



Porting Ethernet Software CRC to NET+OS 5.0, 5.1, and 6.0

Application Note

July 2005

Table of Contents

<i>Porting Ethernet Software CRC to NET+OS 5.0, 5.1, and 6.0</i>	<i>3</i>
Overview	3
Detecting the error	3

Porting Ethernet Software CRC to NET+OS 5.0, 5.1, and 6.0

This application note describes an Ethernet receive packet corruption related to the NET+50 and NS7520 processors and how to correct it.

Overview

Under rare conditions, the NET+50 and NS7520 processors can corrupt Ethernet receive packets. The corruption, which the Ethernet receiver hardware does not detect automatically, causes invalid data to be passed to the protocol stack and onto applications. (For details about this problem, see the errata for the NET+50 and NS7520 processors.)

TCP/IP-based applications usually do not experience this corruption because TCP and UDP checksum the data portion of frames sent to them. You do not need to be concerned about this problem if you are using TCP or UDP and have checksums enabled. The issue may, however, cause problems if you are using a protocol that does not checksum incoming packets.

Detecting the error

A feature was added to the Ethernet driver that detects the error. Added in NET+OS 6.2, this feature is available in all later versions of NET+OS.

If you are using NET+OS 6.2 or later, enable this feature by setting the `BSP_WANT_2ND_ETHERNET_CHECKSUM` constant in the `bsp.h` BSP configuration file. For details, see the online help.

If you are using an earlier version of NET+OS, port the fix to your Ethernet driver. The fix consists of a routine that performs a CRC checksum of incoming Ethernet packets. The routine, which is named `ether_crc32`, is contained in two files, which are located on the **Application Notes** page:

- `ether_crc32.arm`
- `crctable.inc`

The routine takes two arguments: a pointer to an Ethernet packet and the packet length. The function returns either 0 if the packet is valid or nonzero if it isn't. The function is declared as:

```
int ether_crc32(BYTE *const packet, unsigned const length);
```

When a packet is received, the Ethernet hardware starts the Ethernet header, which is 14-bytes long, on a 32-bit boundary. The hardware then inserts two bytes of padding and writes the rest of the packet starting at offset 16. The driver moves the Ethernet header (the first 14-bytes) up two bytes before passing the packet onto the protocol stack.

The result is that the IP portion of the packet starts at a 32-bit boundary, and the packet is contiguous. The packet must be passed to `ether_crc32` in this format, so a packet could start at 0x1002, but not at 0x1000 or 0x1004. This can be done by calling the CRC function just before the packet is passed to the protocol stack (and after the Ethernet driver has moved the Ethernet header).

Because NET+OS 4.0, 5.0, and 5.1 support only TCP/IP, if you are using any of these NET+OS versions with another protocol stack, you must have already modified the Ethernet driver. Modify the driver again to call `ether_crc32` before it passes the packet onto your protocol stack. The packet is valid if `ether_crc32` returns 0. If the packet does not return 0, discard it. You can improve performance by calling only `ether_crc32` for non-TCP/IP packets because TCP/IP performs its own checksums.

Support for additional protocols was added into the Ethernet driver in NET+OS 6.0. The `eth_rcv_up` routine passes receive packets to the correct protocol stack. The packet is passed on by the call to the `packetType[i].rcvFn` function pointer. Modify the `eth_rcv_up` routine to perform the CRC check just before passing on the packet. The function pointer, `packetType[i].rcvFn`, should be called if `ether_crc32` returns 0. If `ether_crc32` does not return 0, set the local variable `result` to `FALSE` without calling the function pointer.

A refinement that will improve performance is to call the CRC routine only for non-TCP/IP packets.

Because NET+OS 6.1 does not support either the NS7520 or NET-50, you do not need to port this fix to it.