APPLICATION NOTE

The SNERT bus specification

AN95127

Philips
Semiconductors

PHILIPS

The SNERT bus specification

Application Note AN95127

Abstract

The SNERT bus is a serial bus used for communication between a micro controller and ICs offering a SNERT interface. It is a single master bus, uses three wires and runs at baud rates of about 1 MBit/s.

SNERT stands for Synchronous No parity Eight bit Reception and Transmission*.*

APPLICATION NOTE

The SNERT bus specification

AN95127

Author:

Heinrich Waterholter
Product Concept & Application Laboratory Hamburg, Germany

Keywords

SNERT bus, serial bus

Date: January 31, 1996

Summary

This report describes and specifies the SNERT bus. SNERT stands for *Synchronous No parity Eight bit Reception and Transmission*.

The SNERT bus is a serial bus used for communication between a micro controller and ICs offering a SNERT interface. It is a single master bus and uses the µC´s serial interface for transmitting and receiving data. Clock is supplied by pin TxD while data is written or read through pin RxD (mode 0 of the serial interface). Address and data bytes are transmitted alternatively. The bus uses a third signal line to determine the correct address / data sequence as well as to update any readable registers in the devices.

The bit rate is 1/12 of the µC oscillator frequency. For a µC running at 12 MHz this means a transfer rate of 1 MBit/s.

CONTENTS

List of Figures

List of Tables

1. Introduction

As ICs become more complex there is an increasing need to control their functions by a microcontroller. Moreover several ICs usually work together rather closely in a system which requires communication between them. For these purposes the I^2C bus has been defined. The bus is widely known and its protocol is supported by many ICs.

As a consequence of its wide acceptance the bus becomes more and more loaded the more ICs are connected to it in a system. And communication grows the more functions the individual ICs comprise. If even time critical or close-to-real-time controlling has to be performed the I^2C bus is not suited very well. Therefore the need arises to have another, a faster bus for these high speed jobs.

One way would be to transfer eight bits in parallel. The µC could then access an IC like a RAM. This parallel bus solution however uses 11 pins of an IC and requires complex wiring. Therefore parallel busses are used only for very high speed requirements.

A solution that is considerably faster than the I^2C bus and yet has the advantages of a serial bus is offered by the typical 8051 µC: it is the standard serial interface. This interface can be configured to run in any of four modes. Operated in mode 0 it allows to implement an economical interface in the target devices. The bus that uses this mode of the serial interface is called *SNERT* bus. *SNERT* stands for *Synchronous No parity Eight bit Reception and Transmission*.

TABLE 1 Overview of different bus concepts for communication between ICs

Bus	speed	addressing	multi-master	number of wires
I^2C bus (7-bit addressing)	slow	112 device addresses[1]	yes	2
SNERT bus	medium	256 register addresses[2]	no	3
Parallel bus	high	256 register addresses	no	11[3]

1. 7 bits for addressing, 1 bit (LSB) indicates read/write; 16 addresses reserved for special purposes (+ 1024 addresses in 10 bit addressing mode)
2. 1 byte for addressing
3. 8 data bits, ALE, RDN, WRN

2. The SNERT bus concept

The SNERT bus follows a master - slave concept. The µC always is the master while the connected devices are the slaves. This simplifies the interfaces as no provisions for arbitration and data collision like in multi-master concepts have to be made. The master initiates the data transfer and supplies the clock. Data is either transmitted to a device or read from it.

The SNERT bus uses three wires. SNDA[1] carries the data, SNCL[2] supplies the clock. The third line SNRST[3] is a reset signal to indicate the start of a message. One or more devices can be hooked onto this bus, and systems can be expanded easily.

1. Another name for the SNERT data signal is SNERT_DA.
2. Another name for the SNERT clock signal is SNERT_CL.
3. Another name for the SNERT reset signal is SNERT_RST.

Address and data bytes are transmitted alternatively. The address is not a device address but accesses an individual register within the device. The following data byte then is loaded into this register or read from it. Different registers in different devices can be accessed consecutively without applying SNRST again.

3. Data transfer

The serial interface of the 8051 uses pins RxD and TxD for communication. In mode 0 data enters and exits through pin RxD while pin TxD outputs the shift clock. One clock pulse is generated for each bit of address or data transferred. Fig. 1 shows that SNDA is connected to RxD and SNCL to TxD of the µC.

A configuration is possible where instead of the µC a SNERT master device represents the interface for the SNERT bus. This master is controlled by the µC which also generates the reset pulse SN_RST. The data transfer on the bus is identical to the one controlled by the µC as master.

A transmission starts with a reset pulse on SNRST. Then any number of bytes follow. All bytes transmitted up to the next reset pulse is in the following referred to as SNERT *message*.

Every byte must be 8 bits long. The number of address/data byte pairs that can be transmitted in a message is unrestricted. All bytes are transferred with the least significant bit (LSB) first. The baud rate is fixed at 1/12 of the oscillator frequency.

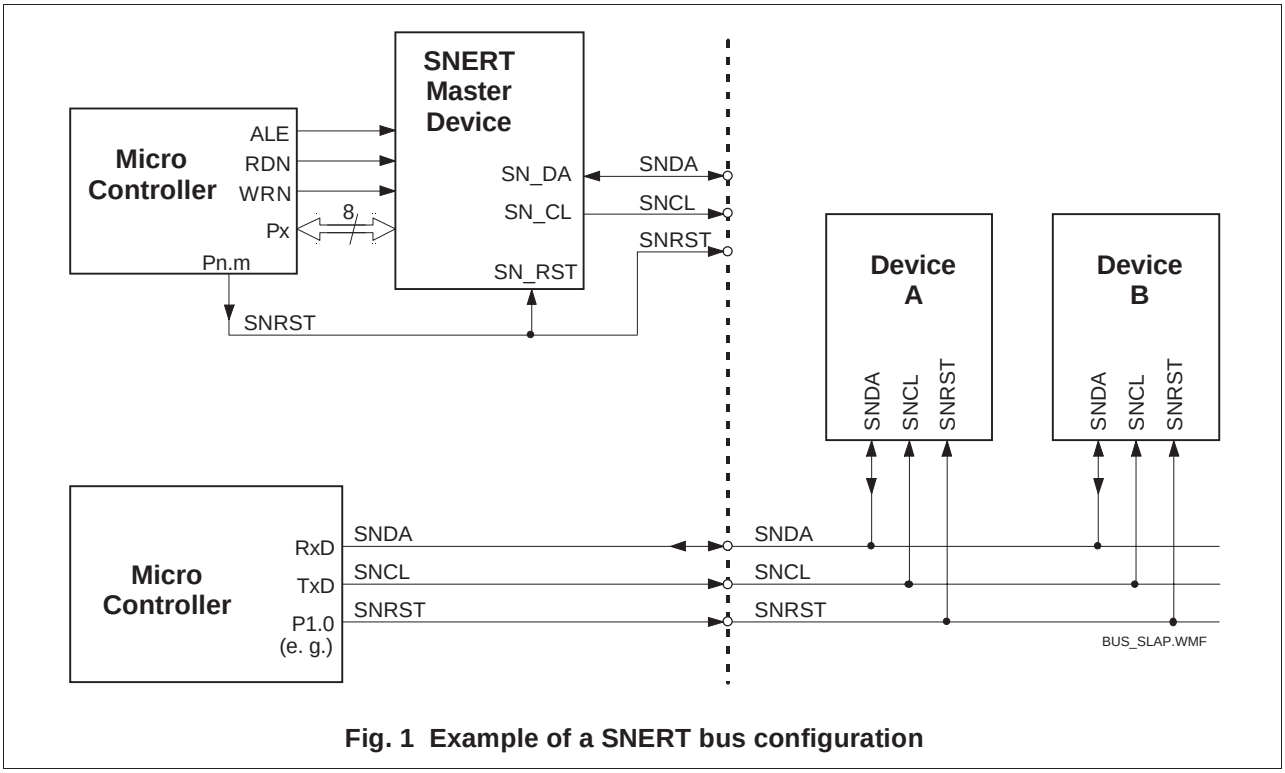

Fig. 1 Example of a SNERT bus configuration

The data transmitted or received is made up of address and data bytes alternatively. A SNERT message can consist of one or more pairs of address and data bytes. The address/data sequence is

synchronized by a pulse on the bus line SNRST. This pulse is transmitted at the beginning of each message. SNRST can be connected to any available port pin of the micro controller.

3.1 Functions of SNRST

Before any data transfer starts the receiver must be synchronized to the data on the bus. Since no start or stop conditions are supported by the serial interface, SNRST is used for this purpose. The functions of this signal are listed below:

- Indicate to the receiver that the following byte is an address byte
- Indicate to the receiver that the following bit is the first one (LSB) of the byte
- If read mode is implemented: load the readable registers in a device with the latest status information of the IC
- In some devices: synchronize data
  When a device has received any data these may become valid either immediately, or they are synchronized by some other signal (e. g. video ICs often use the vertical sync signal). Some devices however may require a pulse on the SNRST line to signal that the received data may now become valid. Refer to the data sheet of the device for information on how data is synchronized.

The signal SNRST is a bus signal and as such is used only for the above purposes. No other constraints or requirements should be imposed on it by any device (e. g. signal processing functions) as this might impair communication with or performance of other devices also connected to the bus.

3.2 Sending data to ICs

The transmission format for sending data to a device is shown in fig. 2.

Each SNERT message starts with a positive pulse on the SNRST line. Then pairs of address and data bytes are transmitted, each one starting with the LSB. No auto-incrementing of addresses is supported, so each data byte must be preceded by its appropriate register address byte. The message ends with the transmission of the last data byte. No stop or acknowledge bits are used.

As mentioned in chpt. 3.1 it may be necessary to activate the transmitted message by a SNRST pulse or some other pulse.

3.3 Receiving data from ICs

The transmission data format is shown in fig. 3.

The message starts with a pulse on the SNRST bus line. This loads all read registers with the latest status information. If the following address accesses a read register the device recognizes this and enables the register data to be read. The µC turns pin RxD into read mode. Upon each pulse that the µC transmits at pin TxD, one bit of data is clocked out of the device.

4. Electrical and timing characteristics

The electrical characteristic of the bus is primarily determined by the characteristic of the µC´s output port. The serial port is an alternate function of port 3, bits 0 and 1. Port 3 outputs contain internal

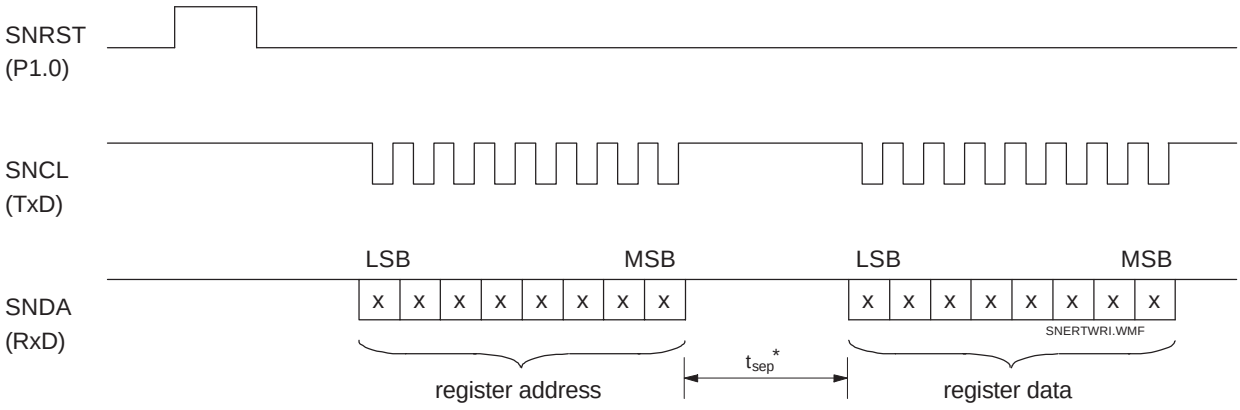

Fig. 2 SNERT bus: write mode

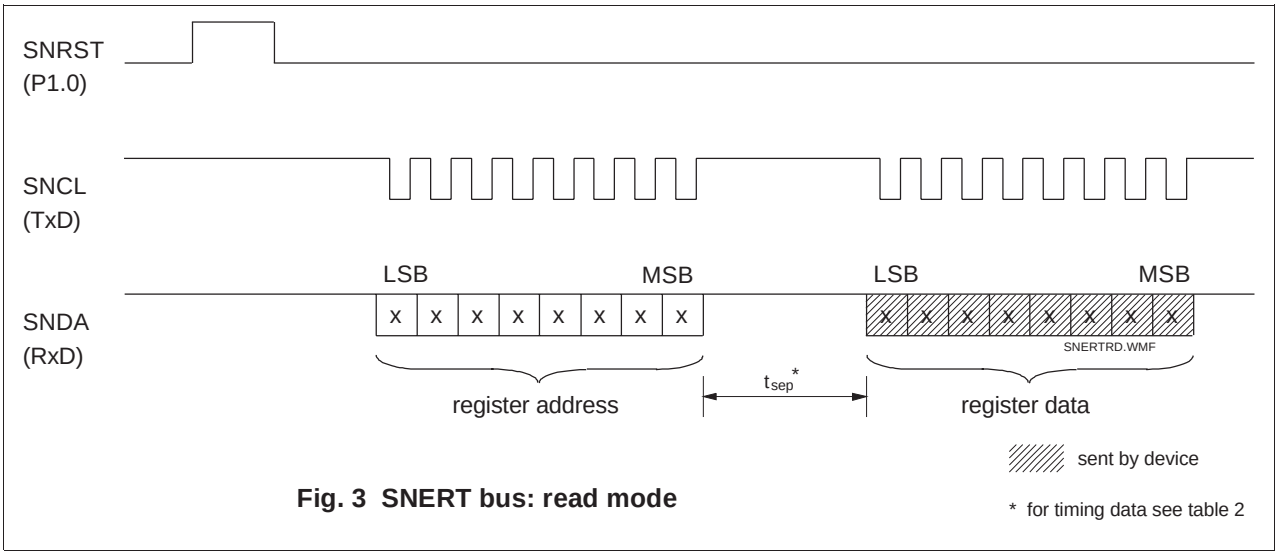

Fig. 3 SNERT bus: read mode

pull-up resistors so the bus is in a logical “1” state when it is not active. Fig. 4 shows a diagram of the port 3 I/O structure. If the port latch contains a “1” then the alternate output function is activated and the serial data from the SBUF register is transmitted.

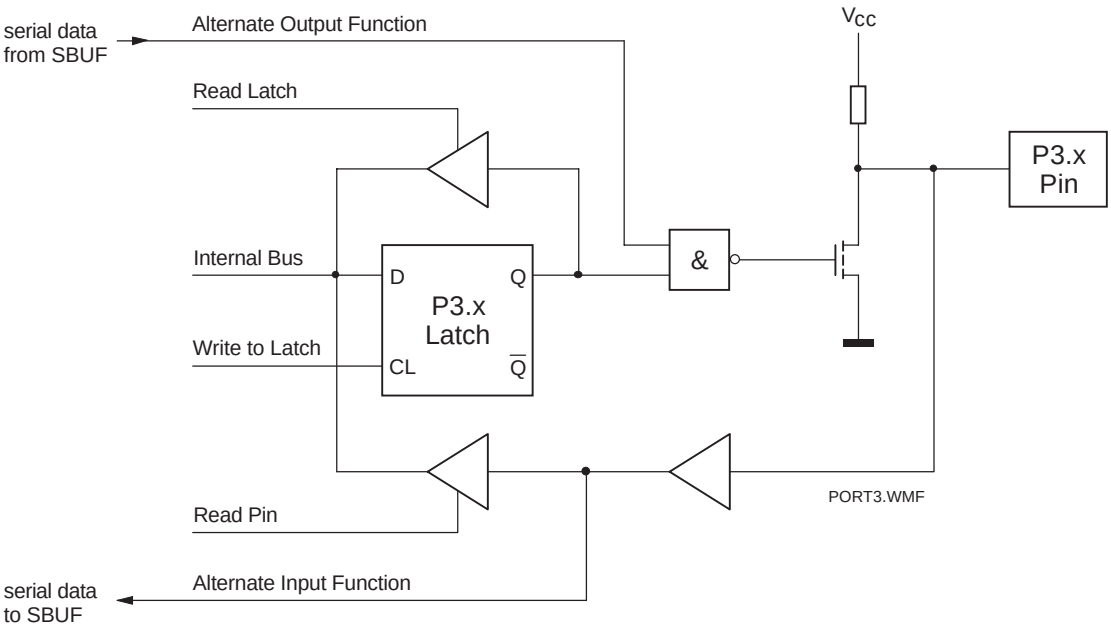

Fig. 4 Port 3 I/O structure

Fig. 5 shows the SNERT bus timing diagram and table 2 gives the timing data.

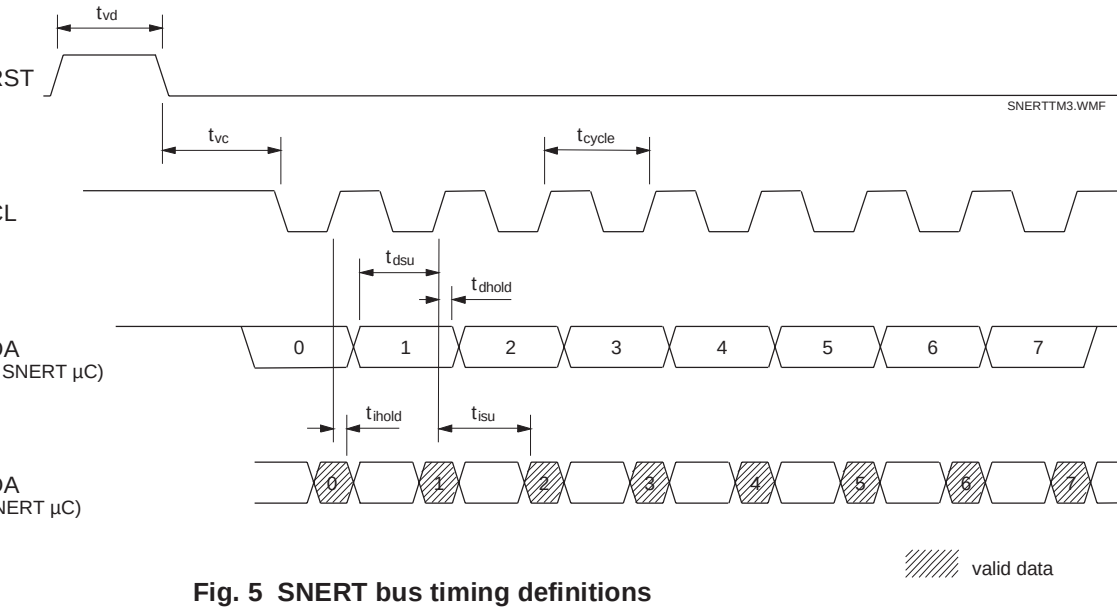

Fig. 5 SNERT bus timing definitions

TABLE 2 Electrical characteristics of the SNERT bus

Symbol	Parameter	Min.	Typ.	Max.	Unit	Remark
V_{OL}	Output low level			0.45	V	I_{OL} = 1.6 mA
V_{OH}	Output high level	2.4			V	I_{OH} = -60 µA
V_{IL}	Input low level			0.8	V	
V_{IH}	Input high level	2			V	
$1/t_{CLCL}$	µC oscillator frequency		12		MHz	
t_{cycle}	SNCL cycle time		12 t_{CLCL}			
K_{clock}	Clock duty cycle		50		%	
t_{dsu}	Data setup time	10 t_{CLCL} - 133			ns	
t_{dhold}	Data hold time	2 t_{CLCL} - 60			ns	
t_{ihold}	Input hold time	0			ns	
t_{isu}	Clock to data valid time			10 t_{CLCL} - 133	ns	
t_{vc}	Delay SNRST pulse -> data	1			t_{cycle}	
t_{vd}	SNRST high period	1			t_{cycle}	
t_{sep}	Byte separation time	1	4		t_{cycle}	see fig. 2 and 3

Above figures are taken from the data sheet of the 20 MHz version of the 8xC652/8xC654.

Initially the SNERT bus was intended to run on 1 MBit/s using an 8051 type µC clocked at 12 MHz. ICs controlled by this bus all support this minimum speed requirement. For higher data transfer rates check the data sheet of the attached devices.

5. References

[1] 8051-based 8-bit microcontrollers, Philips data handbook 1994

[2] The I^2C-bus and how to use it, Philips Semiconductors, 1992