Module 7 – Codes, parity, and error detection

Overview

Many digital applications require the use of non-binary quantities such as letters of the alphabet, punctuation marks, question marks etc., all of which must be represented by groups of binary digits. These groups of digits usually form some type of code. The code can then be used in program development.

Stored program design (concept map)

Objectives

At the completion of this module you will be able to:

- list the principles associated with weighted and non-weighted binary code systems
- describe communication code systems including ASCII, EBCDIC and Bar codes
- outline the principles used by digital transducers to provide logic signals proportional to linear and rotational motion
- explain the concepts of parity checking and describe its use in error detection and correction systems.

7.1 Codes

Codes can be classified in many ways however we will consider a few. Numbers can be represented by ‘weighted’ codes in which each bit position is assigned a numerical value. Numbers may also be represented by ‘non-weighted’ codes in which successive code words differ by one bit only. In digital data transmission it is always possible for errors to occur and special codes are used to minimise these errors. These are called ‘error detecting and correcting’ codes.

A group of binary digits representing some quantity in a digital computing system is called a ‘code word’. Code words may be of various lengths depending upon the particular application.

7.2 Weighed codes

You will recall from module 1 that number systems are created by assigning weights (values) to each column position. In the binary system one of the most popular codes is the 8421 code.

This weighted code is often used to represent decimal numbers by coding each digit in the decimal number separately into binary. This is called Binary Coded Decimal (BCD).

The BCD system can be used with different weighting systems. For example in the 7421 code system, the number 7 can be represented by 1000 or 0111. In this case the preferred coding method would be stated by the user.

7.2.1 Two-out-of-five code

Another code which has an interesting quality for data checking is the 74210 code. Note that each code word must have 2 ones in it, or it will be an error. The word11000 is used for zero in this special case. This code is often known as the two-out-of-five code.

7	4	2	1	0	Count
0	0	0	1	1	1
0	0	1	0	1	2
0	0	1	1	0	3
0	1	0	0	1	4
0	1	0	1	0	5
0	1	1	0	0	6
1	0	0	0	1	7
1	0	0	1	0	8
1	0	1	0	0	9
1	1	0	0	0	0

7.2.2 The excess-three code

Another code which has interesting properties is the ‘excess-three’ code, (X_s3). This code is created by adding three to each of the code words in the normal BCD system.

This code is simple to use and no number is represented by four zeros. This has the advantage that zero has a number and is not confused with 'no data' being transmitted in a communication system. The X_S3 code has a 'self-complementing' property. If the one's complement of a code word is obtained, the result is the nines complement code for the original code word.

Decimal number	X_S3 Code			
	8	4	2	1
0	0	0	1	1
1	0	1	0	0
2	0	1	0	1
3	0	1	1	0
4	0	1	1	1
5	1	0	0	0
6	1	0	0	1
7	1	0	1	0
8	1	0	1	1
9	1	1	0	0

7.2.3 Other weighted codes

There are many other codes which are sometimes used in particular situations. These include the 2421, 5421, 84(–2)(–1) codes.

7.3 Non-weighted codes

Examination of the codes used up to now shows that as we count from one decimal value to another, more than one digit in a code word sometimes changes. For example, in 8421 BCD, counting from 3 to 4 results in 0011 to 0100. This causes a **3 digit change** in the code words. If a positioning device encoded in this form were used and one of the pickup sensors was slightly misaligned, a completely wrong result could emerge.

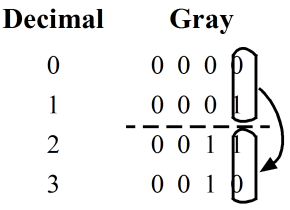
This problem is overcome to a degree by the use non weighted codes or progressive codes as they are often called.

7.3.1 The Gray code

One such code was devised by Mr. E. Gray in 1876. His code, like all progressive codes changes by one bit only as progression occurs from one decimal value to the next.

The Gray code is often referred to as a reflected code. It is created using one/zero combination which are progressively reflected as the system increases in magnitude.

The first two numbers are zero and one and the next two numbers are obtained by reflecting this combination as shown above the dotted line and adding a one in the next highest column. An interesting feature of this code is that the nines complement can be obtained by complementing the most significant digit. In this code more than four bits are needed for numbers greater than fifteen. The Gray code is used extensively in mechanical position encoders because the single digit change from one count to the next prevent the generation of erroneous signals. A table indicating the Gray code is shown. It is easy to convert a binary number into Gray code. A leading zero is placed before the most significant bit of the binary number and then adjacent bits are exclusive or'ed together.



Decimal	Gray Code
0	0 0 0 0
1	0 0 0 1
2	0 0 1 1
3	0 0 1 0
4	0 1 1 0
5	0 1 1 1
6	0 1 0 1
7	0 1 0 0
8	1 1 0 0
9	1 1 0 1
10	1 1 1 1
11	1 1 1 0
12	1 0 1 0
13	1 0 1 1
14	1 0 0 1
15	1 0 0 0

7.3.2 Example 1, decimal to Gray conversion

Determine the Gray code for 14 decimal.

Decimal 14 in Binary is 1 1 1 0

Add a leading zero 0 1 1 1 0

Exor each pair

Gray for 14 is 1 0 0 1

7.3.3 Example 2, Gray to binary conversion

The procedure to convert a Gray codeword to a binary word is not as simple. One technique is to start at the most significant bit of the gray word and consider each bit left to right. The first 1 is written and repeated for every bit in the gray word until the next 1 occurs, then 0's are written until the next 1 occurs. Then 1's are written etc. An example best illustrates this procedure.

Convert the Gray codeword 1110001010 to binary.

Gray is	1	1	1	0	0	0	1	0	1	0
	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
Binary is	1	0	1	1	1	1	0	0	1	1

7.4 Other codes

Currently we have only considered codes for numerical applications. In many systems there is a need for letters of the alphabet and other printing symbols to be represented. No single universal code is generally accepted by all computer manufacturers however some codes are frequently used.

7.4.1 The ASCII code

One such code is the American Standard Code for Information Interchange (ASCII). A hexadecimal listing of this code is shown in the figure following. ASCII is used almost universally in digital communications. It has special control characters which include ENQ (enquiry), used to check the status of a receiver; ACK (acknowledge), used to indicate if a message has been successfully received and NAK (negative acknowledge), used to indicate if an error has occurred. ASCII is a seven bit code which can identify 128 characters including 94 graphic characters and 34 non-printing control characters.

Low / High		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
$B_7B_6B_5B_4$		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
0001	1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
0010	2	SP	!	”	#	$	%	&	’	(	)	*	+	,	-	.	/
0011	3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
0100	4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0101	5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	.	-
0110	6	\	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
0111	7	p	q	r	s	t	u	v	w	x	y	z	{	l	}	~	DEL

Binary - Hex - ASCII

ASCII Code Listing

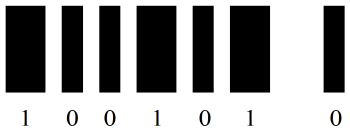
7.4.2 The EBCDIC code

Another code which was designed to compete with ASCII was devised by the IBM company. It is called the Extended Binary Coded Decimal Interchange Code or EBCDIC. It has eight bits per codeword and can therefore represent 256 characters. It is listed in many texts on computers and therefore will not be reproduced here.

7.4.3 Bar codes

Bar codes are digital messages encoded using the widths of printed bars on tags of paper or adhesive labels. They allow fast and accurate hand entry of small amounts of data. Point-of-sale (POS) systems in grocery stores rely on reading these codes at the checkout. The systems automatically enter the price on the customers account, adjust the stores inventory and initiate reordering procedure. Other uses are in stock control in organisations, identification of assemblies for process monitoring, work scheduling and in remote data collection.

There are many different possible coding systems for bar codes however the majority of schemes now use a two-level code. In this, a wide bar (or space) represents a logic one and a narrow bar (or space) represents a logic zero (or vice versa). Usually the first two bars of a tag are used to define the initial value of a width (wide or narrow) and all other bars or spaces are compared to this standard value and assigned values accordingly depending upon their widths. The example shown is a typical two-level code.



Bar Code

Bar codes can be read in one direction only or bidirectionally if required. Most codes include a checksum digit encoded at the end of the bar code to provide security against reading errors. Most codes also use some form of parity checking such as is available in a two-out-of-five code. One version of the two-out-of-five code often used in bar code systems is now described. Consider a version where the bars convey the information only. The code table shown is based on a weighted system of 12470.

This decimal code has start and stop symbols at each end of the tag. These are recognised as such as they have a very wide space (M) and three digits only associated with them. A number encoded in this system is shown.

Bar-Code Pattern	Character
00110	0
10001	1
01001	2
11000	3
00101	4
10100	5
01100	6
00011	7
10010	8
01010	9
M110	Start
101M	Stop

M = wide space

With this code there are three checks which can be effected to validate the decoded data.

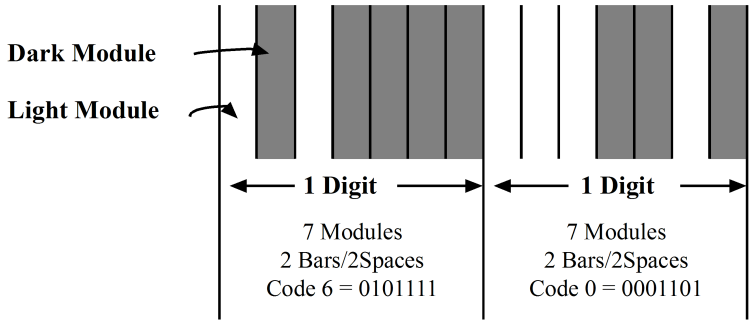
Firstly each character is checked to determine if there are two wide bars only. One or three wide bars constitute an error in this code. Next there must be an integer multiple of five in the total number of bars counted between the start and stop symbols. Finally, to verify the checksum, the last digit on the tag is usually the least significant digit of a number which is the sum of all the numbers on the tag. If this is not the case a reading error has occurred and the data is invalid. Many other codes and systems are used in practice and each has its own check systems.

7.4.4 The Universal Product Code (UPC)

The Grocery industry has adopted the UPC as an industry standard and labels bearing this code are now appearing on virtually every kind of grocery product. The symbol size is infinitely variable to accommodate the range of printing techniques and reading equipment available. The UPC is currently based on a 12 digit numbering system.

Each manufacturer of foodstuffs is issued with a unique 6 digit number for identification while the remaining 6 digits are assigned to generic product categories such as tissues, soup, soap, peas etc. The entire 12 digit number can also be combined with error checking information.

The structure for the UPC code is shown in the diagram below:



UPC Structure for Each Digit

Points to note are:

- Overall shape is rectangular.
- Each digit is represented by 7 modules comprising 2 bars and 2 spaces each of variable width.
- Each digit is independent.

A typical UPC symbol is shown below:

The number on the extreme left is used to describe the number system being used in the symbol. The digits are also always written on the code to allow for manual decoding if necessary.

Notice the two six digit numbers between the two guard bars at the left, centre and right.

On the left-to-right basis, each digit on the left side of the centre bars begins with a light space and ends with a dark bar; digits to the right of the centre bars begin with a dark bar and end with a light space. Dark modules represent 1's while light modules represent 0's. The number of dark modules per digit on the left side is always three or five; the number of dark modules is always two or four for right-hand digits. Encoding is identical for all similar digits on a given side of the symbol, whether it is a number system digit, UPC digit, or check digit. The first two bars at either end encode the guard bar pattern, 101. The guard bars in the centre encode as 01010.

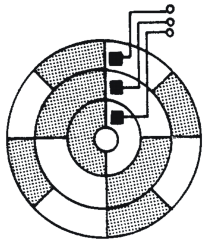
The code table for the UPC is shown below.

Decimal Value	Left Characters (Odd Parity – O)	Right Characters (Even Parity – E)
0	0001101	1110010
1	0011001	1100110
2	0010011	1101100
3	0111101	1000010
4	0100011	1011100
5	0110001	1001110
6	0101111	1010000
7	0111011	1000100
8	0110111	1001000
9	0001011	1110100

7.5 Mechanical encoders

Often the input to a digital system may arise from the position of a lever on the rotation of a shaft. This can be done easily with a mechanical encoder similar to the diagram shown.

A disk with concentric bands containing regions of light and dark areas is constructed and attached to the shaft or lever to be sensed. The disk position can be sensed either by shining a light through it optical sensing, or by a set of brushes making contact with it providing electrical sensing. For the disk shown a three bit binary number proportional to rotational displacement will be obtained.

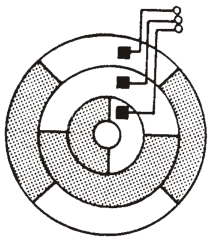


Binary coded disc

The precision of the device increases if more bands are added resulting in a larger binary word. In the disk shown a one is produced from the dark bands and a zero from the light bands.

This results in readouts of 000 for angles between 0 and 45 degrees, 001 between 45 and 90 degrees, and so on. The contact or sensor width is a factor in determining accuracy particular at the interfaces between code changes.

One way to minimise these problems is to use a disk encoded with the Gray code as shown. Here, bits change value only one at a time for successive binary numbers. This reduces the error to only one bit in the number being encoded.



Gray coded disc

7.6 Parity and error detection and correction

Coded information is often fed into a logical machine in the form of punched paper tape in which **a hole** represents a **1** and the **absence of a hole** represents a **0**.

Since tape punches are of necessity electro-mechanical devices they are inherently less reliable than pure electrical devices. Error detecting codes are used to detect any failure of the punch unit. Also in data transmission, mutilation of a message may occur where a 1 in changed to a zero or vice versa.

Such an error must be at least detected and if possible corrected before the information is further processed. One code which can be used for error detection is the 74210 code. Each codeword has always 2 ones in it and this characteristic can always be checked continuously.

7.6.1 Parity checking

A simple method widely used is known as **parity checking**. In this, an extra bit known as a parity bit is associated with each codeword, the value of the bit (i.e. 0 or 1) being determined by some relationship to the digits in the codeword.

Even or **odd** parity means that the total number of ones in a codeword are adjusted by varying the parity bit to be either even or odd as required. For example consider even and odd parity bits associated with the BCD code.

Decimal	Even Parity	Odd Parity
	8421	8421
0	0000 0	0000 1
1	0001 1	0001 0
2	0010 1	0010 0
3	0011 0	0011 1
4	0100 1	0100 0
5	0101 0	0101 1
6	0110 0	0110 1
7	0111 1	0111 0
8	1000 1	1000 0
9	1001 0	1001 1

Parity bit added to make code-word have an even number of ones.

A **parity check** will tell if a single digit has been changed (including the parity bit itself) but it **cannot define which bit is in error**. Note also if **2** bits complement, then the parity can still be correct.

7.6.2 Error detection and correction

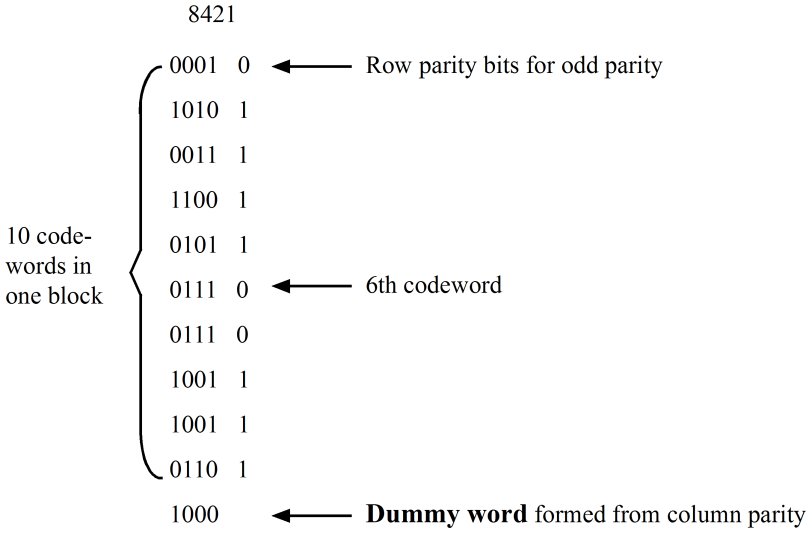
Some error detecting and correcting codes can locate the bits actually in error and complement them. Such codes are known as error detecting and correcting codes (EDC). We will look at a simple one now.

The basis of an EDC system is the precise location of the bit in error and thus 'correction' is just a matter of complementing that bit.

Let us consider a system of EDC using **odd parity bits after** the **LSB** of an **8421 codeword** (row parity).

Suppose these codewords are arranged in **blocks** of data with 10 codewords per block.

A **dummy** word formed from the **parity** bits of each **column** is added at the end of each block (column parity).



Notice that the parity bit for the dummy word is blank, since the **row** and **column** parity requirements may be different.

7.6.3 Example

Let us suppose that this data block is transmitted to another digital system and that the 6th word in the block is received as **01100** instead of **01110**.

A 6th **row** parity error will be noticed for this word but of course we have no means of knowing where the error is in that word.

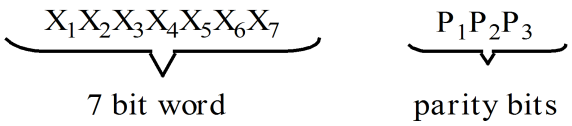
When the block has finished being transmitted and the column parity checked by the dummy word an error will be found in the 4th **column**.

So our precise error is 6th word, 4th column which specifies the bit in error. Complementing this bit will fix the error.

Obviously this simple system could not tackle multiple errors but some extremely ingenious systems have been devised for this purpose.

7.6.4 Hamming code system

One such system devised by R.W. Hamming uses 3 parity bits for a 7 bit word.



where P_1 = parity for $X_1X_3X_5X_7$

P_2 = parity for $X_1X_2X_5X_6$

P_3 = parity for $X_1X_2X_3X_4$

Activity 7.1

1. Express the following decimal numbers in 8421 code: (a) 27; (b) 628; (c) 4372.
2. Decode the following numbers expressed in 8421 code: (a) 0111 1000 0101; (b) 1001 1001 0001 0101.
3. Express the following decimal numbers in 4221 code: (a) 648; (b) 2643.
4. Decode the following numbers expressed in 4221 code: (a) 0011 1100 1110; (b) 0010 0110 1110 1111.
5. Express the following decimal numbers in excess-3 code: (a) 821; (b) 6243.
6. Express the Gray Code for the decimal number 47.
7. Express the following in ASCII code: (a) PIN 4; (b) RAM.
8. Refer to the ELE1301 course page on 'Study Desk' and complete the following experiments:
   a. Home experiment 7-1 – Code conversion 1
   b. Home experiment 7-2 – Code conversion 2

Selected solutions to activity 7.1

1. a. 00100111
2. a. 785
3. a. 1100 0110 1110
4. a. 368
5. a. 1011 0101 0100
6. 111000
7. a. 0101 0000 0100 1001 0100 1110 0011 0100

Self assessment

1. Explain the features of:
   i. a two-out-of-five code.
   ii. a non-weighted code.
   iii. an excess-three code

   Include in your answer a truth table (10 rows minimum) for each code.
2. Describe, with the aid of a sketch, the construction of a mechanical encoder suitable for providing binary information from a rotating shaft.
3. Explain how the precision of the above encoder can be improved.
4. Using an error detection and correction system (EDC) as an example, explain how errors in digital data transmissions may be minimized.