Econet_CreateReceive (SWI &40000)

Creates a Receive Control Block

On entry

R0 = port number
R1 = station number
R2 = net number
R3 = buffer address
R4 = buffer size in bytes

On exit

R0 = handle
R2 = 0 if R2 on entry is the local net number

Interrupts

Interrupt status is unaltered
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is re-entrant

Use

This call creates a Receive Control Block (RxCB) to control the reception of an Econet packet. It returns a handle to the RxCB.

The buffer must remain available all the time that the RxCB is open, as data received over the Econet is read directly from hardware to the buffer. You must not use memory in application space if your program is to run under the Desktop. Instead, you should use memory from the RMA. To do so, claim the memory using OS_Module 6 (see page 1-233), and – after abandoning the receive control block – return the space to the RMA using OS_Module 7 (see page 1-234).

Econet_ExamineReceive (SWI &40001)

Reads the status of an RxCB

On entry

R0 = handle

On exit

R0 = status

Interrupts

Interrupt status is unaltered
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is re-entrant

Use

This call reads the status of an RxCB, which may be one of the following:

7	Status_RxReady
8	Status_Receiving
9	Status_Received

It returns less information than Econet_ReadReceive, so is faster and corrupts fewer registers. You should use it to poll a reception when not using Econet_WaitForReception.

Related SWIs

Econet_CreateReceive (page 2-647), Econet_WaitForReception (page 2-654), Econet_ConvertStatusToString (page 2-664), Econet_ConvertStatusToError (page 2-666)

Econet_ReadReceive (SWI &40002)

Returns information about a reception, including the size of data

On entry

R0 = handle

On exit

R0 = status
R1 = 0, or flag byte if R0 = 9 (Status_Received) on exit
R2 = port number
R3 = station number
R4 = net number
R5 = buffer address
R6 = buffer size in bytes, or amount of data received if R0 = 9 on exit (Status_Received)

Interrupts

Interrupt status is unaltered
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is re-entrant

Use

This call returns information about a reception; most importantly, it tells you how much data was received, if any, and the address of the buffer in which it was placed. The buffer address is the same as that passed to Econet_CreateReceive (page 2-647). You can call this SWI before a reception has occurred.

The status of the RxCB may be one of the following:

7	Status_RxReady
8	Status_Receiving
9	Status_Received

The returned values in R3 and R4 (the net and station numbers) are those of the transmitting station if the status is Status_Received; otherwise they are the same values that were passed in to Econet_CreateReceive.

Related SWIs

Econet_CreateReceive (page 2-647), Econet_WaitForReception (page 2-654), Econet_AbandonAndReadReceive (page 2-683)

Related vectors

None

Econet_AbandonReceive (SWI &40003)

Abandons an RxCB

On entry

R0 = handle

On exit

R0 = status

Interrupts

Interrupts are disabled
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is re-entrant

Use

This call abandons an RxCB, returning its memory to the RMA. The reception may have completed (R0 = 9 – Status_Received – on exit), in which case the information in the RxCB (such as the sending station number, and the amount of data sent) will be lost. The data in the receive buffer remains unaffected. If the reception is in progress when this SWI is called, then information in the RxCB is lost, as above.

Related SWIs

Econet_CreateReceive (page 2-647), Econet_WaitForReception (page 2-654), Econet_AbandonAndReadReceive (page 2-683)

Related vectors

None

Econet_WaitForReception (SWI &40004)

Polls an RxCB, reads its status, and abandons it

On entry

R0 = handle
R1 = delay in centiseconds
R2 = 0 to ignore Escape; else Escape ends waiting

On exit

R0 = status
R1 = 0, or flag byte if R0 = 9 (Status_Received) on exit
R2 = port number
R3 = station number
R4 = net number
R5 = buffer address
R6 = buffer size in bytes, or amount of data received if R0 = 9 on exit (Status_Received)

Interrupts

Interrupts are enabled
Fast interrupts are enabled

Processor mode

Processor is in SVC mode and in USR mode

Re-entrancy

SWI is not re-entrant

Use

This call repeatedly polls an RxCB (that you have already set up with Econet_CreateReceive) until a reception occurs, or a timeout occurs, or the user interferes (say by pressing Escape). It then reads the status of the RxCB before abandoning it.

The status of the RxCB may be one of the following:

8	Status_Receiving
9	Status_Received
10	Status_NoReply
11	Status_Escape

The returned values in R3 and R4 (the net and station numbers) are those of the transmitting station if the status is Status_Received; otherwise they are the same values that were passed in to SWI Econet_CreateReceive.

Note that because this interface enables interrupts it should not be called from within either interrupt service code or event routines.

During the loop when the polling of the RxCB and of Escape takes place, the processor is put in USR mode with IRQs enabled; this allows callbacks to occur.

Related SWIs

Econet_ExamineReceive (page 2-649), Econet_ReadReceive (page 2-651), Econet_AbandonReceive (page 2-653), Econet_AbandonAndReadReceive (page 2-683)

Related vectors

None

Econet_EnumerateReceive (SWI &40005)

Returns the handles of open RxCBs

On entry

R0 = index (1 to start with first receive block)

On exit

R0 = handle (0 if no more receive blocks)

Interrupts

Interrupt status is unaltered
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

Not defined

Use

This call returns the handles of open RxCBs. On entry R0 is the number of the RxCB being asked for (1, 2, 3...). If the value of R0 is greater than the number of open RxCBs, then the value returned as the handle will be 0, which is an invalid handle.

This call should not be made from an IRQ or event routine as, although it will not fail, errors and omissions are likely to occur in the returned information.

Related SWIs

Econet_CreateReceive (page 2-647),
Econet_ReadReceive (page 2-651), Econet_AbandonReceive (page 2-653)

Related vectors

None

Econet_StartTransmit (SWI &40006)

Creates a Transmit Control Block and starts a transmission

On entry

R0 = flag byte
R1 = port number
R2 = station number
R3 = net number
R4 = buffer address
R5 = buffer size in bytes
R6 = count
R7 = delay in centiseconds

On exit

R0 = handle
R1 corrupted
R2 = buffer address
R3 = station number
R4 = net number

Interrupts

Interrupt status is unaltered
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is re-entrant

Use

This call creates a Transmit Control Block (TxCB) to control the transmission of an Econet packet. It then starts the transmission.

The buffer must remain available all the time that the TxCB is open, as data transmitted over the Econet is read directly from the buffer to hardware. You must not use memory in application space if your program is to run under the Desktop.

Instead, you should use memory from the RMA. To do so, claim the memory using OS_Module 6 (see page 1-233), and – after abandoning the transmit control block – return the space to the RMA using OS_Module 7 (see page 1-234).

The value returned in R4 (the net number) will be the same as that passed in R3 unless that number is equal to the local net number; in that case the net number will be returned as zero.

Related SWIs

Econet_PollTransmit (page 2-659), Econet_AbandonTransmit (page 2-660), Econet_DoTransmit (page 2-661)

Related vectors

None

Econet_PollTransmit (SWI &40007)

Reads the status of a TxCB

On entry

R0 = handle

On exit

R0 = status

Interrupts

Interrupt status is unaltered
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is re-entrant

Use

This call reads the status of a TxCB, which may be one of the following:

0	Status_Transmitted
1	Status_LineJammed
2	Status_NetError
3	Status_NotListening
4	Status_NoClock
5	Status_TxReady
6	Status_Transmitting

Related SWIs

Econet_StartTransmit (page 2-657), Econet_AbandonTransmit (page 2-660)

Related vectors

None

Econet_AbandonTransmit (SWI &40008)

Abandons a TxCB

On entry

R0 = handle

On exit

R0 = status

Interrupts

Interrupt status is unaltered
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is re-entrant

Use

This call abandons a TxCB, returning its memory to the RMA. The returned status is the same as for Econet_PollTransmit.

Related SWIs

Econet_StartTransmit (page 2-657), Econet_PollTransmit (page 2-659)

Related vectors

None

Econet_DoTransmit (SWI &40009)

Creates a TxCB, polls it, reads its status, and abandons it

On entry

R0 = flag byte
R1 = port number
R2 = station number
R3 = net number
R4 = buffer address
R5 = buffer size in bytes
R6 = count
R7 = delay in centiseconds

On exit

R0 = status
R1 corrupted
R2 = buffer address
R3 = station number
R4 = net number

Interrupts

Interrupts are enabled
Fast interrupts are enabled

Processor mode

Processor is in SVC mode and in USR mode

Re-entrancy

SWI is not re-entrant

Use

This call creates a TxCB and repeatedly polls it until it finishes transmission, or it exceeds the count of retries. It then reads the final status of the TxCB before abandoning it.

The status of the TxCB may be one of the following:

0	Status_Transmitted
1	Status_LineJammed
2	Status_NetError
3	Status_NotListening
4	Status_NoClock

The value returned in R4 (the net number) will be the same as that passed in R3 unless that number is equal to the local net number; in that case the net number will be returned as zero.

Note that because this interface enables interrupts it should not be called from within either interrupt service code or event routines.

During the loop when the polling of the TxCB and of Escape takes place, the processor is put in USR mode with IRQs enabled; this allows callbacks to occur.

Related SWIs

Econet_StartTransmit (page 2-657), Econet_PollTransmit (page 2-659), and Econet_AbandonTransmit (page 2-660)

Related vectors

None

Econet_ReadLocalStationAndNet (SWI &4000A)

Returns a computer's station number and net number

On entry

No parameters passed in registers

On exit

R0 = station number
R1 = net number

Interrupts

Interrupts are enabled
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is not re-entrant

Use

This call returns a computer's station number and Econet net number. The net number will be zero if there are no Econet bridges present on the network.

For more information, see the section entitled *Reading your station and net numbers* on page 2-638.

Related SWIs

None

Related vectors

None

Econet_ConvertStatusToString (SWI &4000B)

Converts a status to a string

On entry

R0 = status
R1 = pointer to buffer
R2 = buffer size in bytes
R3 = station number
R4 = net number

On exit

R0 = buffer
R1 = updated buffer address
R2 = updated buffer size in bytes

Interrupts

Interrupt status is unaltered
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is re-entrant

Use

This call converts a status to a string found in the messages file. This is then copied into RAM, including the station and net numbers, giving a string such as:

```
Network station 59.254 not listening
```

If the status given in R0 is invalid (ie not in the range 0 - 14), this will cause a data abort or an address exception. If the station/net number given in R3/R4 is invalid, no station information is given.

Under RISC OS 2 the string is not read from the messages file, but is instead read direct from the ROM.

Related SWIs

Econet_ConvertStatusToError (page 2-666)

Related vectors

None

Econet_ConvertStatusToError (SWI &4000C)

Converts a status to a string, and then generates an error

On entry

R0 = status
R1 = pointer to error buffer
R2 = error buffer size in bytes
R3 = station number
R4 = net number

On exit

R0 = pointer to error block
V flag is set

Interrupts

Interrupt status is unaltered
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is re-entrant

Use

This call converts a status to a string found in the messages file. This is then copied into RAM, including the station and net numbers, giving a string such as:

```
Network station 59.254 not listening
```

If the station/net number given in R3/R4 is invalid, no station information is given.

Finally this call returns an error by setting the V flag, with R0 pointing to the error block.

If you use a buffer address of zero, then the string is left in a buffer in the MessageTrans workspace.

Under RISC OS 2 the string is not read from the messages file, but is instead read direct from the ROM.

Related SWIs

Econet_ConvertStatusToString (page 2-664)

Related vectors

None

Econet_ReadProtection (SWI &4000D)

Reads the current protection word for immediate operations

On entry

No parameters passed in registers

On exit

R0 = current protection value

Interrupts

Interrupt status is unaltered
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is re-entrant

Use

This call reads the current protection word for immediate operations. Various bits in the word, when set, disable corresponding immediate operations:

Bit	Immediate operation
0	Peek
1	Poke
2	Remote JSR
3	User procedure call
4	OS procedure call
5	Halt
6	Continue – always zero on RISC OS computers
7	Machine peek – always zero on RISC OS computers
8	Get registers
9 - 31	Reserved – must be zero

Note – This call is deprecated. You should preferably use the call Econet_SetProtection (page 2-670) to read the protection word instead of this call.

Related SWIs

Econet_SetProtection (page 2-670)

Related vectors

None

Econet_SetProtection (SWI &4000E)

Sets or reads the protection word for immediate operations

On entry

R0 = EOR mask word
R1 = AND mask word

On exit

R0 = old value

Interrupts

Interrupts are enabled on write-through to CMOS, preserved otherwise
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is not re-entrant

Use

This call sets the protection word for immediate operations as follows:

New value = (old value AND R1) EOR R0

Various bits in the word, when set, disable corresponding immediate operations:

Bit	Immediate operation
0	Peek
1	Poke
2	Remote JSR
3	User procedure call
4	OS procedure call
5	Halt
6	Continue – must be zero on RISC OS computers
7	Machine peek – must be zero on RISC OS computers

8	Get registers
9 - 30	Reserved – must be zero
31	Write new value to the CMOS RAM

Normally this call sets or reads the current value of the word. A default value for this word is held in CMOS RAM.

The most useful values of R0 and R1 are:

Action	R0	R1
Set current value	new value (0 - &1FF)	0
Read current value	0	&FFFFFFFF
Set new default value	&80000000 + new value	0

You should use this call to read the value of the protection word, rather than Econet_ReadProtection (page 2-668).

Using this call to read is also the preferred method for detecting the presence of the Econet drivers, since doing so can never return an unexpected error. Detecting the error 'No such SWI' allows software dependent upon Econet to report its absence. Example code is given in the section entitled *Application notes* on page 2-691.

Related SWIs

None

Related vectors

None

Econet_ReadStationNumber (SWI &4000F)

Extracts a station and/or net number from a supplied string

On entry

R1 = address of string to read

On exit

R1 = address of terminating space or control character
R2 = station number (–1 for not found)
R3 = net number (–1 for not found)

Interrupts

Interrupts are enabled
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is re-entrant

Use

This call extracts a station and/or net number from a supplied string. For an example of its use, see the section entitled *Extracting station numbers from a string* on page 2-638.

Related SWIs

None

Related vectors

None

Econet_PrintBanner (SWI &40010)

Prints the string 'Acorn Econet' followed by a newline

On entry

—

On exit

—

Interrupts

Interrupts are enabled
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is not re-entrant

Use

This call prints the string 'Acorn Econet' followed by a newline. The string is fetched from a message file with the token 'AcrnEco'. If the Econet network data clock is not present then this call instead prints the string 'Acorn Econet, no clock' followed by a newline. In this case, the token used is 'EcoNClk'.

This call uses OS_Write0 and OS_NewLine, and so cannot be called from within either interrupt service code or event routines.

Related SWIs

None

Related vectors

None

Econet_ReadTransportType (SWI &40011)

Returns the underlying transport type to a given station

On entry

R0 = station number
R1 = net number
R2 = 2

On exit

R0, R1 preserved
R2 = transport type (0 ⇒ not known, 1 ⇒ Internet, 2 ⇒ Econet, 3 ⇒ Nexus)

Interrupts

Interrupt status is unaltered
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is re-entrant

Use

This call is used by clients to determine the underlying transport type to a given station. They can then use this information to determine the optimum transmission strategy to use, based on prior empirical knowledge of the different transport types.

This call is unnamed – but still available by number – in both RISC OS 2 and RISC OS 3 (version 3.00).

Related SWIs

None

Related vectors

None

Econet_ReleasePort (SWI &40012)

Releases a port number that was previously claimed

On entry

R0 = port number

On exit

—

Interrupts

Interrupt status is unaltered
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is re-entrant

Use

This call releases a port number that was previously claimed by calling Econet_ClaimPort (page 2-678).

You must not use this call for port numbers that have been previously claimed using Econet_AllocatePort (page 2-676); instead, you must call Econet_DeAllocatePort (page 2-677).

Related SWIs

Econet_ClaimPort (page 2-678)

Related vectors

None

Econet_AllocatePort (SWI &40013)

Allocates a unique port number

On entry

No parameters passed in registers

On exit

R0 = port number

Interrupts

Interrupt status is unaltered
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is re-entrant

Use

This call allocates a unique port number that has not already been claimed or allocated.

When you have finished using the port number, you should call Econet_DeAllocatePort (page 2-677) to make it available for use again.

Related SWIs

Econet_DeAllocatePort (page 2-677)

Related vectors

None

Econet_DeAllocatePort (SWI &40014)

Deallocates a port number that was previously allocated

On entry

R0 = port number

On exit

—

Interrupts

Interrupt status is unaltered
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is re-entrant

Use

This call deallocates a port number that was previously allocated by calling Econet_AllocatePort (page 2-676).

You must not use this call for port numbers that have been previously claimed using Econet_ClaimPort (page 2-678); instead, you must call Econet_ReleasePort (page 2-675).

Related SWIs

Econet_AllocatePort (page 2-676)

Related vectors

None

Econet_ClaimPort (SWI &40015)

Claims a specific port number

On entry

R0 = port number

On exit

—

Interrupts

Interrupt status is unaltered
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is re-entrant

Use

This call claims a specific port number. If it has already been claimed or allocated, an error is generated.

When you have finished using the port number, you should call Econet_ReleasePort (page 2-675) to make it available for use again.

Related SWIs

Econet_ReleasePort (page 2-675)

Related vectors

None

Econet_StartImmediate (SWI &40016)

Creates a TxCB and starts an immediate operation

On entry

R0 = operation type
R1 = remote address or Procedure number
R2 = station number
R3 = net number
R4 = buffer address
R5 = buffer size in bytes
R6 = count
R7 = delay in centiseconds

On exit

R0 = handle
R1 corrupted
R2 = buffer address
R3 = station number
R4 = net number

Interrupts

Interrupts are disabled
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is re-entrant

Use

This call creates a TxCB and starts an immediate operation. For full details see the section entitled *Immediate operations* on page 2-629.

The buffer must remain available all the time that the TxCB is open, as data transmitted over the Econet is read directly from the buffer to hardware. You must not use memory in application space if your program is to run under the Desktop.

Instead, you should use memory from the RMA. To do so, claim the memory using OS_Module 6 (see page 1-233), and – after abandoning the transmit control block – return the space to the RMA using OS_Module 7 (see page 1-234).

The value returned in R4 (the net number) will be the same as that passed in R3 unless that number is equal to the local net number; in that case the net number will be returned as zero.

Related SWIs

Econet_DoImmediate (page 2-681)

Related vectors

None

Econet_DoImmediate (SWI &40017)

Creates a TxCB for an immediate operation, polls it, reads its status, and abandons it

On entry

R0 = operation type
R1 = remote address or procedure number
R2 = station number
R3 = net number
R4 = buffer address
R5 = buffer size in bytes
R6 = count
R7 = delay in centiseconds

On exit

R0 = status
R1 corrupted
R2 = buffer address
R3 = station number
R4 = net number

Interrupts

Interrupts are enabled
Fast interrupts are enabled

Processor mode

Processor is in SVC mode and in USR mode

Re-entrancy

SWI is re-entrant

Use

This call creates a TxCB for an immediate operation, and repeatedly polls it until it finishes transmission or it exceeds the count of retries. It then reads the final status of the TxCB before abandoning it. For full details see the section entitled *Immediate operations* on page 2-629.

The value returned in R4 (the net number) will be the same as that passed in R3 unless that number is equal to the local net number; in that case the net number will be returned as zero.

Note that because this interface enables interrupts it should not be called from within either interrupt service code or event routines.

During the loop when the polling of the TxCB and of Escape takes place, the processor is put in USR mode with IRQs enabled; this allows callbacks to occur.

Related SWIs

Econet_StartImmediate (page 2-679)

Related vectors

None

Econet_AbandonAndReadReceive (SWI &40018)

Abandons a reception and returns information about it, including the size of data

On entry

R0 = handle

On exit

R0 = status
R1 = 0, or flag byte if R0 = 9 (Status_Received) on exit
R2 = port number
R3 = station number
R4 = net number
R5 = buffer address
R6 = buffer size in bytes, or amount of data received if R0 = 9 on exit (Status_Received)

Interrupts

Interrupt status is unaltered
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is re-entrant

Use

This call abandons an RxCB, returning its memory to the RMA. It also returns information about the reception; most importantly, it tells you how much data was received, if any, and the address of the buffer in which it was placed. The buffer address is the same as that passed to Econet_CreateReceive (page 2-647). You can call this SWI before a reception has occurred.

The status of the RxCB may be one of the following:

7	Status_RxReady
9	Status_Received

The returned values in R3 and R4 (the net and station numbers) are those of the transmitting station if the status is Status_Received; otherwise they are the same values that were passed in to Econet_CreateReceive.

This call is not available in RISC OS 2, nor in RISC OS 3 (version 3.00).

Related SWIs

Econet_CreateReceive (page 2-647), Econet_ReadReceive (page 2-651), Econet_AbandonReceive (page 2-653)

Related vectors

None

Econet_Version (SWI &40019)

Returns the version of software for the underlying transport to a given station

On entry

R0 = station number
R1 = net number

On exit

R0, R1 preserved
R2 = version number × 100 (eg 547 for version 5.47)

Interrupts

Interrupt status is unaltered
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is re-entrant

Use

This call is used by clients to determine the version of software that handles the underlying transport to a given station. If both R0 and R1 are set to zero on entry, this call instead returns the version number of the top-level software to which RISC OS passes the Econet SWIs.

This call is not available in RISC OS 2, nor in RISC OS 3 (version 3.00).

Related SWIs

None

Related vectors

None

Econet_NetworkState (SWI &4001A)

Returns the state of the underlying transport to a given station

On entry

R0 = station number
R1 = net number

On exit

R0, R1 preserved
R2 = transport state (0 ⇒ fully functional, 1 ⇒ no clock signal)

Interrupts

Interrupt status is unaltered
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is re-entrant

Use

This call returns the state of the underlying transport to a given station. The state returned is transport type dependent, but you may always assume that a value of zero means that the transport is fully functional.

You should only use the returned value as a hint to the exact state; in other words, it is suitable for display but not for decision making. Using this call is no substitute for proper error handling; to determine if a particular transmit will fail, you must do the transmit and be prepared for it to fail.

Related SWIs

Econet_PrintBanner (page 2-673)

Related vectors

None

Econet_PacketSize (SWI &4001B)

Returns the maximum packet size recommended on the underlying transport to a given station

On entry

R0 = station number
R1 = net number

On exit

R0, R1 preserved
R2 = maximum permitted packet size, in bytes

Interrupts

Interrupt status is unaltered
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is re-entrant

Use

This call returns the maximum recommended packet size on the underlying transport to a given station. Larger packets will not necessarily be rejected, but their use is not recommended. The size returned is transport type dependent.

This call is intended for use by modules supplying protocols; you do not need to use it in application software. For maximum efficiency the protocol module should negotiate the packet size once. Since the recommended packet size may differ between the stations at either end of a transmission, the protocol module should interrogate both stations and take the lower value returned.

Related SWIs

None

Related vectors

None

Econet_ReadTransportName (SWI &4001C)

Returns the name of the underlying transport to a given station

On entry

R0 = station number
R1 = net number

On exit

R0, R1 preserved
R2 = pointer to null terminated name of transport

Interrupts

Interrupt status is unaltered
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is re-entrant

Use

This call returns the name of the underlying transport to a given station. You can use this to insert the transport name into (for example) a status conversion.

Related SWIs

None

Related vectors

None