# Hamming Block Code Channel Decoder*

Ed Doering



**Abstract**

Channel encoding inserts additional information into a transmitted bit stream to facilitate error detection and error correction at the receiver. Block coding breaks up a bit stream into words of length k bits and appends check bits to form a codeword of length n bits. A corresponding channel decoder examines the complete codeword, and detects and even corrects certain types of erroneous bits caused by the channel. In the prerequisite project "Hamming Block Code Channel Encoder" you developed a channel encoder using a special class of block code called a Hamming code. In this project, develop the companion channel decoder, and then evaluate the performance of the complete encoder/decoder system.

| | |
|---|---|
| | This module refers to LabVIEW, a software development environment that features a graphical programming language. Please see the LabVIEW QuickStart Guide module for tutorials and documentation that will help you: |
| | • Apply LabVIEW to Audio Signal Processing |
| | • Get started with LabVIEW |
| | • Obtain a fully-functional evaluation edition of LabVIEW |

**Table 1**

NOTE: Visit LabVIEW Setup to learn how to adjust your own LabVIEW environment to match the settings used by the LabVIEW screencast video(s) in this module. Click the "Fullscreen" button at the lower right corner of the video player if the video does not fit properly within your browser window.

## 1 Summary

**Channel encoding** inserts additional information into a transmitted bit stream to facilitate **error detection** and **error correction** at the receiver. **Block coding** breaks up a bit stream into words of length k bits and appends check bits to form a **codeword** of length n bits. A corresponding **channel decoder**

*Version 1.2: Jan 11, 2010 11:29 pm +0000
†http://creativecommons.org/licenses/by/2.0/

examines the complete codeword, and detects and even corrects certain types of erroneous bits caused by the channel.

In the prerequisite project Hamming Block Code Channel Encoder you developed a channel encoder using a special class of block code called a Hamming code. In this project, develop the companion channel decoder, and then evaluate the performance of the complete encoder/decoder system.

## 2 Objectives

1. Develop an (n,k) Hamming block code channel decoder capable of error detection and correction
2. Examine the behavior of the encoded bitstream before and after passing through the decoder
3. Evaluate the performance of the complete encoder/decoder system

## 3 Deliverables

1. Summary write-up of your results
2. Hardcopy of all LabVIEW code that you develop (block diagrams and front panels)
3. Any plots or diagrams requested

NOTE: You can easily export LabVIEW front-panel waveform plots directly to your report. Right-click on the waveform indicator and choose "Export Simplified Image."

## 4 Setup

1. LabVIEW 8.5 or later version

## 5 Textbook Linkages

Refer to the following textbooks for additional background on the project activities of this module; see the "References" section below for publication details:

- Carlson, Crilly, and Rutledge – Ch 13 (basis for notation used in this module)
- Haykin – Ch 10
- Lathi – Ch 16
- Proakis and Salehi (FCS) – Ch 13
- Proakis and Salehi (CSE) – Ch 9
- Stern and Mahmoud – Ch 10

## 6 Prerequisite Modules

If you have not done so already, please complete the prerequisite module Hamming Block Code Channel Encoder. If you are relatively new to LabVIEW, consider taking the course LabVIEW Techniques for Audio Signal Processing[1] which provides the foundation you need to complete this project activity, including: block diagram editing techniques, essential programming structures, subVIs, arrays, and audio.

[1] <http://cnx.org/content/col10440/latest/>

## 7 Introduction

**Error control coding** describes a class of techniques that prepare a digital message bitstream to pass through a noisy channel so that the receiver can detect and in some cases correct transmission errors. The prerequisite project Hamming Block Code Channel Encoder describes how to create a specific type of channel encoder based on the **(n,k) Hamming code**. The codeword length "n" and message length "k" are specific values calculated from the user-defined number of checkbits "q". As discussed in the prerequisite module, the code rate of the Hamming code approaches 1 (100% efficiency) as "q" increases, but the minimum Hamming distance "dmin" is fixed at 3. Therefore, the Hamming code can detect up to two bit errors in a received codeword, and can correct up to one bit error.

The **channel decoder**, a subsystem of the receiver, serves as a complement to the channel encoder in the transmitter. The channel decoder examines each received codeword, indicates detectable errors, fixes correctable errors, and extracts the message. Not all types of errors are detectable nor correctable, therefore the channel decoder can certainly emit garbled messages. Fortunately the channel noise must be rather severe before this becomes a problem.

The channel decoder developed in this project is called a **table lookup syndrome decoder**. View the Figure 1 screencast video to learn how to calculate the **syndrome** of a codeword, how to develop a **lookup table** of most-likely error patterns indexed by syndrome value, and how to use these results as the basic components of a channel decoder capable of detecting and correcting some types of error patterns.

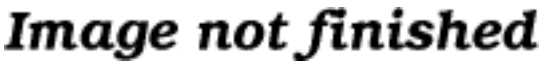


**Figure 1:** [video] Table lookup syndrome channel decoder for Hamming block code

## 8 Procedure

### 8.1 Manual calculations

Work through the syndrome calculation process by hand to lay a good foundation for developing a correct and understandable computer implementation. Write up this work on a separate page.

The end of the Figure 1 screencast video presents an example of a specific Hamming code generator matrix "G", a specific message vector "M" and associated codeword vector "X", and three received versions of the same transmitted codeword with varying severity of bit errors.

1. Determine the parity check matrix "HT" (the transpose of the matrix "H") that corresponds to the generator matrix "G".
2. Write the three received codeword vectors.
3. Calculate the syndrome for each of the three received codewords. Remember to use modulo-2 arithmetic for the matrix calculations.
4. Discuss your results in terms of the potential to detect and correct errors for each of the three received codewords based on their calculated syndromes.

### 8.2 SubVI construction

Build the subVIs listed below. You may already have some of these available from previous projects.

Demonstrate that each of these subVIs works properly before continuing to the next part.

1. hamming_ParityCheckMatrix.vi
2. hamming_SyndromeTable.vi
3. hamming_DetectorCorrector.vi
4. util_BitstreamFromText.vi
5. util_BitstreamToText.vi

### 8.3 Hamming block code channel decoder

Use your top-level application VI from the prerequisite channel encoder project as a starting point for this project.

Review again the background theory presented earlier for the Hamming block code channel decoder, then extend the top-level application VI to decode the output of the channel. Follow the block diagram described near the end of the Figure 1 screencast video.

Display Boolean array front-panel indicators for the following values:

- message – original message words
- encoded message – message words with appended checkbits (transmitted codewords)
- received message – received codewords after passing through noisy channel
- pre-decoding errors – bit error locations in received codewords
- corrected message – received codewords with error correction applied
- post-decoding errors – bit error locations in corrected codewords
- error detected (1-D array) – error detected (non-zero syndrome)

### 8.4 Combined channel encoder/decoder performance

1. Generate 50 words, and begin with 3 checkbits. Run the VI repeatedly and observe the channel decoder output indicators. What bit error rate tends to limit the received codeword errors to single-bit errors?
2. Increase to 4 checkbits, then to 5 checkbits, and so on while holding the bit error rate fixed. Recalling the positive effect of increasing the number of checkbits (increased code rate), what appears to be the negative effect of an increased number of checkbits? Explain.
3. Return to 3 checkbits. Run the VI until you observe a two-bit error in a received codeword. Does the "error detected" indicator work properly? How about the corrected codeword? Explain these results.
4. Set the number of checkbits to 2 and run the VI several times. What is another name (hopefully familiar to you) for this code?

### 8.5 Text messaging

Replace the random number generator in the transmit section with the text data source util_BitstreamFromText.vi. Use the companion subVI util_BitstreamToText.vi to display the receiver output. Experiment with short messages and long messages, making sure that the intermediate Boolean displays make sense.

Experiment with intelligibility in the received message as a function of bit error rate (BER). Determine specific BER values you associate with the following qualitative labels: **excellent**, **good**, **barely acceptable**, and **unintelligible**.

## 9 References

1. Carlson, A. Bruce, Paul B. Crilly, and Janet C. Rutledge, "Communication Systems," 4th ed., McGraw-Hill, 2002. ISBN-13: 978-0-07-011127-1
2. Haykin, Simon. "Communication Systems," 4th ed., Wiley, 2001. ISBN-10: 0-471-17869-1

3. Lathi, Bhagwandas P., "Modern Digital and Analog Communication Systems," 3rd ed., Oxford University Press, 1998. ISBN-10: 0-19-511009-9
4. Proakis, John G., and Masoud Salehi, "Fundamentals of Communication Systems," Pearson Prentice Hall, 2005. ISBN-10: 0-13-147135-X
5. Proakis, John G., and Masoud Salehi, "Communication Systems Engineering," 2nd ed., Pearson Prentice Hall, 2002. ISBN-10: 0-13-061793-8
6. Stern, Harold P.E., and Samy A. Mahmoud, "Communication Systems," Pearson Prentice Hall, 2004. ISBN-10: 0-13-040268-0