Reading Packets Information on handling data

If a client of **The .ENET Driver** has used the **EAttachPH** function to provide a pointer to its own protocol handler, **The .ENET Driver** calls that protocol handler, which must in turn call **The .ENET Driver**'s ReadPacket and ReadRest routines to read the data. Your protocol handler calls **The .ENET Driver**'s ReadPacket and ReadRest routines in essentially the same way as you call **The .MPP Driver**'s ReadPacket and ReadRest routines.

The following sections describe how **The .ENET Driver** calls a custom protocol handler and the ReadPacket and ReadRest routines.

Note: Because an Ethernet protocol handler must read from and write to the CPU's registers, you cannot write a protocol handler in high-level language such as Pascal or C.

How the .ENET Driver Calls Your Protocol Handler

You can provide an Ethernet protocol handler for a particular protocol type and use the **EAttachPH** function to attach it to **The .ENET Driver**. When the driver receives an Ethernet packet, it reads the packet header into an internal buffer, reads the protocol type, and calls the protocol handler for that protocol type. The CPU is in interrupt mode, and the registers are used as follows:

Registers on call to Ethernet protocol handler

A0	Reserved for internal use by **The .ENET Driver**. You must preserve this register until after the ReadRest routine has completed execution.
A1	Reserved for internal use by **The .ENET Driver**. You must preserve this register until after the ReadRest routine has completed execution.
A2	Free for your use.
A3	Pointer to first byte past data-link header bytes (the first byte after the 2-byte protocol-type field).
A4	Pointer to the ReadPacket routine. The ReadRest routine starts 2 bytes after the start of the ReadPacket routine.
A5	Free for your use until after the ReadRest routine has completed execution.
D0	Free for your use.
D1	Number of bytes in the Ethernet packet left to be read (that is, the number of bytes following the Ethernet header).
D2	Free for your use.
D3	Free for your use.

If your protocol handler processes more than one protocol type, you can read the protocol-type field in the data-link header to determine the protocol type of the packet. The protocol-type field starts 2 bytes before the address pointed to by the A3 register.

Note: The source address starts 8 bytes before the address pointed to by the A3 register, and the destination address starts 14 bytes before the address pointed to by the A3 register.

If you know that the packet contains pad bytes and you know the actual size of the data, you can reduce the number in the D1 register by the number of pad bytes so that **The .ENET Driver** can keep accurate track of the number of bytes remaining to be read. In all other circumstances, you should not change the value in the D1 register.

After you have called the ReadRest routine, you can use registers A0 through A3 and D0 through D3 for your own use, but you must preserve all other registers. You cannot depend on having access to your application global variables.

How Your Protocol Handler Calls the .ENET Driver

Your protocol handler must call **The .ENET Driver** routines ReadPacket and ReadRest to read the incoming data packet. You may call the ReadPacket routine as many times as you like to read the data piece by piece into one or more data buffers, but you must always use the ReadRest routine to read the final piece of the data packet. The ReadRest routine restores the machine state (the stack pointers, status register, and so forth) and checks for error conditions.

Before you call the ReadPacket routine, you must allocate memory for a data buffer and place a pointer to the buffer in the A3 register. You place the number of bytes you want to read in the D3 register. You must not request more bytes than remain in the data packet.

To call the ReadPacketroutine, execute a JSR instruction to the address in the A4 register. The ReadPacket routine uses the registers as follows:

Registers on entry to the ReadPacket routine

A3	Pointer to a buffer to hold the data you want to read
D3	Number of bytes to read; must be nonzero

Registers on exit from the ReadPacket routine

A0	Unchanged
A1	Unchanged
A2	Unchanged
A3	First byte after the last byte read into buffer
D0	Changed
D1	Number of bytes left to be read
D2	Unchanged
D3	Equals 0 if requested number of bytes were read, nonzero if error

The ReadPacket routine indicates an error by clearing to 0 the zero (z) flag in the status register. If the ReadPacketroutine returns an error, you must terminate execution of your protocol handler with an RTS instruction without calling ReadPacket again or calling ReadRest at all.

Call the ReadRest routine to read the last portion of the data packet, or call it after you have read all the data with ReadPacket routines and before you do any other processing or terminate execution. You must provide in the A3 register a pointer to a data buffer and must indicate in the D3 register the size of the data buffer. If you have already read all of the data with calls to the ReadPacket routine, you can specify a buffer of size 0.

Warning: If you do not call the ReadRest routine after your last call to the ReadPacket routine, the system will crash.

To call the ReadRest routine, execute a JSR instruction to an address 2 bytes past the address in the A4 register. The ReadRest routine uses the registers as follows:

Registers on entry to the ReadRest routine

A3	Pointer to a buffer to hold the data you want to read
D3	Size of the buffer (short length); may be 0

Registers on exit from the ReadRest routine

A0	Unchanged
A1	Unchanged
A2	Unchanged
A3	Pointer to first byte after the last byte read into buffer
D0	Changed
D1	Changed
D2	Unchanged
D3	Equals 0 if requested number of bytes were read; less than 0 if more data was left than would fit in buffer (extra data equals -D3 bytes); greater than 0 if less data was left than the size of the buffer (extra buffer space equals D3 bytes)

The ReadRest routine indicates an error by clearing to 0 the zero (z) flag in the status register.

You must terminate execution of your protocol handler with an RTS instruction whether or not the ReadRest routine returns an error.