

Chapter 8 The “Traffic Control” of the Serial Ports – Flow Control

Overview

Explain why flow control is needed and how to implement flow control system between LabVIEW block diagram and 8051 serial ports experiment.




Objective

- Learn how to make use of serial port function provided by LabVIEW to implement flow control.





Keyword

- Flow Control
- Xon/Xoff
- CTS/RTS

■ Explanation of control

	path	Controls>All Controls>Boolean
	explanation	Logical input provided : true or false
	path	Controls> All Controls>Boolean
	explanation	Logical output – like LED – lighten (true) / darken (false)
	path	Controls>Graph Indicators>Waveform Graph
	explanation	Display serial value by graph – X: time & Y: output value

■ Explanation of function

 <p style="text-align: center;">In Port.vi</p>	path	All Functions ► Advanced ► Port I/O ► In Port.vi
	explanation	Reads a signed integer from a specific address .
	Input	address specifies the address from which you want to read a 8-, 16-, or 32-bit signed integer.
	Output	data read is the byte (8 bits) of data read from the address specified.
 <p style="text-align: center;">Out Port.vi</p>	path	All Functions ► Advanced ► Port I/O ► Out Port.vi
	explanation	Writes a signed integer to the specified address .
	Input	address specifies the address in the I/O memory range to begin the read from. write value is the byte (8-bit value) to write to the systems's memory.
	path	Functions ► All Functions ► Boolean
	explanation	execute AND operator
	Input	value one & value two
	Output	The result after operation – true/false.
	path	All Functions ► Comparison
	explanation	compare if value one is equal to equal two
	Input	Two-value comparing
	Output	true(equal) / false(unequal)

In busy life of city, traffic is always crowded. So we always need a pointsman to indicate traffic to remain clear, therefore everyone can reach destination safely. This is difficult mission of pointsman. The serial port just likes freeway that connects data sending equipment and data receiving equipment. If data is blocked, flow control “the pointsman” should restrict stream of data, and data won’t be lost by accident. In real operating, if data receiver can’t process data immediately, data transmitter will be asked for flow control to prevent from loss of data.

There are two common ways for flow control in serial port.

◆ *Hardware flow control (RTS/CTS)*

To use two signal control pins in serial port (RTS and CTS) have handshaking capability to control flow. When data transmitter sends data to receiver, RTS signal will be asked to inform receiver to prepare for receiving data. In this case, if receiver has got RTS signal and it will be available to receive data. At the same time the receiver will activate CTS signal to send signal to transmitter. After getting this signal, transmitter will deliver data from serial port to receiver. As soon as the transmitter has got the signal from CTS, it will output data directly to the serial port in order to receive it by the receiver. On the contrary, if receiver is unable to receive any data, CTS signal will be asked to inform transmitter not to deliver data.

This method is used in the case that data processing of receiver is slower than transmitter. It means that receiver asks transmitter to control flow only in one direction. Since it needs two handshaking signal lines to control data flow, cost of hardware will be increased.

◆ *Software flow control (Xon/Xoff)*

This method controls data flow by sending Xon/Xoff code. Xon code is 11_H, and Xoff is 13_H. The method is that data receiving buffer will throw Xoff when buffer is going to be overflowed. When data transmitting buffer receive this control code, it will stop the transfer of data. As soon as data receiving buffer is empty and sends Xon code, the transmitter will send data again.

The method of control flow can save hardware line expense. It needs control code by using 2 byte numbers, so these two control codes can't be in delivery data for information. But ASCII code should be all right because 11_H and 13_H in ASCII code is control code and isn't conflict with another number. But it is very dangerous to deliver other data such as graph and Chinese code which may be conflict with these 2 control code and determined wrongly. So when you write Xon/Xoff program for flow control, should be careful. The solution to this problem is re-coding what you want to deliver not to use these 2 codes. Therefore, cost of software will be increased.

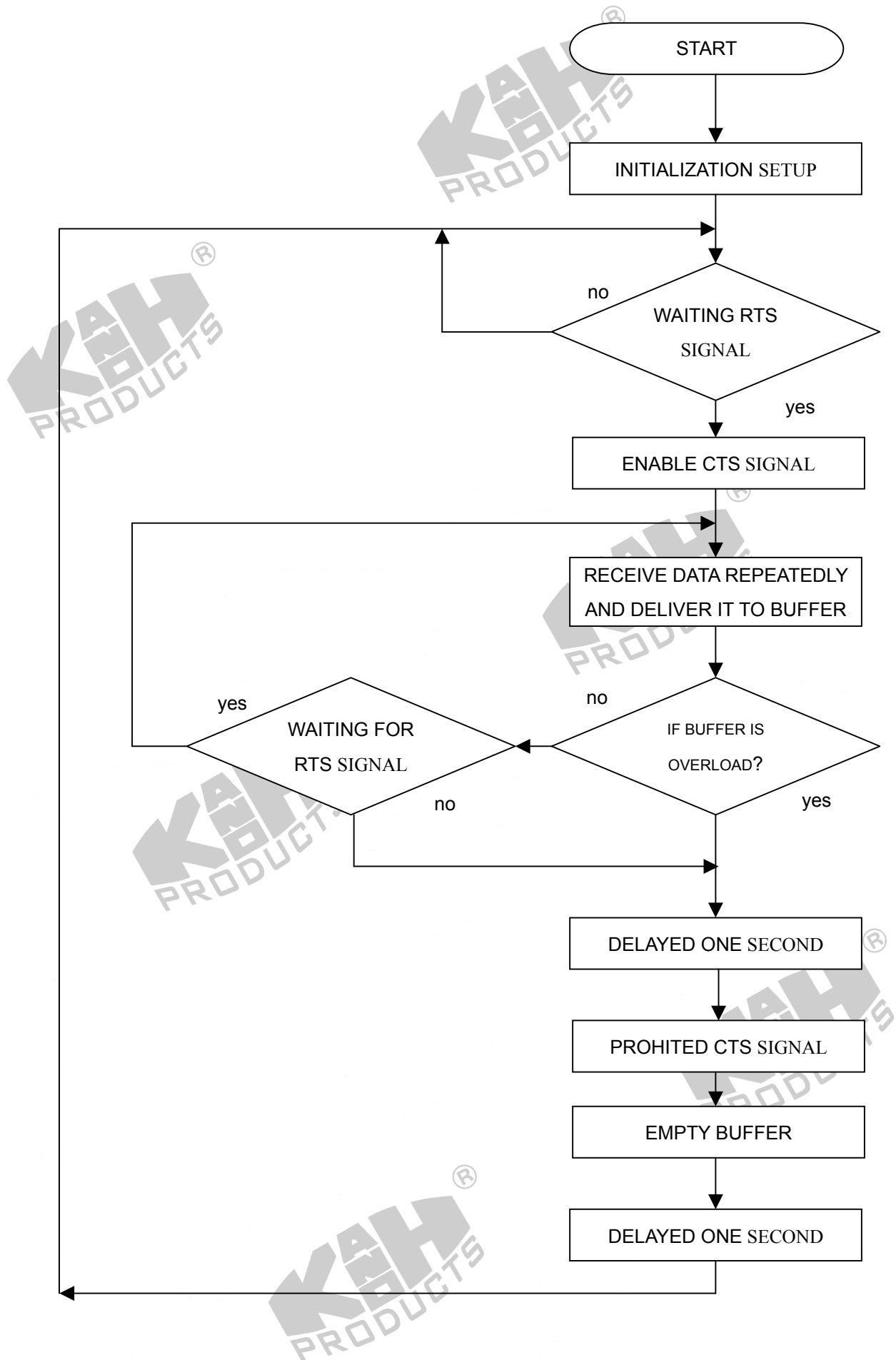
Practice 8-1 Hardware flow control

Objective

To continue the previous chapter, send the value of timer inside 8051 to display in LabVIEW by making use of hardware flow control (RTS/CTS)

1 、Beneath the page, it's flowchart of 8051 program. In this diagram, it simulates 8051 as data receiver. Explain the process step by step in the following.

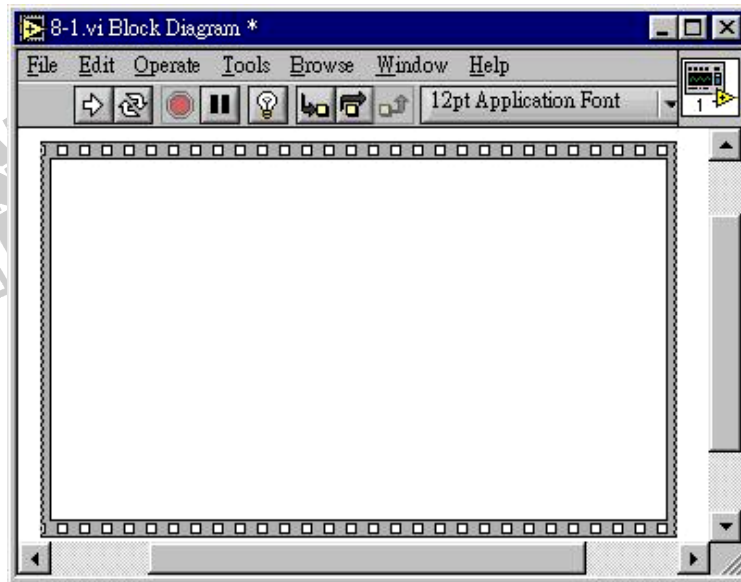
- A 、Processing speed of data receiver is slow. Although buffer exists, overflow may still happen.
- B 、When overflow happened, 8051 disables CTS signal which means it's not able to receive any data and ready to enter the procedure to process data.
- C 、Before overflow but RTS signal has not yet inspected, CTS signal will be disabled, and 8051 ready to enter the procedure to process data.
- D 、After buffer overflowed, it will be clear and hold for 2 seconds to simulate the situation of data processing.
- E 、After holding 2 seconds, next step is inspecting RTS signal then executing loop repeatedly.



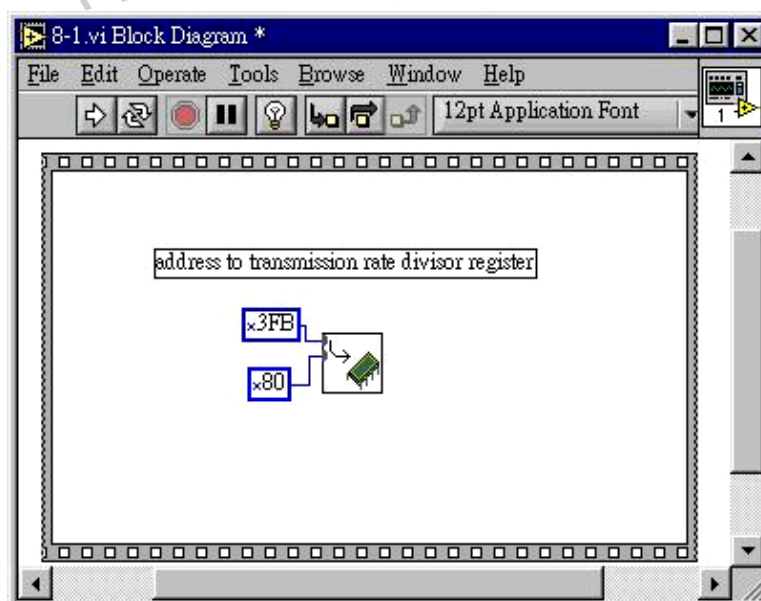
2 、 8051 source code of program.

RTS	reg	p2.7	Define name of pins
CTS	reg	p2.6	
	org	0h	
	jmp	init	
	org	30h	
init:	mov	sp,#30h	
	call	set_bps	
	clr	ri	
	clr	ti	
	setb	RTS	;Set initialization of pins
	setb	CTS	
start:	jb	RTS,\$;Waiting for RTS signal because IC of voltage level is
changed			;MAX-232will make signal inverse
	clr	CTS	;Set CTS signal as 1
	jnb	ri,\$;Receive data
	clr	ri	
	mov	a,sbuf	
	call	delay	;delay
	setb	CTS	;Set CTS signal as 0
	call	delay	;delay
	jmp	start	;Return to loop
set_bps:			
	mov	scon,#01010000B	
	mov	tmod,#00100000B	
	mov	th1,#FDH	
	setb	tr1	
	ret		
delay:			
	mov	r0,#ffh	
d1:			
	mov	r1,#ffh	
d2:			
	mov	r2,#20	
	djnz	r2,\$	
	djnz	r1,d2	
	djnz	r0,d1	
	ret		

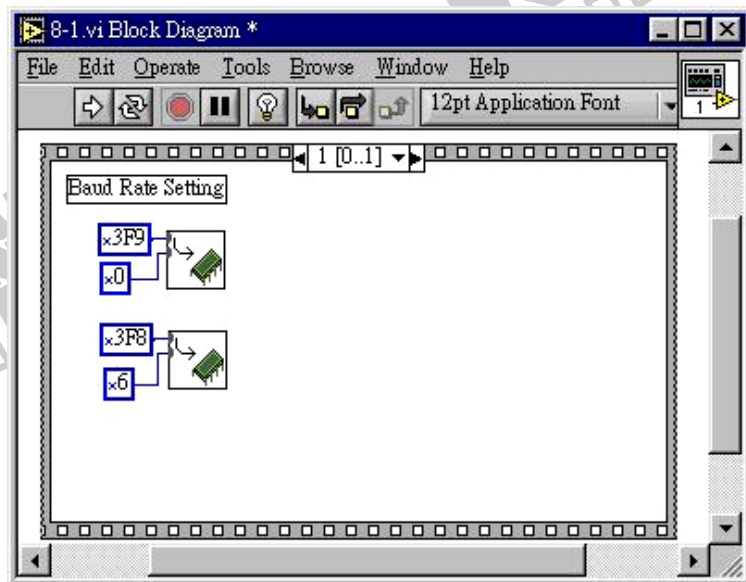
3、In the **Block Diagram**, please put into the **Stacked Sequence Structured** first. Due to the order of **Sequence** action in this practice, we arrange the order of execution with the “**Sequence**” function. Next the following steps take the COM 1 as the example 。



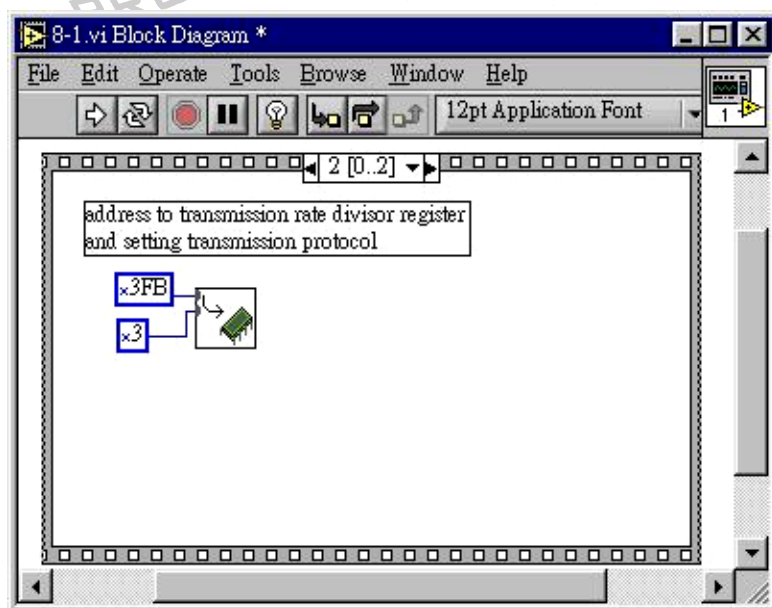
4、Firstly, address $3F8_H$ 、 $3F9_H$ to transmission rate divisor register to set transmission rate. All the variables here are all set as hexadecimal format, so it will be more convenient when setting values. For example, address is showed as hexadecimal, so you can fill in it directly here. It is also convenient to set input value according to bit function inside the register.



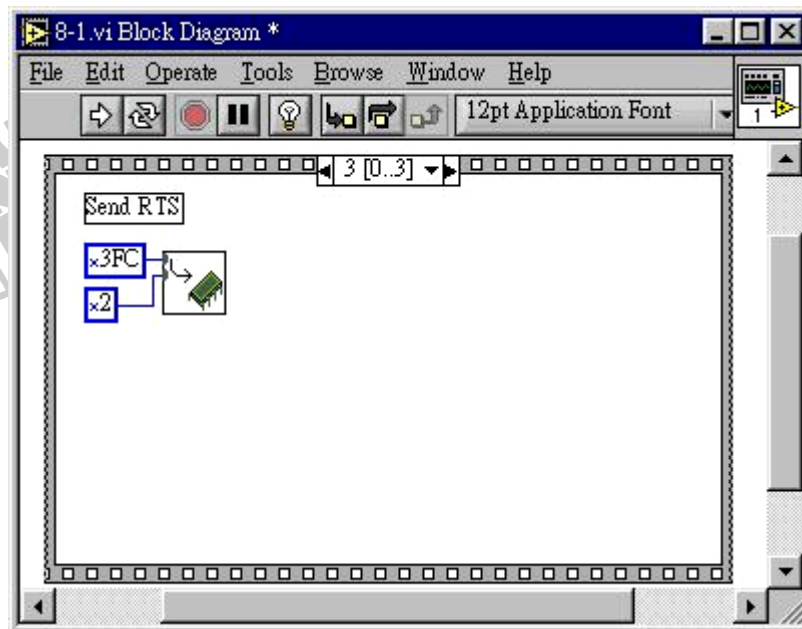
5、The responsibility of next page frame window is setting transmission rate of COM1 as 19200 bps. For the magnitude value of setting, please refer to the instruction of previous chapter.



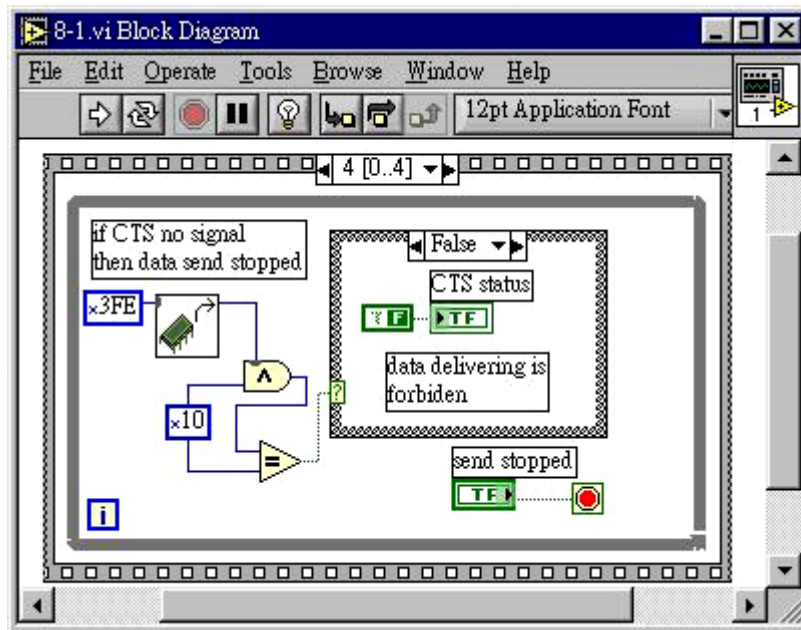
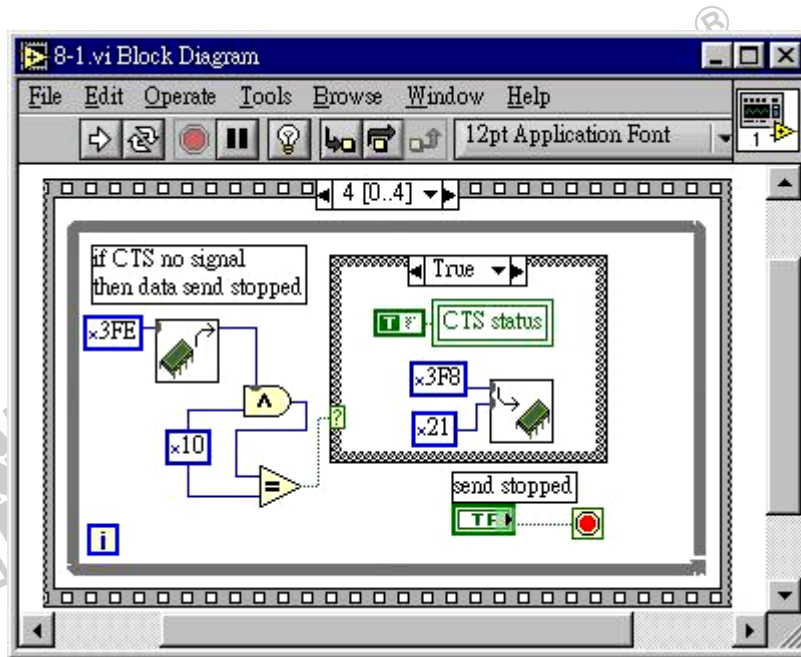
6、Then address $3F8_H$ 、 $3F9_H$ to ordinal address ($3F8_H$ for THR: transmitter holding register, $3F9_H$ for IER: interrupt enable register).Then set transmission protocol at the same time. **Break isn't set. Don't use parity check. Stop bit is set as 1. Width of data is set as 8 bit.** After comparing above-mentioned setting with bit function of register, you can find value in buffer is 03_H , so it's filled in directly.



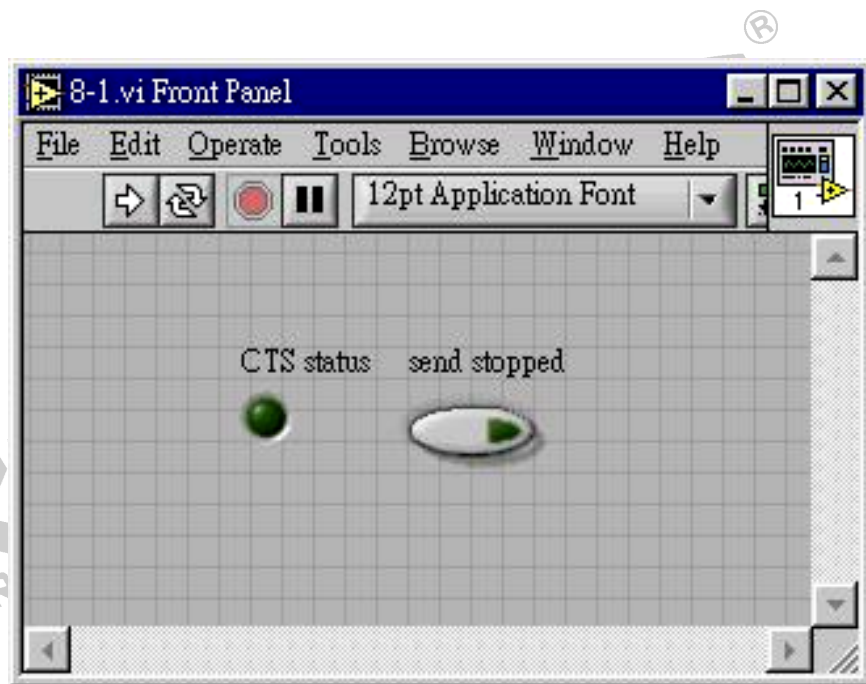
7、Then send RTS signal to tell 8051 that LabVIEW is ready to transmit data. RTS bit set in MCR (MODEM Control Register) is 1, so register is equal to 00000010_H which is equal to 2_H . Therefore the input value is set as 2. For more details, please refer to introduction of chapter 5.



8、This is the last page frame window. We want to check whether CTS signal exists here, so firstly we get CTS signal back for checking. CTS signal indicates bit is located at the 4th bit inside MSR (MODEM Status Register), therefore read it back and then operate the value with 00010000_B by AND operator. If the result is the same which means CTS to be 1, data delivering goes on (**Case is true**). On the other hand, it shows us CTS is 0 and data delivering is forbidden (**Case is false**). In the flow chart, you can find a while loop that keeps flow working on, and you can decide whether program is finished just by clicking a button. There is logical indicating light within construction of Case showing the status of CTS. The following diagram shows false status of Case.



9 、 It displays graphic user interface of flow control. When program runs, status indicating light of CTS will indicate status. When CTS is 1, LED will be lightened, and program will keep going unless stop button is pressed. After checking if there is no error, next you can connect 8051 circuit and start to execute program. At normal condition CTS status light will be lightening for one second and darkening for the other second.



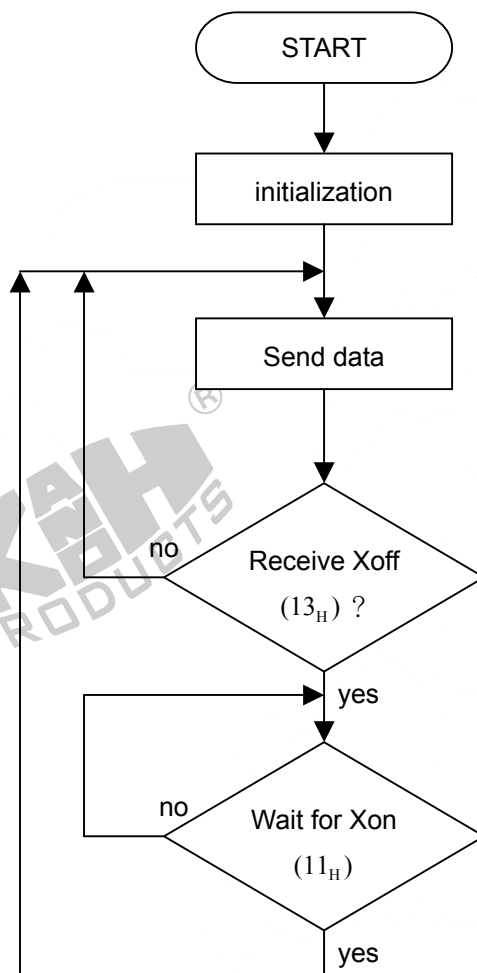
The end of practice 8-1

Practice 8-2 Software flow control

Objective

Use software flow control (Xon/Xoff) to exchange data with 8051 program

1. This is flow chart of 8051. Program for simulating Xon/Xoff flow control is simpler than RTS/CTS. Now it's LabVIEW's turn to ask flow control. When LabVIEW informs 8051 not to send data. 8051 will stop transmitting data unless it gets the command of Xon again.



2 、 8051 source code of program.

```
org      0h
jmp      init
org      30h

init:
    mov    sp,#30h
    call   set_bps
    clr    ri
    clr    ti
    mov    r0,#0h

start:
    mov    a,r0                ;Input data
    mov    sbuf,a
    jnb    ti,$
    inc    r0

    jnb    ri,$                ;If Xoff is received
    clr    ri
    mov    a,sbuf
    cjne   a,#13h,start

wait:
    jnb    ri,$                ;If Xon is received
    clr    ri
    mov    a,sbuf
    cjne   a,#11h,wait
    jmp    start                ;Return to loop

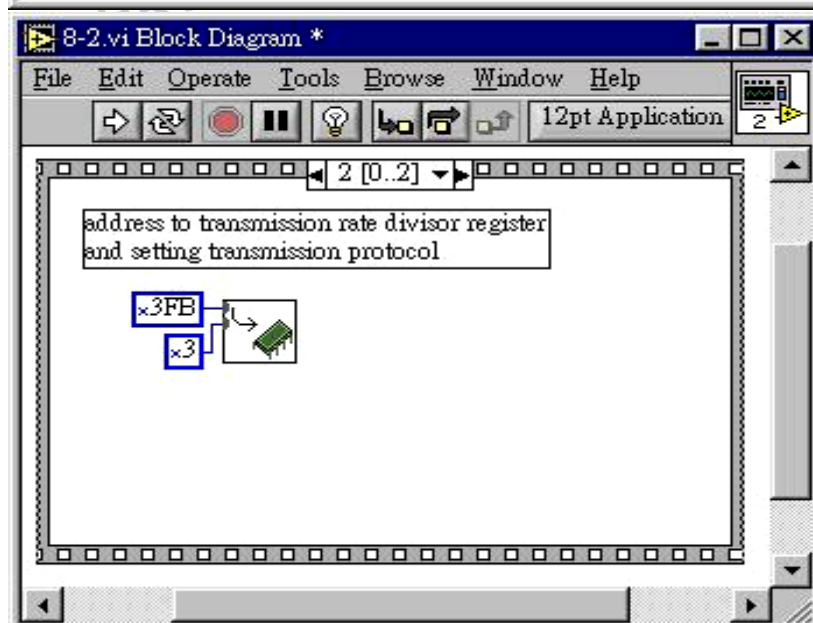
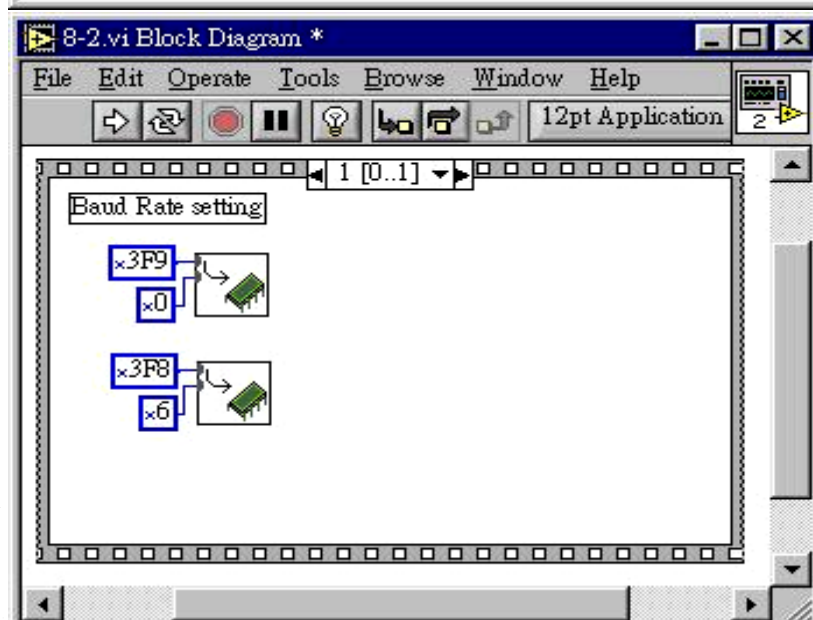
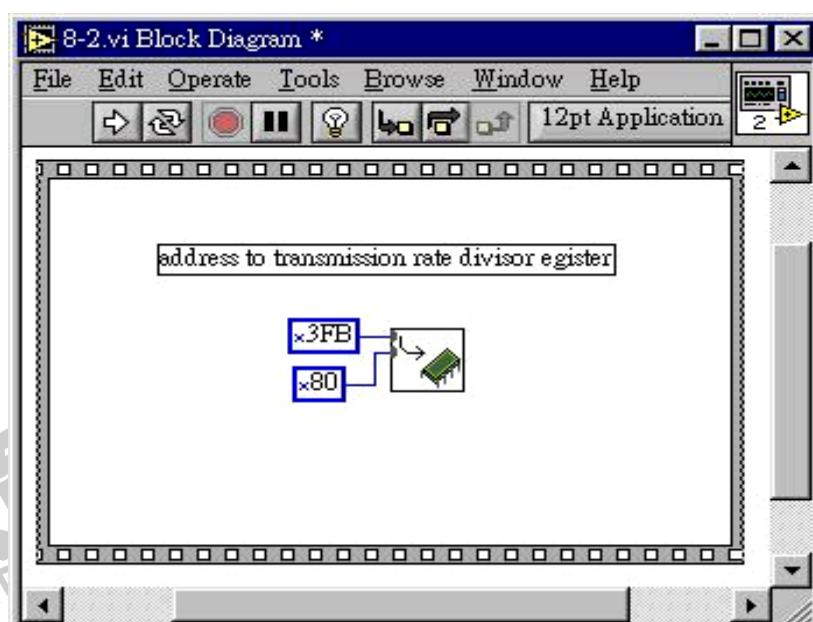
set_bps:
    mov    scon,#01010000B

    mov    tmod,#00100000B
    mov    th1,#FDH
    setb   tr1

    ret
```

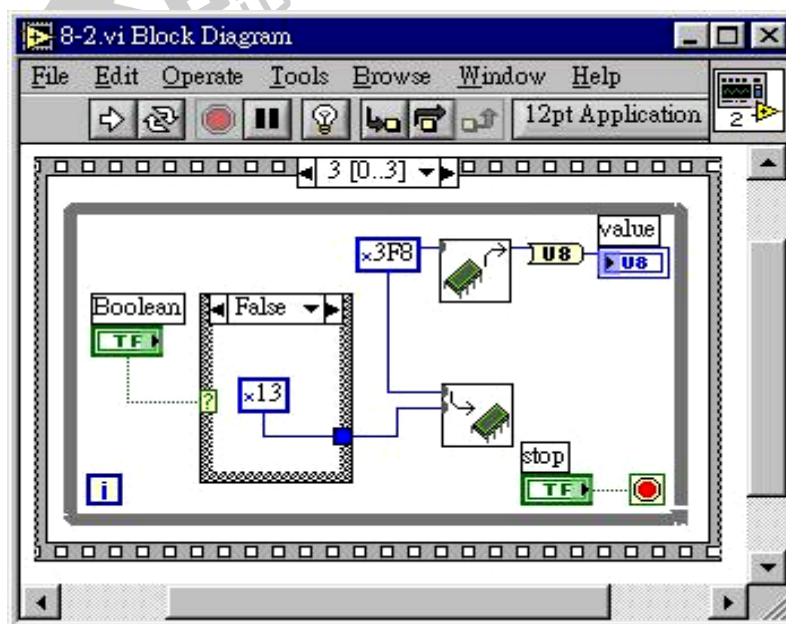
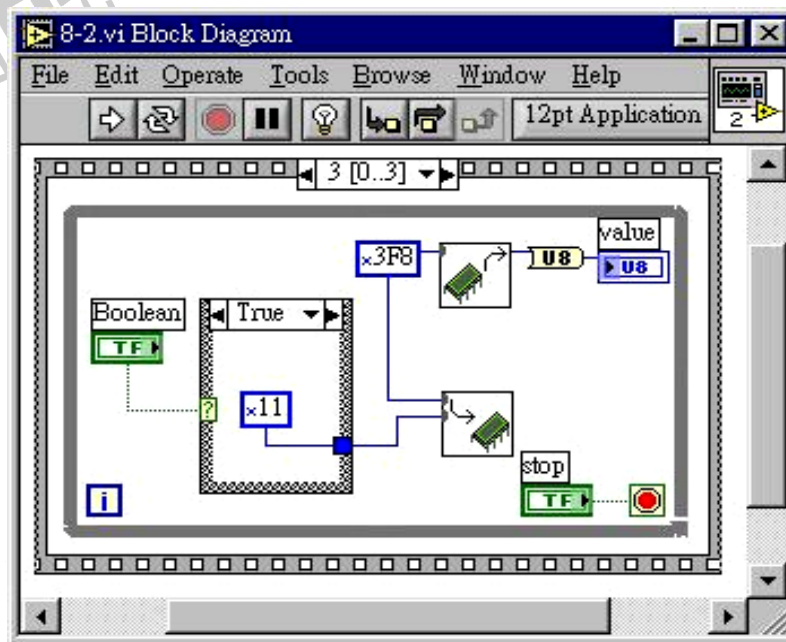
3 、 Program of LabVIEW is simpler, too. The following 3 steps is same as practice 8-1. It

is responsible for setting transmission rate and protocol.

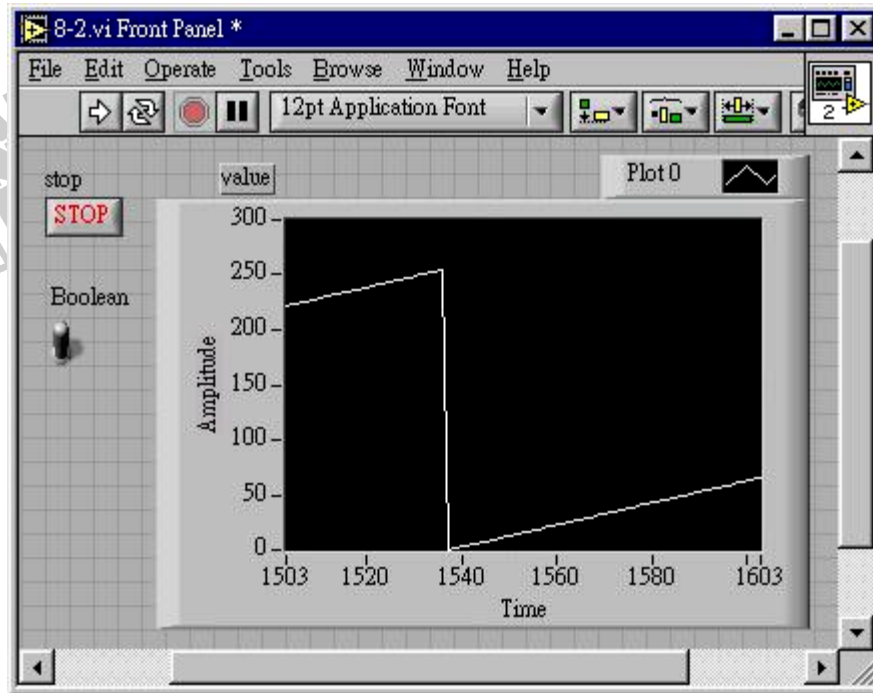


4、The next page frame 3 is the key point.

First please create a **While Loop**, then a **Case structure** is generated. The **True Case** puts into Xon 11_H, the **False Case** puts into Xoff 13_H, and using a software switch transfers cases between the two. In addition using “**In Port.vi**” is responsible for reading data from 8051, and using “**Waveform Chart**” displays data into the waveform. Another using “**Out Port.vi**” is responsible for sending variables that is selected from Case to 8051. The variables are Xon or Xoff.



5、The following window is the shape of the **Front Panel** of LabVIEW. This graph is the situation of normal transmission. When the switch is upward which means Xon, 8051 would keep on outputting the data, but if the switch is downward which means Xoff, 8051 stops outputting the data, and **Waveform Chart** did not display any waveform。



The end of practice 8-2

Exercise and discussion

- 1 、 The hardware flow control is just the single direction control of process, but how to do the mutual direction of the flow control? Try to find out more related materials, and write a flow control program of LabVIEW that can transmit data in mutual direction simultaneously 。
- 2 、 The method of flow control provided here has no debug function, and is there any other method of flow control that could attain this goal?

