CONFIGURING THE PORT I/O CROSSBAR DECODER

Introduction

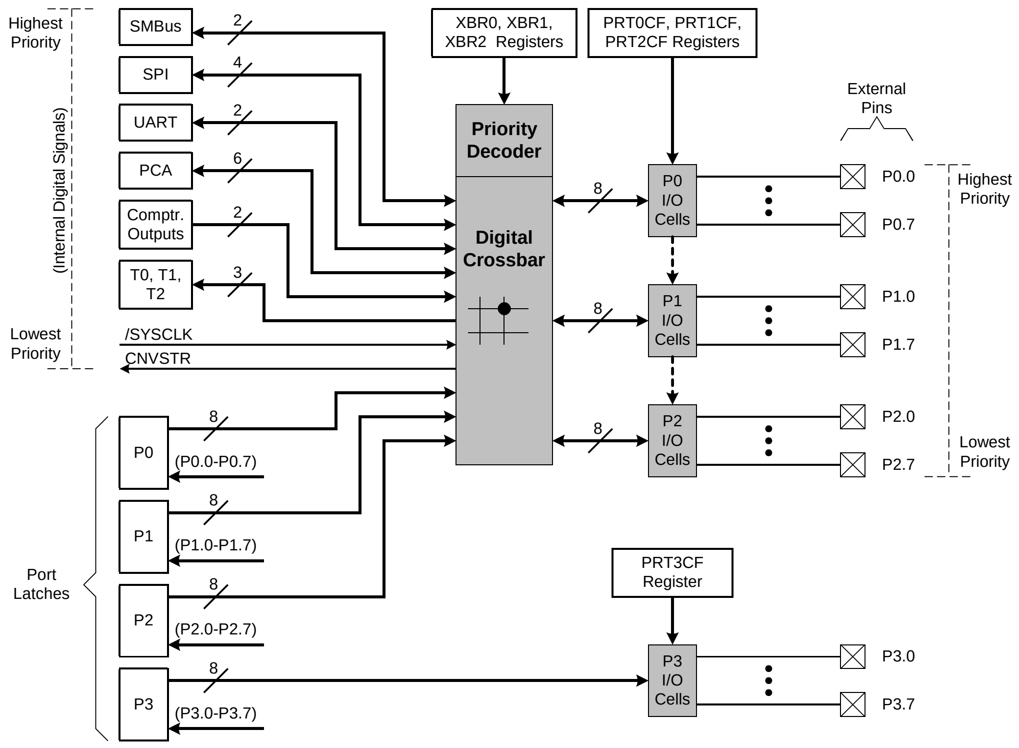
The Crossbar, shown in Figure 1, is a multiplexer that maps internal digital signals to Port I/O pins on the device.

This application note describes how to configure and use the Port I/O Crossbar Decoder. Software and examples are provided.

Key Points

- The Crossbar MUST be enabled in order to use any Port 0, Port 1, or Port 2 port pins as outputs.
- The Crossbar registers should be configured before any of the digital peripherals are enabled.
- The Crossbar registers are typically configured once at reset, near the top of the Reset handler, and then left alone.
- The Crossbar settings change the pinout of the device.
- Each Crossbar setting results in a device pinout that is unique. If you enable or disable peripherals in the Crossbar, the pinout WILL change.
- The **output** mode of the port pins (open-drain or push-pull) must be explicitly set for the output port pins, even those assigned by the Crossbar. Exceptions to this are the SDA and SCL pins on the SMBus and the RX pin on the UART, which are automatically configured as open-drain.
- The open-drain or push-pull mode of Crossbar-assigned **input** pins (like NSS or /INT0, for example) is not important. These pins are con-

Figure 1. Crossbar Decoder Block Diagram



figured as inputs regardless of the corresponding Port Configuration Register setting.

- To configure a GPIO pin as an input, the Port Configuration Register bit associated with that pin must be cleared, which selects that pin to have an open-drain output. Additionally, the Port bit associated with that pin must be set to a '1' which tri-states the pin or loosely pulls it high if WEAKPUD in XBR2 is set to '0'. This is the reset configuration of the port pins.
- The value at the port pins can be read at any time by reading the associated port SFR, regardless of the Crossbar register setting or whether the pin is configured as an input or an output.
- The Enable bits in the Crossbar registers are unique and distinct from the enable bits in the digital peripherals themselves.
- Peripherals do not need to be enabled in the Crossbar in order to be used (for example, a PCA module can generate interrupts even if its output is not routed to a pin).
- Peripherals that are enabled in the Crossbar, but disabled in their own SFRs still control the output state of the port pins. That is, the port pins can be read at any time, but the outputs are controlled exclusively by the owning peripheral and cannot be accessed as general-purpose outputs.
- The four external interrupts on Port 1 (P1.[4..7]) are triggered by a falling edge at the pin, regardless of the source of the falling edge, the Crossbar setting, or the output mode of the port pin.
- Unlike the standard 8051, true push-pull outputs are provided. If the 'strong-then-weak' pull-up function of the 8051 is required, it can be emulated in software by configuring the associated port output as 'push-pull' followed by a configuration to 'open-drain'.

Overview

The Crossbar governs the pin mapping of internal digital signals and port latches to physical I/O pins on the device. Optionally, it can connect port pins to a weak pull-up to prevent them from floating when they are in a high impedence state. The Crossbar manages the following digital input and output signals, grouped by peripheral. **Note: The Crossbar must be configured and enabled before the I/O of any of these peripherals can be accessed.** Also, output pins such as UART TX should be configured to push-pull mode in the corresponding PRTnCN register.

- **SMBus** – SDA and SCL signals can be routed to 2 port pins.
- **SPI** – MISO, MOSI, SCK, and NSS can be routed to 4 port pins.
- **UART** – RX and TX signals can be routed to 2 port pins.
- **PCA** – ECI input may be routed to a port pin and up to 5 of the CEXn outputs may be routed to port pins.
- **ADC** – The Convert Start (CNVSTR) signal may be routed to a port pin.
- **Comparators** – Comparator 0 and Comparator 1 outputs may be routed to port pins.
- **Timers** – Timer 0, Timer 1, and Timer 2 clock inputs and the T2EX reload signal may be routed to port pins.

The crossbar can also map the following signals to the to the port pins on the device:

- **/SYSCLK** – the system clock, inverted and buffered.
- **/INT0** and **/INT1** – external interrupt input pins.

The Crossbar affects the pins in Port 0, Port 1, and Port 2. If available, Port 3 as well as unassigned pins on lower ports operate as normal GPIO and are controlled by their respective port latches.

The Crossbar provides two key system benefits:

- All unassigned GPIO port pins on Port 0, Port 1, and Port 2 are grouped contiguously.
- It provides flexibility in peripheral selection for reduced pin-count devices where some I/O ports may not be available. Peripheral selection is limited only by the **number** of port pins available, not by **which** port pins are available. This allows the system designer to choose which digital peripherals are available at the digital
  I/O pins on a pin-limited device.

Determining Device Pinout

The Crossbar assigns port pins to signals in priority order according to Table 4 . Therefore, the device pinout is based on peripheral selection in the Crossbar registers: XBR0, XBR1, and XBR2.

To determine the pinout, first configure the Crossbar registers based on the peripherals needed. Then, starting at the top of Table 4 , scan down until you reach the first enabled signal. This signal will use P0.0. If the signal belongs to a group of signals associated with a peripheral, the remaining signals in that group will be assigned from P0.1 up. For example, if the SPI is the first peripheral enabled, then SCK, MISO, MOSI, and NSS will be mapped to P0.0, P0.1, P0.2, and P0.3 respectively. The next enabled signal will be assigned P0.4. All unassigned pins behave as GPIO.

Example 1

Assume that the application calls for:

- •SPI
- •UART
- •2 PCA capture modules
- •/INT0
- •T2

Referencing the Port I/O Crossbar Register descriptions in the datasheet, the Crossbar registers are configured as follows:

```
XBR0 = 0x16 ; enable UART,
            ; 2 PCA capture
            ; modules, and SPI

XBR1 = 0x24 ; enable T2 and INT0

XBR2 = 0x40 ; enable Crossbar
```

Table 1 shows the pinout resulting from the above Crossbar configuration.

Table 1. Pinout for Example 1

Pin	Signal/Function
P0.0	SCK
P0.1	MISO
P0.2	MOSI
P0.3	NSS
P0.4	TX
P0.5	RX
P0.6	CEX0
P0.7	CEX1
P1.0	/INT0
P1.1	T2
P1.2 – P1.7	GPIO
P2.0 – P2.7	GPIO

Example 2

Assume that the application calls for:

- •UART
- •/INT1
- •/SYSCLK
- •CNVSTR

The Crossbar registers are configured as follows:

```
XBR0 = 0x04 ; enable UART

XBR1 = 0x90 ; enable /INT1 and
            ; /SYSCLK

XBR2 = 0x41 ; enable Crossbar and
            ; CNVSTR
```

Table 2 shows the pinout resulting from the above Crossbar configuration.

Table 2. Pinout for Example 2

Pin	Signal/Function
P0.0	TX
P0.1	RX
P0.2	/INT1
P0.3	/SYSCLK
P0.4	CNVSTR
P0.5 – P0.7	GPIO
P1.0 – P1.7	GPIO
P2.0 – P2.7	GPIO

Example 3

Assume that the application calls for:

- SMBus
- UART
- ECI (PCA Counter Input)
- /INT0
- /INT1
- T2EX
- CNVSTR

To enable the above signals, the Crossbar registers are configured as follows:

```
XBR0 = 0x45 ; enable ECI, UART,
            ; and SMBus

XBR1 = 0x64 ; enable /INT0, /INT1,
            ; and T2EX

XBR2 = 0x41 ; enable Crossbar and
            ; CNVSTR
```

Table 3 shows the pinout resulting from the above Crossbar configuration.

Table 3. Pinout for Example 3

Pin	Signal/Function
P0.0	SDA
P0.1	SCL
P0.2	TX
P0.3	RX
P0.4	ECI
P0.5	/INT0
P0.6	/INT1
P0.7	T2EX
P1.0	CNVSTR
P1.1 – P1.7	GPIO
P2.0 – P2.7	GPIO

Table 4. Priority Crossbar Decode Table

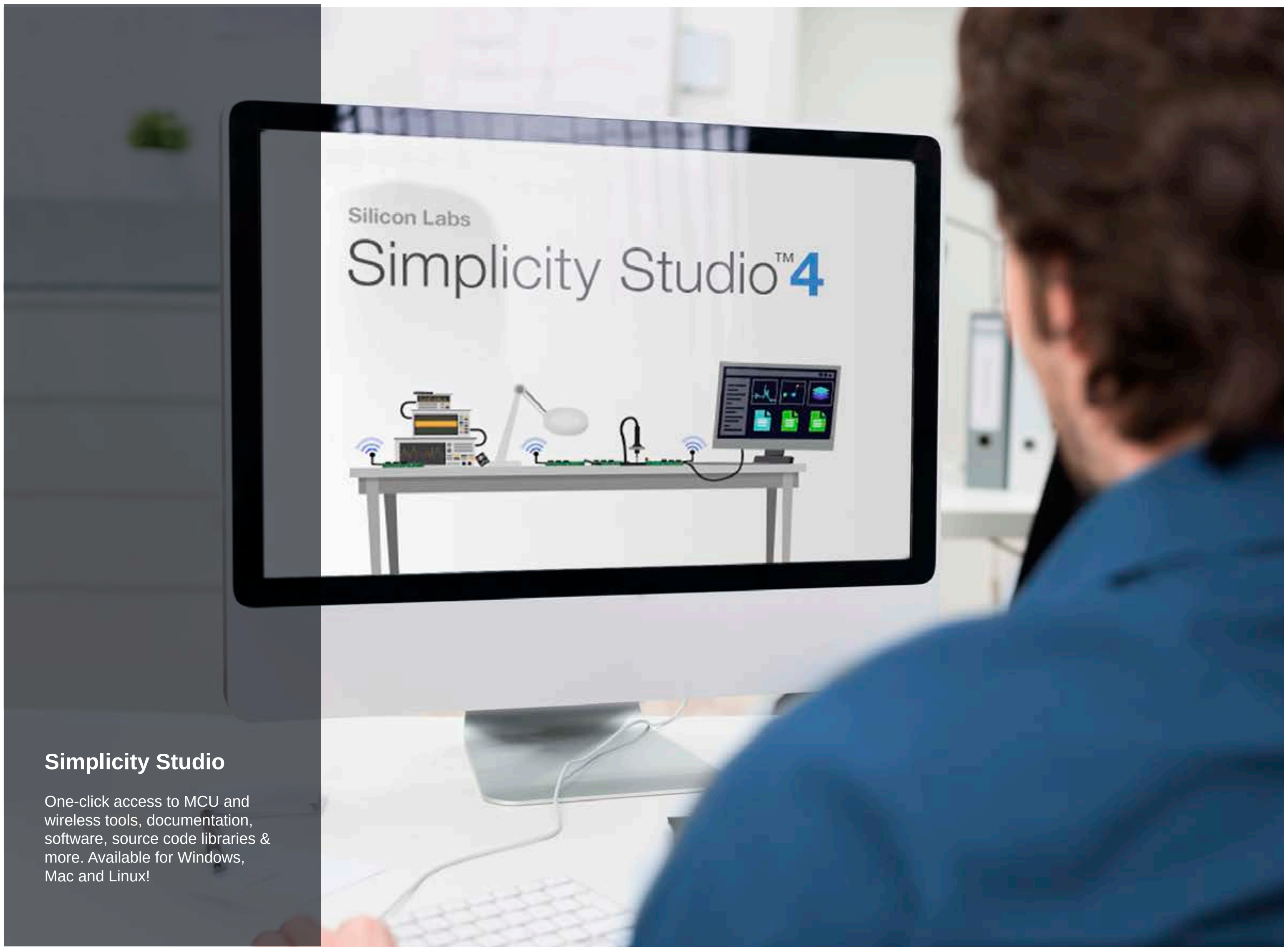
	P0								P1								P2							
PIN I/O	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
SDA	●																							
SCL		●																						
SCK	●		●																					
MISO		●		●																				
MOSI			●		●																			
NSS				●		●																		
TX	●		●		●		●																	
RX		●		●		●		●																
CEX0	●		●		●		●		●															
CEX1		●		●		●		●		●														
CEX2			●		●		●		●		●													
CEX3				●		●		●		●		●												
CEX4					●		●		●		●		●											
ECI	●	●	●	●	●	●	●	●	●	●	●	●	●	●										
CP0	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●									
CP1	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●								
T0	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●							
/INT0	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●						
T1	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●					
/INT1	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●				
T2	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●			
T2EX	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●		
/SYSCLK	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
CNVSTR	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●

In the Priority Decode Table, a dot (●) is used to show the external Port I/O pin (column) to which each signal (row) can be assigned by the user application code.

Software Example

```
//-----------------------------------------------------------------------------
// Crossbar1.c
//-----------------------------------------------------------------------------
// Copyright 2002 Cygnal Integrated Products, Inc.
//
// AUTH: FB
// DATE: 16 DEC 02
//
// This example shows how to configure the Crossbar.
//
// Target: C8051F00x, C8051F01x
//
// Tool chain: KEIL Eval ‘c’
//

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------
#include <c8051f000.h>                    // SFR declarations


//-----------------------------------------------------------------------------
// Global CONSTANTS
//-----------------------------------------------------------------------------
sbit  LED = P1^6;                         // green LED: ‘1’ = ON; ‘0’ = OFF

//-----------------------------------------------------------------------------
// Function PROTOTYPES
//-----------------------------------------------------------------------------
void PORT_Init(void);

//-----------------------------------------------------------------------------
// MAIN Routine
//-----------------------------------------------------------------------------
void main (void) {

   // disable watchdog timer
   WDTCN = 0xde;
   WDTCN = 0xad;

   PORT_Init();                           // set up Crossbar and GPIO

   EA = 1;                                // enable global interrupts

   LED = 1;                               // turn on LED

   while (1) {
      PCON |= 0x01;                       // set IDLE mode
   }
}

//-----------------------------------------------------------------------------
// PORT_Init
//-----------------------------------------------------------------------------
//
// This routine initializes the Crossbar and GPIO pins
```

```
//
// Pinout:
//    P0.0 - UART TX
//    P0.1 - UART RX
//    P0.2 - /SYSCLK
//
//    P1.6 - LED (GPIO)
//
void PORT_Init(void)
{
   XBR0 = 0x04;                          // Enable UART
   XBR1 = 0x80;                          // Enable /SYSCLK
   XBR2 = 0x40;                          // Enable crossbar and weak pull-ups

   PRT0CF |= 0x05;                       // Enable UART TX and /SYSCLK as push-pull
                                         // outputs
   PRT1CF |= 0x40;                       // Enable P1.6 (LED) as a push-pull output

}
```



Simplicity Studio

One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!

IoT Portfolio
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support and Community
community.silabs.com

SILICON LABS

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

http://www.silabs.com