Protection for the Asynchronous Information Phases (COMMAND, MESSAGE, and STATUS)

To: T10 Technical committee
From: Mark Evans and Bruce Leshay
Quantum Corporation
500 McCarthy Boulevard
Milpitas, CA USA 95035
Phone: 408-894-4019
Fax: 408-952-3620
Email: mark.evans@quantum.com
bruce.leshay@quantum.com
Date: 6 April 1999

1 Introduction

With the advent of CRC on the DT data phases, customers have requested similar types of protection on non-data information phases (COMMAND, MESSAGE, and STATUS) in those cases where the information may become corrupted (e.g., by hot plugging events). These requests include having protection coverage on the phase lines (MSG, C/D, and I/O) as well. The constraint for the solution to this issue is that it should be a simple, modular addition to the existing SCSI protocol (i.e., these customers desire this capability in a non-Packetized SCSI protocol). The solution in this proposal resolves this issue.

This revision of this proposal includes the bulk of the material in T10/99-162r2 from Larry Lammers and John Lohmeyer on “Enabling Upper Byte Hamming” wherein the initiator and target determine if the BCH code is being generated by a simple test during the first I/O process following a power on or reset condition.

2 Overview

In the DT protocol with CRC, the SCSI bus is restricted to 16-bit (wide) data phases only. COMMAND, MESSAGE, and STATUS information is sent along the 8-bit (narrow) bus as normal. For these phases, the upper eight data bits on the wide bus could be used for other purposes such as carrying enhanced protection information, including protection coverage on the phase lines (MSG, C/D, and I/O).

Using some of the upper eight bits for added protection would also allow the remaining bits on the bus not required for the protection scheme to be used for additional functions. The following describes this scheme.

3 Protection Code

The following are the items to be encoded and details of the protection code to be used on the asynchronous information phases (COMMAND, MESSAGE, and STATUS).

3.1 Covered signals

The following signals are to be covered by the code. Associated with each signal is its bit location in the 21-bit code word. When a device receives the information byte, it also latches the state of the other SCSI signals and values noted in the table.

Codeword Bit Location	SCSI Signal	Meaning
0	DB0	Data bit 0 of the information byte
1	DB1	Data bit 1 of the information byte
2	DB2	Data bit 2 of the information byte
3	DB3	Data bit 3 of the information byte
4	DB4	Data bit 4 of the information byte
5	DB5	Data bit 5 of the information byte
6	DB6	Data bit 6 of the information byte
7	DB7	Data bit 7 of the information byte
8	DB8	Reserved
9	DB9	Reserved
10	MSG	Phase control line (see note)
11	C/D	Phase control line (see note)
12	I/O	Phase control line (see note)
13	Seq ID 0	Sequence ID bit 0
14	Seq ID 1	Sequence ID bit 1
15	DB10	Redundant bit 0 of the code word
16	DB11	Redundant bit 1 of the code word
17	DB12	Redundant bit 2 of the code word
18	DB13	Redundant bit 3 of the code word
19	DB14	Redundant bit 4 of the code word
20	DB15	Redundant bit 5 of the code word
Note: For calculation purposes the application client uses the values that should be on these signal lines as determined by the current phase.		

The reserved signals (DB8 and DB9, or bits 8 and 9 of the codeword) can be used for other functions in the future.

The Sequence ID is a “virtual” signal which is carried in the encoding. The Sequence ID serves as the seed for the BCH calculation by the sender. The receiver then uses the expected seed for the transfer for decoding.

The Sequence ID increments in value during a “run”. A “run” is a sequence of transfers during a MESSAGE, COMMAND, or STATUS phase, without an intervening DATA, BUS FREE, ARBITRATION, SELECTION, or RESELECTION phase. In addition, a new run begins with every phase change and every time that ATN is negated.

For each new run, the Sequence ID is set to zero for the first word transferred, one for the second word transferred, two for the third word transferred, and three for the fourth word transferred. The Sequence ID then cycles back to zero for the fifth word transferred, and so forth until the run is complete. At the beginning of the next run, the Sequence ID starts at zero again.

The Sequence ID provides protection for errors where a data transfer is missed or double clocked. If a BCH code error is detected or if a sequence ID value is missing during a run, then the transfer is invalid. Recovery from these protection errors is the same as parity error recovery (see clause 6).

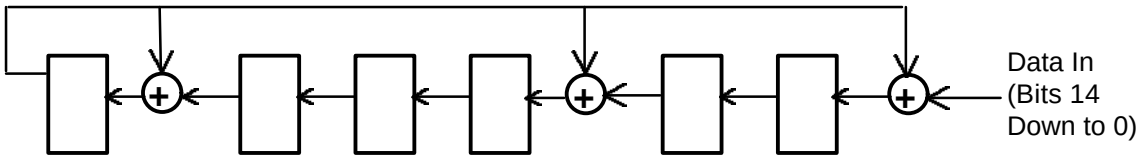
3.2 Code Description

Several protection codes were examined by Dr. Lih Weng of Quantum. The following is the code that he recommends for this application. The code is a cyclic binary BCH code:

Code	Maximum data bits allowed	Number of redundant bits	Minimum distance of the code
(21,15,4)	15	6	4

Given the less stressful nature of the asynchronous information transfer phases, and the extremely short code words (approximately 20 bits compared to the thousands of bits during a DT data phase), the requirements for Hamming distance for the BCH code should not be stricter than for the CRC used on the high speed synchronous DT data phases. Earlier calculations indicated that the data CRC has a Hamming distance of at least four for data transfers less than eight kilobytes. This is the same minimum distance provided by this BCH code.

The code is a cyclic code with a generator polynomial of $x^6 + x^5 + x^2 + 1$. The canonical form of the code generator is shown below. This is a serial implementation: the register is initialized to zero, then the data is fed in one bit at a time, codeword bit 14 (as defined above) first, followed by codeword bit 13, 12, 11,… and so on until bit 0. As each data bit is input, the shift register is clocked. When all 15 bits have been clocked in, the check bits are available in the registers, check bit 0 (codeword bit 15) on the right in the diagram, and check bit 5 (codeword bit 20) on the left. The **+** signs represent an XOR operation.



Using this representation as a baseline, it is possible to construct logic to generate the six check bits from an input data stream of n-bit width, including all 15 bits simultaneously, which would be the expected implementation.

A computer program for the code written in the C language has been made available on the T10 reflector.

4 Enabling

1) All devices supporting the protection code protection shall generate the code during all COMMAND, MESSAGE, and STATUS phases.
2) If detection of the protection code is enabled, then the device shall check the code when receiving all COMMAND, MESSAGE, and STATUS information.
3) If the protection code is detected during a message or command received by the target, then the target shall use protection code detection on that I_T nexus for subsequent I/O processes until a power on or reset condition occurs.
4) If protection is detected on the status and command completion message received by the initiator, the initiator shall use protection code detection on that I_T nexus for subsequent I/O processes until a power on or reset condition occurs.
5) If a device receives two bytes of COMMAND, MESSAGE, or STATUS information with good parity but having a protection code error, the device should disable protection code checking for that I_T nexus. (This is to handle hot-plug situations where a device supporting protection code checking is replaced by one that does not.)

5 Parity

During a protected transfer of COMMAND, MESSAGE, or STATUS information P(0) shall contain the negotiated parity for the low order byte (DB0 through DB7) as normal. P(1) shall contain the negotiated parity for the high order byte of the transfer (DB8 through DB15).

6 Error handling

If the device server detects a protection error during a COMMAND, STATUS or MESSAGE phase while protection code detection is enabled, then the target shall terminate the current nexus with a CHECK CONDITION. The device server shall set the Sense Key, ASC and ASCQ to be HARDWARE ERROR (04H), SCSI PARITY ERROR (47H,00H).

If the initiator detects a protection error during a COMMAND or STATUS phase while protection code detection is enabled, then the initiator shall assert ATN and transmit the message INITIATOR DETECTED ERROR (05h). If the initiator detects a protection error during a MESSAGE phase it shall assert ATN and transmit the message MESSAGE PARITY ERROR (09h).

Note: The exception handling for hot plug events probably needs more enumeration and discussion. A good implementor's note is needed. The most likely scenario is that detection of a protection code error will result in the application client aborting the outstanding I/O process for that nexus and re-initializing the connection. The considerations for rebuild operations on RAID implementations is beyond the scope of this standard.