Imperial College
London

Topic 12

Digital Basics

Prof Peter Y K Cheung
Dyson School of Design Engineering
Imperial College London

URL: www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/
E-mail: p.cheung@imperial.ac.uk

Although the first half of this module is about analogue circuits and signals, we will nevertheless introduce the topic of digital representations and signals in this lecture. We will come back to digital in later lectures.

Learning outcomes on digital electronics

- Understand the **formalism of logic** and able to **analyse logical processes**.
- Implement **simple logical operations** using combinational logic circuits.
- Understand common forms of **number representation** in digital electronic circuits and to be able to convert between different representations.
- Understand the logical operation of simple **arithmetic** and other **MSI circuits (Medium Scale Integrated Circuits)**
- Understand the concepts of **sequential circuits** enabling you to analyse sequential systems in terms of **state machines** and **counters**.
- Understand how **digital storage** (e.g. memory) works and how its **content is accessed**.
- Understand the basics of **microprocessors** and **microcontrollers**.
- Able to **integrate hardware and software** together in a simple electronic system.
- Interface electronic circuits to the physical world and **process analogue signals on microcontroller** systems in digital form.

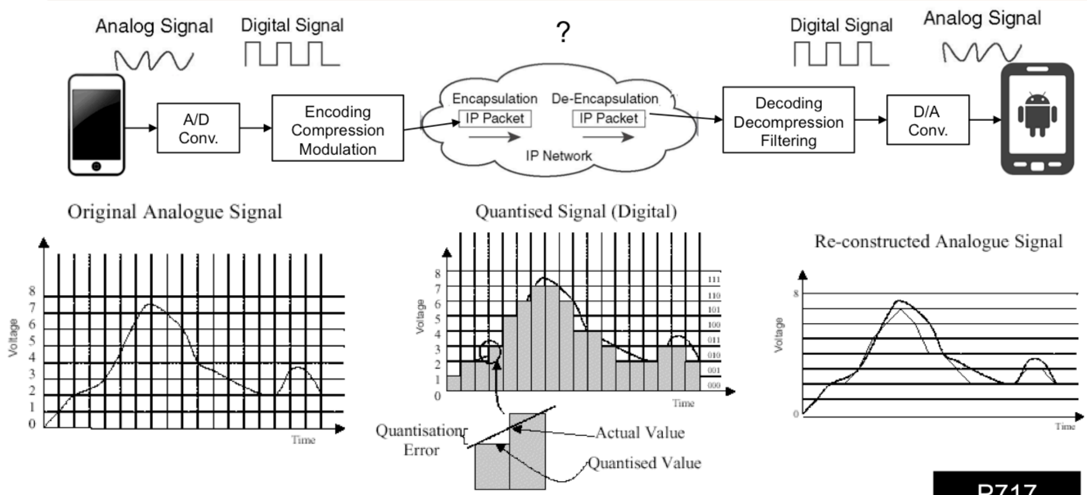
PYKC 4 June 2020 DE1.3 - Electronics 1 Topic 12 Slide 2

By way of introduction to digital electronics, I will spend the next period just going through some of the basic ideas in digital circuits. If you know this already, just be patient. Some of your classmates may not be familiar with at least some aspects of this.

I will be covering in this course quite a bit of digital electronics, but not down to transistor level. However, you will be learning something about what's inside a digital circuits, the idea of combinational and sequential circuits; something about digital counters, microprocessors. You will also learn how to get digital hardware and computer software working together to do something interesting.

Although this lecture introduces you to digital circuits, I will not come back to digital again until the second part of the course.

Analogue vs Digital



P717

- Most physical phenomena are in the analogue domain.
- Most modern electronics systems operate in the digital domain.
- Analogue-to-Digital (A/D) converters, and Digital-to-Analogue (D/A) converters links the two worlds together.

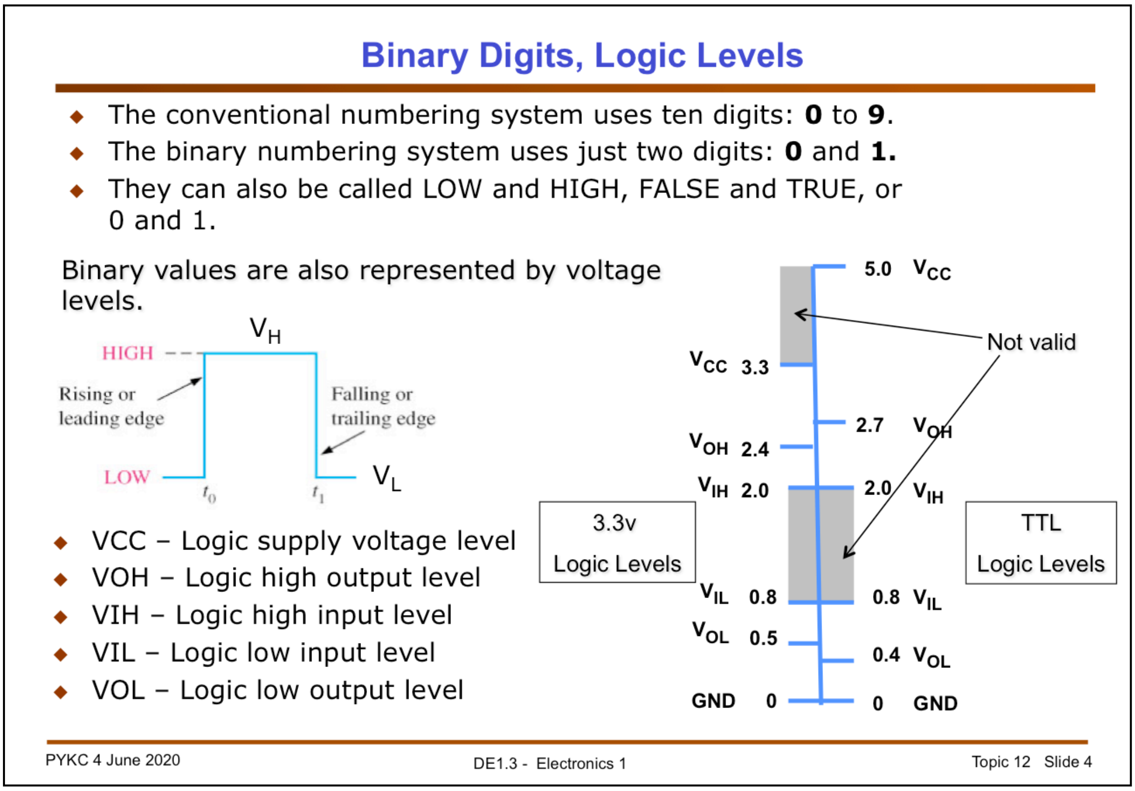
It is important for you to appreciate the high-level view of an electronic system, end-to-end. Shown here is a mobile phone linked to another mobile phone. The speech signal, like many electrical signal in the physical world, are analogue in nature. That is, the signal varies continuously in time and in amplitude. A modern electronic system converts the analogue signal into digital form in two steps. It **samples** the data into discrete time, a process known as "**sampling**". It then **digitize** each sample into discrete levels, a process known as "**quantization**".

You will learn in the second year course in EEE that the sampling processing does NOT destroy information. We know how to recover the original signal without loosing any information. However, quantization will always loose information – we will only have an approximation of the original signal.

Once the speech is in digital form, it goes through many digital circuits which compresses the speech signal so that you try to send as little information as possible, these are then turn into electrical signals that are suitable for transmission through air, cable or optical fibre. This process is called modulation.

For modern phones, the transmission could very well be via the internet (known as Voice over IP or VoIP).

At the receiving end, the reverse happens.



Digital information are handled in electronics a binary bits, i.e. 0's and 1's. In electrical signals forms, these are represented by electrical voltage V_L and V_H.

Very common in electronics are two standards of voltage levels: TTL (stands for transistor-transistor-logic) and 3.3V Low Voltage (3.3LV). For TTL, the supply voltage (V_{CC}) is 5V with 0V as the reference. For 3.3V logic, the supply is 3.3V. In both cases, the reference voltage is 0V or Ground (GND).

For this course, we will exclusively use 3.3V logic. That mean Logic 0 (False) is 0V and logic 1 (True) is 3.3V.

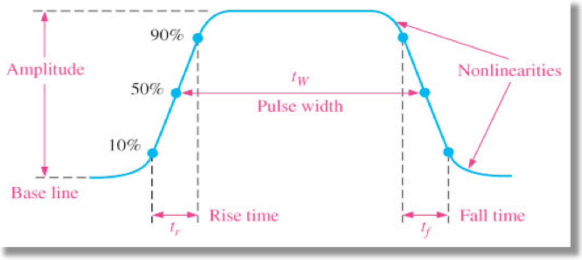
In logic circuits we define four "threshold" voltages: V_{OH} is the **guaranteed minimum** voltage FROM AN OUTPUT NODE which is a logic HIGH. V_{IH} is the **required minimum** voltage before AN INPUT NODE would regard the signal to be logic HIGH. Similarly for the low threshold voltages.

Therefore if you are using TTL logic, the gap between V_{OH} and V_{IH} is 0.7V. In other words, a high output signal could be "corrupted" by noise up to 0.7V and the logic circuit should still interpret the signal as logical high (or '1'). This difference is called "noise immunity".

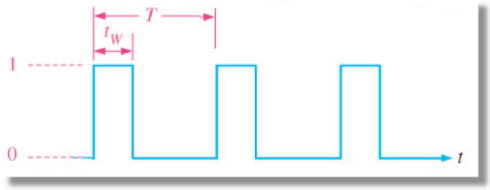
The shaded region in the voltage scale are illegal "no-man" zone. A logic signal node should never take a voltage value in this region.

Digital Waveforms

- Major parts of a digital pulse
- Base line
- Amplitude
- Rise time (t_r)
- Pulse width (t_W)
- Fall time (t_f)
- Period (T)
- Frequency (f)



$f = 1/T$ in Hz



The duty cycle of a binary waveform is defined as:

Duty Cycle $= (t_w / T) \times 100\ \%$

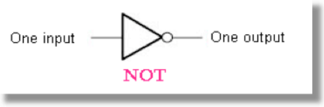
P729

A digital waveform is reality do not have infinitely fast rising and falling edges. Shown here is a real digital waveform with the most important characteristics.

Often used are: the rise time (time it takes to rise from 10% to 90% of the final value) and the fall time. In repetitive digital signals, we also are interested in its frequency, period and duty cycle. Duty cycle is the ratio between the time the signal is high to that of the period.

Basic Logic Operations

There are only three basic logic operations:

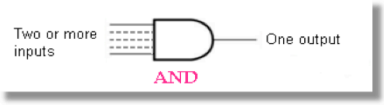


NOT gate

Input	Output
0	1
1	0

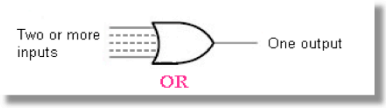
AND gate

Input A	Input B	Output
0	0	0
1	0	0
0	1	0
1	1	1



OR gate

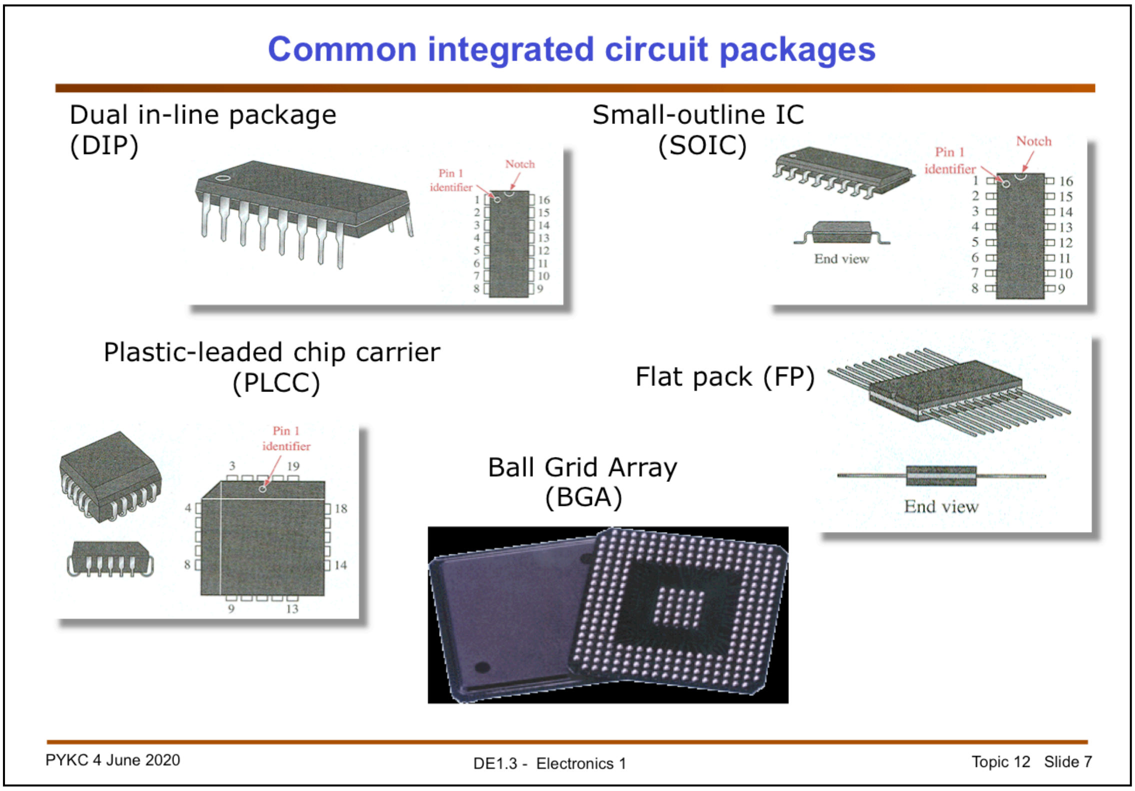
Input A	Input B	Output
0	0	0
1	0	1
0	1	1
1	1	1



P718-722

The basic logic electronic components are the NOT gate (output is the inverse of the input), AND gate (output is 1 only if ALL the inputs are 1's), and the OR gate (output is 1 if ANY of the inputs are 1).

Interestingly it can be proven that given these three gates, one could in theory build ANY digital circuits, including a Pentium CPU! We will come back to this later on the course.



All electronic integrated circuits come in different packages, particularly for digital electronics.

You will be using the Dual in-line packages in the laboratory. However, on the project, you will be using a small microcontroller board (known as the Pyboard) which comes as printed-circuit board (PCB) with other packages such as the SOIC, FP and PLCC.

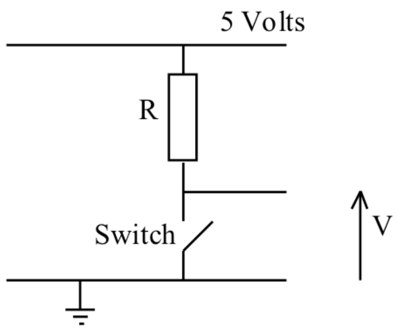
What do we mean by data?

- Many definitions are possible depending on context
- We will say that:
  - data is a physical representation of information
- Data can be stored
  - e.g. computer disk, memory chips
- Data can be transmitted
  - e.g. internet
- Data can be processed
  - e.g. inside a microprocessor

We will now consider digital signals representing “data” or information. 1’s and 0’s are meaningless unless we have a way of interpreting what they mean. For that, we need context.

Electronic Representation of Data

- Information can be very complicated
  - e.g.:
    - Numbers Sounds
    - Pictures Codes
  - We need a simple electronic representation
- What can we do with electronics?
  - Set up voltages and currents
  - Change the voltages and currents
- A useful device is a switch
  - Switch Closed: V = 0 Volts
  - Switch Open: V = 5 Volts

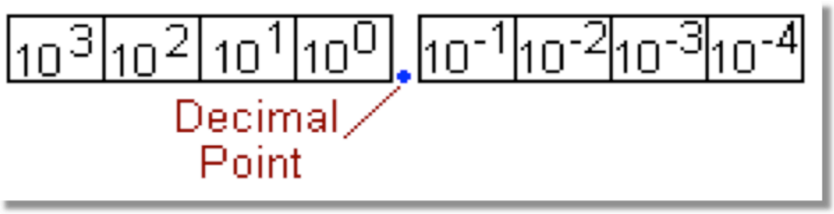


Data or information is actually quite a complex thing. For example, the current hot topic for research is called “Big Data”. This word, data, could mean many things: your name (i.e. alphabets), your age (i.e. numbers), your picture (image data), your voice (speech) or encoded signal of your speech etc.

In digital electronics, we use voltages to represent such information. The basic digital building block is a switch as implemented in a transistor. Here is a simple schematic of a switch which can be used to present a digital 1 (switch open) or digital 0 (switch closed).

Decimal Numbers

- The decimal number system has ten digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9
- The decimal numbering system has a base of 10 with each position weighted by a factor of 10:



How to use such simple circuit to represent or store information? To answer this, we need to under number system. Here is a familiar decimal number system with a decimal point in the middle. Its interpretation is straight forward.

Binary Numbers

- The binary number system has two digits: 0 and 1
- The binary numbering system has a base of 2 with each position weighted by a factor of 2:

POSITIVE POWERS OF TWO (WHOLE NUMBERS)									NEGATIVE POWERS OF TWO (FRACTIONAL NUMBER)					
2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}
256	128	64	32	16	8	4	2	1	1/2	1/4	1/8	1/16	1/32	1/64
									0.5	0.25	0.125	0.0625	0.03125	0.015625

In the binary system, we simply use binary instead of decimal weighting. Note that the binary point is also in the middle. All binary digits (bits) on the left of the binary point have weighting in increasing positive powers of 2 (i.e. 2^0, 2^1 … 2^8 etc. On the right of the binary point, the power are negative.

Binary Number System

- Uses 2 symbols by our previous rule
  - 0 and 1
- Example: 10011 in binary is

  $1 \times 2^4 + 1 \times 2^1 + 1 \times 2^0 = 19$

2^4	2^3	2^2	2^1	2^0
1	0	0	1	1

- Binary is the base 2 number system
- Most common in digital electronics

Converting a binary integer to decimal is simple but tedious. Anyone can do it.

Integer and Fractional Parts

- Binary numbers can contain fractional parts as well as integer parts

2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}
1	0	0	1	1	0	1	1

Binary Point

$(19.375)_{10}$

- This 8-bit number is in Q3 format
  - 3 bits after the binary point
- How could 19.376 best be represented using an 8-bit binary number?
  - Quantization error

Here is a fractional binary number with 3 bits AFTER the binary point. The whole number is has 8 bits. This is known as Q3 format. This binary number has a decimal value of 19.375.

If you want to represent 19.376 in the same 8-bit format, you have a problem. You can't. 19.375 is the closest you can use. This is an example why digital representation of an analogue quantity may generate quantization error.

Conversion: decimal to binary (Method 1)

- The decimal number is simply expressed as a sum of powers of 2, and then 1s and 0s are written in the appropriate bit positions.

$$
\begin{aligned}
50_{10} &= 32 + 18 \\
&= 32 + 16 + 2 \\
&= 1\times 2^5 + 1\times 2^4 + 1\times 2^1 \\
50_{10} &= 110010_2
\end{aligned}
$$

$$
\begin{aligned}
346_{10} &= 256 + 90 \\
&= 256 + 64 + 26 \\
&= 256 + 64 + 16 + 10 \\
&= 256 + 64 + 16 + 8 + 2 \\
&= 1\times 2^8 + 1\times 2^6 + 1\times 2^4 + 1\times 2^3 + 1\times 2^1 \\
346_{10} &= 101011010_2
\end{aligned}
$$

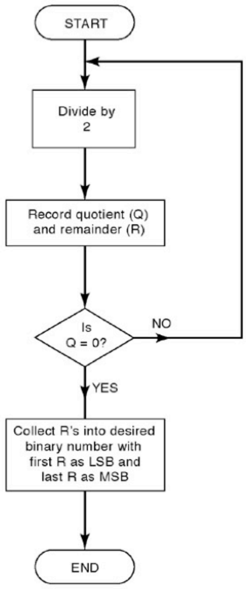
Simple – no need to explain.

Conversion: decimal to binary (method 2)

- Repeated division

		quotient	remainder	
50/2	=	25	0	LSB
25/2	=	12	1	
12/2	=	6	0	
6/2	=	3	0	
3/2	=	1	1	
1/2	=	0	1	MSB

$$50_{10}=110010_2$$



This is somewhat tedious. We will see how we can make this easier with hexadecimal (instead of binary) representation.

Conversion: binary to decimal

- The simplest way is to represent an n-bit binary number as

  $a_n \times 2^{n-1} + ... + a_2 \times 2^2 + a_1 \times 2^1 + a_0 \times 2^0$

- The conversion can be done by substituting the a's with the given bits then multiplying and adding:
  - eg: Convert $(1101)_2$ into decimal
  - $1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = (13)_{10}$

- Other algorithms can be used as alternatives if you prefer

Binary Addition

- First recall decimal addition

	1	1	1	
A	1	2	3	4
+ B		9	8	7
Sum	2	2	2	1

- In binary addition we follow the same pattern but
  - 0 + 0 = 0 carry-out 0
  - 0 + 1 = 1 carry-out 0
  - 1 + 0 = 1 carry-out 0
  - 1 + 1 = 0 carry-out 1
  - 1 + 1 + carry-in = 1 carry-out 1

		1		
A	0	1	1	1
+ B	0	1	1	0
Sum	1	1	0	1

One reason why digital circuit is useful is that it can do computation such as addition very quickly. Binary addition is similar to decimal addition with slight difference. In binary addition, the carry and the two operands (which you add) are the same – they are all BINARY. So in binary addition, you have three inputs (A, B and Carry) and two binary outputs (SUM and Carry).

- Note that we need to consider 3 inputs per bit of binary number
  - A, B and carry-in
- Each bit of binary addition generates 2 outputs
  - sum and carry-out

PYKC 4 June 2020 DE1.3 - Electronics 1 Topic 12 Slide 18

In binary addition, the carry and the two operands (which you add) are the same – they are all BINARY. So in binary addition, you have three inputs (A, B and Carry) and two binary outputs (SUM and Carry).

Hexadecimal Numbers

- Decimal, binary, and hexadecimal numbers

DECIMAL	BINARY	HEXADECIMAL
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Handling large binary number is tedious and prone to error. We therefore usually handle binary numbers in groups of four, with values going from 0 to decimal 15. We represent the values 10 to 15 in single alphabet A to F. This is the hexadecimal number system or HEX for short.

It is worthwhile to memorise that 1010_2 is decimal 10_{10} and A_{16}, and 1111_2 is decimal 15_{10} and F_{16}. For all other patterns, you just count up from A_{16} or down from F_{16}.

Hexadecimal Numbers conversions

- **Binary-to-hexadecimal conversion**
  1. Break the binary number into 4-bit groups
  2. Replace each group with the hexadecimal equivalent
- **Hexadecimal-to-decimal conversion**
  1. Convert the hexadecimal to groups of 4-bit binary
  2. Convert the binary to decimal
- **Decimal-to-hexadecimal conversion**
  - Repeated division by 16

Binary Coded Decimal (BCD)

- Use 4-bit binary to represent one decimal digit
- Easy conversion
- Wasting bits (4-bits can represent 16 different values, but only 10 values are used)
- Used extensively in financial applications

DECIMAL DIGIT	0	1	2	3	4	5	6	7	8	9
BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

This is using 4-bit unit of binary representation to represent decimal digits. Since a 4-bit binary number has a range of 0 to 15, and decimal digit only goes up to 9, we are effectively NOT using the range 1010_2 to 1111_2.

Binary Coded Decimal (BCD)

- Convert 0110100000111001(BCD) to its decimal equivalent.

  0110 1000 0011 1001

  6 8 3 9

- Convert the BCD number 011111000001 to its decimal equivalent.

  0111 1100 0001

  7 ↑ 1

  The forbidden code group indicated an error

Summary – binary, hexadecimal and BCD

Decimal	Binary	Octal	Hexadecimal	BCD
0	0	0	0	0000
1	1	1	1	0001
2	10	2	2	0010
3	11	3	3	0011
4	100	4	4	0100
5	101	5	5	0101
6	110	6	6	0110
7	111	7	7	0111
8	1000	10	8	1000
9	1001	11	9	1001
10	1010	12	A	0001 0000
11	1011	13	B	0001 0001
12	1100	14	C	0001 0010
13	1101	15	D	0001 0011
14	1110	16	E	0001 0100
15	1111	17	F	0001 0101

P723

ASCII code

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

PYKC 4 June 2020 DE1.3 - Electronics 1 Topic 12 Slide 24

Codes representing letters of the alphabet, punctuation marks, and other special characters as well as numbers are called *alphanumeric* codes.

The most widely used alphanumeric code is the American Standard Code for Information Interchange(**ASCII**). The ASCII (pronounced “askee”) code is a seven-bit code.