

## **The Software Developer's View of the Hardware**

- Representations of data within the computer
  - Character Representation
  - Integer Representation
  - Representation of fractions
  - Binary arithmetic
- Electronic Circuits to perform standard Software Operations
  - Logic Gates
  - Truth Tables
  - Circuit Design Steps
  - Special Circuits
- Programming of Hardware Devices
  - Input Data
  - Processing of data
  - Output Data
  - Control Systems
  - Printer Operation
  - Specialist Devices – Input and Output

### **Representation of Data within the Computer**

- All data within a computer system is stored and processed in binary.
- Decimal numbers are usually stored in the computer using a direct conversion to binary.
- BCD – Binary Coded Decimal, each decimal digit is allocated 4 bits. This allows for rapid conversion between binary and decimal, but a greater amount of memory is used.

Limitations of representation of data when programming:

- The range of acceptable integers 16 or 32 bits.
- The maximum size of a real number. How many decimal places may be used.
- The disk space that can be used.
- The amount of RAM available for data storage.

### **Characters**

- Characters are letters of the alphabet, digits and other symbols from the keyboard. Characters have to be changed into binary numbers in order to be stored and processed by the computer.
- It is much simpler to group all the characters together to form a string when representing a piece of data. The advantage of using a string over using the individual characters is that the data can be referred to by a single name and manipulated as a group.
- ASCII – American Standard Code for Information Interchange, stores characters using 7 bits – 128 different characters.
- EBCDIC – Extended Binary Coded Decimal Interchange Code, uses 8 bits – 256 different characters.
- UNICODE – 16 bit ASCII.
- Hexadecimal numbers are base 16.

## Decimal to Binary

$287_{10} = 100011111_2$								
287				31	15	7	3	1
1	0	0	0	1	1	1	1	1
256	128	64	32	16	8	4	2	1
$2^8$	$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$
31				15	7	3	1	0

$24.125_{10} = 11000.001_2$								
$24\frac{1}{8}$	$8\frac{1}{8}$						$\frac{1}{8}$	
1	1	0	0	0	0	0	1	
16	8	4	2	1	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	
$2^4$	$2^3$	$2^2$	$2^1$	$2^0$	$2^{-1}$	$2^{-2}$	$2^{-3}$	
$8\frac{1}{8}$	$\frac{1}{8}$						0	

$19.5_{10} = 10011.1_2$					
$19\frac{1}{2}$			$3\frac{1}{2}$	$1\frac{1}{2}$	$\frac{1}{2}$
1	0	0	1	1	1
16	8	4	2	1	$\frac{1}{2}$
$2^4$	$2^3$	$2^2$	$2^1$	$2^0$	$2^{-1}$
$3\frac{1}{2}$			$1\frac{1}{2}$	$\frac{1}{2}$	0

$-23_{10} = -10111_2$					
23		7	3	1	For basic negative binary numbers, just put a (–) sign.
1	0	1	1	1	
16	8	4	2	1	
$2^4$	$2^3$	$2^2$	$2^1$	$2^0$	
7		3	1	0	

- Find the highest power of 2 that will go into the number.
- Subtract and keep repeating till you get no remainder or till you reach  $2^0$ .
- Where the number can be subtracted, is a 1. If the number cannot be subtracted, it is a 0.
- For decimal numbers, place a decimal point, and continue the process with negative powers of 2.
- Reading the tables:
  1. Number
  2. 1 if can be subtracted, 0 if cannot be subtracted
  3. Number to be subtracted
  4. Number to be subtracted represented as a power of 2
  5. Remainder

## Decimal to Hexadecimal

$287_{10} = 11F_{16}$		
287	31	15
1	1	F
256	16	1
$16^2$	$16^1$	$16^0$
31	15	0

$24.125_{10} = 18.2_{16}$		
$24\frac{1}{8}$	$8\frac{1}{8}$	$\frac{1}{8}$
1	1	2
16	1	$\frac{1}{16}$
$16^1$	$16^0$	$16^{-1}$
$8\frac{1}{8}$	$\frac{1}{8}$	0

$19.5_{10} = 13.8_{16}$		
$19\frac{1}{2}$	$3\frac{1}{2}$	$\frac{1}{2}$
1	3	8
16	1	$\frac{1}{16}$
$16^1$	$16^0$	$16^{-1}$
$3\frac{1}{2}$	$\frac{1}{2}$	0

$-23_{10} = -17_{16}$	
23	7
1	7
16	1
$16^1$	$16^0$
7	0

DECIMAL	HEX
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F

- Find the highest power of 16 that will go into the number.
- Subtract and keep repeating till you get no remainder or till you reach  $16^0$ .
- When the number is subtracted, it is either a 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.
- For decimal numbers, place a decimal point, and continue the process with negative powers of 16.
- Reading the tables:
  1. Number
  2. 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
  3. Number to be subtracted
  4. Number to be subtracted represented as a power of 16
  5. Remainder

## Decimal with Recurrence

Change  $-7.8_{10}$  to Binary and Hexadecimal

$-7.8_{10} = -111.\dot{1}10\dot{0}_2$						
$7\frac{4}{5}$	$3\frac{4}{5}$	$1\frac{4}{5}$		$\frac{4}{5}$	$\frac{3}{10}$	$\frac{1}{20}$
1	1	1		1	1	0
4	2	1		$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$
$2^2$	$2^1$	$2^0$		$2^{-1}$	$2^{-2}$	$2^{-3}$
$3\frac{4}{5}$	$1\frac{4}{5}$	$\frac{4}{5}$		$\frac{3}{5}$	$\frac{1}{20}$	$\frac{1}{20}$

The decimal part keeps repeating itself after 0.8

$$\therefore -7.8_{10} = -111.\dot{1}10\dot{0}_2$$

	0.8	×
	2	
1	1.6	-1
	0.6	×
	2	
1	1.2	-1
	0.2	×
	2	
0	0.4	-0
	0.4	×
	2	
0	0.8	-0
	0.8	×
	2	
	1.6	

$-7.8_{10} = -7.\dot{C}_{16}$		
$7\frac{4}{5}$	$\frac{4}{5}$	$\frac{4}{5}$
7	C	C
1	$\frac{1}{16}$	$\frac{1}{256}$
$16^0$	$16^{-1}$	$16^{-2}$
$\frac{4}{5}$	$\frac{4}{5}$	

	0.8	×
	16	
C	12.8	-12
	0.8	×
	16	
C	12.8	

The decimal part keeps repeating itself after 0.8

## Binary to Decimal

$100011111_2 = 287_{10}$								
1	0	0	0	1	1	1	1	1
256	128	64	32	16	8	4	2	1
$2^8$	$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$

$$256 + 16 + 8 + 4 + 2 + 1 = 287$$

$11000.001_2 = 24.125_{10}$							
1	1	0	0	0	0	0	1
16	8	4	2	1	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$
$2^4$	$2^3$	$2^2$	$2^1$	$2^0$	$2^{-1}$	$2^{-2}$	$2^{-3}$

$$16 + 8 + \frac{1}{8} = 24.125$$

$10011.1_2 = 19.5_{10}$					
1	0	0	1	1	1
16	8	4	2	1	$\frac{1}{2}$
$2^4$	$2^3$	$2^2$	$2^1$	$2^0$	$2^{-1}$

$$16 + 2 + 1 + \frac{1}{2} = 19.5$$

$-10111_2 = -23_{10}$				
1	0	1	1	1
16	8	4	2	1
$2^4$	$2^3$	$2^2$	$2^1$	$2^0$

$$16 + 4 + 2 + 1 = 23$$

But because it is negative,  $-23$

- Line up the binary digits with their ‘place value’
- Where there is a 1, add the place value.
- The total will give you the decimal result
- Reading the tables:
  1. Binary digit
  2. ‘place value’
  3. place value represented as a power of 2

### Integer Representation

Integers are positive or negative whole numbers

Negative numbers can be represented using:

- Sign/ Modulus – the left-most bit represents the sign. 0 positive, 1 negative.
- One’s Complement – Shows negative numbers by replacing all the ones in the positive form of the number with zeros and all the zeros in the positive form of the number with ones.
- Two’s Complement – Shows negative numbers by adding 1 to the one’s complement.

### Fractions

Binary fractions are represented as negative powers.

### Comparison of integer representations

Bit Pattern	Bit pattern interpreted as :			
	Unsigned	2’s comp.	1’s comp.	Sign/ Modulus
0000	0	0	0	0
0001	1	1	1	1
0010	2	2	2	2
0011	3	3	3	3
0100	4	4	4	4
0101	5	5	5	5
0110	6	6	6	6
0111	7	7	7	7
1000	8	-8	-7	-0
1001	9	-7	-6	-1
1010	10	-6	-5	-2
1011	11	-5	-4	-3
1100	12	-4	-3	-4
1101	13	-3	-2	-5
1110	14	-2	-1	-6
1111	15	-1	-0	-7

### Floating Point or Real Numbers

Real numbers are represented as floating point numbers made up of a Sign bit, Mantissa, and Exponent – IEEE 754.

Many decimal fractions cannot be represented as exact binary numbers. When the number of digits in a number exceeds the space allocated for storage, it must be truncated or cut to fit the space available.

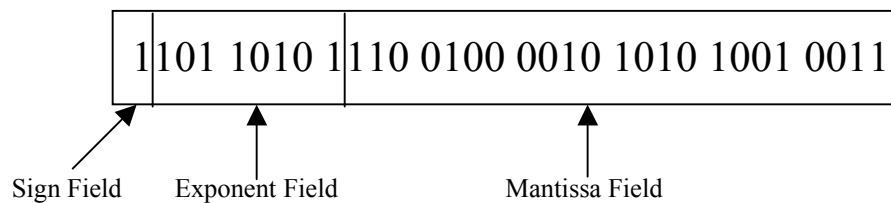
### Decimal to Floating Point IEEE 754

A floating point number is represented by 3 parts. Sign, Exponent, Mantissa. Uses (32) bits.

Sign (1) Describes whether the number is positive or negative.  
0 is positive; 1 is negative.

Exponent (8) Describes the notation of the number.  
Exponent = power of 2 + 127

Mantissa (23) normalized to the range  $1 \leq x < 2$   
Mantissa or Significand





**$735.87_{10} = 0100\ 0100\ 0011\ 0111\ 1111\ 0111\ 1010\ 1110$**

➤ Break up the base<sub>10</sub> number in integers and decimals

$735_{10} = 1011011111_2$									
735		223		95		31		15	
1	0	1	1	0	1	1	1	1	1
512	256	128	64	32	16	8	4	2	1
$2^9$	$2^8$	$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$
223		95		31		15		7	

**0.87**

### THE DECIMAL PART

- With each step, multiply by 2.
- If the result is greater than 1, subtract 1 and continue to multiply by 2 again.
- Keep doing this process until you have 23 bits for the mantissa.
- The results greater than 1 represent a 1 bit
- The results lesser than 1 represent a 0 bit
- DO NOT ADD THE LEAD 0 (eg. 0.87) IN THE MANTISSA

### CALCULATING THE EXPONENT (Normalizing)

Because the integer part is greater than 1, we must shift the decimal point (.) to the left.

$1011011111.$  becomes  $1.011011111 \times 2^9$

↖  
9 places

$\therefore \text{Exponent} = 9 + 127 = 136$

$136 = 1000\ 1000$

### THE MANTISSA

The mantissa is all the binary digits after the normalized decimal point

Sign            0  
Exponent      100 0100 0  
Mantissa      011 0111 1111 0111 1010 1110

$735.87_{10} = 0100\ 0100\ 0011\ 0111\ 1111\ 0111\ 1010\ 1110$  in IEEE 754 Floating Point

**$-0.00064_{10} = 1011\ 1010\ 0010\ 0111\ 1100\ 0101\ 1010\ 1100$**

➤ There is no integer to break up

**0.00064**

0.00128

0.00256

0.00512

0.01024

0.02048

0.04096

0.08192

0.16384

0.32768

0.65536

1.31072

0.62144

1.24288

0.48576

0.97152

1.94304

1.88608

1.77216

1.54432

1.08864

0.17728

0.35456

0.70912

1.41824

0.83648

1.67296

1.38368

0.76736

1.53472

1.06944

0.13888

0.27776

0.55552

### THE DECIMAL PART

- With each step, multiply by 2.
- If the result is greater than 1, subtract 1 and continue to multiply by 2 again.
- Keep doing this process until you have 23 bits for the mantissa.
- The results greater than 1 represent a 1 bit
- The results lesser than 1 represent a 0 bit
- DO NOT ADD THE LEAD 0 (eg. 0.87) IN THE MANTISSA

### CALCULATING THE EXPONENT (Normalizing)

Because there is no integer part, we must shift the decimal point (.) to the right.

So that the number is  $1 \leq x < 2$ , we shift it right 11 places  $2^{-11}$

$$\therefore \text{Exponent} = -11 + 127 = 116$$

$$116 = 0111\ 0100$$

### THE MANTISSA

The mantissa is all the binary digits after the normalized decimal point.

Sign	1
Exponent	011 1010 0
Mantissa	010 0111 1100 0101 1010 1100

$-0.00064_{10} = 1011\ 1010\ 0010\ 0111\ 1100\ 0101\ 1010\ 1100$  in IEEE 754 Floating Point

## Binary Arithmetic

## Binary Addition

1	1	0	0	
1	0	1	0	+
10	1	1	0	

Addition Table

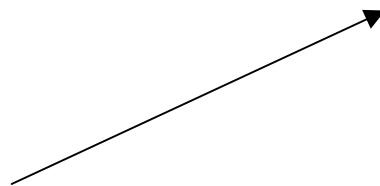
1000111	
1010111	+
10011110	

## Binary Multiplication

1	1	0	0	
1	0	1	0	x
1	0	0	0	

Multiplication Table

1001	
1100	x
Like normal multiplication, the 0's can be added on later	



1001	
11	x
1001	
1001	+
1101100	

## Binary Division

Unfortunately, you need to use long division here....

$$\begin{array}{r}
 10111 \\
 1100 \overline{)100010100} \\
 \underline{1100 \phantom{000000}} \\
 10101 \\
 \underline{1100 \phantom{000000}} \\
 10010 \\
 \underline{1100 \phantom{000000}} \\
 1100 \\
 \underline{1100 \phantom{000000}} \\
 0
 \end{array}$$

Grey numbers represent numbers ‘brought down’

$$\begin{array}{r}
 100.1 \\
 1000 \overline{)100100.0} \\
 \underline{1000 \phantom{000000}} \\
 100.0 \\
 \underline{100.0 \phantom{000000}} \\
 0
 \end{array}$$

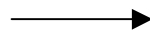
$$\begin{array}{r}
 1101 \\
 110 \overline{)1^{-1}0^{-1}01110} \\
 \underline{1 \phantom{0}10 \phantom{00}} \\
 111 \\
 \underline{110 \phantom{0000}} \\
 110 \\
 \underline{110 \phantom{0000}} \\
 0
 \end{array}$$

## Binary Subtraction

(1) 0	1	1	0	
1	1	0	0	– Subtraction Table
1	0	1	0	

1010111	
1000111	–
10000	

1001100	
1010101	–
– ?	



If the top number is smaller than the bottom, we know the answer will be negative. Switch the numbers around and subtract.

1010101	
1001100	–
1001	

But because we switched them around,  $1001100 - 1010101 = -1001$

.....

One’s and Two’s Complements are a different way of subtraction binary numbers. The basic concepts:

### One’s Complements

- Bottom number is reversed. 0’s become 1’s; 1’s become 0’s. Remember: spaces are counted as 0’s.
- Add top and reversed bottom number together.
- Add one bit to the result.
- The leading 1 is removed.

### Two’s Complements

- Change the bottom number. From the right-hand side, all the bits stay the same until the first 1. After the first 1, all the bits are reversed. 0’s become 1’s; 1’s become 0’s. Remember: spaces are counted as 0’s.
- Add top and changed bottom number together.
- The leading 1 is removed.

One's Complements

- Bottom number is reversed. 0's become 1's; 1's become 0's. Remember: spaces are counted as 0's.
- Add top and reversed bottom number together.
- Add one bit to the result
- The leading 1 is removed.

$$\begin{array}{r}
 1000111 \\
 0110111 \quad - \\
 \hline
 \end{array}$$

Change the bottom number

$$\begin{array}{r}
 1000111 \\
 1001000 \quad + \\
 \hline
 10001111 \\
 1 \quad + \\
 \hline
 1\ 0010000
 \end{array}$$

- Add top and reversed bottom number together.
- Add one bit to the result
- The leading 1 is removed.

$$\therefore 1000111 - 110111 = 10000$$

$$\begin{array}{r}
 1101010 \\
 1100001 \quad - \\
 \hline
 \end{array}$$

Change the bottom number

$$\begin{array}{r}
 1101010 \\
 0011110 \quad + \\
 \hline
 10001000 \\
 1 \quad + \\
 \hline
 1\ 0001001
 \end{array}$$

- Add top and reversed bottom number together.
- Add one bit to the result
- The leading 1 is removed.

$$\therefore 1101010 - 1100001 = 1001$$

Two's Complements

- Change the bottom number. From the right-hand side, all the bits stay the same until the first 1. After the first 1, all the bits are reversed. 0's become 1's; 1's become 0's. Remember: spaces are counted as 0's.
- Add top and changed bottom number together.
- The leading 1 is removed.

**OR**

- Change the bottom number. All numbers are reversed. 0's become 1's; 1's become 0's. Remember: spaces are counted as 0's. Add 1 bit.
- Add top and changed bottom number together.
- The leading 1 is removed.

$$\begin{array}{r}
 1000100 \\
 0111011 \quad - \\
 \hline
 \end{array}$$

Change the bottom number

$$\begin{array}{r}
 1000100 \\
 1000101 \quad + \\
 \hline
 1\ 0001001
 \end{array}$$

- Add top and reversed bottom number together.
- The leading 1 is removed.

$$\therefore 1000100 - 111011 = 1001$$

$$\begin{array}{r}
 1110001 \\
 0001111 \quad - \\
 \hline
 \end{array}$$

Change the bottom number

$$\begin{array}{r}
 1110001 \\
 1110001 \quad + \\
 \hline
 1\ 1100010
 \end{array}$$

- Add top and reversed bottom number together.
- The leading 1 is removed.




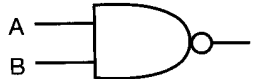


$$\therefore 1110001 - 1111 = 1100010$$

## Electronic Circuits to perform Standard Software Operations

### Logic Gates

Logic gates are hardware circuits that produce a 1 or 0 output signal if the input requirements are satisfied.

Integrated circuits are small silicon semiconductors called chips, containing electrical components such as transistors, diodes, resistors and capacitors.

Name	Description	Logic gate	Truth table		
<b>AND</b>	The output value of C is 1 when both input values A and B are 1.		Input		Output
			A	B	$C = AB$
			0	0	0
			0	1	0
			1	0	0
			1	1	1
<b>OR</b>	If either of the input values are 1 then the output will be 1.		Input		Output
			A	B	$C = A+B$
			0	0	0
			0	1	1
			1	0	1
			1	1	1
<b>NOT</b>	The state of the output is the inverse of the input.		Input		Output
			A		$C = \bar{A}$
			0		1
<b>NAND</b>	Not AND. The output value C is 1 except when both A and B are 1.		Input		Output
			A	B	$C = (\overline{AB})$
			0	0	1
			0	1	1
			1	0	1
			1	1	0
<b>NOR</b>	The output C is 1 when both inputs A and B are 0.		Input		Output
			A	B	$C = \overline{(A+B)}$
			0	0	1
			0	1	0
			1	0	0
			1	1	0
<b>XOR</b>	The output C is 1 except when both inputs A and B are the same.		Input		Output
			A	B	$C = A \oplus B$
			0	0	0
			0	1	1
			1	0	1
			1	1	0



### Truth Tables

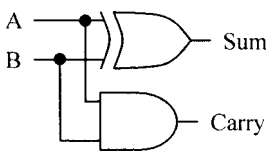
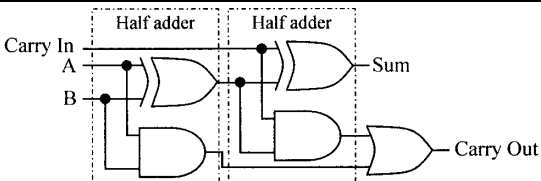
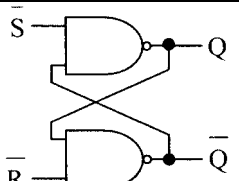
A truth table is a compact way of showing the possible outputs from all possible variations of inputs into a logic gate.

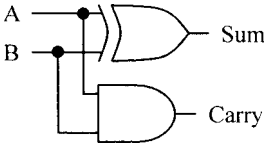
### Circuit Design Steps

1. *Identify inputs and outputs:* The number of inputs into the circuit must be determined, as well as the number of outputs.
2. *Identify required components:* The circuit designer must then determine which logic gates will be needed.
3. *Check solution with a truth table:* Once a possible solution has been developed, it must be tested to ensure it works as required.
4. *Evaluate the circuit design:* Using logic gates, there is often more than one way to combine a number of gates to produce a combinational circuit.

### Specialty Circuits

- All circuits within the computer are made from a combination of basic logic gates.
- As the number of inputs increase, the circuits become more complex.
- Combinational circuits produce instant output, determined by the combination of logic gates.
- Sequential circuits contain memory cells as well as logic gates.

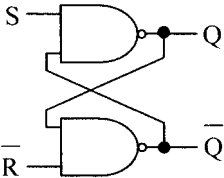
Half Adders	A half adder is a combinational circuit that perform the addition of two bits. AND and XOR gates	
Full Adders	A full adder is a combinational circuit that can be used to add three binary digits. It consists of three inputs and two outputs. A full adder can be implemented by combining two half adders.	
Flip-Flops (RS)	A flip-flop is a circuit that can store a binary value as long as power is supplied to the circuit. Flip-flops are the basic building blocks of computer memory. Two NOR gates If either of the NOR inputs are 1, then the outputs has to be 0 regardless of the other input. When implemented in the computer, both inputs remain at 0 unless the state of the flip-flop has to be changed.	
 Half-Adder	 Full-Adder	 RS Flip-Flop

Half Adder				
Input		Output		
X	Y	Carry	Sum	
0	0	0	0	
0	1	0	1	
1	0	0	1	
1	1	1	0	

Full Adder				
Input			Output	
X	Y	Z	Carry	Sum
0	0	0	0	0
0	1	0	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

The logic diagram illustrates a Full Adder circuit. It features two dashed boxes, each labeled 'Half adder'. The first Half adder takes inputs A and B. Its outputs are a Sum (XOR of A and B) and a Carry (AND of A and B). The second Half adder takes the Sum from the first Half adder and a 'Carry In' as inputs. Its outputs are the final Sum (XOR of the first Sum and Carry In) and a 'Carry Out' (OR of the first Carry and the second Carry). The final Sum and Carry Out are labeled on the right side of the diagram.

RS Flip-Flop			
R	S	Q	Q'
0	1	1	0
1	0	0	1
0	0	Prohibited	Prohibited
1	1	No Change	No Change



## Programming of Hardware Devices

The way that the data is structured varies with different communications protocols.

### Input Data

- Data read into the computer is accessed as a sequence of zeros and ones.
- A protocol must be established before two hardware devices can exchange data.
- Data streams are composed of a header, data characters and a trailer.
  - The header contains a set sequence of bits to indicate the start of a block of data. Headers often contain error-checking data such as parity or CRC characters. A header will also contain information that specifies the device from where the data is coming.
  - The body of data will contain the instructions that are to be processed by the CPU. Control transfers enable the system software to configure devices when they are first attached. Bulk data consists of large amounts of sequential data. Interrupt data consists of event notification, characters or coordinates such as input from the mouse or keyboard. Isochronous data is continuous real-time data.
  - The trailer contains data bits to indicate the end of a block of data. The use of a trailer and a header help the CPU to manage the data that it receives.
- Control characters permit the checking and correct reassembling of a message.
- The input data stream will be structured according to the standards of protocol followed by the hardware developer.
- Driver or extension software such as DLL (Dynamic Link Libraries) may be required to enable an operating system to communicate with a particular hardware device.
- Technical documentation accompanying hardware devices should specify the communications protocols used for that device and the format of data.

### Processing of Data

- The protocol used for data transfer will determine how control characters are recognized and stripped from the body of data.
- Header files may contain information concerning how much data is being transmitted.
- The hardware device connected to the CPU is responsible for extracting and using data sent to it from the computer.

### Generating Output

- The output data stream functions in a similar way to the input data stream.
- Output data packets must specify the device to which the information is being transmitted. The header may also contain data to indicate the start of a block of data and a description of the type of data being transmitted.

### Control Systems

- A computer-controlled system is a combination of hardware and software designed to instruct the computer to control a connected device.
- Sensor – Sensors are used to capture information from the environment.
- Process Controller – Instructions that control the actuators and effectors and allow the controller to interpret the input from the sensors in an appropriate way. The process controller may be a computer or a specially designed circuit built into an appliance to allow stand-alone operation.
- Effectors and Actuator – Effectors and Actuators perform the actions of modifying the environment. The processor sends output signals to the actuators, and these actuators perform a task.
- Analog signals are those which are continuously variable with the possibility of the signal having any value within a particular range.
- Digital signals have discrete levels only and because of this they are easily converted to a number or digit. Data must be converted into a digital signal to be used by the computer.
- An Analog to Digital Converter (ADC) is used to convert analog to digital.
- Open-loop systems do not respond to information provided by sensors.
- Closed-loop systems respond to information provided by sensors.
- The computer software, or embedded chip in an appliance, will determine the action to be taken. The code is recursive, so will continue indefinitely.

### Printer Operation

- Communication with printers require the use of specialized control characters to generate the correct responses from the printer. Many printers also use customized control characters to indicate formatting changes such as page throw, font changes and line spacing.

### Special Devices

- Digital transfer allows data to be accurately transmitted without conversion losses created by analog to digital and digital to analog converters.
- FireWire protocol allows data to be transmitted directly from digital source to the computer. A perfect digital copy is made with no conversion losses.