AN1753
APPLICATION NOTE

SOFTWARE UART USING ST7 12-BIT AUTORELOAD TIMER

by Microcontroller Division Applications

INTRODUCTION

This application note describes a software implementation of a Universal Asynchronous Receiver/Transmitter (UART). This can be used on devices, like the ST7LITE0, with no on-chip SCI peripheral.

In this example, a software UART is implemented for the ST7FLITE0, using the 12-bit Autoreload timer and two I/O ports for asynchronous receive and transmit. The UART software provides the following features:

- Half-duplex operation
- Asynchronous operation
- Flexible data formats (7 or 8 data bits, 1 or 2 stop bits)
- Baudrate: 2400 to 19200 baud

To test this interrupt-driven software UART, you can use the “Hyperterminal” application running on a Windows PC.

The program code is quite small (357 bytes) and can easily be adapted to specific application requirements.

Rev. 1.0

Table of Contents

1 UART COMMUNICATION

The main features of a standard UART are summarized below.

1.1 MAIN FEATURES

The UART offers a flexible means of full-duplex data exchange

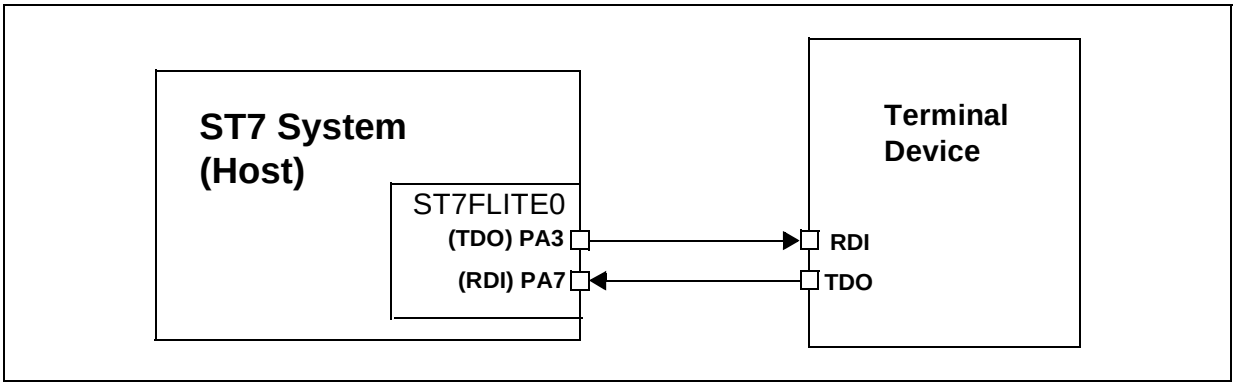
with external equipment requiring an industry standard NRZ asynchronous serial data format.

The UART allows a very wide range of baud rates and different baud rates for transmission and

reception.

In UART communication, a minimum of only two signals are needed, one for transmission and other for reception. No clock signal is needed as it works in asynchronous mode. Each device has to have a Transmit Data Output pin (the PA3 pin is used in our example for the ST7FLITE0) and a Receive Data Input pin (PA7 pin in our example). (Refer to Figure 1.)

Figure 1. ST7 UART Communication Set-Up



You must be very careful to identify the use of each pin. A simple method is to put the device in transmission and check with an oscilloscope if a transmission frame is present or not.

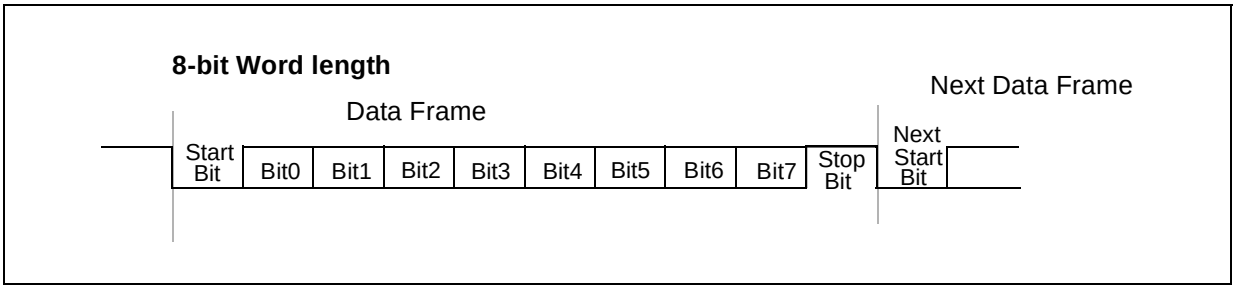
1.2 BAUD RATE

Transmission and reception can be driven by their own baud rate generator. However be aware that for correct communication, the receiver must have a reception baud rate strictly equal to the transmission baud rate of the transmitter. **If not, the communication will be corrupted**. As long as this condition is met, a wide range of baud rates is possible.

1.3 FRAME

Any transmission is Least Significant Bit first. A data word is usually 8 bits long. A data frame begins with a «start bit», which is a ‘0‘ bit and ends with a «stop bit», which is a ‘1‘ bit. See Figure 2.

Figure 2. Frames



In some cases, a 9th bit can be used, as a parity bit or as a second stop bit.

2 RS232 COMMUNICATION WITH A PC

2.1 MAIN FEATURES

Electrical and protocol characteristics of RS232 are different from those used by the UART. In RS232 communication, high level is typically +7V and low level is typically -7V, while the ST7 I/Os work at CMOS levels (0, +5V).

Furthermore, the polarities are different. A ‘1‘ bit coming from the UART corresponds to a ‘0‘ bit

in RS232, and a ‘0‘ bit to a ‘1‘ bit. It is true for all bits including the START and STOP bits. So it is necessary to implement a conversion between the PC and the ST7. In the application, a MAX232 is used for this purpose.

2.2 PC CONFIGURATION

The PC will be used as a terminal interface. “Hyperteminal”, the terminal application software is used as a interfacing software to test the functionality [.zip file attached]. The test environment is Windows 2000.

Under Windows, open the Hyperterminal application. To configure it, go to the port settings and set the parameters to your application requirements. The options in this window must be the same as the ones defined for your ST7FLITE0, communication device, except the port.

After selecting the right serial communication port, select the same baud rate as the one set for the ST7. As the PC accepts only one baud rate, transmission and reception baud rates will have the same value. Data word can be 8/7 bits, but you can choose to use 1 or 2 stop bits. «Flow control» can be either Xon/Xoff or none. The PC is then correctly configured.

3 ST7FLITE0 CONFIGURATION

3.1 CLOCK SOURCE

This application is implemented using ST7FLITE0 device with an 8 MHz internal clock. PLL * 8 is used to generate this 8 MHz clock.

3.2 INPUT INITIALIZATION

Two pins of the ST7FLITE0 are used:

- PA3: Pin of PortA

- PA7: Pin of PortA with interrupt

Pin PA3 is normal Input/Output port pin with no alternate function, used for transmission.

During initialization it is configured as an output.

Pin PA7 is normal Input/Output port pin with no alternate function, used for data receive.

During initialization it is configured as an input. While in receive mode, at start, same pin is used with interrupt enabled (“ei1”) to sense “START” bit. So this pin is configured with “pull up interrupt input” by setting PADDR to 0 & PAOR to 1. And to set interrupt sensitivity “Falling edge only” we set IS11=1 & IS10=0 in the EICR register.

Refer to the device datasheet for detailed description of the I/O and interrupt control registers.

3.3 AUTO-RELOAD TIMER REGISTER CONFIGURATION

The AT timer is based on free running 12-bit up counter with 12-bit auto reload register (ATR). Apart from this it also includes other functionality like, PWM signal generator & Output Compare Function & etc.

We are using “Output Compare” functionality for this application. To use this function, the OE bit must be 0, otherwise the compare is done with the shadow register instead of the DCRx register. Software must then write a 12-bit value in the DCR0H and DCR0L registers. When the 12-bit upcounter (CNTR) reaches the value stored in the DCR0H and DCR0L registers, the CMPF0 bit in the PWM0CSR register is set and an interrupt request is generated if the CMPIE bit is set.

The registers that are used in application note are:

TIMER CONTROL STATUS REGISTER (ATCSR):

0	0	0	CK1	CK0	OVF	OVFIE	CMPIE

- CK1 & CK0: They select the clock frequency of the counter.

For fcounter = fcpu, set CK1=1 & CK0=0,

- CMPIE: It allows to mask the interrupt generation when CMPF bit is set.

When it is 0: CMPF interrupt disabled

1: CMPF interrupt enabled

PWM0 CONTROL/STATUS REGISTER (PWM0CSR):

0	0	0	0	0	0	OP0	CMPF0

- CMPF0:

It indicates that the upcounter value matches the DCR register value. When it is,

0: Upcounter value does not match DCR value.

1: Upcounter value matches DCR value.

PWM OUTPUT CONTROL REGISTER (PWMCR):

0	0	0	0	0	0	0	OE0

- OE0: When set to 1, PWM0 output enabled

Timer Initialization:

Write to DCRx registers with required value. Then Output Compare functionality must be enabled. To do this, reset the OE bit in the PWMCR register. This will disable PWM output.

To configure fcounter = fcpu, set CK1=1 & CK0=0, and to enable the CMPF interrupt, set CMPIE=1 in the ATCSR register.

4 UART IMPLEMENTATION

4.1 BAUD RATE DEFINITION

The Autoreload timer is used to generate the baud rate. Autoreload timer clock (fCOUNTER) is same as fcpu clock, which is 8 MHz internal clock.

The description here below is the example to show how this baud is generated:

Example:

- baudrate = 2400

- fcpu = 8MHz (so clock period is 125 ns)

- In the application fCOUNTER = fcpu

So to get 2400 baud the “Prescaler” required is,

Prescaler = fcpu / baud

Prescaler = $8 * 10^6 / 2400$

= $0.0033333333 * 10^6$

= 3333.33 decimal

= D05 Hex (= 1 bit delay count)

So in software DCRx will be

DCR0H = 0D + CNTRH

DCR0L = 05 + CNTRL

to generate one bit delay when 2400 baud is required

For half bit period “Prescaler” is,

Prescaler = one bit / 2

= 1666.66 (decimal)

= 682 Hex (= 1/2 bit delay)

DCR0H = 06 + CNTRH

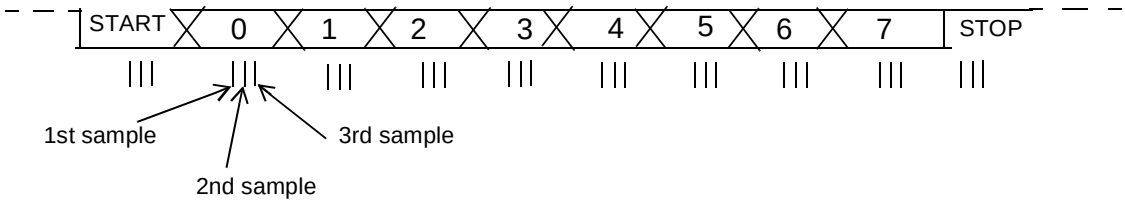
DCR0L = 82 + CNTRL

The software provided with this application note has been functionally tested in the range 2400 to 19200 baud.

4.2 MAJORITY VOTING SYSTEM

The receive section is complex compared to the transmit section. So, to avoid possible errors in detecting exact bit status because of line noise etc, this program uses a majority-voting system. When the ST7 is in receive mode, it reads (samples) each bit three times at the middle of the bit time. To determine its exact state, it compares the number of 1's with the number of 0's. If there are more '1's than '0's, the bit received is a '1', else it is a '0'. (see Figure 3.)

Figure 3. Majority voting system



This section is only used for the receive part. You can change it to suit your requirements.

4.3 STATUS HANDLING

To keep track of status during program execution, software uses variable, "sci_status". Bits 0-4 of this variable are used to hold the system status.

			TE	BS	RE	BR	SP

Definition of each flag:

SP: Reception mode sampling phase

BR: Byte Received flag

RE: Reception Enable

BS: Byte Sent flag

TE: Transmission Enable

4.4 TRANSMIT & RECEIVE IMPLEMENTATION

Transmission mode gets executed after power on. To transmit data, the program uses port pin PA3 in output mode with “Output Compare” interrupt. One bit is transmitted during each interrupt, generated after a bit delay when the 12-bit upcounter (CNTR) reaches the value (=1 bit delay) stored in the DCR0H and DCR0L registers. The value for duty cycle resister (DCR0L/ DCR0H) depends on the required baud rate & clock frequency.

For receive mode, the program uses port pin PA7 in input mode along with its interrupt & output compare functionality. To sense START bit, interrupt “ei1” is used, configured as “Falling edge only”. In this interrupt routine, program sets the ATR - output compare value for half* bit delay & disable “ei1” interrupt to use PA7 in normal input mode to receive entire frame. This half bit delay is used to sample bit at middle of bit. After receiving START bit in output compare interrupt, the same interrupt routine gets executed each time a bit must be received. To receive each bit correctly majority voting system is implemented in this interrupt routine.

The interrupt strategy used in software for transmit & receive allows to have other applications to work at the same time.

Note:

– Error handling**:** This program does not handle communication errors (e.g. Frame Errors).

– No handshaking is implemented.

– For better performance at high baud rates it is recommended to use character transmission rather than string transmission.

*User can adjust this value depending on baud rate.

5 HARDWARE SETUP

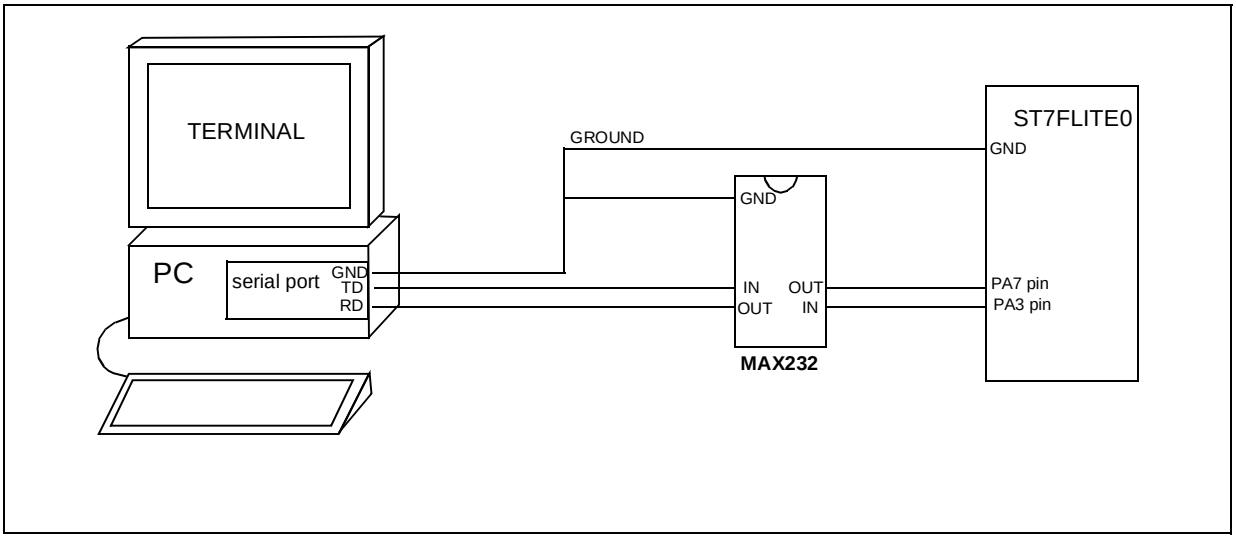
A general system configuration is shown in Figure 1. The ST7FLITE0 is connected to an RS-232 line driver chip which is in turn connected to any RS-232-compatible device. The RS-232 line driver is needed to convert the 5-volt logic level of the ST7FLITE0 to the proper RS-232 line voltages, and vice versa.

Figure 4. shows a specific example of a hardware setup required for UART in which the ST7FLITE0 processor and RS-232 Driver/ Receiver are used. You can use different driver/receiver in your application. For example you can use a MC14C88/ 89 as a low cost solution.

The Receive Data pin (RD) of the serial port of the PC must correspond to the PA3 pin of the ST7FLITE0, and the Transmit Data pin (TD) to PA7 pin.

Be sure that the three main devices (PC, ST7, MAX232) have the same electrical reference (GND). For a detailed description of MAX232, please refer to the datasheet.

Figure 4. Test Setup



6 FUNCTIONAL SOFTWARE FLOW

The character format used is “ASCII”. Before starting communication you should configure the communication parameters to your requirements. You can select the baudrate, number of bits and STOP bits in different combinations. However, for correct communication, the receiver must have a reception baud rate strictly equal to the transmission

baud rate of the transmitter. If not, the communication will be corrupted. So you have to con figure “Hyperterminal” correctly.

To configure the device, you must set del_1bh, del_1bl, del_sampl, del_samph, STOP_BIT and TxRx_data_lnth correctly in your software.

After power on, the ST7FLITE0 goes in to transmit mode. It initiates communication by transmitting character “$” in ASCII format & then goes into receive mode. The PC then receives this character on its serial port, which can be viewed in the Hyperterminal window.

Now you have to press a key (character) to transmit. After pressing a character key, it gets transmitted and the ST7 receives it. The ST7 then sends same received character to PC. This sequence can be repeated continuously (Refer to the flow charts below).

6.1 MAIN SOFTWARE ROUTINES

The UART software consists of main 4 routines:

– Tx_data

– Rx_data

– EI1_Interrupt

– OPCOMP_int

– **Tx_data:** This routine is used to transmit data in bit format stored in variable “TX_byte”. It also takes care of synchronization & the output compare interrupt. The AT timer is initialized to the application configuration inside this routine.

– **Rx_data:** This routine is used to receive data on the input pin in bit format & stores it in “RX_byte”. To detect a “START” bit it initializes the PA7 pin in interrupt mode.

– **EI1_Interrupt:** This “ei1” interrupt routine gets executed only once during receive data when PA7 receives a “START” (High to Low edge) bit. It also initiates the AT timer for half bit duration.

– **OPCOMP_int:** This routine gets executed in both transmit & receive mode. During Tx_dataroutine execution this gets executed for each data bit transfer. And it is actually used to transmit data on the TxD (PA3) pin along with START & STOP bits.

During receive mode it gets executed for each data bit received including the START & STOP bits.

The majority-voting system is implemented in the same section of code.

6.2 SOFTWARE FLOW CHARTS

The flowcharts below represent the application program flow.

Figure 5. Main Flowchart

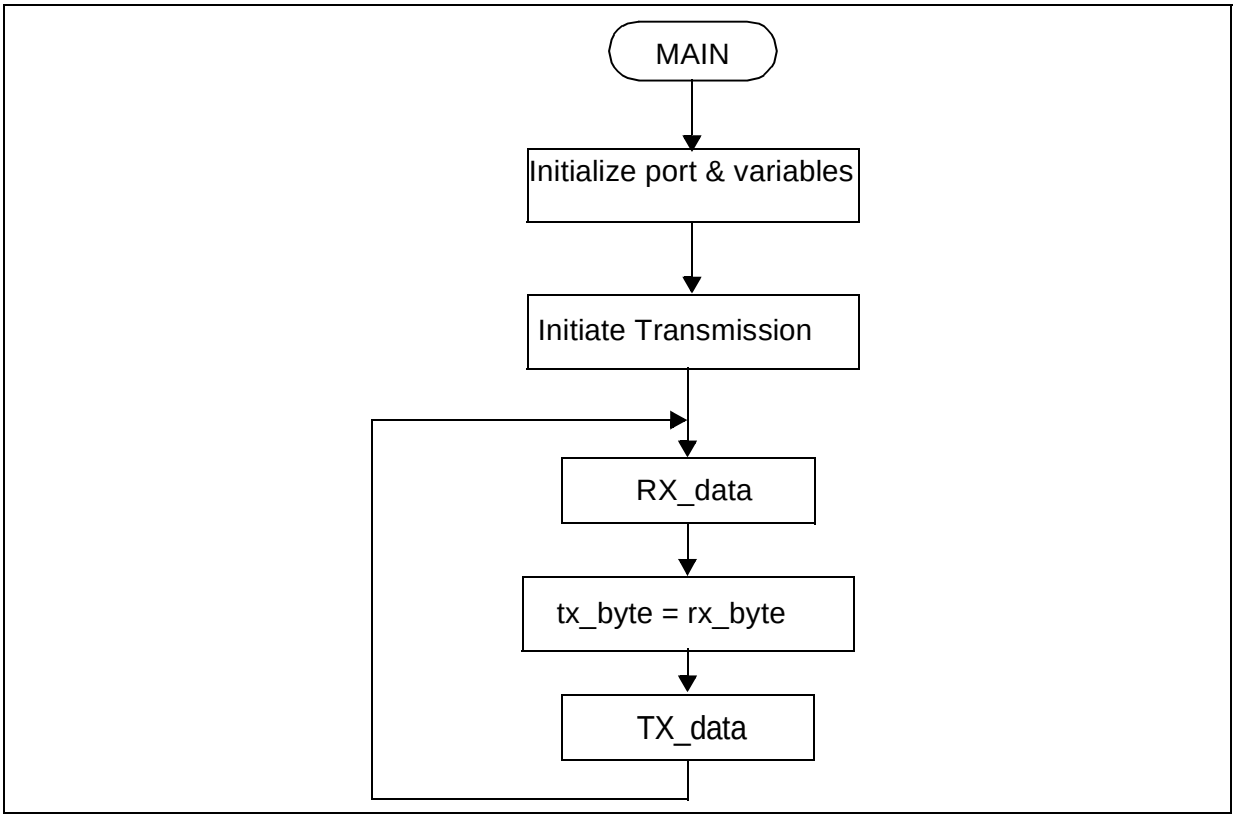


Figure 6. Output Compare Interrupt

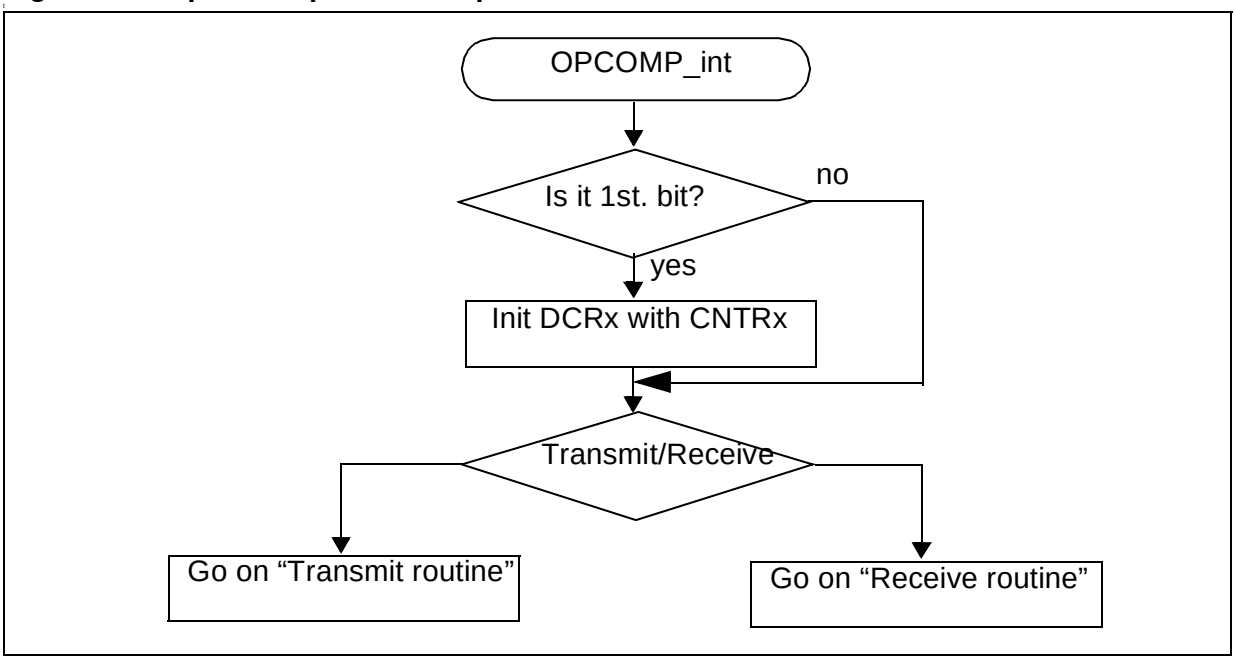


Figure 7. Transmit routine

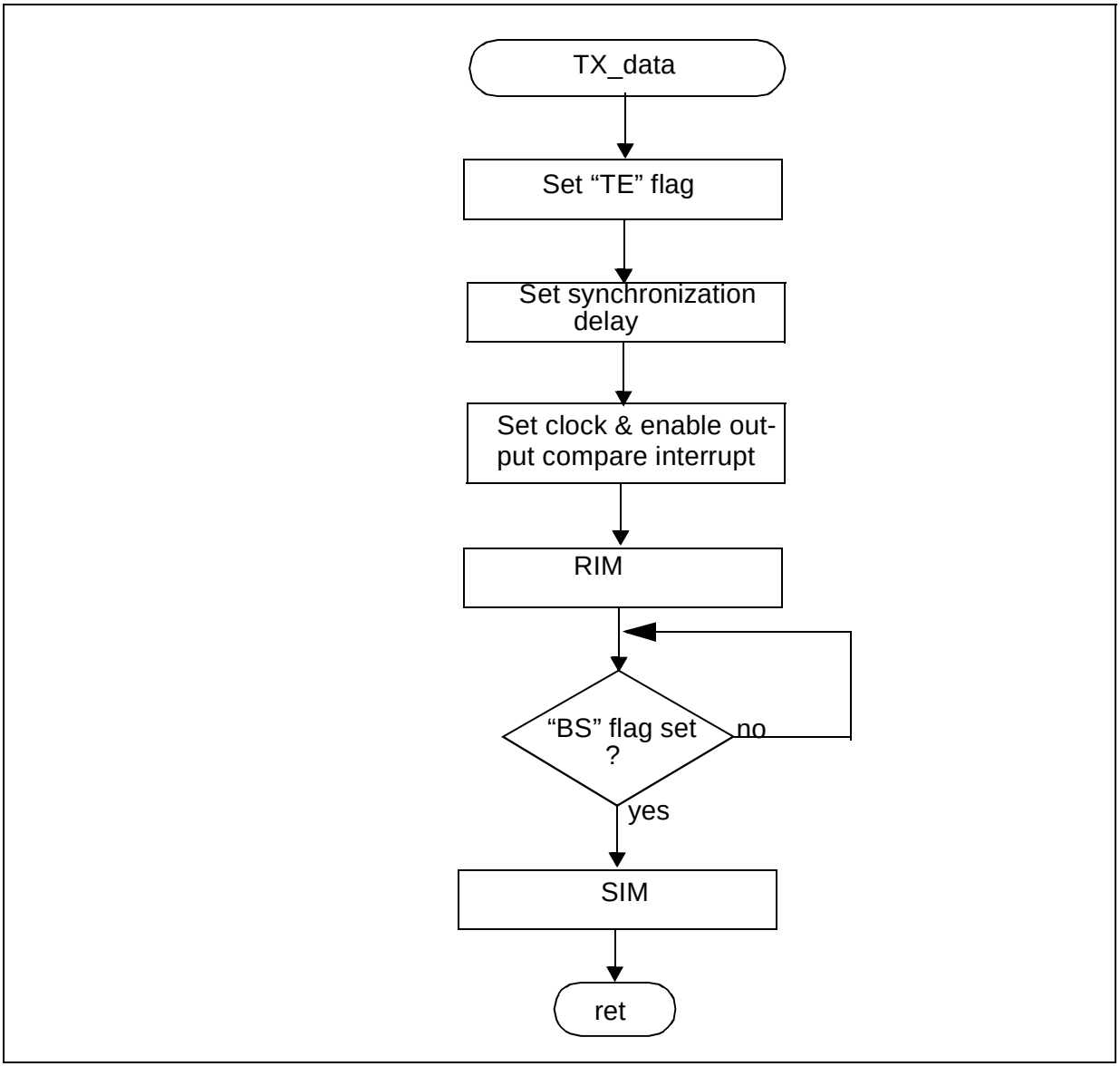


Figure 8. Receive data routine

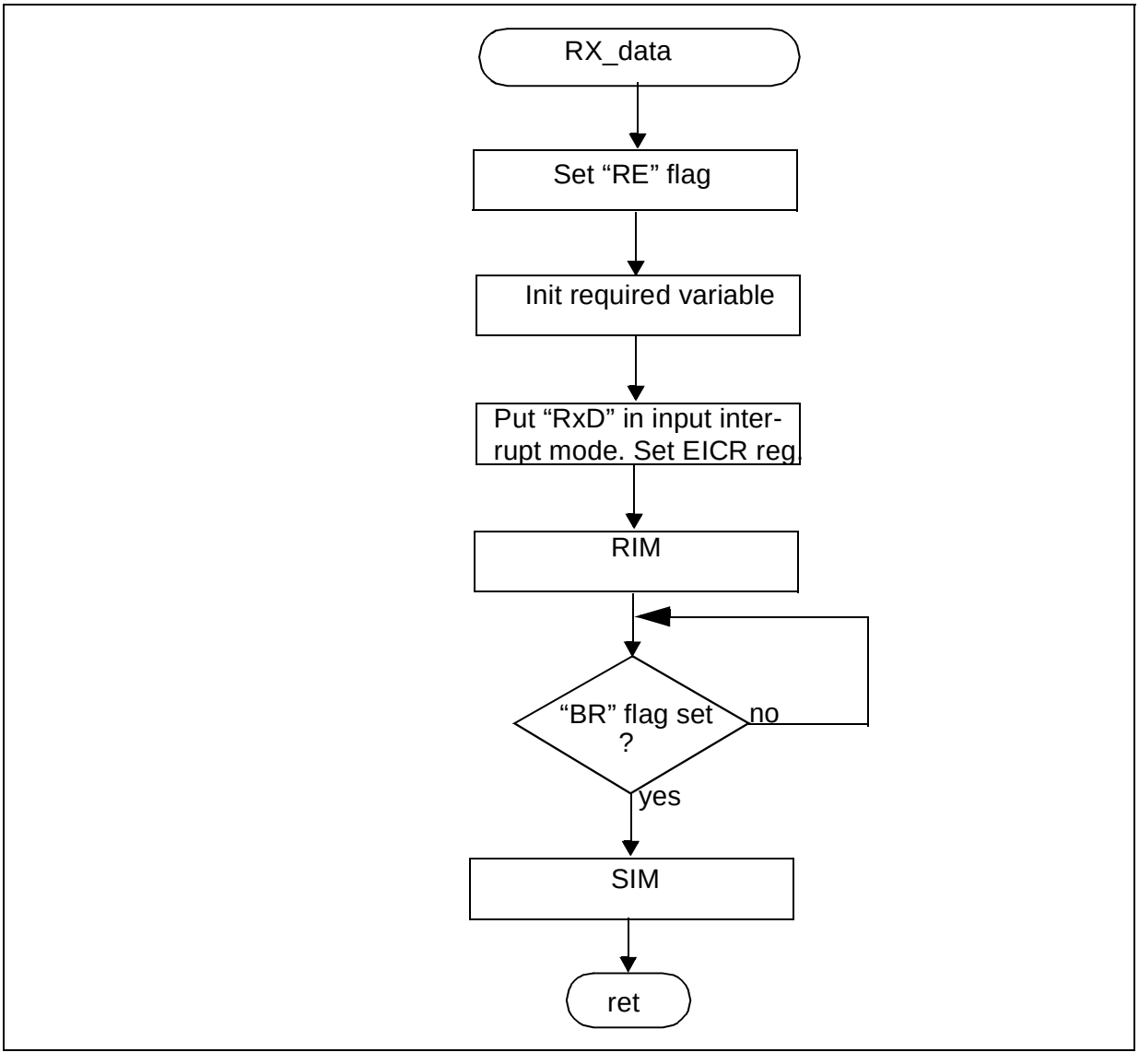


Figure 9. Output Compare Transmit flow

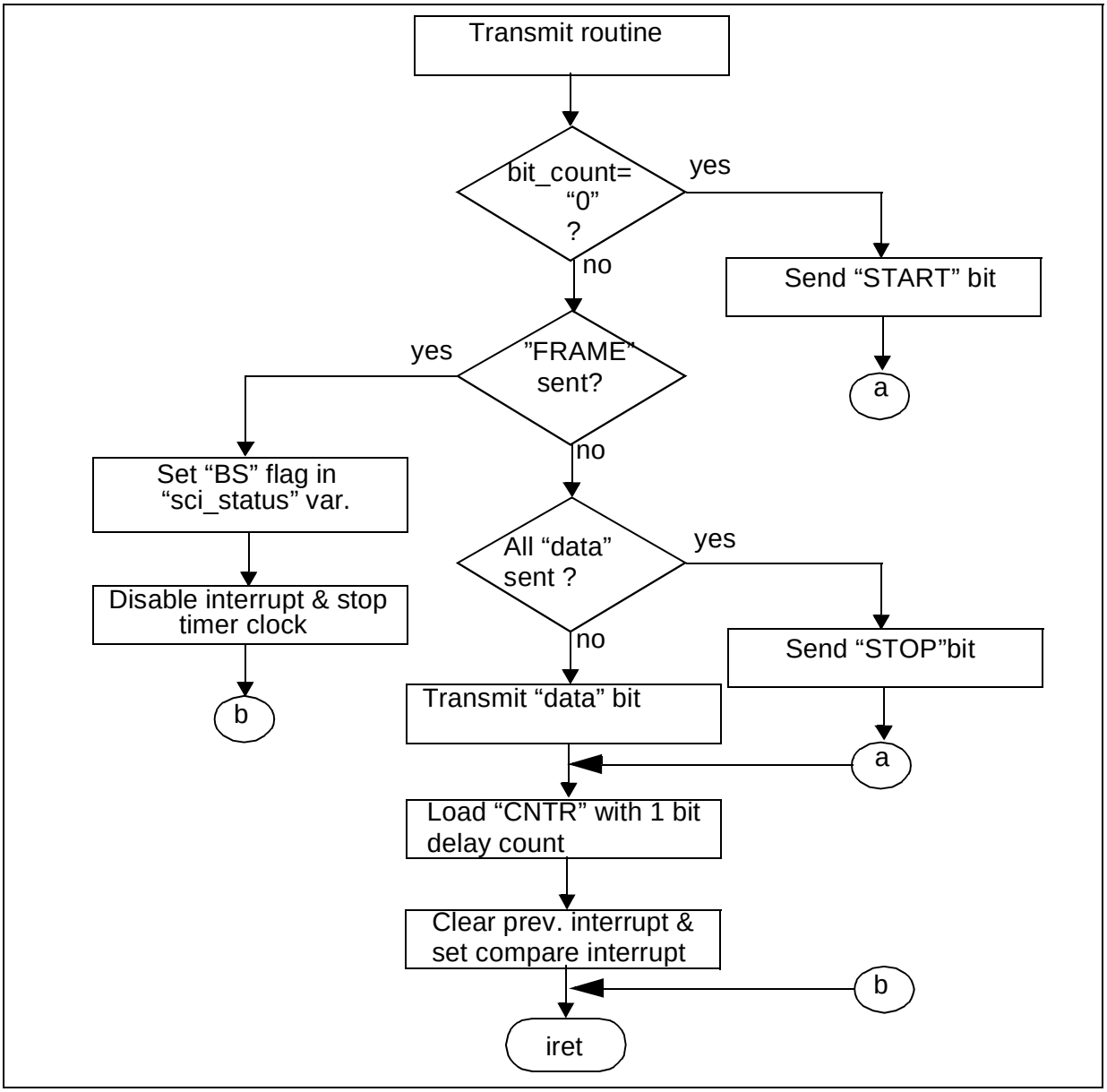


Figure 10. Output Compare Receive Flow

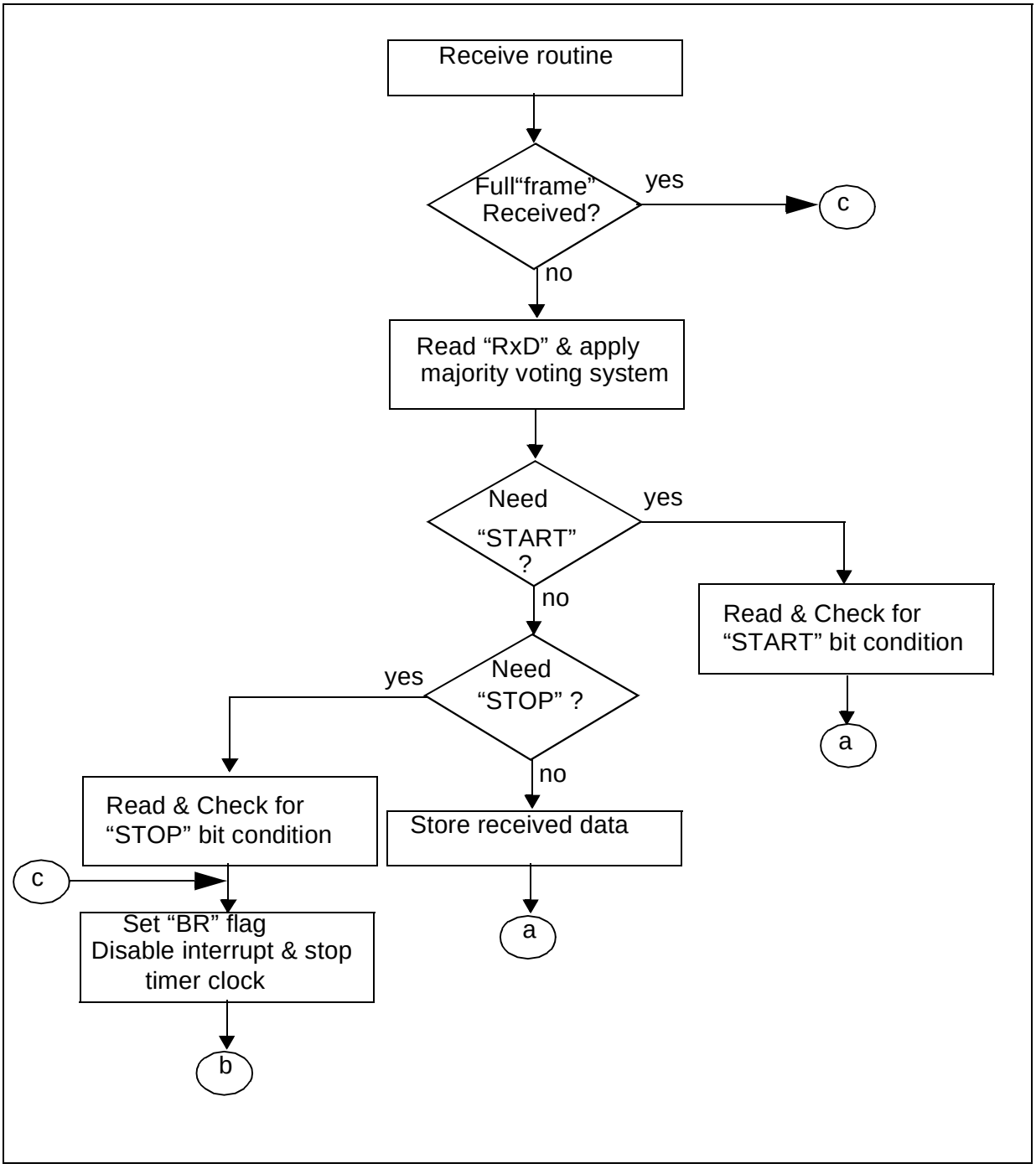
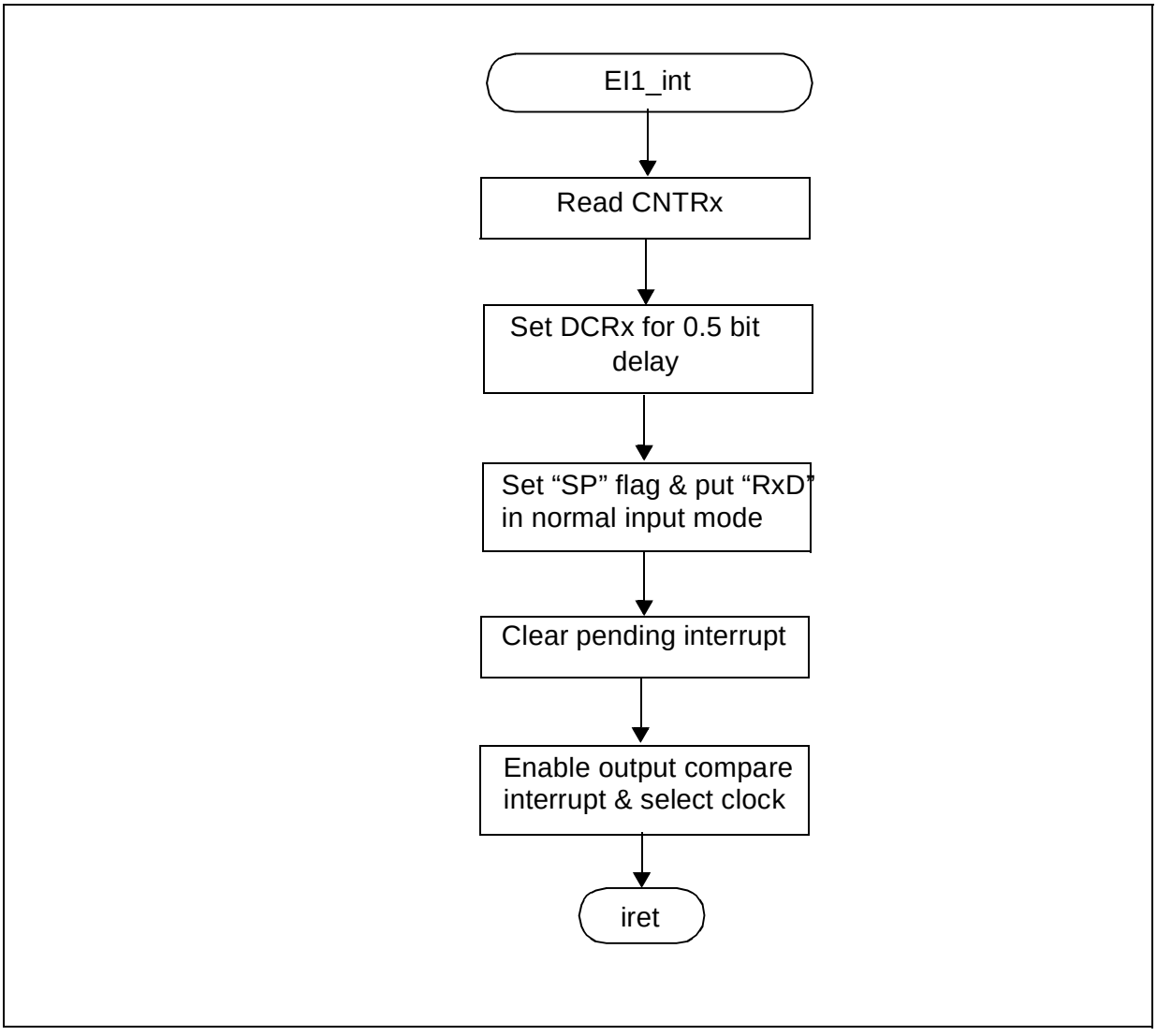


Figure 11. EI1 Interrupt routine



7 TEST PROCEDURE

– Connect both the communication & terminal equipment as shown in fig.4

– Open & configure “Hyperterminal “application [Port, Baud rate & etc.].

– Power on ST7

– On the terminal you should get “$” as a starting character (If not then check link between terminal and the ST7 and check the communication settings)

– Then press a key to send a character

– The same character should be returned by the ST7 (check “Hyperterminal” window)

– Continue by repeating the previous two steps

8 SOFTWARE

The complete software can be found on the ST internet website in zipped file format. It is intended for use only as an example. It is up to you to adapt it to your specific application.

The source file is for guidance only. STMicroelectronics shall not be held liable for any direct, indirect or consequential damages with respect to any claims arising from use of this software.