SGS-THOMSON MICROELECTRONICS

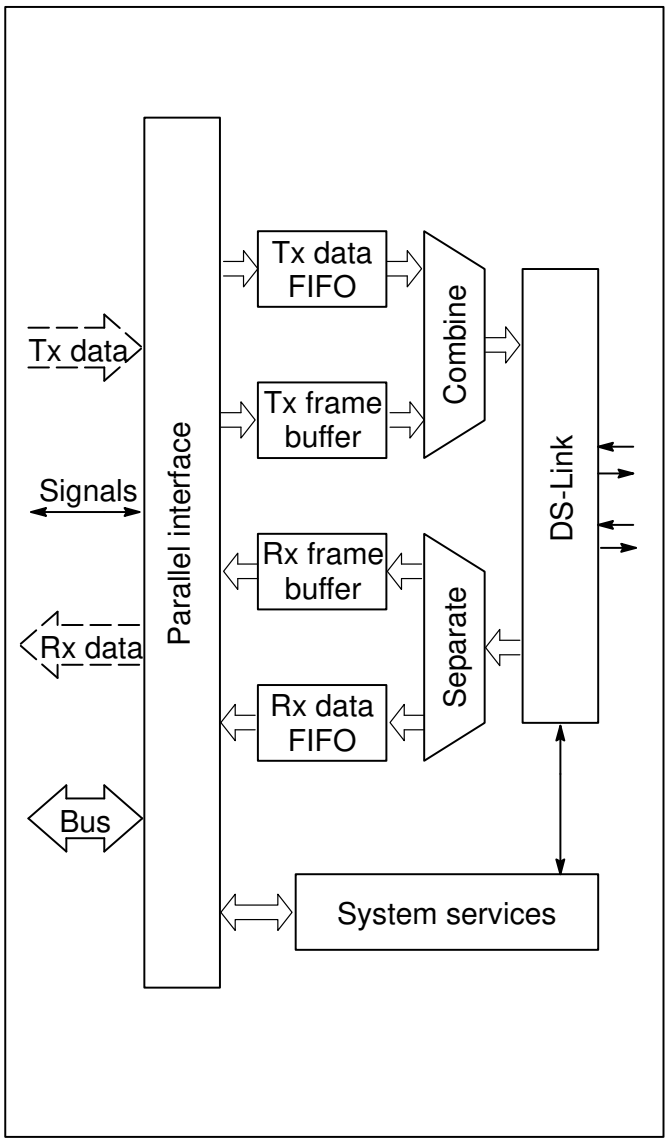
STC101

PARALLEL DS-LINK™ ADAPTOR

ENGINEERING DATA

FEATURES

- High speed parallel to DS-Link converter.
- Data-Strobe Link (DS-Link™) interface device for high speed asynchronous communications avoids the need for high speed clocks within the system. Interfaces directly to STC104 Asynchronous Packet Switch (APS).
- Performs DS-Link packetization for APS.
- Full duplex serial interface operating at 100 Mbits/s serial bandwidth in each direction (19 Mbytes/s bi-directional).
- Programmable parallel bus interface (16 and 32 bit modes).
- Variable packet length capability.
- 64 byte Tx and Rx FIFOs optimize packet processing performance.
- Packetization function can be disabled to provide simple point to point connection.
- Provides point-to-point bi-directional handshaken high speed FIFO function.
- Optional handshaken ports.
- Interrupt capability.
- Link loop back for test purposes.
- Independent clock systems.
- 100 pin quad flat pack package.
- Single +5V ±5% power supply.



APPLICATIONS

- Allow ATM, Fibrechannel, switched Ethernet/ Token Ring and other communications nodes to take advantage of SGS-THOMSON's high speed Asynchronous Packet Switch architecture.
- Connecting microprocessors/peripherals to STC1xx family communications devices.
- High speed heterogeneous link between microprocessors in a multi-processor system.

Contents

1 STC101 introduction 5

2 Communication on an STC101 system 6

2.1 Levels of communication protocols 6

3 STC101 functional overview 8

4 Pin designations 10

5 Processor interface modes 13

5.1 16-bit processor interface 13

5.2 32-bit processor interface 14

5.3 16-bit processor interface with token interfaces 14

5.3.1 16-bit processor interface with non-multiplexed token interfaces 14

5.3.2 16-bit processor interface with multiplexed token interfaces 16

6 Operation of the STC101 17

6.1 Transparent mode 18

6.1.1 Data transmission in transparent mode 18

6.1.2 Data reception in transparent mode 18

6.2 Packetizing mode 19

6.2.1 Data transmission in packetizing mode 19

6.2.2 Data reception in packetizing mode 20

7 Buffering 22

7.1 Data buffering for both modes of operation 22

7.2 Frame buffering for packetizing mode operation 23

7.2.1 Tx frame buffering 23

7.2.2 Rx frame buffering 23

8 Parallel interface 24

8.1 Access to the ports 24

8.1.1 Access to the framing and configuration/status ports 24

8.1.2 Access to the data ports 24

8.2 Valid/Hold protocol of the token interfaces 26

9 Link interface 27

9.1 Data/Strobe links 27

9.2 Low-level flow control 28

9.3 Link speeds 28

9.4 Errors on DS-Links 29

9.4.1 Reliable links 29

9.4.2 More reliable links 29

9.5 Link state on start up 30
9.6 Resetting DS-Links 30
9.7 Link connections 30

10 Interrupts 32

11 Clocking 33

12 Reset 33

13 Programmable register functionality 34

13.1 System services registers 34
13.2 Interrupt registers 35
13.3 Framing data registers 38
13.3.1 Tx framing registers 38
13.3.2 Rx framing registers 40
13.4 FIFO registers 42
13.5 DS-Link registers 43

14 Address map 45

15 Timing specifications 46

15.1 Clock timings 46
15.2 Bus interface timings 47
15.2.1 Asynchronous bus timings 48
15.2.2 Synchronous bus timings 52
15.3 Token interface timings 55
15.4 DS-Link timings 57
15.4.1 Link Input and Output relative skews 58
15.4.2 Skew budget 59

16 Electrical specifications 60

16.1 Absolute maximum ratings 60
16.2 Operating conditions 61
16.3 DC characteristics 61
16.4 Power rating 61

17 Package specifications 63

17.1 STC101 100 pin CQFP package pinout 63
17.2 STC101 100 pin CQFP package dimensions 64
17.3 STC101 100 pin CQFP package thermal data 65

18 Ordering information 66

1 STC101 introduction

The STC101 Parallel DS-Link Adaptor allows high speed serial DS-Links to be interfaced to buses and peripherals. It is part of the family of communications devices, which also includes the STC104 Asynchronous Packet Switch. These communications devices are based on the DS-Link. DS-Links consist of four wires, two in each direction, one carrying data and one carrying a strobe, hence the term DS-Links (data-strobe). Each link can operate at up to 100 Mbits/s, providing a bidirectional bandwidth of 19 Mbytes/s. The link protocol supports virtual channels and dynamic message routing, and provides a high data bandwidth. The DS-Link protocols are part of the proposed standard for Heterogeneous InterConnect (P1355).

The STC101 provides an inter-networking solution for mixed processor systems, allowing systems to be constructed using the optimum mix of microprocessors and peripherals, for processing power, communication bandwidth and system cost. Its potential range of applications include interfacing several different types of microprocessors and peripherals, and also applications such as switches for ATM, FibreChannel, and switched Ethernet and Token Ring systems.

The STC101 converts between the serial DS-Link format and external systems such as buses, peripheral devices and microprocessors. It is particularly suitable for interfacing such devices to interconnects which deal in packets consisting of data and header information. This header information may be used to demultiplex packets from different sources and/or route them through one or more switches.

The STC101 has two basic modes of operation, depending on whether it has packetization enabled or not. With packetization disabled it provides simple access to the DS-Link, all data provided to the STC101 is transmitted down the DS-Link. This mode can be used either by devices which do not need to use a higher level protocol on the DS-Link or by devices which take responsibility for the formation of packets. The STC101 simply transmits the data provided down the DS-Link hence the term *transparent mode* is used. With packetization enabled it can be used by less specialized devices such as processors to exploit efficiently devices such as the STC104 Asynchronous Packet Switch (refer to the *STC104 datasheet (document number 42 1470 06)* for details). In this mode the STC101 builds packets as used by the STC104 and hence this mode is referred to as *packetizing mode*.

In both modes the STC101 parallel interface can be used in one of three ways;

- 16-bit processor interface,
- 32-bit processor interface,
- 16-bit processor interface with token interfaces (providing two 9-bit unidirectional buses for the token interfaces).

The STC101 contains FIFO buffering for data and packet framing information. This smooths out differences in data rates and maximizes the possibilities for parallelism in packet handling.

2 Communication on an STC101 system

The STC101 can transfer packets of arbitrary length. It contains both a Tx and an Rx data FIFO which can each buffer up to 64 bytes (which is sufficient for one complete ATM cell). It also contains buffering for both the Tx and Rx framing information. Support is provided so that the STC101 may be used with a small amount of external logic to reproduce the virtual channel protocol as used by the control channels of the STC104 for a single virtual channel at a time, or with a larger amount of logic or a processor to reproduce the protocol for many virtual channels.

The structure of packets is shown in figure 2.1. To enable packets to be routed by STC104s, each packet has a header at the front which contains routing information. Bytes following the header are treated as the data section of the packet until a packet termination token is received. A packet termination token is either an EOP (end of packet) token or an EOM (end of message) token. Packets containing no data and terminated by an EOP token are called acknowledge packets and may be used for special purposes by higher level protocols.

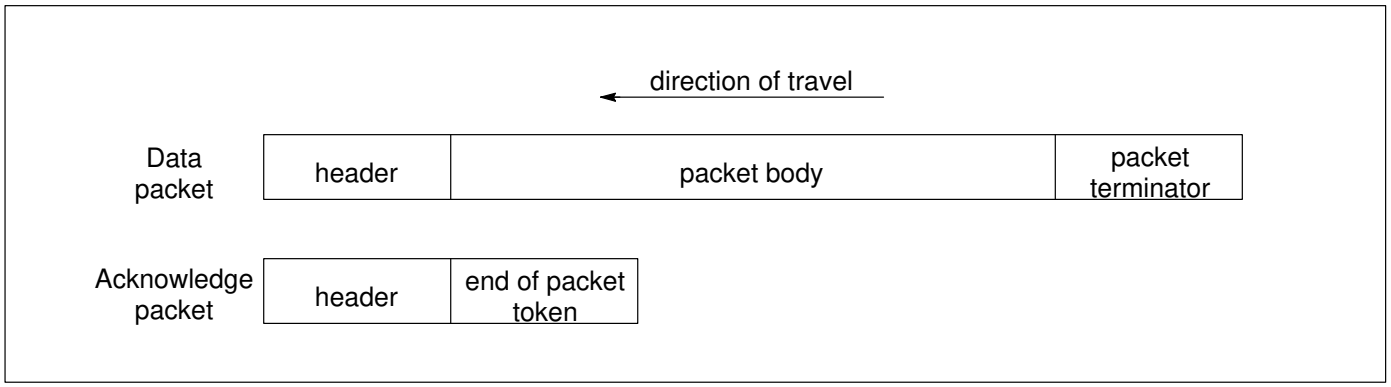


Figure 2.1 Structure of a packet on DS-Links

2.1 Levels of communication protocols

There is a hierarchy of protocols on DS-Links. The five levels of protocol are listed below. These protocols have been adopted by the working group of IEEE standard P1355, although the terminology employed by the working group differs slightly from that used here.

1 electrical

2 bit level (data-strobe encoding)

3 token level (includes device-to-device flow-control)

4 packet level (interface to routing function)

5 message level (e.g. 32 byte packets, packet/acknowledge protocol)

There is a DS-Link macrocell on the STC101 which deals with the first three levels of protocol automatically. In addition the STC101 contains logic which supports the packet level of protocol, that is the addition and subtraction of framing information (headers and termination tokens) from the raw data stream. The information regarding packetization is handled separately from the data to maximize the possibilities for concurrent processing. Note that the packetization function can be disabled if only a simple point-to-point connection is required.

When packetization is enabled, the STC101 handles the packet level protocol. The STC101 does this by either separating out or combining the streams of frame (packet headers and termination tokens) and data information. The device is full-duplex and the two directions are entirely independent.

Table 2.1 shows which of the levels of protocol are handled by the STC101 and which by the external processor when packetization is enabled and when it is disabled.

Protocol	Packetization enabled	Packetization disabled
	Protocol handled by	
electrical	C101	C101
bit level	C101	C101
token level	C101	C101
packet level	C101	external processor
message level	external processor	external processor

Table 2.1 Table showing which device handles the level of protocol

To transmit data in packets, the STC101 assembles four components of information, which are provided by the user on a packet-by-packet basis.

- packet header
- packet length
- number of bytes of data
- packet terminator type

The STC101 provides buffering for this information, so that the components of the next packet can be supplied whilst the previous packet is still being transmitted, enabling the link to be driven continuously.

As the STC101 receives packets, it separates them into the same set of components. Buffering is provided to enable the user to accept the components of a packet while the next packet is being received and separated.

3 STC101 functional overview

An STC101 block diagram is shown in figure 3.1. The STC101 provides an interface between a serial DS-Link and a parallel bus or peripheral. The parallel interface can be used as a 16 or 32-bit processor interface, or as a 16-bit processor interface with additional token interfaces.

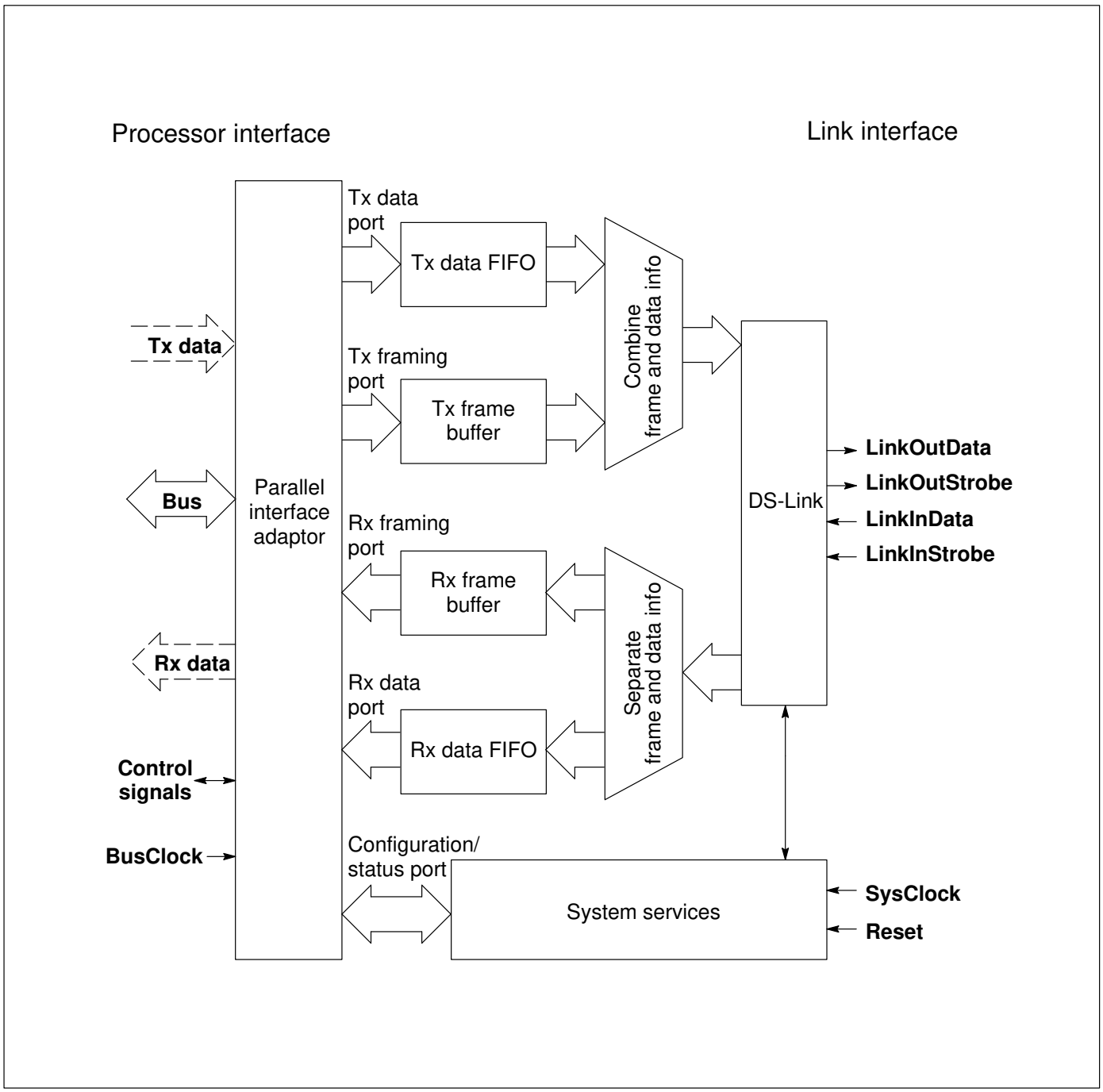


Figure 3.1 STC101 block diagram

The internal interface of the STC101 consists logically of five distinct ports, as follows:

- Tx data port – STC101 receives data for transmission down the DS-Link.
- Tx framing port – tx framing information (headers and termination tokens) that is added to the data to make packets for transmission down the DS-Link.
  The framing ports are disabled in transparent mode.
- Rx framing port – rx framing information (headers and termination tokens) that is separated from the data of packets received on the DS-Link.
  The framing ports are disabled in transparent mode.
- Rx data port – STC101 provides data received from the DS-Link.
- Configuration and status port – configuration registers are set up and read.

The parallel interface adapter multiplexes the logical ports onto the physical ports.

The STC101 contains FIFO buffering for data and packet framing information. This smooths out differences in data rates and maximizes the possibilities for parallelism in packet handling.

System services provides general information for operation of the STC101. System services include a set of configuration registers which contain status and control information.

4 Pin designations

The following tables outline the function of each of the pins. Package pinout details are given in chapter 17.

Signal names are prefixed by **not** if they are active low, otherwise they are active high.

Supplies

Pin	In/Out	Function
VDD		Power supply
GND		Ground

Table 4.1 STC101 supplies

Clocks

Pin	In/Out	Function
BusClock	in	Bus clock
LogicClock	in	50 MHz system clock

Table 4.2 STC101 clocks

System services

Pin	In/Out	Function
Reset	in	System reset

Table 4.3 STC101 system services

Bus

Pin	In/Out	Function
Addr0-4	in	Address bus.
Data0-31	in/out	Data bus. **Data0** is the least significant bit (LSB) and **Data31** is the most significant bit (MSB).
Data16-31		If only 16 bits of the data bus are required, **Data16-31** pins can be used for **RxData0-7** and **TxData0-7** signals (refer to table 4.5).
notCS	in	Chip select.
ALE	in	Address latch enable. Used for multiplexed address/data.
BusWait	out	Used as acknowledge access valid for synchronous bus. Used as a wait in asynchronous mode.
RdnotWr	in	Read or write data.
notBusOE	in	Data output enable.
BusSnotA	in	Synchronous or asynchronous bus operation.
Bus32not16	in	32-bit or 16-bit data bus.
EnableTxRx	in	Enable token interfaces.

Table 4.4 STC101 bus

Token interface

The signals are all synchronized to the bus clock.

Pin	In/Out	Function
RxValid	out	Signals valid data on **RxData0-7**.
RxHold	in	High when the connected device cannot accept data.
RxEOXnotData	out	Signals a packet termination token (EOP or EOM), not data. The two tokens are distinguished by bit 7 of the data bits (**RxData7**). If **RxData7** is 1, the token is an EOP. If **RxData7** is 0, the token is an EOM. The other data bits (**RxData0-6**) will be 0. This pin is enabled/disabled by the **notRxOE** pin.
notRxOE	in	Rx token interface data output enable. Used for multiplexed Rx and Tx token interfaces.
RxData0-7 / **Data24-31**	out	Rx token interface 8-bit data bus. These signals are multiplexed onto the **Data24-31** pins under control of **Bus32not16** and **EnableTxRx** pins (refer to table 4.4).
TxValid	in	Signals valid data on **TxData0-7**.
TxHold	out	High when the STC101 cannot accept data.
TxEOXnotData	in	Signals a packet termination token (EOP or EOM), not data. The two tokens are distinguished by bit 7 of the data bits (**TxData7**). If **TxData7** is 1, the token is an EOP. If **TxData7** is 0, the token is an EOM. The other data bits (**TxData0-6**) should be 0.
TxData0-7 / **Data16-23**	in	Tx token interface 8-bit data bus. These signals are multiplexed onto the **Data16-23** pins under control of **Bus32not16** and **EnableTxRx** pins (refer to table 4.4).

Table 4.5 STC101 token interface

Control signals

The signals are all synchronized to the bus clock.

These outputs reflect the state of the corresponding bits of the Tx and Rx Interrupt Status registers, see section 13.2.

Pin	In/Out	Function
HeaderValid	out	Signal to indicate a valid header has been received.
PacketRx	out	Signal to indicate a complete packet has been received, i.e. a packet terminator token has been received.
TxFifoLevel	out	Signals the Tx FIFO has reached its set level, see section 13.2, page 35 for details.
RxFifoLevel	out	Signals the Rx FIFO has reached its set level, see section 13.2, page 35 for details.
PacketTx	out	When set to 1, the **TxSendPacket** register (see table 13.10) can be written to.
AckRx	out	Signals that an acknowledge packet (an empty packet terminated by an EOP) has been received.
Int	out	Interrupt signal.
DisableInt	in	The **EnableInterrupts** register is cleared when this pin is asserted.

Table 4.6 STC101 control signals

Link

Pin	In/Out	Function
LinkInData	in	Link input data channel
LinkInStrobe	in	Link input strobe
LinkOutData	out	Link output data channel
LinkOutStrobe	out	Link output strobe

Table 4.7 STC101 link

Miscellaneous

Pin	In/Out	Function
HoldToGND		Must be connected to **GND**
HoldToVDD		Must be connected to **VDD**
DoNotWire		Must not be wired

Table 4.8 STC101 miscellaneous pins

5 Processor interface modes

The parallel interface of the STC101 can be used in one of three ways; 16-bit processor interface; 32-bit processor interface; 16-bit processor interface with additional token interfaces.

The parallel bus interface can be used asynchronously or synchronously depending on the **BusSnotA** pin. In both synchronous and asynchronous operation, all outputs are synchronized to the external bus clock.

5.1 16-bit processor interface

Figure 5.1 shows the STC101 processor interface configured as a 16-bit data bus interface with its token interfaces disabled.

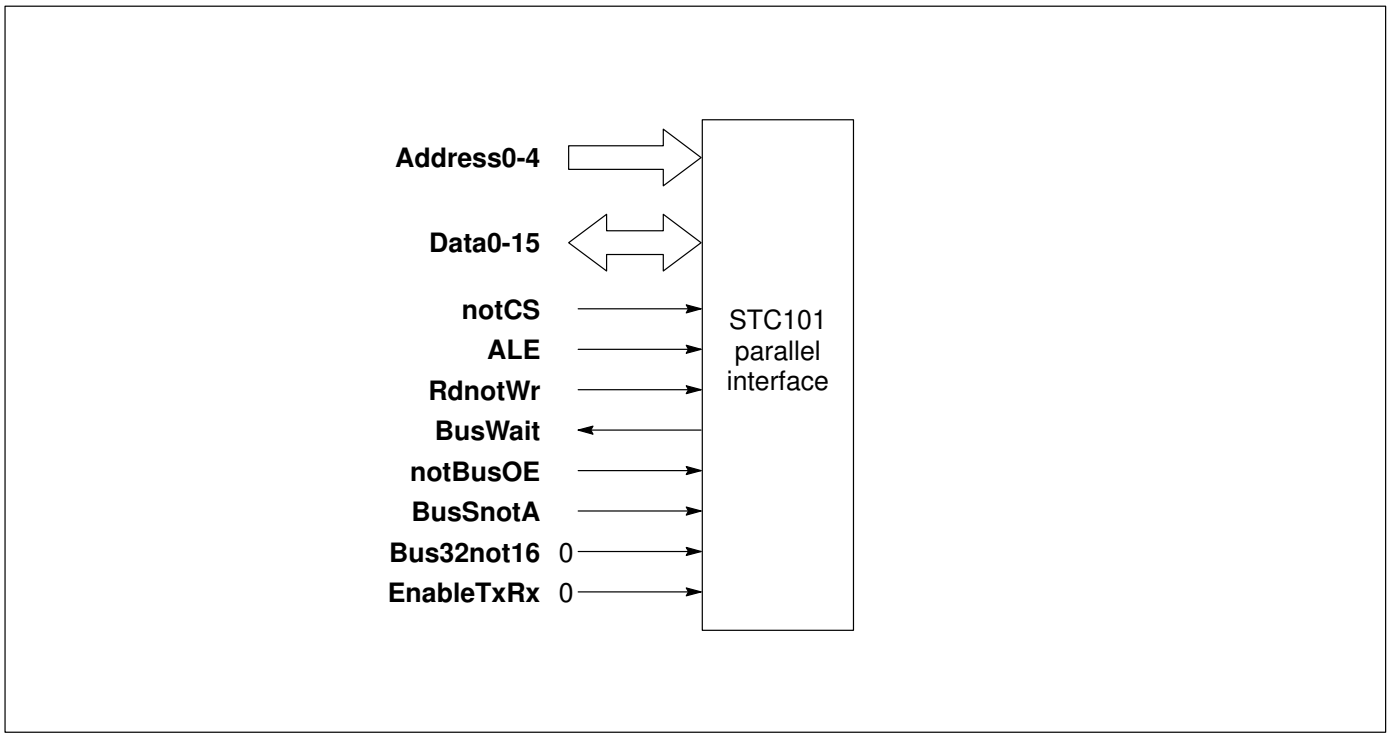


Figure 5.1 16-bit processor interface

The **Bus32not16** pin is held low for 16 bit operation. The 16-bit data bus is on pins **Data0-15**, with **Data0** the least significant bit (LSB) and **Data15** the most significant bit (MSB). In this case the **EnableTxRx** pin is low to disable the token interfaces and the data pins **Data16-31** are not used.

The chip select pin (**notCS**) and the data output enable pin (**notBusOE**) are active low.

The **BusWait** signal is used to signal acknowledge access valid for synchronous operation, or used as a wait in asynchronous mode.

The **ALE** pin is used to signal address latch enable for multiplexed address/data.

5.2 32-bit processor interface

Figure 5.2 shows the STC101 processor interface configured as a 32-bit data bus interface with its token interfaces disabled.

The **Bus32not16** pin is held high for 32 bit operation. The 32-bit data bus is on pins **Data0-31**, with **Data0** the least significant bit (LSB) and **Data31** the most significant bit (MSB). The **EnableTxRx** pin is low to disable the token interfaces.

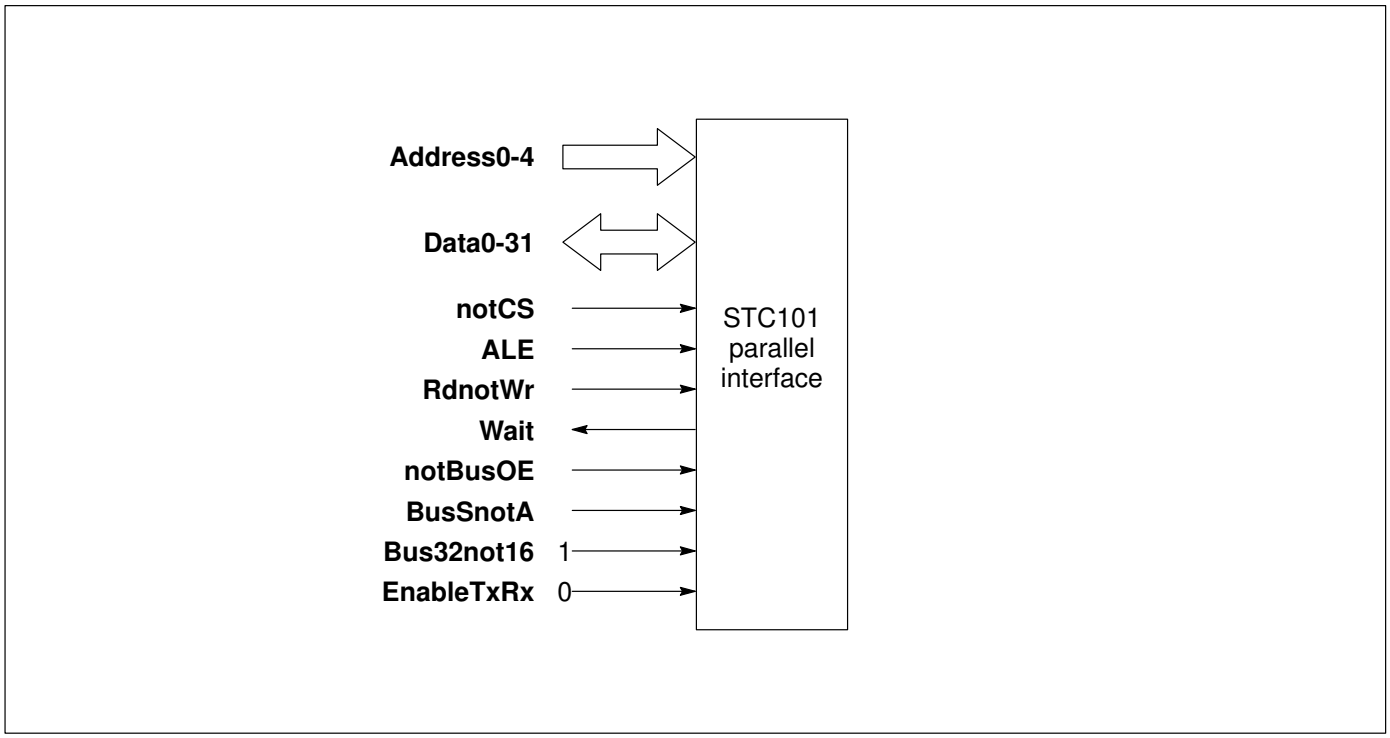


Figure 5.2 32-bit processor interface

5.3 16-bit processor interface with token interfaces

This section describes the STC101 parallel interface configured as a 16-bit processor interface with token interfaces.

5.3.1 16-bit processor interface with non-multiplexed token interfaces

Figure 5.3 shows the STC101 being used as a 16-bit processor interface with Tx and Rx token ports, providing two 9-bit unidirectional buses for the token interfaces.

Data16-31 pins are used for the **RxData0-7** and **TxData0-7** signals. The **RxData0-7** signals are multiplexed onto the **Data24-31** pins under control of the **Bus32not16** and **EnableTxRx** pins. The **TxData0-7** signals are multiplexed onto the **Data16-23** pins under control of the **Bus32not16** and **EnableTxRx** pins.

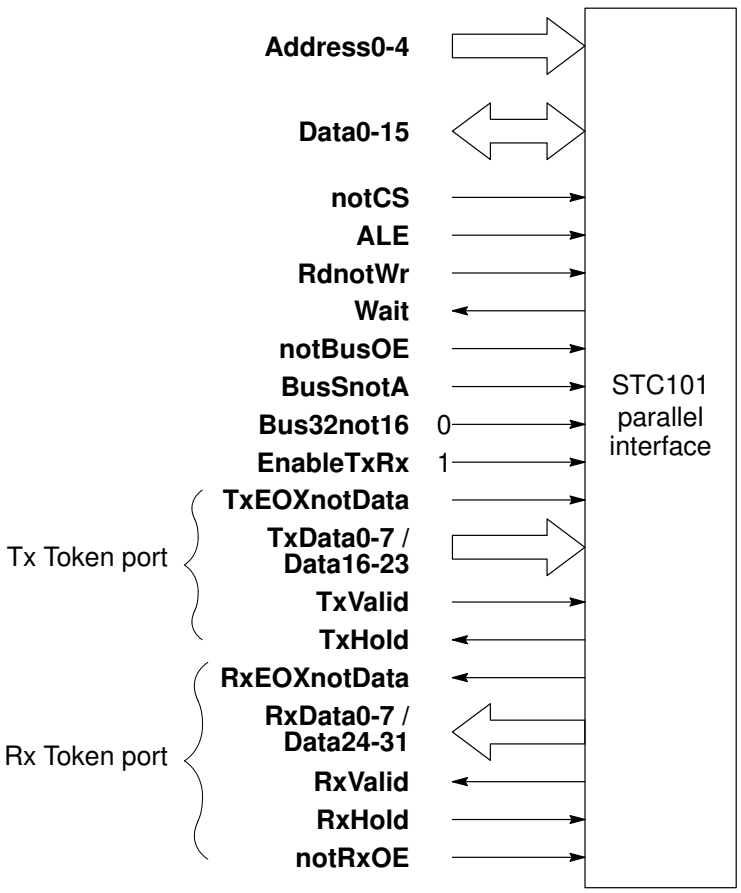


Figure 5.3 16-bit processor interface with token interfaces

5.3.2 16-bit processor interface with multiplexed token interfaces

Figure 5.4 shows the STC101 processor interface configured as a 16-bit processor interface with multiplexed Rx and Tx token interfaces. The **notRxOE** pin is used to enable the Rx output. The combined Rx and Tx data bus (**TxRxData0-7**) is multiplexed onto the **Data16-23** and **Data24-31** pins.

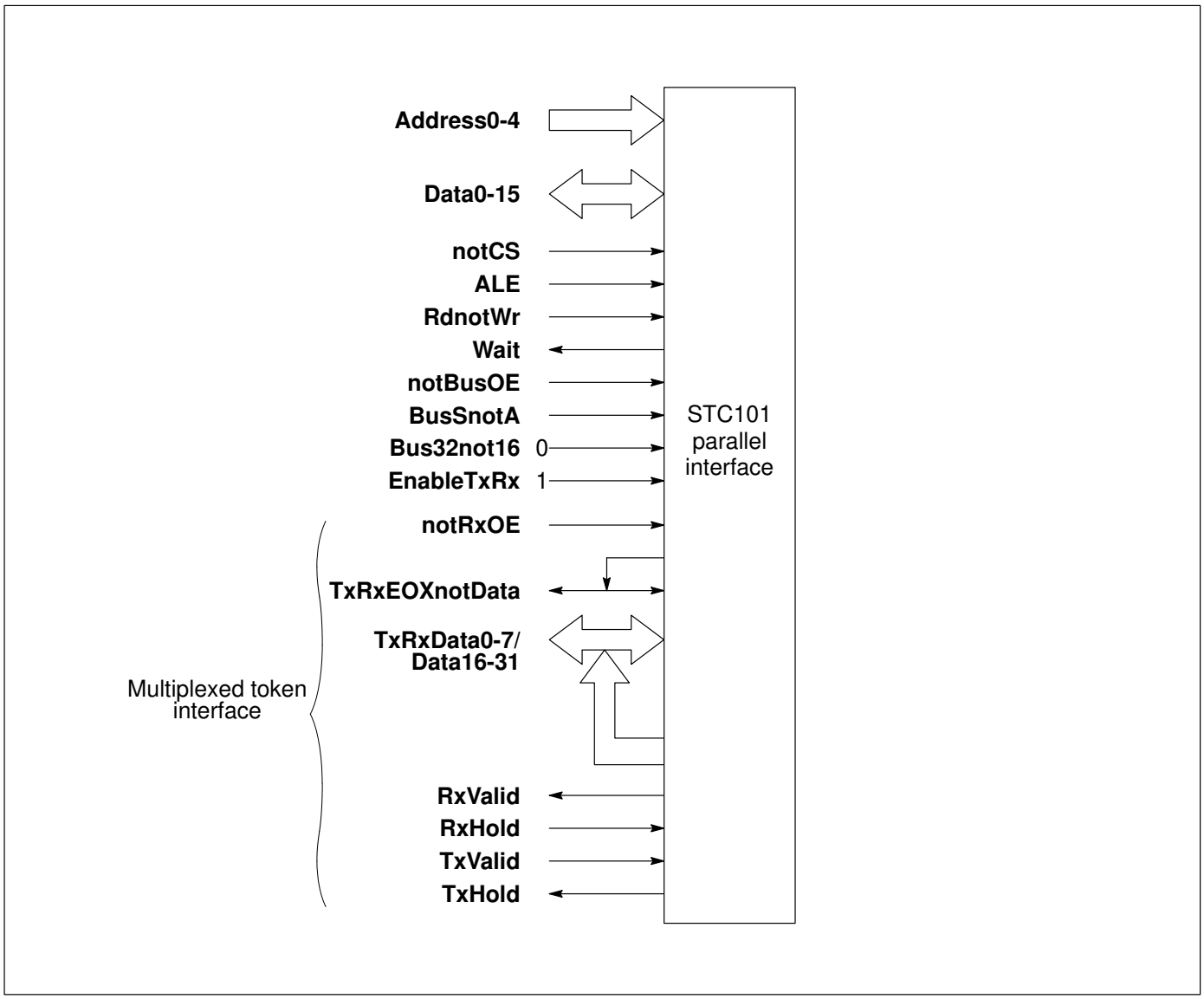


Figure 5.5 16-bit processor interface with multiplexed token interfaces

6 Operation of the STC101

The STC101 has two basic modes of operation, which is determined by the **EnablePacketization** bit in the **DeviceConfig** register (see table 13.3). The default configuration is with packetization disabled.

When packetization is disabled, the STC101 can be used for a simple point to point connection, providing a parallel interface to the DS-Link with FIFO buffering of the data. This mode can be used when the DS-Link is simply transferring data, with no higher level protocol. Alternatively it may also be used where the attached processor or hardware takes responsibility for any higher level protocols. The STC101 simply transmits the data provided down the DS-Link and provides data received from the DS-Link on the parallel interface, hence the term *transparent mode* is used. Figure 6.1 shows an STC101 in transparent mode with the processor interface configured as a 16-bit data bus interface and a pair of handshaken token interfaces.

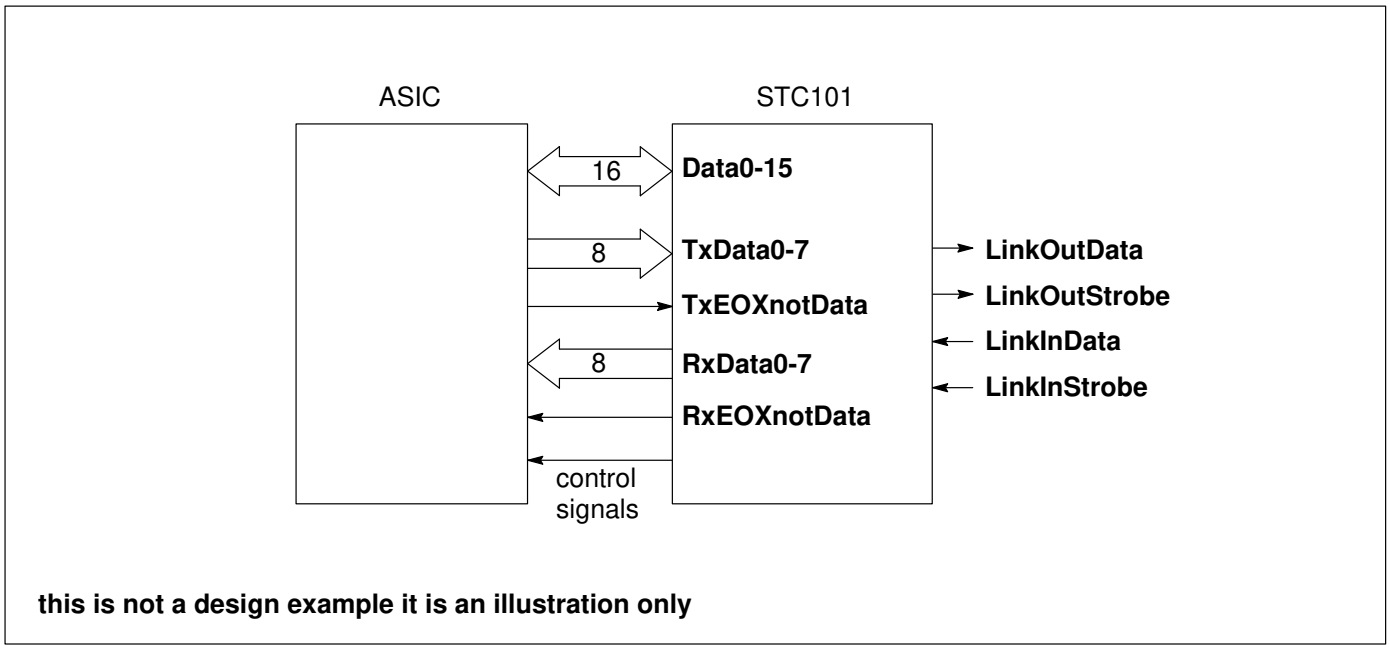


Figure 6.1 Transparent mode

When packetization is enabled, the STC101 can be used by less specialized devices such as processors to exploit efficiently devices such as the STC104 Asynchronous Packet Switch. In this mode the STC101 builds packets, hence this mode is referred to as *packetizing mode*. Figure 6.2 shows a simple case with the processor interface used as a data bus only. The STC101 has its token interface disabled. The microprocessor generates and processes all the packet framing information. It generates packets by writing the framing information into registers addressed by the address bus (**Addr0-4**).

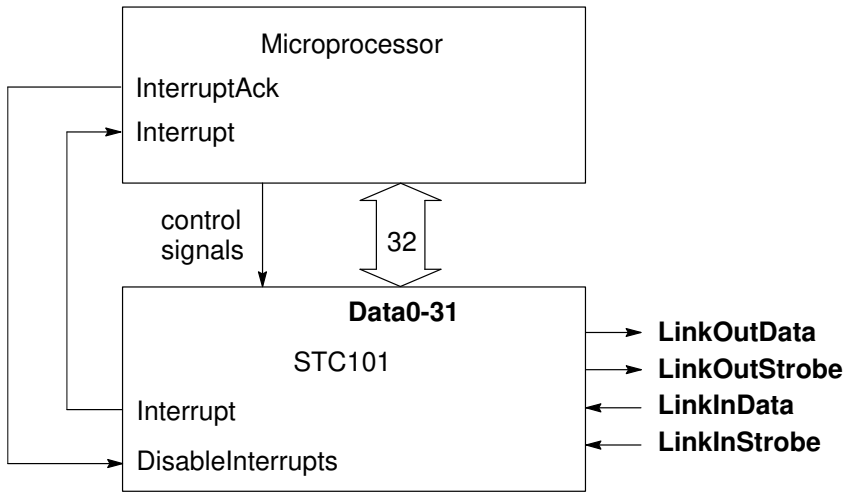


Figure 6.2 Packetizing mode

6.1 Transparent mode

This section describes data transmission and data reception for an STC101 in transparent mode.

6.1.1 Data transmission in transparent mode

When packetization is disabled, data supplied to the Tx port is transmitted immediately from the DS-Link, or buffered in the Tx FIFO if it is supplied more quickly than the DS-Link is able to transmit it. If the token interfaces are not used, and all data is transferred using the processor interface, then it is impossible to cause the DS-Link to transmit EOP or EOM tokens in this mode. Thus in this case the STC101 can only be used as a point to point distributed FIFO, and is not useful as an interface to a system using a packet level of protocol such as one using STC104s. Care must be taken to ensure that the Tx FIFO is not overfilled, this can be done by setting the Tx level register setting (see section 13.2). If the token interfaces are used, then an additional pin (**TxEOxnotData**) is provided on the Tx token interface so that EOM or EOP tokens can be transmitted. In this case the STC101 takes no part in the formation of packets, but simply provides some buffering on an interface to the DS-Link.

6.1.2 Data reception in transparent mode

When packetization is disabled, data supplied to the DS-Link is provided immediately on the Rx port, or buffered in the Rx data FIFO if it is received on the DS-Link faster than it is read out from the STC101.

Note that care must be taken to ensure that the processor does not attempt to read more data than is present in the FIFO because the **BusWait** signal is asserted when there is no valid data present and therefore the processor could be waiting for the read for an indeterminate time, until some data

arrives on the DS-Link. This can be avoided by setting the **RxLevel** register to an appropriate value, and when an interrupt occurs signalling that the level has been reached, then that number of bytes can be read out.

Note that the flow-control protocol, enforced automatically by the DS-Link (see section 9.2), ensures that this FIFO cannot be over-filled.

If the token interface is not used, and all data is transferred out of the STC101 by means of the processor interface, then it is impossible to observe the reception of EOP or EOM tokens, and they should be prevented from being stored in the Rx FIFO by setting the **SuppressRxEOX** bit in the **DeviceConfig** register. Thus in this case the STC101 can only be used as a point to point distributed FIFO, and is not useful as an interface to a system using a packet level of protocol such as one using STC104s. If the token interfaces are used, then an additional pin (**RxEOXnotData**) is provided on the Rx token interface so that EOX tokens can be received. In this case the STC101 takes no part in the decoding of packets, but simply provides buffering on the interface to the DS-Link.

6.2 Packetizing mode

This section describes data transmission and data reception for an STC101 in packetizing mode.

6.2.1 Data transmission in packetizing mode

In this mode the framing information, i.e. the packet length, headers and packet terminators, is supplied separately from the data and the STC101 combines these to form packets. The framing information is set up in registers, see chapter 13 for full details of these registers. The framing information for a packet is generated by setting bits in the **TxSendPacket** register (see table 13.10). The values written to the various bit fields determine the framing information. Note, writes to the **TxSendPacket** register can only be made when the **SendPacket** bit of the **TxInterruptStatus** register (see table 13.4) is 1.

If packetization is enabled, a packet will only be transmitted when both the complete framing information and some data have been supplied. If data is supplied before the framing information, it is buffered in the Tx data FIFO, until the framing information is provided. Care must be taken to ensure that the Tx FIFO is not overfilled, this can be done by setting the Tx level register setting (see section 13.2). If the framing information is provided before there is any data, and the packet length (**PacketLength** bits of the **TxSendPacket** register) is not 0, the framing information is buffered in the Tx frame buffer and the transmission of the packet is delayed until data is available. If the packet length is set to 0, the packet is normally transmitted immediately with the given header and terminator. Headers can be 1 to 4 bytes, and terminators can be either EOP (end of packet) or EOM (end of message).

The headers to be added to packets are defined by writing to the **TxPacketHeaderUpper0-1** or **TxPacketHeaderLower0-1** registers and the corresponding **TxHeaderLength0-1** register. These registers do not need to be written for every packet if the same headers are required.

Note that the **TxPacketHeader0-1** and the **TxHeaderLength0-1** registers can be written to whilst the **SendPacket** bit of the **TxInterruptStatus** register is not set without affecting the framing information for the packet to be transmitted. The **SendPacket** bit must be set before the **TxSendPacket** register bits can be set.

Delayed packet transmission

When the stream of data into the Tx port is slow relative to the speed of the DS-Link, then the performance of a routing network to which the STC101 is connected can be improved by buffering each packet so that it can be sent in a burst from the DS-Link. This can be achieved by withholding the framing information until the necessary data is present in the Tx data FIFO; this can be detected by setting the **TxLevel** register to an appropriate value.

Once the framing information of the packet has been provided, the STC101 starts transmitting the contents of the Tx data FIFO from the DS-Link. The DS-Link will stop transmitting once it has sent an EOP or EOM if the framing information and data of the next packet have not both been supplied.

Long packet transmission

Packets longer than the limit imposed by the **PacketLength** bit field in the **TxSendPacket** register can be sent by sending the first part as a packet with no terminator, and sending subsequent parts as packets with no header. Note that if the packet is very long there may be one or more parts with no header and also no terminator. The last part of the packet must have a terminator (EOP or EOM).

Packet abort

A packet abort command can be used to terminate a packet which has started but for which the supply of data has ceased. Note that it is the responsibility of the user to ensure that the correct packet is aborted. Writing a 1 to the **PacketAbort** bit of the **TxPacketAbort** register causes the packet currently being transmitted to be aborted. When this occurs, the header of the packet is sent, followed by any data which is in the Tx FIFO. An EOM token is then sent to terminate the truncated packet. The user should not provide more data after the packet abort command has been issued until the FIFO has emptied and the packet has been terminated. This condition can be met be setting the Tx level to indicate when the FIFO becomes empty. After that, normal transmission can be resumed.

6.2.2 Data reception in packetizing mode

When packetization is enabled, the packet information is provided in registers. The **RxInterrupt-Status** register contains information about the state of the Rx data FIFO.

The STC101 takes the header (the first 1 to 4 bytes depending on the setting of the **RxHeader-Length** bit-field in the **DeviceConfig** register) of each packet received and places it in the **RxPacke-tHeader** register. The **HdrValid** bit of the **RxInterruptStatus** register is set at the same time. The **HdrValid** bit can be unset by writing a 1 to the **AckHdrValid** bit of the **RxAcknowledge** register. The contents of the **RxPacketHeader** register do not change whilst the **HdrValid** bit is set to 1, but may change at any time when the **HdrValid** bit is 0.

Packet headers can be received either by polling the **RxInterruptStatus** register until the **HdrValid** bit is set, or by writing a 1 to the **HdrValidEnable** bit of the **RxInterruptEnable** register and waiting for an interrupt to occur. The **RxPacketHeader** register can then be read to determine the header of the packet. The Rx frame buffer enables the STC101 to receive the packet header of the next packet and to start inputting the data of the packet into the Rx data FIFO whilst the **RxPacketHeader** register contains a valid value.

The STC101 counts the data bytes of each packet received on the DS-Link. When the packet terminator is received, the length of the body of the packet is recorded in the **RxPacketLength**

register, and the terminator type is recorded in the **RxInterruptStatus** register. Writing a 1 to the corresponding bit in the **RxAcknowledge** register causes the bit in the **RxInterruptStatus** register to be unset, enabling the packet length and terminator type information of the next packet to be recorded. The packet length and terminator type information can not change when the associated bit of the **RxInterruptStatus** register is 1, but can change at any time when it is 0. There is buffering for the packet length and terminator type information inside the STC101 so that another packet can start to be received whilst the **RxPacketLength** register contains valid information.

If a packet ends before the specified number of header bytes are received, the entire packet is discarded and the **ShortPkt** bit of the **RxInterruptStatus** register is set. The appropriate terminator type bit (**EOMRxed** or **EOPRxed**) is also set. While the **ShortPkt** bit is set the **HdrValid** bit cannot become set. Writing a 1 to the **AckShortPkt** bit of the **RxAcknowledge** register unsets the **ShortPkt** bit, enabling it to be set to 1 again if another short packet is received. The short packet information is buffered in the Rx frame buffer in the same way as other header information.

Long packet reception

The **RxFifoLevel** bit of the **RxInterruptStatus** register is set to 1 when the Rx FIFO has reached the level given in the **RxLevel** register. If the **RxLevel** register is programmed with its maximum value of 64 bytes, the **RxFifoLevel** bit is a FIFO full indication. This enables Rx packets which are longer than 64 bytes to be received as the receiving device knows when the Rx FIFO is full to remove the data before a packet terminator is received.

Note that if the token interfaces are used, the **RxEOXnotData** pin is provided on the Rx token interface so that EOX tokens can be received. These tokens can be supplied even when the framing ports are used so that external hardware connected to the token interface (e.g. a DMA controller) knows when the data belonging to a packet is finished.

If a packet is longer than the 4 Kbyte counter size, the **CountOverflow** bit of the **RxInterruptStatus** register is set. This bit remains set until a 1 is written to the **AckCountOverflow** bit of the **RxAcknowledge** register. While it is set none of the other bits of the **RxInterruptStatus** register can become set, except the **RxFifoLevel** and **LinkError** bits, and the STC101 will not receive any more data on the DS-Link. When the **AckCountOverflow** bit is set, the counter is cleared enabling counting to start again. In this case the receiving device must keep a count of the length of the packet. The counter may overflow repeatedly for an exceptionally long packet.

7 Buffering

Buffering on the STC101 allows concurrency in the processing of the packet streams. The STC101 contains two data FIFOs, a Tx FIFO and an Rx FIFO. Both can buffer up to 64 bytes. Data buffering is used in both modes of operation. In addition the STC101 contains a Tx frame buffer and an Rx frame buffer which are used in the packetizing mode of operation to provide FIFO buffering of framing information.

7.1 Data buffering for both modes of operation

Rx and Tx data FIFO buffering is provided on the STC101 in both modes of operation. The data for the next packet can start to be input to the Rx data FIFO while the previous packet is still being read from the STC101, and data can be input to the Tx data FIFO whilst the data for the current packet is still being transmitted from the DS-Link, enabling continuous transmission.

The Tx and Rx FIFOs can be programmed with either a high or a low level, see figure 7.1. The high or low level is determined by the **LevelHighnotLow** bit in the Tx and Rx **Level** registers.

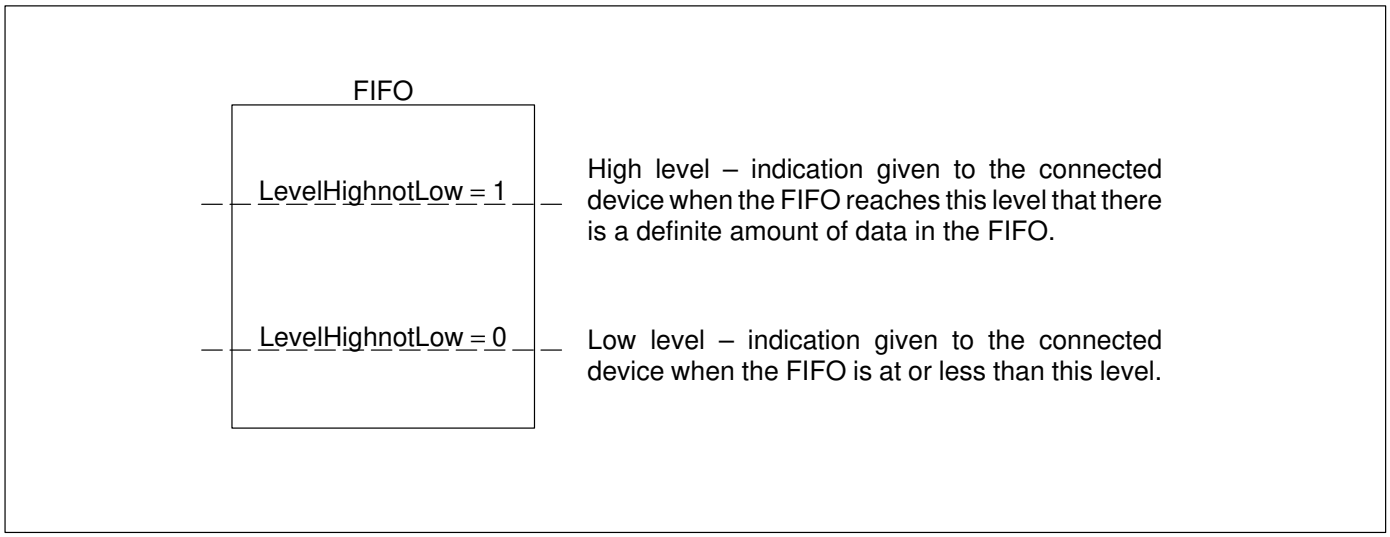


Figure 7.1 Data FIFO programmable levels

The low level provides an indication to a processor or other device that there is a definite amount of space in the FIFO. It guarantees a certain amount of buffer space is available (useful when it is more efficient to send data in a block).

The high level provides an indication to a connected device that there is a definite amount of data in the FIFO. The Tx high level can be used by systems in which the supply of Tx data is slow or irregular by signalling that there is enough data in the Tx FIFO to transmit a complete packet efficiently down the DS-Link. The Rx high level can be used as a 'Fifo full' indication if the **RxLevel** register is programmed with its maximum value of 63 bytes. The interrupt is signalled when the contents of the fifo are greater than 63 bytes, i.e. when the fifo contains 64 bytes. This can be used to enable Rx packets which are longer than 64 bytes to be received. The receiving device knows when the FIFO is full and that it must remove data from the FIFO before a packet terminator can be received.

When the FIFO has reached the programmed level, the corresponding **FifoLevel** bit of the **InterruptStatus** register is set to 1, and the **FifoLevel** pin output is set high.

7.2 Frame buffering for packetizing mode operation

In packetizing mode the framing information (information regarding the length, headers and terminators of packets) is supplied separately from the packet data. This information is set up in programmable registers, as described in chapter 13. In addition, in this mode there are Tx and Rx frame buffers which store the framing information.

Note that frame buffering is not used in the transparent mode of operation.

7.2.1 Tx frame buffering

The framing information of the next packet can be set up whilst the data for the previous packet is still being supplied to the Tx data port, and/or the packet is being transmitted from the DS-Link. There is an internal 2-place FIFO buffer for the framing information of the next packets to be sent. Writing to the **TxSendPacket** register (see table 13.10 for details) sets up the framing information for a packet and 'pushes' the data into the frame buffer. When the framing information has been accepted, the **SendPacket** bit of the **TxInterruptStatus** register is set to 1 and another set of framing information can be provided.

7.2.2 Rx frame buffering

The header information of an incoming packet can be read from the Rx framing port and processed whilst the data of the packet is still arriving. For reasonably short packets (i.e. those smaller than the Rx data FIFO), the information about the packet's length and terminator can be read whilst the next packet is being input.

There are two 1-place Rx frame buffers on the STC101. One buffer contains Rx header information, the other contains Rx terminator information.

The buffer containing Rx header information enables the next packet to be received whilst the framing information for the current packet is still being processed. The frame buffer contains the value of the header and information that the header is valid or that a too-short packet has been received. Information is copied from the buffer to the **RxPacketHeader** and **RxInterruptStatus** registers provided that neither the **ShortPkt** nor the **HdrValid** bits of the **RxInterruptStatus** register are set. One or other of those bits then becomes set, and the buffer is free to receive the information regarding the following packet. Once the buffer is occupied the STC101 will not attempt to take the header of the next packet from the DS-Link until the buffer is cleared.

The other Rx frame buffer contains information about the packet length and terminator. This enables the header to be read and processed while the body of the packet is still being received. Information is copied from the internal buffer to the **RxPacketLength** and **RxInterruptStatus** registers provided the **EOPRxed**, **EOMRxed**, and **AckRxed** bits of the **RxInterruptStatus** register are all zero. One of these bits then becomes set and the buffer is free to receive the information regarding the following packet. Once the buffer is occupied and the terminator of a packet is received, the STC101 will not attempt to take any further data from the DS-Link until the buffer is cleared.

Note that because the Rx header buffer and the Rx terminator buffer are independent, it is possible for the corresponding information in the **RxInterruptStatus** register to contain header information and terminator information for two successive packets.

8 Parallel interface

The parallel interface of the STC101 contains a bus, which implements the Rx and Tx framing ports and the configuration and status port. This bus interface can be treated as 16 or 32 bits wide dependent on the **Bus32not16** pin, and can be used asynchronously or synchronously depending on the **BusSnotA** pin. In both synchronous and asynchronous operation, all outputs are synchronized to the external bus clock.

The parallel interface of the STC101 can be used in 1 of 3 ways:

1 16-bit processor interface

2 32-bit processor interface

3 16-bit processor interface and IO token interfaces

In the third case there are two unidirectional parallel buses, **RxData0-7** and **TxData0-7**, which have associated handshake control lines. Components connected to the interface bus need to provide or receive data and a flag (**RxEOxnotData** or **TxEOxnotData**) indicating whether the token is data or a packet termination token.

8.1 Access to the ports

8.1.1 Access to the framing and configuration/status ports

The framing and configuration/status ports are accessed via the processor interface, some status flags are also signalled on pins (see table 4.6). They are a single set of registers which are accessed by presenting addresses to the bus. For the full address map, see chapter 14. Most of the registers are 16 bits or less and can be accessed in a single bus cycle using 16 data pins. Access to 32 bits can be achieved in a single cycle when using the 32-bit processor interface, or with two 16-bit cycles if only 16 data pins are available.

8.1.2 Access to the data ports

Data can be read/transmitted either via the processor interface or via the token interfaces, but not both. This is controlled by the **EnableTxRx** pin.

Token interfaces

When the token interfaces are enabled, the Rx and Tx data ports are handled separately via the two 9-bit (8 data bits plus a control/data line) handshaken token interfaces.

Bus interfaces

When the token interfaces are disabled, the **TxData** and **RxData** addresses are used to put data into and get data out of the Tx and Rx FIFOs, see section 13.4 for data FIFO addresses. Accessing the FIFOs in this way is equivalent to using the token interfaces, and the two mechanisms are mutually exclusive.

Dealing with unaligned data

Unaligned data can be transmitted and received. Consider a processor with data in its memory, as illustrated in figure 8.1, which is to be transmitted as a packet via the STC101. In this example the data is shown as little-endian.

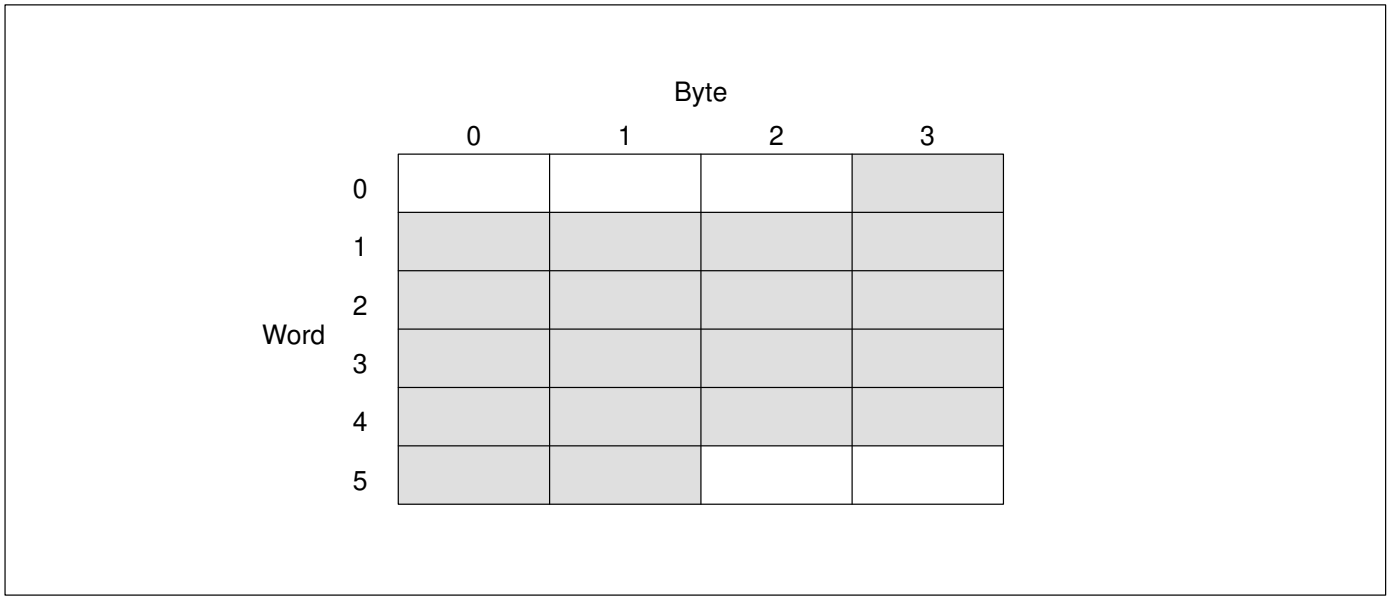


Figure 8.1 Packet in processor's memory

The dynamic width of the Tx port (assuming use of all 32 data pins) can be used to transmit the packet as follows:

- The processor loads the first byte, and stores it to the 8-bit interface to the Tx port;
- Then the processor loads four words and stores them to the 32-bit interface to the Tx port;
- Finally the processor loads the last word and stores it to the 16-bit interface to the Tx port.

In this way unaligned data can be transmitted using the minimum number of loads and stores. If the processor can load data from non word-aligned addresses, shifting can be avoided, otherwise the first word can be shifted to ensure that all valid bytes are at the bottom before storing it to the Tx port.

In a similar way, a packet can be received and stored, as illustrated in the figure, using the different width interfaces to the Rx data port.

8.2 Valid/Hold protocol of the token interfaces

The **Valid**/**Hold** protocol of the Rx and Tx token interfaces allows single cycle transfers of data between two devices.

The handshake signals have the following functions:

- **Valid** indicates the presence of valid data on the bus.
- **Hold** indicates that the data cannot be accepted by the receiving device.

The **Valid**/**Hold** protocol is defined as follows:

- Data is transferred on a rising clock edge when **Valid** is high and the corresponding **Hold** signal is low.
- **Valid** can be held high, in which case data is transferred on each rising edge when **Hold** is low.
- While the **Valid** signal is high, and the **Hold** signal is high, the data must not change.
- If the **Hold** signal is high when **Valid** is asserted, then the **Valid** signal cannot be de-asserted until the **Hold** signal has been seen low at a rising edge of the clock. Thus, **Valid** cannot be taken low again until the data has been transferred.
- The assertion of **Valid** must not depend on the value of **Hold**.
- The assertion of **Hold** must not depend on the value of **Valid**.
- At reset **Valid** must be low and **Hold** must be high.

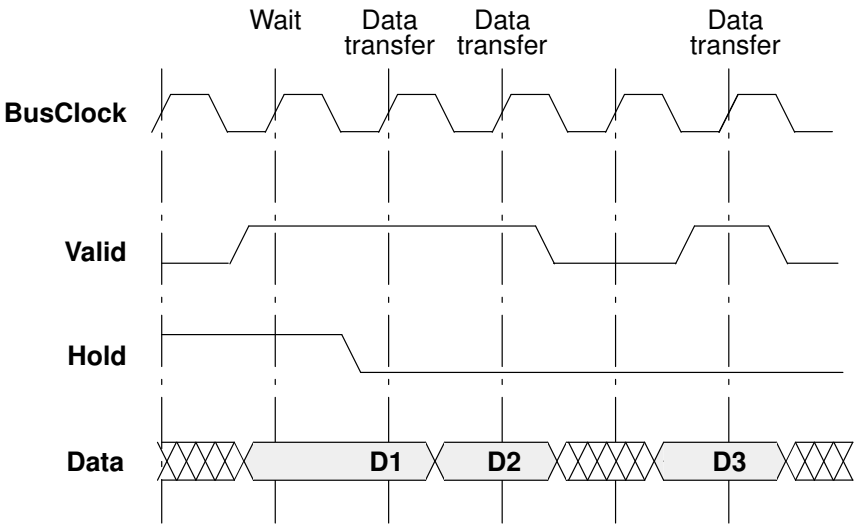


Figure 8.2 Valid/Hold protocol diagram

9 Link interface

9.1 Data/Strobe links

The STC101 DS-Link uses a protocol with two wires in each direction, one for data and one to carry a strobe signal. Hence the link is referred to as a data/strobe (DS-Link) and is capable of:

- Up to 100 Mbits/s.
- Unidirectional peak bandwidth of 10 Mbytes/s per link.
- Support for virtual channels and through routing.

The DS pair carries tokens and an encoded clock. The tokens can be data or control tokens. Figure 9.1 shows the format of data and control tokens on the data and strobe wires. Data tokens are 10 bits long and consist of a parity bit, a flag which is set to 0 to indicate a data token, and 8 bits of data. Control tokens are 4 bits long and consist of a parity bit, a flag which is set to 1 to indicate a control token, and 2 bits to indicate the type of control token.

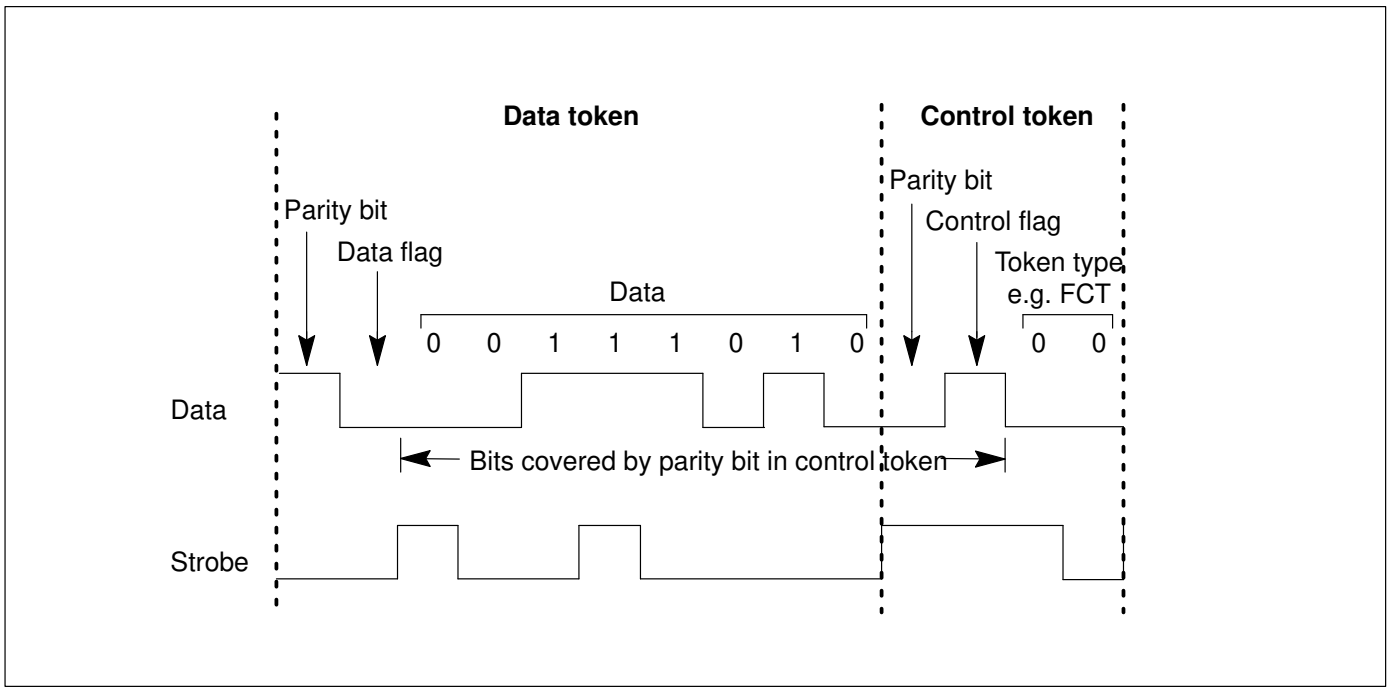


Figure 9.1 Link data and strobe formats

The DS-Link protocol ensures that only one of the two wires of the data strobe pair has an edge in each bit time. The levels on the data wire give the data bits transmitted. The strobe signal changes whenever the data signal does not. These two signals encode a clock together with the data bits, permitting asynchronous detection of the data at the receiving end.

The data and control tokens are of different lengths, for this reason the parity bit in any token covers the parity of the data or control bits in the previous token, and the data/control flag in the same token, as shown in figure 9.1. This allows single bit errors in the token type flag to be detected. **Odd** parity checking is used. Thus the parity bit is set/unset to ensure that the bits covered, inclusive of the parity bit (see figure 9.1), always contain an odd number of 1's. The coding of the tokens is shown in table 9.1. To ensure the immediate detection of parity errors and to enable link disconnection to be detected null tokens are sent in the absence of other tokens.

Token type	Abbreviation	Coding
Data token	–	P0DDDDDDDD
Flow control token	FCT	P100
End of packet	EOP	P101
End of message	EOM	P110
Escape token	ESC	P111
Null token	NUL	ESC P100

P = parity bit
D = data bit

Table 9.1 Token codings

9.2 Low-level flow control

Token-level flow control is performed in each DS-Link module, and the additional flow control tokens used are not visible to the higher-level packet protocol. The token-level flow control mechanism prevents a sender from overrunning the internal input buffer of a receiving link. The receiving link input contains a buffer for at least 8 tokens (20 bytes of buffering is in fact provided). Whenever the link input has sufficient buffering available to consume a further 8 tokens an FCT is transmitted on the associated link output, and this FCT gives the sender permission to transmit a further 8 tokens. Once the sender has transmitted a further 8 tokens it waits until it receives another FCT before transmitting any more tokens. The provision of more than 8 tokens of buffering on the link input ensures that in practice the next FCT is received before the previous block of 8 tokens has been fully transmitted, so the token-level flow control does not restrict the maximum bandwidth of the link.

9.3 Link speeds

The STC101 DS-Link can support a range of communication speeds, which are programmed by writing to the **LinkMode** register (refer to section 13.5). Only the transmission speed of a link is programmed as reception is asynchronous. This means that links running at different speeds can be connected, provided that each device is capable of receiving at the speed of the connected transmitter.

An internal 100 MHz link clock is derived from the 50 MHz logic clock. This root clock is then optionally divided (by programming the **SpeedDivide** bits) by 1, 2, 4 or 8, giving a range of speeds. This arrangement allows the DS-Link to be run at one of four transmission speeds, as shown in table 9.2.

SpeedDivide1:0		Division	Link speed (Mbits/s)
0	0	1	100
0	1	2	50
1	0	4	25
1	1	8	12.5

Table 9.2 Link speed selection

Note also that the DS-Link can be programmed to use a base rate clock of 10 MHz. At reset the DS-Link is configured to run at the base speed of 10 Mbits/s. The **SpeedSelect** bit in the **LinkMode**

register (see table 13.22) when set to 1 sets the link to the speed selected by the **SpeedDivide** bits, as opposed to the default base speed of 10 Mbits/s.

9.4 Errors on DS-Links

DS-Link inputs can detect parity and disconnection conditions as errors. The single bit odd parity system will detect single bit errors at the link token level. The protocol to transmit NUL tokens in the absence of other tokens enables disconnection of a link to be detected. A disconnection error indicates one of two things:

- the link has been physically disconnected;
- an error has occurred at the other end of the link, which has then stopped transmitting.

The **LinkError** bit in the **LinkStatus** register flags that a parity and/or disconnection error has occurred on the DS-Link. The **LinkError** bit is duplicated in the **RxInterruptStatus** register and can therefore be used to generate an interrupt. The bit fields **ParityError** and **DiscError** indicate when parity and disconnect errors occur respectively.

When a DS-Link detects a parity error on its input it halts its output. This is detected as a disconnect error at the other end of the link, causing this to halt its output also. Detection of an error causes the link to be reset. Thus, the disconnect behavior ensures that both ends are reset. Each end can then be restarted.

Note that a disconnect error is only flagged once a token has been received on a link and transmission is subsequently interrupted. Therefore when one end of a link is started up before the other end of a link, a disconnect error does not occur as no tokens have yet been received. As soon as the other end of the link is started communication can begin immediately.

The DS-Link is designed to be highly reliable within a single subsystem and can be operated in one of two environments, dependent on the level of reliability required. A DS-Link can be set to an environment in which any link errors are localized to the link. This is set by the **LocalizeError** bit in the **LinkMode** register. The **LocalizeError** bit is set on a per link basis, therefore it is possible to have some links in a system set to localize link errors and other links which are not. The consequence of a link error depends on which environment the link is in, as described below.

9.4.1 Reliable links

In the majority of applications, the communications system should be regarded as being totally reliable. In this environment errors are considered to be very rare, but are treated as being catastrophic if they do occur. This environment is the default on power-on reset, with all links having their **LocalizeError** bit set to 0. If an error occurs it will be detected and the **LinkError** bit (bit 7) of the **RxInterruptStatus** register will be set to indicate that an error has occurred. Normal practice will then be to reset the subsystem in which the error has occurred and to restart the application. Re-starting the DS-Link will unset the **LinkError** bit of the **RxInterruptStatus** register.

9.4.2 More reliable links

For some applications, for instance when a disconnect or parity error may be expected during normal operation, an even higher level of reliability is required. This level of fault tolerance is supported by localizing errors to the DS-Link. This is achieved by setting the **LocalizeError** bit in the **LinkMode** register to 1. If an error occurs packets in transit at the time of the error are discarded or truncated.

9.5 Link state on start up

After power-on **LinkData** and **LinkStrobe** signals are low, without clocks. Following power-on reset an initialization sequence sets the speed of the link clock. The DS-Link is initially inactive, with a default configuration. It is configured and started by configuration writes via the configuration/status and framing ports. Its status can be determined by configuration reads. The DS-Link must be explicitly started by writing to the **StartLink** bit of the **LinkCommand** register. When the DS-Link is started up it transmits NUL tokens. When the DS-Link starts receiving tokens it transmits 2 FCT tokens.

Data may not be transferred over the DS-Link until the receiving link has sent an FCT to signify that it has enough free buffer space to receive the data.

The receiving link receives and correctly decodes the tokens. However, only when the receiving link has been explicitly started by writing to the **StartLink** bit of the **LinkCommand** register can it send tokens back. NUL tokens are then sent until data is required.

9.6 Resetting DS-Links

If one end of a running DS-Link is reset, that end of the link stops transmitting tokens on a token boundary and any buffered data is discarded. The other end of the link detects a disconnection and also stops transmission. The reset end then also detects disconnection and clears its flow-control state and error status bits, and the link becomes insensitive to transitions on its input for 3.2 μs. In order to ensure that both ends of the link have completed reset and are sensitive to transitions before either end is started there is a further delay of 12.8 μs. Note that the Data and Strobe outputs are simply held at the values they have at the end of the last transmitted token, since forcing them to zero could be decoded as a bit by the other end of the link.

Since the disconnection protocol between the two ends of a DS-Link ensures that both ends become reset automatically if an error is detected, there is normally no reason to explicitly reset either end. However, one end may be reset as a consequence of a reset of a device or subsystem. In this case it is important to ensure that either: both ends of the link have been started before the reset occurs; or that both ends are quiet (by resetting if necessary). This is because if one end of a DS-Link is already running before the other end comes out of reset, the initial transmission of FCTs will be lost, and so the reset end will never receive permission to transmit data. Also, unless the reset end is brought out of reset precisely on a null token boundary (for which there is 1 chance in 8), it will misinterpret the bit-stream and consequently detect a parity error.

9.7 Link connections

DS-Links are TTL compatible and intended to be used in electrically quiet environments, between devices on a single printed circuit board or between two boards via a backplane. Simple connection may be made between devices separated by a distance of less than 200 mm. For longer distances matched 100 ohm transmission lines should be used with series matching resistors, see figure 9.2.

The inputs and outputs have been designed to have minimum skew at the 1.5V TTL threshold.

Buffers may be used for very long transmissions. If so, their overall propagation delay should be stable within the skew tolerance of the link, although the absolute value of the delay is immaterial.

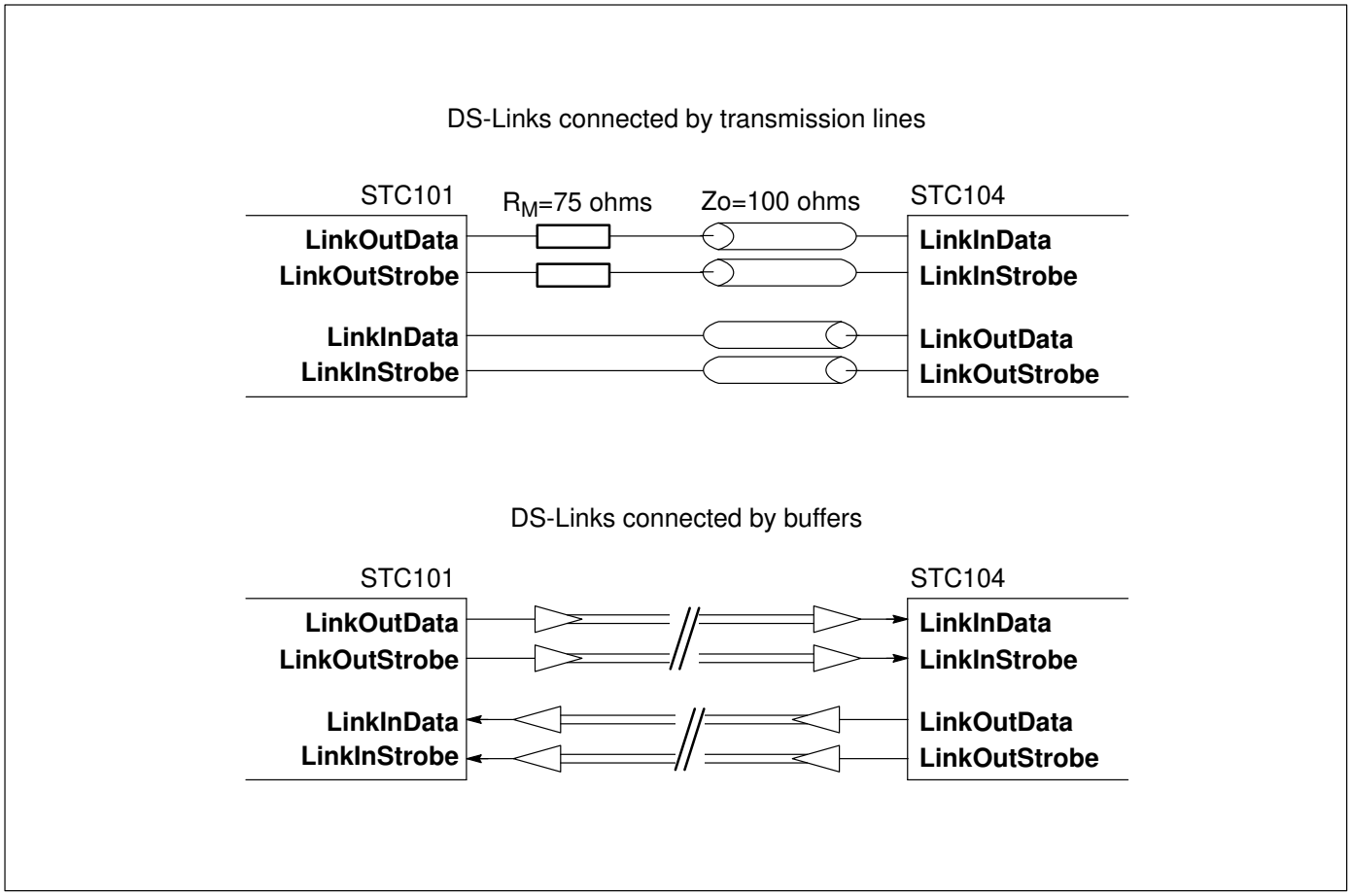


Figure 9.2 DS-Link connections

10 Interrupts

The interrupt system supported directly by the STC101 is level triggered. Thus, a bit is set in one of the interrupt status registers (**RxInterruptStatus** or **TxInterruptStatus**) whenever a condition which causes an interrupt becomes true. The bit remains set until the condition is cleared. If the corresponding bit in the appropriate interrupt enable register (**RxInterruptEnable** or **TxInterruptEnable**) is set, the interrupt will be active. The interrupt signal is active whenever any interrupt bits are set whose interrupt enable bits are also set, and bit 0 of the **EnableInterrupts** register is 1. When bit 0 of the **EnableInterrupts** register is 0, the interrupt signal is inactive, regardless of the state of the **InterruptStatus** and **InterruptEnable** registers. This is summarized in table 10.1 below. Writing 1 to the **EnableInterrupts** register returns the interrupt signal to the state determined by these registers.

InterruptEnable	InterruptStatus	EnableInterrupts bit 0	Interrupt behavior
1	1	1	Interrupt is active
0	1	1	Interrupt is disabled
1	0	1	Interrupt condition is false
x	x	0	Interrupts are disabled

Table 10.1 Interrupt settings

A condition which may be transient, for example the Tx FIFO reaching its high level when either packetization is disabled or the framing information has already been supplied (in which case the FIFO can spontaneously empty), should not be enabled as an interrupt.

Conversion to an edge triggered interrupt system can be achieved in software, by masking all interrupts when an interrupt is taken and then un-masking all appropriate ones at the end of the interrupt service routine. This can be done efficiently by setting the **EnableInterrupts** register bit to 0. This register is also cleared by asserting the **DisableInt** pin.

Note that the **EnableInterrupts** register remains cleared even when the **DisableInt** pin is de-asserted; it is set by writing 1 to the register providing the **DisableInt** pin is not asserted. The **EnableInterrupts** register cannot be set while the **DisableInt** pin is asserted.

11 Clocking

There are two externally supplied clocks on the STC101.

- **LogicClock**
  This must be 50 MHz[1]. It is a logic clock for the internals of the STC101 and is used to provide the base speed 10 MHz clock for the DS-Link.
  The internal 100 MHz link clock is derived from this 50 MHz logic clock. The 100 MHz link clock can be divided to give a range of link speeds (refer to section 9.3 for details).
- **BusClock**
  This can be any frequency up to 30 MHz. It is the bus clock, to which all signals from the STC101 on the parallel interface are synchronous (or at least synchronized).

The STC101 performs all necessary synchronizations between these clock domains.

Note that neither clock may be stopped while the device is operating, they must both be free-running.

12 Reset

The STC101 can be reset by asserting the **Reset** pin high for at least 2 cycles of the bus clock. After a reset the STC101 is in the following state: the DS-Link is in a quiescent state with a default speed of 10 MHz; all registers contain their default values (see chapter 14); and all buffers are cleared.

1. The **LogicClock** can be less than 50 MHz, however if this is the case the links will not be compatible with the standard, and the **LogicClock** must always be greater than or equal to the **BusClock**.

13 Programmable register functionality

The STC101 is controlled via registers accessed via the configuration/status and framing ports. The functionality to be controlled by the registers is described below. The tables below detail the bit fields of each of the registers and give the register address and whether the register is read only, write only, or read and writable.

Note that in the following bit field descriptions the lowest numbered bit is the least significant bit.

Note, all reserved/undefined bits of a register must always be written with 0's, unless otherwise stated.

13.1 System services registers

System services consists of registers which contain control and status information and general information on the STC101.

DeviceID

The **DeviceID** register contains a 16-bit device identification code unique to the device. The value of the device identification code for the STC101 is 336.

DeviceID		**#17** **Read only**
Bit	**Bit field**	**Function**
15:0	**DeviceID**	Device identification code.

Table 13.1 Bit fields in the **DeviceID** register

DeviceRevision

The **DeviceRevision** register contains the revision of the device.

DeviceRevision		**#18**	**Read only**
Bit	**Bit field**	**Function**	
15:0	**DeviceRev**	Device revision.	

Table 13.2 Bit fields in the **DeviceRevision** register

DeviceConfig

The **DeviceConfig** register may be programmed to set the configuration of the device.

DeviceConfig		#19 Read/Write
Bit	**Bit field**	**Function**
2:0	**RxHeaderLength**	Programs the expected length of the incoming Rx packet header (1 to 4 bytes). Only values 1 to 4 inclusive are valid. Note this value must *not* be changed after the DS-Link has been started. 001 1 byte header 010 2 byte header 011 3 byte header 100 4 byte header
4	**EnablePacketization**	When set to 1, the operation of the framing logic is enabled. If set to 0, packetization is disabled.
5	**LinkLoopBack**	When set to 1, the data and strobe outputs of the DS-Link are connected directly back to the data and strobe inputs respectively. This can be used for test purposes. The DS-Link outputs are still connected to the data and strobe output pins.
6	**SuppressRxEOX**	When set to 1, termination tokens received by the link are *not* passed into the Rx FIFO (and hence do not appear on the token interfaces, if enabled). This bit should always be set when the external token interfaces are not enabled.
15:7, 3		Reserved, write 0.

Table 13.3 Bit fields in the **DeviceConfig** register

13.2 Interrupt registers

TxInterruptStatus

The **TxInterruptStatus** register is read only and contains information about the state of the Tx FIFO.

TxInterruptStatus		#E Read/Write
Bit	**Bit field**	**Function**
0	**TxFifoLevel**	The Tx FIFO has reached its level. The setting of this bit is dependent on the setting of the **TxLevelHighNotLow** bit in the **TxLevel** register, see table 13.20. This bit is set when the **TxLevelHighNotLow** bit is 0 and the number of bytes in the Tx Fifo is less than or equal to the Tx Fifo low level. This bit is set when the **TxLevelHighNotLow** bit is 1 and the number of bytes in the Tx Fifo is greater than the Tx Fifo high level. This can be unset by taking data from the Tx data port.
1	**SendPacket**	The **TxSendPacket** register (see table 13.10) is available for writing. This bit is unset by writing to the **TxSendPacket** register.
15:2		Reserved.

Table 13.4 Bit fields in the **TxInterruptStatus** register

TxInterruptEnable

The **TxInterruptEnable** register is a read/write register which contains bits associated with the **TxInterruptStatus** register. This register controls whether the corresponding bit in the **TxInterrupt-Status** register causes an interrupt to be signalled.

TxInterruptEnable		#10	Read/Write
Bit	**Bit field**	**Function**	
0	**TxFifoLevelEnable**	If set the setting of the **TxFifoLevel** bit in the **TxInterruptStatus** register (see table 13.4) causes an interrupt to be signalled.	
1	**SendPacketEnable**	If set the setting of the **SendPacket** bit in the **TxInterruptStatus** register (see table 13.4) causes an interrupt to be signalled.	
15:2		Reserved, write 0.	

Table 13.5 Bit fields in the **TxInterruptEnable** register

RxInterruptStatus

The **RxInterruptStatus** register contains information about the state of the Rx FIFO.

Note that only one of the **ShortPkt** and **HdrValid** bits can be set at any one time. Note also that only one of the **EOPRxed**, **EOMRxed**, and **AckRxed** bits can be set at any one time. The contents of the **RxPacketHeaderLower** and **RxPacketHeaderUpper** registers are only valid when the **HdrValid** bit is set. If either of the **ShortPkt** or **AckRxed** bits are set the **RxPacketLength** register will have the value zero. The **AckRxed** bit is only set if a complete header has been received, followed immediately by an EOP, therefore it cannot be set at the same time as the **ShortPkt** bit.

RxInterruptStatus		#F	Read only
Bit	**Bit field**	**Function**	
0	**ShortPkt**	A packet has been received with too few bytes of header. This can be unset by writing to the **AckShortPkt** bit of the **RxAcknowledge** register (see table 13.8).	
1	**HdrValid**	A complete packet header has been received. This can be unset by writing to the **AckHdrValid** bit of the **RxAcknowledge** register.	
2	**EOPRxed**	A non-empty packet has been received, terminated by EOP. This can be unset by writing to the **AckEOPRxed** bit of the **RxAcknowledge** register.	
3	**EOMRxed**	A packet has been received, terminated by EOM. This can be unset by writing to the **AckEOMRxed** bit of the **RxAcknowledge** register.	
4	**AckRxed**	An empty packet has been received, terminated by EOP. This can be unset by writing to the **AckAckRxed** bit of the **RxAcknowledge** register.	
5	**CountOverflow**	The Rx packet counter has overflowed. This can be unset by writing to the **AckCountOverflow** bit of the **RxAcknowledge** register.	
6	**RxFifoLevel**	The Rx FIFO has reached its level. The setting of this bit is dependent on the setting of the **RxLevelHighNotLow** bit in the **RxLevel** register, see table 13.21. This bit is set when the **RxLevelHighNotLow** bit is 0 and the number of bytes in the Rx Fifo is less than or equal to the Rx Fifo low level. This bit is set when the **RxLevelHighNotLow** bit is 1 and the number of bytes in the Rx Fifo is greater than the Rx Fifo high level. This can be unset by taking data from the Rx data port.	
7	**LinkError**	An error has been detected by the DS-Link. This can be unset by re-starting the DS-Link.	
15:8		Reserved.	

Table 13.6 Bit fields in the **RxInterruptStatus** register

RxInterruptEnable

The **RxInterruptEnable** register contains bits associated with the **RxInterruptStatus** register. This register controls whether the corresponding bit in the **RxInterruptStatus** register causes an interrupt to be signalled.

RxInterruptEnable	#11	Read/Write
Bit	**Bit field**	**Function**
0	**ShortPktEnable**	If set the setting of the **ShortPkt** bit in the **RxInterruptStatus** register (see table 13.6) causes an interrupt to be signalled.
1	**HdrValidEnable**	If set the setting of the **HdrValid** bit in the **RxInterruptStatus** register causes an interrupt to be signalled.
2	**EOPRxedEnable**	If set the setting of the **EOPRxed** bit in the **RxInterruptStatus** register causes an interrupt to be signalled.
3	**EOMRxedEnable**	If set the setting of the **EOMRxed** bit in the **RxInterruptStatus** register causes an interrupt to be signalled.
4	**AckRxedEnable**	If set the setting of the **AckRxed** bit in the **RxInterruptStatus** register causes an interrupt to be signalled.
5	**CountOverflowEnable**	If set the setting of the **CountOverflow** bit in the **RxInterruptStatus** register causes an interrupt to be signalled.
6	**RxFifoLevelEnable**	If set the setting of the **RxFifoLevel** bit in the **RxInterruptStatus** register causes an interrupt to be signalled.
7	**LinkErrorEnable**	If set the setting of the **LinkError** bit in the **RxInterruptStatus** register causes an interrupt to be signalled.
15:8		Reserved, write 0.

Table 13.7 Bit fields in the **RxInterruptEnable** register

RxAcknowledge

The **RxAcknowledge** register contains bits associated with the **RxInterruptStatus** register (see table 13.6). This register can be used to reset to 0 the corresponding bit in the **RxInterruptStatus** register, allowing it to be set to 1 again by the recurrence of the condition which sets it.

Writing 1 to any bit causes the corresponding bit in the **RxInterruptStatus** register to be reset within 2 bus cycles beyond the end of the write cycle. The **Status** bit will be low for at least one cycle.

Note that any number of bits may be written to at once.

RxAcknowledge	#12	Write only
Bit	**Bit field**	**Function**
0	**AckShortPkt**	When set to 1, resets the **ShortPkt** bit in the **RxInterruptStatus** register.
1	**AckHdrValid**	When set to 1, resets the **HdrValid** bit in the **RxInterruptStatus** register.
2	**AckEOPRxed**	When set to 1, resets the **EOPRxed** bit in the **RxInterruptStatus** register.
3	**AckEOMRxed**	When set to 1, resets the **EOMRxed** bit in the **RxInterruptStatus** register.
4	**AckAckRxed**	When set to 1, resets the **AckRxed** bit in the **RxInterruptStatus** register.
5	**AckCountOverflow**	When set to 1, resets the **CountOverflow** bit in the **RxInterruptStatus** register.
15:6		Reserved, write 0.

Table 13.8 Bit fields in the **RxAcknowledge** register

EnableInterrupts

The **EnableInterrupts** register is a read/write register.

This register is cleared by asserting the **DisableInt** pin. Note that the register remains clear even when the pin is de-asserted; it can be set by writing 1 to it providing the **DisableInt** pin is de-asserted.

EnableInterrupts	#13	Read/Write
Bit	**Bit field**	**Function**
0	**EnableInterrupts**	When set to 1, enables the interrupt signal to be active. When it is 0 the interrupt signal is inactive, regardless of the state of the **InterruptStatus** and **InterruptEnable** registers. Writing 1 to it returns the interrupt signal to the state determined by these registers.
15:1		Reserved, write 0.

Table 13.9 Bit fields in the **EnableInterrupts** register

13.3 Framing data registers

Note that in transparent mode these framing data registers are all irrelevant.

13.3.1 Tx framing registers

When the STC101 is operating in packetizing mode, the information regarding the length, headers and terminators of packets is supplied separately from the packet data. This information is set up in registers, as described below.

TxSendPacket register

The **TxSendPacket** register is a 16-bit read/write register. When the **SendPacket** bit of the **TxInterruptStatus** register (see table 13.4) is 1, the framing information for a packet can be generated by writing to this register. The values written to the various bit fields determine the framing information.

When the framing information has been accepted, the **SendPacket** bit of the **TxInterruptStatus** register is set to 1 and another set of framing information can be provided. Note that the **TxPacketHeader0-1** and the **TxHeaderLength0-1** registers can be written to whilst the **SendPacket** bit is not set without affecting the framing information for the packet to be transmitted.

Note that writes should only be made to this register when the **SendPacket** bit is 1. Note also that the return of the **SendPacket** bit to 1 can be enabled as an interrupt, see table 13.4.

TxSendPacket	#4	Read/Write
Bit	**Bit field**	**Function**
11:0	**PacketLength**	Determines the length of the packet (which may be 0). Note that packets which are longer than the maximum allowed by the length of this field may be transmitted in several parts; see section 6.2.1.
12	**HeaderEnable**	When set to 1, a header is added to the packet.
13	**HeaderSelect**	Determines which of the two possible **TxPacketHeader0-1** registers is used to provide the header, as follows: If 0, the **TxPacketHeader0** and **TxHeaderLength0** registers are used; If 1, the **TxPacketHeader1** and **TxHeaderLength1** registers are used.
14	**TerminatorEnable**	When set to 1, a termination token is sent at the end of the packet. If it is 1, the **TerminatorType** bit determines the terminator type (EOP or EOM).
15	**TerminatorType**	If a terminator token is sent at the end of a packet, this bit determines whether an EOP or EOM terminator token is sent, as follows: If 1, an EOP is sent; If 0, an EOM is sent.

Table 13.10 Bit fields in the **TxSendPacket** register

TxPacketHeaderLower0-1

The **TxPacketHeaderLower0** and **TxPacketHeaderLower1** registers are read/write registers which contain the value of a header to be added to a packet transmitted from the link. If the bus is operating as 32-bits wide (as determined by the **Bus32not16** pin) these are 32-bit registers which contain the whole headers, and are the same as the **TxPacketHeaderUpper0-1** registers. Otherwise these are 16-bit read/write registers which contain the least significant two bytes of the header. The presence of these shorter registers allows the setting of headers in systems using a 16-bit bus to access the framing ports. The number of bytes of these registers which are valid is recorded in the corresponding **TxHeaderLength** registers. These registers are little-endian, that is byte 0 is the first byte transmitted, byte 1 the next byte.

TxPacketHeaderLower0		**#5**	**Read/Write**
TxPacketHeaderLower1		**#7**	**Read/Write**
Bit	**Bit field**	**Function**	
31:0 or 15:0	**TxHeaderLower**	Contains the packet header for 32-bit wide bus operation. Contains the least significant 2 bytes of the header for 16-bit wide bus operation.	

Table 13.11 Bit fields in the **TxPacketHeaderLower0-1** registers

TxPacketHeaderUpper0-1

The **TxPacketHeaderUpper0** and **TxPacketHeaderUpper1** registers are read/write registers which contain the value of a header to be added to a transmitted packet from **Link0**. If the bus is operating as 32-bits wide (as determined by the **Bus32not16** pin) these are 32-bit registers which contain the whole headers, and are the same as the **TxPacketHeaderLower0-1** registers. Otherwise these are 16-bit read/write registers which contain the most significant two bytes of the header. The presence of these registers allows the setting of long headers in systems using a 16-bit bus to access the framing ports. The number of bytes of these registers which are valid is recorded in the corresponding **TxHeaderLength** registers (see table 13.13). These registers are little-endian, that is byte 0 is the first byte transmitted, byte 1 the next byte.

TxPacketHeaderUpper0		**#6**	**Read/Write**
TxPacketHeaderUpper1		**#8**	**Read/Write**
Bit	**Bit field**	**Function**	
31:0 or 15:0	**TxHeaderUpper**	Contains the packet header for 32-bit wide bus operation. Contains the most significant 2 bytes of the header for 16-bit wide bus operation.	

Table 13.12 Bit fields in the **TxPacketHeaderUpper0-1** registers

TxHeaderLength0-1

The **TxHeaderLength0** and **TxHeaderLength1** registers are 16-bit read/write registers.

TxHeaderLength0		#C	Read/Write
TxHeaderLength1		#D	Read/Write
Bit	**Bit field**	**Function**	
2:0	**TxHeaderLength**	Programs the number of bytes of the corresponding **TxPacketHeader** register to be used. This number can be 1 to 4 inclusive. Other values are not valid. 001 1 byte 010 2 byte 011 3 byte 100 4 byte	
15:3		Reserved, write 0.	

Table 13.13 Bit fields in the **TxHeaderLength0-1** registers

TxPacketAbort register

The **TxPacketAbort** register is a write-only register.

TxPacketAbort		#16	Write only
Bit	**Bit field**	**Function**	
0	**PacketAbort**	Writing a 1 to this bit causes an EOM token to be sent to the DS-Link provided any data bytes have been sent to the DS-Link since the last EOP or EOM token. This enables a transmitted packet to be terminated (thereby closing a path it may have opened across a network) even if the supply of data fails.	
15:1		Reserved, write 0.	

Table 13.14 Bit fields in the **TxPacketAbort** register

13.3.2 Rx framing registers

When packetization is enabled, the information regarding the length, headers and terminators of incoming packets is separate to the data part of the packets received and is contained in registers.

RxPacketHeaderLower

If the bus is operating as 32-bits wide (as determined by the **Bus32not16** pin) this is a 32-bit read-only register which contains the value of the header of the most recently received packet, if any, and is the same as the **RxPacketHeaderUpper** register. Otherwise this is a 16-bit read-only register which contains the least significant two bytes of the header. This is provided to allow the receipt of packet headers in systems using a 16-bit bus to access the framing ports. The number of bytes of this register which are valid is recorded in the **RxHeaderLength** field of the **DeviceConfig** register. The header is little-endian, that is byte 0 is the first byte transmitted, byte 1 the next byte.

The validity of the value in the **RxPacketHeaderLower** register is indicated by the **HdrValid** bit of the **RxInterruptStatus** register. This register should only be read when the **HdrValid** bit is set.

Writing a 1 to the **AckHdrValid** bit of the **RxAcknowledge** register unsets the **HdrValid** bit of the **RxInterruptStatus** register, and 'pops' the header value, allowing it to be replaced with the header of the next packet.

RxPacketHeaderLower		#9 Read only
Bit	**Bit field**	**Function**
31:0 or 15:0	**RxHeaderLower**	For 32-bit wide bus operation, contains the value of the packet header of the most recently received packet. For 16-bit wide bus operation, contains the least significant 2 bytes of the header.

Table 13.15 Bit fields in the **RxPacketHeaderLower** register

RxPacketHeaderUpper

If the bus is operating as 32-bits wide (as determined by the **Bus32not16** pin) this is a 32-bit read-only register which contains the value of the header of the most recently received packet, if any, and is the same as the **RxPacketHeaderLower** register. Otherwise this is a 16-bit read-only register which contains the most significant two bytes of the header. This is provided to optimize the receipt of packets with more than two bytes of header in systems using a 16-bit bus to access the framing ports. The number of bytes of this register which are valid is recorded in the **RxHeaderLength** field of the **DeviceConfig** register. The header is little-endian, that is byte 0 is the first byte transmitted, byte 1 the next byte.

The validity of the value in the **RxPacketHeaderUpper** register is indicated by the **HdrValid** bit of the **RxInterruptStatus** register and should only be read when this bit is set.

Writing a 1 to the **AckHdrValid** bit of the **RxAcknowledge** register unsets the **HdrValid** bit of the **RxInterruptStatus** register, and 'pops' the header value, allowing it to be replaced with the header of the next packet.

RxPacketHeaderUpper		#A Read only
Bit	**Bit field**	**Function**
31:0 or 15:0	**RxHeaderUpper**	For 32-bit wide bus operation, contains the value of the packet header of the most recently received packet. For 16-bit wide bus operation, contains the most significant 2 bytes of the header.

Table 13.16 Bit fields in the **RxPacketHeaderUpper** register

RxPacketLength

The **RxPacketLength** register contains the length of the most recently received packet (not including the header or terminator). Note that the length may be zero.

The contents of this register are valid if one of the **EOPRxed**, **EOMRxed** or **AckRxed**[2] bits of the **RxInterruptStatus** register are set, and should only be read if this is the case. The set bit is cleared by writing 1 to the corresponding bit of the **RxAcknowledge** register. This also 'pops' the length value and allows it to be replaced with the length of the next packet.

RxPacketLength		#B Read only
Bit	**Bit field**	**Function**
11:0 15:12	**RxPacketLength** 	Contains the length of the most recently received packet (not including the header or terminator). Reserved.

Table 13.17 Bit fields in the **RxPacketLength** register

2. If the **AckRxed** bit of the **RxInterruptStatus** register is set, the contents of the **RxPacketLength** register will always be zero.

13.4 FIFO registers

TxData and RxData

The **TxData** and **RxData** addresses are used when the token interfaces are disabled. Reading from, or writing to, these addresses gets data out of, or puts data into, the Rx or Tx FIFO. Addresses are word addresses where the word width is dependent on the **Bus32not16** pin. The **Tx/RxData24/32Bit** registers are disabled in 16-bit mode. All of the **Tx/RxData** registers are disabled if the token interfaces are enabled. Accessing the FIFOs in this way is equivalent to using the token interfaces, and the two mechanisms are mutually exclusive.

Bit	Register	Function
7:0	**TxData8**	Writing to this address puts 8 bits of data into the Tx FIFO.
15:0	**TxData16**	Writing to this address puts 16 bits of data into the Tx FIFO.
23:0	**TxData24**	Writing to this address puts 24 bits of data into the Tx FIFO.
31:0	**TxData32**	Writing to this address puts 32 bits of data into the Tx FIFO.

Table 13.18 TxData

Bit	Register	Function
7:0	**RxData8**	Reading from this address gets 8 bits of data out of the Rx FIFO.
15:0	**RxData16**	Reading from this address gets 16 bits of data out of the Rx FIFO.
23:0	**RxData24**	Reading from this address gets 24 bits of data out of the Rx FIFO.
31:0	**RxData32**	Reading from this address gets 32 bits of data out of the Rx FIFO.

Table 13.19 RxData

TxLevel

The **TxLevel** register contains the value of the Tx FIFO level. The direction of the level is determined by the **TxLevelHighNotLow** bit. See section 7.1 for further details on the Tx FIFO level.

Note this register should not be written to while the corresponding Tx interrupt is enabled, as it may cause spurious interrupts.

TxLevel		**#14** **Read/Write**
Bit	**Bit field**	**Function**
0	**TxLevelHighnotLow**	Determines whether the Tx FIFO level (given in the **TxLevel** bit field) is a high or low level. When 0, the **TxFifoLevel** bit of the **TxInterruptStatus** register is set when the number of bytes in the Tx FIFO is less than or equal to the Tx level. When 1, the **TxFifoLevel** bit of the **TxInterruptStatus** register is set when the number of bytes in the Tx FIFO is greater than the Tx level.
6:1	**TxLevel**	Tx FIFO level marker value.
15:7		Reserved, write 0.

Table 13.20 Bit fields in the **TxLevel** register

RxLevel

The **RxLevel** register contains the value of the Rx FIFO level. The direction of the level is determined by the **RxLevelHighNotLow** bit. See section 7.1 for further details on the Rx FIFO level.

Note this register should not be written to while the corresponding Rx interrupt is enabled, as it may cause spurious interrupts.

RxLevel		**#15**	**Read/Write**
Bit	**Bit field**	**Function**	
0	**RxLevelHighnotLow**	Determines whether the Rx FIFO level (given in the **RxLevel** bit field) is a high or low level. When 0, the **RxFifoLevel** bit of the **RxInterruptStatus** register is set when the number of bytes in the Rx FIFO is less than or equal to the Rx level. When 1, the **RxFifoLevel** bit of the **RxInterruptStatus** register is set when the number of bytes in the Rx FIFO is greater than the Rx level.	
6:1	**RxLevel**	Rx FIFO level marker value.	
15:7		Reserved, write 0.	

Table 13.21 Bit fields in the **RxLevel** register

13.5 DS-Link registers

The DS-Link interface on the STC101 is controlled via registers, the **LinkMode** register, **LinkCommand** register and **LinkStatus** register.

LinkMode

The **LinkMode** register may be re-programmed before or after the DS-Link has been started.

LinkMode		**#1A**	**Read/Write**
Bit	**Bit field**	**Function**	
1:0	**SpeedDivide**	Sets the DS-Link to transmit at one of four link speeds (see section 9.3). **SpeedDivide1:0** — **Link speed (Mbits/s)** 00 — 100 01 — 50 10 — 25 11 — 12.5	
2	**SpeedSelect**	When set to 0 the DS-Link transmits at 10 Mbits, when set to 1 the DS-Link transmits at the speed determined by the **SpeedDivide** bits.	
3	**LocalizeError**	When set to 1, if a link error occurs it is localized to the DS-Link (see section 9.4) and packets in transit at the time of the error are discarded or truncated. When set to 0, if a link error occurs communication on the DS-Link stops until the link is reset.	
4	**1 (RESERVED)**	This bit must be written as 1.	
15:5		Reserved, write 0.	

Table 13.22 Bit fields in the **LinkMode** register

LinkCommand

The **LinkCommand** register contains four bits which when set cause a specific action to be taken by the DS-Link.

LinkCommand		#1B	Write only
Bit	**Bit field**	**Function**	
0	**ResetLink**	Resets the link engine of the DS-Link. The token state is reset, the flow control credit is set to zero, the buffers are marked as empty, and the parity state is reset.	
1	**StartLink**	When a transition from 0 to 1 occurs the DS-Link will be initialized and commence operation.	
2	**ResetOutput**	Sets both Data and Strobe outputs of the DS-Link low.	
3	**WrongParity**	The DS-Link output will generate incorrect parity. This may be used to force a parity error on the device at the other end of the DS-Link.	
15:4		Reserved, write 0.	

Table 13.23 Bit fields in the **LinkCommand** register

LinkStatus

The **LinkStatus** register contains information about the state of the DS-Link.

LinkStatus		#1C	Read only
Bit	**Bit field**	**Function**	
0	**LinkError**	Flags that an error has occurred on the DS-Link.	
1	**LinkStarted**	Flags that the output DS-Link has been started and no errors have been detected.	
2	**ResetOutputComplete**	Flags that **ResetOutput** has completed on the DS-Link.	
3	**ParityError**	Flags that a parity error has occurred on the DS-Link.	
4	**DiscError**	Flags that a disconnect error has occurred on the DS-Link.	
5	**TokenReceived**	Flags that a token has been seen on the DS-Link since **ResetLink**.	
15:6		Reserved.	

Table 13.24 Bit fields in the **LinkStatus** register

14 Address map

All of the addresses are word addresses where the word width is dependent on the **Bus32not16** pin. For the three 32-bit registers **TxPacketHeader0-1** and **RxPacketHeader** these will be written/read 32 bits at a time at either of their lower/upper addresses when in 32-bit bus mode and 16 bits at a time when in 16-bit mode.

Name	Address	Reset value	Notes
TxData8Bit	0 (W)	–	disabled if token interfaces enabled
TxData16Bit	1 (W)	–	disabled if token interfaces enabled
TxData24Bit	2 (W)	–	disabled if token interfaces enabled disabled in 16-bit mode
TxData32Bit	3 (W)	–	disabled if token interfaces enabled disabled in 16-bit mode
RxData8Bit	0 (R)	–	disabled if token interfaces enabled
RxData16Bit	1 (R)	–	disabled if token interfaces enabled
RxData24Bit	2 (R)	–	disabled if token interfaces enabled disabled in 16-bit mode
RxData32Bit	3 (R)	–	disabled if token interfaces enabled disabled in 16-bit mode
TxSendPacket	4	0	
TxPacketHeaderLower0	5	0	
TxPacketHeaderUpper0	6	0	
TxPacketHeaderLower1	7	0	
TxPacketHeaderUpper1	8	0	
RxPacketHeaderLower	9	0	
RxPacketHeaderUpper	A	0	
RxPacketLength	B	0	
TxHeaderLength0	C	0	
TxHeaderLength1	D	0	
TxInterruptStatus	E	3	
RxInterruptStatus	F	0	
TxInterruptEnable	10	0	
RxInterruptEnable	11	0	
RxAcknowledge	12	0	
EnableInterrupts	13	0	
TxLevel	14	0 (empty)	
RxLevel	15	127 (full)	
TxPacketAbort	16	0	
DeviceID	17	Device ID	
DeviceRevision	18	Device revision	
DeviceConfig	19	0	
LinkMode	1A	0	
LinkCommand	1B	0	
LinkStatus	1C	0	

Table 14.1 STC101 address map and reset state

(R) and (W) mean accessed during a read or a write cycle.

15 Timing specifications

15.1 Clock timings

Symbol	Parameter	Min	Nom	Max	Units	Notes
tBCHBCH	**BusClock** period	33.0			ns	1
tBCHBCL	**BusClock** pulse width high	5.0			ns	
tBCLBCH	**BusClock** pulse width low	5.0			ns	
tBCf	**BusClock** fall time			10.0	ns	2
tBCr	**BusClock** rise time			10.0	ns	2
tLCHLCH	**LogicClock** period	20.0			ns	1
tLCHLCL	**LogicClock** pulse width high	5.0			ns	
tLCLLCH	**LogicClock** pulse width low	5.0			ns	
tLCf	**LogicClock** fall time			10.0	ns	2
tLCr	**LogicClock** rise time			10.0	ns	2

Table 15.1 **BusClock** and **LogicClock** timings

Notes

1 Measured between corresponding points on consecutive falling edges.

2 Clock transitions must be monotonic within the range VIH to VIL (refer to Electrical specifications chapter).

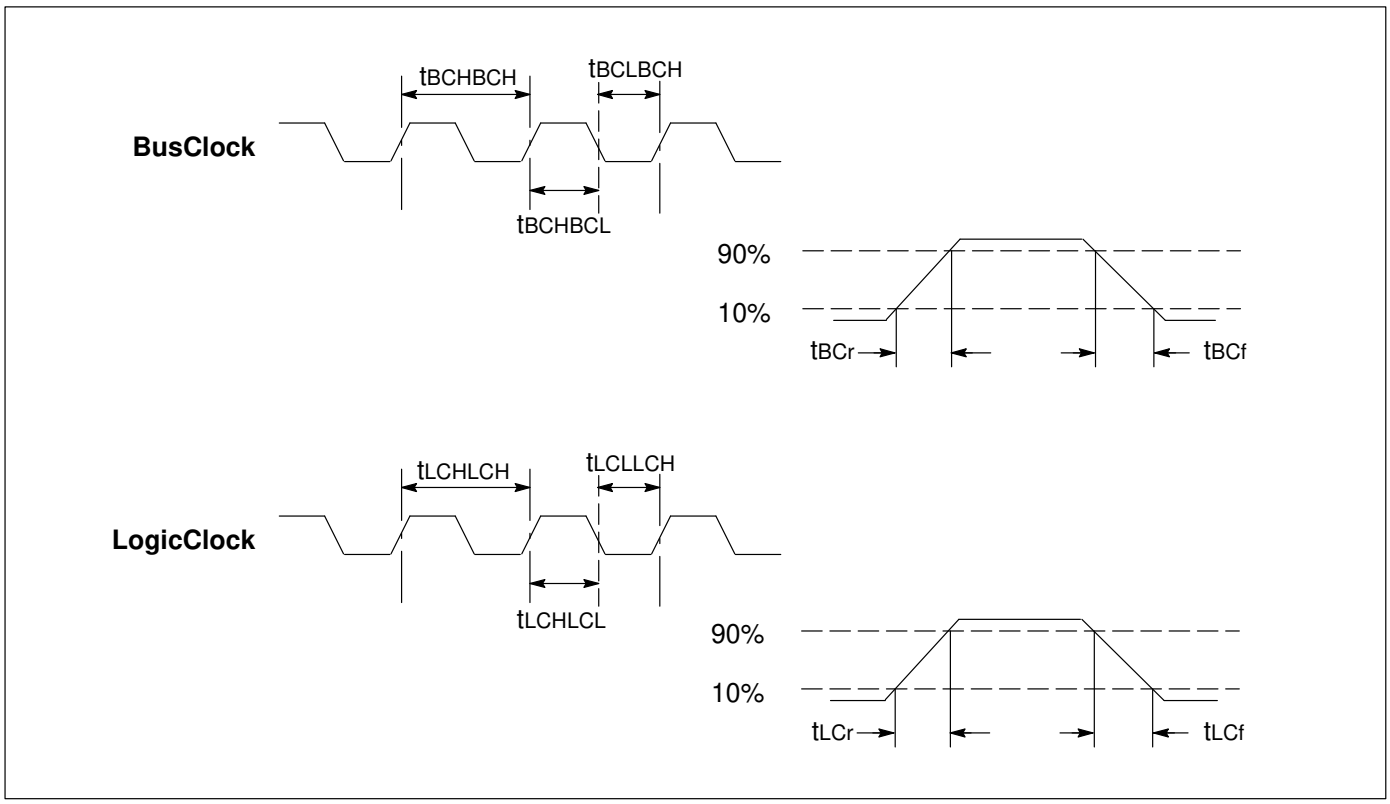


Figure 15.1 **BusClock** and **LogicClock** timings

15.2 Bus interface timings

The timing for the bus interface is given in sections 15.2.1 and 15.2.2, one for the asynchronous and one for the synchronous bus modes.

Address latch enable

The following section describes the relationship between the **ALE** signal and the other signals involved in latching the address.

In asynchronous mode the address is latched on one of the following conditions:

- if **notCS** is high, the address is latched on the falling edge of **ALE**.
- if **notCS** goes low while **ALE** is high, the address is latched on the falling edge of **notCS**.

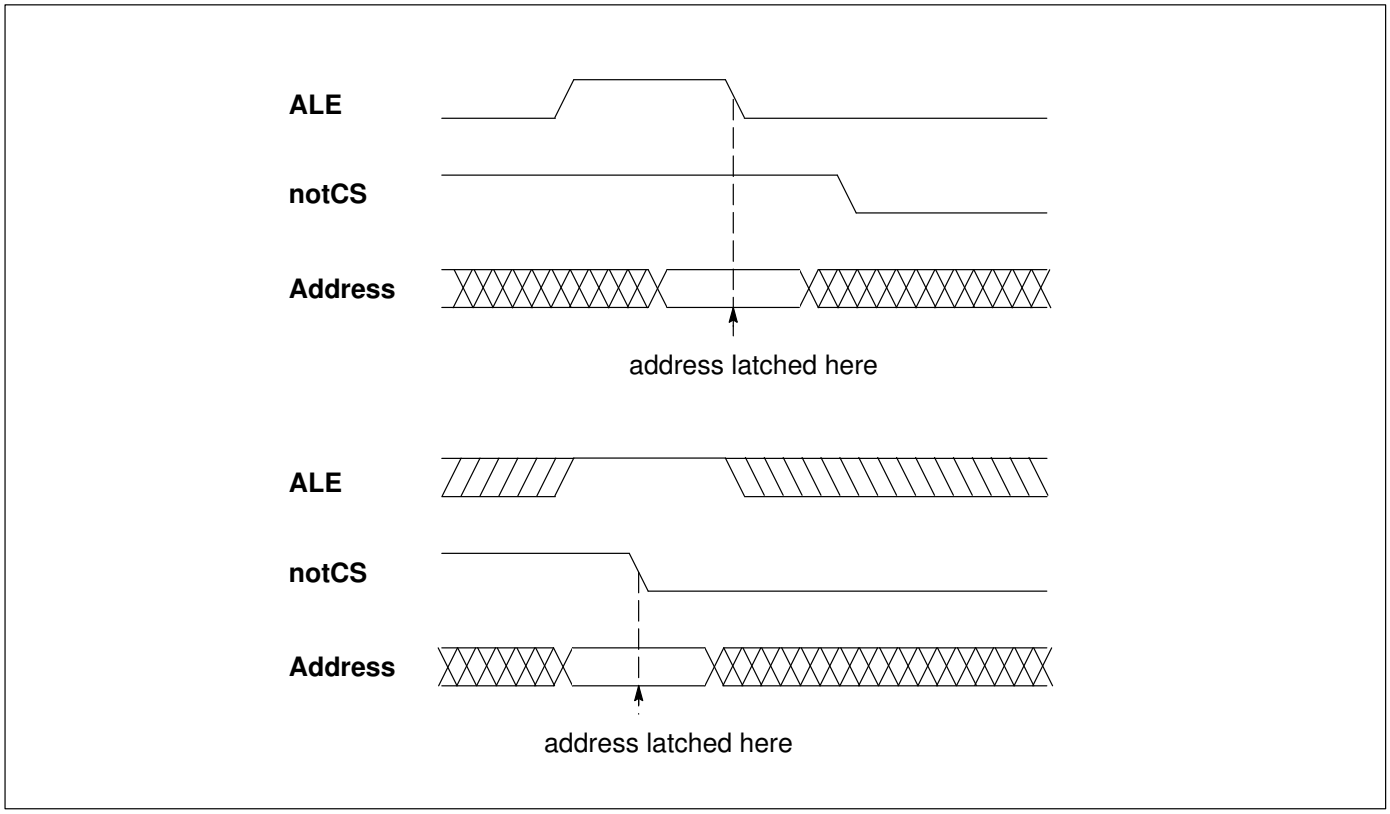


Figure 15.2 Address latching in asynchronous mode

In synchronous mode the address is latched on a rising clock edge, if that occurs while **ALE** is high.

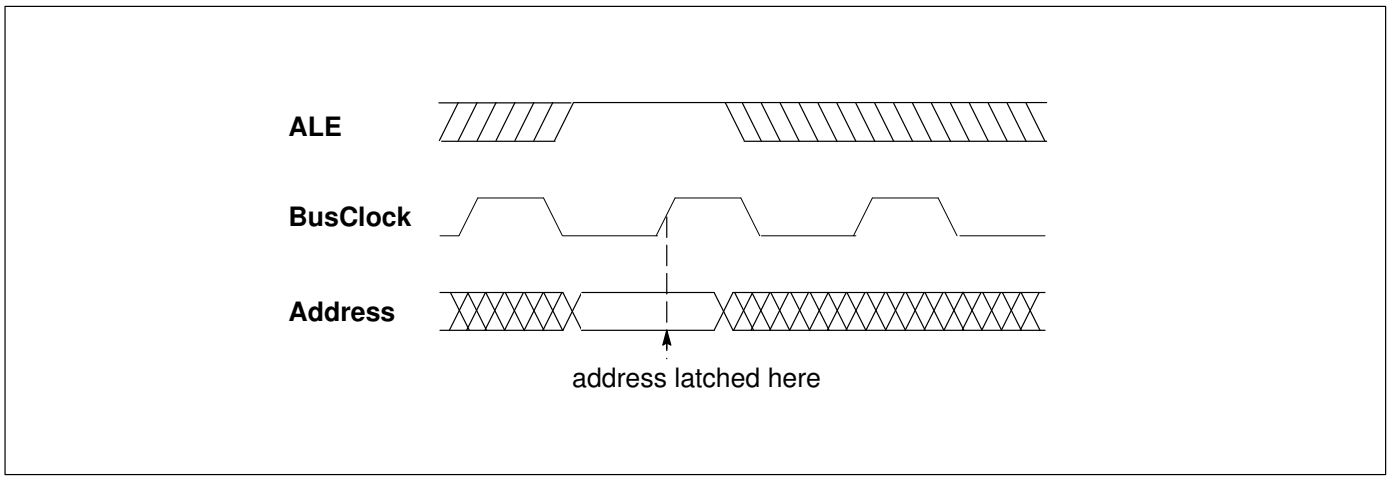


Figure 15.3 Address latching in synchronous mode

15.2.1 Asynchronous bus timings

Accesses start when **notCS** goes low and finish when **notCS** goes high again. **ALE** can be left high for non-multiplexed bus systems with address signals that are valid throughout the access.

Since the **BusWait** signal is synchronized back to the **BusClock** this avoids the use of an external synchronizer between the STC101 and the external processor. It also allows the timing of the **BusWait** signal to be modified by a simple state machine running from the **BusClock** without incurring a synchronizer delay.

Asynchronous read cycle

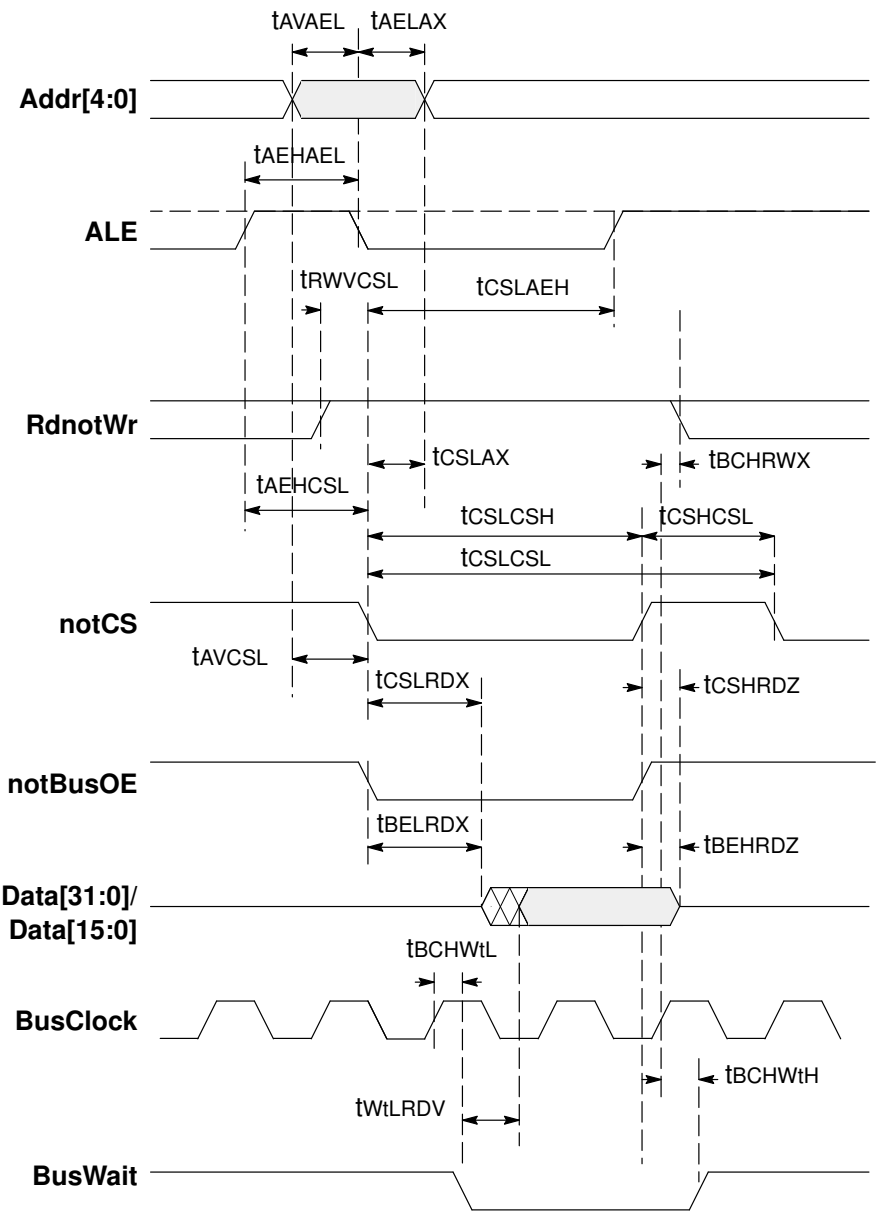


Figure 15.4 Bus interface – asynchronous read cycle

Symbol	Parameter	Min	Nom	Max	Units	Notes
tAEHAEL	**ALE** pulse width high	5.0			ns	
tAEHCSL	**ALE** high to **notCS** low	2.0			ns	
tAELAX	**ALE** low to Address hold	3.0			ns	
tAVAEL	Address valid to **ALE** low	2.0			ns	
tAVCSL	Address valid to **notCS** low	2.0			ns	
tBCHRWX	**BusClock** high to **RdnotWr** hold	5.0			ns	
tBCHWtH	**BusClock** high to **BusWait** high	25.0			ns	
tBCHWtL	**BusClock** high to **BusWait** low	25.0			ns	
tBEHRDZ	**notBusOE** high to Read **Data** tristate			15.0	ns	
tBELRDX	**notBusOE** low to Read **Data** low Z	3.0			ns	
tCSHCSL	**notCS** pulse width high	2			cycles	
tCSHRDZ	**notCS** high to Read **Data** tristate			15.0	ns	
tCSLAEH	**notCS** low to **ALE** high	5.0			ns	
tCSLAX	**notCS** low to Address hold	3.0			ns	
tCSLRDX	**notCS** low to Read **Data** low Z	3.0			ns	
tCSLCSH	**notCS** pulse width low	2			cycles	
tCSLCSL	**notCS** period	4			cycles	
tRWVCSL	Read valid to **notCS** low	2.0			ns	
tWtLRDV	**BusWait** low to Read **Data** valid			8.0	ns	

Table 15.2 Bus interface – asynchronous read cycle timings

Asynchronous write cycle

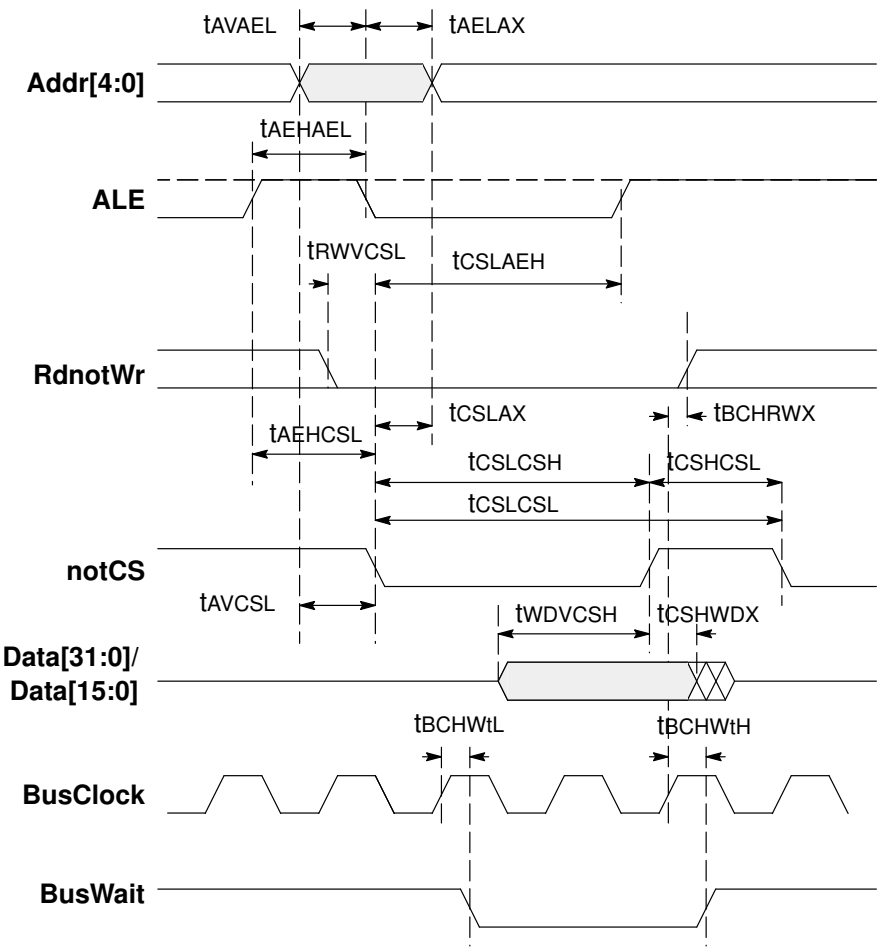


Figure 15.5 Bus interface – asynchronous write cycle

Symbol	Parameter	Min	Nom	Max	Units	Notes
tAEHAEL	**ALE** pulse width high	5.0			ns	
tAEHCSL	**ALE** high to **notCS** low	2.0			ns	
tAELAX	**ALE** low to Address hold	3.0			ns	
tAVAEL	Address valid to **ALE** low	2.0			ns	
tAVCSL	Address valid to **notCS** low	2.0			ns	
tBCHRWX	**BusClock** high to **RdnotWr** hold	5.0			ns	
tBCHWtH	**BusClock** high to **BusWait** high	25.0			ns	
tBCHWtL	**BusClock** high to **BusWait** low	25.0			ns	
tCSHCSL	**notCS** pulse width high	2			cycles	
tCSHWDX	**notCS** high to Write **Data** hold	4.0			ns	
tCSLAEH	**notCS** low to **ALE** high	5.0			ns	
tCSLAX	**notCS** low to Address hold	3.0			ns	
tCSLCSH	**notCS** pulse width low	2			cycles	
tCSLCSL	**notCS** period	4			cycles	
tWDVCSH	Write **Data** valid to **notCS** high	2.0			ns	
tRWVCSL	Write valid to **notCS** low	–2.0			ns	

Table 15.3 Bus interface – asynchronous write cycle timings

15.2.2 Synchronous bus timings

Accesses start on the rising edge of **BusClock** when **notCS** is low and finish on the rising edge of **BusClock** when **BusWait** is low. All synchronous accesses take at least two **BusClock** cycles, **BusWait** is always high during the first clock cycle. In systems in which address signals are valid throughout the access **ALE** can be left high, otherwise the addresses are latched on the rising edge of **BusClock** when **ALE** is high. **ALE** causes the address to be latched and does not imply that an access is about to start.

All signals, with the exception of the **notBusOE** and read data signals, are synchronous to the **BusClock**.

Transfers occur on all rising **BusClock** edges when **notCS** is low and **BusWait** is low. Read data is available in the clock cycle when the **BusWait** signal goes low and is setup to the next rising edge of the clock assuming **notBusOE** is low. If **notBusOE** is held low after a read ends (i.e. after **notCS** goes high) then the read data will be held on the bus, until the next access is initiated.

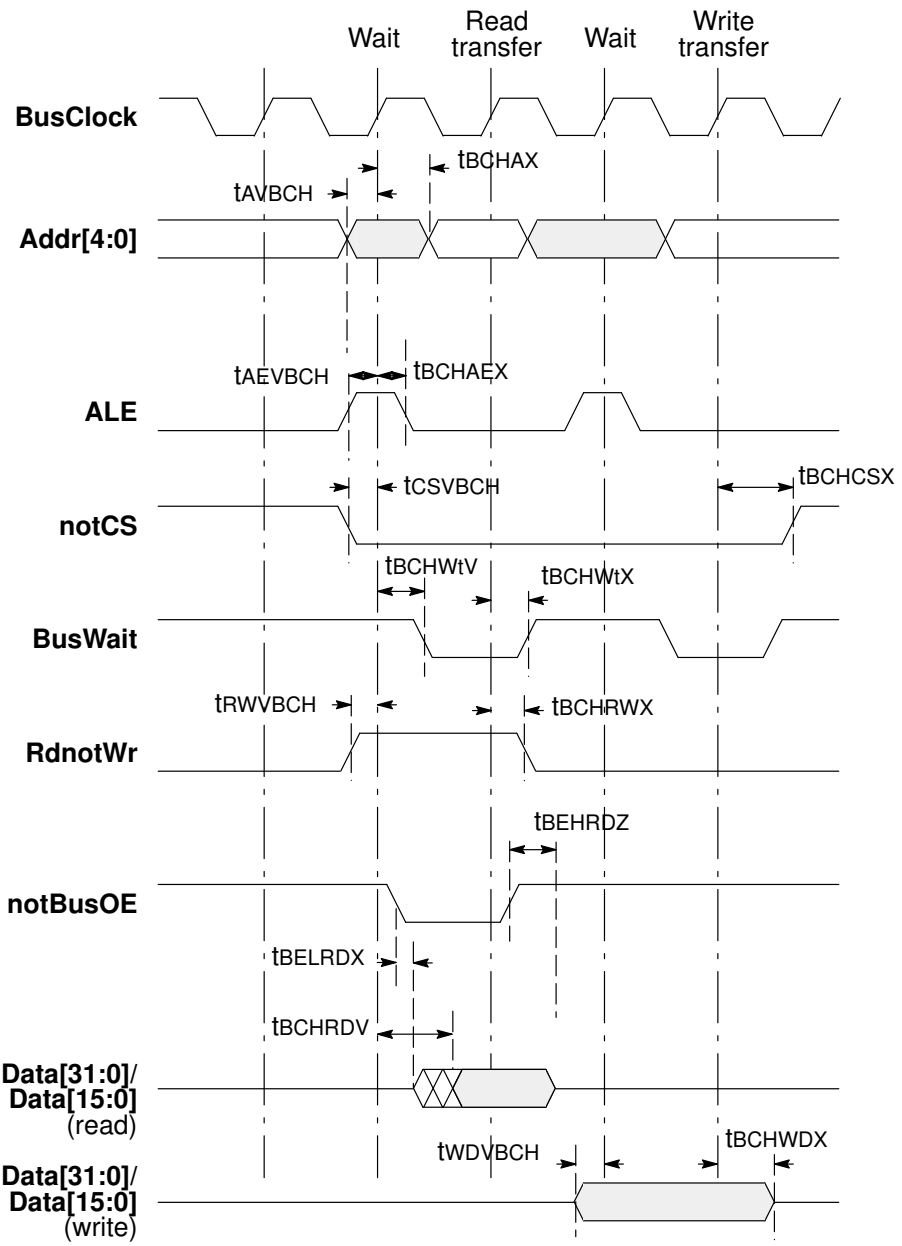


Figure 15.6 Bus interface – synchronous read and write cycles

Symbol	Parameter	Min	Nom	Max	Units	Notes
tAEVBCH	**ALE** valid to **BusClock** high	2.0			ns	
tAVBCH	Address valid to **BusClock** high	2.0			ns	
tBCHAEX	**BusClock** high to **ALE** hold	3.0			ns	
tBCHAX	**BusClock** high to **Address** hold	5.0			ns	
tBCHCSX	**BusClock** high to **notCS** hold	2.0			ns	
tBCHRDV	**BusClock** high to Read **Data** valid	25.0			ns	
tBCHRWX	**BusClock** high to **RdnotWr** hold	5.0			ns	
tBCHWDX	**BusClock** high to Write **Data** hold	5.0			ns	
tBCHWtX	**BusClock** high to **BusWait** hold			25.0	ns	
tBCHWtV	**BusClock** high to **BusWait** valid			25.0	ns	
tBEHRDZ	**notBusOE** high to Read **Data** tristate			15.0	ns	
tBELRDX	**notBusOE** low to Read **Data** low Z	3.0			ns	
tCSVBCH	**notCS** valid to **BusClock** high	3.0			ns	
tRWVBCH	**RdnotWr** valid to **BusClock** high	2.0			ns	
tWDVBCH	Write **Data** valid to **BusClock** high	5.0			ns	

Table 15.4 Bus interface – synchronous read and write cycle timings

15.3 Token interface timings

The setup and hold times for the token interfaces are shown below.

Data transfers take place on a rising bus clock edge when **Valid** is high and **Hold** is low.

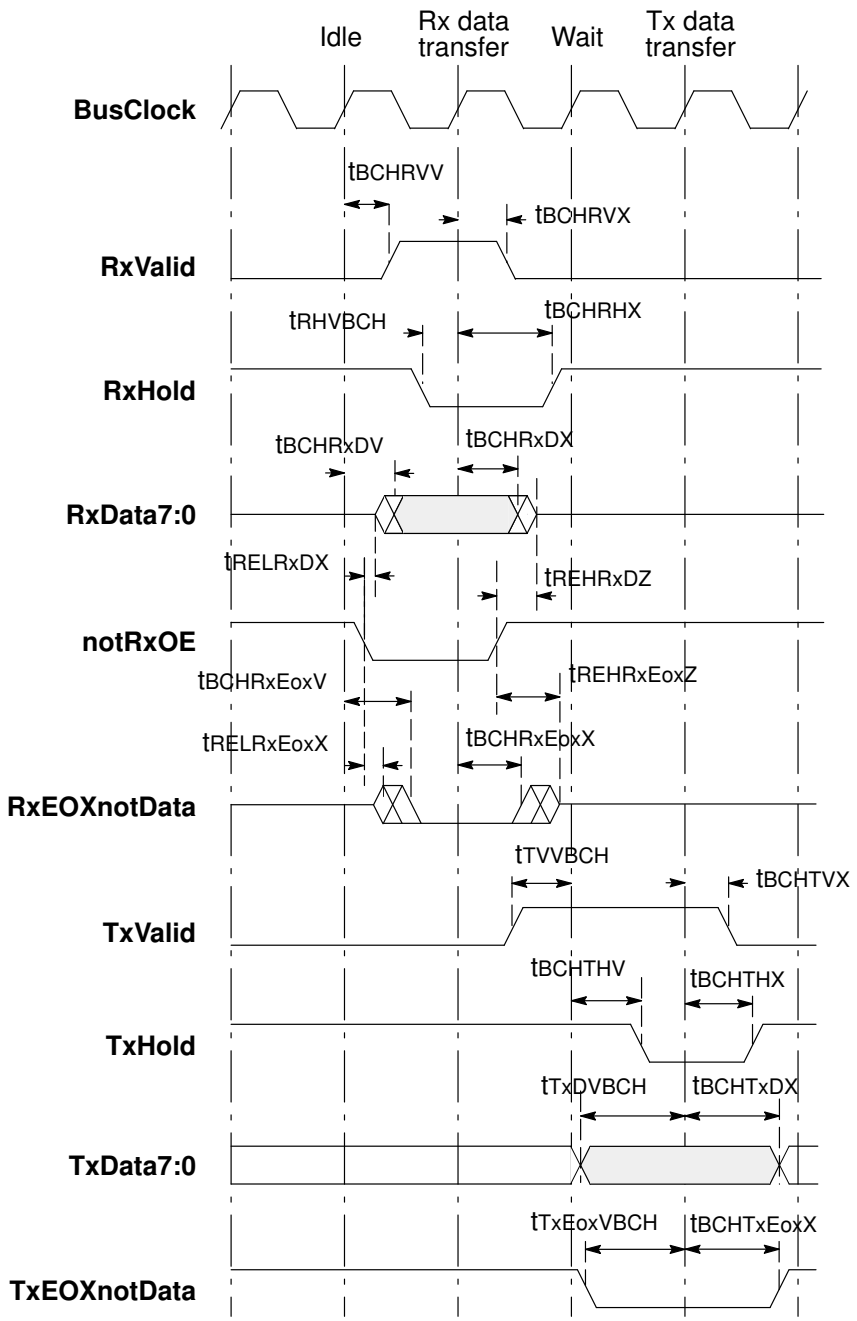


Figure 15.7 Token interface timings

Symbol	Parameter	Min	Nom	Max	Units	Notes
tBCHRHX	**BusClock** high to **RxHold** hold	0.0			ns	
tBCHRVV	**BusClock** high to **RxValid** valid	25.0			ns	
tBCHRVX	**BusClock** high to **RxValid** hold	10.0			ns	
tBCHRxDV	**BusClock** high to **RxData** valid			20.0	ns	
tBCHRxDX	**BusClock** high to **RxData** hold	3.0			ns	
tBCHRxEoxV	**BusClock** high to **RxEOXnotData** valid	20.0			ns	
tBCHRxEoxX	**BusClock** high to **RxEOXnotData** hold	10.0			ns	
tBCHTxEoxX	**BusClock** high to **TxEOXnotData** hold	5.0			ns	
tBCHTHV	**BusClock** high to **TxHold** valid			20.0	ns	
tBCHTHX	**BusClock** high to **TxHold** hold	10.0			ns	
tBCHTVX	**BusClock** high to **TxValid** hold	5.0			ns	
tBCHTxDX	**BusClock** high to **TxData** hold	5.0			ns	1
tREHRxDZ	**notRxOE** high to **RxData** tristate			10.0	ns	
tREHRxEoxZ	**notRxOE** high to **RxEOXnotData** tristate			15.0	ns	
tRELRxDX	**notRxOE** low to **RxData** low Z	3.0			ns	
tRELRxEoxX	**notRxOE** low to **RxEOXnotData** hold	0.0			ns	
tRHVBCH	**RxHold** valid to **BusClock** high	10.0			ns	
tTVVBCH	**TxValid** valid to **BusClock** high	3.0			ns	
tTxDVBCH	**TxData** valid to **BusClock** high	1.0			ns	1
tTxEoxVBCH	**TxEOXnotData** valid to **BusClock** high	10.0			ns	

Table 15.5 Token interface timings

Notes

1 This only needs to be met when a data transfer is taking place.

15.4 DS-Link timings

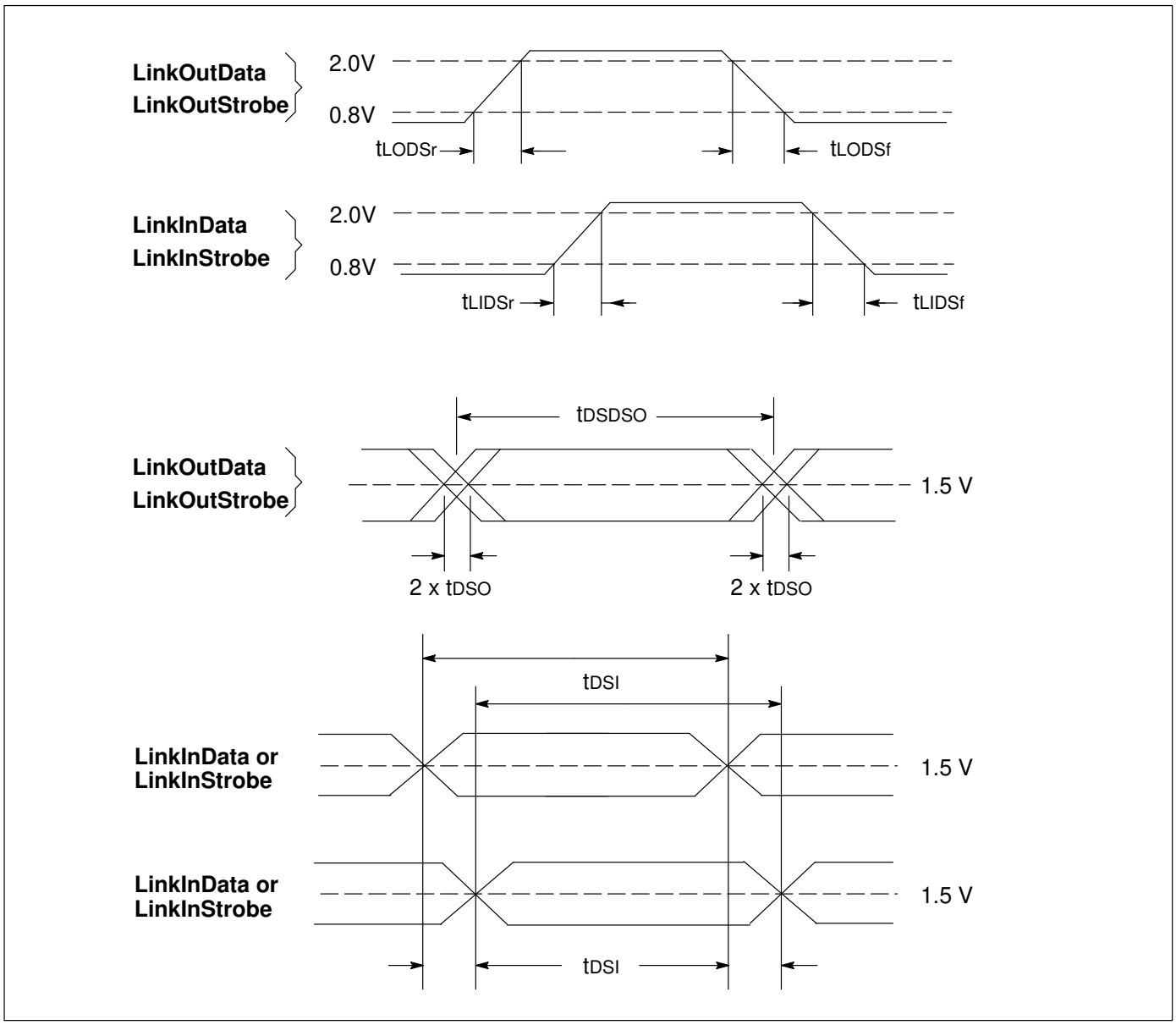


Figure 15.8 DS-Link timing

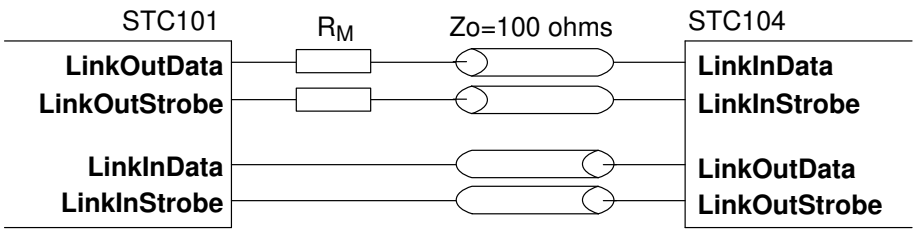


Figure 15.9 DS-Links connected by transmission lines

Symbol	Parameter	Min	Nom	Max	Units	Notes
C_{LIZ}	LinkIn capacitance		7		pF	1
t_{DSDSI}	Sustainable averaged input bit period	9		110	ns	
t_{DSDSO}	Output bit period	10		100	ns	2
t_{DSI}	Data/strobe input edge minimum separation	2.5			ns	3,4,5
t_{DSO}	Data/strobe output skew			$\pm$1	ns	6
t_{LIDSf}	LinkIn fall time (2.0–0.8V)			100	ns	7
t_{LIDSr}	LinkIn rise time (0.8–2.0V)			100	ns	7
t_{LODSf}	LinkOut fall time (2.0–0.8V)			7	ns	8
t_{LODSr}	LinkOut rise time (0.8–2.0V)			7	ns	8
R_{OH}	Output impedance (output driving high)	5		50	Ω	
R_{OL}	Output impedance (output driving low)	5		50	Ω	
R_M	Series resistor for 100 Ω transmission line				ohms	

Table 15.6 DS-Link timings

Notes

1 Sampled, not 100% tested.

2 t_{DSDSO} represents the minimum and maximum programmable bit periods.

3 t_{DSI} is the shortest permissible spacing of 2 consecutive edges on the Data and Strobe wires (1 edge of either sense on each wire). If arriving Data and Strobe edges are skewed to the extent that this parameter is exceeded then the order of the edges becomes ambiguous and a parity error is likely to result.

4 Edge separation includes consecutive edges of a data input or a strobe input.

5 Based on a slew rate of 1.5 V/s, monotonic across the transition region. For other values of slew rate, use the following formula:
1.0 + (k * slew rate) where k=1.0

6 t_{DSO} is the maximum discrepancy between the time when a DS Output edge (either sense) starts a transition and the theoretical ideal (i.e. all consecutive DS edges spaced by t_{DSDSO}).

7 Edges must be monotonic, hence faster edges are recommended unless the link is to be used in a noise free environment.

8 Measurement based on a loading of 25pF.

15.4.1 Link Input and Output relative skews

For the skew parameters to be valid for a wide range of operating speeds (10 – 100 Mbits/s) certain parameters must be made relative to edge rates, as the interaction of edge rates and logic threshold have significant impact on the skew. Note that skew is measured relative to the edges crossing a nominal 1.5V logic threshold.

t_{DSI} = Fixed skew + k * (the larger of t_{LIDSr} and t_{LIDSf})

Where *Fixed Skew* is related to the worst case DSDecoder input skew rejection and internal input path mismatch, and *k* is found by characterization and related to minimum variation in input threshold and input pad propagation delay.

t_{DSO} = Fixed skew

Where *Fixed Skew* is related to the worst case Link Output PLL jitter and internal output path mismatch.

15.4.2 Skew budget

The concept here is that in order to eliminate the risk of DSLink parity errors due to the relative skew between Data and Strobe inputs a system designer must ensure that the sum of $2t_{DSO}$ and the relative skew between Data and Strobe induced by all system interconnect and buffering must be less than $t_{DSDSO} - t_{DSI}$.

Note that an edge rate dependent calculation must be performed for external buffers with variable thresholds in order to calculate worst case $t_{EXTSkew}$ for both Data and Strobe.

i.e. $2t_{DSO} + t_{EXTSkew} < t_{DSDSO} - t_{DSI}$

The parameter t_{DSO} on the left hand side of the expression is multiplied by two to allow for the worst case situation of Data and Strobe undergoing maximum skew in opposite directions.

16 Electrical specifications

Inputs and outputs are TTL compatible.

16.1 Absolute maximum ratings

Symbol	Parameter	Min	Max	Units	Notes
VDD	DC supply voltage	0	7.0	V	1,2,3,4,5
V_I, V_O	Voltage on input and output pins	–0.5	VDD+0.5	V	1,3,4,5
I_I	Input current		10	μA	6
t_{OSC}	Output short circuit time (one pin)		1	s	4
T_S	Storage temperature	–65	150	°C	4

Table 16.1 Absolute maximum ratings

Notes

1 All voltages are with respect to **GND**.

2 Power is supplied to the device via the **VDD** and **GND** pins. Several of each are provided to minimize inductance within the package. All supply pins must be connected. The supply must be decoupled close to the chip by at least one 100 nF low inductance (e.g. ceramic) capacitor between **VDD** and **GND**. Four layer boards are recommended; if two layer boards are used, extra care should be taken in decoupling.

3 Input voltages must not exceed specification with respect to **VDD** and **GND**, even during power-up and power-down ramping, otherwise *latchup* can occur. CMOS devices can be permanently damaged by excessive periods of latchup.

4 This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the operating sections of this specification is not implied. Stresses greater than those listed may cause permanent damage to the device. Exposure to absolute maximum rating conditions for extended periods may affect reliability.

5 This device contains circuitry to protect the inputs against damage caused by high static voltages or electrical fields. However, it is advised that normal precautions be taken to avoid application of any voltage higher than the absolute maximum rated voltages to this high impedance circuit. Unused inputs should be tied to an appropriate logic level such as **VDD** or **GND**.

6 The input current applies to any input or output pin and applies when the voltage on the pin is between **GND** and **VDD**.

16.2 Operating conditions

Symbol	Parameter	Min	Max	Units	Notes
VDD	DC supply voltage	4.75	5.25	V	1
V_I, V_O	Input or output voltage	0	VDD	V	1,2
T_A	Operating temperature range	0	TA_{MAX}	°C	3

Table 16.2 Operating conditions

Notes

1 All voltages are with respect to **GND**.

2 Excursions beyond the supplies are permitted but not recommended.

3 For details of TA_{MAX}, refer to section 17.3 on thermal data.

16.3 DC characteristics

Symbol	Parameter		Min	Max	Units	Notes
V_{IH}	High level input voltage		2.4	VDD	V	1,2,3
V_{IL}	Low level input voltage		–0.5	0.8	V	1,2,3
I_I	Input current	@ GND<VI<VDD		1	µA	1,2
V_{OH}	Output high voltage	@ IOH=2mA	2.4		V	1,2,3,4
V_{OL}	Output low voltage	@ IOL=4mA		0.4	V	1,2,3,4
I_{OZ}	Tristate output current	@ GND<V0<VDD		1	µA	1,2,3
C_{IN}	Input capacitance	@ f=1MHz		7	pF	3
C_{OZ}	Output capacitance	@ f=1MHz		7	pF	3

Table 16.3 DC characteristics

Notes

1 All voltages are with respect to **GND**.

2 Parameters for STC101 measured at 4.7V<**VDD**<5.3V and 0ºC<**TA**<TA_{MAX}.
Input clock frequency = 10 MHz.

3 Characterized on a sample of devices, not tested.

4 For link outputs, IOH=1mA, IOL=1mA.

16.4 Power rating

Power dissipation depends on **VDD**, as shown in figure 16.1. This is for typical power dissipation for an STC101 operating with a **LogicClock** of 50 MHz and **BusClock** of 30 MHz, with the DS-Link running at 100 Mbits/s. This gives a peak power consumption of 1.7W. Power dissipation is also dependent on operating frequency, as shown in figure 16.2, for an STC101 operating at 5 V and **LogicClock** of 50 MHz.

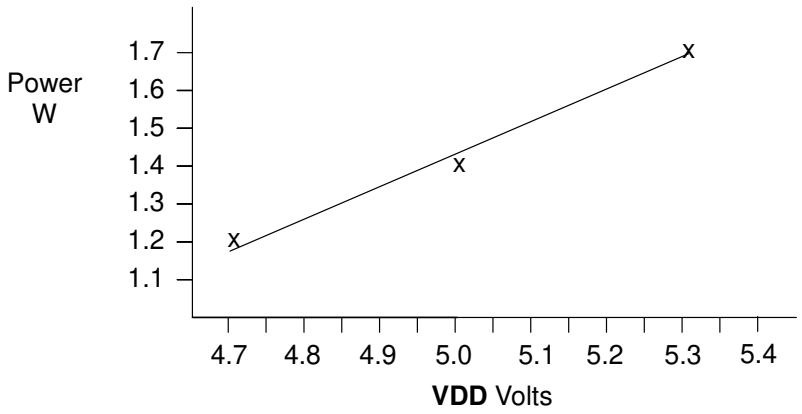


Figure 16.1 Power dissipation vs **VDD**

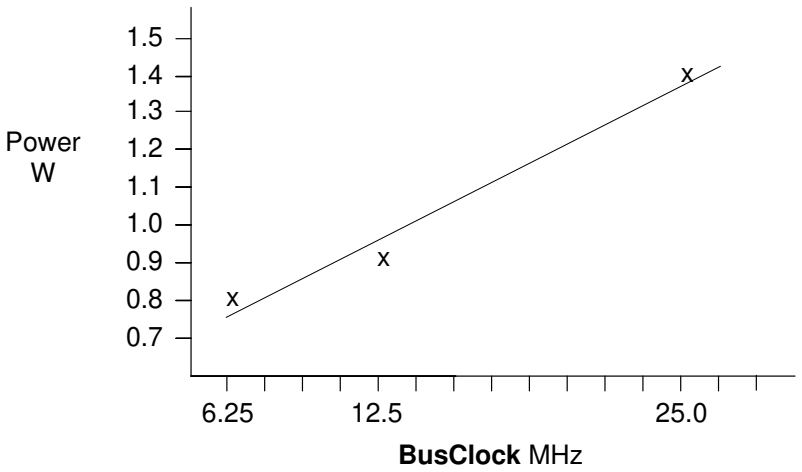


Figure 16.2 Power dissipation vs **BusClock** frequency

17 Package specifications

The STC101 is available in a 100 pin ceramic quad flatpack (CQFP) package.

17.1 STC101 100 pin CQFP package pinout

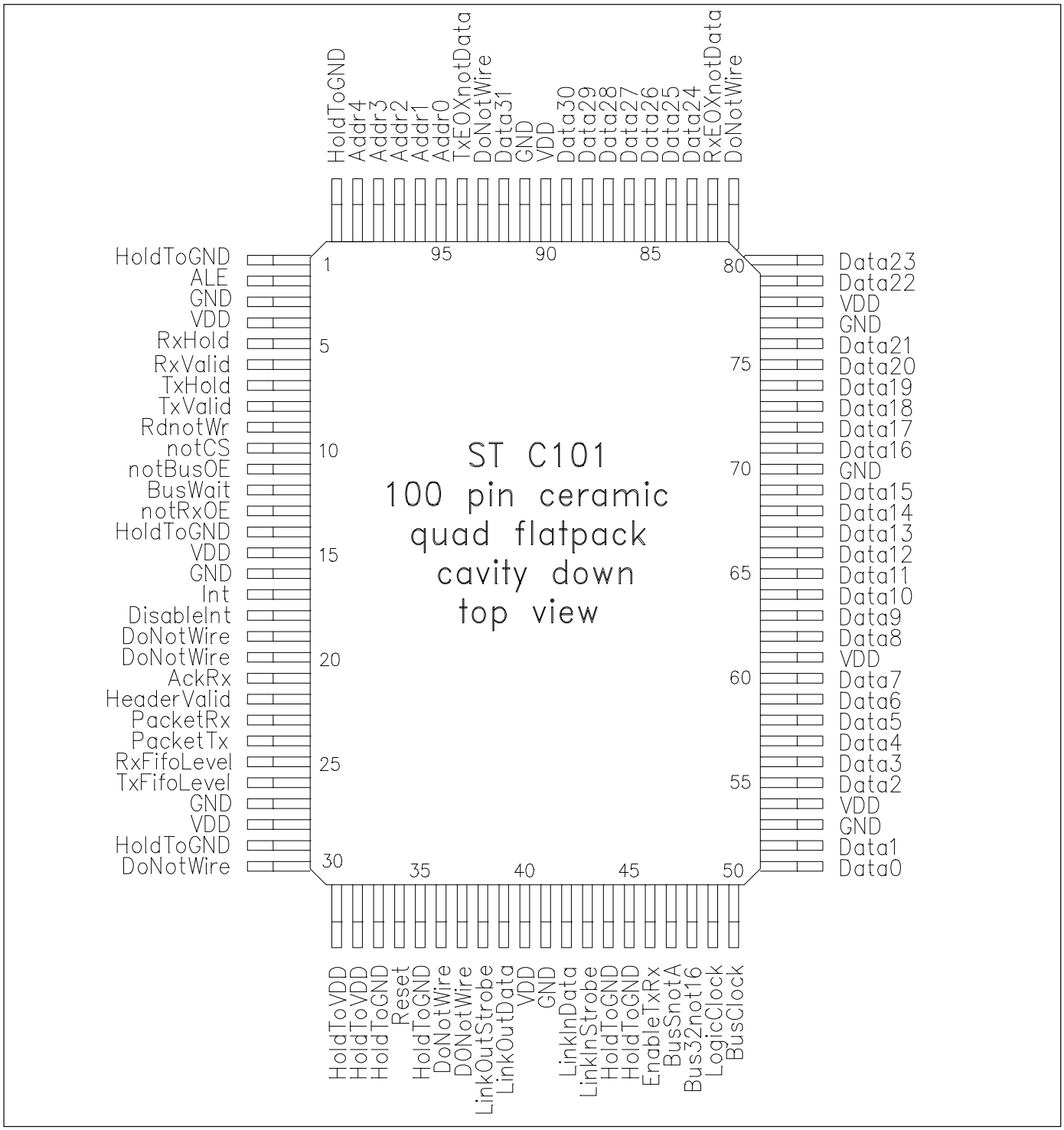


Figure 17.1 STC101 100 pin CQFP package pinout

17.2 STC101 100 pin CQFP package dimensions

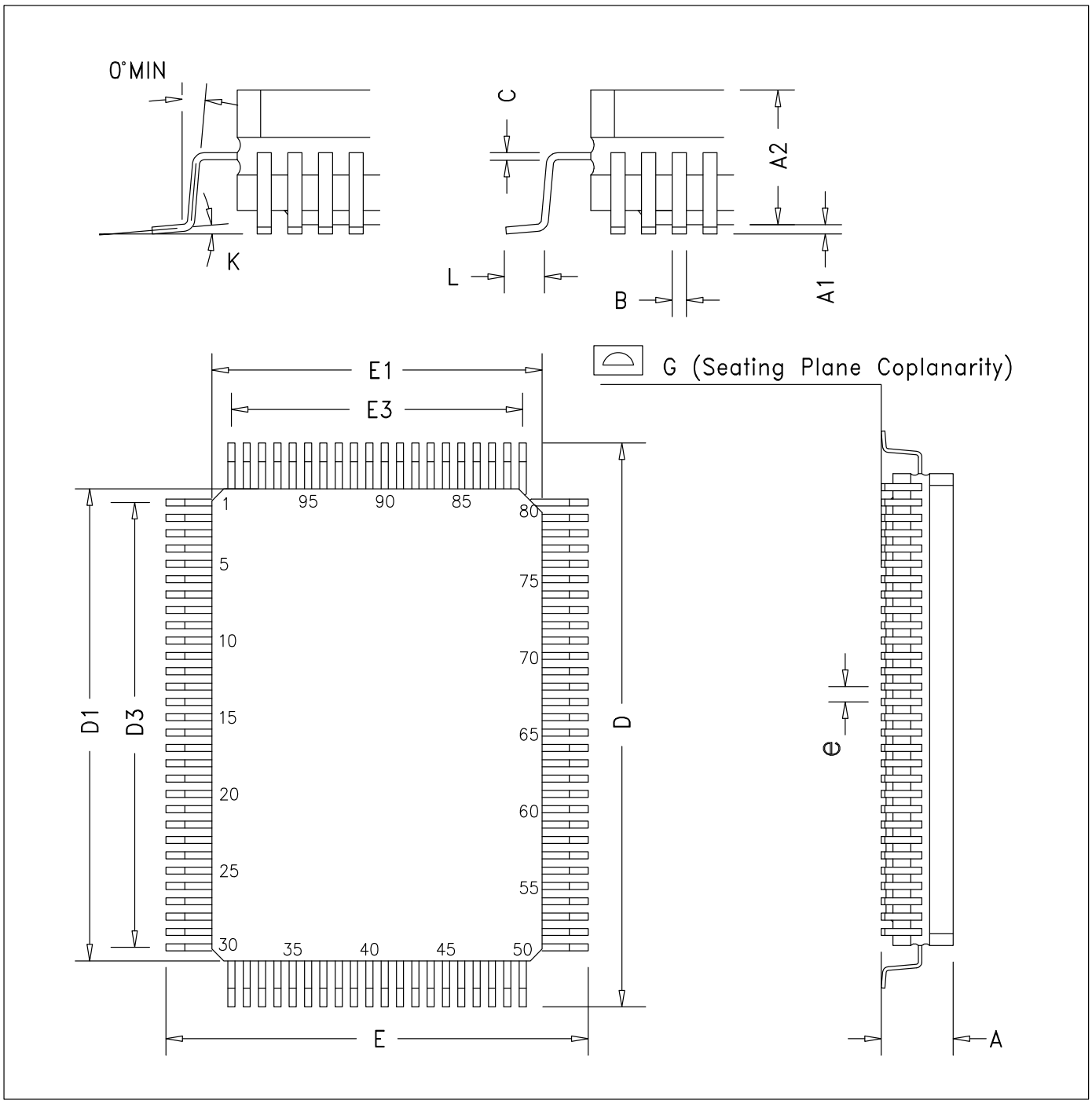


Figure 17.2 STC101 100 pin CQFP package dimensions

REF.	CONTROL DIM. mm			ALTERNATIVE DIM. inches			NOTES
	MIN	NOM	MAX	MIN	NOM	MAX	
A	–	–	3.400	–	–	0.134	
A1	0.250	–	–	0.010	–	–	
A2	–	–	3.073	–	–	0.121	
B	0.220	–	0.432	0.009	–	0.017	
C	0.130	–	0.230	0.005	–	0.009	
D	23.650	–	24.150	0.931	–	0.951	
D1	19.800	20.000	20.200	0.780	0.787	0.795	
D3	–	18.850	–	–	0.742	–	REF
E	17.650	–	18.150	0.695	–	0.715	
E1	13.840	14.000	14.150	0.545	0.551	0.557	
E3	–	12.350	–	–	0.486	–	REF
e	–	0.650	–	–	0.026	–	BSC
G	–	–	0.100	–	–	0.004	
K	0°	–	7°	0°	–	7°	
L	0.650	0.800	0.950	0.026	0.031	0.037	

Table 17.1 STC101 100 pin CQFP package dimensions

Notes

1 Lead finish to be 60 Sn/40 Pb hot solder dip.

2 Maximum lead displacement from the notional center line will be no greater than ± 0.125 mm.

17.3 STC101 100 pin CQFP package thermal data

The STC101 is tested to a maximum silicon junction temperature of 100°C. For operation within the given specifications, the case temperature should not exceed 90°C.

Given a maximum operating junction temperature of 100°C, the following maximum power conditions apply:

Conditions	Maximum power (Watts)
Still air at 35°C	1.65
Still air at 70°C	0.75
Case held at 85°C	2.5

For actual maximum power dissipation see section 16.4.

For temperatures above 100°C the operation of the device cannot be guaranteed and reliability may be impaired.

For further information on reliability refer to the SGS–THOMSON Microelectronics Quality and Reliability Program.

External thermal management is recommended in order to ensure optimum performance and reliability.

18 Ordering information

Device	Package
STC101–F10S	100 pin ceramic quad flatpack (CQFP)

For further information contact your local SGS–THOMSON sales office.

Information furnished is believed to be accurate and reliable. However, SGS-THOMSON Microelectronics assumes no responsibility for the consequences of use of such information nor for any infringement of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of SGS-THOMSON Microelectronics. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all information previously supplied. SGS-THOMSON Microelectronics products are not authorized for use as critical components in life support devices or systems without express written approval of SGS-THOMSON Microelectronics.

© 1995 SGS-THOMSON Microelectronics - All Rights Reserved

DS-Link is a trademark of SGS-THOMSON Microelectronics Limited.

SGS-THOMSON MICROELECTRONICS is a registered trademark of the SGS-THOMSON Microelectronics Group.

SGS-THOMSON Microelectronics GROUP OF COMPANIES
Australia - Brazil - France - Germany - Hong Kong - Italy - Japan - Korea - Malaysia - Malta - Morocco - The Netherlands - Singapore - Spain - Sweden - Switzerland - Taiwan - Thailand - United Kingdom - U.S.A.