PIN MUX UTILITY

User's Guide

V1.00

August 30, 2011

IMPORTANT NOTICE

Texas Instruments and its subsidiaries (TI) reserve the right to make changes to their products or to discontinue any product or service without notice, and advise customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgement, including those pertaining to warranty, patent infringement, and limitation of liability.

TI warrants performance of its semiconductor products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Customers are responsible for their applications using TI components.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards must be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance or customer product design. TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such products or services might be or are used. TI's publication of information regarding any third party's products or services does not constitute TI's approval, license, warranty or endorsement thereof.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations and notices. Representation or reproduction of this information with alteration voids all warranties provided for an associated TI product or service, is an unfair and deceptive business practice, and TI is not responsible or liable for any such use.

Resale of TI's products or services with *statements different from or beyond the parameters* stated by TI for that products or service voids all express and any implied warranties for the associated TI product or service, is an unfair and deceptive business practice, and TI is not responsible nor liable for any such use.

Also see: Standard Terms and Conditions of Sale for Semiconductor Products.
www.ti.com/sc/docs/stdterms.htm

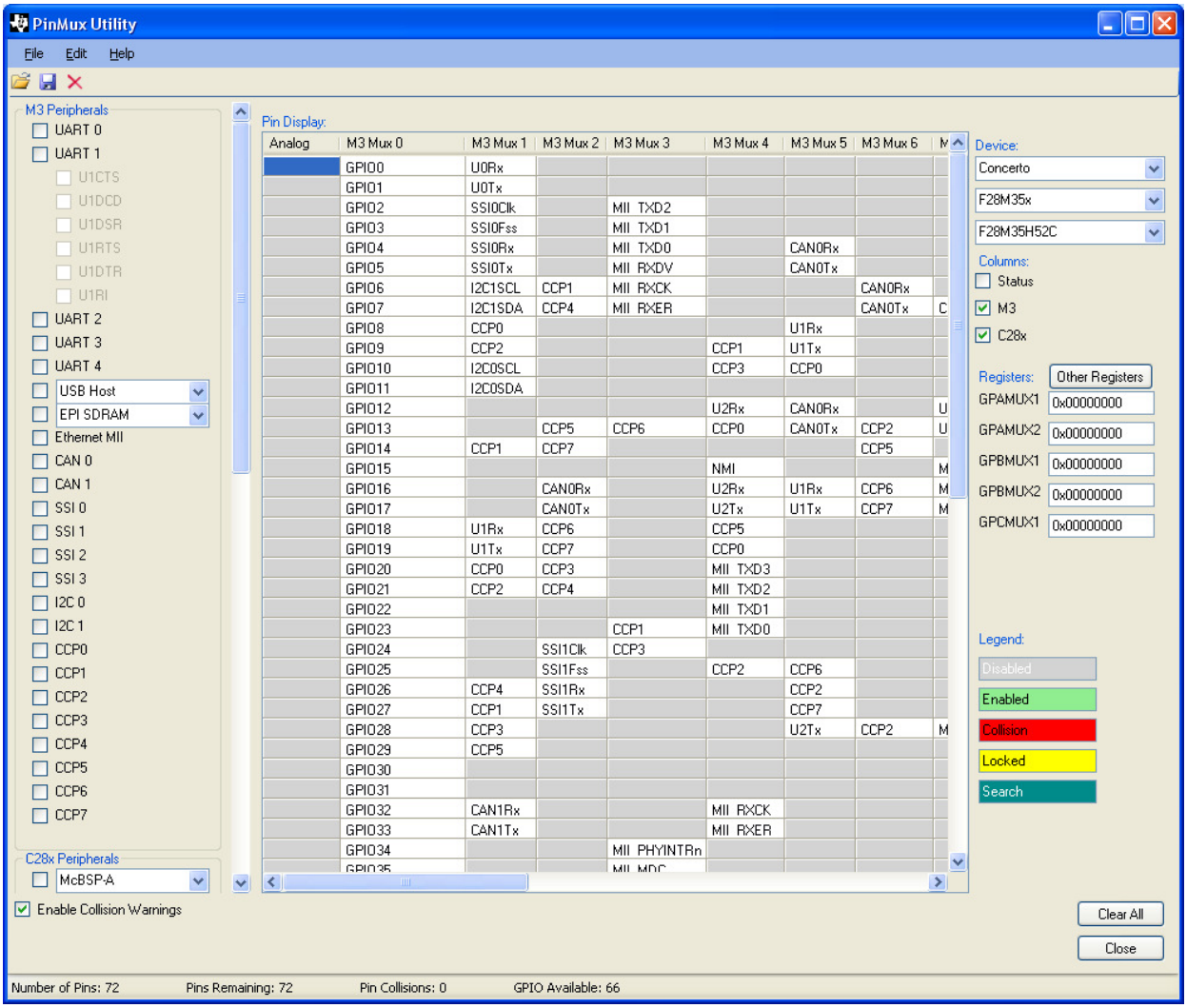
Mailing Address:
Texas Instruments
Post Office Box 655303
Dallas, Texas 75265

Table of content

1 Introduction 4
2 Device Selection 5
3 Pin Configuration 7
4 Collisions 9
5 Locking Pins 11
6 Search 12
7 Device Files 13
7.1 Directory configuration files 13
7.2 Mux configuration files 13
7.3 Peripherals configuration files 14
7.4 Registers Configuration Files 16
7.5 Cores Configuration Files 16
7.6 Adding an Analog Column 16

1 Introduction

The PinMux Utility is useful for experimenting and understanding the pin capabilities of various devices. The content – the various devices and the corresponding peripherals – are in the corresponding folder called devices. The primary window for the PinMux Utility is shown below as it will first appear when running. The screenshot picture below shows a Concerto device.

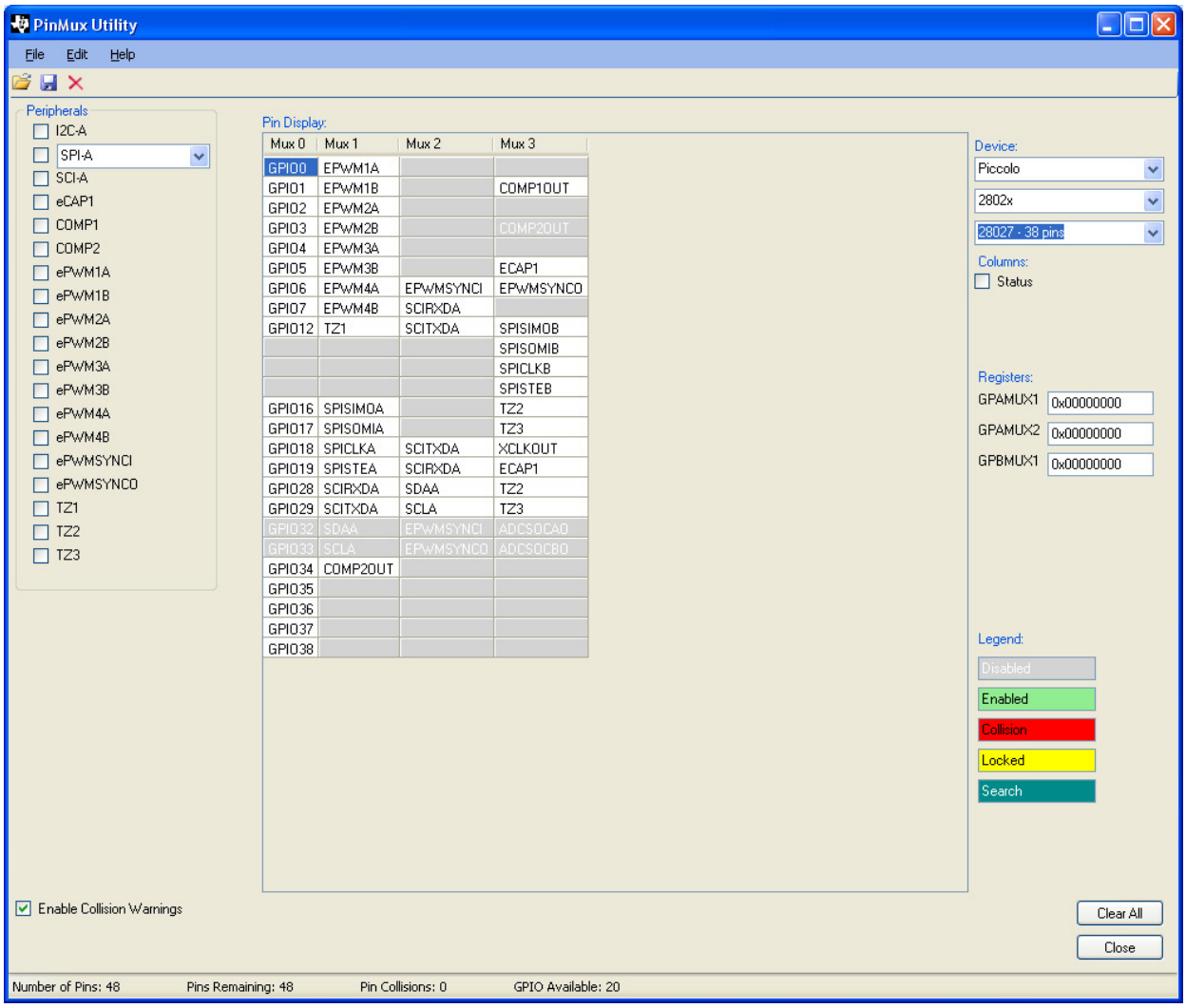


2 Device Selection

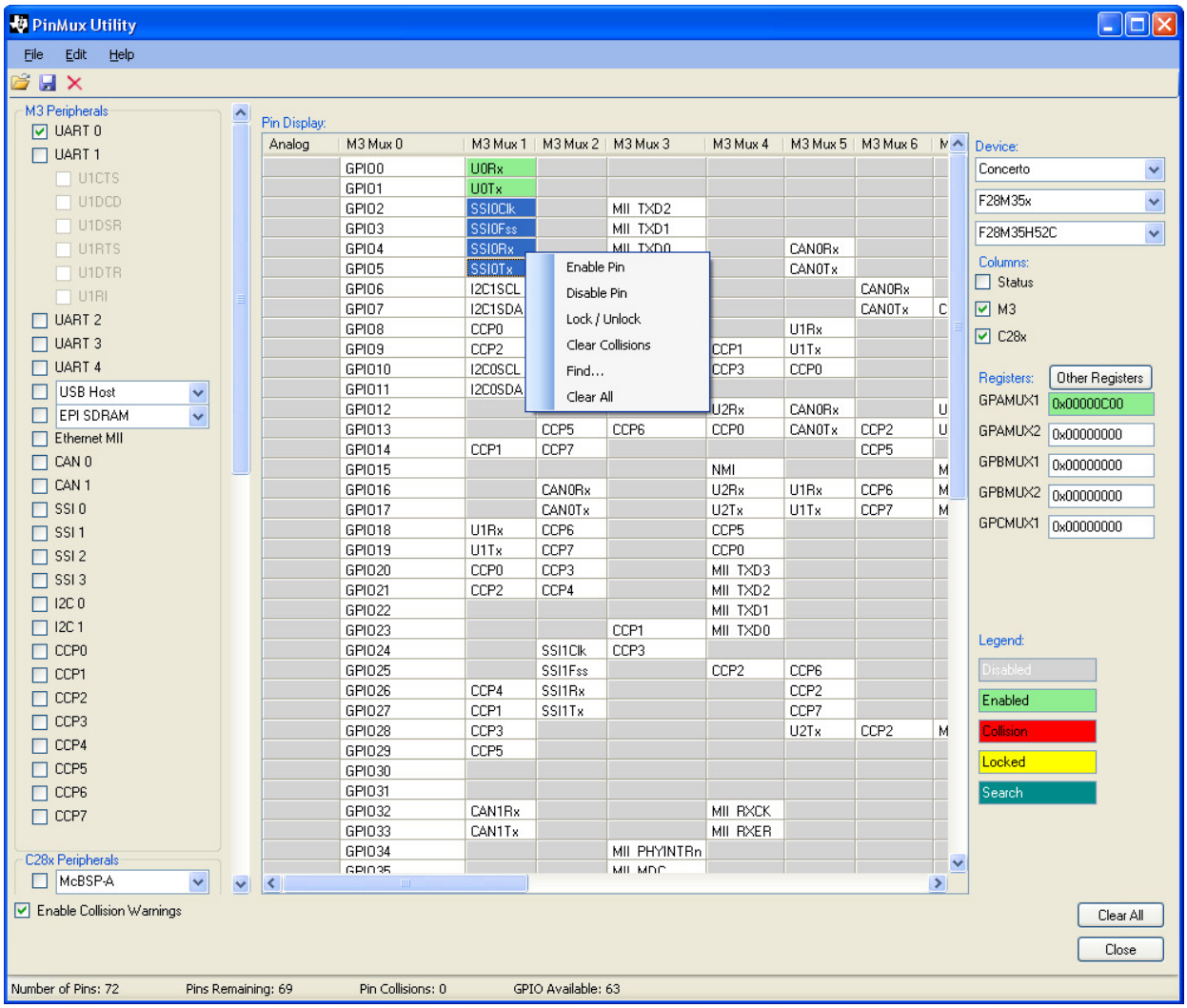
A specific device is selected by using the “Device:” combo box shown in the upper right corner of PinMux Utility window.

1. Select the device family, for example “Piccolo”
2. Select the sub-family, for example “2802x”.
3. Select the specific package, for example “28027 – 38 pins”.

The screenshot below shows the 28027 Piccolo device.



Alternative to selecting a device from the combo boxes, a previous pin configuration can be loaded from the menus at the top. A saved pin configuration remembers the device used and the enabled pins, but it does not save locked pins. The screenshot below shows a Concerto device configuration. Notice that there are more options on the right side.

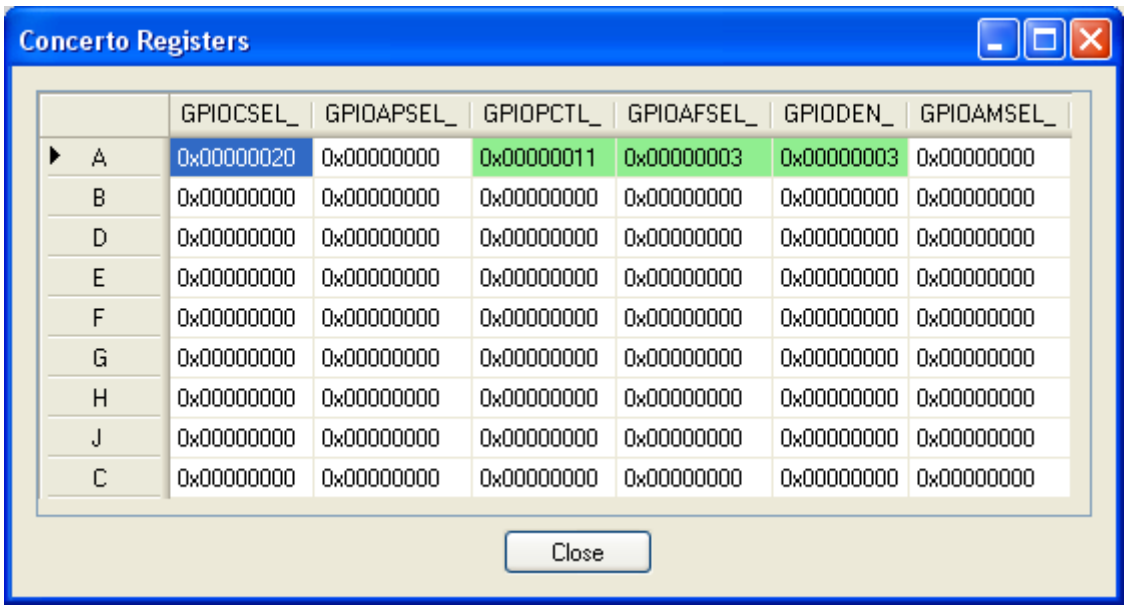

PinMux Utility
File
Edit
Help
M3 Peripherals
UART 0
UART 1
U1CTS
U1DCD
U1DSR
U1RTS
U1DTR
U1RI
UART 2
UART 3
UART 4
USB Host
EPI SDRAM
Ethernet MII
CAN 0
CAN 1
SSI 0
SSI 1
SSI 2
SSI 3
I2C 0
I2C 1
CCP0
CCP1
CCP2
CCP3
CCP4
CCP5
CCP6
CCP7
C28x Peripherals
McBSP-A
Enable Collision Warnings
Pin Display:
Analog
M3 Mux 0
M3 Mux 1
M3 Mux 2
M3 Mux 3
M3 Mux 4
M3 Mux 5
M3 Mux 6
GPIO0 U0Rx
GPIO1 U0Tx
GPIO2 SSI0Clk MII_TXD2
GPIO3 SSI0Fss MII_TXD1
GPIO4 SSI0Rx MII_TXD0 CAN0Rx
GPIO5 SSI0Tx CAN0Tx
GPIO6 I2C1SCL CAN0Rx
GPIO7 I2C1SDA CAN0Tx
GPIO8 CCP0 U1Rx
GPIO9 CCP2 CCP1 U1Tx
GPIO10 I2C0SCL CCP3 CCP0
GPIO11 I2C0SDA
GPIO12 U2Rx CAN0Rx
GPIO13 CCP5 CCP6 CCP0 CAN0Tx CCP2
GPIO14 CCP1 CCP7 CCP5
GPIO15 NMI
GPIO16 CAN0Rx U2Rx U1Rx CCP6
GPIO17 CAN0Tx U2Tx U1Tx CCP7
GPIO18 U1Rx CCP6 CCP5
GPIO19 U1Tx CCP7 CCP0
GPIO20 CCP0 CCP3 MII_TXD3
GPIO21 CCP2 CCP4 MII_TXD2
GPIO22 MII_TXD1
GPIO23 CCP1 MII_TXD0
GPIO24 SSI1Clk CCP3
GPIO25 SSI1Fss CCP2 CCP6
GPIO26 CCP4 SSI1Rx CCP2
GPIO27 CCP1 SSI1Tx CCP7
GPIO28 CCP3 U2Tx CCP2
GPIO29 CCP5
GPIO30
GPIO31
GPIO32 CAN1Rx MII_RXCK
GPIO33 CAN1Tx MII_RXER
GPIO34 MII_PHYINTRn
Enable Pin
Disable Pin
Lock / Unlock
Clear Collisions
Find...
Clear All
Device:
Concerto
F28M35x
F28M35H52C
Columns:
Status
M3
C28x
Registers:
Other Registers
GPAMUX1 0x00000C00
GPAMUX2 0x00000000
GPBMUX1 0x00000000
GPBMUX2 0x00000000
GPCMUX1 0x00000000
Legend:
Disabled
Enabled
Collision
Locked
Search
Clear All
Close
Number of Pins: 72
Pins Remaining: 69
Pin Collisions: 0
GPIO Available: 63

3 Pin Configuration

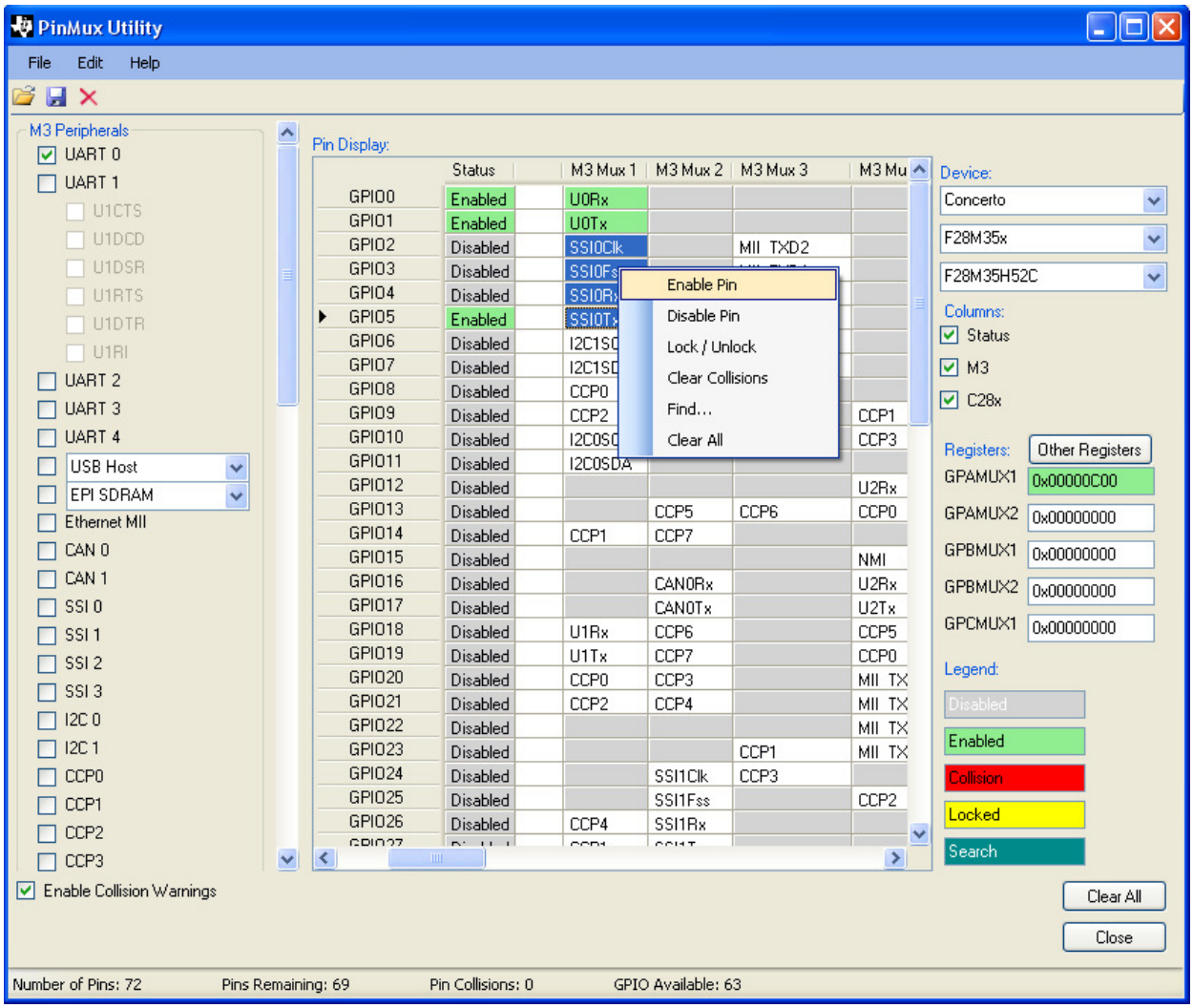
There are a few ways to select pins:

- Choose a peripheral from the menu on the left.
- Select the pinmux fields individually, then right-click to enable, disable, or lock.
- Double-click a pinmux field to enable or disable it.
- Modify the registers on the right to the desired value. (Does not apply to the ARM cortex CPU registers on Concerto)

Changes made to the peripherals, pin display, or registers will be reflected in the other two. For example, if the UART 0 is selected on the left, the U0Rx and U0Tx cells will turn green. If the four SSI cells are selected and enabled with the right-click context menu, the SSI0 peripheral would become selected on the left. Modifying the register on the right affects only the selection of the C28x pinmux functionality.
There are a number of registers associated only with the Concerto device's M3 CPU, which can be viewed by pressing the "Other Registers" button next to the "Registers" label.

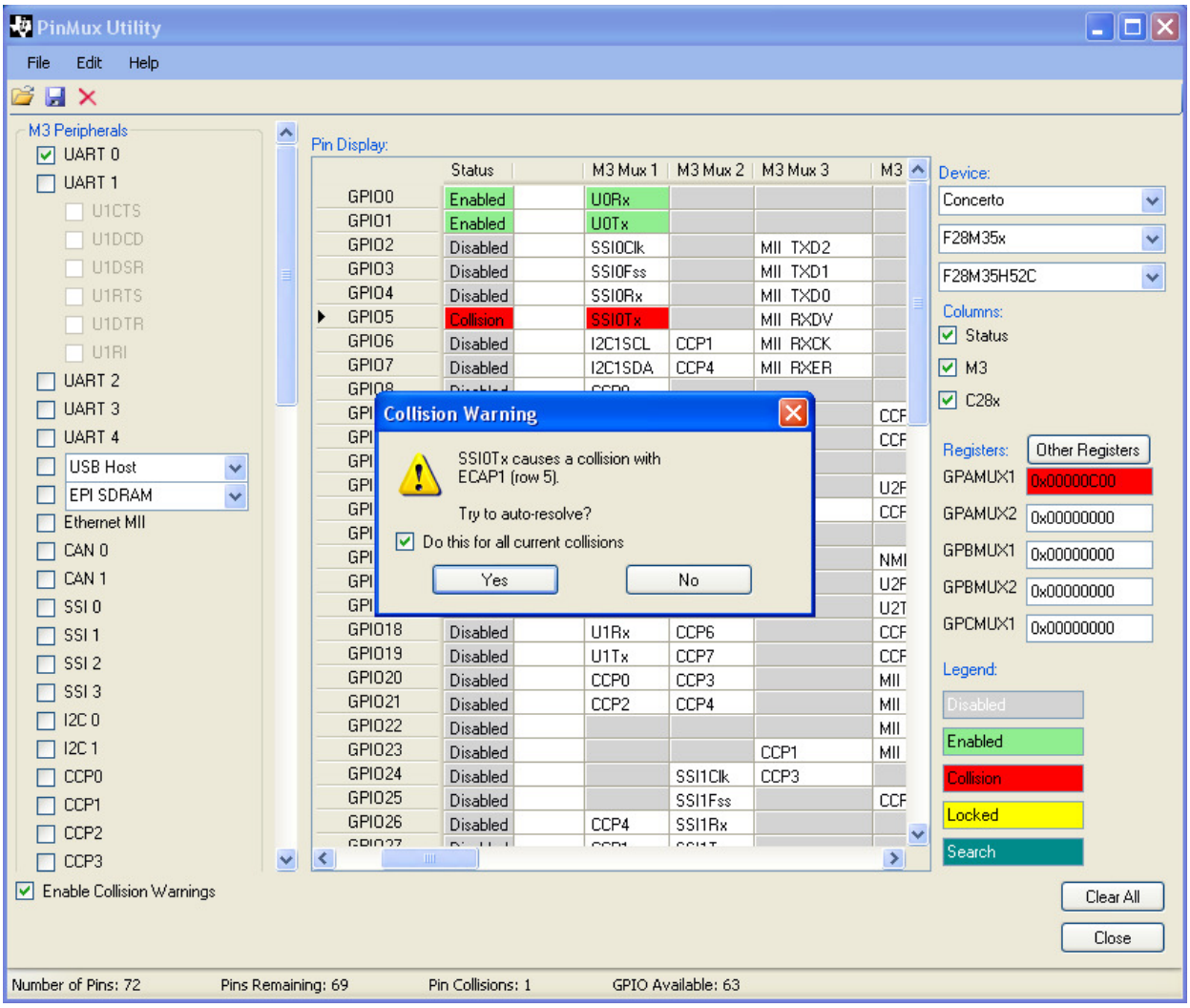


The checkbox(es) found between the device selection and registers display allow columns to be turned on and off. By default, the status column (showing GPIO pin number and pin state) are turned off, but these can be very useful when viewing a larger pinmux display, such as that found on Concerto. Doing so here shows that the GPIO5 pin has been enabled, but the enabled cell is out of view (see the picture below). When someone tries to enable the four SSI0 pins visible, a collision will occur.

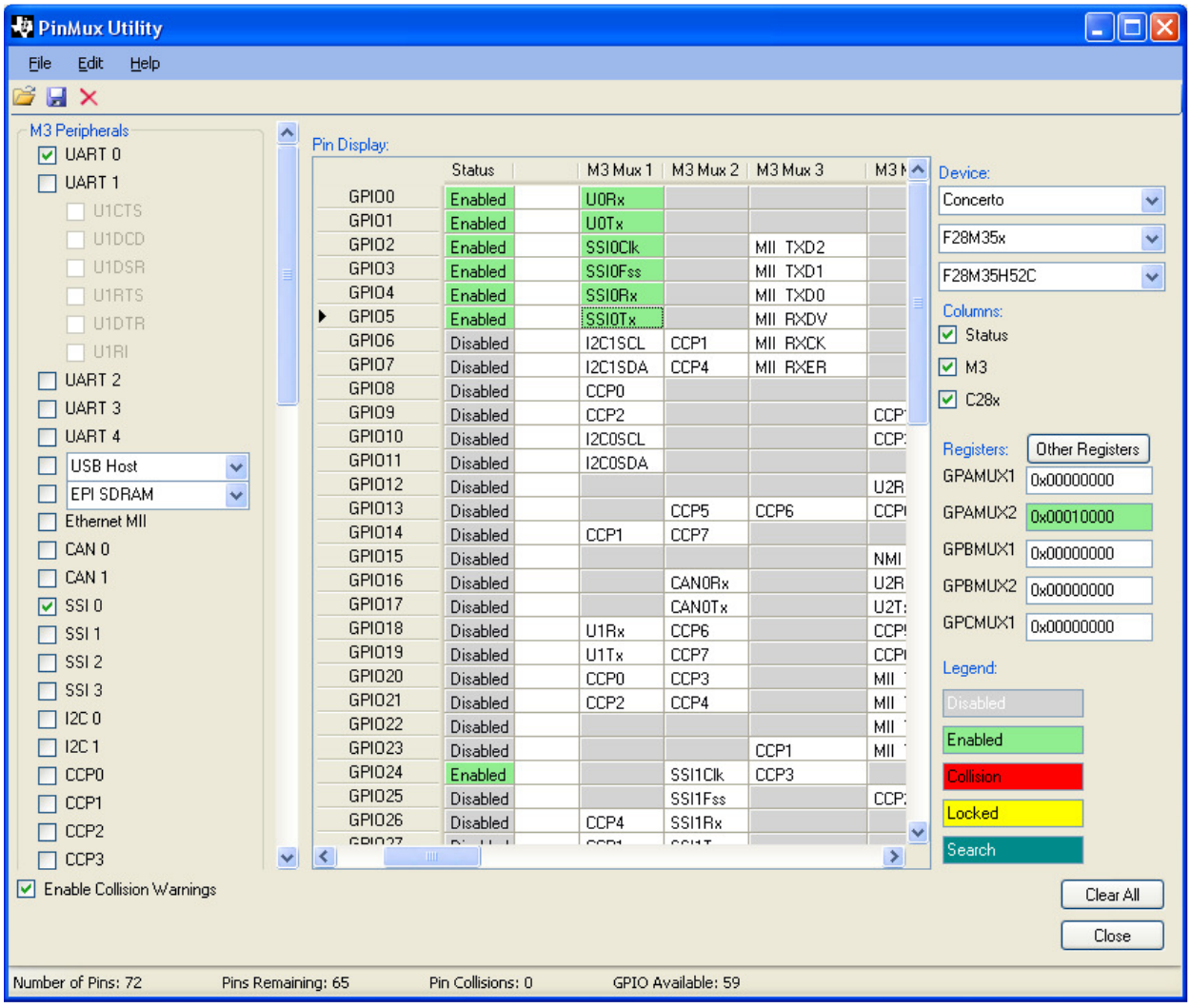


4 Collisions

If the Enable Collision Warnings flag below the Peripherals list is enabled, a warning will appear when a collision occurs, asking for permission to auto-resolve the collision. When a collision is automatically resolved, the program will try to find a way to rearrange the selected pinmux cells such that the desired functionality is selected without a collision. The attempts to move functionality will often cause additional collisions, and keeping the “Do this for all current collisions” checkbox checked will skip additional warnings for the current operation as shown in the figure below.

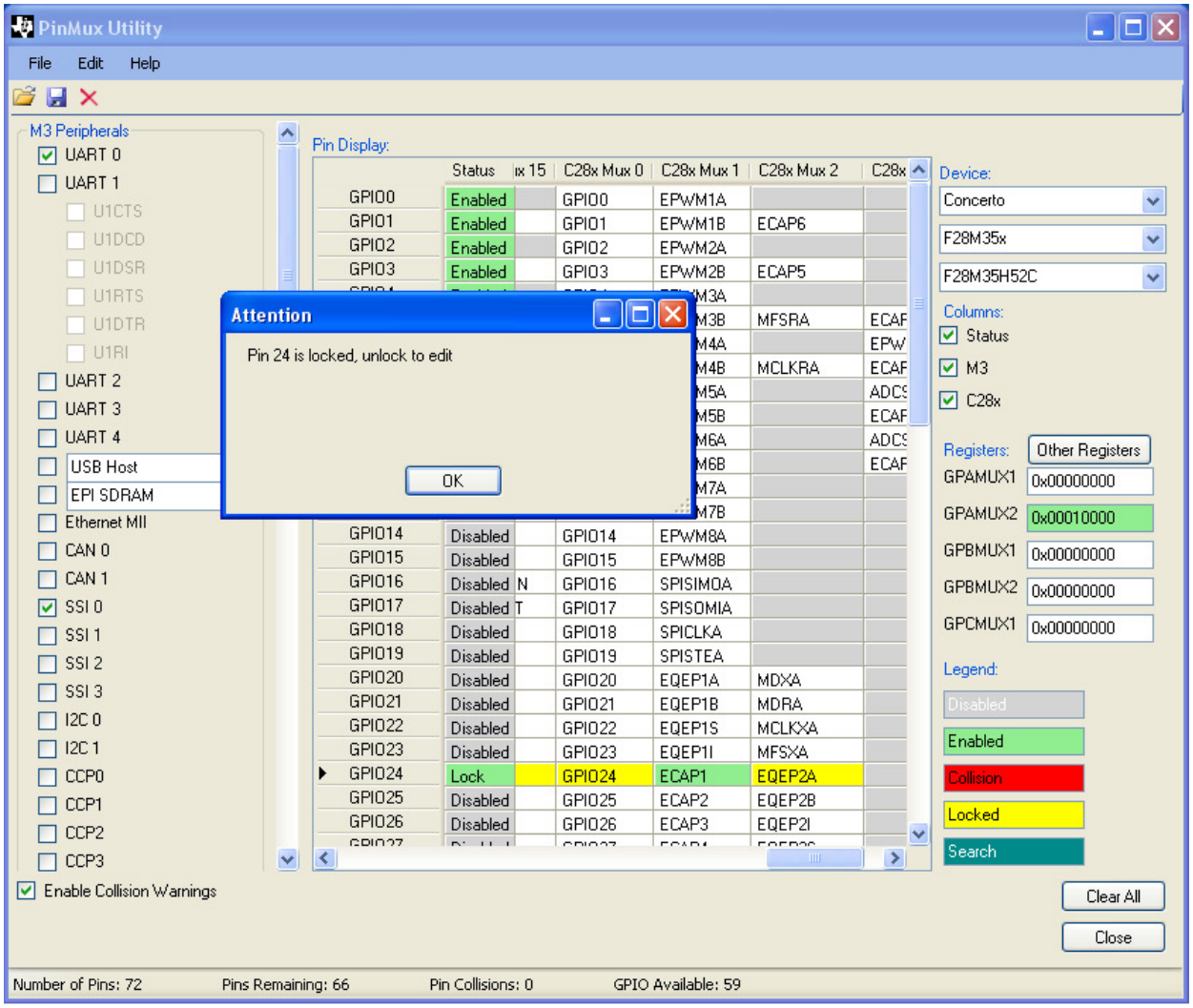


When the Collision has been resolved, the enabled C28x pin has moved from row 5 to row 24, and the changes are visible in the Registers and Peripherals list. Note the changes in pin counts found in the status bar at the bottom of the window. This shows the pins remaining, a count of available GPIO pins, and keeping count of collisions



5 Locking Pins

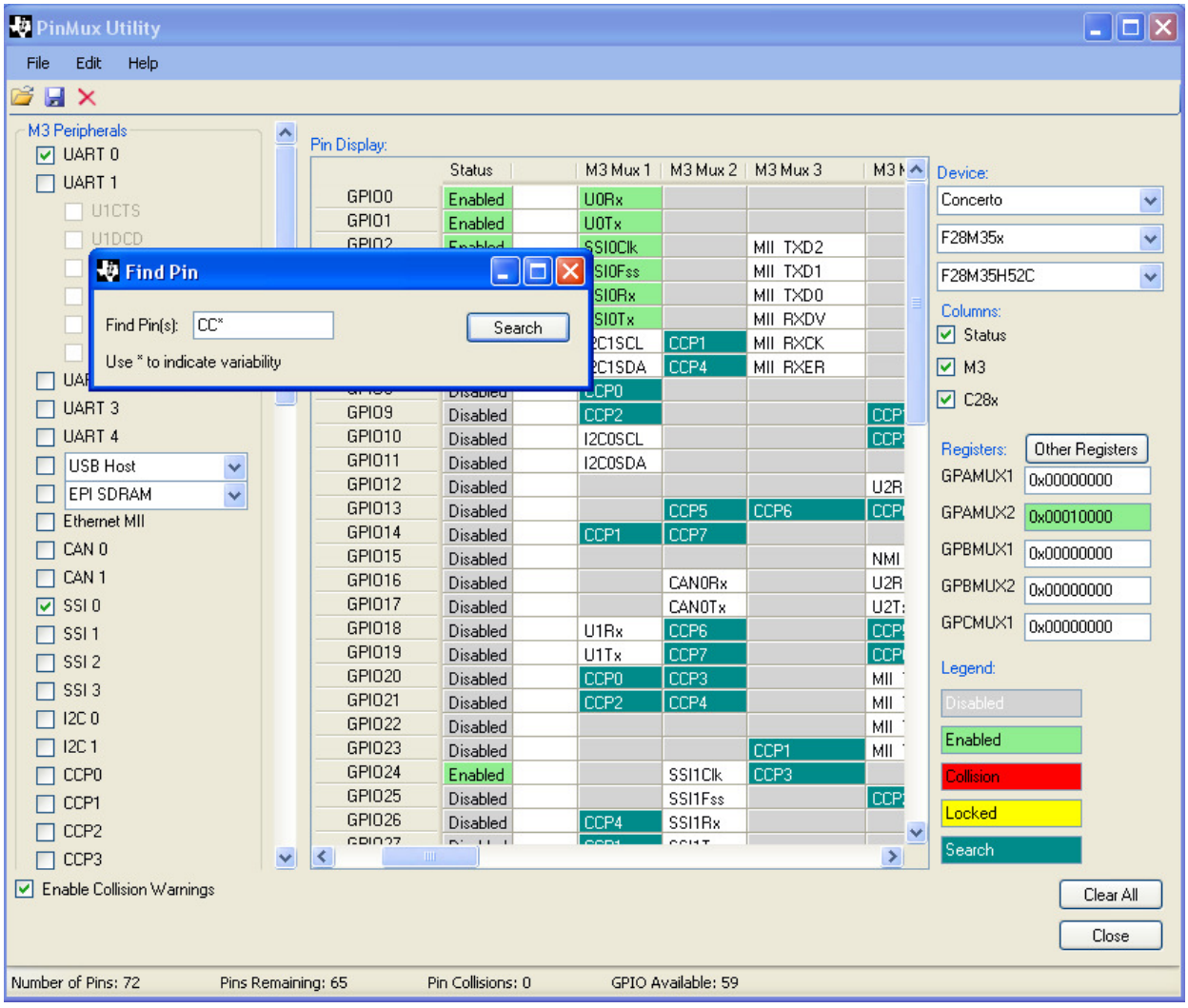
Enabled, non-colliding pins , can be selected and locked via the right-click context menu. This prevents the locked pins from moving around during collision resolution. A locked row will be highlighted in yellow and the status changed to Lock. If an attempt to change that pin is made, a message will appear stating that the pin is locked. To modify a pin, first unlock it.



A configuration can be saved using the menus at the top, or exported as a header file with defined register values. If the file has not been saved, then changing devices, clearing the program state, or closing the program will trigger a save prompt.

6 Search

To find pins, press Ctrl + F or select Find from the Edit menu. Searching is an easy way to find alternate locations for the same functionality. An asterisk can be used to indicate multi-character variability, such as in the search for CCP pins shown below:



7 Device Files

The PinMux Utility makes use of several configuration files to store device data. These files make changing the content of the program without needing to know C#, Windows Forms, or have the project and code. The folder structure for these files is as follows:

- PinSetup.exe
- devices
    - devicefamilies.txt
    - [devicefamily]
        - devicelines.txt
        - [deviceline]
            - devices.txt
            - [device]
                - mux.csv
                - peripherals.txt
                - registers.txt
                - cores.txt (optional)

7.1 Directory configuration files

The files devicefamilies.txt, devicelines.txt, and devices.txt will each be simple text files featuring the names for the folders at the same level, one line per name. The names will appear in the program as they do in the text file, but the corresponding folder needs to be a lowercase version of the name.

7.2 Mux configuration files

The mux.csv configuration file corresponds to the pin display. As a comma separated file, commas should not be used in the fields, and forward slashes (/) should be used to indicate two functionalities for the same cell. Cells that do not have functionality should say “reserved” to prevent being unlisted when saved, but it will appear as empty in the program. To disable a cell but still have text visible, end the value with the letter D in parentheses. Ex: “GPIO66(d)” would display GPIO66 and be grayed out.
Any columns or rows that are completely blank in the configuration file are hidden in the program’s pin display.

7.3 Peripherals configuration files

Peripherals files start with gpioextint or an integer value, but currently this line is not associated with any functionality. The peripherals are listed, each beginning with the word variable, static, or combo, with an optional core index afterward, separated by a comma (value of 0 or 1. -1 is used in a special case; see Adding an Analog Column below).

Static peripherals begin with the name of the peripheral, followed by the pins it requires. If there is only a name listed, the name will also be used as the only pin.
Ex:

```
static          // the peripheral named CAN-A has two pins: CANRXA and CANTXA
CAN-A
CANRXA
CANTXA

static          // This peripheral has one pin, and has the same name
HRCAP3

static          // Same as the previous peripheral definition, without the shortcut.
HRCAP3
HRCAP3
```

Variable peripherals have a number of options. Each option name is followed on the same line by a comma and the number of pins that are specific to that option. Then, the number of pins are listed for that option. After the last option, additional lines for the peripheral will be interpreted as base pins needed by all options:

Ex:

```
variable,0              // The 0 indicates this peripheral belongs to the first core
SPI-B,2                 // First option has 2 additional pins
SPISIMOB
SPISOMIB
SPI-B (3 pin master),1  // Second option has only one additional pin
SPISIMOB
SPI-B (3 pin slave),1   // Last option has only one additional pin
SPISOMIB
SPISTEB                 // These two “base” pins are activated with each option.
SPICLKB

variable,1              // The 1 indicates this peripheral belongs to the second core
McBSP-A,2               // The first option has two additional pins
MFSRA
MCLKRA
McBSP (SPI),0           // The 0 here indicates there are no additional pins for this option
MDXA                    // The four remaining pins are base pins.
MDRA
MCLKXA
```

```
MFSXA

variable,1              // Equivalent to previous definition, but each option is fully spelled out
instead
McBSP-A,6               // of using base pins.
MFSRA
MCLKRA
MDXA
MDRA
MCLKXA
MFSXA
McBSP (SPI),4
MDXA
MDRA
MCLKXA
MFSXA
```

Combo peripherals allow a number of options to be selected or unselected in an organized way. After the combo line, the main peripheral name is followed by a comma and the number of options. Options are then listed with the name of the option, followed by a 1 or 0 for checked or unchecked, and a number indicating the number of pins. That line is followed by that number of pins, listed one per line. After the options are base pins that are enabled from the main peripheral checkbox.

Ex:

```
combo
eQEP1,2         // Peripheral is named eQEP1 and has two additional options
Strobe,1,1      // First option is selected by default and controls one pin
EQEP1S
Index,1,1       // Second option is similar to the first with a different pin
EQEP1I
EQEP1A          // Two pins are controlled by the eQEP1 checkbox alone.
EQEP1B

combo,1         // Similar to previous, except activated on the second core
eQEP1,2         // and options are unselected by default.
Strobe,0,1
EQEP1S
Index,0,1
EQEP1I
EQEP1A
EQEP1B
```

7.4 Registers Configuration Files

The registers.txt files begin with a number indicating the bits per register. Subsequent lines will be used as replacement labels for the register fields and used in the definitions for header files. Concerto M3 registers are handled differently.

Ex:
32
GPAMUX1
GPAMUX2
GPBMUX1
GPBMUX2
GPCMUX1

7.5 Cores Configuration Files

This file is not required for single core devices, but can still be useful. However, this file must be included to define two cores. Each core listed will have a name, followed by a type. The type determines whether the additional considerations for concerto devices will be used or the default setup. The Concerto type will have both normal and alternate ranges that can be used in place of the later columns of the core.

Ex:

```
M3              // name
concerto        // type (can be concerto or simple)
0               // start
15              // end
16              // alternate start (concerto only)
19              // alternate end   (concerto only)
C28x
simple
20
23
```

7.6 Adding an Analog Column

To add an analog column to the front of the first core, have the first column of mux.csv correspond to the analog functionality and have the column stand outside of the range of the first core (start the range at 1). To have a peripheral find a cell in the analog column, set the core index to -1 instead of 0, 1, or leaving it blank.