Protean LOGIC Inc.
11170 Flatiron Dr.
Lafayette, CO 80026
303.828.9156 (voice)
303.828.9316 (fax)
sales@protean-logic.com

Product Data Sheet

RSB509C-0 or RSB509C-1

32 byte RS232 Serial Data Buffer IC

Overview

The RSB509C is an easy to use, low cost, data buffer IC. RS232 serial format data is received at the input pin and stored inside the RSB509C until the host signals that it is ready to process the data. The data can then be processed one byte at a time, or the entire buffer contents can be sent to the host.

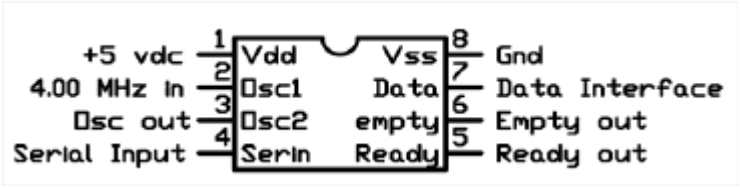
The RSB509C allows the host processor to perform other tasks independent of the timing of incoming RS232 data. This allows single thread processors, like Parallax's Basic STAMP, to receive up to 32 bytes of data in background. More sophisticated processors also benefit from the use of the RSB509C when multiple intermittent input streams need to be handled.

The RSB509C is easily interfaced to a host via a single bi-directional I/O pin. An RSB509-C0 and RSB509C-1 can share a single host interface. Each device is addressed independently during configuration. Device Configuration of the RSB509C is also accomplished through this single I/O pin. A pulse on the interface pin signals the RSB509C to send data contained in its buffer. If the pulse is longer than 1200us, the RSB509C prepares to receive one or two configuration bytes from the host. Using this simple pulse protocol, received data bytes can be individually retrieved and processed.

Features

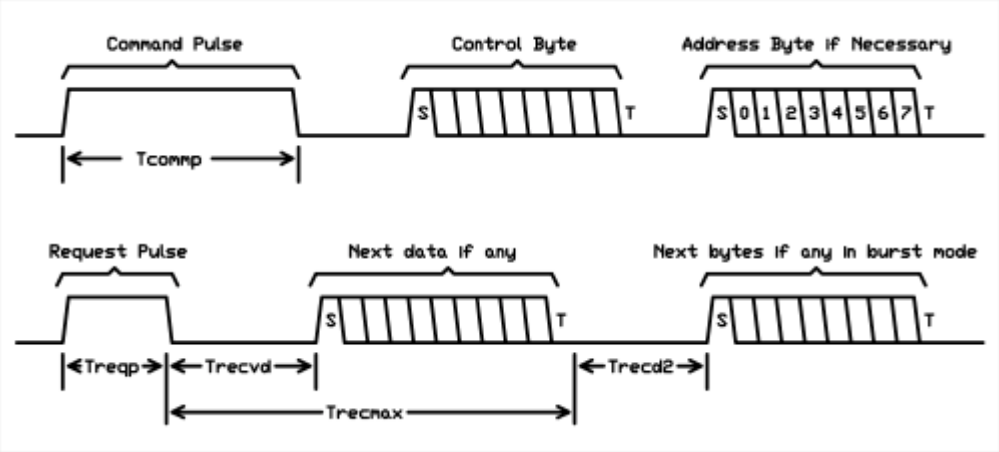
- Small 8-pin DIP or SOIC package.
- Only a 4MHz crystal or clock source, a diode, and a few resistors are required for operation.
- 32 byte buffer receives data and retransmits it to the host when requested.
- Only one host I/O pin is required to interface to two RSB509BC (provided they have different address).
- A simple pulse protocol is used to configure the RSB509C and to gate the output bytes.
- Programmable input baud rates of 9600, 4800, 2400, or 1200 baud. Lower rates possible with slower crystal.
- Input stream can be either inverted (direct connect to RS232 levels) or true (connection to signal receiver like MAX232 or RS1489).
- Communication with host is fixed at 9600 baud, inverted.
- Pulse protocol compatible with Basic STAMP's SERIN command.
- Packet modes for implementing simple multi-drop addressed serial networks
- Buffer Empty output pin for status or handshaking.
- Buffer Full output pin for status or handshaking.

RSB509C Pin Assignments



1. Vdd - +5 vdc power supply for the device
2. Osc1 - connect to a crystal or to a 4MHz square wave oscillator
3. Osc2 - connect to a crystal or leave open when an oscillator is used
4. Serin - RS232 input. Connect to an RS232 receiver or connect to an RS232 source via a 22K resistor, when connecting to an RS232 source, this pin must be over voltage protected by placing a diode from this pin to Vdd.
5. Ready - This open source output indicates that the buffer is not full. This line must be pulled low.
6. Empty - This open source output indicates that the buffer is empty. This line must be pulled low.
7. Data - This open source directional pin is connected to the host processor. This line must be pulled low.
8. Gnd - Ground connection for the device

Configuration Information



Operational Timing Parameters: Condition: Osc=4MHz

Parameter		Min	Max
Tcommp	Duration of Command Initiate Pulse	1200 us	-
Treqp	Data Retrieve Initiate Pulse	52 us	1000 us
Trecvd	Delay from request pulse to start bit of data if any data is present within the buffer.	52 us	78 us
Trecd2	Delay between subsequent data bytes if any subsequent data is present in a burst mode transmission.	52 us	52 us
Trecmax	Maximum delay before complete transmission of data after initial request pulse	1092 us	1118 us

When the RSB509C is first powered up, it is idle waiting to be configured. Configuration is accomplished by pulling the data interface pin high for a minimum of 1200 us. The host then lowers the interface pin for a minimum of 52 us then sends two bytes of configuration data. The first byte is an address byte used only for packet addressing but required as a place holder in the configuration sequence. The second byte is the control byte used for setting input polarity, baud rate, reception modes, and host interface mode.

The address byte is used only when the RSB509C is in an addressed packet reception mode. When in this mode, The RSB509C will not buffer data until the address match criteria have been met. The RSB509C will then buffer data until it is reconfigured. An address byte should only be sent for packet configurations.

The configuration byte is a bit mapped register with the meaning of the bits described in the following table:

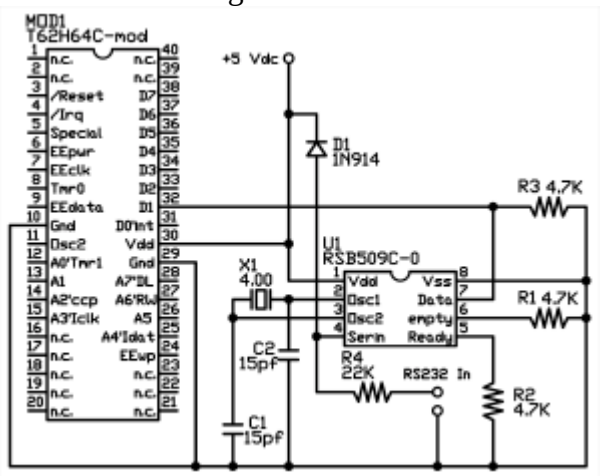
Bit 0	High = This configuration command is addressed to a router. Low = Address an endpoint device, like an RSB509C-0 or RSB509C-1
Bit 1	High = Configure an RSB509C-1 if it exists on the wire Low = Configure an RSB509C-0 if it exists on the wire
Bit 2	High = Input is inverted, open drain for direct connection to RS232 cable levels Low = Input is true, full totem for connection to an RS232 receiver buffer IC like the MAX232
Bit 3	Input Baud rate setting: 00 is 9600, 01 is 4800, 10 is 2400, 11 is 1200
Bit 4	
Bit 5	Reception mode setting: 00 is normal byte oriented mode. 01 is address match packet mode 10 is break delineated address match packet mode.
Bit 6	
Bit 7	High = Request pulse causes sequential send of entire buffer contents Low = Request pulse causes send of the next data byte only.

After the configuration bytes, the RSB509C clears it's buffer and prepares for incoming data while monitoring the interface pin for a configuration or a request pulse. If no data is in the buffer when a request pulse is detected, the pulse is simply ignored. The host can time out while waiting for data after 78us. This allows the host to poll the RSB509B for data.

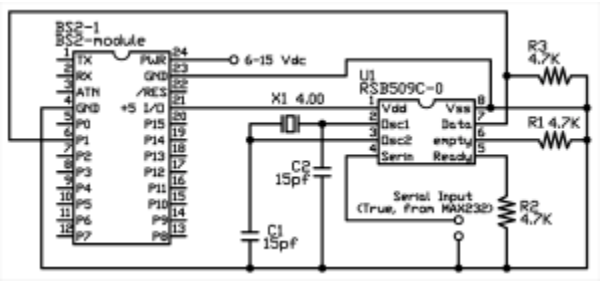
Example Connection Diagrams

The host used is not significant to either RS232 input connection, but is shown for reference.

TICkit 62 using "inverted" direct RS232



Basic STAMP II using "true" RS232 receivers



Sample Programs to Configure and Receive Data from an RSB509C

TICkit 63 Sample

```
; Program for RSB509C serial buffering
DEF tic62_c
LIB fbasic.lib
LIB rsb509c.lib

FUNC none main
BEGIN
   ; generate a long pulse on interface pin to
   ; signal initialization to RSB509
   pin_high( pin_d0 )
   delay( 2 )
   =( in_err, pin_in( pin_d0 ))

   rs_param_set( rs_invert | rs_9600 | pin_d0 )

   ; serial input is inverted and 9600 baud.
   rs_send( rsb509_invert | rsb509_baud1 )

   ; no match byte used but one could be sent
   ; rs_send( ' ' )

   ; wait for RSB509 to reset.
  delay( 1 )
   REPEAT
      ; generate a quick pulse for each
    ; byte to be read. pin high creates start
      ; of pulse.
    ; rs_receive creates end of pulse
      ; when it makes the pin an input.
      pin_high( pin_d1 )

      ; pin D1 will be used for input
      rs_param_set( rs_invert |~
     ~ rs_9600 | pin_d0 )
      ; read each byte with a short wait
      =( in_val, rs_receive( 100, 0b, in_err ))
      IF in_err
      ELSE
         rs_param_set( debug_pin )
         con_out_char( in_val )
      ENDIF
   LOOP
ENDFUN
```

Basic Stamp II Sample

```
' sample program for RSB509C on STAMP II
inval VAR byte

' generate a 2 ms pulse to signal
' initialization
HIGH 1
PAUSE 2
INPUT 1


' pin P1 is used for input.
' we are using open, inverted, 9600,
' 8N1 format.

' pin P1 is used for input.
' we are using open, inverted, 9600,
' 8N1 format
' serial input is inverted and 9600 baud
SEROUT 1, 49236, [%00000100]

' No match byte is used
' but we could send a byte here
' during initialization
' SEROUT 1, 49236, [" "]

' wait for RSB509 to reset
PAUSE 1

around:
' generate a quick pulse for each byte
' to be read. HIGH creates the start of
' the pulse. SERIN creates the end of the
' pulse when it makes the pin an input
' because the interface pin is pulled low
HIGH 1

' pin P1 is used for input.
' we are using open, inverted, 9600,
' 8N1 format. If a byte is not received within
' 1 ms branch to "around" to poll again.
' data will only be displayed when
' byte is received
SERIN 1, 16468, 1, around, [inval]

' show character received on debug window
DEBUG inval

GOTO around
END
```