MOTOROLA
SEMICONDUCTOR APPLICATION NOTE

Order this document
by AN1221/D

AN1221

Hamming Error Control Coding Techniques with the HC08 MCU

by Mark McQuilken & Mark Glenewinkel
CSIC Applications

INTRODUCTION

This application note is intended to demonstrate the use of error control coding (ECC) in a digital transmission system. The HC08 MCU will be used to illustrate the code development of this process. A message frame consisting of a 4-bit data field with three parity bits will be encoded to allow the original four bits to be recovered, even if any single bit is corrupted during the transmission and reception processes. This process is based upon a class of linear error-correcting codes called Hamming codes. The process of using time diversity is also discussed as a way to control burst errors in a transmission system.

BACKGROUND

MODEL FOR A DATA TRANSMISSION SYSTEM

Generally, the processing of digital data (in a whole system and/or within a system) can be modelled as a generalized data transmission or storage system. Such a system consists of an information source, the "channel" (medium or storage), and the destination device. Some possible specific implementations that fit this generalized model are shown in Table 1 below:

Table 1. Digital Data Transmission System Implementations

Example System	Information Source	Channel	Destination
Personal computer	CPU	RAM or hard disk	CPU or serial interface
Digital telephone	Your digitized voice	Telephone company's wires and switching system	Whoever you're calling (another telephone)
Stereo system	Compact disc	Laser optics and electronics	Audio output (ultimately to transducers like speakers)
Avionic weapon development system	Cockpit command to release weapon stores	Wiring/electronics between cockpit and weapons	Electro-mechanical assembly which activates and deploys weapons

There are many other systems that also fit this simplistic model of a data transmission system.

Problems with Data Transmission Systems

In a perfect world, data would be transmitted and received completely intact. In the real world, we must often combat the effects of errors induced in our transmitted/received information. For example, in the above-mentioned avionics system, if a command from the cockpit to release the weapons was transmitted and corrupted so that the deployment electronics interpreted the message as "hold" rather

than “release,” the intended target may never be hit. A worse case would be the pilot wanting to abort deployment and the “abort” command was corrupted by noise to become “release” at the deployment assembly. A system must be built in a way to avoid disastrous effects of data corruption. ECC is the field of study that deals with methods of coding information to reduce the effects of errors.

One of the general challenges facing the system designer is to implement ECCs without degrading the data transfer rates too significantly. For example, one of the simplest ways to hedge against channel-induced errors, is to specify a transmission protocol that requires multiple transmissions of the same data. Such a protocol will reduce the effective rate of transmission by a factor equal to the number of retransmissions per information block. Even if this retransmission strategy guaranteed that at least one of the data blocks was correct at the receiver, there are two problems:

1) How will we know which of the data blocks is the correct one?
2) How do we make up for the channel bandwidth lost by transmitting the same data multiple times?

ECCs can actually be more effective in using the channel's existing bandwidth than the above scenario. Instead of merely retransmitting data, ECCs take up some of the channel bandwidth (but less than retransmissions) to send some data redundancies that may be used for the detection and, sometimes, correction of corrupted received data.

Types of Noise

There are many types of ECCs. Each ECC has its own set of strengths and weaknesses. One important aspect of all the codes is their ability or inability to deal with each of the error types: random channel errors/noise, burst-type errors/noise, or a combination of each. The codes discussed in this presentation (Hamming codes) are a class of algebraic codes that deal effectively with random channel noise. Burst errors are typically longer in duration than random errors and thus require more robust code types to prevent unrecoverable data. This application note also describes time-diversity coding used to prevent burst-type noise. The technique of coupling time-diversity ECCs with random error ECCs can help improve the performance of the ECC in the presence of burst errors.

One-way Data Transmission

In a system where only one-way communication is possible and precautions must be taken against data corruption, error control codes must be employed by using forward error correction (FEC). FEC is accomplished by employing codes that allow for “automatic” correction of specific types of errors induced by noise in the channel and received by the receiver. Hamming codes are one of the simplest classes of FEC codes and are characterized by the following traits:

— The number of parity-check symbols (bits) must be greater than or equal to 3. Let these symbols be represented by the variable m.

— The number, k, of information symbols (bits) is:

$$k = 2^m - m - 1$$

— The total length, n, of the code is:

$$n = 2^m - 1$$

— Error correcting capability is exactly one symbol.

— The error detecting capability is all error patterns of two errors or less.

— The parity check matrix is:

$$\mathbf{H} = [\ \mathbf{I}_m\ \ \mathbf{Q}_{m,k}\ ]$$

where $\mathbf{I}_m$ is an m x m identity matrix and $\mathbf{Q}_{m,k}$ is the submatrix that consists of k columns which are of weight two or more (i.e., have two or more ones in them).

—The generator matrix is:

$$\mathbf{G} = [\ \mathbf{Q}^T_{m,k}\ \ \mathbf{I}_k\ ]$$

where $\mathbf{I}_k$ is a k x k identity matrix and $\mathbf{Q}^T_{m,k}$ is the transpose of the submatrix $\mathbf{Q}$ that consists of m columns and k rows.

Two-way Data Transmission

Error control for a two-way system can be accomplished with both error control coding as well as some rational scheme of data retransmission. A system that utilizes retransmission of messages is said to use an automatic repeat request (ARQ) protocol.

HAMMING ENCODING

HAMENC1 — HAMMING ENCODER #1, TABLE LOOK-UP

The 4-bit information word to be encoded is used as an index into a look-up table. A (7,4) Hamming code represents a 7-bit word with four data bits and three code bits. A (7,4) Hamming code will have 2^4 (16) different codeword possibilities. The 16-element look-up table consists of the pre-encoded (i.e., pre-calculated) codewords. This is the speediest of the encoding techniques, but consumes a lot of memory for anything except the smallest code length. For example, the next Hamming codeword size up from the (7,4) Hamming shown in this example would be a (15,11) Hamming code according to the above formulas. This means that the look-up table would have to be 2^{11} (2,048) elements deep (a total of 4,096 bytes @ 2 bytes per element). The flowchart and HC08 assembly code for HAMENC1 is listed in Appendix A.

HAMENC2 — HAMMING ENCODER #2, MATRIX CALCULATION

HAMENC2 actually performs the matrix arithmetic used to encode information into Hamming codewords. This is done with the generator matrix, $\mathbf{G}$, described earlier. The generator matrix consists of two submatrices: a k x k identity matrix, $\mathbf{I}_k$, and the transpose of matrix Q, $\mathbf{Q}^T_{m,k}$, where matrix Q consists of k columns which are of weight two or more. The generator matrix used to generate the look-up table in HAMENC1 is:

$$\mathbf{G} = [\mathbf{Q}^T_{m,k} \; \mathbf{I}_k]$$

$$\mathbf{G} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

where m = 3 and k = 4.

The 4-bit information word is first bit-wise multiplied (logically anded) by each column in the generator matrix. Each bit in each of the column products is then added together, modulo-2, to create a parity bit for each product. An example of a 4-bit infoword and its generated codeword is given below.

Information Word	**G** Matrix		Code Word
[1 0 1 0]	$\begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$	=	[0 0 1 1 0 1 0]

The flowchart and HC08 assembly code for HAMENC2 is listed in Appendix B. Once an information word is encoded, it may be transmitted over the channel for recovery by the HAMDEC routine explained next.

HAMMING DECODING

ERROR DETECTION AND CORRECTION OF RECEIVED CODEWORD

A 7-bit Hamming codeword will be decoded into the original 4-bit information word even with up to one bit in the codeword being corrupted by the channel. This routine will use matrix math to recover the information word. The parity-check matrix used in the HAMENC routine(s) is:

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

The parity-check matrix has the property such that when a 7-bit codeword generated by the generator matrix and uncorrupted by the channel is multiplied by the transpose of the parity-check matrix, $\mathbf{H}^T$, a zero vector is obtained. The three-element result is called the syndrome. For uncorrupted data, the syndrome should be a zero vector. An example is given below.

Code Word	**H** Transpose Matrix		Syndrome
[0 0 1 1 0 1 0]	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$	=	[0 0 0]

Suppose that we transmit a code vector, **v**, and that an error occurs in the fourth bit position. This is the same as adding a vector, **e**, to the codeword **v** where **e** looks like:

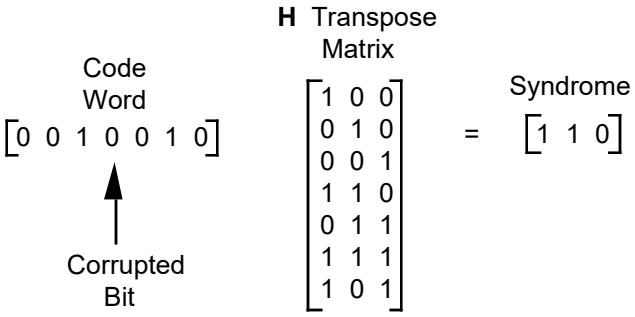
$$\mathbf{e} = [\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ ]$$

By multiplying the received vector by the transpose of the parity-check matrix, we obtain:

$$(\ \mathbf{v} + \mathbf{e}\ )\ \mathbf{H}^T = \mathbf{v}\mathbf{H}^T + \mathbf{e}\mathbf{H}^T = \mathbf{e}\mathbf{H}^T$$

since $\mathbf{v}\mathbf{H}^T = 0$.

The resultant non-zero vector, $\mathbf{e}\mathbf{H}^T$, indicates a problem in the received codeword. As mentioned above, this product of the received vector and the transpose of the parity-check matrix is called the syndrome. An example of a corrupted codeword and its syndrome is given below.



The syndrome bit pattern of a single bit error will be the pattern of one row within $\mathbf{H}^T$.

The row number is numerically equivalent to the corrupted bit position in the received codeword. Thus, by calculating the syndrome and obtaining a non-zero vector we have:

1) Identified that one or two errors have occurred.
2) In the case of a single bit error, we have identified the location of the error within the received word.

The last item to be accomplished is to correct the identified error, which is accomplished by complementing the bit position in the received codeword. The flowchart and HC08 assembly code for HAMDEC is listed in Appendix C.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.