Simplifying Crossbar and Peripheral Pin-Out Configuration for 32-Bit ARM-Based Microcontrollers

The process of evaluating 32-bit microcontroller options for a new embedded design can be tedious and time-consuming. During the evaluation process, embedded developers often must learn how to configure and use each new MCU's general-purpose input/output (GPIO) pins and peripherals. As 32-bit MCUs become more complex, the degree of configurability for each GPIO pin and peripheral increases significantly. A highly configurable GPIO and peripheral set should give developers additional design flexibility instead of adding design hurdles.

Having access to an easy-to-use configuration interface paired with software tools that graphically configure a device's pin-out and peripherals enables embedded developers to quickly determine an application's pin-out options and generate initialization code. Let's take a closer look at how a microcontroller configuration utility with an intelligent graphical user interface (GUI) can ease the application development process and ultimately accelerate time-to-market.

Once an application's design specification is complete, the developer can assess potential MCUs to use in the design. Given that ARM has become a de facto standard for 32-bit MCUs, many new MCU options now incorporate ARM Cortex CPUs. Although an MCU may meet the developer's specifications in terms of code space, RAM, peripherals and analog functionality, the microcontroller must also be capable of physically pinning out all needed functions. If every peripheral or function was available on every port pin without any limitations, the developer would need to review only the specifications. In most cases, however, the MCU's peripherals are multiplexed on various port pins, and it is the developer's responsibility to determine if the microcontroller can be configured in a way that meets the design requirements.

Ideally, developers will select a microcontroller with highly configurable port pins, enabling them to choose smaller, less expensive devices for their application needs. Additional configuration flexibility also makes it easy for developers to make last-minute changes to a design. One of the latest additions to the ARM Cortex-based MCU market – the Precision32™ MCU family from Silicon Laboratories – offers an easy-to-configure dual-crossbar architecture that provides developers with this pin-out flexibility. The dual-crossbar architecture allows the developer to assign a mix of digital and analog functions to GPIO pins.

As developers begin to evaluate potential 32-bit candidates for their embedded applications, they must learn how to configure and use various microcontroller products and associated tool chains. This is often a time-consuming task. Providing developers with an easy-to-use graphical configuration tool that allows them to determine a device's available pin-outs and quickly generate initialization code will significantly reduce the total development time. For example, to help embedded designers accelerate development with Precision32 MCUs, Silicon Labs offers a GUI-based AppBuilder software tool that enables developers to easily configure port pins and peripherals while providing a snapshot of where the pin functionality will appear on the MCU package and which peripherals are configured for use. The following figure shows a sample printout generated by the AppBuilder tool.

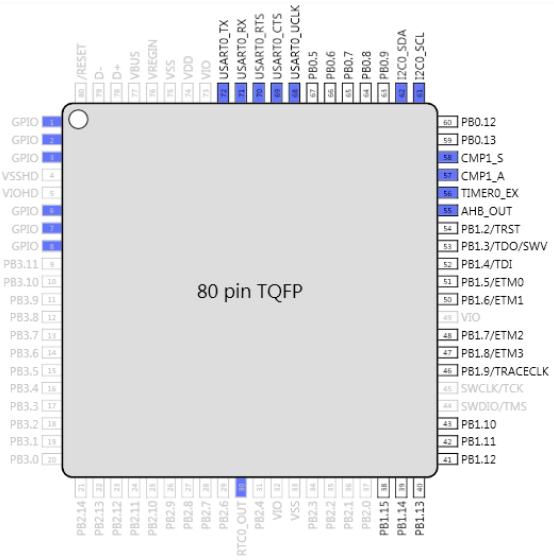

Figure 1. Example of Peripheral Pin Outs with AppBuilder Tool

Using the AppBuilder tool, developers can easily change pin configurations and immediately see how these changes affect other pin functions in real time. This GUI-based pin configuration capability is extremely helpful if the developer is starting a board design in parallel with code development. For example, the firmware developer can provide a printout of potential pin-out options to be evaluated by the hardware designer. In an effort to reduce overall system cost, the hardware designer then selects the pin-out that results in the simplest printed circuit board (PCB) layout and the smallest number of board layers. In addition, the AppBuilder tool enables developers to easily see and assess the feasibility of last-minute pin out changes.

Another significant advantage of using a graphical tool for pin configuration is that it eliminates the need for pin configuration code development and abstracts any complexity that may exist within pin configurations. As changes are made to the pin-out or pin configuration, the tool automatically generates the corresponding code. The following figure shows an example of the pin configuration code generated by AppBuilder software. In Figure 2, the pin configuration code uses functions defined in the Hardware Access Layer, which is part of the Precision32 Software Development Kit (SDK) and provides an access layer for device registers.

```
#include "gPB.h"

// Include peripheral access modules used in this file
#include <SI32_PBCFG_A_Type.h>
#include <si32_device.h>
#include <SI32_PBSTD_A_Type.h>
#include <SI32_PBHD_A_Type.h>

void pb_enter_default_mode_from_reset(void)
{
  SI32_PBCFG_A_unlock_ports(SI32_PBCFG_0);

  // PB0 Setup
  SI32_PBSTD_A_set_pins_analog(SI32_PBSTD_0, 0xF000);
  SI32_PBSTD_A_set_pins_push_pull_output(SI32_PBSTD_0, 0x01E4);
  SI32_PBSTD_A_write_pbskipen(SI32_PBSTD_0, 0x0E00);

  // PB1 Setup
  SI32_PBSTD_A_set_pins_push_pull_output(SI32_PBSTD_1, 0x1C08);
  SI32_PBSTD_A_write_pbskipen(SI32_PBSTD_1, 0x0008);

  // Enable Crossbar0 signals & set properties
  SI32_PBCFG_A_enable_xbar0l_peripherals(SI32_PBCFG_0,
                                         SI32_PBCFG_A_XBAR0L_USART0EN |
                                         SI32_PBCFG_A_XBAR0L_USART0FCEN |
                                         SI32_PBCFG_A_XBAR0L_USART0CEN |
                                         SI32_PBCFG_A_XBAR0L_CMP0SEN |
                                         SI32_PBCFG_A_XBAR0L_CMP0AEN);
  SI32_PBCFG_A_enable_crossbar_0(SI32_PBCFG_0);

  // Enable Crossbar1 signals & set properties
  SI32_PBCFG_A_enable_crossbar_1(SI32_PBCFG_0);
}
```

Figure 2. Configuration Code Generated by the AppBuilder Tool

In addition to generating code, a pin configuration tool should be intelligent enough to prompt the user for any errors or warnings detected. For example, if the user incorrectly configures a pin for digital mode when the pin is selected as an analog input to the analog-to-digital converter (ADC), the tool should prompt the user to fix the issue with an automatic notification that an analog pin is currently configured for digital mode.

The AppBuilder tool, for example, provides an error list for any detected errors. Double-clicking on any error then takes the user to the configuration setting that is incorrect and highlights it in red. Once any detected conflicts are resolved, errors are automatically removed from the error list. This same feature is not limited to pin configuration but also exists for peripheral configuration. For example, if the developer configures the I2C0 module for use but forgets to enable the clock gate to the I2C0 module, AppBuilder will generate an error notification.

Error List

	Name	Location	Description
!	CLKCTRL_0	Default Mode	APB clock must be enabled to I2C0.
!	PB0.0	Default Mode Crossbar0	Signal USART0_TX requires Digital Push-Pull Output mode.
!	PB0.2	Default Mode Crossbar0	Signal USART0_RTS requires Digital Push-Pull Output mode.
!	CLKCTRL_0	Default Mode	APB clock must be enabled to USART0.
!	PB2.11	Default Mode Crossbar1	CMP1 requires pin be skipped.

Figure 3. Example of AppBuilder Error List

If the user double-clicks the error, the tool automatically brings up the parameter that must be changed and highlights it in red. If the error, as shown in Figure 3, is double-clicked, the Clock Control Configuration window will open with the I2C0 clock enable selector highlighted in red. Checking the I2C0 box within the Clock Control Configuration window removes the error from the error list window.

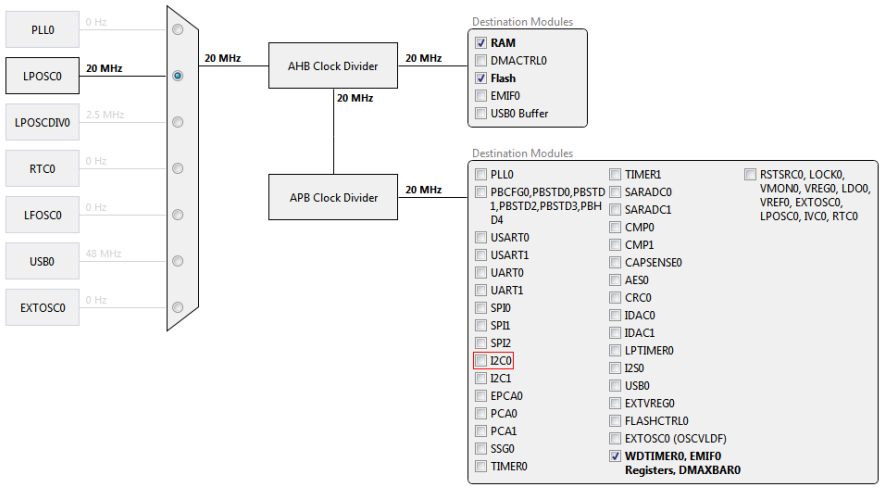

Figure 4. Clock Control Configuration with I2C Error

Without access to a graphical tool for configuration, a developer may need to spend a significant amount of time debugging a configuration, especially if the error is a single-bit setting. By using a graphical

configuration tool, errors are automatically highlighted, and the developer can spend his or her energy and time on further development instead of debugging.

Configuring microcontroller peripherals poses another obstacle for developers. Without the assistance of a software configuration tool, this step is typically a tedious process. Developers need to read through lengthy data sheets and reference manuals to determine which bits within a peripheral must be configured to achieve the desired setting. This is a tedious process for developers, especially when simultaneously evaluating several MCUs from multiple vendors. However, with the help of a software tool, such as AppBuilder, developers can easily see all of the configurable parameters and change configurations as needed. The AppBuilder tool, for example, enables quick configuration through check boxes, drop-down fields and text-boxes.

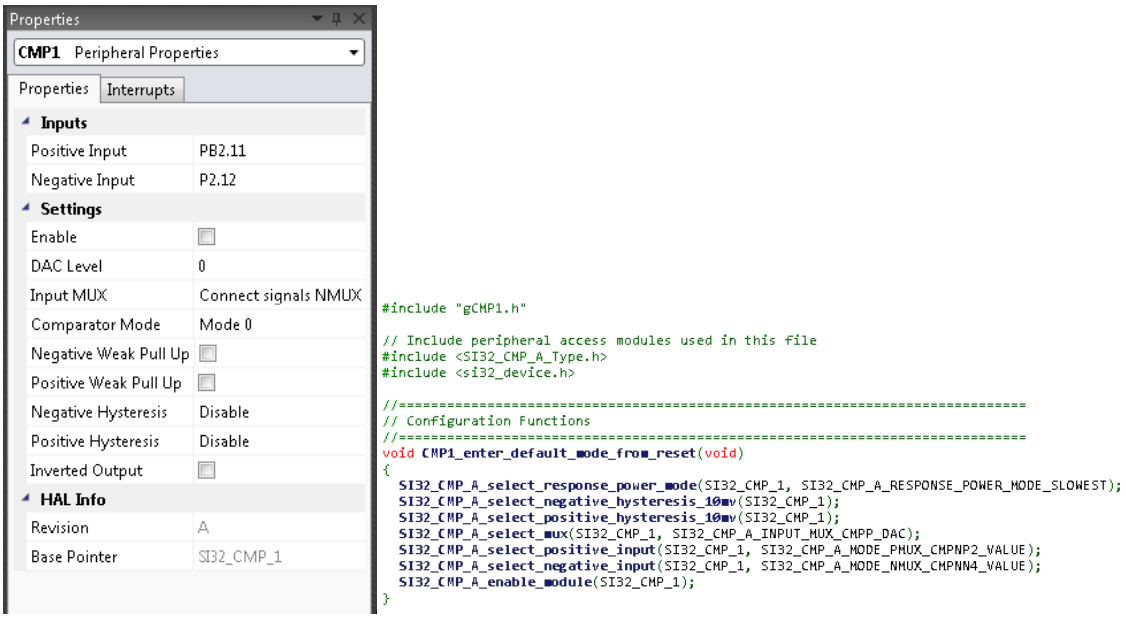

Figure 5. Example of Simplified Configuration with AppBuilder Tool

As with the port configuration, any peripheral settings will automatically generate the corresponding configuration code, and the peripheral configuration errors are highlighted in the error list window. AppBuilder software also enables developers to graphically configure the advanced high-speed bus (AHB) and advanced peripheral bus (APB) clock sources. Figure 4 shows AppBuilder's Clock Control Configuration window. As a result, there is no need for any configuration code development. The developer can perform all of the initial startup configuration within the tool, which further reduces overall development time.

At this point in the development process, the port pins, peripherals and clock sources are configured within the software tool, and the developer is now ready to move on to writing firmware to implement the application. At a bare minimum, a software configuration tool should output the generated files that can then be imported or added to an existing project. However, with the AppBuilder tool, the system configuration can be exported into a new project that can open automatically after exporting the source to a project. Once the new project is open within an integrated development environment (IDE), the developer can begin adding firmware and testing the applications. The AppBuilder tool allows developers to start from scratch and end with a new project with automatically generated code that configures the microcontroller for use opened in an IDE and ready for further development.

Final Considerations

When evaluating a microcontroller platform for a new embedded design, developers should select a solution that includes GUI-based software tools that streamline and automate the prototyping and configuration process. Next-generation GUI-based software tools, such as Silicon Labs' AppBuilder, can help developers configure the port pins and peripherals of a 32-bit microcontroller quickly and easily, thereby speeding time-to-market. Once developers have identified potential MCUs that meet the specifications of their applications, they can use GUI-based tools to determine valid pin-out options. If the developer identifies more than one valid microcontroller option, candidate pin-outs can be provided to the hardware designer who can choose the pin-out that will result in the simplest PCB layout. To further eliminate design complexity, peripherals, clocks and port pins can all be configured using the graphical interface.

Rapid prototyping utilities like Silicon Labs' AppBuilder tool enable developers to quickly and graphically configure a 32-bit microcontroller, often without having to read data sheets or learn register settings. To further simplify the design process, the AppBuilder tool also generates source code and a project that the developer can use as a starting point for their firmware development. Ultimately, GUI-based software configuration tools like AppBuilder provide an excellent resource for developers who want to reduce their overall design cost and complexity and speed time-to-market.

# # #