ALPHANUMERIC CODE

Earlier computers were used only for the purpose of calculations i.e. they were only used as a calculating device. But now computers are not just used for numeric representations, they are also used to represent information such as names, addresses, item descriptions etc. Such information is represented using letters and symbols. Computer is a digital system and can only deal with l's and 0's. So, to deal with letters and symbols they use alphanumeric codes.

Alphanumeric codes, also called character codes, are binary codes used to represent alphanumeric data. The codes write alphanumeric data, including letters of the alphabet, numbers, mathematical symbols and punctuation marks, in a form that is understandable and process able by a computer. Using these codes, we can interface input-output devices such as keyboards, monitors, printers etc. with computer.

Several coding techniques have been invented that represent alphanumeric information as a series of l's and O's. The earliest better-known alphanumeric codes were the Morse code used in telegraph and 12-bit Hollerith code used when punch cards were used as a medium of inputting and outputting data. As the punched cards have completely vanished with the evolution of new mediums, the Hollerith code was rendered obsolete. Now the ASCII and EBCDIC codes are the two most widely used alphanumeric codes. The ASCII Code is very popular code used in all personal computers and workstations whereas the EBCDIC code is mainly alphanumeric code, the UNICODE, has evolved to overcome the limitation of limited character encoding as in case of ASCII and EBCDIC code.

American Standard-Code for Information Interchange (ASCII)

The American Standard-Code for Information Interchange (ASCII) pronounced "as-kee" is a 7-bit code based on the ordering of the English alphabets. The ASCII codes are used to represent alphanumeric data in computer input/output.

Historically, ASCII developed from telegraphic codes. It was first published as a standard in 1967. It was subsequently updated and many versions of it were launched with the most recent update in 1986. Since it is a seven-bit code, it can almost represent 128 characters. These include 95 printable characters including 26 upper-case letters (A to Z), 26 lowercase letters (a to z), 10 numerals (0 to 9) and 33 special characters such as mathematical symbols, space character etc. It also defines codes for 33 non-printing obsolete characters except for carriage return and/or line feed. The below table lists the 7 bit ASCII code containing the 95 printable characters.

An eight-bit version of the ASCII code, known as US ASCII-8 or ASCII-8, has also been developed. Since it uses 8-bits, so this version of ASCII can represent a maximum of 256 characters.

The table below lists some ASCII-8 codes.

When the ASCII -7 codes was introduced, many computers dealt with eight -bit groups (or bytes) as the smallest unit of information. This eight bit code was commonly used as parity bit for detection of error on communication lines. Machines that did not use the parity bit typically set the eighth bit to O. In that case, the ASCII code format would be

X_7 X_6 X_5 X_4 X_3 X_2 X_1 X_0‘ In case of ASCII-7 code, if this representation is chosen to represent characters then X7 would always be zero. So, the eight-bit ASCII-7 code for ‘A’ would be 01000001 and for 4’ would be 00101011.

Digits $X_3X_2X_1X_0$	$X_7X_6X_5X_4$ (Zoned bits) 0101	1010	1011
0000	0		P
0001	I	A	Q
0010	1	B	R
0011	2	C	S
0100	3	D	T
0101	4	E	U
0110	S	F	V
0111	6	G	X
1000	7	H	Y
1001	8	I	Z
1010	9	J	
1011		K	
1100		L	
1101		M	
1110		N	
1111		O	

ASCII-8 Code table containing code For some characters

Example 1: With an ASCII-7 keyboard, each keystroke produces the ASCII equivalent of the designated character. Suppose that you type PRINT X. What is the output of an ASCII-7 keyboard?

Solution: The sequence is as follows:

The ASCII-7 equivalent of P = 101 0000

The ASCII-7 equivalent of R = 101 0010

The ASCII-7 equivalent of I = 1001010

The ASCII-7 equivalent of N = 100 1110

The ASCII-7 equivalent of T = 1-010100

The ASCII-7 equivalent of space = 010 0000

The ASCII-7 equivalent of X = 101 1000

So the output produced is 10100001010010100101010011101010100010000001011000. The output in hexadecimal equivalent is 50 52 49 4E 54 30 58

Example2: A computer sends a message to another computer using an odd-parity bit. Here is the message in ASCII-8 Code.

1011 0001

1011 0101

1010 0101

1010 0101

1010 1110

What do these numbers mean?

Solution

On translating, the 8-bit numbers into their equivalent ASCII-8 code we get the word 1011 0001 (Q), 10110101 (U), 10100101 (E), 1010 0101 (E), 1010 1110 (N)

So, on translation we get QUEEN as the output.

ASCII 7 Code table containing 95 printable characters

(Digits) $X_3X_2X_1X_0$	$X_6X_5X_4$(Zoned bits) 010	011	100	101	110	111
0000	SP	0	@	P	a	p
0001	!	1	A	Q	b	q
0010	"	2	B	R	c	r
0011	#	3	C	S	d	s
0100	$	4	D	T	e	t
0101	%	5	E	U	f	u
0110	&	6	F	V	g	v
0111	(	7	G	W	h	w
1000	)	8	H	X	i	x
1001	*	9	I	Y	j	y
1010	*	:	J	Z	k	z
1011	+	;	K	[	l	{
1100	'	<	L	\	m	;
1101	-	+	M	]	n	y
1110	.	>	N	^	o	~
1111	/	?	O	-		DEL

The format of ASCII code for each character is X_6‘ ·X_5, X_4, X_3, X_2, X_1‘ X_0 where each X is 0’ or 1. For instance, letter D is coded as .1000100. For making reading easier, we leave space as follows: 1000100.

Similarly, from the above table, we see that letter ‘A’ has X_6 X_5X_4 of 100 and X_3 X_2 X_1 X_o of *0001 (A).* Similarly, the digit ‘9’ has X_6 X_5 X_4 values of 011 and X_3 X_2 X_1 X_0 of 1001 so the ASCII-7 code for digit 9 is 0111001.

More examples are:

The ASCll-7 code for ‘d’ is 1100100 as seen from the table 3.4.

The ASCll-7 code for ‘+’ is 0101011 as seen from the table 3.4.

Extended Binary Coded Decimal Interchange Code (EBCDIC)
The Extended Binary Coded Decimal Interchange Code (EBCDIC) pronounced as “ebi-si disk” is another frequently used code by computers for transferring alphanumeric data. It is 8-bit code in which the numerals (0-9) are represented by the 8421 BCD code preceded by 1111. Since it is a 8-bit code, it can almost represent 23 (= 256) different characters which include both lowercase and uppercase letters in addition to various other symbols and commands.

Digits $X_3X_2X_1X_0$	$X_7X_6X_5X_4$(Zoned bits) 0100	0101	0110	0111	1000	1001	1100	1101	1110	1111
0000	SP	&		a	j	-	{		/	0
0001				b	k	s	A	J		1
0010				c	l	t	B	K	S	2
0011				d	m	u	C	L	T	3
0100				e	n	v	D	M	U	4
0101				f	o	w	E	N	V	5
0110				g	P	x	F	O	W	6
0111				h	q	y	G	P	X	7
1000				i	r	z	H	Q	Y	8
1001						l	R	Z	9	
1010	#	!	:		;					
1011	-	$	,		#					
1100	<	*	%		@					
1101	(	)			·					
1110	+	;	>		=					
1111	[	]	?		"					

EBCDIC was designed by IBM corp. so it is basically used by several IBM· models. In this code, we do not use a straight binary sequence for representing characters, as was in the case of ASCII code. Since it is a 8-bit code, so it can be easily grouped into groups of 4 so as to represent in arm of hexadecimal digits. By using the hexadecimal number system notation, the amount of digits used to represent various characters and special characters using EBCDIC code is reduced in volume of one is to four. Thus 8-bit binary code could be reduced to 2

hexadecimal digits which are easier to decode if we want to view the internal representation in memory. The above table lists the EBCDIC code for certain characters.

Read the above table as you read the graph. Suppose you want to search for EBCDIC code for letter 'A'. To that case, the value of X3 X2 Xl X0 bits is 0001 and value X7 X6 X5 X4 bits is 1100.

Therefore, EBCDIC code for letter 'A' is 11000001(A). Similarly, the EBCDIC code for 'B' is 11000010(B).

The EBCDIC code '=' is 01111110

The EBCDIC code for '$' is 0101 1011

UNICODE

The ASCII and EBCDIC encodings and their variants that we have studied suffer from some limitations.

1. These encodings do not have a sufficient number of characters to be able to encode alphanumeric data of all forms, scripts and languages. As a result, they do not permit multilingual computer processing.

2. These encoding suffer from incompatibility. For example: code 7A (in hex) represents the lowercase letter 'Z' in ASCII code and the semicolon sign ';' in EBCDIC code.

To overcome these limitations, UNICODE also known as universal code was developed jointly by the Unicode Consortium and the International Organization for Standardization (ISO). The Unicode is a 16-bit code so it can represent 65536 different characters. It is the most complete character encoding scheme that allows text of all forms and languages to be encoded for use by computers. In addition to multilingual support, it also supports a comprehensive set of mathematical and technical symbols, greatly simplifying any scientific information interchange.

UNICODE has a number of uses

1. It is increasingly being used for internal processing and storage of text. Window NT and its descendants, Java environment, Mac OS all follow Unicode as the sole internal character encoding.

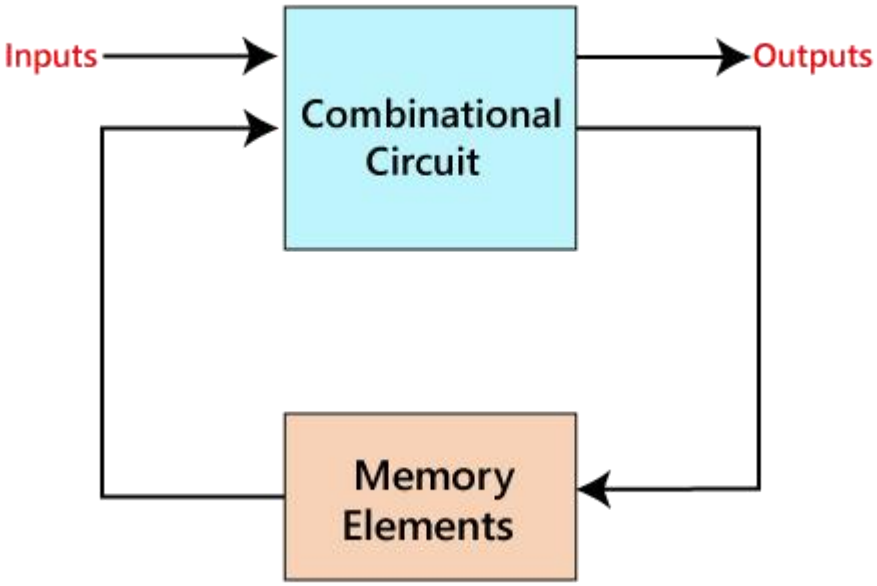
2. All World Wide Web consortium recommendations have used Unicode as their document character set since HTML 4.0.

3. It partially addresses the new line problem that occurs when trying to read a text file on different platforms. It defines a large number of characters that can be recognized as line terminators.

Unicode is currently being adopted by top computer industry leaders like Microsoft, Apple, Oracle, Sun, SAP and many more in their products.

Introduction

In our previous sections, we learned about combinational circuit and their working. The combinational circuits have set of outputs, which depends only on the present combination of inputs. Below is the block diagram of the synchronous logic circuit.



The sequential circuit is a special type of circuit that has a series of inputs and outputs. The outputs of the sequential circuits depend on both the combination of present inputs and previous outputs. The previous output is treated as the present state. So, the sequential circuit contains the combinational circuit and its memory storage elements. A sequential circuit doesn't need to always contain a combinational circuit. So, the sequential circuit can contain only the memory element.

Difference between the combinational circuits and sequential circuits are given below:

	Combinational Circuit	Sequential Circuit
1)	The outputs of the combinational circuit depend only on the present inputs.	The outputs of the sequential circuits depend on both present inputs and present state (previous output).
2)	The feedback path is not present in the combinational circuit.	The feedback path is present in the sequential circuits.

3)	In combinational circuits, memory elements are not required.	In the sequential circuit, memory elements play an important role and require.
4)	The clock signal is not required for combinational circuits.	The clock signal is required for sequential circuits.
5)	The combinational circuit is simple to design.	It is not simple to design a sequential circuit.

Types of Sequential Circuits

Asynchronous sequential circuits

The clock signals are not used by the **Asynchronous sequential circuits**. The asynchronous circuit is operated through the pulses. So, the changes in the input can change the state of the circuit. The asynchronous circuits do not use clock pulses. The internal state is changed when the input variable is changed. The un-clocked flip-flops or time-delayed are the memory elements of asynchronous sequential circuits. The asynchronous sequential circuit is similar to the combinational circuits with feedback.

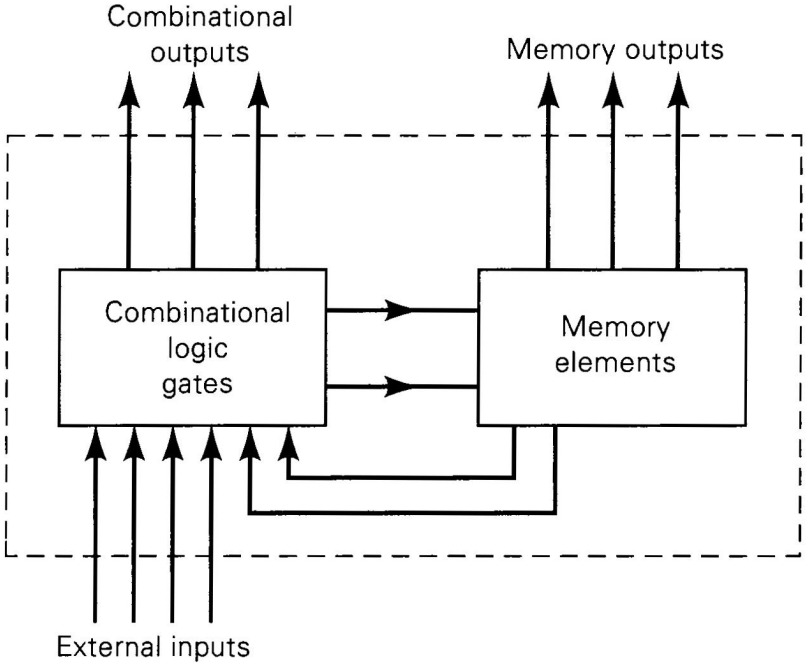
Synchronous sequential circuits

In synchronous sequential circuits, synchronization of the memory element's state is done by the clock signal. The output is stored in either flip-flops or latches(memory devices). The synchronization of the outputs is done with either only negative edges of the clock signal or only positive edges.

FLIP-FLOP

The logic circuits considered thus far have been combinational circuits whose output levels at any instant of time are dependent on the levels present at the inputs at that time. Any prior input-level conditions have no effect on the present outputs

FIGURE 5-1 General digital system diagram.



because combinational logic circuits have no memory. Most digital systems are made up of both combinational circuits and memory elements.

Figure 5-1 shows a block diagram of a general digital system that combines combinational logic gates with memory devices. The combinational portion accepts logic signals from external inputs and from the outputs of the memory elements. The combinational circuit operates on these inputs to produce various outputs, some of which are used to determine the binary values to be stored in the memory elements. The outputs of some of the memory elements, in turn, go to the inputs of logic gates in the combinational circuits. This process indicates that the external outputs of a digital system are a function of both its external inputs and the information stored in its memory elements.

The most important memory element is the **flip-flop,** which is made up of an assembly of logic gates. Even though a logic gate, by itself, has no storage capability, several can be connected together in ways that permit information to be stored. Several different gate arrangements are used to produce these flip-flops (abbreviated FF).

Figure 5-2(a) is the general type of symbol used for a flip-flop. It shows two outputs, labeled Q and $\overline{Q}$, that are the inverse of each other. $Q/\overline{Q}$ are the most common designations used for a FF's outputs. From time to time, we will use other designations such as $X/\overline{X}$ and $A/\overline{A}$ for convenience in identifying different FFs in a logic circuit.

The Q output is called the *normal* FF output, and $\overline{Q}$ is the *inverted* FF output. Whenever we refer to the state of a FF, we are referring to the state of its normal

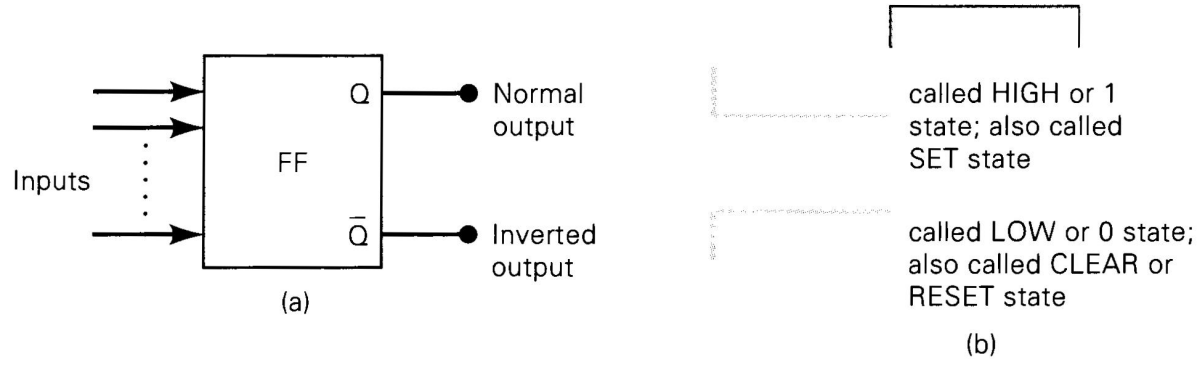


FIGURE 5-2 General flip-flop symbol and definition of its two possible output states.

(Q) output; it is understood that its inverted output ($\overline{Q}$) is in the opposite state. For example, if we say that a FF is in the HIGH (1) state, we mean that $Q = 1$; if we say that a FF is in the LOW (0) state, we mean that $Q = 0$. Of course, the $\overline{Q}$ state will always be the inverse of Q.

The two possible operating states for a FF are summarized in Figure 5-2(b). Note that the HIGH or 1 state ($Q = 1/\overline{Q} = 0$) is also referred to as the **SET** state. Whenever the inputs to a FF cause it to go to the $Q = 1$ state, we call this *setting* the FF; the FF has been set. In a similar way, the LOW or 0 state ($Q = 0/\overline{Q} = 1$) is also referred to as the **CLEAR** or **RESET** state. Whenever the inputs to a FF cause it to go to the $Q = 0$ state, we call this *clearing* or *resetting* the FF; the FF has been cleared (reset). As we shall see, many FFs will have a **SET** input and/or a **CLEAR (RESET)** input that is used to drive the FF into a specific output state.

As the symbol in Figure 5-2(a) implies, a FF can have one or more inputs. These inputs are used to cause the FF to switch back and forth ("flip-flop") between its possible output states. We will find out that most FF inputs need only to be momentarily activated (pulsed) in order to cause a change in the FF output state, and the output will remain in that new state even after the input pulse is over. This is the FF's *memory* characteristic.

The flip-flop is known by other names, including *latch* and *bistable multivibrator*. The term *latch* is used for certain types of flip-flops that we will describe. The term *bistable multivibrator* is the more technical name for a flip-flop, but it is too much of a mouthful to be used regularly.

A digital logic circuit that can store a single BIT of information, and is therefore used as the basis for the construction of MEMORY chips, LATCHES and the REGISTERS within processors. A flip-flop can exist in two states, with either a high or low voltage at its output, and flips from one state to the other at each pulse of a CLOCK SIGNAL. Two different implementations of flip-flop are commonly used, called the D flip flop and the j-K flip flop.

Electronic flip flop circuits are electronic circuits with two stable states used to store binary data. They are essential in data storage devices. Such a circuit has one or more control inputs and one or two outputs. By applying varying input, the data stored can be changed. In sequential logic, the flip flop is the basic storage element. They are fundamental building blocks of electronics systems such as computers and communication devices.

A flip flop stores a single bit or binary digit of data. The two states of a flip flop represent "one" and "zero." The output and the next state of a flip flop depend on its current input and current state when used in a finite-state machine. Used for counting of pulses and synchronizing variably-timed input signals to some reference timing signal, a flip flop can be level-triggered or edge-triggered. Level-triggered flip flops can be asynchronous, transparent, or opaque, while edge-triggered flip flops can be synchronous or clocked.

Flip flops are related to clocked devices or clocking. Clocked devices ignore their inputs except at the transition of a dedicated clock signal. A flip flop either change or retain its output signal

based on the values of the input signals at the transition caused by clocking. Some flip flops change the output on the rising edge of the clock, others on the falling edge. Since the elementary amplifying stages are inverting, two stages can be connected in cascade to form the needed non-inverting amplifier.

COMMON AND PRACTICAL USES OF FLIP FLOPS

A flip flop has many possible uses. In digital electronics, edge-triggered flip flops are used as a main component for sequential circuits. Among its uses is storing or transferring binary data from a certain location to another and as a counter. Some applications make use of the flip flop's clocked operation, and such applications fall under the category of sequential circuits. Flip flops are seen in counters, storage registers, shift registers, frequency divider circuits, and data transfer.

COUNTERS

These electronic devices are widely used in electronics especially in digital systems. Counters count the number of a specific event occurring in a specific interval of time. Because counters remember past states, they need to have a memory and use flip flops for this purpose. Counters could be synchronous or asynchronous. For synchronous counters, flip flops are connected to the same clock signal. These flip flops will then trigger at the same time. For asynchronous counters, flip flops are connected and complemented together.

REGISTERS

As mentioned earlier, flip flops store single bits of data; either "one" or zero". On the other hand, registers are used to store multiple bits of data. But as such, flip flops are used to design registers.

DATA TRANSFER

This is the process of transferring data from one register to another. Nowadays, data can be transferred using flip flops.