

PRECISION32™ PORT I/O CROSSBAR DECODER

1. Introduction

Precision32™ devices use one or more Port I/O Crossbar Decoders to assign internal digital signals to Port I/O pins. A crossbar decoder provides the system designer with flexibility to customize the pinout according to the needs of the application. The Port I/O Crossbar Decoder is particularly useful in low pin count devices where the number of internal digital signals outnumber the available I/O pins.

The crossbars are fully supported by the Silicon Labs Precision32 SDK, including Hardware Access Layer (HAL) routines and code examples showing how to configure a crossbar for a particular peripheral. Additionally, the Precision32 AppBuilder application provides a graphical interface to easily configure pins in a crossbar.

Figure 1 shows an example of how internal signals are routed to the Port Banks of SiM3U1xx devices through the use of two crossbar decoders. Port Bank 0 (PB0) and Port Bank 1 (PB1) are connected to crossbar 0, and Port Bank 2 (PB2) and Port Bank 3 (PB3) are connected to crossbar 1.

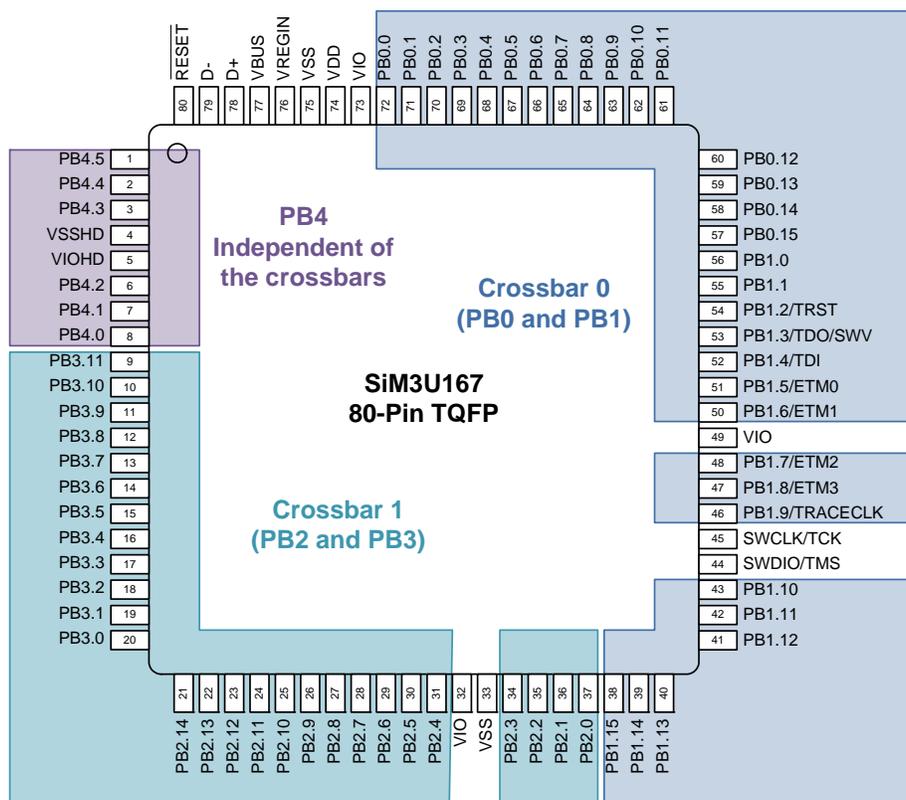


Figure 1. Crossbar Example on a SiM3U167 Device

2. Relevant Documentation

Precision32 Application Notes are listed on the following website: www.silabs.com/32bit-mcu.

- AN664: Precision32™ CMSIS and HAL User's Guide
- AN670: Getting Started with the Silicon Labs Precision32™ AppBuilder

3. Crossbar Function

The primary function of a crossbar decoder is to route internal digital signals to Port Bank pins. Figure 2 is a block diagram of crossbar 0 on SiM3U1xx devices. The inputs to the crossbar are a number of internal digital signals inside the device. The XBAR0H, XBAR0L, and PBSKIP registers in the port configuration module (PBCFG) define how the internal digital signals are mapped to the I/O pins of PB0 and PB1.

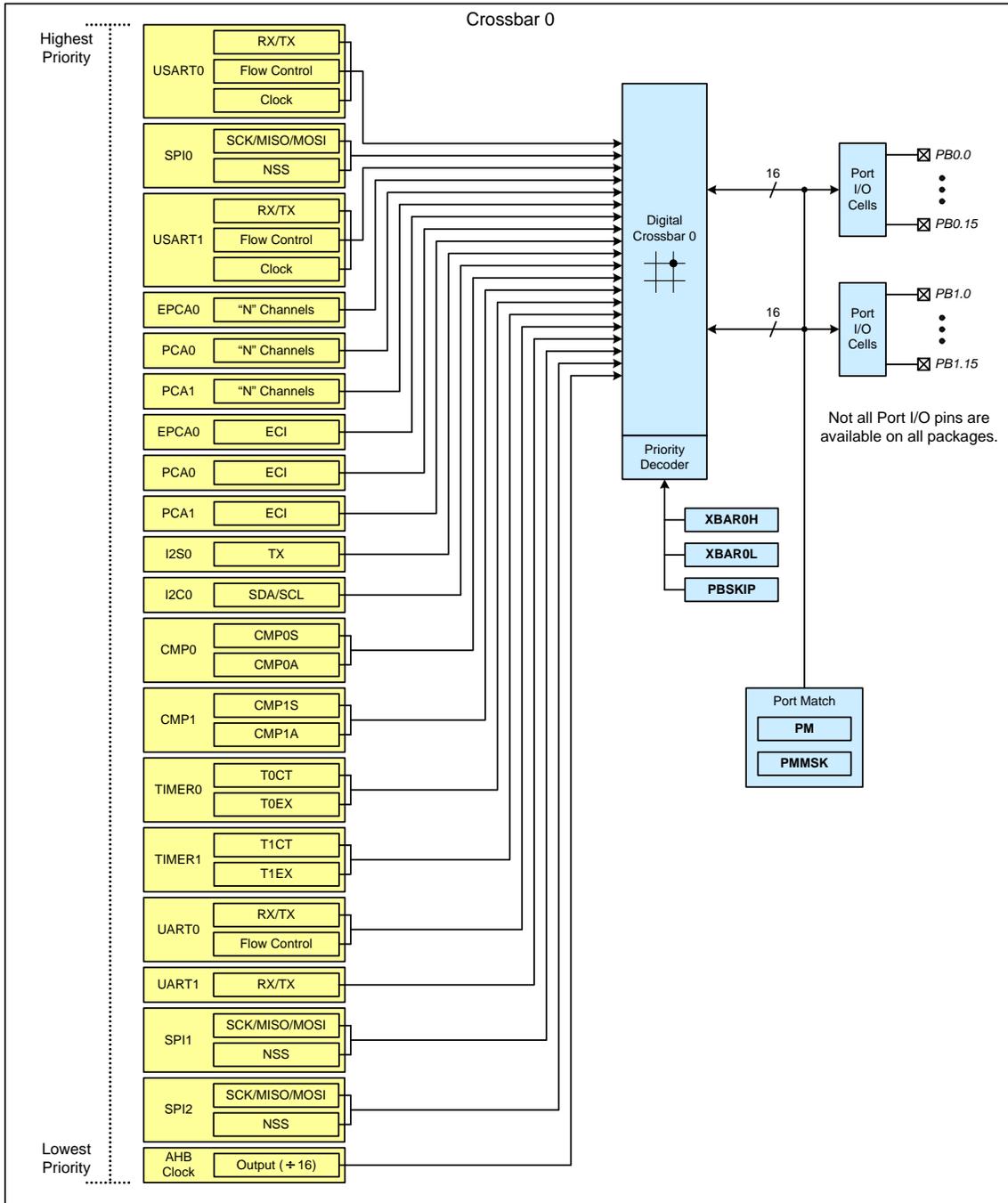


Figure 2. Example Routing of Internal Digital Signals to Port Banks on SiM3U1xx Devices

3.1. Crossbar Functionality on Reset

After a device reset, all crossbars enter a disabled default reset state. Port Bank pins connected to a disabled crossbar are forced into a high impedance digital input mode. Firmware must enable the crossbar associated with a specific Port Bank pin in order to use that pin as an output. In most applications, firmware will enable all crossbars on the device to control all the available I/O pins on the device.

When a crossbar is enabled with no internal signals selected to be routed to I/O pins, the crossbar provides full general purpose input/output (GPIO) access to the Port Banks associated with it. Pins with full GPIO access can be used as digital inputs, digital outputs, or may be used by various analog functions on the device. As internal signals are selected to be routed to I/O pins (or “enabled in the crossbar”), the crossbar claims pins from the associated port banks. Pins claimed by the crossbar cannot be used as GPIO and are under the full control of the crossbar and the associated peripheral.

3.2. Skipping Pins in the Crossbar

The crossbars have a pin-skipping feature for pins that must be reserved GPIO or analog functions. Any Port Bank pin with its corresponding PBSKIPEN bit set to 1 cannot be claimed by the crossbar and will remain available for GPIO or analog functions. The ability to have the crossbar skip certain pins is useful when a system designer is trying to achieve a specific pinout for the device.

3.3. Crossbar Priority Order

As internal signals are enabled in a crossbar, the crossbar claims pins from the Port Banks to connect to the internal signal, starting with the least significant Port Bank pin and finishing with the most significant Port Bank pin. As an example, crossbar 0 of SiM3U1xx devices would start with PB0.0, then PB0.1, and continue in this fashion until reaching PB1.15. If the crossbar encounters a pin that has its PBSKIPEN bit set to 1, it skips over the pin and claims the next available pin. Any pin not claimed by the crossbar can be used for GPIO or analog functions.

The crossbar uses a priority order to assign enabled internal signals to claimed Port Bank pins. This priority order varies with the specific crossbar implementation. Figure 3 shows an example priority order from crossbar 0 of SiM3U1xx devices. In this example, there are four enabled peripherals that require pin assignment: SPI0, EPCA0, UART0, and UART1. From the enabled peripherals, SPI0 has the highest priority, so it will be assigned to the first three pins claimed by the crossbar. Note that in this example configuration, firmware configured the first 8 pins of PB0 (PB0.0 - PB0.7) to be skipped by the crossbar; the crossbar will assign the SPI0 pins to PB0.8, PB0.9, and PB0.10. Following the priority order, the EPCA0 pins are assigned to PB0.11, PB0.12, PB0.15, PB1.0. The PB1.2, PB0.13 and PB0.14 pins are not assigned to EPCA0 because they are configured to be skipped by the crossbar. UART0 and UART1 are assigned to the next four available pins: PB1.3, PB1.4, PB1.5, and PB1.6. The remaining pins (PB1.7–PB1.15) are not claimed by the crossbar.

3.4. Creating a Flexible Device Pinout

The definition of a system can sometimes change in the middle of the design cycle, necessitating a pinout change. Planning ahead for such changes in pinout can save costly PCB revisions and decrease time to market when a system definition change does occur. In the example pinout shown in Figure 3, SPI0 is used in 3-wire mode. If the communication protocol was changed from 3-wire to 4-wire mode, then PB0.11 would be used for the NSS signal, causing all peripherals of lower priority order to shift by one pin. Using the crossbar’s skip functionality, the system designer can plan ahead for such a change by skipping PB0.11 when the specification calls for 3-wire SPI mode. The skipped pin can later be un-skipped if the specification later requires the use of 4-wire SPI without affecting the location of peripherals with a lower priority order. If the specification does not change, the skipped pin can be used for GPIO (e.g., to control an LED or as a debug signal). Adding a few skipped pins when determining the original device pinout can allow future functionality to be added with minimal impact on the device pinout.

4. Configuring the Crossbar and Port I/O in Firmware

The Precision32 AppBuilder application provides a graphical interface to easily configure pins in the crossbars. This software uses the Hardware Access Layer (HAL), a part of the Silicon Labs SDK package that enables rapid development on SiM3xxx devices. The crossbars and Port Banks on SiM3xxx devices are part of the PBCFG and PBSTD modules.

The following steps show an example of how to initialize the crossbars and pins on SiM3U1xx devices to achieve the pinout shown in Figure 3 using the Silicon Labs HAL:

1. Enable the APB clock to the I/O modules:

```
// enable APB clock to the Port Bank module
SI32_CLKCTRL_A_enable_apb_to_modules_0 (SI32_CLKCTRL_0, SI32_CLKCTRL_A_APBCLKGO_PBOCEN_MASK);
```

2. Configure pins to be skipped by the crossbars and enable signals in the crossbars. A full list of signal names that may be enabled in the crossbars can be found in a file named SI32_PBCFG_A_Support.h.

```
// Configure PBO.0 - PBO.7, PBO.13 and PBO.14 to be skipped by the crossbar
SI32_PBSTD_A_write_pbskipen(SI32_PBSTD_0, 0x60FF);

// enable signals in the crossbar
SI32_PBCFG_A_enable_xbar0_signal(SI32_PBCFG_0, SI32_XBAR0_SPIO);
SI32_PBCFG_A_enable_xbar0_signal(SI32_PBCFG_0, SI32_XBAR0_EPCAO_CEXO_5);
SI32_PBCFG_A_enable_xbar0_signal(SI32_PBCFG_0, SI32_XBAR0_UART0);
SI32_PBCFG_A_enable_xbar0_signal(SI32_PBCFG_0, SI32_XBAR0_UART1);
```

3. Configure the functional and output mode of each pin:

```
// SPIO input/output mode setup (3-wire mode)
// configure PBO.8 and PBO.10 as outputs, and PBO.9 as an input
SI32_PBSTD_A_set_pins_push_pull_output(SI32_PBSTD_0, 0x0500);
SI32_PBSTD_A_set_pins_digital_input(SI32_PBSTD_0, 0x0200);

// EPCA input/output mode setup (6 channels)
// configure PBO.11, PBO.12, PBO.15 and PB1.0-PB1.2 as outputs
SI32_PBSTD_A_set_pins_push_pull_output(SI32_PBSTD_0, 0x9800);
SI32_PBSTD_A_set_pins_push_pull_output(SI32_PBSTD_1, 0x0007);

// UART0 input/output mode setup
// configure PB1.3 as an output, and PB1.4 as an input
SI32_PBSTD_A_set_pins_push_pull_output(SI32_PBSTD_1, 0x0008);
SI32_PBSTD_A_set_pins_digital_input(SI32_PBSTD_1, 0x0010);

// UART0 input/output mode setup
// configure PB1.5 as an output, and PB1.6 as an input
SI32_PBSTD_A_set_pins_push_pull_output(SI32_PBSTD_1, 0x0020);
SI32_PBSTD_A_set_pins_digital_input(SI32_PBSTD_1, 0x0040);
```

4. Enable the crossbar or crossbars:

```
// enable crossbar(s)
SI32_PBCFG_A_enable_crossbar_0(SI32_PBCFG_0);
SI32_PBCFG_A_enable_crossbar_1(SI32_PBCFG_0);
```

CONTACT INFORMATION

Silicon Laboratories Inc.

400 West Cesar Chavez
Austin, TX 78701
Tel: 1+(512) 416-8500
Fax: 1+(512) 416-9669
Toll Free: 1+(877) 444-3032

Please visit the Silicon Labs Technical Support web page:
<https://www.silabs.com/support/pages/contacttechnicalsupport.aspx>
and register to submit a technical support request.

The information in this document is believed to be accurate in all respects at the time of publication but is subject to change without notice. Silicon Laboratories assumes no responsibility for errors and omissions, and disclaims responsibility for any consequences resulting from the use of information included herein. Additionally, Silicon Laboratories assumes no responsibility for the functioning of undescribed features or parameters. Silicon Laboratories reserves the right to make changes without further notice. Silicon Laboratories makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Silicon Laboratories assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. Silicon Laboratories products are not designed, intended, or authorized for use in applications intended to support or sustain life, or for any other application in which the failure of the Silicon Laboratories product could create a situation where personal injury or death may occur. Should Buyer purchase or use Silicon Laboratories products for any such unintended or unauthorized application, Buyer shall indemnify and hold Silicon Laboratories harmless against all claims and damages.

Silicon Laboratories and Silicon Labs are trademarks of Silicon Laboratories Inc.
Other products or brandnames mentioned herein are trademarks or registered trademarks of their respective holders.