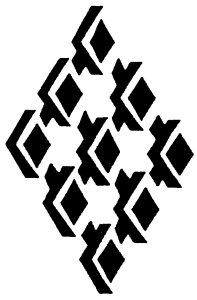


TRANSTECH

PARALLEL TECHNOLOGY

TTM21 RS232
& TTM23 RS422
TRAM

Document version 2.1, 23 Nov 1992

TABLE OF CONTENTS

1 Introduction

This document describes the Transtech TTM21 and TTM23 Transputer modules (TRAMs). Respectively, these enable up to four RS232 or RS422 connections per TRAM to be made to a Transputer network. The RS232 and RS422 interfaces support serial transfer rates of 9.6Kbaud and can buffer up to 256 bytes internally. This is more than adequate for interfacing to the following:

- mouse
- keyboard
- printer
- instrumentation

The TTM21 and TTM23 TRAMs are logically identical. The only significant hardware difference between them is the circuitry used to drive the different serial line standards.

In its simplest mode of operation, the TTM21/TTM23 TRAM can be treated as a standalone Inmos Transputer link to RS232/RS422 interface converter which requires no initialisation - any bytes sent down the link will appear at the interface with no software setup required.

An alternative mode of operation is to configure the serial interface in software. In this mode multiple TTM21 and or TTM23 TRAMs can be daisy-chained allowing large numbers of RS232 and or RS422 connections to be controlled from a single Transputer link. Transfer rates in software mode can be up to 38.4Kbps.

Transputer libraries, include files and example programs are supplied for Inmos ANSI C and occam to aid development of applications using the TTM21/TTM23 in software mode.

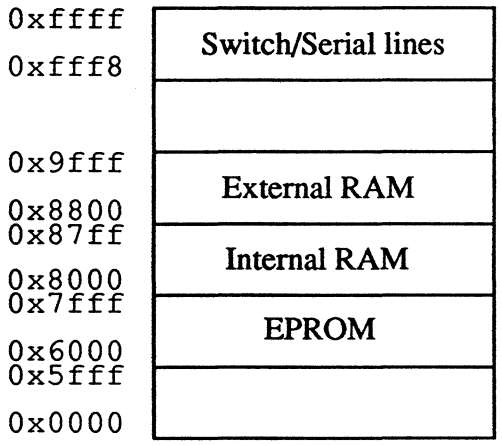
Features of the Transtech TTM21 RS232 TRAM and the TTM23 RS422 TRAM are demonstrated by a program which runs on a PC hosted Transputer connected to a TTM21/TTM23. This program can be used to determine the hardware defined configuration of the TTM21 TRAM.

2 Hardware Description

Both the TTM21 and the TTM23 TRAM have an Inmos T222 Transputer fitted. The T222 is normally used in boot from ROM mode *not* boot from link mode. This is selected by a zero ohm resistor which is marked on the underside of the board.

2.1 T222 system functions

The T222 Transputer has a 64K memory map:



The 2K of internal RAM has a 1 processor cycle access time. For the 8K of external RAM this is 4 cycles.

The top 8 bytes (0xfff8 to 0xffff) are mapped to the jumper switches and the serial lines. The eight switch inputs are read on the data lines D15 to D8 and the four input serial lines on D3 to D0. The four output serial data lines are writeable on D3 to D0.

The TRAM has an 8K EPROM fitted which contains the Transputer code executed by the T222 Transputer after reset. This allows the TTM21 and TTM23 TRAMs to function standalone.

The EPROM resides in addresses up to 0x7fff. Transputer accesses to the EPROM are subject to the wait state generator which can be set to use 200ns (9 cycle access) or 250ns (11 cycle access) ROM devices.

2.2 Serial interfaces

The TTM21 TRAM has eight signal lines and eight grounds organised in pairs. Four of these pairs are used for input and four for output. The TTM23 TRAM also

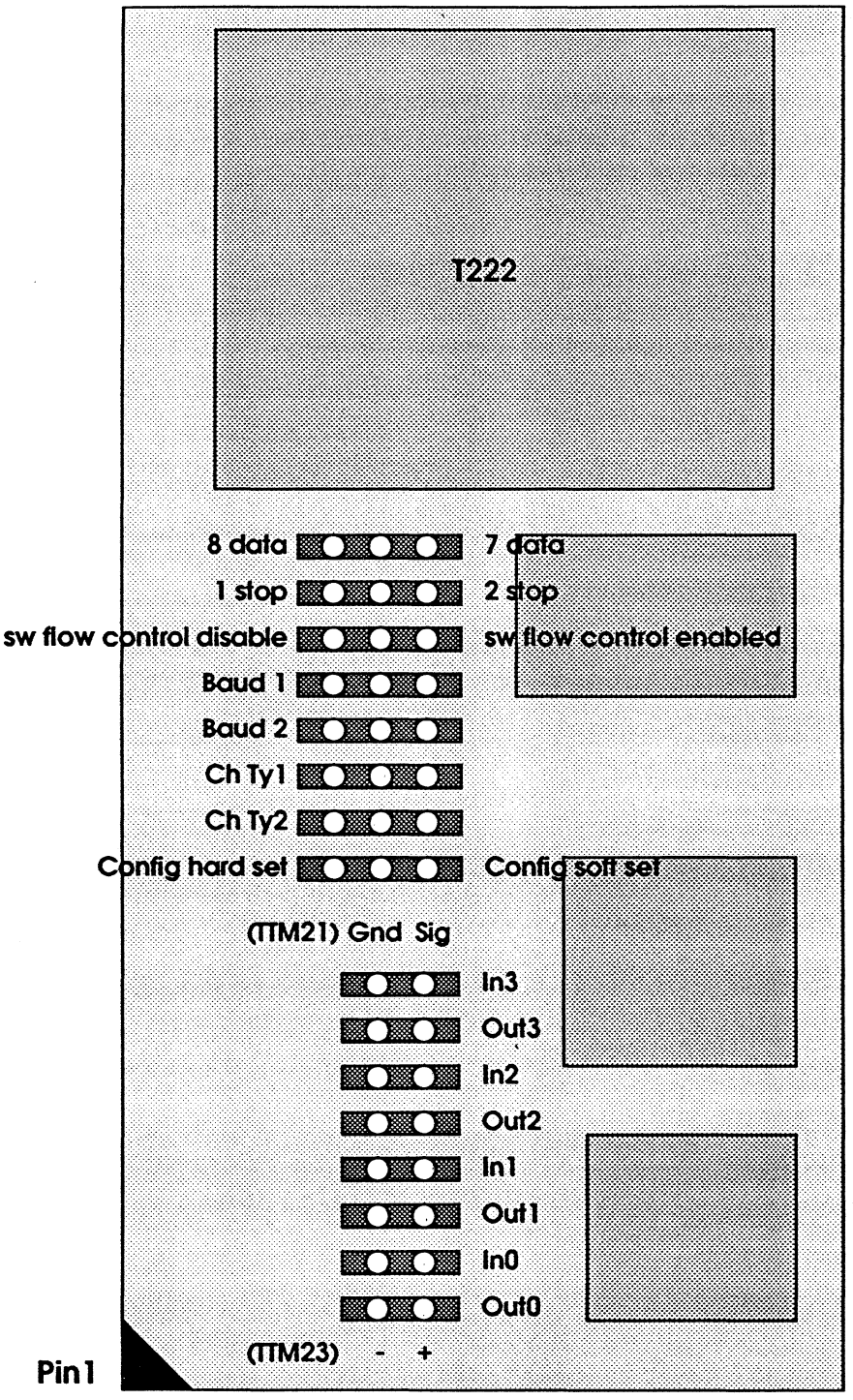


Figure 1 TTM21/TTM23 TRAM

has four input and four output pairs of connections except that these are organised as signal + and signal - instead of signal and ground.

The serial port lines are controlled from a 22v10 PAL which is mapped into the Transputer memory space at address 0xfff8. The state of the input lines is read as the value of the four least significant bits. The state of the output lines can be set by writing to the same bits.

The TTM21 RS232 and TTM23 RS422 TRAM has four input and four output lines controlled directly using a software Uart. Baud rate performance is determined by the frequency at which the Transputer can sample or set the serial lines. With the critical timing routines running in single cycle RAM it is possible to control the port lines to emulate serial interfaces at rates of up to 38.4kbps.

The four serial port lines are equivalent in every way. It is thus possible for the software Uart to implement various types of serial interfaces provided suitable cables are used. For example, for an RS232/RS422 interface with full Modem control the following data lines can be used:

PORT	INPUT	OUTPUT
0	RxData1	TxData1
1	CTS	RTS
2	DSR	DTR

For four sets of 3 wire RS232 or 4 wire RS422 interfaces with software Xon/Xoff control:

PORT	INPUT	OUTPUT
0	RxData1	TxData1
1	RxData2	TxData2
2	RxData3	TxData3
3	RxData4	TxData4

By moving away from the obvious functions of serial interfacing it can be seen that this module can also be used in areas where direct signal sampling at moderate data rates is required.

2.3 Cables

There are many possible RS232/RS422 connections that can be made with the Transtech TTM21/TTM23 TRAM. There is also a wide variety of connectors that are used for such connections. It is for these reasons that no cable is supplied with either the TTM21 or the TTM23 TRAM.

3 RS232/RS422 Modes

The TTM21 TRAM can support three types of RS232 interfaces which are characterised by the number of cable wires used. These are three, five and seven wire modes. The equivalent RS422 modes supported by the TTM23 TRAM are four, eight and twelve wire modes.

3.1 3 wire RS232/4 wire RS422

In the three wire RS232 mode, two signal lines (Txdata and Rxdata) and one ground wire are used. No control lines are used so this mode cannot perform hardware handshaking. This mode would typically be used in a connection for ASCII character data only with software Xon Xoff flow control.

The RS232 signal lines map on to the TTM21 serial port lines in the following way:

Txdata	**OUT**
Rxdata	**IN**

This only uses two serial port lines, thus the TTM21 can support up to four of these three wire RS232 connections.

The equivalent RS422 mode is a four wire mode. Each TTM23 can support up to four of these connections.

3.2 5 wire RS232/8 wire RS422

In the five wire RS232 mode, two control lines (CTS and RTS) are used in addition to the two signal and the ground wire used in the three wire mode. These control lines are used to implement hardware flow control. In certain systems this 5 wire mode is converted into a simple 3 wire connection by shorting RTS & CTS together.

The RS232 signal lines map on to the TTM21 serial port lines in the following way:

Txdata	**OUT**
Rxdata	**IN**
RTS	**OUT**
CTS	**IN**

As four signal lines are used then the TTM21 TRAM can support up to two of these five wire RS232 connections.

The equivalent RS422 mode is an eight wire mode. Each TTM23 TRAM can support up to two of these connections.

3.3 7 wire RS232/12 wire RS422

In the seven wire RS232 mode, two additional control lines (DTR and DSR) are used. A TTM21 TRAM connected to a modem would typically use such an RS232 connection.

The RS232 signal lines map on to the TTM21 serial port lines in the following way:

Txdata	**OUT**
Rxdata	**IN**
RTS	**OUT**
CTS	**IN**
DTR	**OUT**
DSR	**IN**

The TTM21 can support one seven wire RS232 connection.

The equivalent RS422 mode is a twelve wire mode. Each TTM23 TRAM can support one of these connections.

4 Hardware Setup Mode

The hardware setup mode is selected by the "Config hard set" jumper switch setting. See figure 1 on page 4 for the positions of the jumper switches on the TTM21/TTM23 TRAM.

In this mode the other seven jumper switches are used to select a specific RS232/RS422 configuration.

4.1 Configuration options

The configuration options are:

- 7 or 8 data bits
- 1 or 2 stop bits
- Enable or disable software (Xon/Xoff) flow control
- Baud rate selection:

Baud 1	Baud 2	Baud Rate
Left	Left	9600
Left	Right	4800
Right	Left	2400
Right	Right	1200

- RS232 connection mode selection (TTM21):

Switch Settings		Port 0 Link 1	Port 1 Link 0	Port 2 Link 2	Port 3 Link 3
Ch Ty 1	Ch Ty 2				
Left	Left	5	N/A	5	N/A
Left	Right	7	N/A	N/A	3
Right	Left	3	3	3	3
Right	Right	3	3	5	N/A

- RS422 connection mode selection (TTM23):

Switch Settings		Port 0 Link 1	Port 1 Link 0	Port 2 Link 2	Port 3 Link 3
Ch Ty 1	Ch Ty 2				
Left	Left	8	N/A	8	N/A
Left	Right	12	N/A	N/A	4
Right	Left	4	4	4	4
Right	Right	4	4	8	N/A

It is possible to make the TRAM produce an ASCII description of the hardware setup selected by the jumper switches. To do this, temporarily change the jumper to "Config soft set" and run the demonstration program as described in section 6.2 on page 15. For details of how to make the TRAM produce this information see section 5.1 on page 11. Remember to change the jumper back to "Config hard set" once the desired configuration has been selected.

In hardware mode any bytes sent on a link to the TTM21/TTM23 TRAM are output on the corresponding serial port, any bytes input on the port are sent by the TRAM on the corresponding link. Serial ports 0, 1, 2 and 3 correspond to Transputer links 1, 0, 2 and 3 respectively.

4.2 Parity

There is no support for parity on the TTM21 or TTM23 TRAM in hardware setup mode. However, 7 bit data plus parity can be generated for output and checked on input by the user's application.

5 Software Setup Mode

When the "Config soft set" jumper switch setting is used, the TRAM is in software setup mode. In software setup mode the TTM21/TTM23 TRAM can:

- describe the configuration determined by the switch settings when run in hardware mode.
- support multiple RS232 and or RS422 ports on one or more TTM21 and or TTM23 TRAMs from one Transputer link.

In the latter case, called software mode, commands can be issued to the TRAM either by sending messages to it over a link as described below or by using the procedures supplied in the library as described in section 6.3 on page 17.

5.1 Hardware mode description

The byte `cmd.Setup.H` (3) can be sent on a Transputer link to the TRAM in software setup mode. The TRAM responds to this by supplying a number of ASCII characters describing the hardware configuration. This is a description of the TTM21 RS232 configuration as selected by the jumper switches that would be used in hardware setup mode (i.e. if the "Config hard set" switch setting is used).

When using this feature remember to change the jumper switch back to "Config hard set" to use the TRAM in hardware setup mode.

5.2 Software mode

In software mode, one or more TTM21 and or TTM23 TRAMs can be used (daisy-chained as a link 2 to link 1 pipeline) allowing a large number of RS232 and or RS422 ports to be used using just one Transputer link (connected to link 1 of the first TTM21 or TTM23 TRAM).

5.2.1 TRAM identity

The jumper switches (apart from the set configuration switch) have no RS232/RS422 configuration meaning in software mode. Instead they are used to specify an id for the TRAM. So if more than one TTM21 or TTM23 TRAM is used in software mode, ensure that they each have a different set of jumper switch settings.

5.2.2 Mode initialisation

Software mode is initialised by sending the byte **cmd.RS232** (1) down the link to the first TTM21 or TTM23. A reply of the form **INT16::[]BYTE** is returned in which the data consists of a number of four byte words corresponding to the switch settings on each of the TRAMs in the pipeline. These are used as TRAM identities in subsequent commands.

5.2.3 Command packet

Once the TRAM pipeline has been initialised in software mode, all commands and replies have the form **INT16::[]BYTE** where the data is in the form:

```
INT32 id:
INT32 port:
INT32 command:
INT32 length:
[]BYTE data:
```

Where:

id	The TRAM id returned after **cmd.RS232** is sent reflecting the jumper switch settings.
command	The command to perform. Valid values are **read.buffer** (1), **write.buffer** (2), **write.buffer.status** (3) and **init.port** (4).
length	The number of data bytes or the number of bytes free in the write buffer.
data	Data bytes.

On successful completion of a command, the **command** field of the reply packet will contain the command number as sent, otherwise an error value of -1, -2, -3 or -4 is specified.

5.2.4 Commands

Because software mode can support multiple serial ports and multiple TTM21/TTM23 TRAMs using just one Transputer link, then all commands are required to be non-blocking. This means that the **read.buffer** command returns the data (if any) stored in the read buffer which has previously been read from a serial port - it does not wait for new data to appear on the RS232/RS422 line. Similarly the **write.buffer** command writes data to the write buffer and does not wait for it to be output on the serial line.

5.2.4.1 read.buffer

Reads data from the serial link input buffer. Returns a packet of the above format where **length** contains the quantity of bytes returned as **data** and may be 0 if no data is available.

5.2.4.2 write.buffer.status

Returns a packet of the above format where **length** contains the quantity of bytes available in the write buffer available to be written to. The buffer has a capacity of 256 bytes.

5.2.4.3 write.buffer

Writes data to the serial link output buffer (write buffer). Takes a packet of the above format where **length** contains the quantity of bytes in **data** to be written to the write buffer.

5.2.4.4 init.port

Initialises a port for use. Takes a packet of the format:

```
INT32 id:
INT32 port:
INT32 command = 4:
INT32 length = 20:
INT32 chan.type:
INT32 bit.period:
INT32 word.size:
INT32 stop.bits:
INT32 xon.xoff:
```

Where:

chan.type	Describes the RS232/RS422 channel type and can be **wire3** (1), **wire5** (2), **wire7** (3) or **skip** (0) for none.
bit.period	This value in micro seconds specifies the length of one bit. This is determined from the baud rate required by the formula:

```
bit.period = 1000000(INT32)/baud.rate
```

word.size	The number of data bits in an RS232/RS422 word. Normally this would be 7 or 8 but any value from 0 to 8 can be used.

stop.bits	The number of stop bits in a serial data word. This would normally be 1 or 2 but any value from 1 to 8 could be used.
xon.xoff	Software flow control. Determines if the RS232/RS422 channel responds to Xon Xoff characters in the data stream.

An error will occur if it is impossible to place the type of RS232 or RS422 connection specified in **chan.type**, for instance, if a 7 wire connection is attempted on port 3 or if a 5 wire connection is attempted on port 2 when a 3 wire connection has already been configured on port 3.

In the current version of the TRAM firmware, a port cannot be reinitialised once it has been initialised in software. Thus a hardware reset is required if, for instance, a 3 wire serial port needs to be reconfigured as a 5 wire one.

5.3 Parity

There is no support for parity on the TTM21 or TTM23 TRAM in software mode. However, as suggested for hardware setup mode, 7 bit data plus parity can be generated for output and checked on input by the user's application.

6 Software

A PC format diskette is supplied containing software for both the Transtech TTM21 RS232 and the TTM23 RS422 TRAM. The software consists of:

- Demonstration program.
- Library and include files for Inmos C and occam.
- Example C and occam programs.

The software runs on a Transputer connected to a TTM21 or TTM23 TRAM. The TRAM itself is programmed in firmware which is automatically loaded after reset.

6.1 Installation

The software can be installed onto the PC hard disk by using the installation batch file **install.bat**. For instance, to install from diskette drive **A** onto hard disk **C** use:

```
C:\> a:install a: c:
```

6.2 Demonstration program

A demonstration program is supplied which runs on a PC hosted T8 Transputer connected to a TTM21 or TTM23. It is recommended that an ascii terminal or similar peripheral is used in conjunction with the demonstration program to familiarise yourself with the operation of the TTM21 and TTM23 TRAMs.

With a TTM21 RS232 TRAM, it is possible to make a loopback 3 wire RS232 connection simply by connecting the signal pins **Out0** to **In0** using a jumper (see figure 1 on page 4 for details). However, a bug in the present version of the TTM21 firmware sometimes causes data to be corrupted over this connection.

A similar loopback RS422 connection can be made on a TTM23 RS422 TRAM by connecting **Out0+** to **In0+** and **Out0–** to **In0–** using two jumpers.

The demonstration program is run from DOS:

```
C:\> cd ttm21\demo

C:\TTM21\DEMO> demo
```

The program starts by asking:

```
Which link is the T2 i/o pipe off ? [2] :
```

If the TTM21/TTM23 is connected to link 2 of the T8 Transputer, as it will be on most PC Transputer motherboards, just press **Enter** as link 2 is the default. If a different link is used, type the link number followed by **Enter**.

The program then gives a menu of four options:

```
1  Module is in Hardware setup mode & run
2  Module is in Software setup mode & run 4
 chan
3  Module is in Software setup mode & run 1
 chan fast
4  Module is in Software setup mode & show
 Hardware settings

Module is in which i/o mode ? (1 to 4) :
```

Select one of these modes by typing the required number followed by **Enter**.

6.2.1 Hardware setup mode & run

Option 1 should be selected if the TTM21/TTM23 is in hardware setup mode, i.e. the jumper switch setting "Config hard set" is used. This selection allows anything that is typed on the PC to be sent down the Transputer link to the TTM21/TTM23 TRAM which will output it on the corresponding serial port.

6.2.2 Software setup mode & run

Options 2 and 3 can be used when the TTM21/TTM23 is in software setup mode. This option allows an RS232/RS422 connection to be defined and then used.

Prompts are given to determine the TRAM to use (if more than one is present), the port to use, the type (number of wires) of the RS232 connection, the word length, the number of stop bits, the baud rate and whether or not to use software (Xon/Xoff) flow control.

Once the serial port is initialised, it can be tested by pressing **s**, **r** or **w** which returns the number of bytes free in the write buffer, returns the data (if any) in the read buffer or writes the letters of the alphabet to the write buffer respectively.

6.2.3 Software setup mode & show hardware settings

Option 4 can be used if the TTM21/TTM23 is in software setup mode. A description of the hardware configuration as selected by the jumper switches is

displayed. This is the RS232 configuration used by a TTM21 TRAM in hardware setup mode. The corresponding RS422 configuration for a TTM23 TRAM in hardware setup mode can be deduced from this.

6.3 Programming library

A library of functions is supplied to aid programming the TTM21 RS232 and TTM23 RS422 TRAM when in software configuration mode.

The functions can be used from Inmos ANSI C and occam and are supplied in the library **trs.lib**. ANSI C function prototypes and other useful declarations are supplied in the C include file **trs.h**. Declarations used in occam are supplied in the include file **trs.inc**.

To use the TTM21/TTM23 library and include files the environment variable **ISEARCH** should contain the pathname of the TTM21 **libs** directory. This can be done in **autoexec.bat**. For instance, to use the TTM21 library and include files with Inmos ANSI C, a line in **autoexec.bat** is required similar to:

```
set ISEARCH=c:\ictools\libs\ c:\ttm21\libs\
```

Transputer programs that use TTM21/TTM23 library routines are linked using **ilink**. Ensure that **trs.lib** is included in the list of library files used.

The C functions and occam procedures supplied in the library are listed below. Documentation for these routines are printed in the style of UNIX **man** pages.

6.3.1 C functions

The C functions supplied are:

```
int trs_probe( Channel *from_trs,
 Channel *to_trs, long *ids, int size );

int trs_transaction( Channel *from_trs,
 Channel *to_trs, void *reply,
 int reply_size, void *message,
 int message_length );

int trs_init_port( Channel *from_trs,
 Channel *to_trs, long id, long port,
 long chan_type, long bit_period,
 int word_size, int stop_bits, int xon_xoff );
```

```
int trs_read_port_buffer( Channel *from_trs,
 Channel *to_trs, long id, long port,
 char *data, int length );

int trs_write_port_buffer( Channel *from_trs,
 Channel *to_trs, long id, long port,
 char *data, int length );

int trs_write_buffer_status( Channel *
 from_trs, Channel *to_trs, long id,
 long port );
```

6.3.2 Occam procedures

The occam procedures supplied are:

```
PROC trs.probe( CHAN OF ANY from.trs, to.trs,
 INT number, []INT32 ids )

PROC trs.init.port( CHAN OF ANY from.trs,
 to.trs, VAL INT32 id, port, chan.type,
 bit.period, VAL INT word.size, stop.bits,
 VAL BOOL xon.xoff, INT status )

PROC trs.read.port.buffer( CHAN OF ANY
 from.trs, to.trs, VAL INT32 id, port,
 []BYTE data, INT status )

PROC trs.write.port.buffer( CHAN OF ANY
 from.trs, to.trs, VAL INT32 id, port,
 VAL []BYTE data, INT status )

PROC trs.write.buffer.status( CHAN OF ANY
 from.trs, to.trs, VAL INT32 id, port,
 INT status )
```

These are equivalent to the corresponding C functions except that they return values in the parameters **number** and **status.**

6.4 Example programs

Two example programs **c.c** and **occam.occ** are supplied for the Inmos ANSI C and occam Toolsets respectively. These can be found in the **examples** subdirectory. Both programs use routines from the library **trs.lib** and run on a

single Transputer connected to link 1 of a TTM21 or TTM23 TRAM in software setup mode i.e. with the "Config soft set" jumper switch setting.

When compiling these programs, ensure that the environment variable **ISEARCH** contains the name of the directory containing the library and include files as described in section 6.3 on page 17.

A Manual Pages

The C functions and occam procedures supplied in the TTM21/TTM23 library **trs.lib** are documented in the style of UNIX man pages.

The routines can be used to program a Transputer which is connected to link 1 of a TTM21 RS232 or TTM23 RS422 TRAM running in software setup mode i.e. with the jumper switch "Config soft set". Multiple TTM21 and or TTM23 TRAMs configured as a link 2 to link 1 pipeline can be controlled from this Transputer.

If multiple TTM21 and or TTM23 TRAMs are to be controlled in this manner, ensure that each one has a unique setting of its jumper switches as these are used to identify individual TRAMs in the pipeline.

trs_probe

NAME

trs_probe - probe pipeline of TTM21/TTM23 TRAMs

SYNOPSIS

```
#include <channel.h>
#include <trs.h>

int trs_probe( Channel *from_trs, Channel *to_trs,
 long *ids, int size );
```

DESCRIPTION

Sends CMD_RS232 to the Transtech TTM21 RS232 or TTM23 RS422 TRAM and receives back a list of jumper settings which are used as TRAM ids in subsequent function calls. Returns the number of TTM21/TTM23 TRAMs found in a link 2 to link 1 pipeline.

This function should be the first routine to be used from trs.lib and should only be called once.

PARAMETERS

from_trs	Pointer to channel from TTM21/TTM23.
to_trs	Pointer to channel to TTM21/TTM23.
ids	Array of TTM21/TTM23 ids (jumper settings) found.
size	Size of id array.

trs_transaction

NAME

trs_transaction - perform TTM21/TTM23 TRAM operation

SYNOPSIS

```
#include <channel.h>
#include <trs.h>

int trs_transaction( Channel *from_trs, Channel *to_trs,
 void *reply, int reply_size, void *message,
 int message_length );
```

DESCRIPTION

Sends a message to the TTM21 RS232 or TTM23 RS422 TRAM and receives a reply. Returns the length of the reply message.

PARAMETERS

from_trs	Pointer to channel from TTM21/TTM23.
to_trs	Pointer to channel to TTM21/TTM23.
reply	Reply from TTM21/TTM23.
reply_size	Size allocated for reply.
message	Message sent to TTM21/TTM23.
message_length	Length of message.

trs_init_port

NAME

trs_init_port - initialise a TTM21 RS232 or TTM23 RS422 port

SYNOPSIS

```
#include <channel.h>
#include <trs.h>

int trs_init_port( Channel *from_trs, Channel *to_trs,
 long id, long port, long chan_type, long bit_period,
 int word_size, int stop_bits, int xon_xoff );
```

DESCRIPTION

Initialises an RS232 port on a TTM21 RS232 TRAM or an RS422 port on a TTM23 RS422 TRAM.

PARAMETERS

from_trs	Pointer to channel from TTM21/TTM23.
to_trs	Pointer to channel to TTM21/TTM23.
id	TRAM id.
port	RS232/RS422 port number (0-3).
chan_type	RS232 channel type. One of wire3, wire5 or wire7 for three, five or seven wire RS232 or skip for none. The equivalent RS422 modes are four, eight and twelve wire modes.
bit_period	RS232/RS422 bit period in micro-seconds.
word_size	RS232/RS422 word size. Typically 7 or 8.
stop_bits	RS232/RS422 stop bits. Typically 1 or 2.
xon_xoff	Software flow control flag. One enables Xon/Xoff protocol, zero disables it.

trs_read_port_buffer

NAME

trs_read_port_buffer - read TTM21/TTM23 input buffer

SYNOPSIS

```
#include <channel.h>
#include <trs.h>

int trs_read_port_buffer( Channel *from_trs,
 Channel *to_trs, long id, long port, char *data,
 int length );
```

DESCRIPTION

Reads the contents of the TTM21 RS232 or TTM23 RS422 TRAM input buffer returning the number of bytes read. This function returns data already in the buffer, it does not wait for more.

PARAMETERS

from_trs	Pointer to channel from TTM21/TTM23.
to_trs	Pointer to channel to TTM21/TTM23.
id	TRAM id.
port	Port number (0-3).
data	Data from input buffer.
length	Size of data array.

trs_write_port_buffer

NAME

trs_write_port_buffer - write to TTM21/TTM23 output buffer

SYNOPSIS

```
#include <channel.h>
#include <trs.h>

int trs_write_port_buffer( Channel *from_trs,
 Channel *to_trs, long id, long port, char *data,
 int length );
```

DESCRIPTION

Writes data to the TTM21 RS232 or TTM23 RS422 TRAM write buffer ready for output. This function returns the number of bytes written to the buffer, it does not wait until the data has actually been output.

PARAMETERS

from_trs	Pointer to channel from TTM21/TTM23.
to_trs	Pointer to channel to TTM21/TTM23.
id	TRAM id.
port	RS232/RS422 port number.
data	Data to write.
length	Length of data to write.

trs_write_buffer_status

NAME

trs_write_buffer_status - return TTM21/TTM23 write buffer status

SYSNOPSIS

```
#include <channel.h>
#include <trs.h>

int trs_write_buffer_status( Channel *from_trs,
 Channel *to_trs, long id, long port );
```

DESCRIPTION

Returns the number of bytes free in the TTM21 RS232 or TTM23 RS422 TRAM write buffer. Use this function to determine how many bytes can safely be written to the buffer using the function trs_write_port_buffer.

PARAMETERS

from_trs	Pointer to channel from TTM21/TTM23.
to_trs	Pointer to channel to TTM21/TTM23.
id	TRAM id.
port	RS232/RS422 port number.

trs.probe

NAME

trs.probe - look for TTM21 RS232 and or TTM23 RS422 TRAMs

SYNOPSIS

```
#INCLUDE "trs.inc"
#USE "trs.lib"

PROC trs.probe( CHAN OF ANY from.trs, to.trs, INT number,
 []INT32 ids )
:
```

DESCRIPTION

Looks for TTM21 RS232 and or TTM23 RS422 TRAMs in a link 2 to link 1 pipeline returning the number of TRAMs found and their link jumper settings which are used in subsequent procedure calls to identify individual TRAMs.

This procedure should be the first routine from trs.lib to be used and should only be called once.

PARAMETERS

from.trs	Channel from TTM21/TTM23.
to.trs	Channel to TTM21/TTM23.
number	Number of TTM21/TTM23 TRAMs found.
ids	Array of TRAM ids (jumper settings) found.

trs.init.port

NAME

trs.init.port - initialise TTM21 RS232 port or a TTM23 RS422 port

SYNOPSIS

```
#INCLUDE "trs.inc"
#USE "trs.lib"

PROC trs.init.port( CHAN OF ANY from.trs, to.trs,
 VAL INT32 id, port, chan.type, bit.period,
 VAL INT word.size, stop.bits, VAL BOOL xon.xoff,
 INT status )
:
```

DESCRIPTION

Initialises an RS232 port on a TTM21 RS232 TRAM or an RS422 port on a TTM23 RS422 TRAM.

PARAMETERS

from.trs	Channel from TTM21/TTM23.
to.trs	Channel to TTM21/TTM23.
id	TTM21/TTM23 TRAM id.
port	RS232/RS422 port number (0-3).
chan.type	RS232 mode. Can be one of wire3, wire5 or wire7 for three, five or seven wire RS232 or skip for none. The equivalent RS422 modes are four, eight and twelve wire modes.
bit.period	Bit period in micro-seconds.
word.size	RS232/RS422 word size. Typically 7 or 8.
stop.bits	RS232/RS422 stop bits. Typically 1 or 2.
xon.xoff	Software flow control. TRUE enables and FALSE disables Xon/Xoff protocol.

trs.read.port.buffer

NAME

trs.read.port.buffer - read data from input buffer

SYNOPSIS

```
#INCLUDE "trs.inc"
#USE "trs.lib"

PROC trs.read.port.buffer( CHAN OF ANY from.trs, to.trs,
 VAL INT32 id, port, []BYTE data, INT status )
:
```

DESCRIPTION

Reads data from a TTM21 RS232 or TTM23 RS422 TRAM input buffer. The number of data bytes read from the buffer is returned. If no data is available then a length of zero is returned. This procedure does not wait for data to be read from the RS232/RS422 link.

PARAMETERS

from.trs	Channel from TTM21/TTM23.
to.trs	Channel to TTM21/TTM23.
id	TTM21/TTM23 TRAM id.
port	RS232/RS422 port number (0-3).
data	Data array.
status	Return status. Number of bytes read from input buffer or error code.

trs.write.port.buffer

NAME

trs.write.port.buffer - write data to output buffer

SYNOPSIS

```
#INCLUDE "trs.inc"
#USE "trs.lib"

PROC trs.write.port.buffer( CHAN OF ANY from.trs, to.trs,
 VAL INT32 id, port, VAL []BYTE data, INT status )
:
```

DESCRIPTION

Writes data into an output buffer of a TTM21 RS232 or TTM23 RS422 TRAM. This procedure does not wait until the data is actually outputted on the RS232/RS422 port.

PARAMETERS

from.trs	Channel from TTM21/TTM23.
to.trs	Channel to TTM21/TTM23.
id	TTM21/TTM23 TRAM id.
port	RS232/RS422 port number.
data	Data to write.
status	Return status. Number of bytes written or error code.

trs.write.buffer.status

NAME

trs.write.buffer.status - TTM21/TTM23 write buffer status

SYNOPSIS

```
#INCLUDE "trs.inc"
#USE "trs.lib"

PROC trs.write.buffer.status( CHAN OF ANY from.trs,
 to.trs, VAL INT32 id, port, INT status )
:
```

DESCRIPTION

Returns the number of bytes free in the output buffer of a TTM21 RS232 or TTM23 RS422 TRAM. This procedure is used to determine how many bytes can be safely sent using the procedure trs.write.port.buffer.

PARAMETERS

from.trs	Channel from TTM21/TTM23.
to.trs	Channel to TTM21/TTM23.
id	TTM21/TTM23 TRAM id.
port	RS232/RS422 port number (0-3).
status	Return status. Number of bytes free in output buffer or error code.