



Driver Tools Manual

Version 0.4

Table of Contents

1.	Introduction	1
2.	New Device	1
3.	Peripheral Driver Configuration.....	2
3.1	Port I/O Setting.....	2
3.1.1	Layout Window.....	3
3.1.2	Function Setting	5
3.1.3	Pins Attributes Setting.....	5
3.1.4	Generate Port I/O Source Code	6
4.	Save Driver Source Code	6
	Release History.....	8

Confidential

1. Introduction

Quintic Driver Tools is a visual Windows tool to ease writing peripheral driver codes for developer based on Quintic QN9020 platform. The main window is shown in Figure 1. The main window contains the text representing the current configuration file and this file is blank initially.

The operation of Driver Tools is easy. A detailed introduction of peripheral driver configuration is included as following.

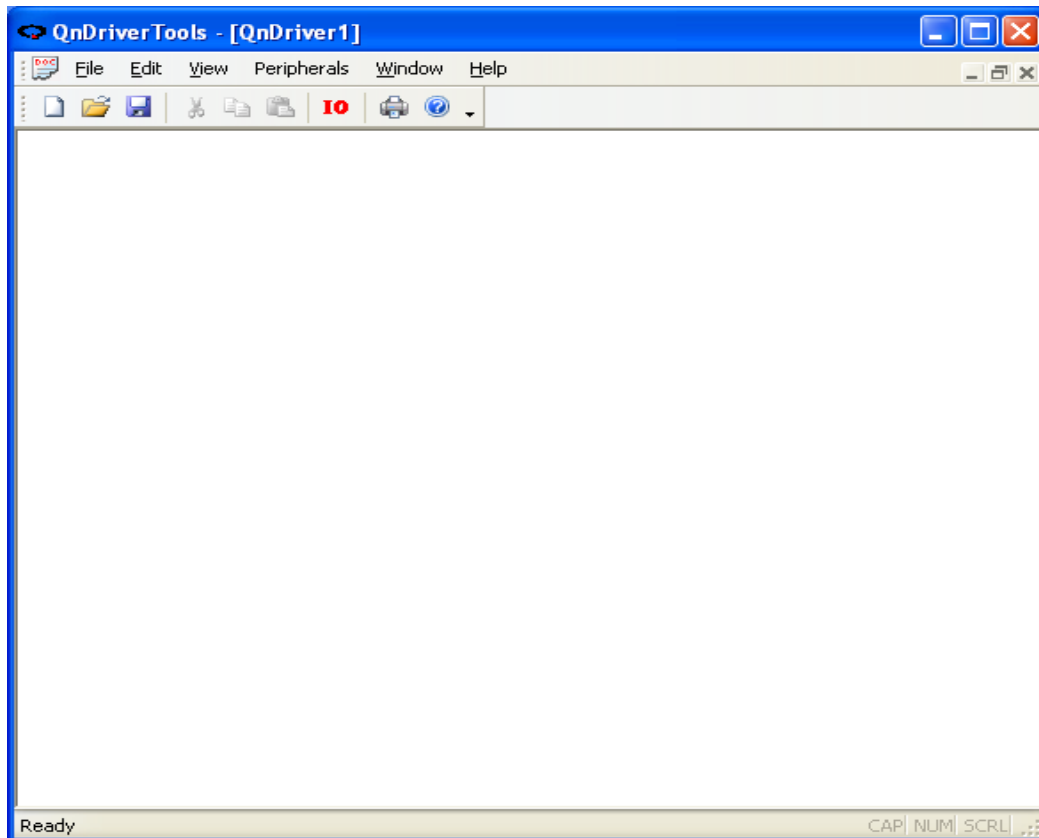



Figure-1 Main Window

2. New Device

The tool support QN9020 and QN9021 driver configuration. Click “File->New” on menu or the  button on tool bar that can call the selecting device window. And the window is shown as Figure 2.

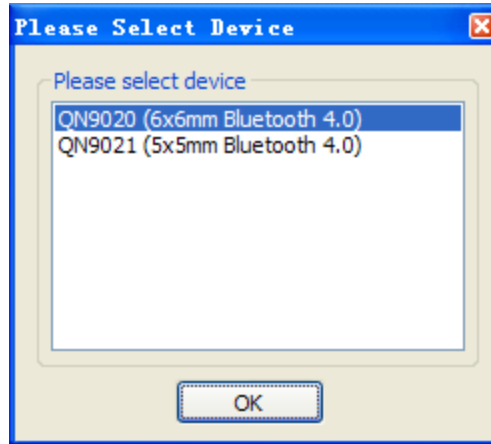


Figure-2 Select Device

3. Peripheral Driver Configuration

The peripherals menu contains all peripherals driver setting. At the moment, there is one item on it, which is Port I/O configuration.

3.1 Port I/O Setting

Click “Peripherals->Port I/O” on menu or the **IO** button on tool bar that can call the Port I/O setting window. And the window is shown as Figure 3. Most of the area of the window is built up by grids. Port I/O setting mainly contains the function setting and the attributes setting of each port.

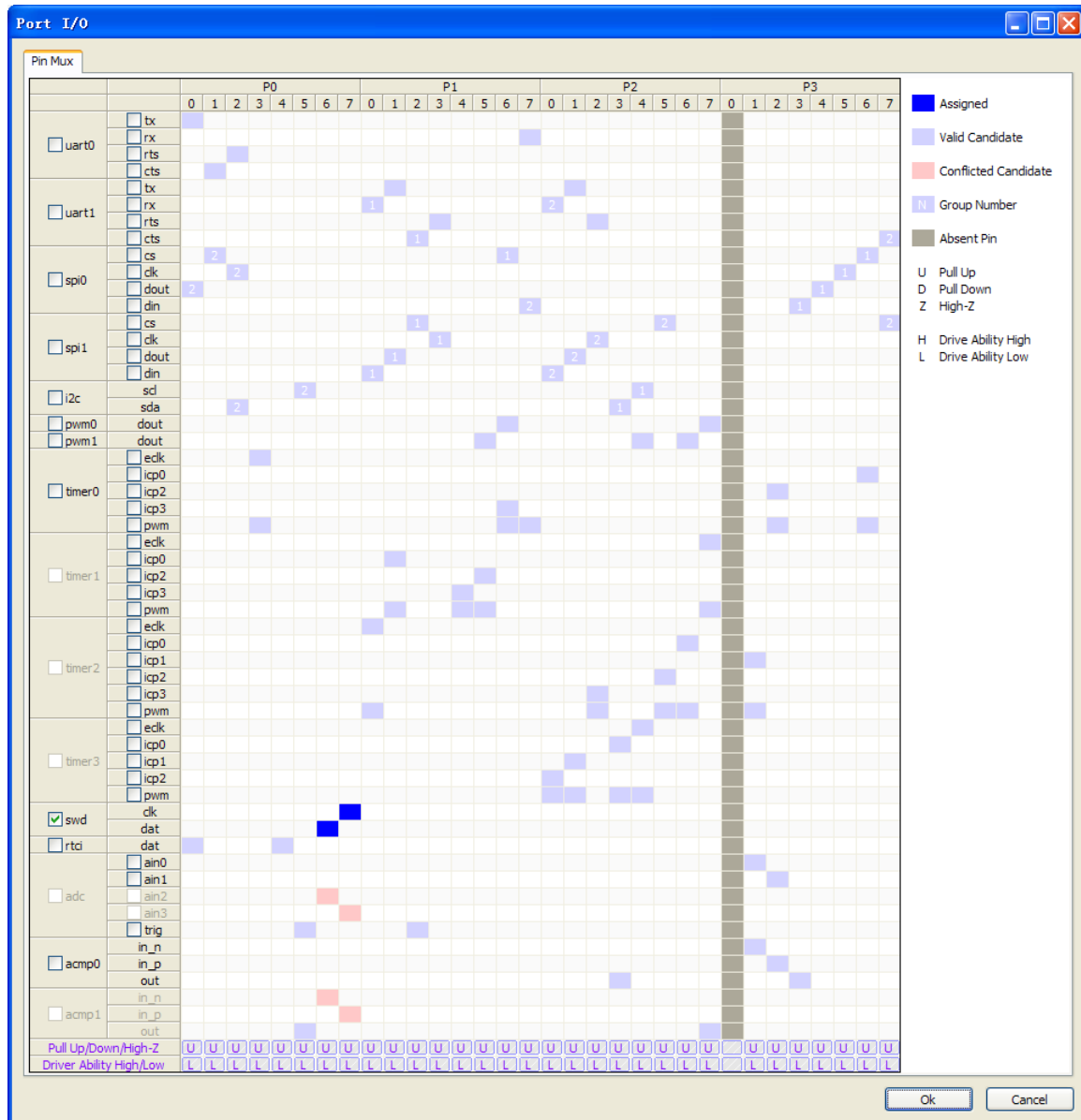


Figure-3 Port I/O

Please see a brief introduction of the window layout and a detailed description of how to set the function and attributes of each port as indicated below.

3.1.1 Layout Window

This window can be divided into five parts: graphical interpretation, status grids, external pins, functions, pins attributes.

3.1.1.1 Area of Graphical Interpretation and Status Grids

The most of grids are status grids which represent the relation between the interface function and external pins. It looks similar to Figure 4. The meaning of grid color is shown as Figure 5 which is the graphical interpretation on the right side of the window.

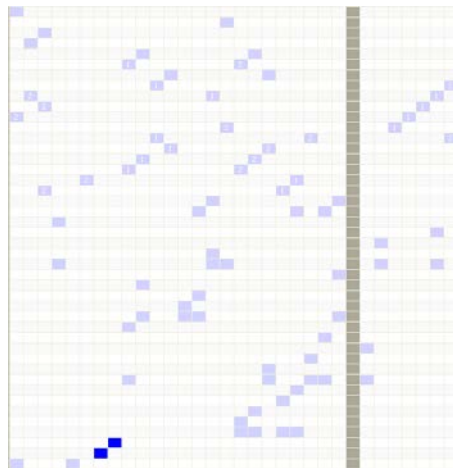


Figure-4 Status Grids

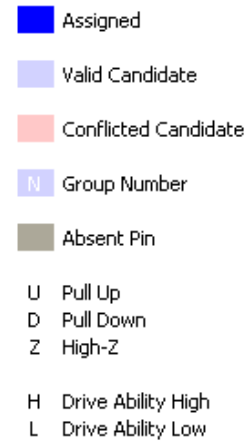


Figure-5 Graphical Interpretation

The candidate pins of each interface function are shown all the time. And those pins have four kinds of status.

- **Assigned:** This pin has been assigned to current interface function.
- **Valid Candidate:** This pin is not assigned, and it can be assigned to current interface function in future.
- **Conflicted Candidate:** This pin is not assigned, and it cannot be assigned to current function, because other interface function has occupied this pin.
- **Absent:** This pin does not exist.

These statuses can help developers to manage the assignment of functions and pins easily.

3.1.1.2 Area of External Pins

The top side grids represent external pins of the chip, such as P0.1, P1.2, etc. It looks similar to Figure 6:

P0							
0	1	2	3	4	5	6	7

Figure-6 External Pins

3.1.1.3 Area of Functions

The left side grids represent the interface function of the chip, such as UART, SPI, or I2C etc. It is shown in Figure 7.

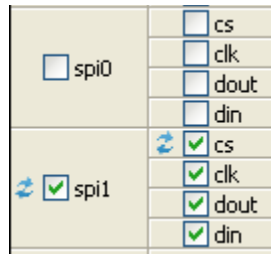


Figure-7 Interface Function

3.1.1.4 Area of Pins Attributes

The bottom side grids represent the pin attribute. It is shown as Figure 8.

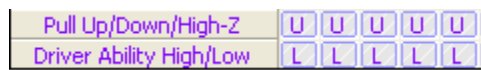


Figure-8 Pins Attributes

3.1.2 Function Setting

In Figure 7, this area can set the functions of each pin.

Once select an interface function, the valid pins will be shown as “assigned” status. If other pins are also valid, the icon will be shown, and clicking the icon can switch the valid pin circularly. The icon may appear in two kinds of positions. Such as SPI1, one is in the front of “spi1”, the other is in the front of “cs”. There are multiple groups of valid pins for spi1, so clicking the icon which is in the front of “spi1” can switch the group of pins, and clicking the icon which is in the front of “cs” can switch the pin of spi1-cs in one group.

Sometimes, may want to use spi1 to control two devices, so two spi1-cs pins, but one clk, dout and din pin is necessary. How to achieve it? Press the “CTRL” key, and click left mouse button on the grid (in the figure 4) whose status is “Valid Candidate” in the same line of spi1-cs. This process is shown in Figures 9.

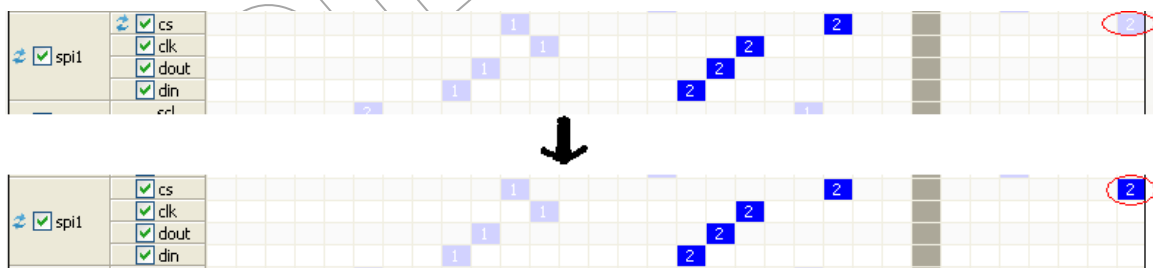


Figure-9 Multiple Select

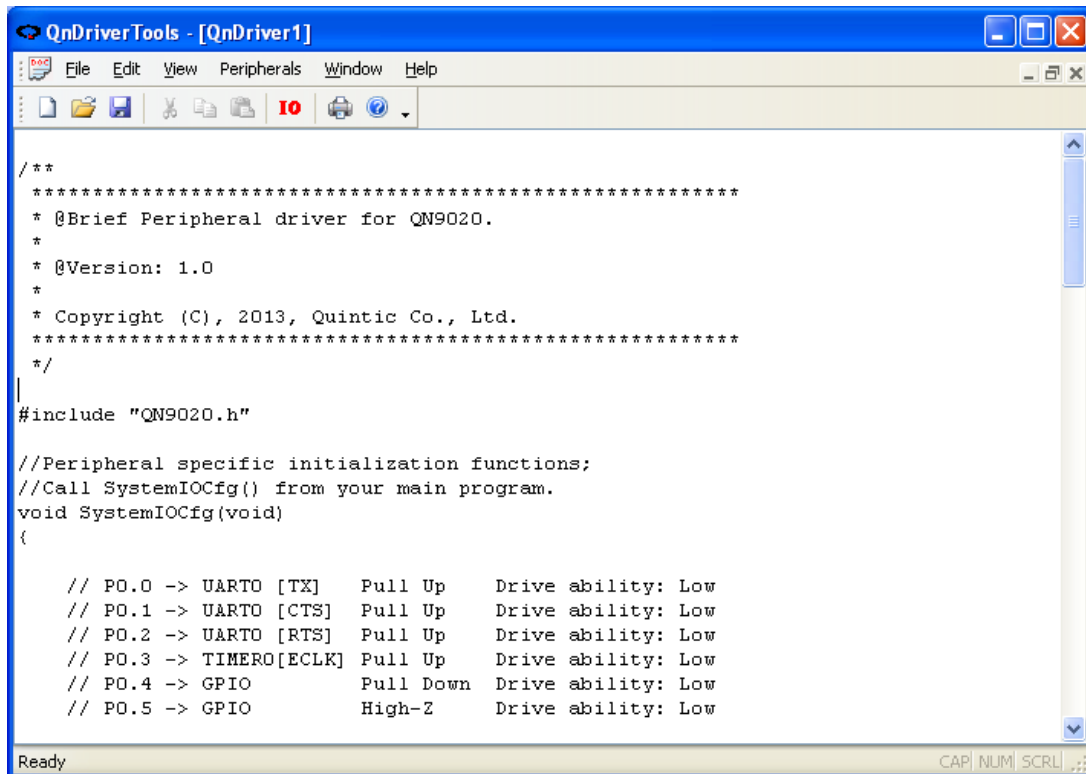
3.1.3 Pins Attributes Setting

In Figure 8, this area can set the pins attributes.

The first line of Figure 8 can set the pull way of each pin: pull up, pull down or High-Z. Second line can set the driver ability: High or Low. Click these buttons can change the pin attribute circularly.

3.1.4 Generate Port I/O Source Code

After setting the needed functions and attributes of each pin, click on the OK button, the source code is automatically generated in the main window. Functions and attributes of each pin are listed as a comment. It is shown as Figure 10.



```

QnDriverTools - [QnDriver1]
File Edit View Peripherals Window Help
I/O

/**
*****
 * @Brief Peripheral driver for QN9020.
 *
 * @Version: 1.0
 *
 * Copyright (C), 2013, Quintic Co., Ltd.
*****
 */
#include "QN9020.h"

//Peripheral specific initialization functions;
//Call SystemIOCfg() from your main program.
void SystemIOCfg(void)
{
    // PO.0 -> UART0 [TX]   Pull Up   Drive ability: Low
    // PO.1 -> UART0 [CTS]  Pull Up   Drive ability: Low
    // PO.2 -> UART0 [RTS]  Pull Up   Drive ability: Low
    // PO.3 -> TIMERO[ECLK] Pull Up   Drive ability: Low
    // PO.4 -> GPIO         Pull Down Drive ability: Low
    // PO.5 -> GPIO         High-Z    Drive ability: Low
}
    
```

Figure-10 Generate Source Code

4. Save Driver Source Code

After all of the peripherals have been configured, go to "File->Save" or "File->Save As" to save the current driver source code into a file. It is shown in Figure 11.

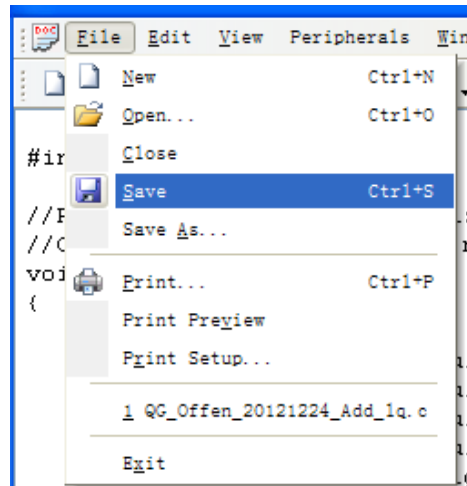


Figure-11 Save Source Code

The Save Source dialog will come up, and from here simply choose where you want to save your file, and name it accordingly. Here is an example of the Save Source Code dialog as Figure 12:

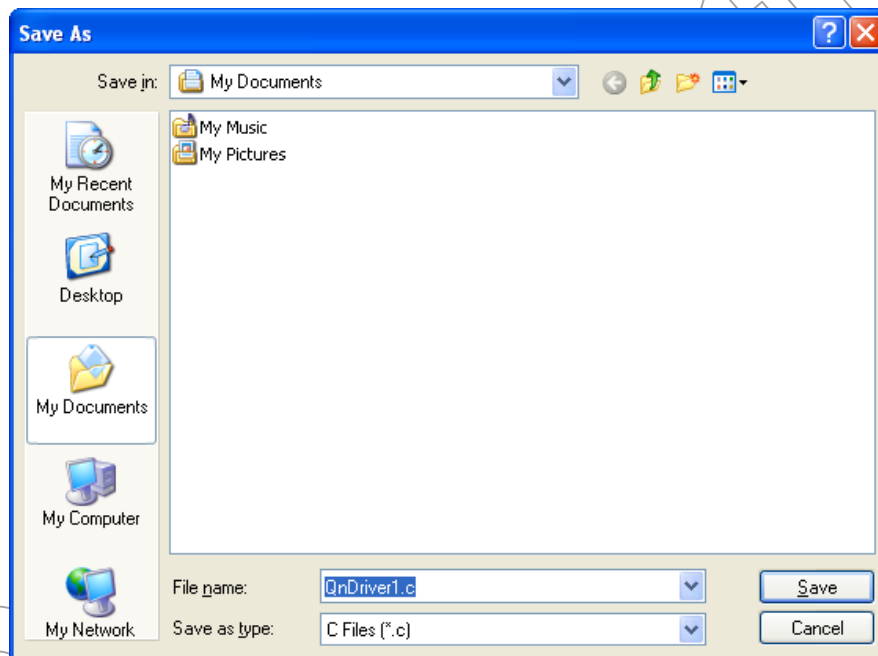


Figure-12 Save Dialog

When save this source file, it can be used by including it and calling the "SystemIOCfg ()" function from the main program.

Release History

REVISION	CHANGE DESCRIPTION	DATE
0.1	Initial	2013-2-18
0.2	Update main window Figure-2	2013-2-22
0.3	Update Port I/O window Figure-2	2013-5-7
0.4	Add New Device chapter	2014-2-11

Confidential