Keywords: SCI, serial, serial port, synchronous, asynchronous, baud, baud rate, I/O, UART, UARTs, USART, interrupt-driven, RS-232, RS232

APPLICATION NOTE 3421

Interfacing 8051-based Microcontrollers to an SCI Port

By: Kevin Self
Dec 17, 2004

Abstract: This application note describes how to configure the UARTs of a High-Speed Microcontroller or Ultra-High-Speed Flash Microcontroller to communicate with an SCI-enabled device. It begins with a brief discussion of the differences between SCI and UART modules, and concludes with a practical example of how to configure an 8051-based Dallas Semiconductor microcontroller UART to communicate with an SCI module.

Introduction

The Serial Communications Interface (SCI) is a high-speed serial I/O port that permits synchronous or asynchronous communication between devices. It allows a microcontroller to interface to a wide range of similarly-enabled peripherals, as well as a standard RS-232 interface. The exact implementation of the SCI differs between device manufacturers; many devices accommodate features such as full-duplex communication in asynchronous mode, parity checking, error detection, and a programmable character length from one to nine bits.

All 8051-based Dallas Semiconductor microcontrollers are capable of communicating with SCI-enabled devices, even though SCI functionality is not explicitly listed on the microcontrollers' list of features. All our microcontrollers incorporate one to three 8051-type UARTs which can be configured to operate in most of the common SCI modes.

This application note describes how to configure the UARTs of a high-speed microcontroller or ultra-high-speed flash microcontroller to communicate with an SCI-enabled device. It begins with a brief discussion of the differences between SCI and UART modules, and concludes with a practical example of how to configure an 8051-based Dallas Semiconductor microcontroller UART to communicate with an SCI module. A code example is provided that demonstrates how to initialize the microcontroller and perform a simple test to ensure that the devices are communicating correctly.

Features of SCI

As mentioned above, SCI is a high-speed serial interface. It has many similarities to the 8051-style UARTs present on the Dallas Semiconductor 8051-based microcontrollers. The following is a list of SCI features and their counterparts in the UART. The user should be aware that not all SCI modules support all the features listed, so the user should carefully read the data sheet of the SCI-enabled device to understand how it will be used.

Feature	SCI	Dallas Semiconductor UART
Asynchronous mode	Available on most implementations	Serial mode 1, 2, 3
Synchronous mode	Available on some implementations	Serial mode 0 only
Character length	1 to 9, if optional character length is supported	8 or 9
Parity	Available on some implementations	Supported by software in 9-bit mode
Framing error	Yes	Yes
Idle character	Detects idle characters to wake device.	UART can not detect idle characters, but the UART microprocessor communication mode can be used to signal the UART to treat the next byte as an address/identifer.
Break character	SCI can transmit and receive break characters (00h).	Can transmit break character by taking serial port RX pin to logic 0. Receipt of break character may cause a framing error, depending on selected character length.

Example

Most SCI modules support the asynchronous communication format, many of them exclusively. The example here shows how to configure a Dallas Semiconductor 8051-based microcontroller to communicate asynchronously with an SCI-enabled device. In this case we will configure the microcontroller to communicate with a target SCI configured with the following characteristics:

- 10-bit Asynchronous mode; 1 start, 8 data, 1 stop bit
- Baud rate: 19200 bps

To communicate with this device, we will make the following decisions for setting up the Dallas Semiconductor microcontroller:

- Use Serial Port 0 for communications
- External clock source is 22.1184MHz
- The serial port will be configured for 10-bit asynchronous mode; 1 start, 8 data, 1 stop bit (This is serial port mode 1.)
- The baud-rate-generator clock source will be Timer 1 in auto-reload mode (Timer mode 2).

As all Dallas Semiconductor 8051-based microcontroller timers default to the original divide-by-12 mode of operation, this example has the advantage of being applicable to all Dallas Semiconductor devices, regardless of the clock divisor of the core. This is because the DS5000FP (divide-by-12), DS80C320 (divide-by-four), and DS89C450 (divide-by-1) all use the same serial port timing if the higher speed option for the timers is not selected. Detailed information about operation of the UART can be found in Serial I/O section of the appropriate User's Guide.

Because the SCI has dictated the format of the data, the Dallas Semiconductor microcontroller must next be initialized to the correct baud rate. The 8-bit auto-reload mode (timer mode 2) generates the baud rate from a user-selectable timer overflow driven by the external clock source. This adds a considerable flexibility to the design and simplifies development because the baud rate is easily selected in software, permitting a number of baud rates from the same clock source. The equation for determining baud rate is shown below:

$$baud\_rate = \frac{2^{SMOD\_0}}{32} \cdot \frac{osc\_frequency}{12 \cdot (256 - TH1)}$$

where osc_frequency is the frequency of the external clock source in MHz, TH1 is the 8-bit re-load value placed into the Timer 1 MSB SFR, and SMOD_0 (PCON.7) is the serial port 0 doubler enable bit. Alternatively, the following equation can be used to solve for the value of the 8-bit reload number TH1, if the baud rate and oscillator frequency are known:

$$TH1 = 256 - \frac{2^{SMOD\_0} \cdot osc\_frequency}{384 \cdot baud\_rate}$$

Assuming an external clock source of 22.1184MHz, a TH1 value of FDh will produce a target baud rate of 19200 with the doubler bit cleared. More information about baud rate selection can be found in Serial I/O section of the appropriate User's Guide.

The following brief assembly code example shows how to initialize serial port 0 to communicate with an SCI module configured for 10-bit asynchronous mode at 19200 bps. When operating successfully it will echo back any received characters. This function can be easily removed, making it a generic shell for any user-desired SCI communication application.

```
;SCI emulation example
; Simple transmit test to demonstrate how to configure 8051 UART to
; emulate an SCI module. Test code embedded in this example echoes back
; received characters.

org 0h            ;Reset vector.
ljmp start

org 23h           ;Serial port 0 vector.
ljmp SP0_ISR


org 100h          ;Start of code.
start:            ;Initialize Serial Port 0 for mode 1, 19200 baud
MOV TMOD,  #020h  ;Set timer 1 for mode 2 (8-bit auto reload)
MOV SCON0, #050h  ;SP0  10-bit asynchronous mode with receive enabled

;Now select the reload value based on baud rate and xtal frequency.
MOV TH1,   #0FDh ;19200 baud at 22.11 MHz
;MOV TH1,   #0FDh ;9600 baud at 11.059 MHz
;MOV TH1,   #0FAh ;9600 baud at 22.11  MHz

SETB TR1          ;Serial port is initialized, now start timer

;Enable Interrupts
MOV IE, #90h      ;This example supports interrupt-driven communications, so
                  ; enable global and serial port 0 interrupts.


;Test code in receive interrupt routine echoes back any received characters
; when combined with the loop here.
```

```
loop: sjmp loop

SP0_ISR:            ;Serial port 0 Interrupt Service Routine
jb  RI0, RIO_INT    ;Determine if receiver/transmitter was cause of interrupt.

TIO_INT:            ;Interrupt was caused by transmission.
;
;                    Placeholder for transmitter routine
;
CLR TI0
RETI

RIO_INT:            ;Interrupt was caused by reception
;
;                    Placeholder for receiver routine
;

MOV A, SBUF0        ;Test code that echoes back received character
MOV SBUF0, A        ; Remove for real code.

CLR RI0
RETI
```

Summary

The UARTs found in Dallas Semiconductor' 8051-based microcontrollers can easily be configured to interface with the SCI module found in many devices. This popular serial interface can operate in a variety of modes, but the most common are the 10/11-bit asynchronous modes used in RS-232 communications. Allowing Dallas Semiconductor microcontrollers to interface to SCI modules increases the overall system flexibility, as they can be connected to a wider array of embedded systems.

Although this example concentrated on the asynchronous mode of operation, Dallas Semiconductor microcontrollers can also be configured to interface with SCIs operating in the synchronous mode. The similarity of the SCI module to the 8051 UART allows this interface to be accomplished with minimal effort. Details about the synchronous mode (serial port mode 0) are found in the Serial I/O section of the appropriate User's Guide.

Related Parts		
DS2250T	Soft Microcontroller Module	Free Samples
DS2251T	128k Soft Microcontroller Module	Free Samples
DS2252T	Secure Microcontroller Module	Free Samples
DS5000	Soft Microcontroller Module	Free Samples
DS5000FP	Soft Microprocessor Chip	
DS5000T	Soft Microcontroller Module	Free Samples
DS5001FP	128k Soft Microprocessor Chip	Free Samples
DS5002FP	Secure Microprocessor Chip	Free Samples
DS5250	High-Speed Secure Microcontroller	
DS80C310	High-Speed Microcontroller	Free Samples

DS80C320	High-Speed/Low-Power Microcontrollers	Free Samples
DS80C323	High-Speed/Low-Power Microcontrollers	Free Samples
DS80C390	Dual CAN High-Speed Microprocessor	Free Samples
DS80C400	Network Microcontroller	Free Samples
DS80C410	Network Microcontrollers with Ethernet and CAN	Free Samples
DS87C520	EPROM/ROM High-Speed Microcontrollers	Free Samples
DS87C530	EPROM Microcontrollers with Real-Time Clock	Free Samples
DS89C430	Ultra-High-Speed Flash Microcontrollers	Free Samples
DS89C450	Ultra-High-Speed Flash Microcontrollers	Free Samples