 Chaining multiple devices

This section describes procedures for specifying chained devices and their order. Refer to the other subsections of *3.7: Validating and programming the target device* for operations common to both single and multiple devices.

If *Multiple device view* is selected, the program displays *Figure 43.: Programming dialog for multiple devices* when the **Program Device** button is clicked. The multiple device view adds dialog for chaining devices to the single device dialog.

Click the **Library** button to add non-ST devices to the chain.

Figure 43. Programming dialog for multiple devices



The edit windows contain default values for the current project.

Follow these steps **for each device** in the device programming chain.

1. Click the **Browse** button to use the file browser to load a saved programming data (*.obj*) file.
2. After opening the programming data file, click the **Add** button to add the file to the JTAG chain list window. Highlight a file in the list and click **Delete** to remove the file from the listed. Highlight the device and click the **Move** scroll buttons to reorder the chain list.

Note: 1 The list device order must match the physically device order.

2 Device #1 should be the first device on the board that has its TDI pin connected to the JTAG/ ISP programmer.

3. To change the device type, highlight the device in the device list, then select the desired device type from the *Select device* drop-down menu. Click the **Update** button to apply the change.
4. To change the memory region of a device, highlight the device in the device list, then click the *Select region* checkbox for the desired region. Multiple regions, including *All*, may be checked. Click the **Update** button to apply the change.

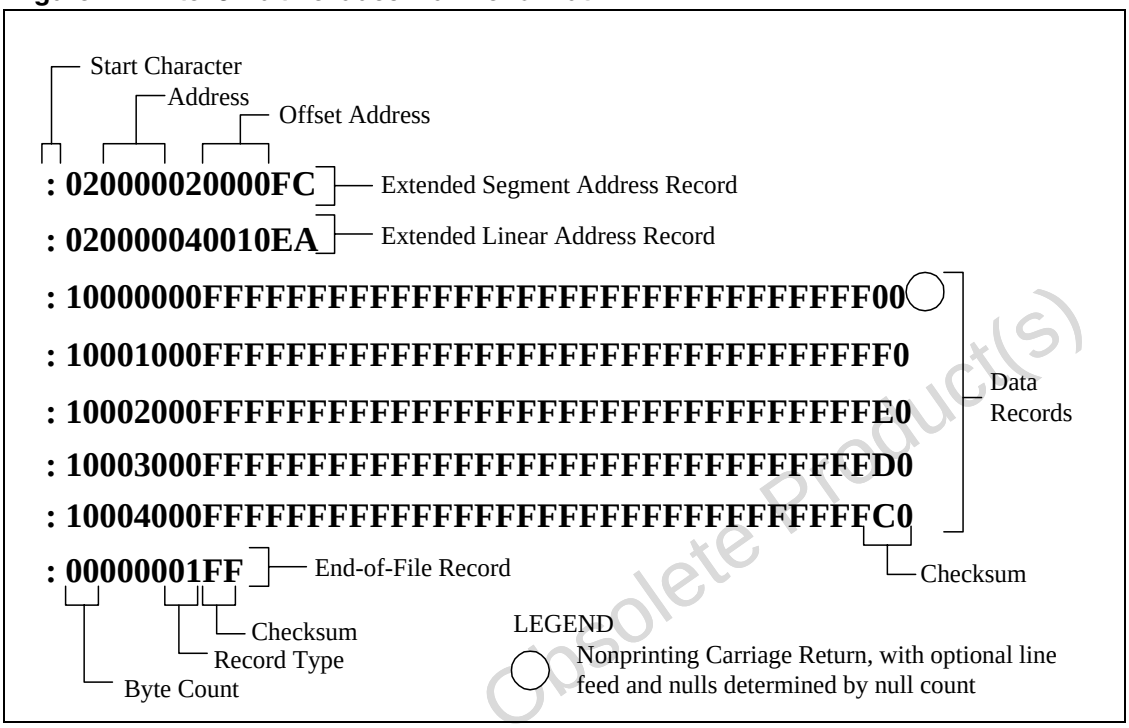
Note: Different regions can be specified for each device.

5. Once the JTAG chain and device properties are defined, click the **Execute** button to perform the selected programming operation.

Appendix A Intel hex-32 record format

The Intel 32-bit hexadecimal file record format has a 9-character (4-field) prefix that defines the start of the record, byte count, load address, record type, and a 2-character checksum suffix. The *.hex* file (*Figure 44.* illustrates the sample records of this format).

Figure 44. Intel 32-bit hexadecimal file format



Four record types are defined:

- Data record
- End record
- Extended segment address record
- Extended linear address record

A.1 Data record

This record begins with the colon (:) start character, which is followed by the byte count (in hexadecimal notation), the address of the first data byte, and the record type (equal to “00”). Following these are the data bytes. The checksum follows the data bytes and is the two’s complement (in binary) of the preceding bytes in the record, including the byte count, address, record type, and data bytes.

A.2 End record

This end-of-file record also begins with the colon (:) start character and is followed by the byte count (equal to “00”), the address (equal to “0000”), the record type (equal to “01”), and the checksum, “FF.”

A.3 Extended segment address record

This is added to the offset to determine the absolute destination address. The address field for this record must contain ASCII zeros (hexadecimal 30s). This record type defines Bits 4 to 19 of the segment base address; it can appear randomly, anywhere within the file and affects the absolute memory address of subsequent data records in the file. The following example illustrates how the extended segment address is used to determine a byte address.

Problem:

Find the address for the first data byte for the following file:

```
:02 0000 04 0010 EA
:02 0000 02 1230 BA
:10 0045 00 55AA FF ..... BC
```

Solution:

1. Find the extended linear address offset for the data record (0010 in the example).
2. Find the extended segment address offset for the data record (1230 in the example).
3. Find the address offset for the data from the data record (0045 in the example).
4. Calculate the absolute address for the first byte of the data record as follows:

```
00100000  Linear address offset, shifted left 16 bits
+  12300  Segment address offset, shifted left 4 bits
+   0045  Address offset from data record

________
00112345  32-bit address for first data byte
```

The address for the first data byte is 112345.

Note: Always specify the address offset when using this format, even when the offset is zero.

During output translation, the firmware forces the record size to 16 (decimal) if the record size is specified greater than 16. There is no such limitation for record sizes specified less than 16.

A.4 Extended linear address record

This record specifies bits 16–31 of the destination address for the data records that follow. This address is added to the offset to determine the absolute destination address, and can appear anywhere within the file. The address field for this record must contain ASCII zeros (hexadecimal 30s).

Appendix B Project Report

This is an example of a project report.

The report includes:

- CAPS version, project name, location, description and target device
- Design entry settings and configuration specifications
- Peripheral/pin assignment

Sample report:

```
*************************************************************************
* Project file generated by CAPS Version x.xx - 3/20/2006 15:32:08
* Project Name           : Example
* Project Folder         : C:\uPSDsoft\Projects\Example
* Project Description    : This is an example
* Target Device          : STR912FW44X6
*************************************************************************
Voltage
=======

  VDDQ LVD threshold voltage : 2.7 V
  LVD reset option           : VDD or VDDQ
  LVD warning option         : VDD or VDDQ
-------------------------------------------------------------------------
Clocks
======

  Master clock source        : PLL
  Main crystal frequency     : 25000000.00 Hz
  Master clock frequency     : 96000000.00 Hz
  M value of PLL             : 25
  N value of PLL             : 192
  P value of PLL             : 2
  RClk divisor value         : 4
  RClk frequency             : 24000000.00 Hz
  HClk divisor value         : 1
  HClk frequency             : 24000000.00 Hz
  PClk divisor value         : 2
  PClk frequency             : 12000000.00 Hz
  FMI divisor value          : 1
  FMI CLK frequency          : 24000000.00 Hz
-------------------------------------------------------------------------
Additional Setting
==================

  At power up, CPU boot from     : Main Flash
  JTAG user code                 : 24689ADC
  Device Security Protection     : Off

  Sector Protection :
        Main Flash           Protection Status
        ----------           -----------------
        Sector 0             protected
        Sector 1             unprotected
        Sector 2             unprotected
        Sector 3             unprotected
        Sector 4             unprotected
        Sector 5             unprotected
```

```
        Sector 6            protected
        Sector 7            protected

        2nd  Flash          Protection Status
        ----------          -----------------
        Sector 0            protected
        Sector 1            protected
        Sector 2            unprotected
        Sector 3            unprotected
  Firmware Setting
  ================
    Main Flash start address       : 0
    Main Flash end address         : 5FFF
    File path for Main Flash       : C:\uPSDsoft\Projects\Example\MyCode.hex
  OTP Setting
  ===========
    OTP Content :
      Byte >>   14  13  12  11  10  9   8   7   6   5   4   3   2   1   0
      Data >>   FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF

      Byte >>   29  28  27  26  25  24  23  22  21  20  19  18  17  16  15
      Data >>   FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF

    OTP lock bit  : Off

--------------------------------------------------------------------------------
Peripheral/Pin Assignment
=========================

  Peripheral   Pin        Pin         Pin
  Name         Name       Number      Function
  -------      --------   ------      --------
  GPIO :
               p9_1       47          GP I/O mode - Output (Push-Pull)
  External DMA Request :
               p3_0       55          External DMA Request 0
  Embedded Trace Interface :
               p0_0       67          ETM Packet 0
               p0_1       69          ETM Packet 1
               p0_2       71          ETM Packet 2
               p0_3       76          ETM Packet 3
               p0_4       78          ETM Pipe Stat 0
               p0_5       85          ETM Pipe Stat 1
               p0_6       88          ETM Pipe Stat 2
               p0_7       90          ETM Trace Sync
               p1_0       98          ETM External Trigger
               p1_5       110         ETM Trace Clock
  SSP0 :
               p2_4       37          Master Clock Out
               p2_5       45          Master Data Out
               p2_6       53          Master Data In
               p2_7       54          Master Select Out
  UART0 :
               p5_0       12          UART0 TxD
               p5_1       18          UART0 RxD
  I2C0 :
               p1_4       109         I2C0 bus - Clock In
               p1_6       114         I2C0 bus - Data Out
               p1_4       109         I2C0 bus - Clock Out
               p1_6       114         I2C0 bus - Data In
  Timer0 :
               p4_0       3           Timer 0 - Compare/PWM A
               p4_1       2           Timer 0 - Capture B
```

Appendix C HAL library C-header file example

This is an example of the HAL library C-header file, *91x_conf.h*.

```
/************* Copyright (C) 2005-2006 STMicroelectronics, Inc. ************
* Header file generated by CAPS Version x.xx Alpha - 4/20/2006 11:52:28
* Project Name           : new912_128
* Project Folder         : C:\CAPS\Projects\new912_128
* Project Description    : This is an example
* Target Device          : STR912FW44X6
*******************************************************************************/
#ifndef __new912_128_h
#define __new912_128_h
// *****************************************************************
// Device name
// *****************************************************************
#define _Device_Type STR912FW44X6
// *****************************************************************
// System (AHB) and peripheral (APB) Buses
// *****************************************************************
#define _AHBAPB  // always
#define _AHBAPB0  // always
#define _AHBAPB1  // always
// *****************************************************************
// Vectored interrupt controller
// *****************************************************************
#define _VIC  // always
#define _VIC0  // always
#define _VIC1  // always
// *****************************************************************
// Flash memory interface
// *****************************************************************
#define _FMI  // always
// *****************************************************************
// Wakeup interrupt unit
// *****************************************************************
#define _WIU  // always
// *****************************************************************
// Real time clock
// *****************************************************************
#define _RTC  // always
// *****************************************************************
// System control unit
// *****************************************************************
#define _SCU  // always
// *****************************************************************
// Watch dog
// *****************************************************************
#define _WDG  // always
// *****************************************************************
// DEBUG
// *****************************************************************
#define DEBUG  // always
// *****************************************************************
// Direct memory access
// *****************************************************************
#define _DMA  // always
#define _DMA_Channel0  // always
#define _DMA_Channel1  // always
#define _DMA_Channel2  // always
```

```
#define _DMA_Channel3  // always
#define _DMA_Channel4  // always
#define _DMA_Channel5  // always
#define _DMA_Channel6  // always
#define _DMA_Channel7  // always
// ************************************************************
// Clock source
// ************************************************************
#define _MCLK_Source SCU_MCLK_PLL
#define _Main_Crystal     25000.00  // In KHz
// ************************************************************
// Phase Lock Loop (PLL) setting
// ************************************************************
#define _PLL_M 25 // M (1 <= M <= 255) value for phase lock loop
#define _PLL_N 192 // N (1 <= N <= 255) value for phase lock loop
#define _PLL_P 2 // P (1 <= P <= 5) value for phase lock loop
// ************************************************************
// clocks divisor setting
// ************************************************************
#define _RCLK_Divisor   SCU_RCLK_Div1 // Reference clock divisor
#define _HCLK_Divisor   SCU_HCLK_Div1 // ARM high speed bus divisor
#define _PCLK_Divisor   SCU_PCLK_Div2 // ARM Peripheral bus divisor
#define _FMICLK_Divisor SCU_FMICLK_Div1 // FMI divisor
// ************************************************************
// GPIO input register setting
// ************************************************************
#define GPIOIN0 0x0
#define GPIOIN1 0x1
#define GPIOIN2 0x0
#define GPIOIN3 0x1
#define GPIOIN4 0x4
#define GPIOIN5 0x0
#define GPIOIN6 0x8
#define GPIOIN7 0x0
// ************************************************************
// GPIO output register setting
// ************************************************************
#define GPIOOUT0 0xFFFF
#define GPIOOUT1 0xC00
#define GPIOOUT2 0x0
#define GPIOOUT3 0x30C
#define GPIOOUT4 0x80
#define GPIOOUT5 0x0
#define GPIOOUT6 0x0
#define GPIOOUT7 0x0
// ************************************************************
// GPIO type register setting
// ************************************************************
#define GPIOTYPE0 0x0
#define GPIOTYPE1 0x0
#define GPIOTYPE2 0x0
#define GPIOTYPE3 0x0
#define GPIOTYPE4 0x0
#define GPIOTYPE5 0x0
#define GPIOTYPE6 0x0
#define GPIOTYPE7 0x0
#define GPIOTYPE8 0x0
#define GPIOTYPE9 0x0
// ************************************************************
// GPIO direction register setting
// ************************************************************
#define GPIO_DIR0 0x0
#define GPIO_DIR1 0x0
```

```
#define GPIO_DIR2 0x0
#define GPIO_DIR3 0x0
#define GPIO_DIR4 0x0
#define GPIO_DIR5 0x0
#define GPIO_DIR6 0x0
#define GPIO_DIR7 0x0
#define GPIO_DIR8 0x0
#define GPIO_DIR9 0x0
// ****************************************************************
// GPIO
// ****************************************************************
#define _GPIO
#define _GPIO0
#define _GPIO1
#define _GPIO2
#define _GPIO3
#define _GPIO4
// #define _GPIO5
#define _GPIO6
// #define _GPIO7
// #define _GPIO8
// #define _GPIO9
// ****************************************************************
// ADC
// ****************************************************************
#define _ADC // ADC enabled : At least one of ADC channels is used
#define GPIO_ANAChannel4 0x10 // ADC Channel 4 is selected
#define GPIO_ANAChannel5 0x20 // ADC Channel 5 is selected
// ****************************************************************
// CAN
// ****************************************************************
// #define _CAN  // CAN
// ****************************************************************
// EMI (External Memory Interface)
// ****************************************************************
// #define _EMI  // EMI
#define SCU_EMI 0x0 // Port8 and Port 9 is not connected to EMI block
// ****************************************************************
// ENET (Ethernet)
// ****************************************************************
// #define _ENET  // Ethernet
// ****************************************************************
// I2C
// ****************************************************************
// #define _I2C  // I2C
// #define _I2C0  // I2C0 serial interface
// #define _I2C1  // I2C1 serial interface
// ****************************************************************
// MC (Motor Control)
// ****************************************************************
// #define _MC  // Induction Motor Control
// ****************************************************************
// SSP (Synchronous Serial Port)
// ****************************************************************
// #define _SSP  // Synchronous Serial Port
// #define _SSP0  // Synchronous Serial Port 0
// #define _SSP1  // Synchronous Serial Port 1
// ****************************************************************
// Timer
// ****************************************************************
#define _TMR  // Timer => At least one of the timers is enabled
// #define _TMR0  // Timer 0
#define _TMR1  // Timer 1 enabled
```

```
// #define _TMR2  // Timer 2
// #define _TMR3  // Timer 3
// *****************************************************************
// UART
// *****************************************************************
#define _UART  // UART => At least of the UARTs is enabled
#define _UART0  // UART 0 enabled
// #define _UART1  // UART 1
// #define _UART2  // UART 2
// *****************************************************************
// USB
// *****************************************************************
#define _USB  // USB enabled
#define _USBCLK SCU_USBCLK_EXT // Use external clock

void Device_Init(void);

#endif /* __new912_128_h */

/************* Copyright (C) 2005-2006 STMicroelectronics, Inc. ************/
```

Appendix D FlashLINK Cable – Install fast JTAG driver

Warning: Dual-Processor System or HyperThreading Enabled System: Please do NOT install the fast JTAG driver (JTD) for FlashLINK cable. The reason is that the JTD driver is not designed to handle code reentrancy as such it cannot support two processes at the same time. Refer to the workaround solution, described below.

D.1 Driver installation

To install the FlashLINK Cable fast JTAG driver (JTD),

1. Locate the “Drivers\FLink" folder under your CAPS folder.
2. At command prompt,
   a) To install the JTD driver
      `>>JTDINSTALL`
   b) To uninstall the JTD driver
      `>>JTDCFGW uninstall`
3. You must reboot the system after installing or uninstalling the driver.

Note: To achieve better programming performance in conjunction with the JTD driver, we recommend that you use the PCI parallel port card from SIIG: Cyber Parallel PCI Single Parallel Port

D.2 Workaround solutions

There are two workaround options for dual-processor systems, which are described below.

1. Boot into single processor mode and use the fast JTD driver.
2. Use the standard Parallel Port driver (PEP).

There are two workaround options for systems with hyper-thread enabled, which are described below.

1. Disable hyper-thread and use the fast JTD driver.
2. Leave hyper-thread as enabled and use the standard parallel port driver (PEP).

From JTAG/ISP standpoint, the tradeoff between the two solutions is the programming performance.

D.2.1 Dual-processor System: using solution 1

1. Boot your system in single processor mode and install JTD driver:
   If you have not already installed the JTD driver, go to the CAPS subfolder “Drivers\Flink” and invoke *JTDINSTALL.BAT*.
2. Reboot your system.

D.2.2 Dual-processor System: using solution 2

Follow the steps below to uninstall the JTD driver and install the PEP driver.

1. To uninstall the JTD, go to the command line prompt and type,

   `>> JTDCFGW uninstall`

   This command uninstalls the fast JTAG driver from your system.

   If you are unable to boot into Windows,

   a) press and hold F8 immediately during boot up.

   b) Select Safe Mode to skip loading the JTD driver.

      On systems that continue loading JTD driver, boot into Safe Mode with command prompt. Rename *JTD.sys* and *JTDMSG.dll* in the "WINDOWS\system32\drivers" folder to prevent loading the JTD driver upon bootup.

   c) Reboot into Normal mode and restore the above two file names before executing "`JTDCFGW uninstall`."

2. If you have not already installed PEP driver, go to CAPS subfolder "Drivers\Needhams" and invoke *INSTALL.BAT*. If *INSTALL.BAT* does not exist, then go to your prompt command line prompt and follow the steps below:

   `>> Run regpep`

   `>> Copy pepnt.sys %windir%\system32\drivers\*.*`

3. Go to the CAPS folder and update *uPSDsoft.INI* as follows:

   `[JTAG]`

   `Driver=PEP` (indicates the PEP driver is used. OD indicates the JTD driver is used.)

4. Reboot your system.

D.2.3 Hyper-thread enabled system: using solution 1

Use the following steps to disable hyper-thread and use the fast JTD driver.

1. During power-on – Press F2 to enter CMOS setup.
2. Move to CPU Information.
3. Choose hyper-thread and toggle to disabled.
4. Save the configuration and continue with the boot process.

Follow the steps below to install JTD driver:

1. If you have not already intalled JTD driver, go to CAPS subfolder "Drivers\Flink" and invoke *JTDINSTALL.BAT*.
2. Reboot your system.

D.2.4 Hyper-thread enabled system: using solution 2

Use the following steps to leave hyper-thread enabled and use the standard parallel port driver (PEP). (Uninstall the JTD driver and install the PEP driver).

1. To uninstall the JTD, enter the following at the command line prompt:

   `>> JTDCFGW uninstall`

   If you are unable to boot into Windows,

   a) press and hold F8 immediately during boot up.

   b) Select Safe Mode to skip loading the JTD driver.

      On systems that continue loading JTD driver, boot into Safe Mode with command prompt. Rename *JTD.sys* and *JTDMSG.dll* in the "WINDOWS\system32\drivers" folder to prevent loading the JTD driver upon bootup.

c) Reboot into Normal mode and restore the above two file names before executing "`JTDCFGW uninstall`."

2. If you have not already installed the PEP driver, go to the CAPS subfolder "Drivers\Needhams" and invoke *INSTALL.BAT*. If *INSTALL.BAT* does not exist, then enter the following commands at the command prompt.

   ```
   >> Run regpep
   >> Copy pepnt.sys %windir%\system32\drivers\*.*
   ```

3. Go to the CAPS folder and update *uPSDsoft.INI* as follows,

   ```
   [JTAG]
   Driver=PEP
   ```
   <<< this is to indicate PEP driver is used. (OD indicates JTD driver is used)

4. Reboot your system with hyper-thread enabled.

4 Revision history

Date	Revision	Changes
22-May-2006	1	Initial release.
9-Aug-2006	2	New note added, *Section 1.2 on page 4* Notes modified, *Section 1.2: Setting up the target hardware* New text added, *Section 3.7.3 on page 38*

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