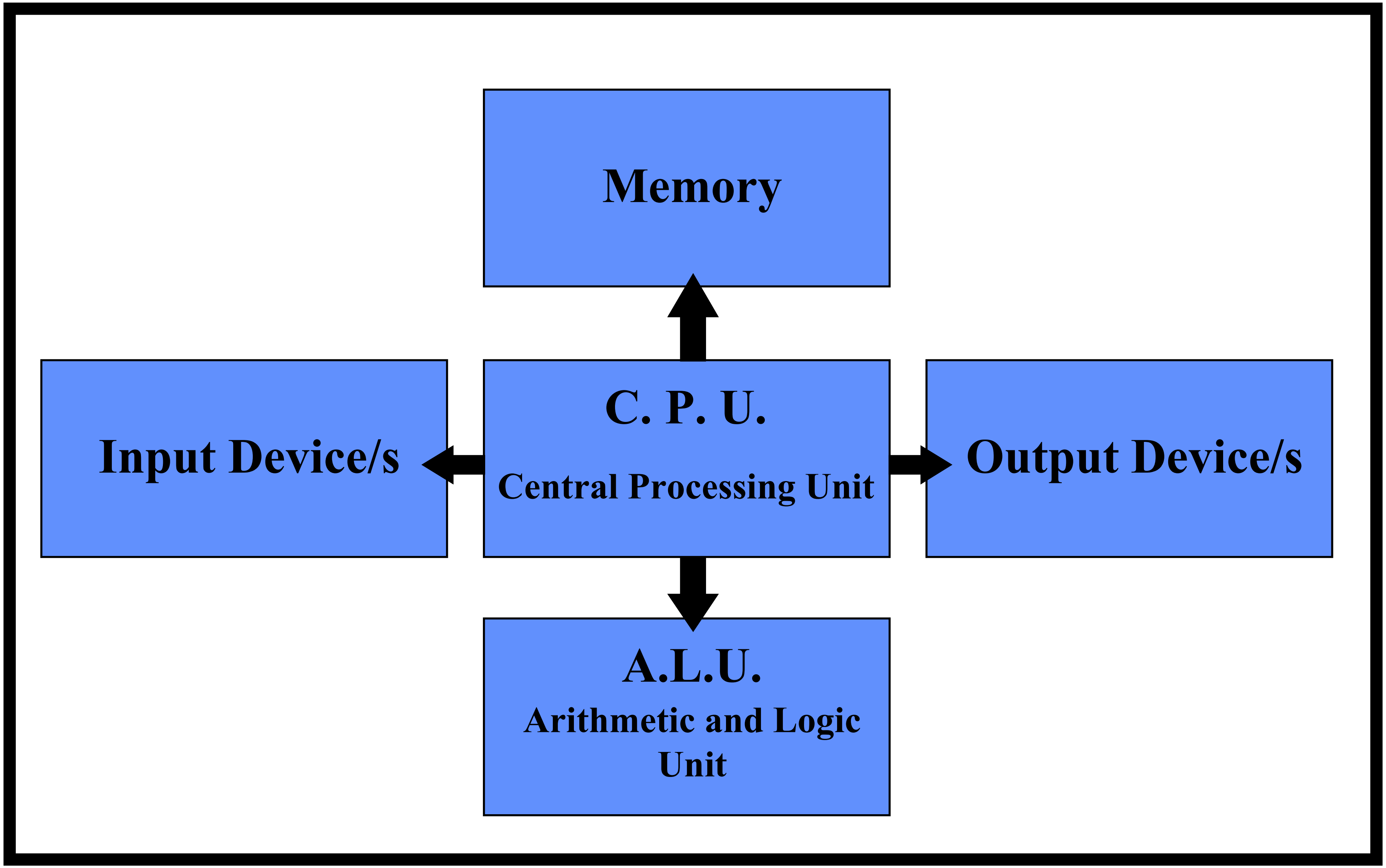
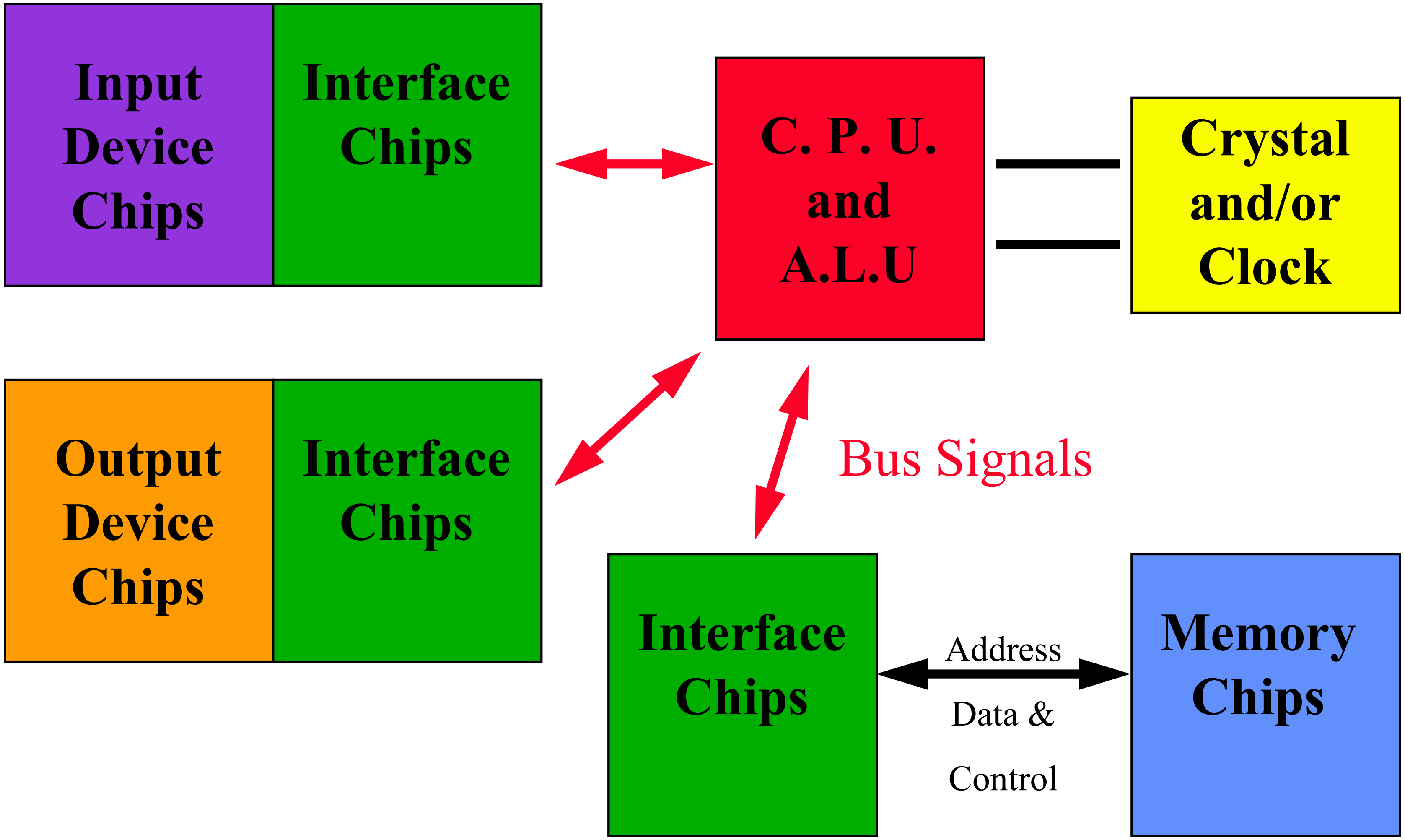
Microprocessors and Microcontrollers

What is a Computer is made from ?



The Microprocessor

Block diagram of a Microprocessor System.



Input Device Chips

Interface Chips

C. P. U. and A.L.U

Crystal and/or Clock

Output Device Chips

Interface Chips

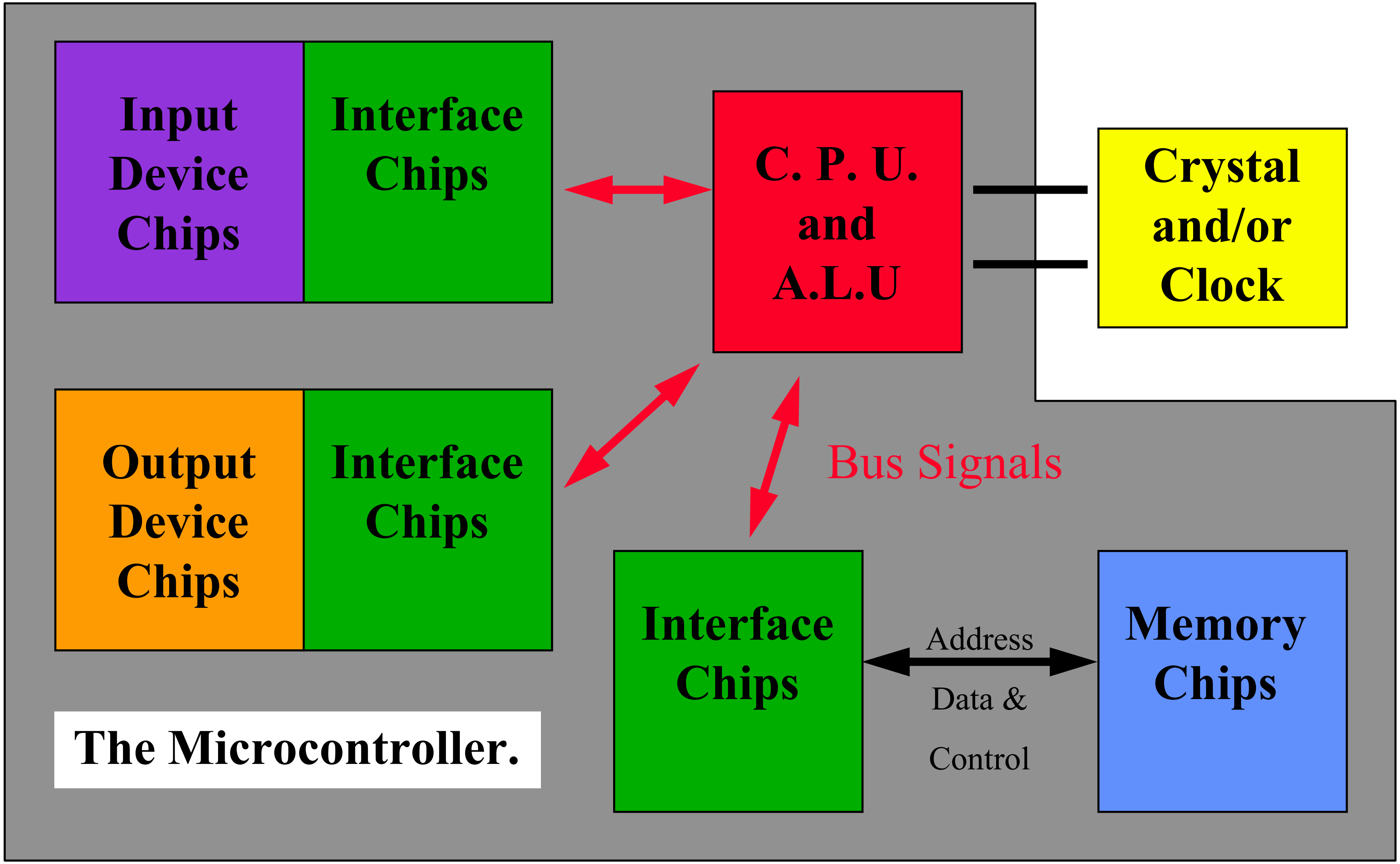
Bus Signals

Interface Chips

Address Data & Control

Memory Chips

Block diagram of a Microcontroller.

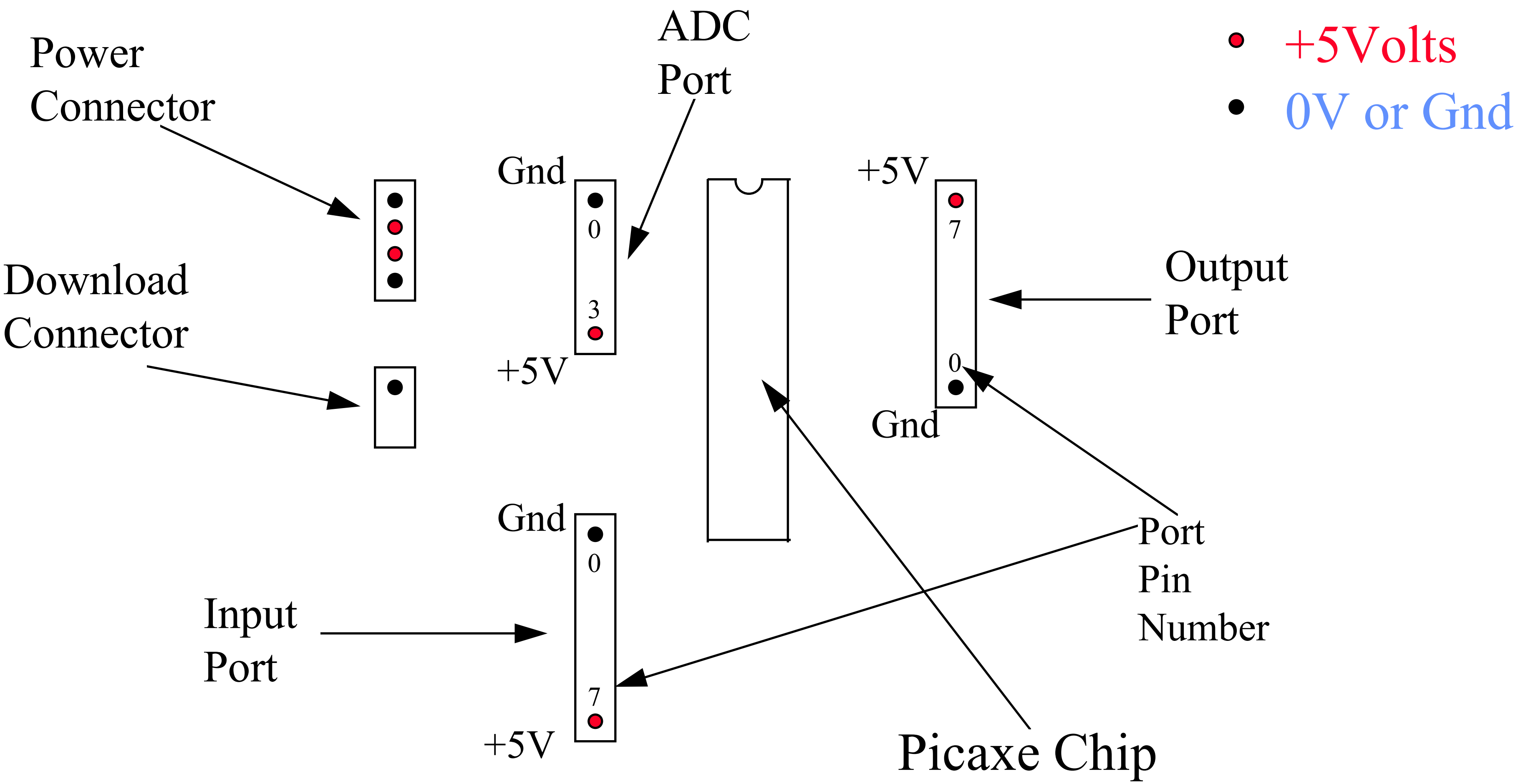


PICAXE

Picaxe.

- What are the Picaxe interfaces ?
- Eight default Output Port Lines.
- Eight default Input Port Lines.
- Four Analogue to Digital conversion Input channels.
- The Input and Output ports can be reconfigured if required.

Using the Picaxe PCB Ports.



Power Connector

Download Connector

ADC Port

Gnd

0

3

+5V

+5V

7

0

Gnd

Output Port

Gnd

0

7

+5V

Input Port

Port Pin Number

Picaxe Chip

- ● +5Volts
- ● 0V or Gnd

What does this code do ?

Programme code evaluation.

What does this code do ?.

```
Rem Set B0 to some value e.g. value 3
Rem Set B1 to some value e.g. value 27
                B0 = 3
                B1 = 27

                B2 = B0
                B0 = B1
                B1 =B2

Rem What value does B0 now contain.
Rem What value does B1 now contain.
```

What does this code do ?.

```
Rem Set B0 to some value e.g. value 3
Rem Set B1 to some value e.g. value 27
                B0 = 3
                B1 = 27

                B0 = B0 +B1
                B1 = B0 - B1
                B0 = B0 - B1

Rem What value does B0 now contain.
Rem What value does B1 now contain.
```

What does this code do ?.

```
Rem Set B0 to some value e.g. value 3
Rem Set B1 to some value e.g. value 27
                B0 = 3
                B1 = 27

                B0 = B0 ^ B1
                B1 = B0 ^ B1
                B0 = B0 ^ B1

Rem What value does B0 now contain.
Rem What value does B1 now contain.
```

What does this code do ?.

```
Rem Set B0 to some value e.g. value 3
                B0 = 3
                B0 =B0 + B0
                B1 = B0
                B0 = B0 + B0
                B0 = B0 + B0
                B0 = B0 + B1

Rem What value does B0 now contain.
Rem What other way could you implement this code.
```

What does this code do ?.

```
Rem Set B0 to some pattern e.g. value 3
                B0 = 3
Hell:           B3=0
                B2 = B0 & 128
                IF B2=0 THEN Pass
                B3 = 1
Pass:           B0 = B0 + B0 + B3
                PINS = B0
                PAUSE 500
                GOTO       Hell

Rem Process about six cycles of this code.
Rem Showing register contents for each cycle.
```

The Programming Activity.

Report.

What goes in the Report.

- Take the general report template and modify and customise the report to your product.
- Create a new section called "**Research**"
- In the research section insert your finding from your research or experimental activities.
- Remember to relate your experimental work to the five block computer model.
- When you start the main design activity complete the appropriate sections and delete any unwanted headers and contents.

What goes in the Report.

- The Section header "**Research**"
- In this section place a diagram of the 5 Block model and photograph of prototype board.
- Describe the actions of programmes and how they relate to the 5 Block model.
- A sub-section describing
  - programming of Output devices or ports.
- A sub-section describing
  - programming of Input devices or ports.
- A sub-section describing
  - programming of ADC devices or ports.

What goes in the Report.

Introduction.

Specification.

Research.

Design.

Testing.

Conclusion.

Appendix. (Optional)

Bibliography. (Optional)

Report.

A formal document that identifies the work you have completed in standard format or style.

Main sections you will need for your report.

What goes in the Report.

Introduction.

Specification.

Research.

Design.

Testing.

Conclusion.

Appendix.	(Optional)
Bibliography.	(Optional)

Introduction,

What you trying to produce.

General details of expected outcomes or similar information.

Subsection

User and Technical Guide information. Operation and use.

What goes in the Report.

Introduction.

Specification.

Research.

Design.

Testing.

Conclusion.

Appendix.	(Optional)
Bibliography.	(Optional)

Specification

User / Customer Requirement.

A general specification that will meet the needs of the User / Customer Requirement.

Technical or detailed specification that formalizes the parameters of the general specification.

What goes in the Report.

Introduction.

Specification.

Research.

Design.

Testing.

Conclusion.

Appendix.	(Optional)
Bibliography.	(Optional)

Research

Research work you completed to be able to develop the final project. for example :-

Five block model

Experimental work

Sections on

- Output Ports
- Input Ports
- ADC Ports

What goes in the Report.

Introduction.

Specification.

Research.

Design.

Testing.

Conclusion.

Appendix. (Optional)

Bibliography. (Optional)

Design

How you have implemented the specification.

Will also include :-

- Descriptions.
- Flowcharts and or state diagrams.
- Programme code with comments.
- Data descriptions (usage).
- Test Rigs required etc.

What goes in the Report.

Introduction.

Specification.

Research.

Design.

Testing.

Conclusion.

Appendix. (Optional)

Bibliography. (Optional)

Testing

What tests you have used to verify that the product meets the specification.

Your tests should verify that you get a correct response for a correct input but also a correct response when an incorrect input is applied.

Positive and negative test scripts should be produced.

Test against the specification.

What goes in the Report.

Introduction.

Specification.

Research.

Design.

Testing.

Conclusion.

Appendix.	(Optional)
Bibliography.	(Optional)

Conclusion

What was the outcome of your work or activity, include both positives and negatives.

What would you change if you were to do the activity again?

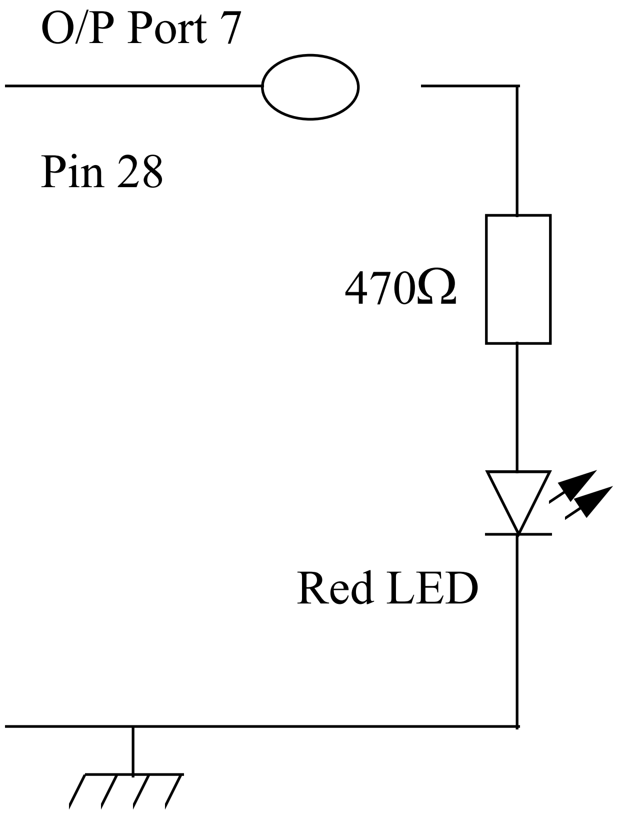
What have you learnt or gained from the experience?

How might you improve what you have done?

Programming Activity 1.

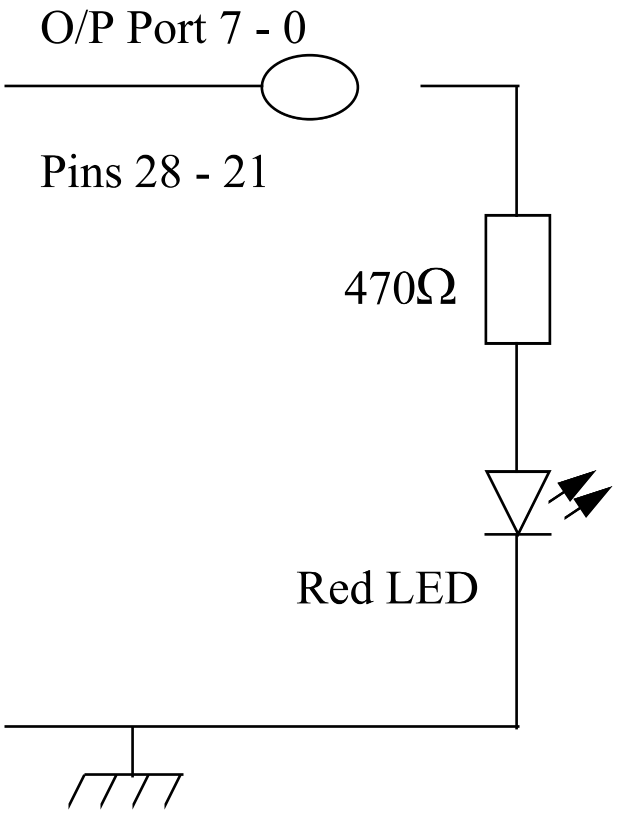
Using The Output Ports.

Using the Output Ports.



```
Rem The Programme Name Author/Date
Rem What the programme does .... (Describe)
‘Label    Code              Comment
Main:
          High 7            ‘Switch LED ON
          Pause 1000        ‘Wait a while
          Low 7             ‘Switch LED OFF
          Pause 1000        ‘Wait a while
          Goto Main         ‘Loop forever
```

Using the Output Ports.



```
Rem The Programme Name Author/Date
Rem What the programme does .... (Describe)
‘Label   Code              Comment
Rem Note Pins on LHS = all of the O/P Port and
Rem B0 is one of the internal registers.
         B0 = 0            ‘Initialize Counter
Main:
         Pins = B0         ‘Switch LEDs
         Pause 1000        ‘Wait a second
         B0 = B0 +1        ‘Change count pattern
         Goto Main         ‘Loop forever
```

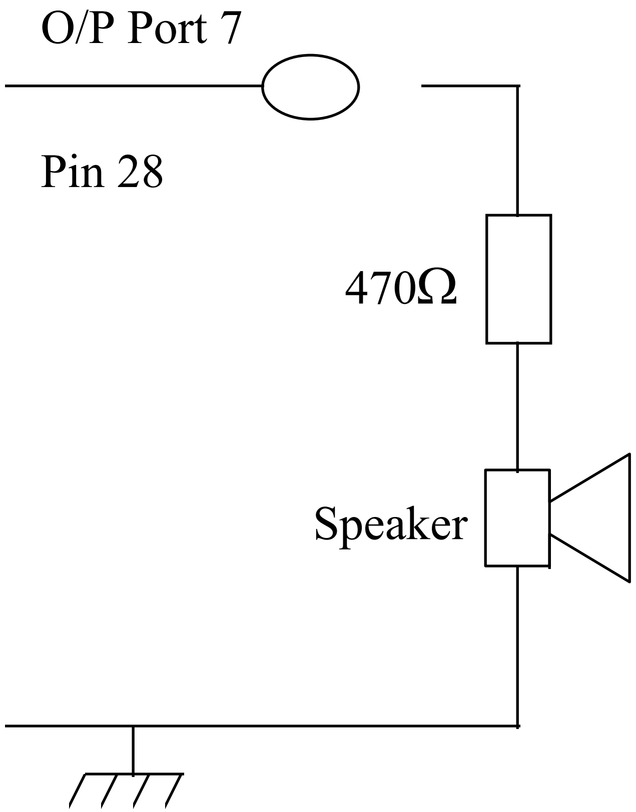
Using the Output Ports.

- More LED display activities.
- Create count up and down pattern.
- Create Knight Rider pattern.
- Create sequences of your choice.

Programming Activity 1A.

Making Sounds

Frequency and Music.



```
Rem The Programme Name Author/Date

Rem What the programme does .... (Describe)

‘Label   Code            Comment

Main:

         High 7          ‘Speaker Cone Out

         Pause 1         ‘Wait 1mSec

         Low 7           ‘Speaker Cone In

         Pause 1         ‘Wait 1mSec

         Goto Main       ‘Loop forever


Rem Period = 2mS therefore Frequency = 500Hz
```

Hint . . . . also try the Sound Command.

Frequency and Music.

Guitar

Tone	Frequency
E	329.628
B	246.942
G	195.998
D	146.832
A	110.000
E	82.4069

Bass

Tone	Frequency
G	97.9989
D	73.4162
A	55.0000
E	41.2034

Frequency and Music.

Note		Sharp/Flat	
Tone	**Frequency**	**Tone**	**Frequency**
E	165		
F	176	F#/Gb	185
G	198	G#/Ab	206
A	220	A#/Bb	233
B	247		
C	261.6	C#/Db	277
D	293.7	D#/Eb	311

Frequency and Music.

Note		Sharp/Flat	
Tone	**Frequency**	**Tone**	**Frequency**
E	329.2		
F	349.2	F#/Gb	370
G	392	G#/Ab	415
A	440	A#/Bb	466
B	495		
C	523.2	C#/Db	554
D	587.4	D#/Eb	622

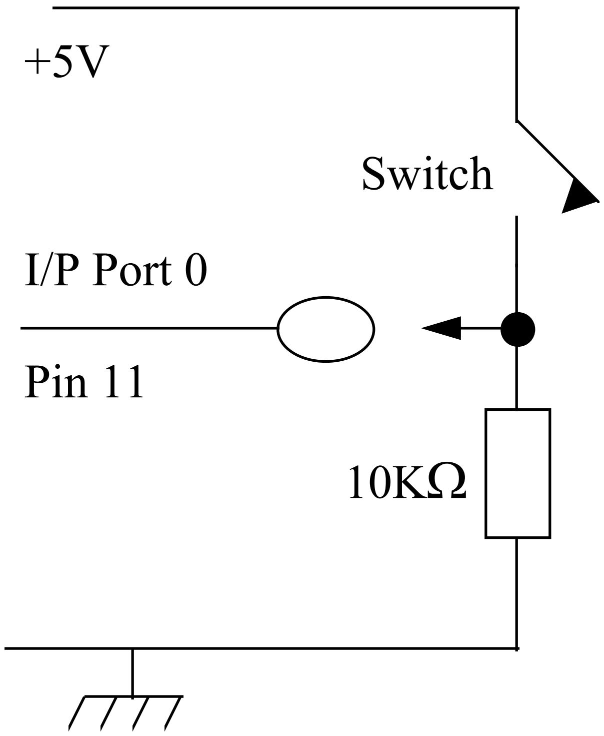
Frequency and Music.

- More sound activities.
- Create a siren of choice.
- Create simple tune.
- Create a sequences of siren or ring tones to represent alarm or system status states.

Programming Activity 2.

Using The Input Ports.

Using the Input Ports.



```
Rem The Programme Name Author/Date
Rem What the programme does .... (Describe)
'Label     Code                 Comment
Main:
           IF Pin0 = 0 THEN Ledoff
           High 7               'Switch LED ON
            Goto Main           'Loop forever

Ledoff:    Low 7                'Switch LED OFF
           Goto Main            'Loop forever
```

I/P Port 1= pin12, 2= pin13, ... 7 = pin 18

Using the Input Ports.

- Use Input ports to control LED Display Patterns.
- Use Input ports to control tones.
- Develop a simple application using Input ports.

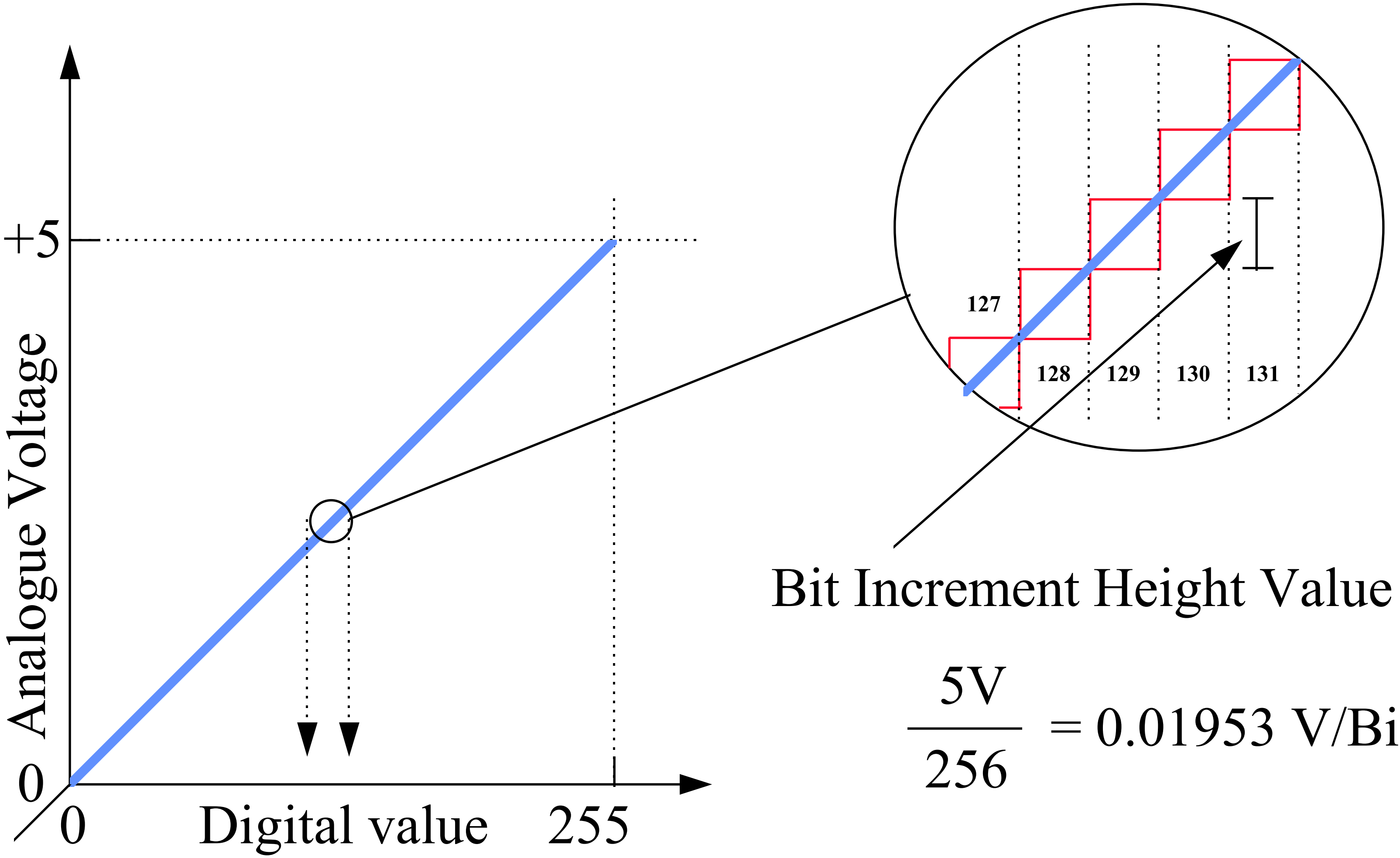
Programming Activity 3.

Using The ADC Ports.

The ADC Ports.

- ADC = Analogue Digital Converter/Conversion.
- These ports or channels convert an analogue signal into its digital representation.
- The Picaxe chip ADC channels divide the input signal into 256 discrete levels.
- The time to complete a conversion may take as long as 2mSecs.

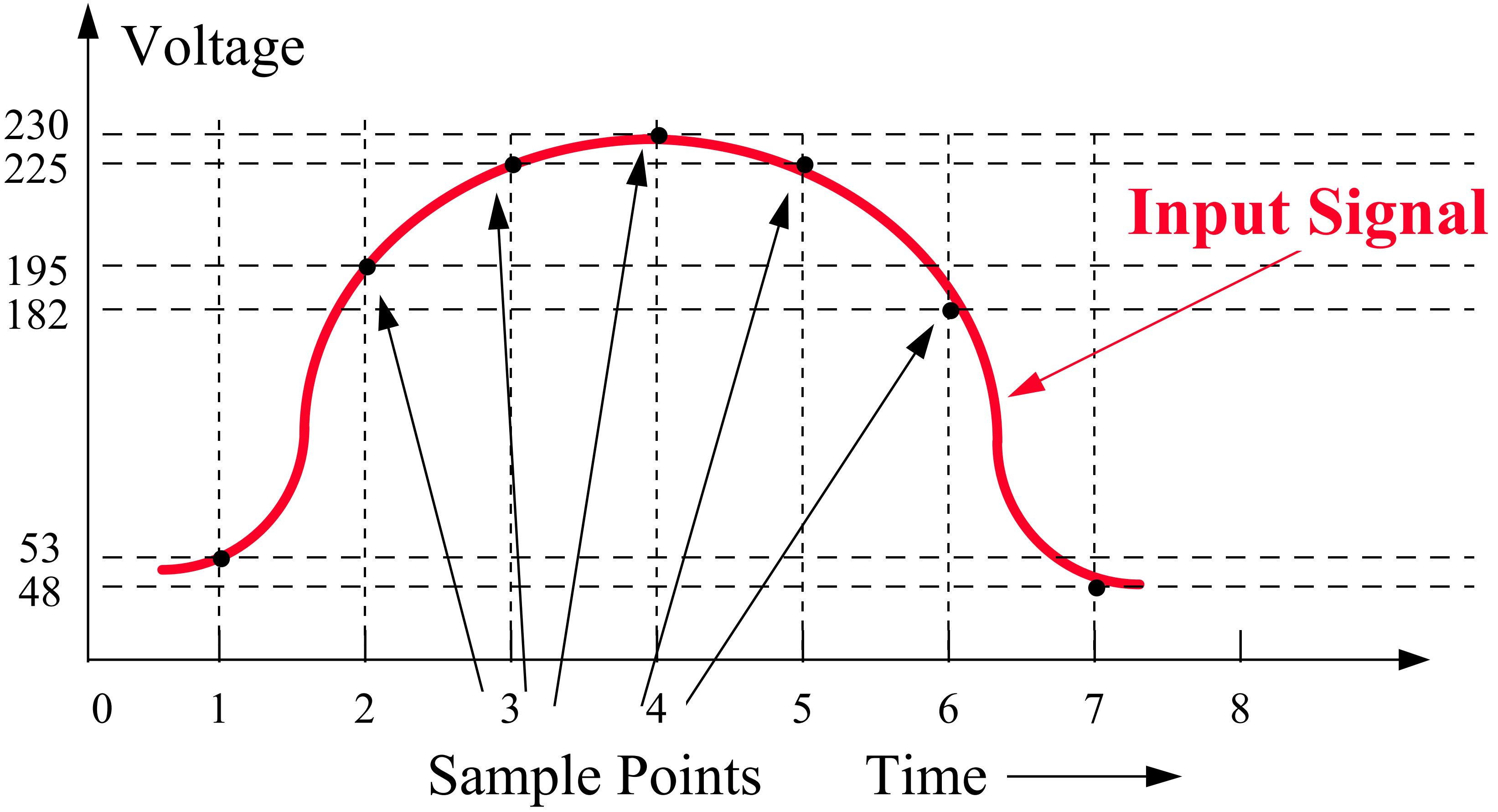
The ADC Ports.



Bit Increment Height Value

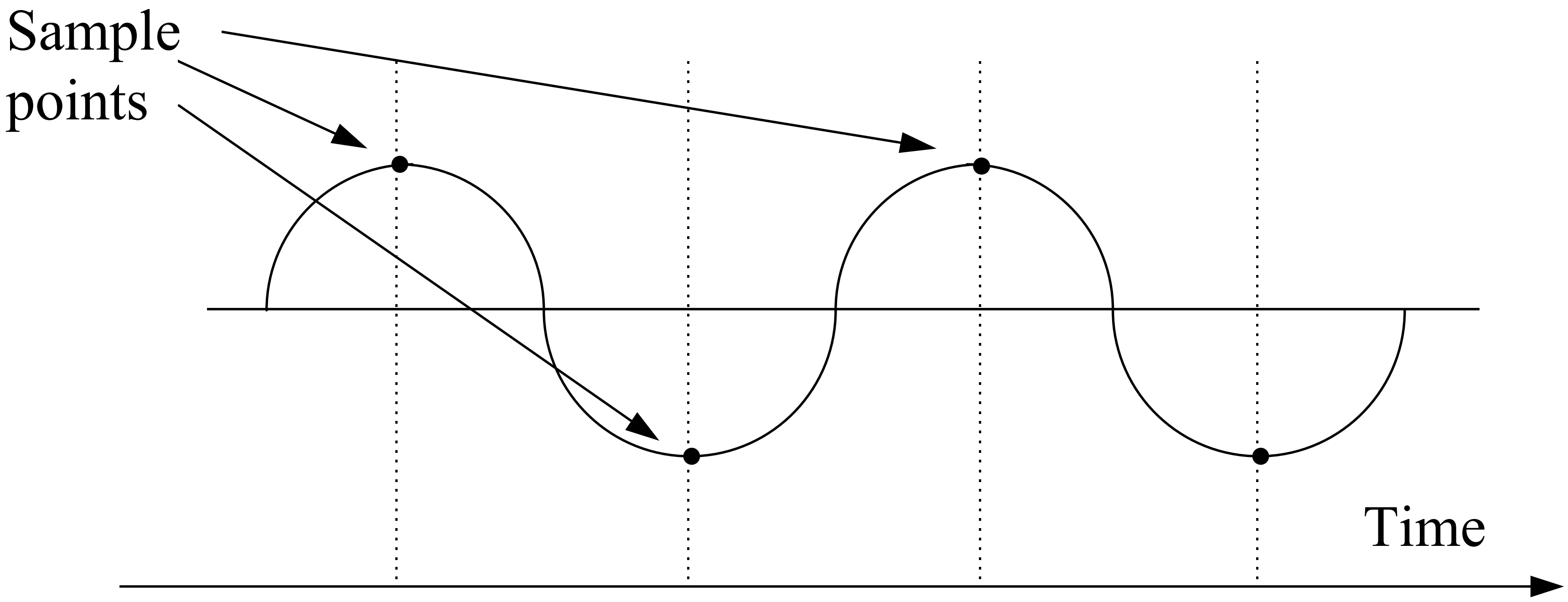
$$\frac{5V}{256} = 0.01953 \text{ V/Bit}$$

The ADC Ports.



Example of ADC Reading = 53, 195, 225, 230, 225, 182, 48....

The ADC Ports.

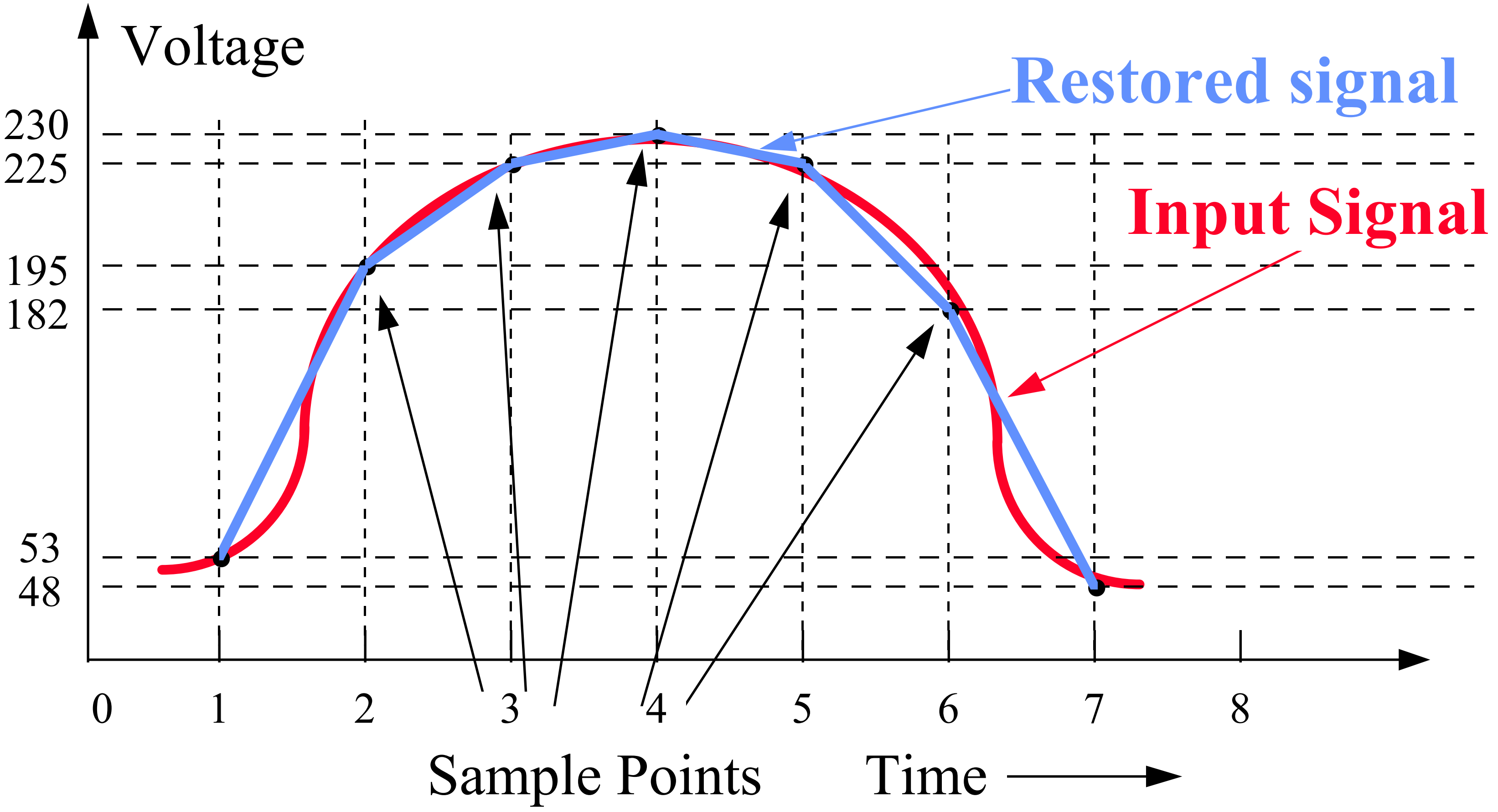


Nyquist Criterion

The Minimum Sample frequency is twice the frequency of the signal being sampled.

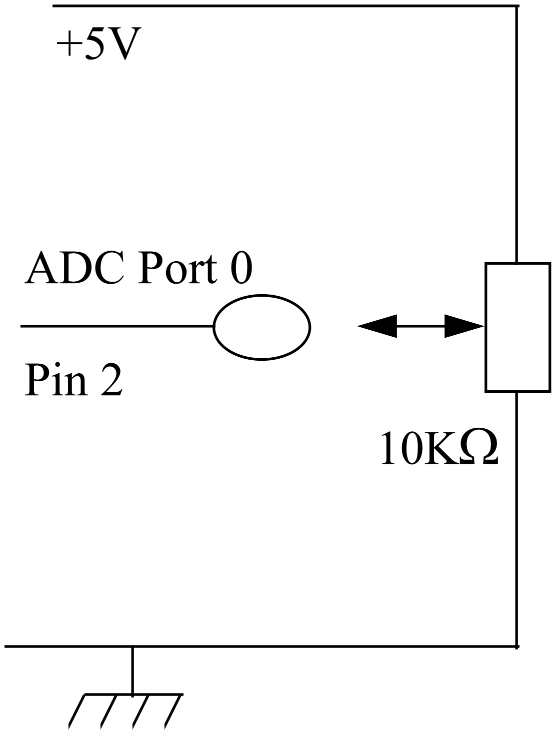
Recommended sample rate minimum is times ten to enable a successful signal restoration.

The ADC Ports.



Example of ADC Reading = 53, 195, 225, 230, 225, 182, 48....

Using the ADC Ports.

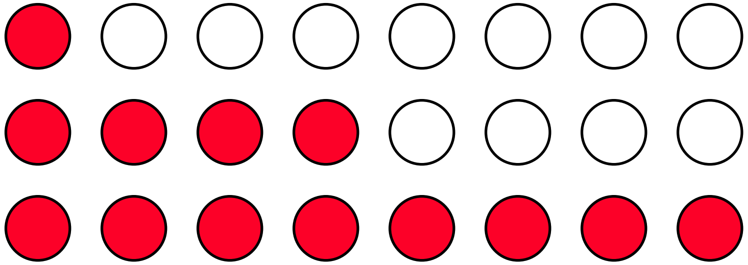


```
Rem The Programme Name Author/Date
Rem What the programme does .... (Describe)
‘Label   Code                Comment
Main:
         READADC 0,B0  ‘Read port Zero
         Pins = B0           ‘Switch LED ON
          Goto Main          ‘Loop forever
```

ADC Port 1= pin 3,   2= pin 4,   3 = pin 5

Using the ADC Ports.

- Use ADC port/s to control LED Display Patterns.
- Use ADC port/s to control tones.
- Develop a simple application using ADC port/s.
- Use Window voltage to activate control status displays or tones.
- Use ADC input to drive Simulated Bar Graph display on LEDs.

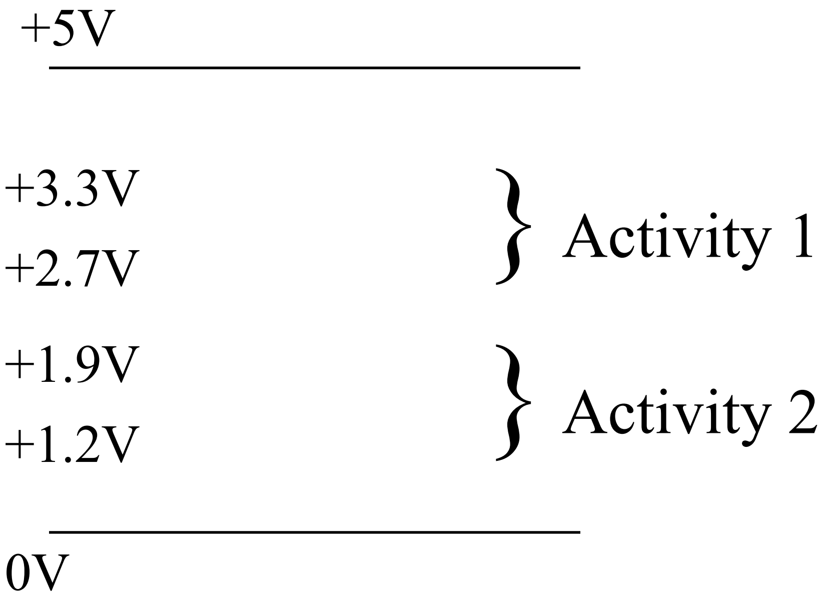


Low Voltage

Medium Voltage

High Voltage

Using the ADC Ports.



+5V

+3.3V

+2.7V

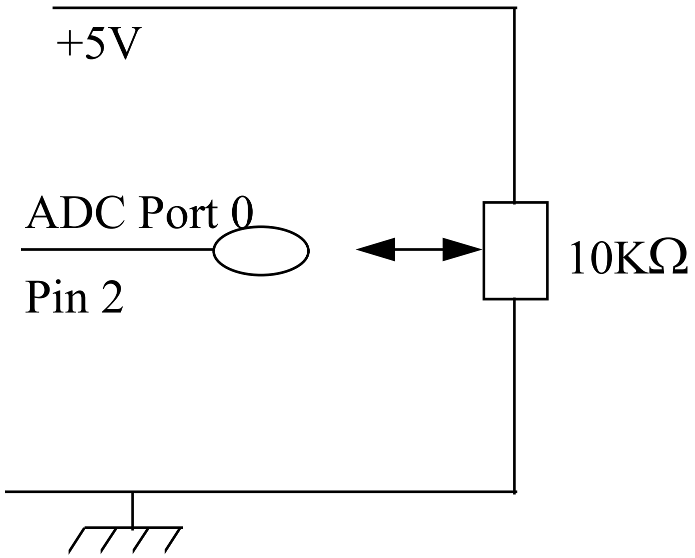
} Activity 1

+1.9V

+1.2V

} Activity 2

0V



Procedure

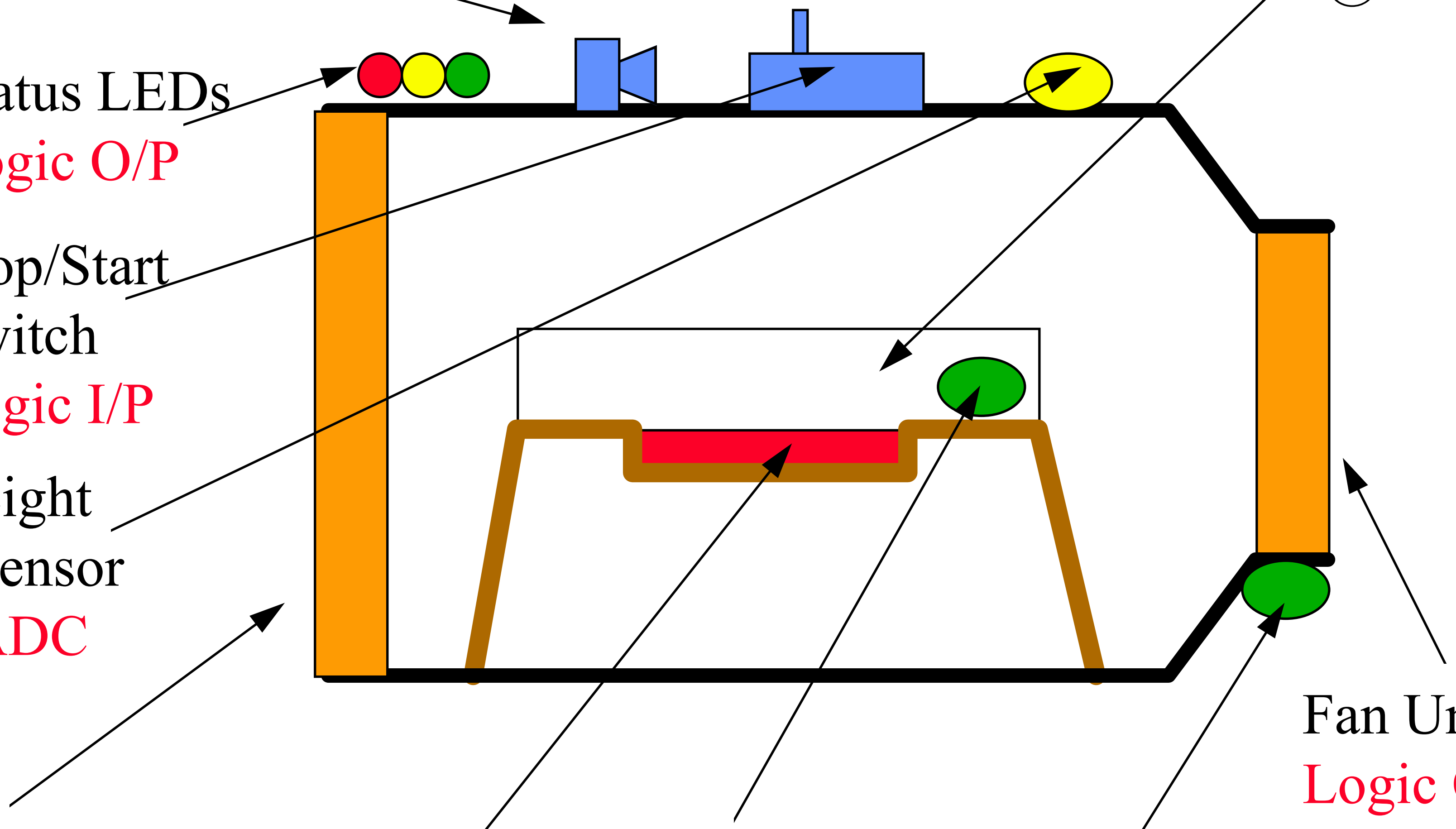
1. Find out what binary code equals target values.
2. Create programmes to identify Activity 1 and Activity 2.

ADC Port 1= pin 3, 2= pin 4, 3 = pin 5

Programming Activity 4.

Develop the Basic Project.

Project A.



Alarm
Logic O/P

Status LEDs
Logic O/P

Stop/Start
Switch
Logic I/P

Light
Sensor
ADC

Heat Tank @38°C

Fan Unit
Logic O/P

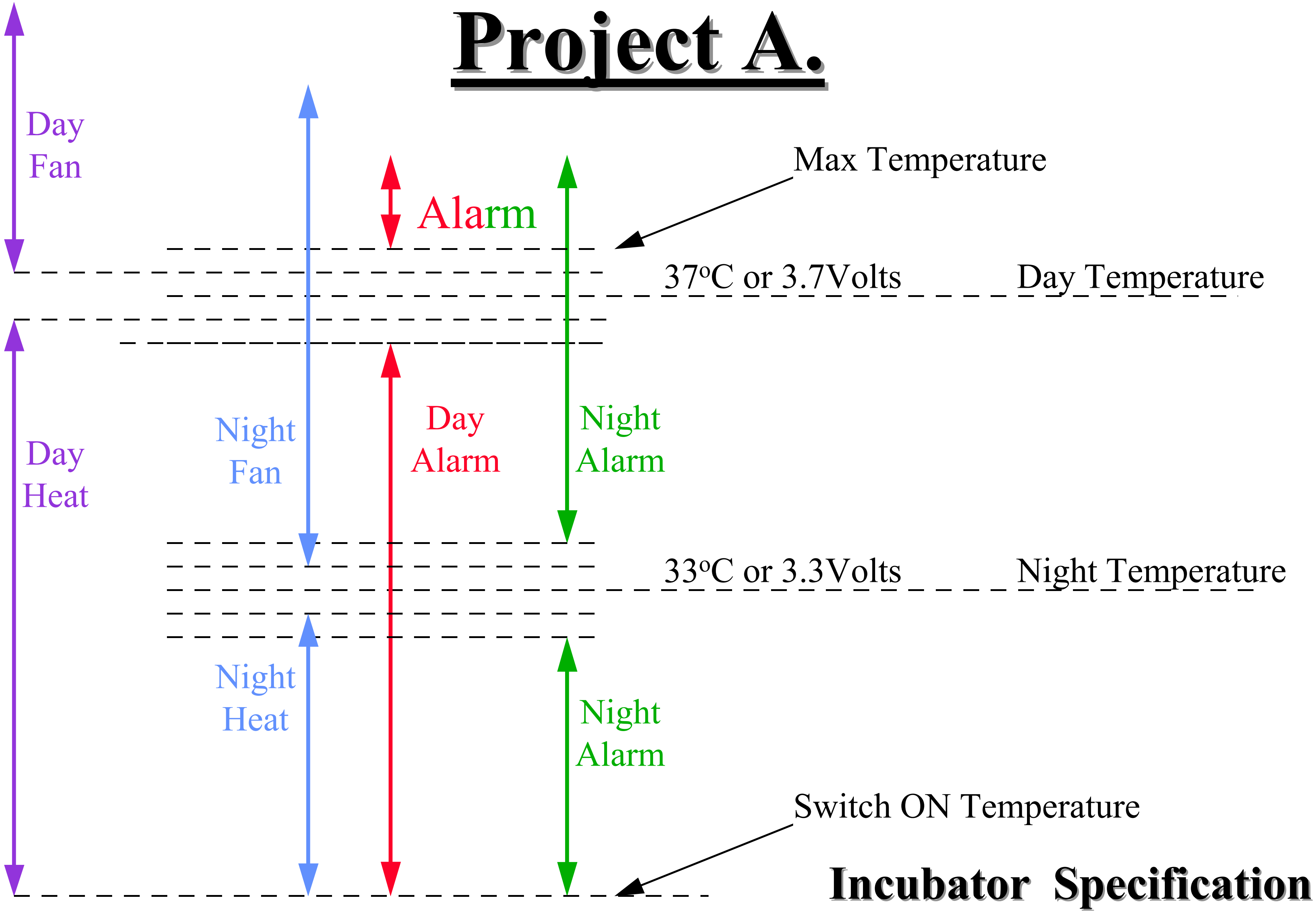
Vents

Heater
Logic O/P

Temperature
Sensor Water
ADC

Temperature
Sensor Air
ADC

Project A.



Incubator Specification Temperature Settings.

Project B Notes.

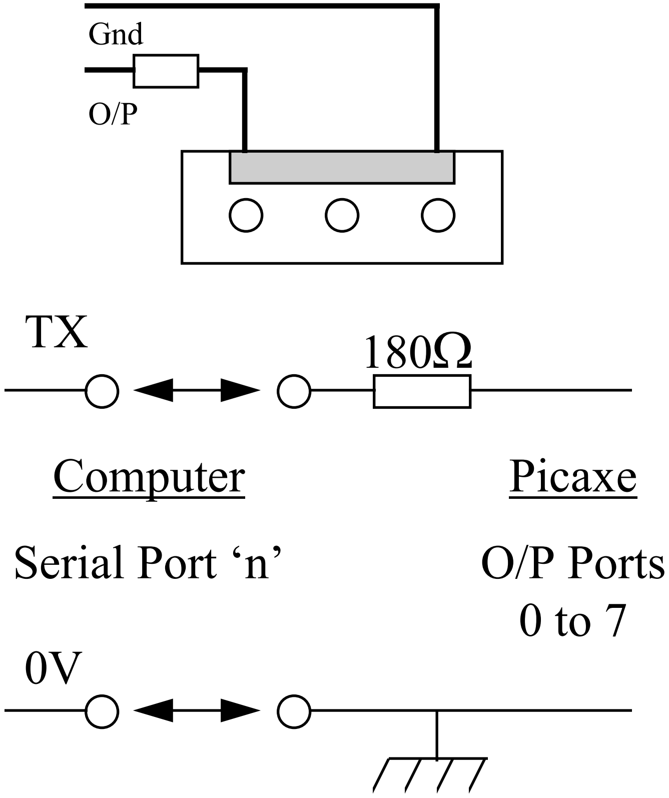
- Tower Full, ADC Voltage Circa 4.5 volts.
- Tower Empty, ADC Voltage Circa 0.5 volts.
- Daylight / Daytime ADC Voltage Circa 2.3 volts.
- Night time ADC Voltage Circa 4.0 volts.
- Remember eventually you will need to record the status of the majority of the controls you are monitoring.

Programming Activity 5.

Using Serial Communications.

Using Serial Communications.

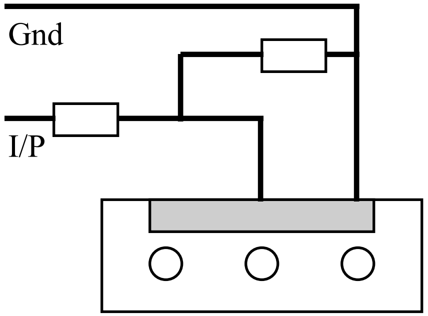
Sending data via an Output Port

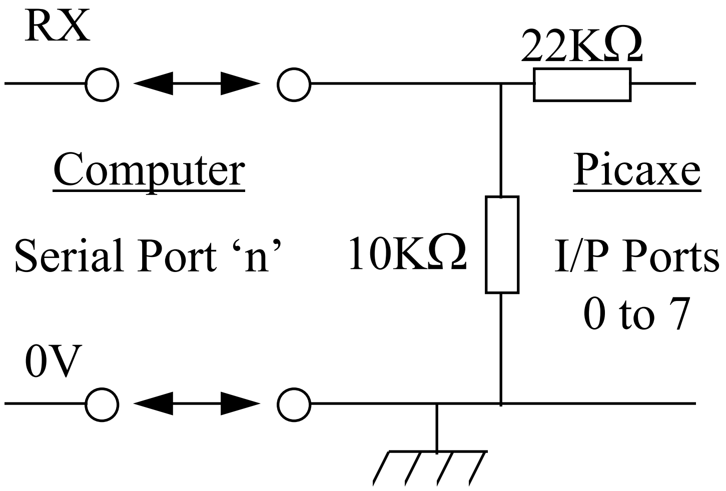


```
Rem The Programme Name Author/Date
Rem What the programme does .... (Describe)
‘Label    Code      Comment
                    ‘Serial Transmission programme
                    ‘Speed 1200 baud, using O/P Port 7
Main:               ‘Send Message on O/P Port Seven
          SEROUT  7,N1200,(”Hello World”)
                    ‘Send line Terminator on Port Seven
           SEROUT  7,N1200,(10,13)
          PAUSE 1000
           Goto Main         ‘Loop forever
```

Using Serial Communications.

Receiving data via an Input Port

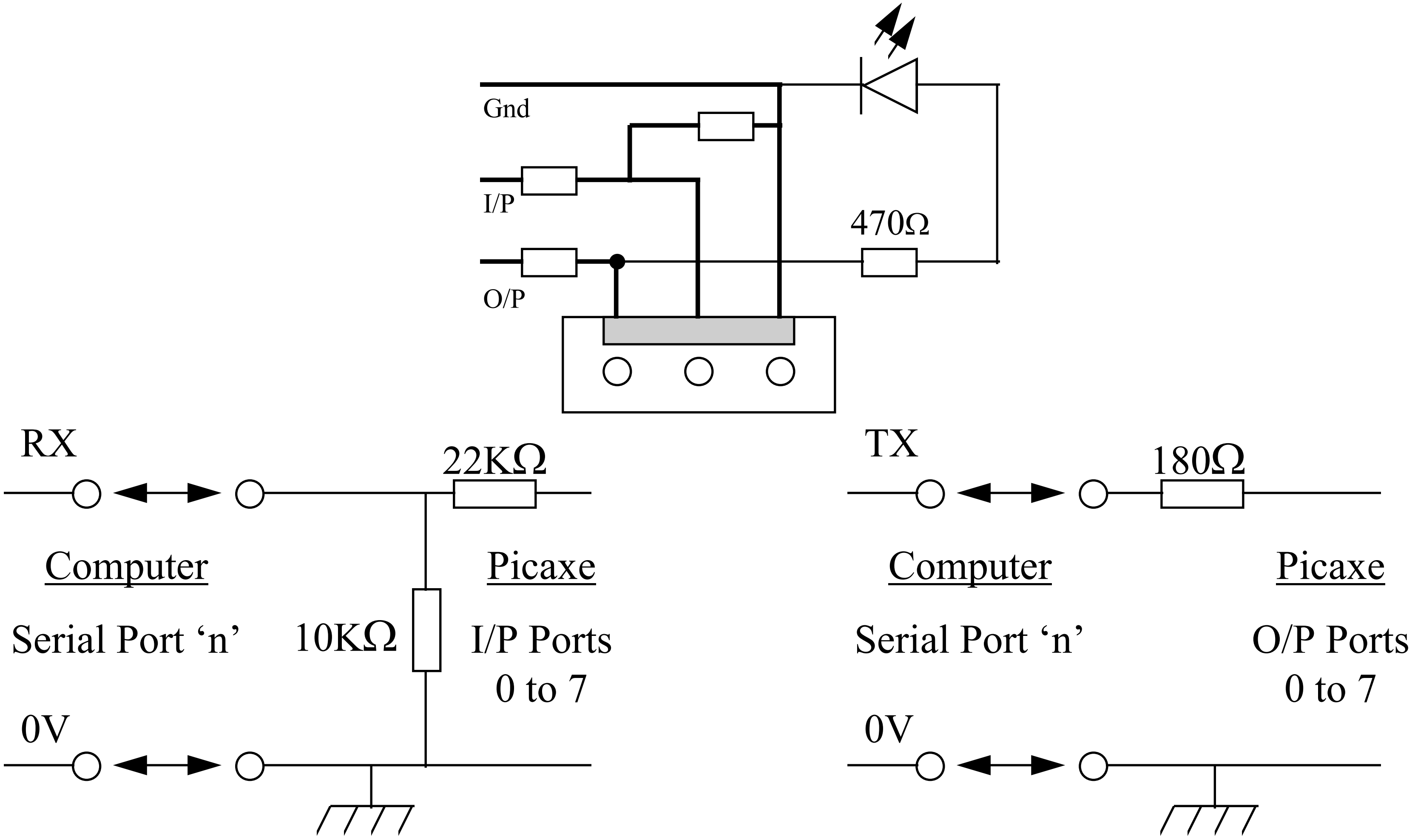




```
Rem The Programme Name Author/Date
Rem What the programme does .... (Describe)
'Label    Code      Comment
                    'Serial Transmission programme
                    'Speed 1200 baud, using I/P Port 7
Main:               'Wait for Message on I/P Port Seven
          SERIN  7,N1200,B2
                    'Display character pattern on LEDs
          Pins = B2
          Goto Main         'Loop forever
```

Using Serial Communications.

Full Serial Interface with optional LED Monitor.



Using Serial Communications.

Communications Programming Exercise.

- Get PICAXE to echo back on a serial line connection all characters sent to it.
- If the character “X” is received then send an additionally the message of “Hello World” with a new line terminator sequence.
- Set Transmit and Receive to 1200Baud, No Parity, Eight Bits, One Stop Bit. Set the system to indicate that there is no hardware handshake to be implemented on PC system.

Programming Activity 6.

Develop the Enhanced Project.

Enhanced Project B Notes.

- Record the status of the following controls :-
- Analogue Controls
  - Water Depth
  - Air Temperature
  - Light Intensity
- Digital Control
  - Status of Water Tower controller and / or
    - ~Alarms or other Status signals.
    - ~Manual Pump Status switch settings.
    - ~Pump driver signals.

Mini ITX

The Mini ITX.

- This is a complete PC mother board that has a small footprint.
- It has built in interface and connections for :-
  - Floppy Disk
  - Hard Disk and CD ROM
  - Keyboard, Mouse and Display
  - USB Connectors and Network capability
  - Audio and TV out capability
  - A PCI Bus Slot

System Concepts. and Terms

System Concepts and Terms.

- Basic considerations when building a system :-
- Hardware requirements.
  - Processor performance, type and Speed.
  - Quantity of Memory (RAM, ROM .. EPROM etc.)
  - Storage requirements (Disk Drives, R/W CD/DVD)
  - Common interfaces (Keyboards, Mouse, Display)
  - Specialist interfaces
    - Ease of connection.
    - Do they need specialist drivers?
  - Power requirements (Mains, UPS , Battery ...)

System Concepts and Terms.

- Basic considerations when building a system :-
- Software requirements.
  - Single or Multi-user.
  - Single Task or Multi-tasking.
  - Foreground or Background processing.
  - Interactive or Batch processing use.
  - Security, Backup, virus checking or attack.
  - Single processor, Multi processor or Networked.
  - Reliability and Crash recovery.
  - Bespoke or commercial (Windows, Linux ...)

System Concepts and Terms.

- Common Hardware usage terms :-
  - BIOS Basic Input/Output System
  - ISA Industry Standard Architecture
  - SCSI Small Computer System Interface
  - USB Universal Serial Bus
    - **USB1=12MBs , USB2=480MBs**
  - PCI Peripheral Component s Interconnect
  - AGP Accelerated Graphics Port
  - AMR Audio Modem Riser
  - SDRAM Synchronous Dynamic RAM,

System Concepts and Terms.

- Common Hardware usage terms :-
- BUS a common hardware interface that allows more than one device to be connected to the computer.
  - AT , ISA (8MHz), SCSI , PCI (33MHz) , USB.
- Specialist Interface Connectors
  - IDE, FDC, AGP , AMR.
- Memory Connectors.
  - 168 Pin DIMM for SDRAM,
  - 184 Pin DIMM for DDR

System Concepts and Terms.

- Common Hardware usage terms :-
- NorthBridge used for :-
  - AGP Graphic interface
  - IDE Hard Drive interface
  - USB , Audio (AC97) MIDI and Game ports
  - Modem interfaces, Serial and Parallel ports
- SouthBridge.
  - Network interfaces ie. 100BASE-TX
  - IDE Hard Drive DMA interface
  - Firewire (IEEE 1394) controller

Hardware Build Activities

System Configuration.

- With a PC system the first configuration activity is usually setting up the required hardware options.
- Hardware options are usually set up by :-
  - Hardware Jumper connections.
  - Setting in non volatile memory (CMOS RAM).
  - Current trends tend to use CMOS to configure everything that can be configured through CMOS.
- The CMOS settings will also contain the opportunity to set such things as **Date** and **Time** and what should happen if a **System Error** should occur.

Hardware Build Activity.

Assignment activity 1 (Part 1).

- Build basic ITX system, with floppy disk and small hard disk attached.
- Remove any partition information from Hard Disk.
- Create a primary DOS partition of 2GB.
- Create an extension DOS partition of 2GB.
- Create two logical drives in the extension partition one 1.2GB the other 0.8GB (Drives D: and E:)
- Make Primary partition active as the C: drive.

Hardware Build Activity.

Assignment activity 1 (Part 2).

- Format Drives C: , D: and E:
- Use LABEL command to name the drive D: as “DATASPACE” and drive E: as “WORKSPACE”
- Install DOS on your ITX.
- Backup your AUTOEXEC.BAT and CONFIG.SYS files to a protected area.
- Create a backup startup disk for your system.
- Run Scandisk to check your system (Keep a copy of the Log report).

Hardware Build Activity.

Assignment activity 1 (Part 3).

- Configure Start up menu to allow two or more start up options.
- Design/Build Serial capture application.
- Test your Application or an Issued Application.
- Use ITX and the Application to Log data from your controller system.
- Demonstrate how you would implement a Backup or Archive process to protect your data.
- Transfer Logged data to removable media.

Hardware Build Activity.

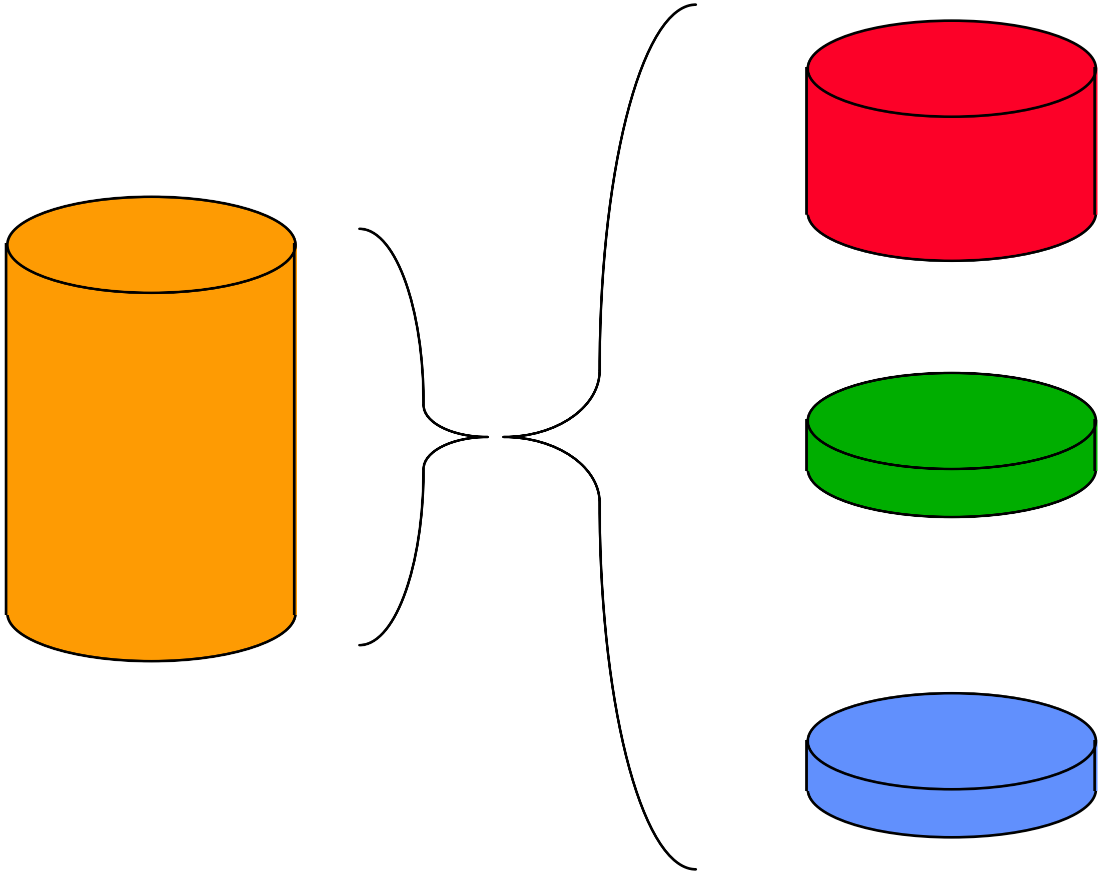
Assignment activity 1 (Part 4 & 5).

- Create and Run a Macro/s that will generate a file of test data suitable for processing by the Data logger analysis programme.

- Remove or uninstall your application/s and any associated data and control files from your system.
- Reset your machine to an initial state by removing the extension partition.

Resource Allocation Activity

Resource Allocation Activity.



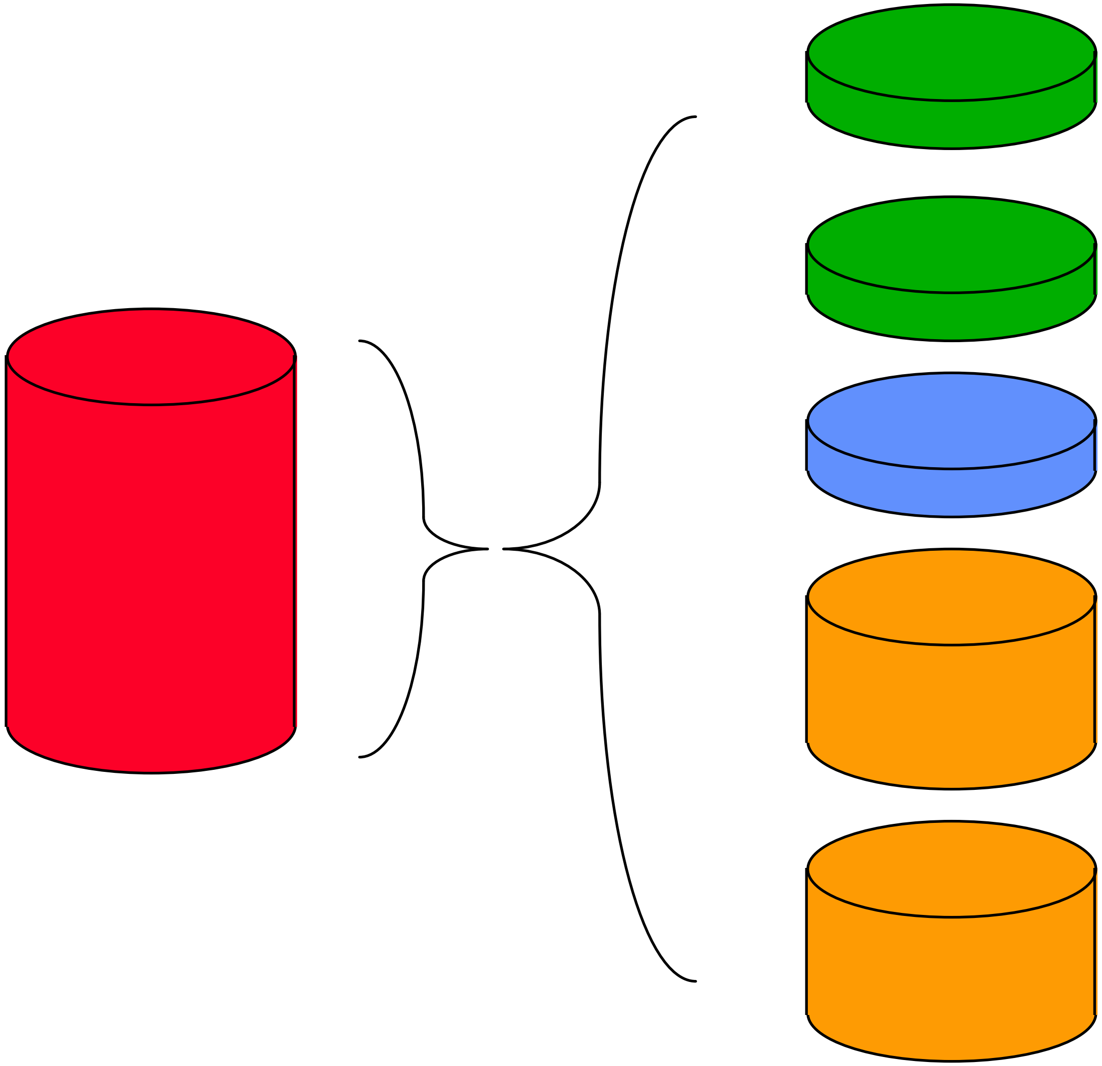
System Disk
C: Drive

Data Disk 1
D: Drive

Data Disk 2
E: Drive

The Main Disk is Divided into three separate sub disks.

Resource Allocation Activity.



Operating System
C:\DOS

Configurations
C:\DOS\CONFIG

Applications
C:\APP

Main Data Store
C:\Data_M

1st Archive
C:\Data_S

The System Disk is Sub Divided into the above areas.

Example System Specification

System Specification.

Typical Headings that could be used based on user requirements.

- System Specification
  - General Specification
  - Hardware Specification
    - Hardware Requirements
      - Resources required
      - Explanation of requirement and for resources
  - Software Specification
    - Software Requirements (With explanation as above)

DOS System Start.

DOS System Start.

- The BIOS starts the BOOT block code on your System Disk.
- The BOOT block code loads the Operating System (O/S) from your System Disk.
- IF the Operating System is DOS (Disk Operating System) then DOS will interpret the commands stored in the text file CONFIG.SYS and then will process the commands in the AUTOEXEC.BAT file.
- As a General guide CONFIG.SYS contents are **only processed once** commands whereas the contents of AUTOEXEC.BAT may be processed more than once.

System Configuration.

System Configuration.

Brief description of some common CONFIG.SYS Commands.

Command	**Description**
BREAK	Extended BREAK checking [Ctrl][C]
BUFFERS	Number of sector buffers.
COUNTRY	Country Specific parameter selection.
DEVICE	Device driver installations.
DRIVPARM	Override the drive parameters for a logical drive
FCBS	Maximum number of file control blocks available concurrently.
FILES	Maximum number of file handles open concurrently
INSTALL	Execute a command during CONFIG.SYS processing.
LASTDRIVE	Maximum drive letter allowable.
REM or ;	Allows a comment.
SET	Set the value of environmental variables such as PROMPT or TEMP
SHELL	Top level command processor specification
STACKS	Override the default DOS stack resource.

System Configuration.

Brief description of the MENU Commands for CONFIG.SYS.

Command	Description
Menu	[MENU] ;Where the start menu begins
Menucolor	menucolor=text[,background]
Menuedefault	menudefault=blockname[,timeout (0 to 90 seconds)]
Menuitem	menuitem=blockname[,menu_text]
Common	[COMMON] Note a sequential Block of Common commands (repeatable)
Submenu	submenu=blockname[,menu_text]
Include	include=blockname Include the contents of one configuration block within another. This can only be used in CONFIG.SYS
numlock	numlock=[on/off] Note this command can be used anywhwere in CONFIG.SYS however it is especially useful when defining a startup menu.

System Configuration.

```
REM Example of a Menu controlled CONFIG.SYS
[Menu]
menuitem = Auto_Config, Auto Start Application Programme
menuitem = User_Config, Start in Command Mode.
menudefault = Auto_Config, 30

[Default_setting]
Lastdrive=M
COUNTRY=044,437, C:\DOS\COUNTRY.SYS
FILES=40
BUFFERS=25

[Auto_Config]
include = Default_setting

[User_Config]
include = Default_setting
```

System Configuration.

```
REM Example of a typical Menu controlled AUTOEXEC.BAT
@ECHO OFF
CLS
PROMPT $p$g
PATH C:\;C:\DOS;C:\APP
KEYB UK,437,C:\DOS\KEYBOARD.SYS
VER
DOSKEY
ECHO Current Configuration is %CONFIG%
IF "%CONFIG%"=="Auto_Config" C:\DOS\QBASIC /RUN C:\APP\LOGGER.BAS
ECHO .
ECHO . Starting DOS Command Mode
ECHO .
DIR/W
```

DOS Commands.

DOS Commands.

Brief description of some common DOS Commands.

Command	Description
HELP	Gives assistance with or about the DOS commands.
DIR	Shows contents of a Directory / Folder.
PROMPT	Set or adjusts the DOS Prompt.
CD	Change to a different directory.
MD	Make a new directory.
RD	Remove or Delete an existing directory.
TYPE	Display the contents of a Text file.
DEL	Delete a file.
COPY	Copy a file from one location to another.
MOVE	Move a file from one location to another.
DOSKEY	Allow recording and editing of command lines
DELTREE	Delete files and Directories as a block activity.

DOS Commands.

Brief description of some common DOS Commands.

Command	**Description**
RENAME	Change the name of a file
PATH	Show or Set Directory Paths to search to find a file.
SET	Set an environmental variable
DISKCOPY	Make an image copy of a floppy disk disk
DISKCOMP	Compare images of two floppy disks
FC	Compare contents of two files
DATE	Change or display Date
TIME	Change or display Time
LABEL	Change disk volume label
XCOPY	Improved version of COPY
CD .	Go to Current Directory
CD ..	Go to Parent Directory
CD \	Go to Root Directory

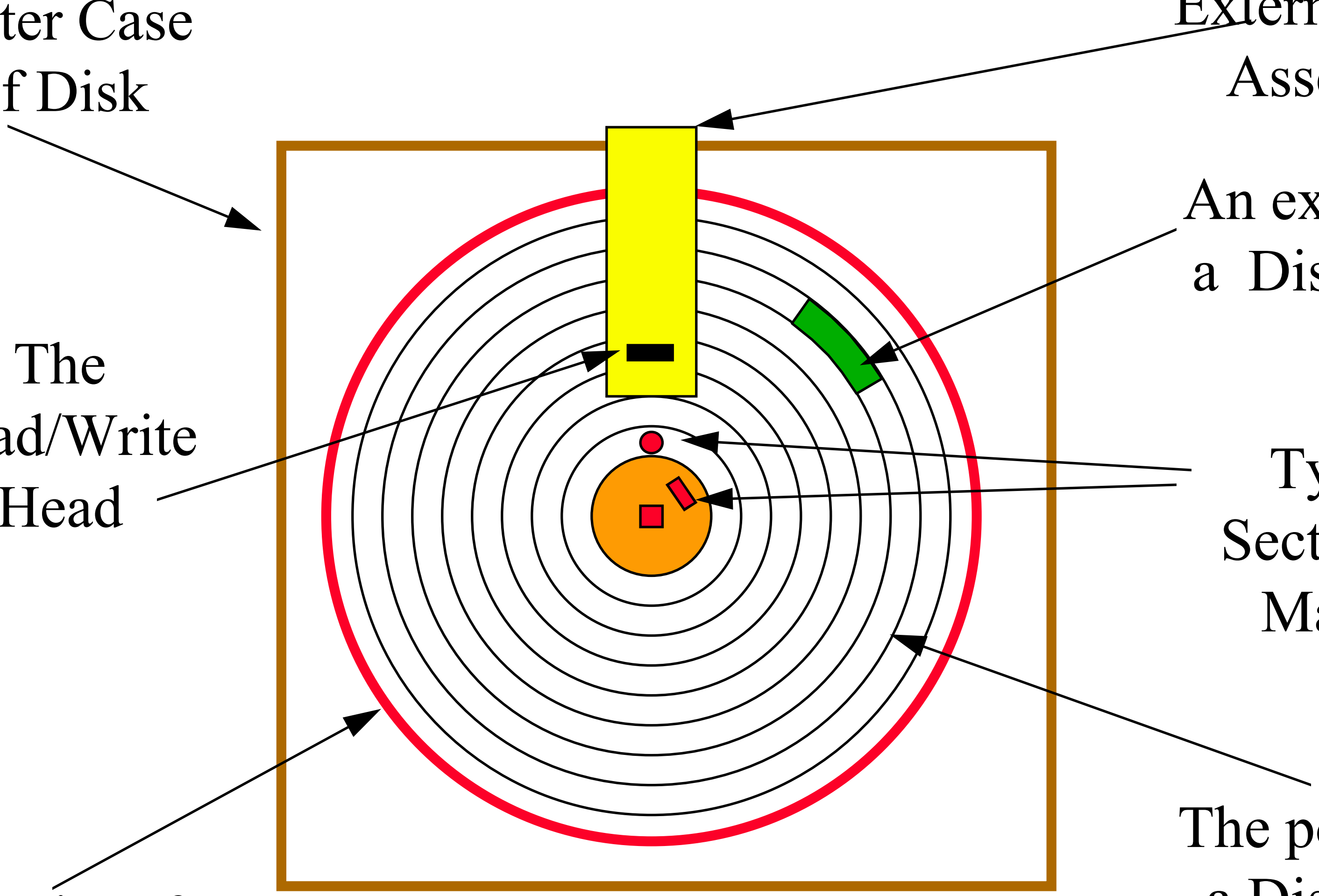
DOS Commands.

Brief description of some common DOS Commands.

Command	Description
FDISK	Programme to set up Fixed Disk Partitions.
EDIT	Programme to Edit Text Files (Needs file QBASIC.EXE to be available.)
QBASIC	BASIC programming language.
FORMAT	Programme to Format/prepare disks.
SCANDISK	Programme to check integrity of a disk.
DEFRAG	Programme to tidy or sort out contents of disk.
MSD	Diagnostics programme
PRINT	Print a Text File (if you have a printer available)
MORE	Special Pipe programme to show screen page at a time.

Disk Format.

Disk Format.



Outer Case of Disk

External Head Assembly

An example of a Disk Sector

The Read/Write Head

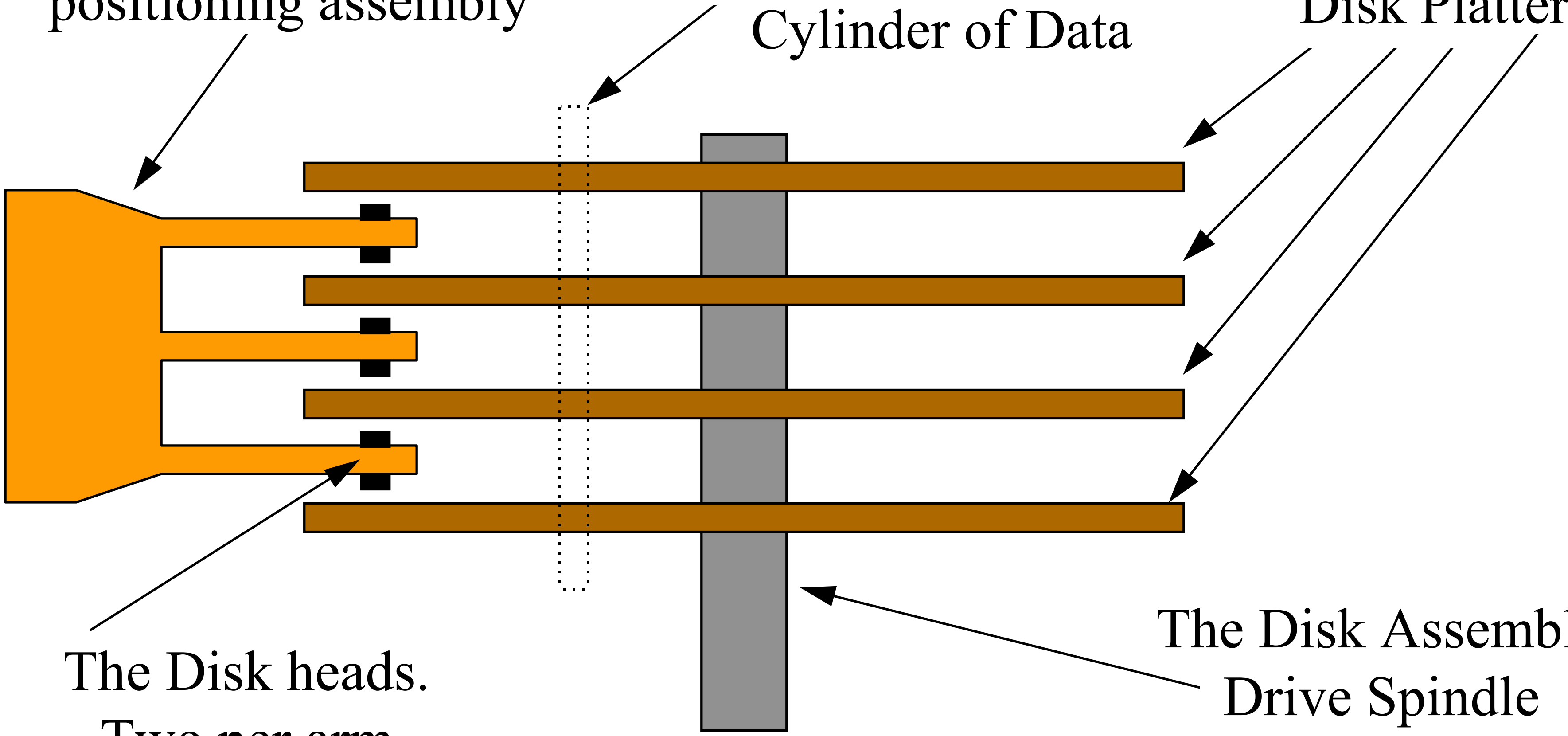
Typical Sector Zero Markers

The position of a Disk Track

Outer rim of the Disk

Typical Disk Layout

Disk Format.



Voice coil head positioning assembly

An Example of a Cylinder of Data

Disk Platters

The Disk heads. Two per arm

The Disk Assembly Drive Spindle

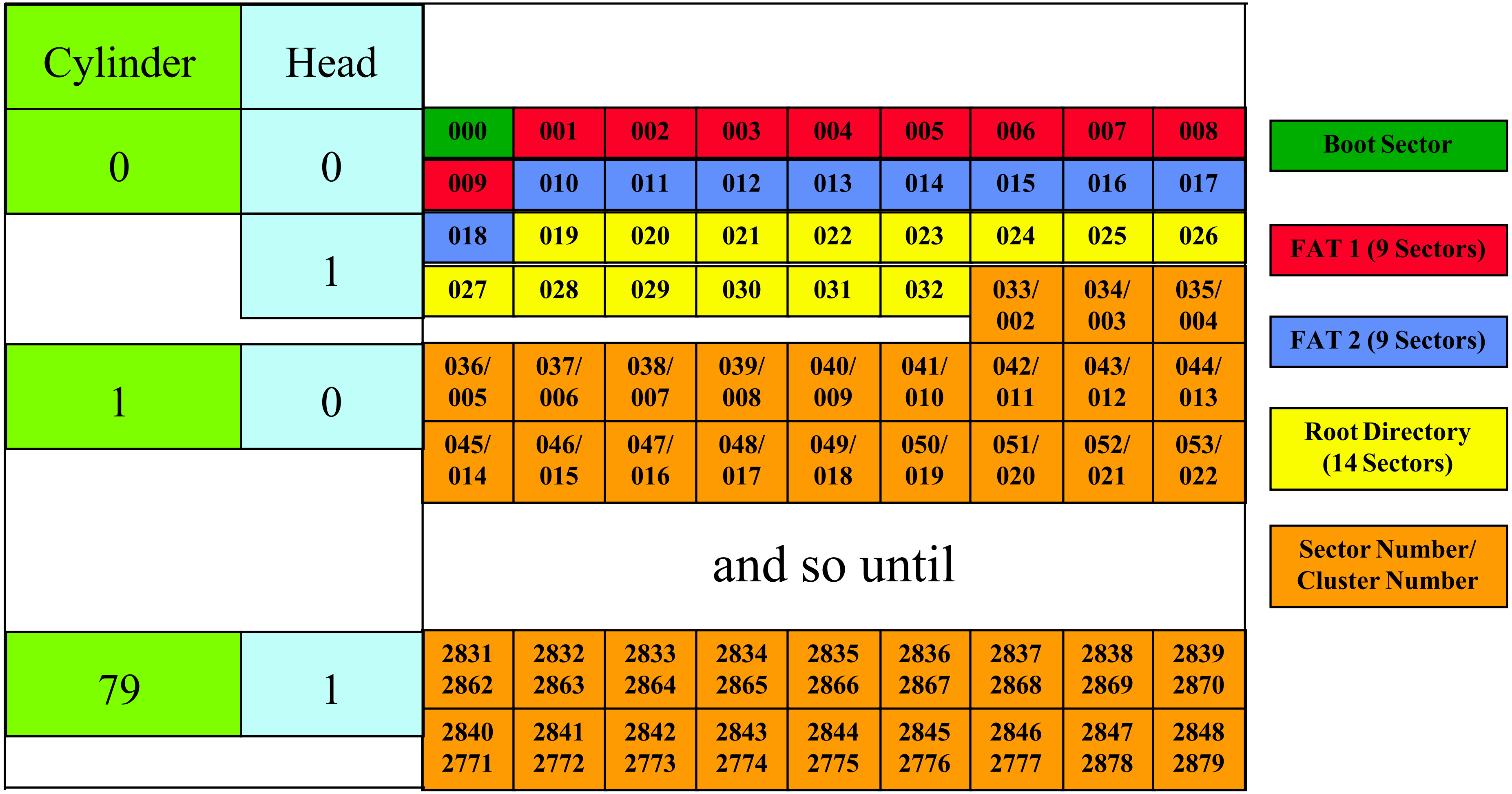
Typical Multi Platter Disk Assembly Layout

Definitions

Disk Format.

- Some basic Terms :-
- A "Block" is 512 bytes of data.
- A "Sector" the smallest area of data that can be read from or written to a disk surface.
- A Cylinder is all the sectors that are accessible without moving the head or heads
- A "Cluster" is one or more "Sectors".
- File Allocation Table "FAT" a map of what areas on a disk have been used. Usually two or more just in case one get corrupted.

Disk Format.



Cylinder	Head									
0	0	000	001	002	003	004	005	006	007	008
		009	010	011	012	013	014	015	016	017
	1	018	019	020	021	022	023	024	025	026
		027	028	029	030	031	032	033/ 002	034/ 003	035/ 004
1	0	036/ 005	037/ 006	038/ 007	039/ 008	040/ 009	041/ 010	042/ 011	043/ 012	044/ 013
		045/ 014	046/ 015	047/ 016	048/ 017	049/ 018	050/ 019	051/ 020	052/ 021	053/ 022
		and so until								
79	1	2831 2862	2832 2863	2833 2864	2834 2865	2835 2866	2836 2867	2837 2868	2838 2869	2839 2870
		2840 2771	2841 2772	2842 2773	2843 2774	2844 2775	2845 2776	2846 2777	2847 2878	2848 2879

- 🟩 Boot Sector
- 🟥 FAT 1 (9 Sectors)
- 🟦 FAT 2 (9 Sectors)
- 🟨 Root Directory (14 Sectors)
- 🟧 Sector Number/ Cluster Number

Typical MS Floppy 12 Bit Fat Disk Structure.

Disk Format.

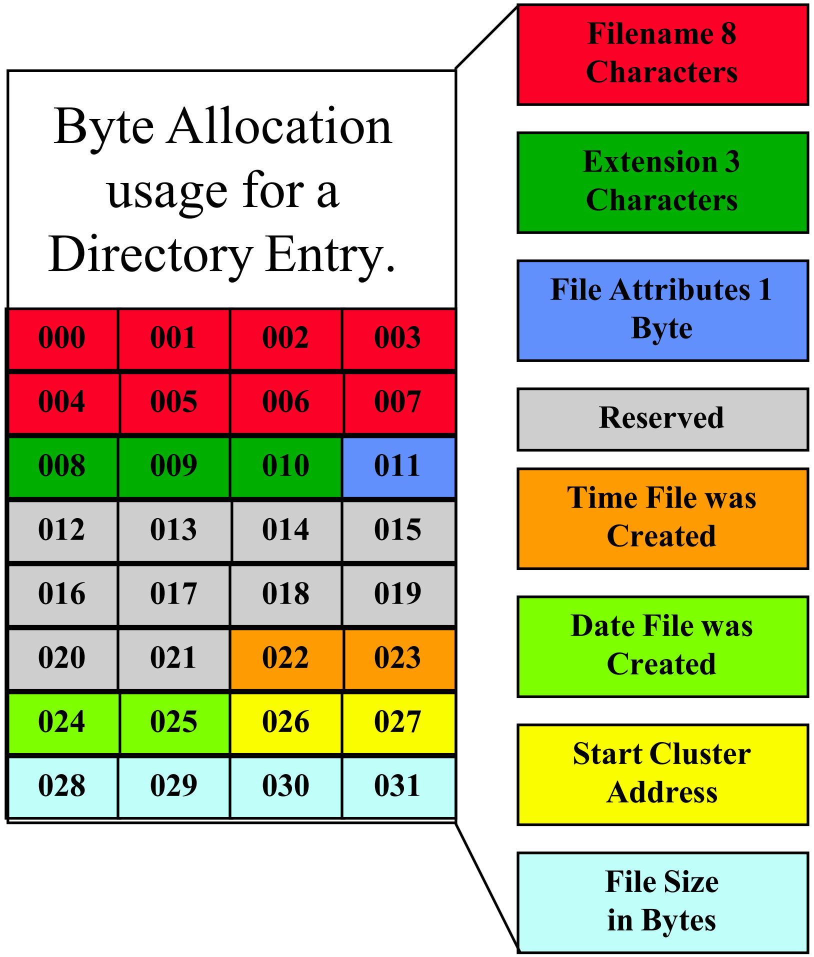
Calculations and Notes.

32 Bytes per Directory Entry.

14 Sectors * 512 Bytes allocated.

Therefore maximum number of entries in the Root Directory is 224.

Note: Once all the Root Directory Entries have been used then the disk is full.



Byte Allocation usage for a Directory Entry.

000	001	002	003
004	005	006	007
008	009	010	011
012	013	014	015
016	017	018	019
020	021	022	023
024	025	026	027
028	029	030	031

Filename 8 Characters

Extension 3 Characters

File Attributes 1 Byte

Reserved

Time File was Created

Date File was Created

Start Cluster Address

File Size in Bytes

Typical Directory Entry Structure.

Disk Format.

- So how is a **File** stored:
- The system locates a FREE Directory entry.
- The system locates a FREE Cluster from the FAT.
- The Directory Entry has its Start cluster slot updated
- The FAT is updated with the cluster marked as the end of file.
- Store the data using the **File Records** procedure.
- Complete any outstanding file handling house keeping activities.

Disk Format.

- So how are the **File Records** stored:
- A Record or part Record is written to the Cluster. The Directory entry File length file is updated.
- If Another Cluster is required
  - The system locates a FREE Cluster from the FAT.
  - The Previous FAT Cluster entry is updated with the Current Cluster address. (Chained)
  - The Current FAT Cluster entry is updated with the Cluster marked as the end of file.
- Repeat till all Data is transferred to the file.

Disk Format.

What happens as we :-

Create,

Modify

and Delete files ?

Animation Slides follow.

Disk Format.

Empty Clusters

A Initial Formatted Disk with no files on the disk.

First activity: Write Three files to the disk.

		002	003
004	005	006	007
008	009	010	011
012	013	014	015
016	017	018	019
020	021	022	023
024	025	026	027
028	029	030	031

File Fragmentation (First Part of Disk Only).

Disk Format.

Empty Clusters

File 1
4 Clusters Long

Write a File

		002	003
004	005	006	007
008	009	010	011
012	013	014	015
016	017	018	019
020	021	022	023
024	025	026	027
028	029	030	031

File Fragmentation (First Part of Disk Only).

Disk Format.

Empty Clusters

File 1
4 Clusters Long

Write a File

		002	003
004	005	006	007
008	009	010	011
012	013	014	015
016	017	018	019
020	021	022	023
024	025	026	027
028	029	030	031

File Fragmentation (First Part of Disk Only).

Disk Format.

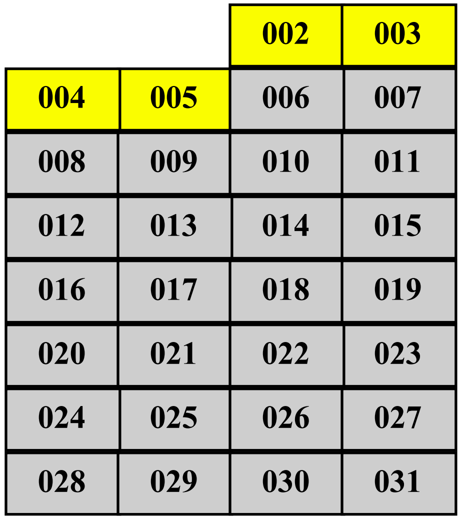
Empty Clusters

File 1
4 Clusters Long

File 2
3 Clusters Long

Write a File

Write a File



		002	003
004	005	006	007
008	009	010	011
012	013	014	015
016	017	018	019
020	021	022	023
024	025	026	027
028	029	030	031

File Fragmentation (First Part of Disk Only).

Disk Format.

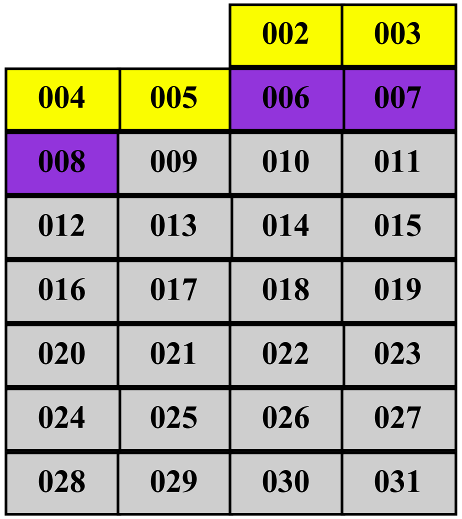
Empty Clusters

File 1
4 Clusters Long

File 2
3 Clusters Long

Write a File

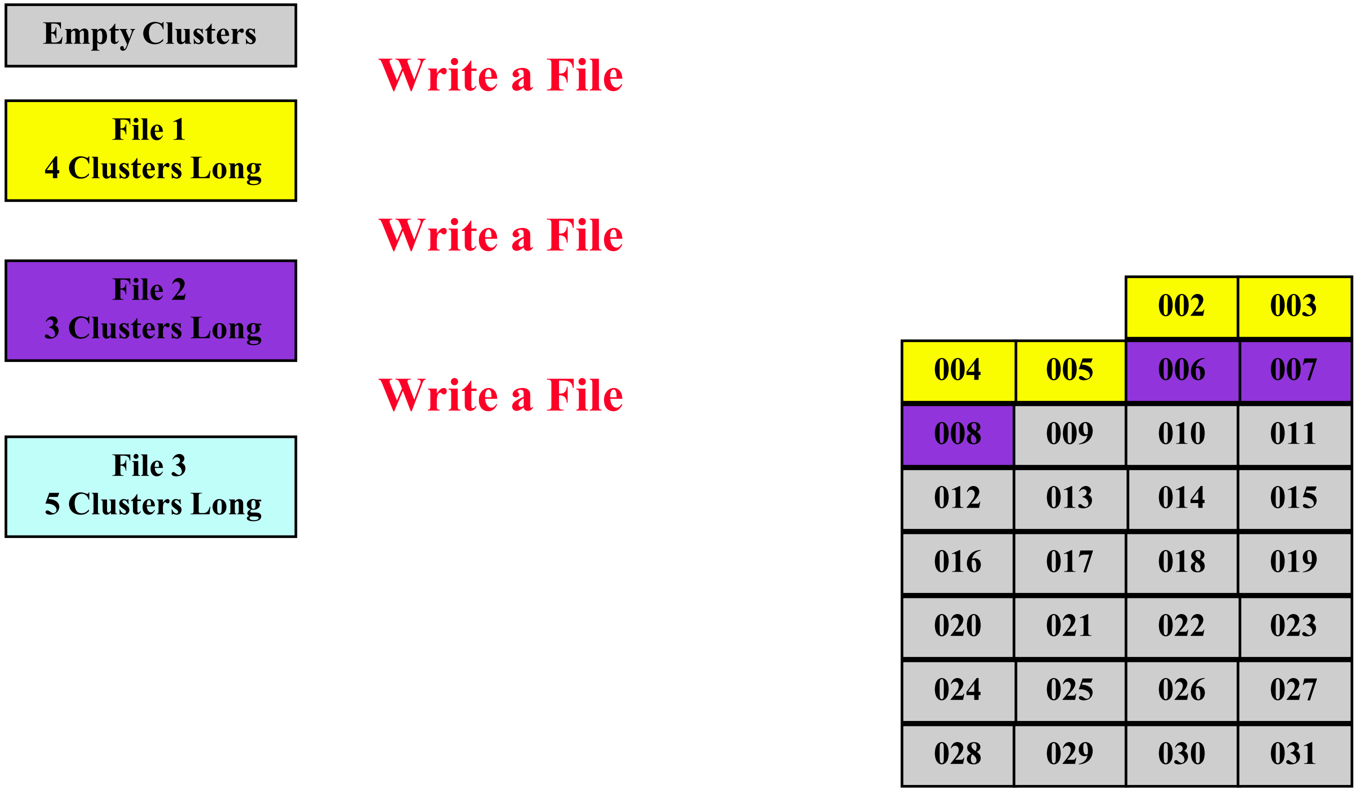
Write a File



		002	003
004	005	006	007
008	009	010	011
012	013	014	015
016	017	018	019
020	021	022	023
024	025	026	027
028	029	030	031

File Fragmentation (First Part of Disk Only).

Disk Format.



Empty Clusters

File 1
4 Clusters Long

File 2
3 Clusters Long

File 3
5 Clusters Long

Write a File

Write a File

Write a File

		002	003
004	005	006	007
008	009	010	011
012	013	014	015
016	017	018	019
020	021	022	023
024	025	026	027
028	029	030	031

File Fragmentation (First Part of Disk Only).

Disk Format.

Empty Clusters

File 1
4 Clusters Long

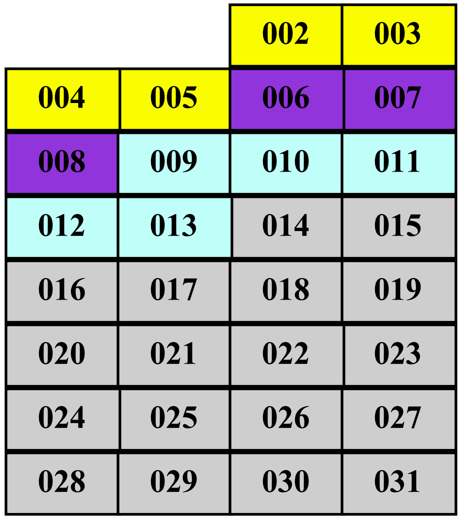
File 2
3 Clusters Long

File 3
5 Clusters Long

Write a File

Write a File

Write a File



		002	003
004	005	006	007
008	009	010	011
012	013	014	015
016	017	018	019
020	021	022	023
024	025	026	027
028	029	030	031

File Fragmentation (First Part of Disk Only).

Disk Format.

Empty Clusters

File 1
4 Clusters Long

File 2
3 Clusters Long

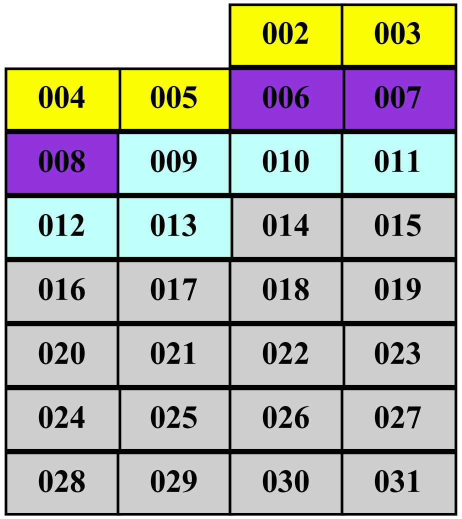
File 3
5 Clusters Long

Write a File

Write a File

Write a File

Delete File 2



		002	003
004	005	006	007
008	009	010	011
012	013	014	015
016	017	018	019
020	021	022	023
024	025	026	027
028	029	030	031

File Fragmentation (First Part of Disk Only).

Disk Format.

Empty Clusters

File 1
4 Clusters Long

File 2
3 Clusters Long

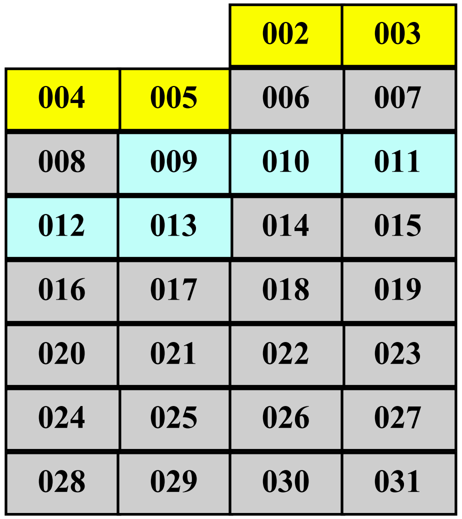
File 3
5 Clusters Long

Write a File

Write a File

Write a File

Delete File 2



		002	003
004	005	006	007
008	009	010	011
012	013	014	015
016	017	018	019
020	021	022	023
024	025	026	027
028	029	030	031

File Fragmentation (First Part of Disk Only).

Disk Format.

Empty Clusters

File 1
4 Clusters Long

File 2
3 Clusters Long

File 3
5 Clusters Long

File 4
10 Clusters Long

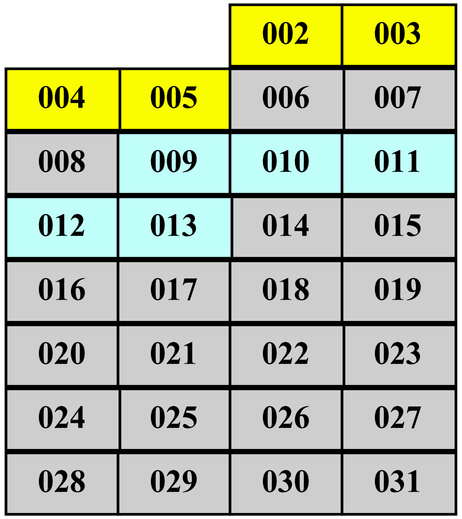
Write a File

Write a File

Write a File

Delete File 2

Write a File



		002	003
004	005	006	007
008	009	010	011
012	013	014	015
016	017	018	019
020	021	022	023
024	025	026	027
028	029	030	031

File Fragmentation (First Part of Disk Only).

Disk Format.

Empty Clusters

Write a File

File 1
4 Clusters Long

Write a File

File 2
3 Clusters Long

Write a File

File 3
5 Clusters Long

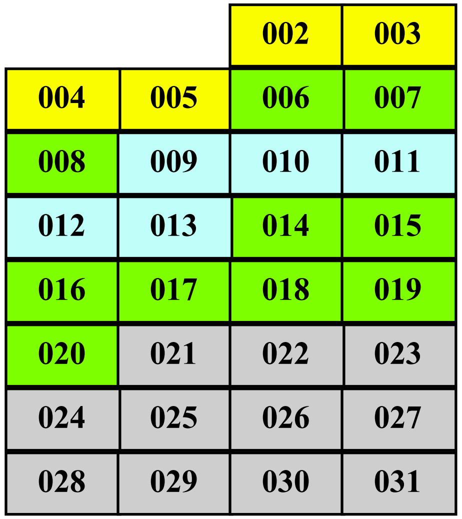
Delete File 2

Write a File

File 4
10 Clusters Long

Write a File

File 5
7 Clusters Long



		002	003
004	005	006	007
008	009	010	011
012	013	014	015
016	017	018	019
020	021	022	023
024	025	026	027
028	029	030	031

File Fragmentation (First Part of Disk Only).

Disk Format.

Empty Clusters

Write a File

File 1
4 Clusters Long

Write a File

File 2
3 Clusters Long

Write a File

File 3
5 Clusters Long

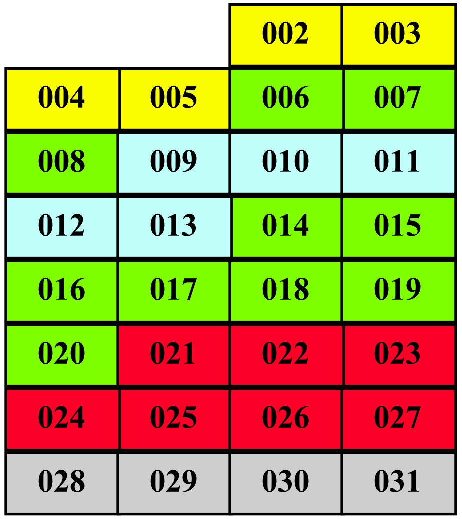
Delete File 2

Write a File

File 4
10 Clusters Long

Write a File

File 5
7 Clusters Long



		002	003
004	005	006	007
008	009	010	011
012	013	014	015
016	017	018	019
020	021	022	023
024	025	026	027
028	029	030	031

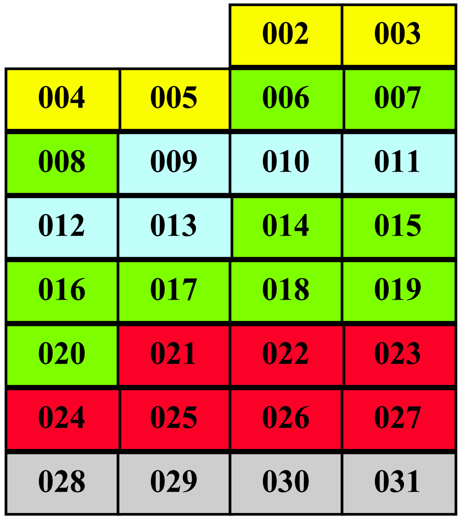
File Fragmentation (First Part of Disk Only).

Disk Format.

Empty Clusters

File 3
5 Clusters Long

Modify File 3



		002	003
004	005	006	007
008	009	010	011
012	013	014	015
016	017	018	019
020	021	022	023
024	025	026	027
028	029	030	031

File Fragmentation (First Part of Disk Only).

Disk Format.

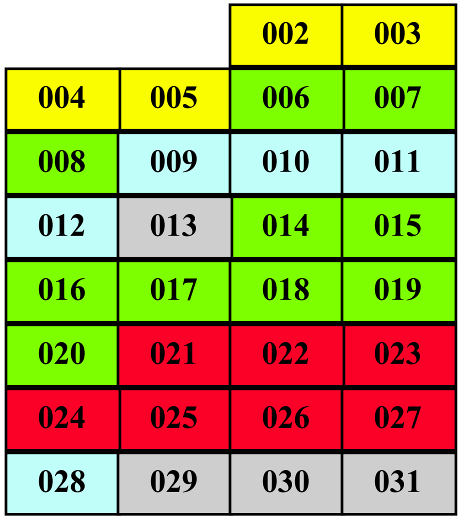
Empty Clusters

File 3
5 Clusters Long

File 6
3 Clusters Long

Modify File 3

Write a File



		002	003
004	005	006	007
008	009	010	011
012	013	014	015
016	017	018	019
020	021	022	023
024	025	026	027
028	029	030	031

File Fragmentation (First Part of Disk Only).

Disk Format.

⬜ Empty Clusters

🟦 File 3
5 Clusters Long

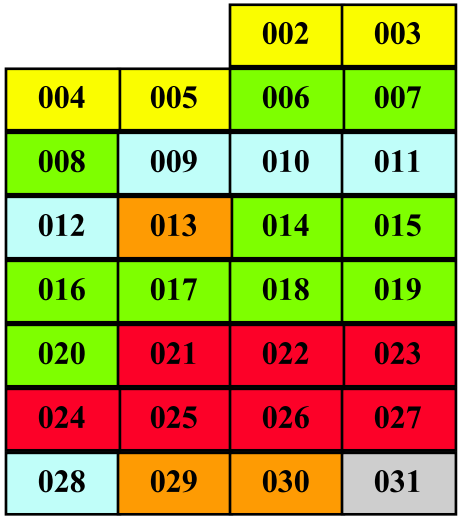
🟧 File 6
3 Clusters Long

🟦 File 3
5 Clusters Long

Modify File 3

Write a File

Modify File 3



		🟨 002	🟨 003
🟨 004	🟨 005	🟩 006	🟩 007
🟩 008	🟦 009	🟦 010	🟦 011
🟦 012	🟧 013	🟩 014	🟩 015
🟩 016	🟩 017	🟩 018	🟩 019
🟩 020	🟥 021	🟥 022	🟥 023
🟥 024	🟥 025	🟥 026	🟥 027
🟦 028	🟧 029	🟧 030	⬜ 031

File Fragmentation (First Part of Disk Only).

Disk Format.

Empty Clusters

File 3
5 Clusters Long

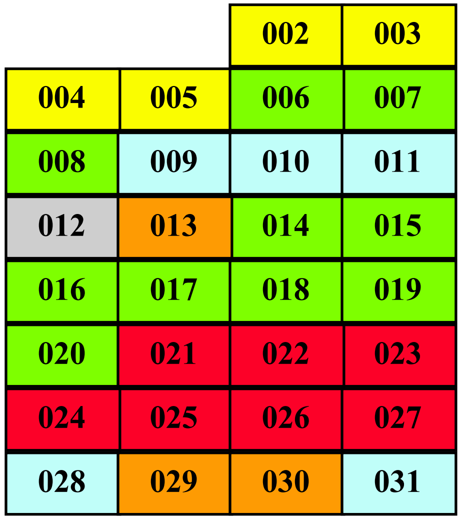
File 6
3 Clusters Long

File 3
5 Clusters Long

Modify File 3

Write a File

Modify File 3



		002	003
004	005	006	007
008	009	010	011
012	013	014	015
016	017	018	019
020	021	022	023
024	025	026	027
028	029	030	031

File Fragmentation (First Part of Disk Only).

Disk Format.

Empty Clusters

File 3
5 Clusters Long

File 6
3 Clusters Long

File 3
5 Clusters Long

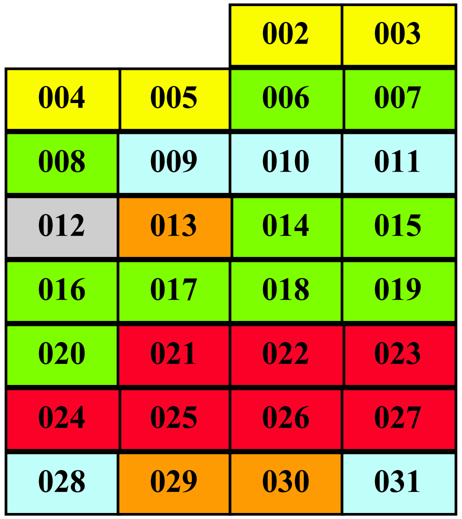
File 1
4 Clusters Long

Modify File 3

Write a File

Modify File 3

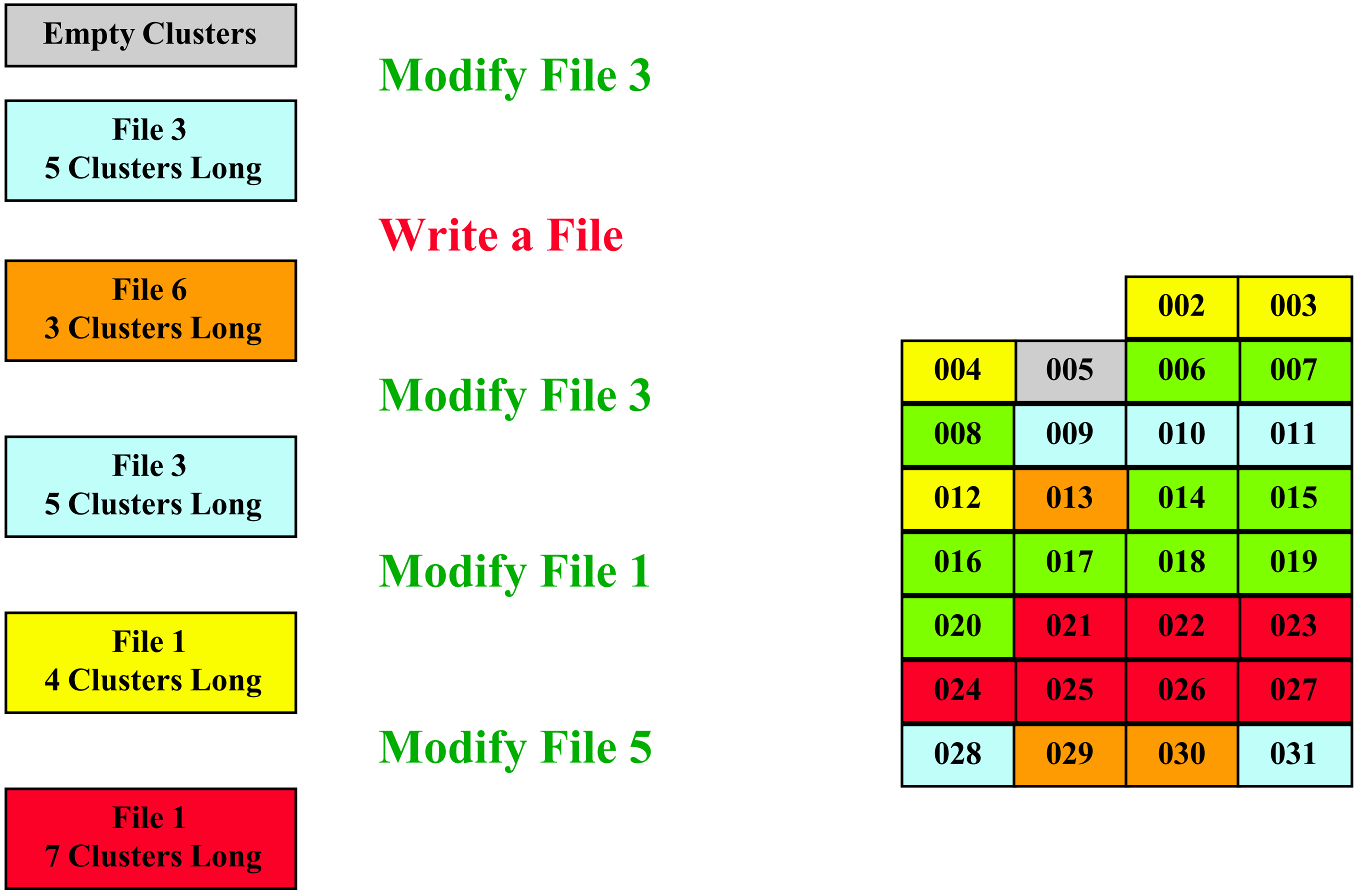
Modify File 1



		002	003
004	005	006	007
008	009	010	011
012	013	014	015
016	017	018	019
020	021	022	023
024	025	026	027
028	029	030	031

File Fragmentation (First Part of Disk Only).

Disk Format.



Empty Clusters

File 3
5 Clusters Long

File 6
3 Clusters Long

File 3
5 Clusters Long

File 1
4 Clusters Long

File 1
7 Clusters Long

Modify File 3

Write a File

Modify File 3

Modify File 1

Modify File 5

		002	003
004	005	006	007
008	009	010	011
012	013	014	015
016	017	018	019
020	021	022	023
024	025	026	027
028	029	030	031

File Fragmentation (First Part of Disk Only).

Disk Format.

Empty Clusters

File 3
5 Clusters Long

File 6
3 Clusters Long

File 3
5 Clusters Long

File 1
4 Clusters Long

File 1
7 Clusters Long

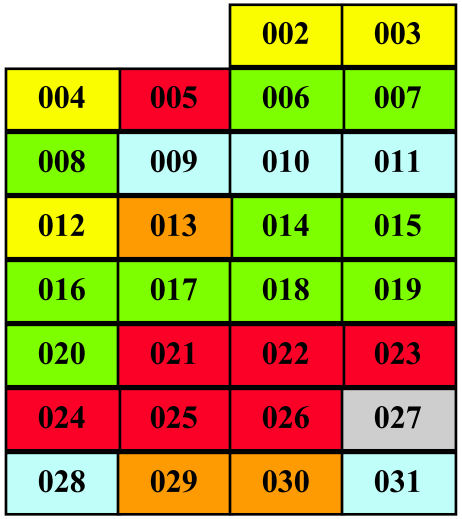
Modify File 3

Write a File

Modify File 3

Modify File 1

Modify File 5



		002	003
004	005	006	007
008	009	010	011
012	013	014	015
016	017	018	019
020	021	022	023
024	025	026	027
028	029	030	031

File Fragmentation (First Part of Disk Only).

Disk Format.

The Disk is getting rather messed up

so :-

What do we do?

Disk Format.

Empty Clusters

File 3
5 Clusters Long

File 6
3 Clusters Long

File 3
5 Clusters Long

File 1
4 Clusters Long

File 1
7 Clusters Long

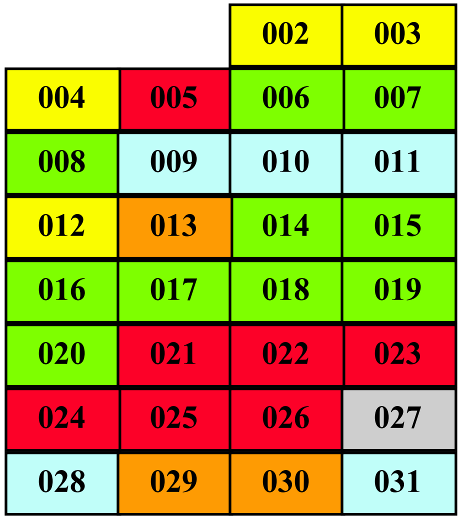
Modify File 3

Write a File

Modify File 3

Modify File 1

Modify File 5



		002	003
004	005	006	007
008	009	010	011
012	013	014	015
016	017	018	019
020	021	022	023
024	025	026	027
028	029	030	031

Disk now needs De-Fragmenting

File Fragmentation (First Part of Disk Only).

Disk Format.

Empty Clusters

File 1
4 Clusters Long

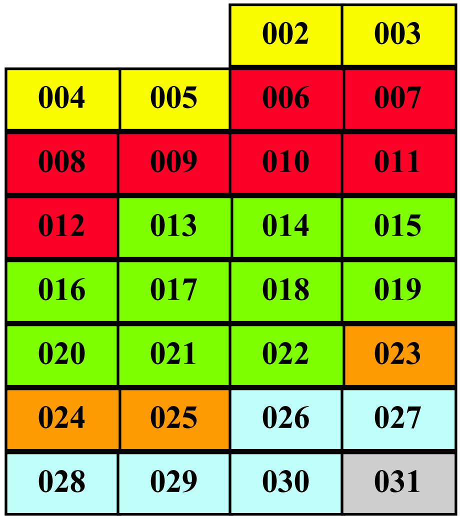
File 5
7 Clusters Long

File 4
10 Clusters Long

File 6
3 Clusters Long

File 3
5 Clusters Long

Note: You need some Free space on the Disk to Defragment it.



		002	003
004	005	006	007
008	009	010	011
012	013	014	015
016	017	018	019
020	021	022	023
024	025	026	027
028	029	030	031

Disk now De-Fragmented

File Fragmentation (First Part of Disk Only).

Disk Format.

- Why do we need to Defragment:
- As the files get more fragmented it takes longer to locate the clusters of the files.
  - Less time needed to find clusters.
- It takes time to move the heads as well as needing to wait for the disk to rotate so that correct data is under the heads.
  - This gives Faster access to contiguous data.

Disk Format.

- Why do we need a programme like Scandisk:
- As you have seen after a while of file activity the disk has had a lot of changes.
- When reading or writing to the disk if the data transfers should get corrupted the structure of the disk could be effected.
- Scandisk checks that the structure of the disk is valid and corrects any errors that may of occurred.

Disk Format.

- Why do we need to Archive Disks :
- All Disk systems involve mechanical components and hence they will eventually fail. (Guaranteed)
- All Semi-conductor memory components are subject to loss of storage charge or sensitive to electromagnetic radiation. (Guaranteed Data Loss)
- Therefore all storage systems are likely involve data lose at sometime. (Guaranteed to Fail)
- Regular Archiving is the only way to protect your valuable data.

Protect and Survive

(It is **YOUR** Responsibility to Protect **your** Data)

Data Storage.

Data Storage.

- Some basic Concepts :-
- Storage of the data on the media should use the available space as **Efficiently** as possible
  - (Do **NOT** waste space on the media).
- Use of the data stored on the media needs to as **Effective** as possible so you can :-
  - Locate the data quickly.
  - Locate the data accurately.
  - Locate data with minimum of effort.
  - Access or modify data with minimum of effort.

Definitions

Data Storage.

- Some basic Definitions :-
- A Byte or Character:
  - The smallest unit of data a system deals with.
- A Field:
  - A collection of related bytes or characters.
- A Record:
  - A collection of related fields.
- A File:
  - A collection of related records.
- A Directory or Folder:
  - A collection of related Files.

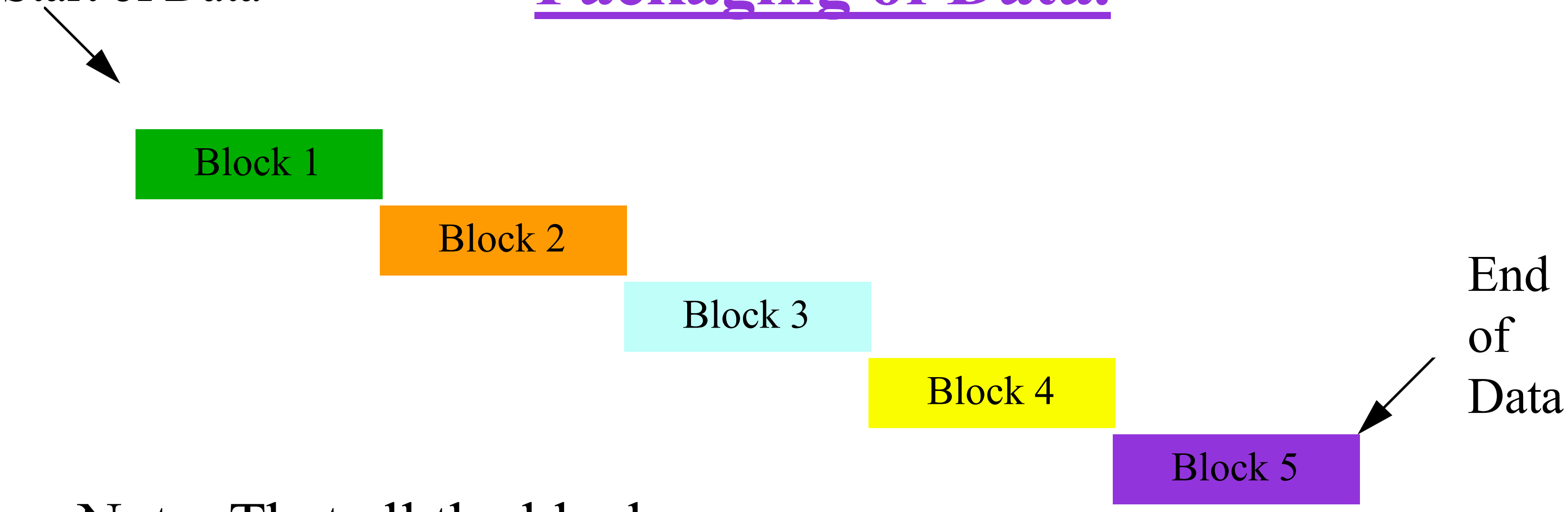
Data Storage.

How might we package our data?

Data Storage.

A **Block** of Data could be a Byte, a Field, a Record.

Packaging of Data.



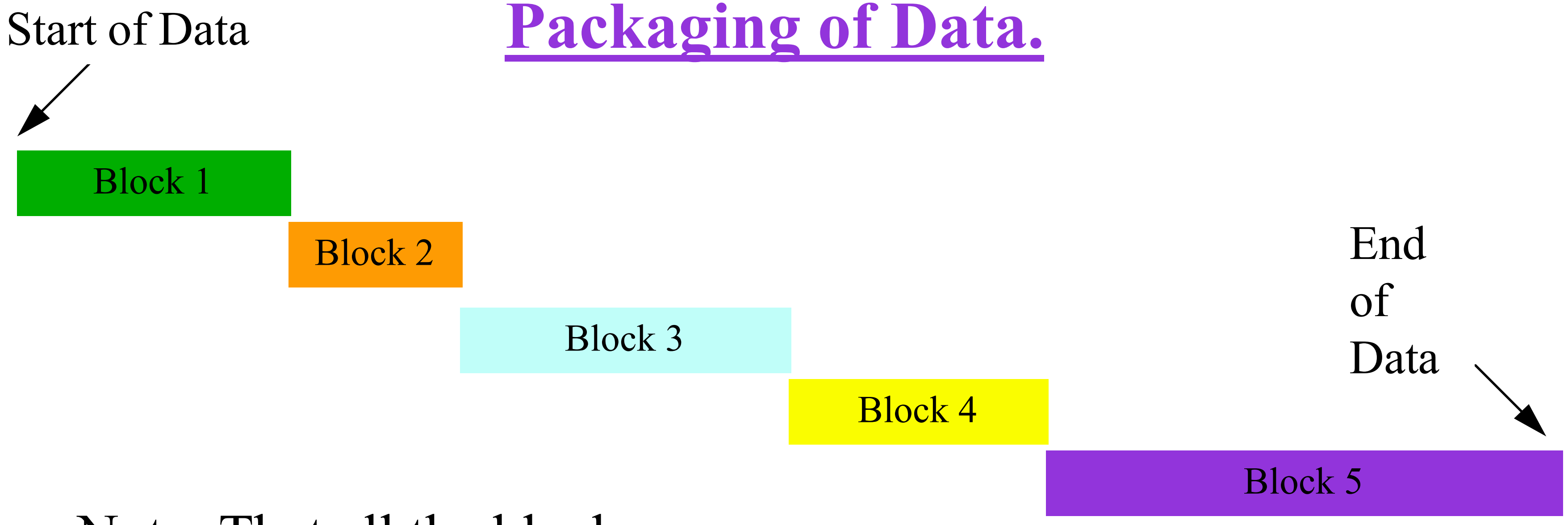
Note: That all the blocks are of the same size.

FIXED LENGTH Data blocks.

Data Storage.

A **Block** of Data could be a Byte, a Field, a Record.

Packaging of Data.



Start of Data

Block 1

Block 2

Block 3

Block 4

Block 5

End of Data

Note: That all the blocks could be a different size.

VARIABLE LENGTH Data blocks.

Definitions

Data Storage.

- Some basic Concepts :-
- You have two basic methods of packaging data.
  - **Fixed** Length blocks of data.
  - **Variable** Length blocks of data.

Data Storage.

How might we store our Data?

Data Storage.

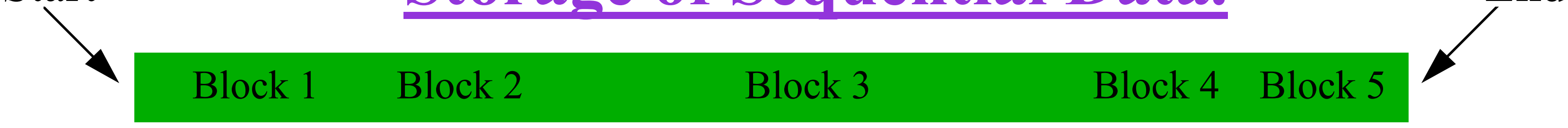
Sequential Storage

Like music on a Tape.

Data Storage.

A **Block** of Data could be a Byte, a Field, a Record.

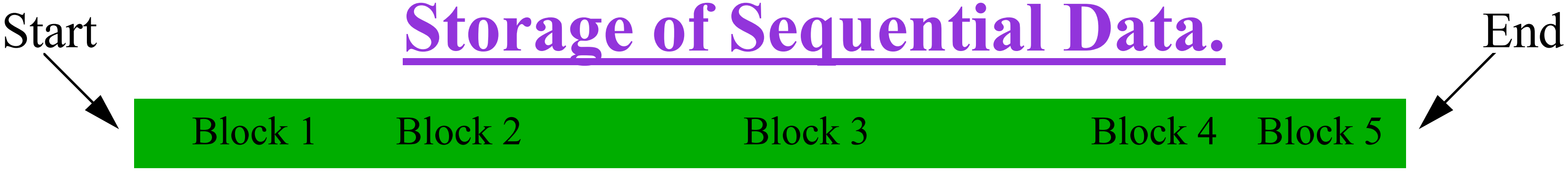
Storage of Sequential Data.



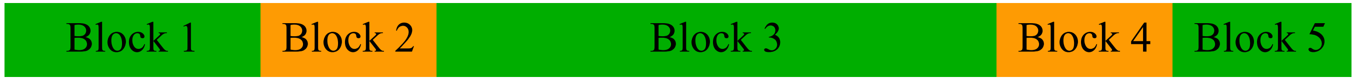
Data Storage.

A **Block** of Data could be a Byte, a Field, a Record.

Storage of Sequential Data.



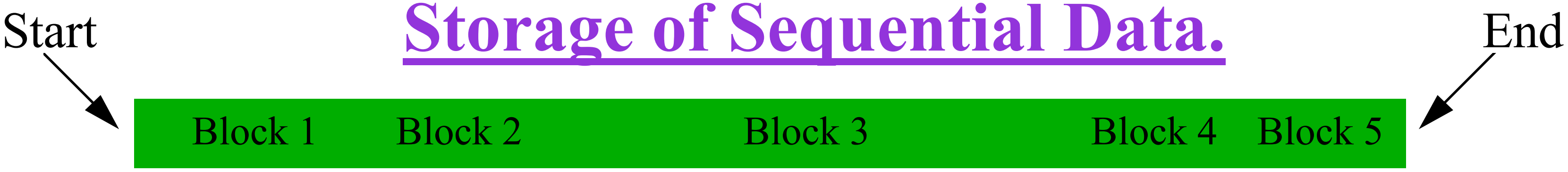
Data: How do you know where one block starts and the previous block stops?



Data Storage.

A **Block** of Data could be a Byte, a Field, a Record.

Storage of Sequential Data.

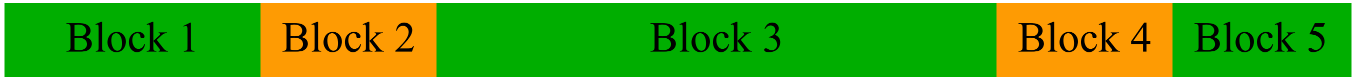


Start

End

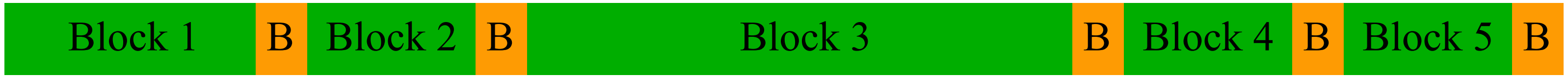
Block 1 Block 2 Block 3 Block 4 Block 5

Data: How do you know where one block starts and the previous block stops?



Block 1 Block 2 Block 3 Block 4 Block 5

You could place a marker between each block showing where it ends.

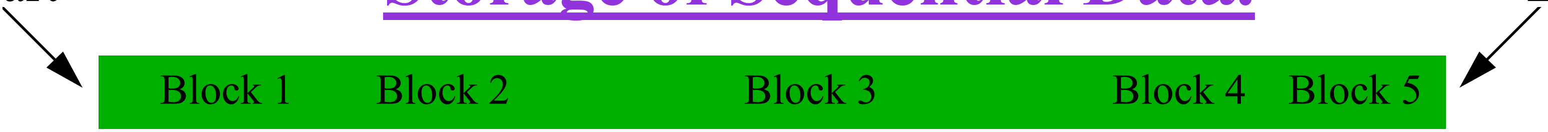


Block 1 B Block 2 B Block 3 B Block 4 B Block 5 B

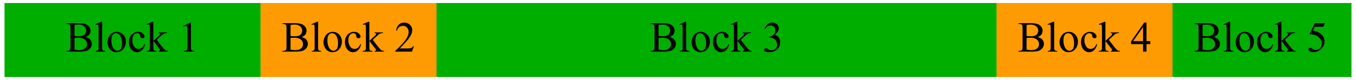
Data Storage.

A **Block** of Data could be a Byte, a Field, a Record.

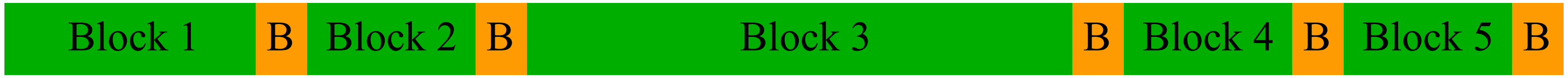
Storage of Sequential Data.



Data: How do you know where one block starts and the previous block stops?



You could place a marker between each block showing where it ends.



Files: How do you know where one file starts and the previous file stops?

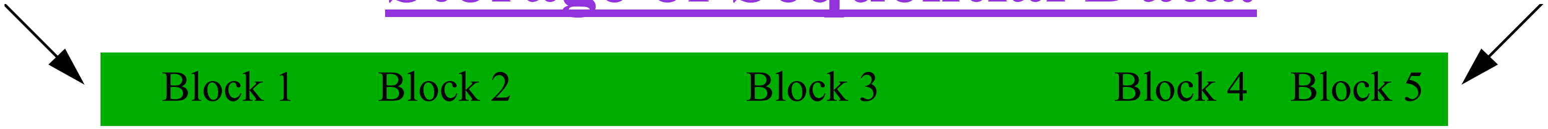
Data Storage.

A **Block** of Data could be a Byte, a Field, a Record.

Storage of Sequential Data.

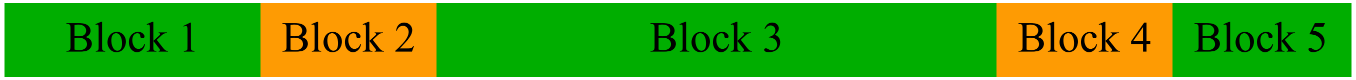
Start

End



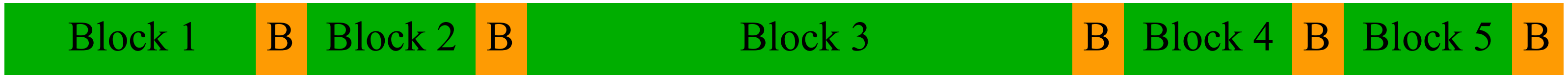
Block 1 Block 2 Block 3 Block 4 Block 5

Data: How do you know where one block starts and the previous block stops?



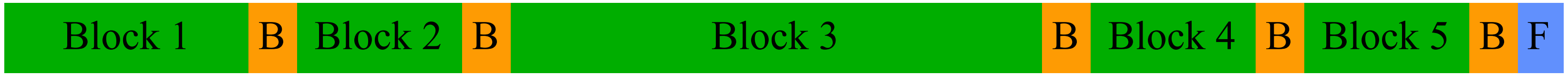
Block 1 | Block 2 | Block 3 | Block 4 | Block 5

You could place a marker between each block showing where it ends.



Block 1 | B | Block 2 | B | Block 3 | B | Block 4 | B | Block 5 | B

Files: How do you know where one file starts and the previous file stops?



Block 1 | B | Block 2 | B | Block 3 | B | Block 4 | B | Block 5 | B | F

You could place a marker between each File showing where it ends (EOF).

Data Storage.

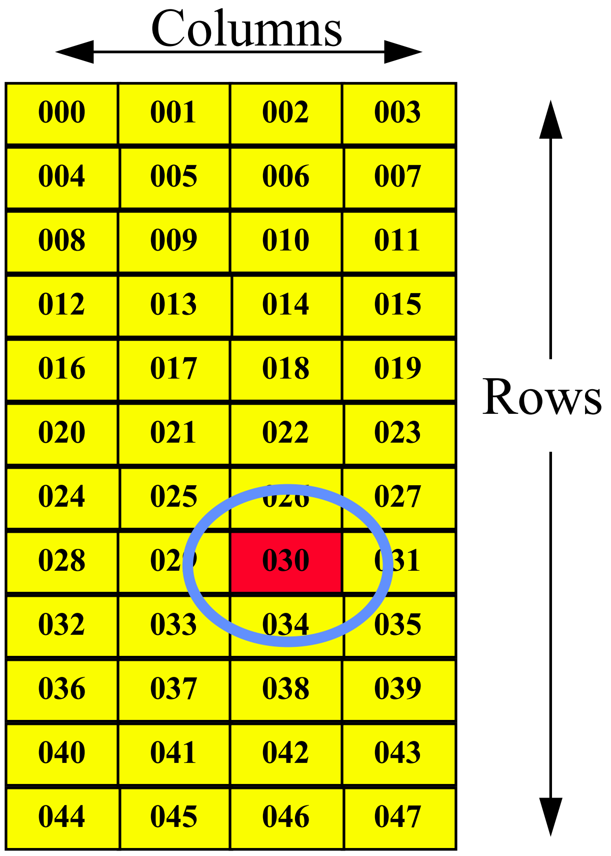
Direct or Random Storage

Like letters being delivered to a block of Flats.

Data Storage.

Storage of Direct Access Data.

- Every Area or Cell in a Direct Access storage system is of a fixed size.
- Each entity in the storage area is given an address to identify that Area or Cell.
- We could describe the **Circled cell** as (Column 3 , Row 8) or in this case as cell (030).

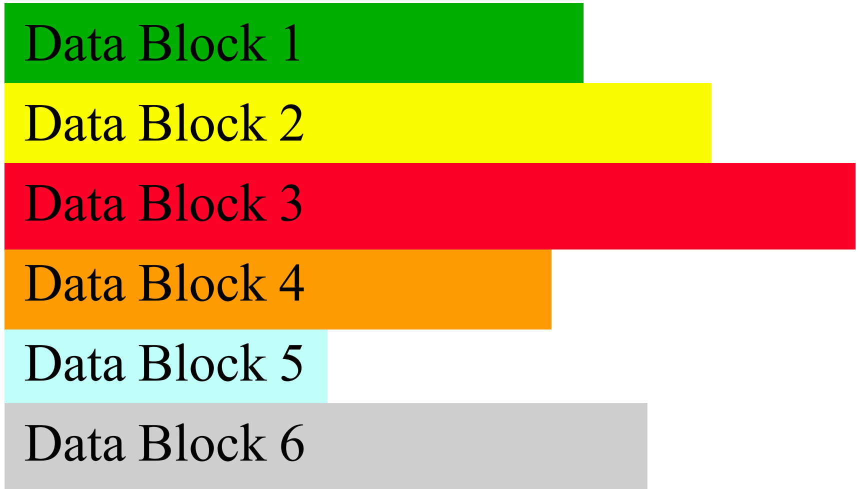


000	001	002	003
004	005	006	007
008	009	010	011
012	013	014	015
016	017	018	019
020	021	022	023
024	025	026	027
028	029	🟥 030	031
032	033	034	035
036	037	038	039
040	041	042	043
044	045	046	047

Data Storage.

Storage of Direct Access Data.

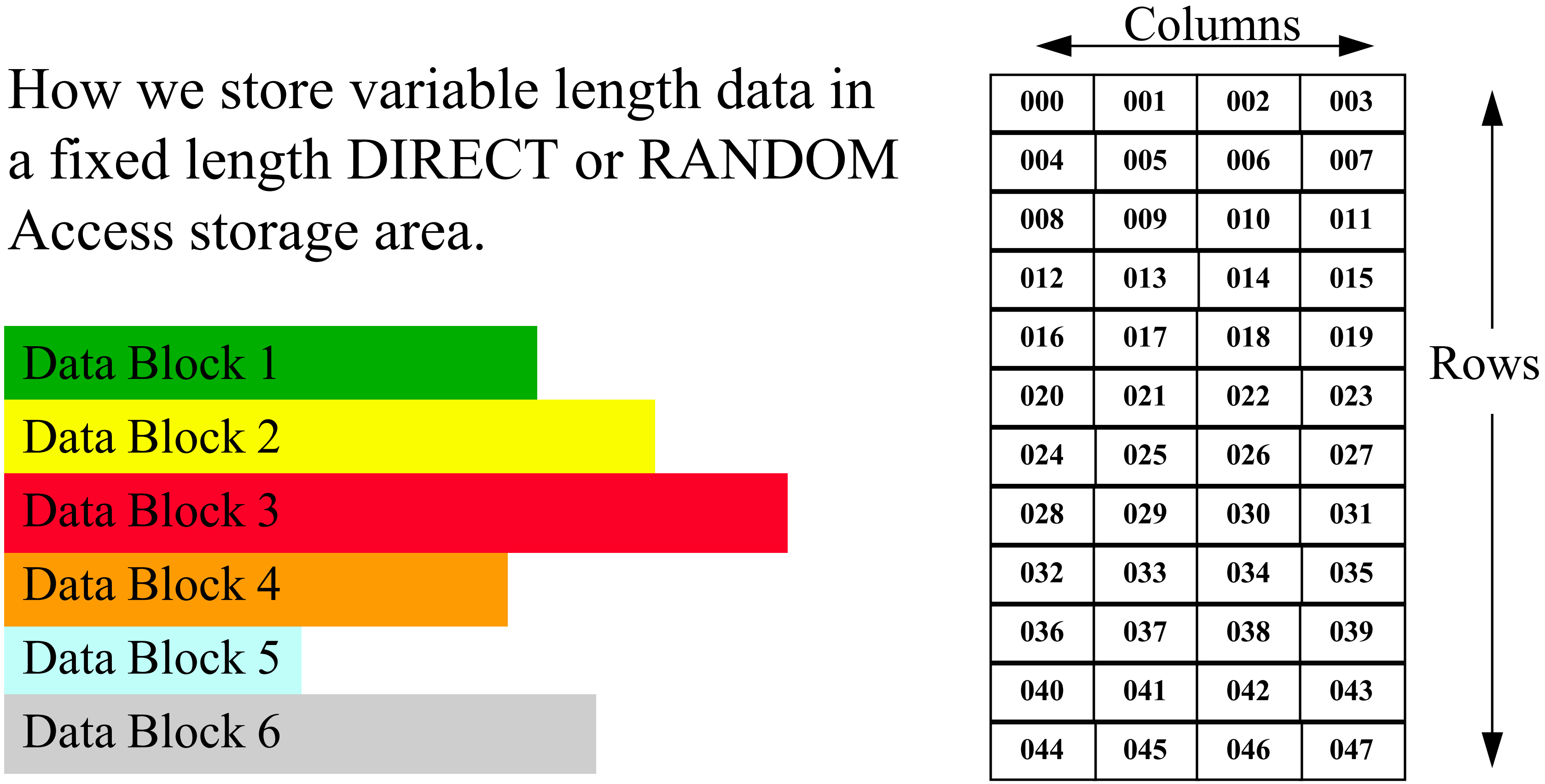
How we store variable length data in a fixed length DIRECT or RANDOM Access storage area.



Data Storage.

Storage of Direct Access Data.

How we store variable length data in a fixed length DIRECT or RANDOM Access storage area.



Data Block 1

Data Block 2

Data Block 3

Data Block 4

Data Block 5

Data Block 6

Columns

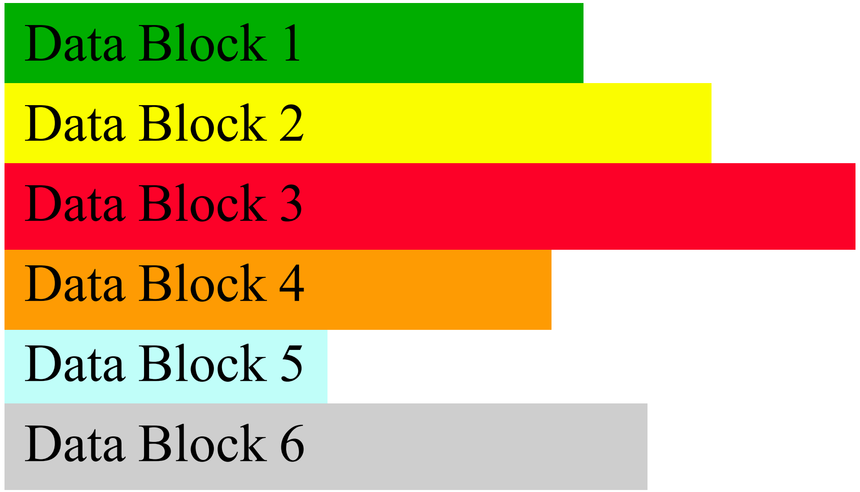
000	001	002	003
004	005	006	007
008	009	010	011
012	013	014	015
016	017	018	019
020	021	022	023
024	025	026	027
028	029	030	031
032	033	034	035
036	037	038	039
040	041	042	043
044	045	046	047

Rows

Data Storage.

Storage of Direct Access Data.

How we store variable length data in a fixed length DIRECT or RANDOM Access storage area.



Data Block 1

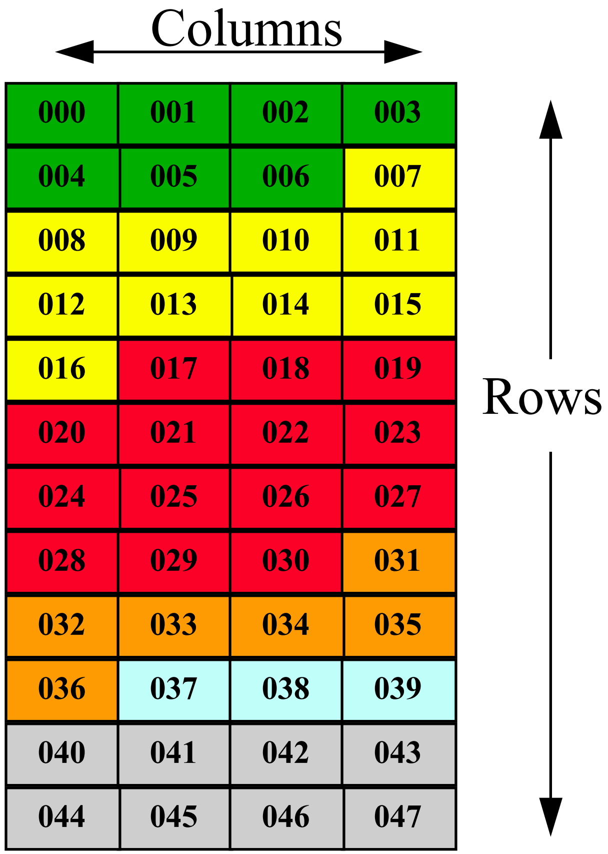
Data Block 2

Data Block 3

Data Block 4

Data Block 5

Data Block 6



Columns

Rows

000	001	002	003
004	005	006	007
008	009	010	011
012	013	014	015
016	017	018	019
020	021	022	023
024	025	026	027
028	029	030	031
032	033	034	035
036	037	038	039
040	041	042	043
044	045	046	047

Definitions

Data Storage.

- Some basic Concepts :-
- You have two basic methods of packaging data.
  - **Fixed** Length blocks of data.
  - **Variable** Length blocks of data.
- You have two basic methods of storing or accessing data.
  - Serial or **Sequential** Access.
  - Direct or **Random** Access.
- You can however sometimes use a combination of both methods.

Data Storage Requirements.

Data Storage Requirements.

- In computing to make number handling easier we use the metric multiplier K, M, G etc however they have a slightly different value from the normal values.
- 1K in the metric system = 1000 however
  - in computing 1K = 2^{10} = 1024
- 1M in the metric system = 1,000,000 however
  - in computing 1M = 2^{20} = 1,048,576
  - however it may also be described as 1M = 1,024,000
  - or sometimes it just means the metric value
  - you may need to read small print to clarify.

Data Storage Requirements.

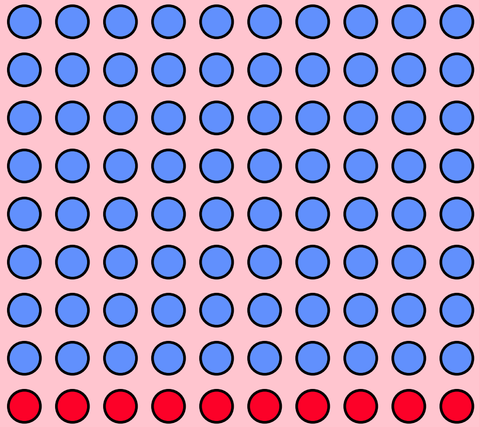
- 1G in the metric system = 1,000,000,000 however
  - in computing 1M = 2^{30} = 1,073,741,824
  - or sometimes it just means the metric value
  - again you may need to read small print to clarify.
- 1T in the metric system = 1,000,000,000,000
  - in computing 1M = 2^{40} = 1G * 1024
  - again you may need to read small print to clarify size.
- Metric units K = Kilo M = Mega
  G = Giga T = Tera

Data Storage Requirements.

- A typical fixed format A4 page of text contains:-
  - 66 lines of text.
- Each line of text contains:-
  - 80 columns of characters.
- Therefore an A4 page would consist of:-
  - 66 * 80 = 5,280 printed characters.
- Each character requires 8 bits of data or one Byte of storage therefore an A4 page needs 5,280 bytes of storage space or just over 5K bytes

Data Storage Requirements.

- If instead of using fixed printer font characters we used graphics image characters then the amount of data that is needed to be stored would increase significantly. A typical low quality 9 by 10 dot matrix character would be built as follows:-



The **Blue Dots** are where the character is formed and displayed

The **Red Dots** are used for underline etc.

Data Storage Requirements.

- An A4 page using a low density dot matrix format would require 10 bytes of storage per character therefore with:-
  - 66 lines of text.
  - 80 columns of characters.
  - 10 bytes storage per character
- Therefore an A4 page would consist of:-
  - 66 * 80 * 10 = 52,800 bytes of storage.
- However this format allows significant choice of Fonts and limited graphics.

Data Storage Requirements.

- The next generation printers used a 16 pin dot matrix and the basic matrix was 16 by 16 dots.
- An A4 page using this higher density dot matrix format would require 32 bytes of storage per character therefore with:-
  - 66 lines of text.
  - 80 columns of characters.
  - 32 bytes storage per character
- Therefore an A4 page would consist of:-
  - 66 * 80 * 32 = 168,960 bytes of storage.

Data Storage Requirements.

- The next stage was to define both printing and scanning using the measurement Dots per Inch or DPI.
- A monochrome image (just black and white) with a density of 100 DPI an A4 page would require the following amount of storage space:-
  - Page length = 11.66 inches long.
  - Page width = 8.3 inches wide.
  - 8 bits of information per byte
- The storage needs of an A4 page would be:-
  - (11.66*100 * 8.3*100) / 8 = 1,209,725 bytes.

Data Storage Requirements.

- If we use 16 basic colour printing and scanning we need 4 bits of information per dot or pixel (Picture element).
- A minimal colour image with a density of 100 DPI an A4 page would require the following amount of storage space:-
  - Page length = 11.66 inches long.
  - Page width = 8.3 inches wide.
  - 2 pixels of information per byte
- The storage needs of an A4 page would be:-
  - (11.66*100 * 8.3*100) / 2 = 4,838,900 bytes.

Data Storage Requirements.

- If we use 256 colour printing and scanning we need 8 bits of information per dot or pixel (Picture element).
- A medium colour image with a density of 100 DPI an A4 page would require the following amount of storage space:-
  - Page length = 11.66 inches long.
  - Page width = 8.3 inches wide.
  - 1 pixels of information per byte
- The storage needs of an A4 page would be:-
  - (11.66*100 * 8.3*100) / 1 = 9,677,800 bytes.

Data Storage Requirements.

- Current colour printing and scanning we need 24 or 40 bits of information per dot or pixel (Picture element).
- A colour image with a density of 1200 DPI an A4 page would require the following amount of storage space:-
  - Page length = 11.66 inches long.
  - Page width = 8.3 inches wide.
  - 24 bits is 3 bytes of information per pixel.
- The storage needs of an A4 page would be:-
  - (11.66*1200 * 8.3*1200) * 3 = 418 M bytes.

Data Storage Requirements.

- As current ICT and graphics requirements are for higher quality images keeping data in its raw form becomes a significant problem.
- The solution is often the use data compression.
- Some of the popular graphics compression formats are:-
  - JPEG , GIF , TIFF
- However some compression techniques tend to loose information as compression ratios rise.
- Note BMP is usually a raw graphics data format.

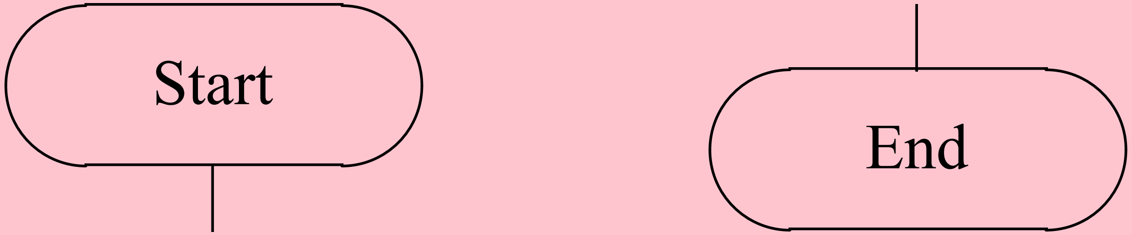
Flow Charts

Flow Charts.

- Flow charts are used to give a pictorial representation of how a system operates.
- The flow charts use a limited number of symbols that are fairly intuitive in their use.
- The Main symbols are :-
  - The Start and End Symbols.
  - The Process block symbol.
  - The Decision block symbol.
- Additional or Optional symbols :-
  - Input / Output block symbol.
  - On and Off Page Connectors.
  - Repeat or Iterative Block Symbols.

Flow Charts.

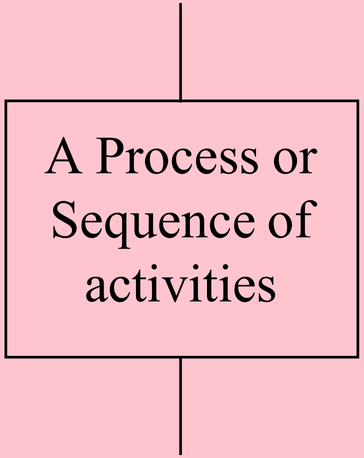
Start , End Symbols



Start

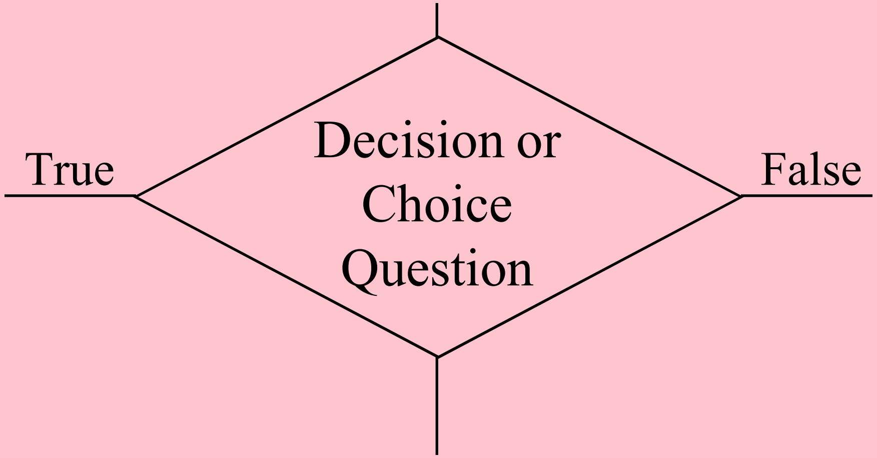
End

The Process Symbol



A Process or Sequence of activities

The Decision Symbol



True

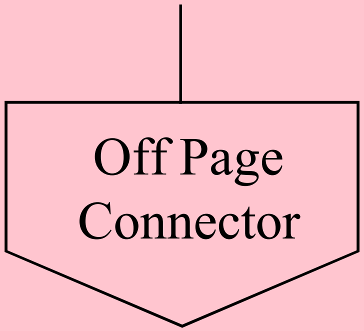
Decision or Choice Question

False

Other 2 Decision answers could also be YES/NO , Success/Error etc.
The 3 Decision answers are usually Higher , Same , Lower (FORTRAN)

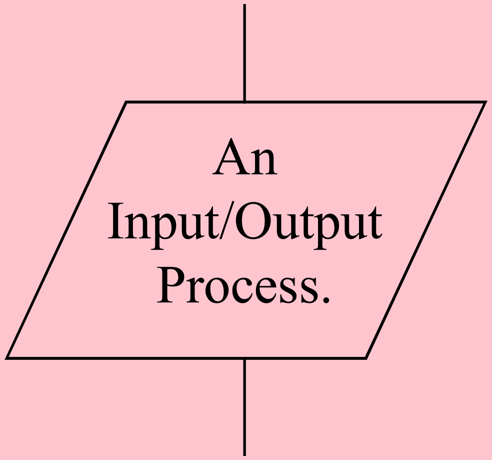
Flow Charts.

Off-Page Connector Symbols



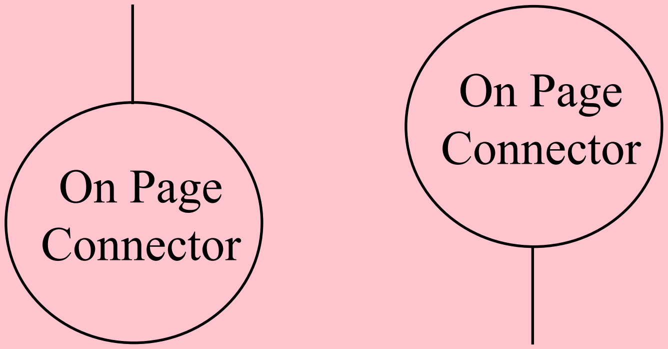
Off Page Connector

The Input/Output Process Symbol



An Input/Output Process.

On-Page Connector Symbols

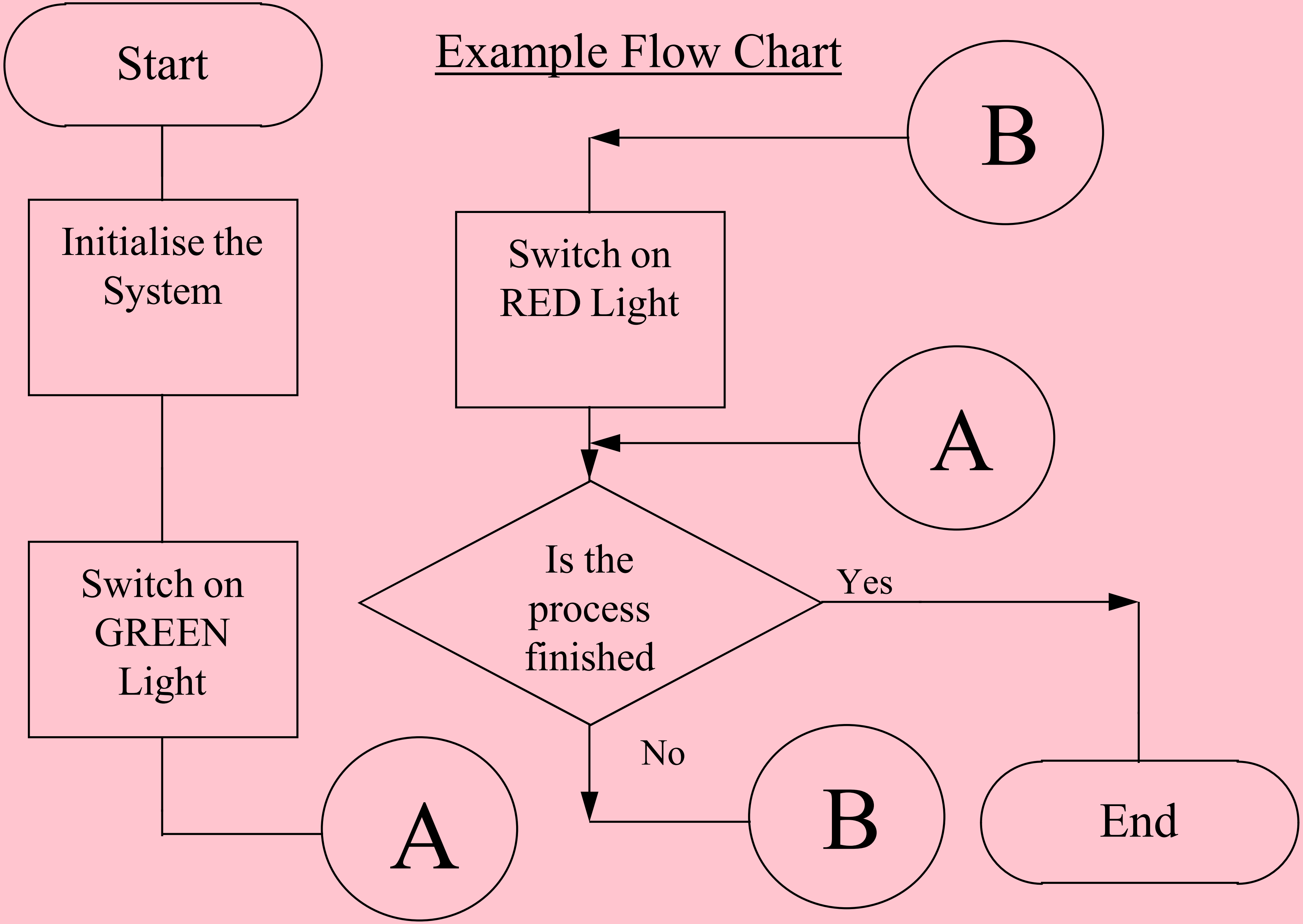


On Page Connector

On Page Connector

Flow Charts.

Example Flow Chart



QBASIC

QBASIC.

- What is **QBASIC** ?
- It is an advanced version of the computer language called **BASIC**
- **B**eginner **A**ll-purpose **S**ymbolic **I**nstruction **C**ode.
- The **Q** prefix indicated that it is a variant of the Quick BASIC computer language.
- So called because it is:-
  - – Quick to Learn.
  - – Quick to Write.
  - – Quick to test and debug.

Variables.

Variables.

- Place where information can be stored.
- A number of Variants
  - String Variables (Text stores)
  - Integers (Numbers NO decimal point)
  - Floating (Numbers with decimal point)
- Arrays (Groups of variables of the same type)
- User defined data types. (Structures developed from combinations of String, Integer or Floating)
- See STK_QUE (more detail on variables)

Programming

Programming.

- What is Programming ?
- A method of overcoming the deficiencies in a Hardware system.
- All programming activities can be subdivided into the following three basic constructs :-
  - Sequence.
  - Selection.
  - Iteration.

Programming.

Instruction

Instruction

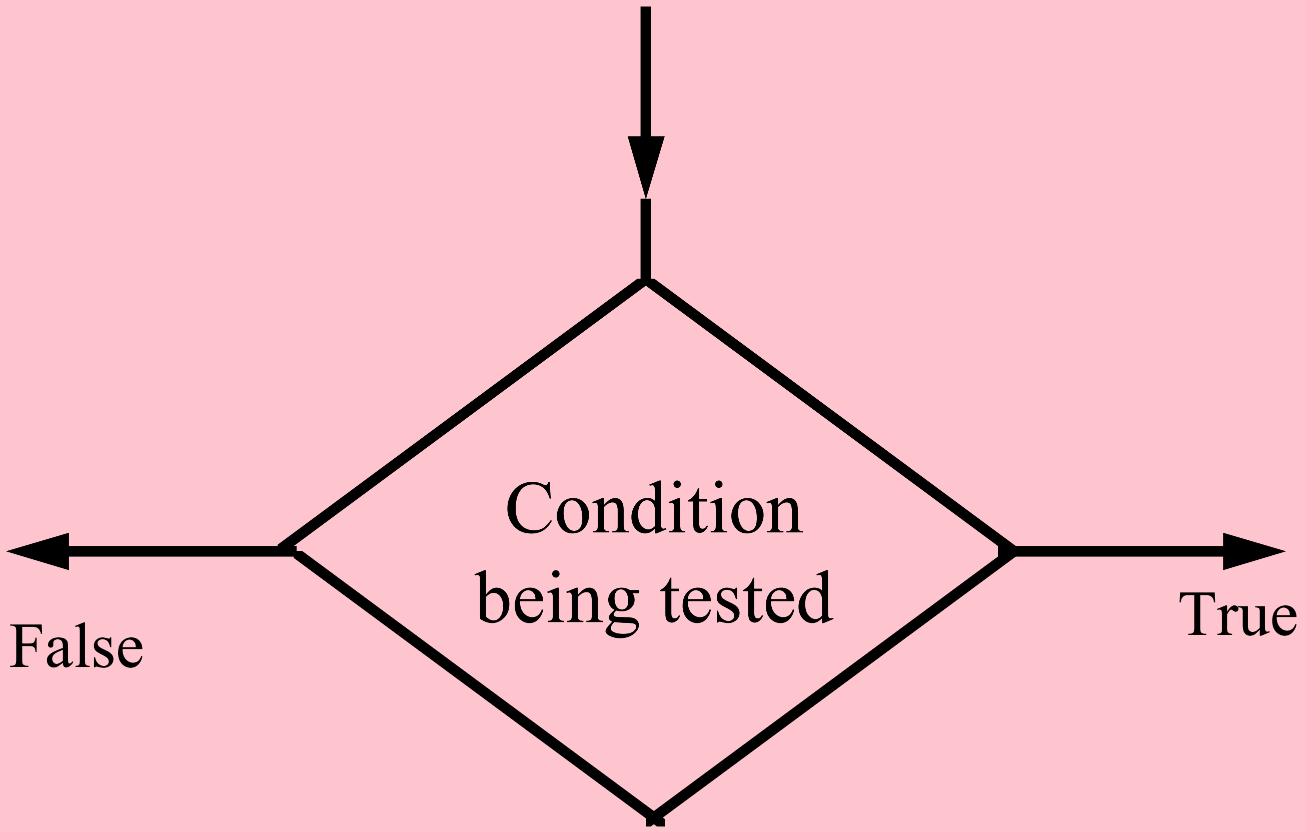
Instruction

. . . . .

Instruction

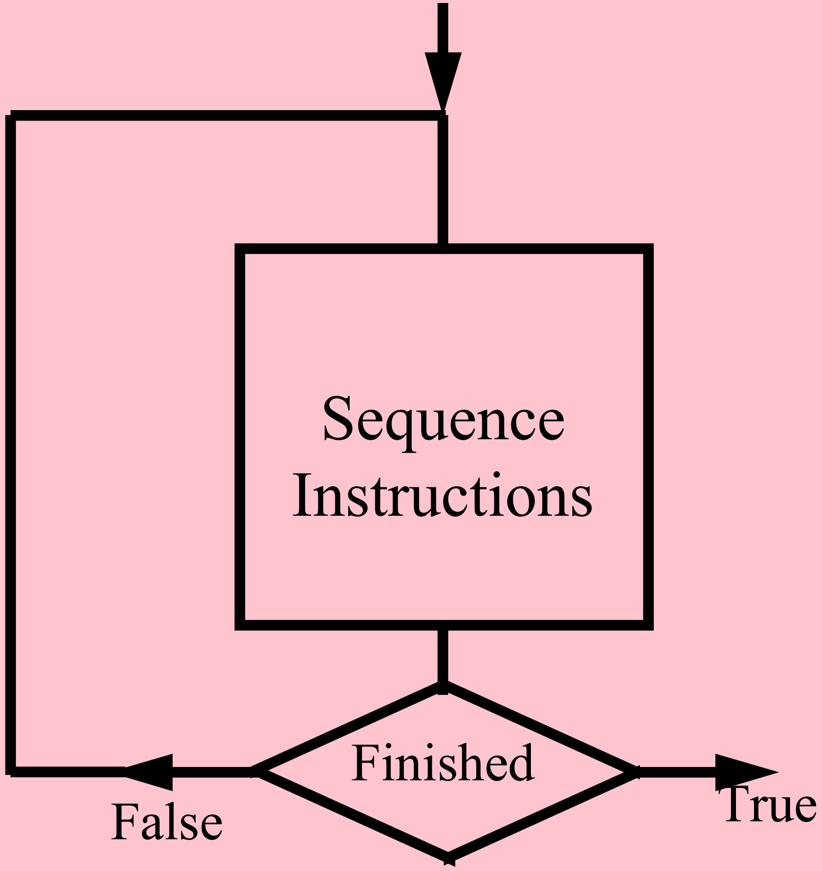
Sequence.

Programming.



Selection.

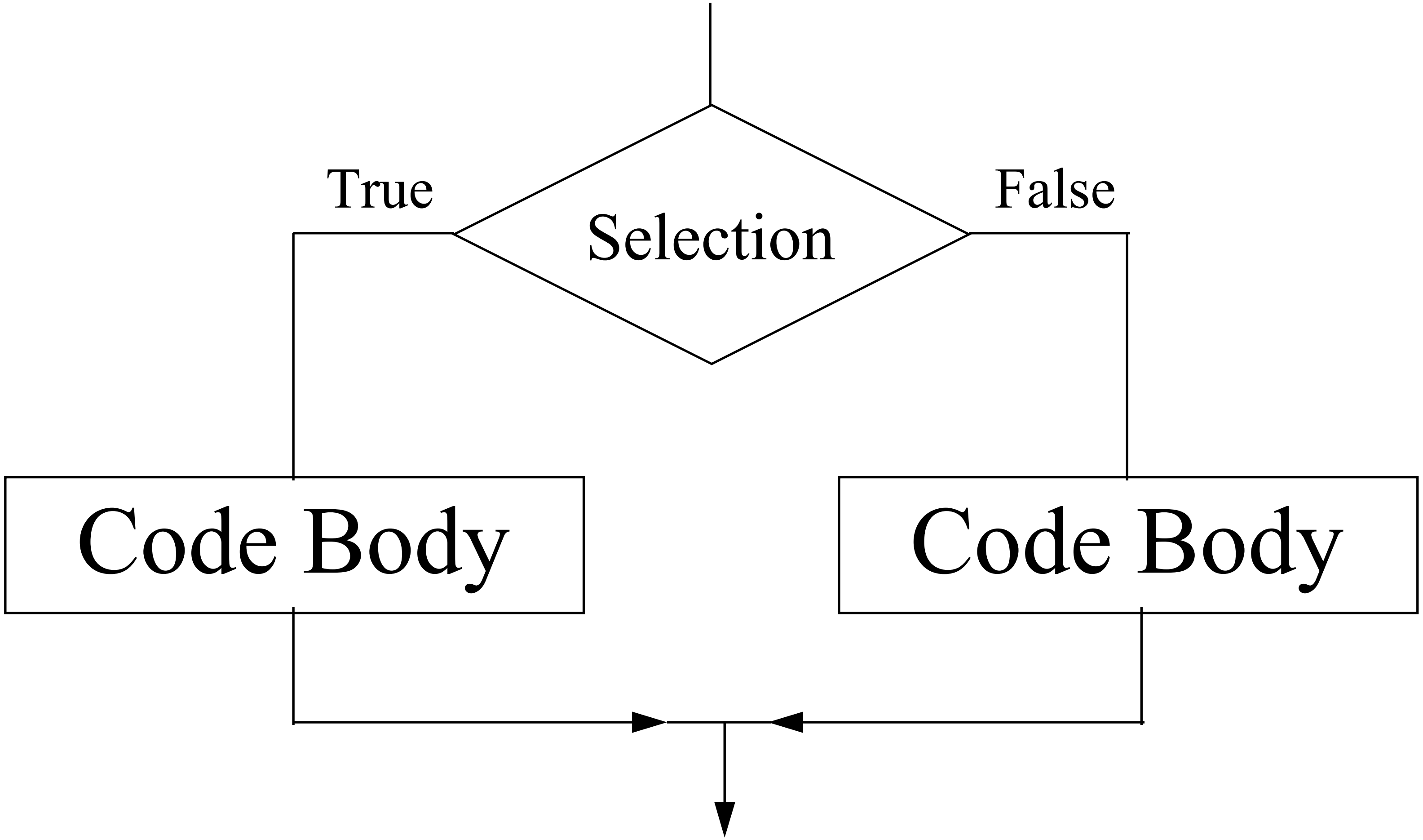
Programming.



Iteration created from Sequence & Selection.

Selection Statements.

Selection.



Selection.

```
IF <condition> THEN <Statement>
```

```
IF <condition> THEN <Statement> ELSE <Statement>
```

```
IF <condition> THEN
        <Statement Block>
ENDIF
```

```
IF <condition> THEN
        <True Statement Block>
ELSE
        <False Statement Block>
ENDIF
```

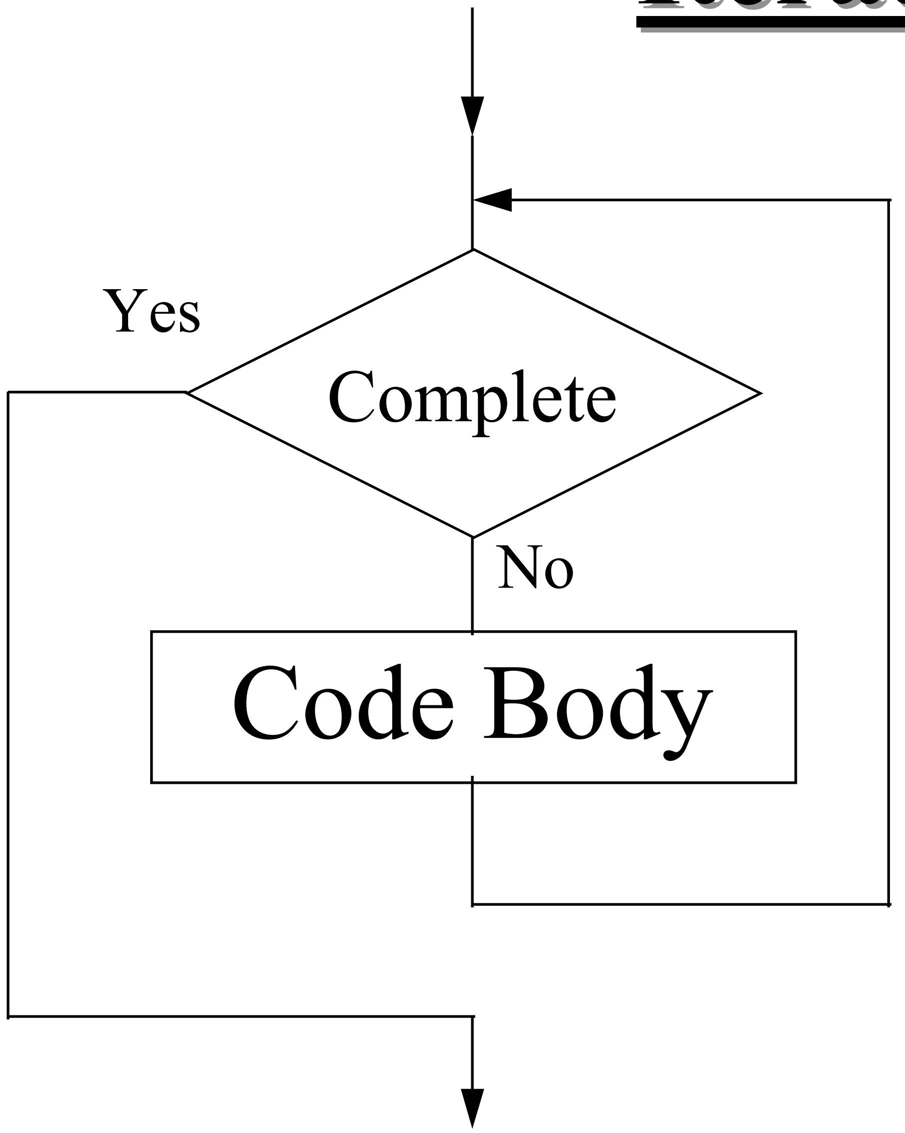
Selection.

```
SELECT CASE <test_expression>
        CASE <Expresion1>
                 <Statements>
        CASE <Expresion2>
                 <Statements>
        CASE ELSE
                <Statement>
END SELECT
```

This is an advanced form of the **IF** statement.

Iteration or Repetition Statements.

Iteration.



FOR is the Counting Statement

For (Assign) **To** (End) (**Step**)

Code Body

Next

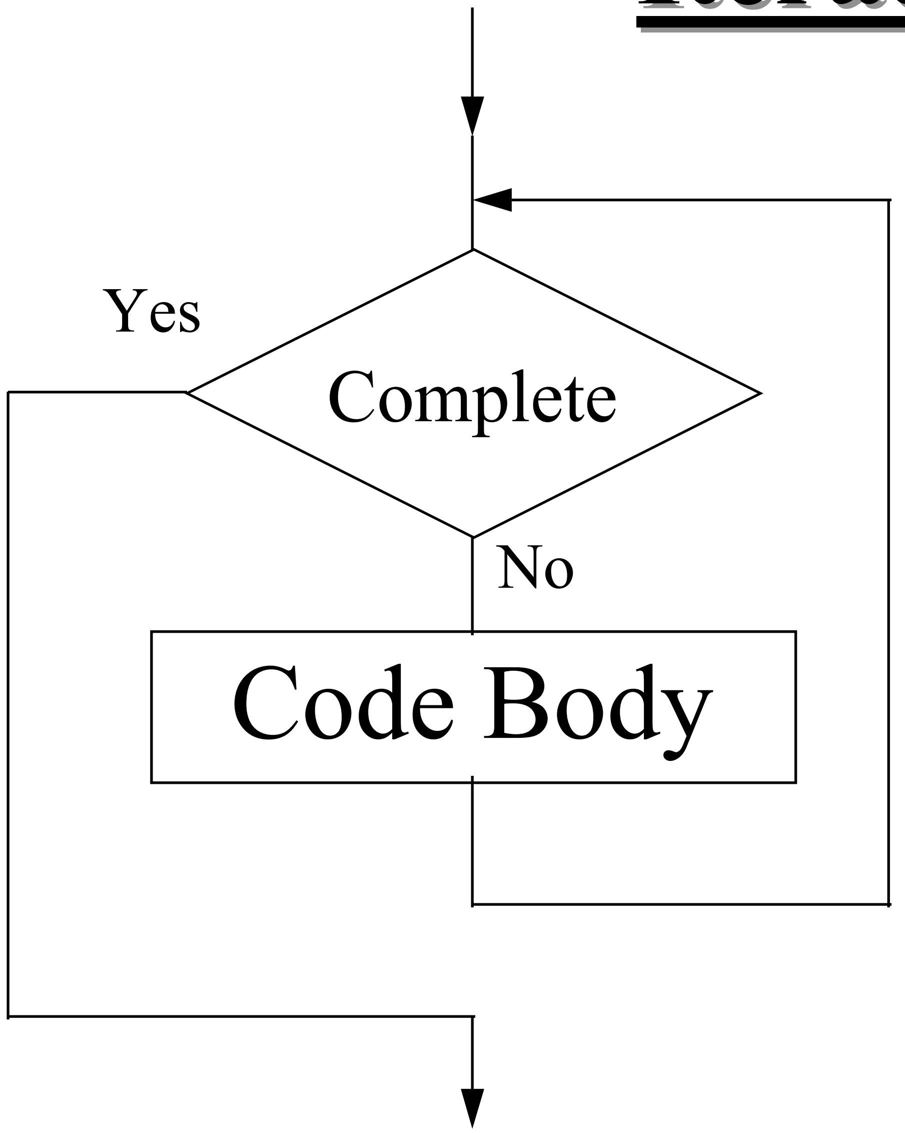
OR

While (Condition)

Code Body

Wend

Iteration.



Do Until

Code Body

Loop

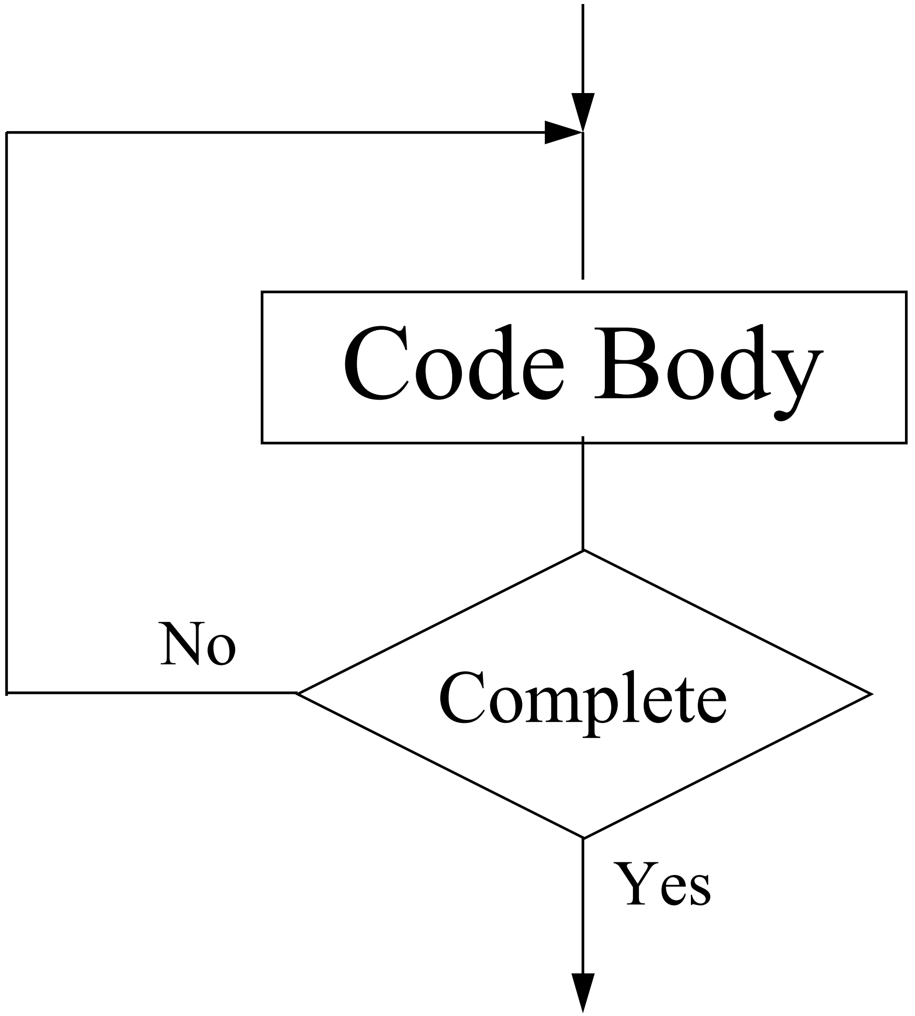
OR

Do While

Code Body

Loop

Iteration.



Code Body

No

Complete

Yes

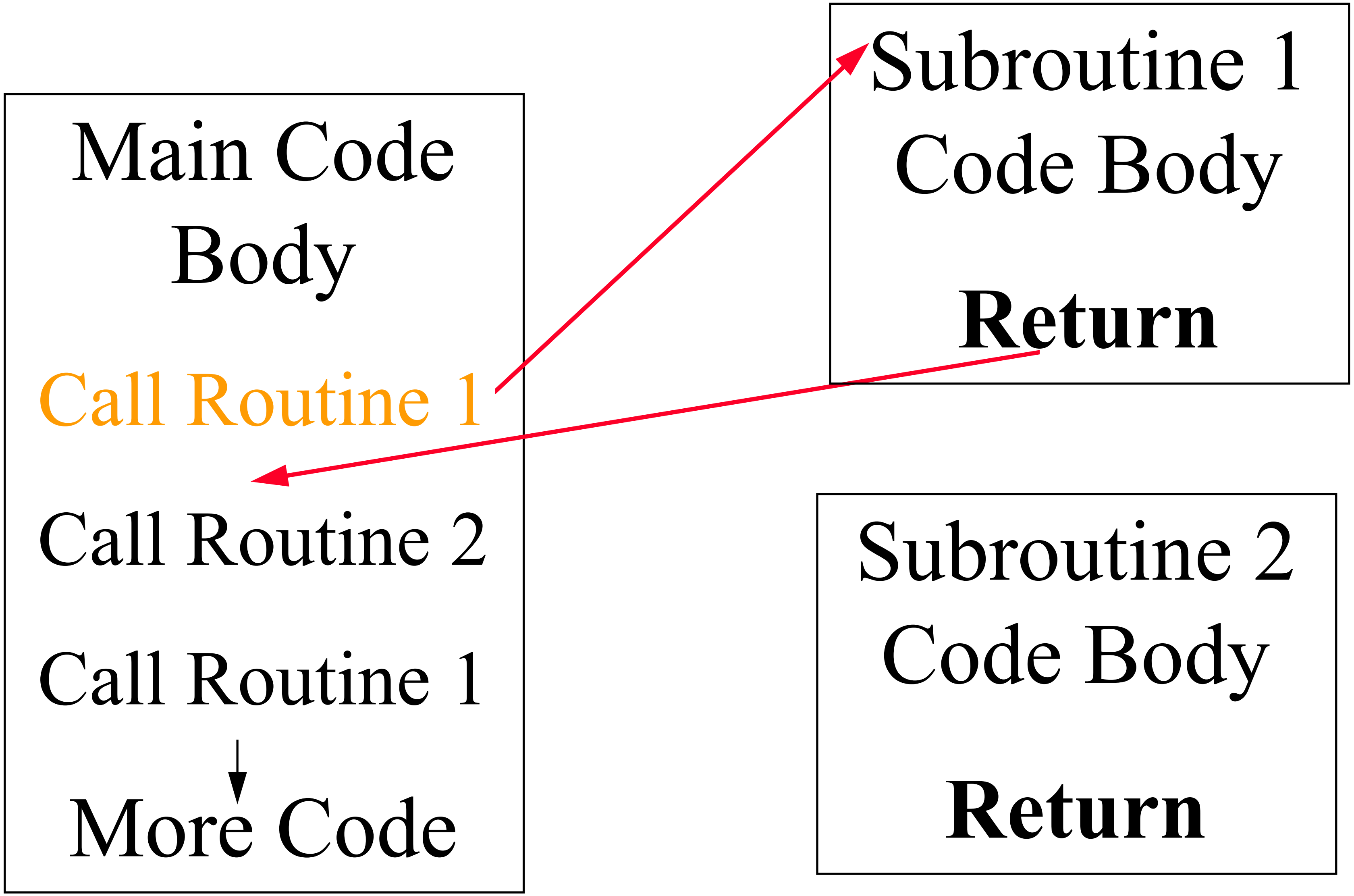
```
Do
    Code Body
Loop Until
```

OR

```
Do
    Code Body
Loop While
```

Subroutines and Functions.

Subroutines and Functions.



Main Code Body

Call Routine 1

Call Routine 2

Call Routine 1

More Code

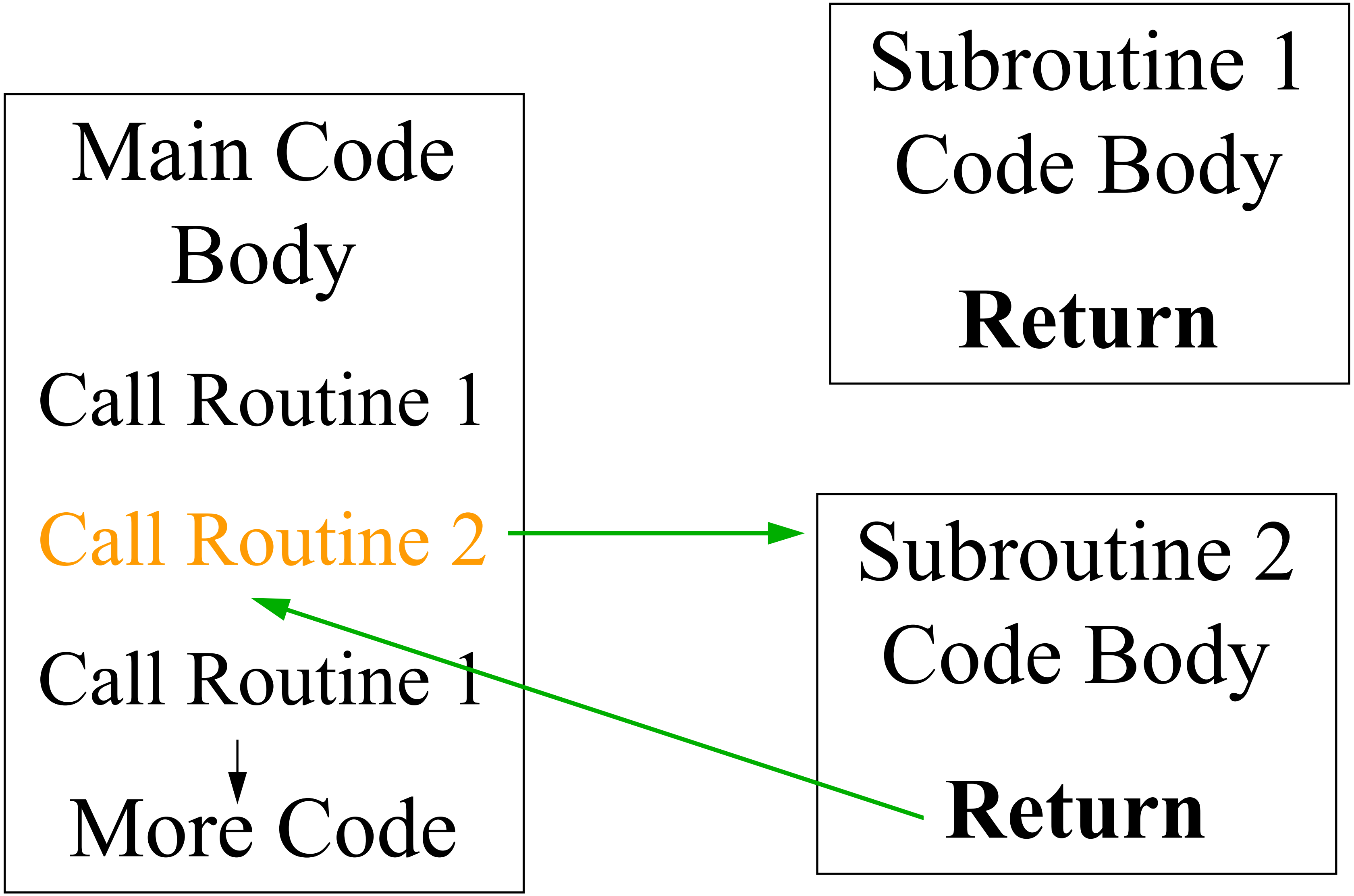
Subroutine 1 Code Body

Return

Subroutine 2 Code Body

Return

Subroutines and Functions.



Main Code Body

Call Routine 1

Call Routine 2

Call Routine 1

More Code

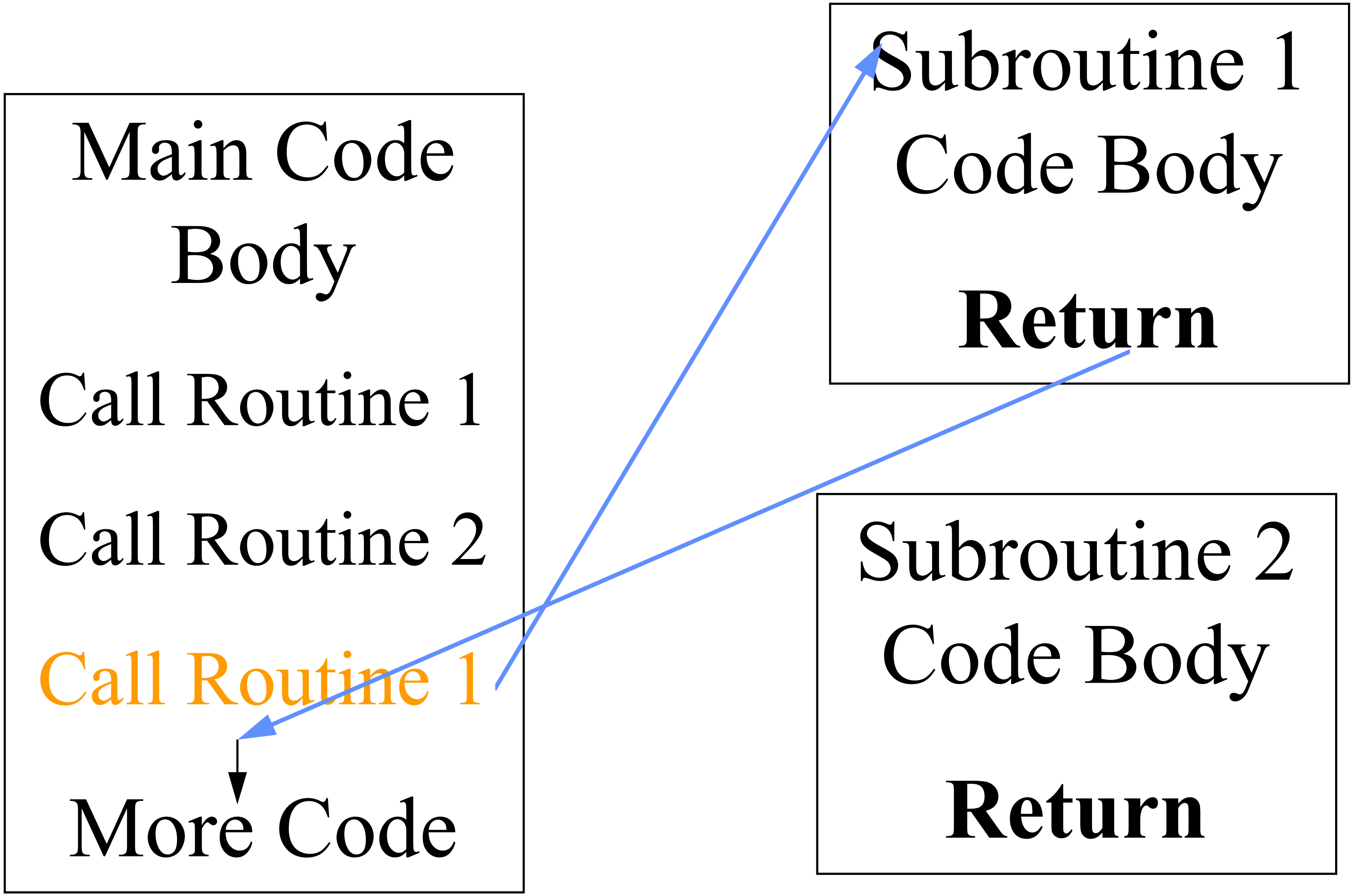
Subroutine 1 Code Body

Return

Subroutine 2 Code Body

Return

Subroutines and Functions.



Main Code Body

Call Routine 1

Call Routine 2

Call Routine 1

More Code

Subroutine 1 Code Body

Return

Subroutine 2 Code Body

Return

Subroutines and Functions.

- Functions and Subroutines are basically the same.
- The Subroutine performs a common block of code
- The Function also performs a common block of code however it reply as an answer assignment.
  example   A = INT(23.5) ‘A= 23
- Both Subroutines and Functions may also additionally accept parameters.
- The GOSUB / RETURN variants do NOT permit the passing of parameters.

Subroutines and Functions.

- Advantages of Subroutines and Functions is that they allow programmer to :-
  - Reduce size of their programmes.
  - Re-use code.
  - Develop their code in a modular form.
- Note: Small modular code sets are easier to maintain as each section can be understood, tested and validated independently.
- Disadvantages
- Call and Return statements creates a small code size and time overhead.

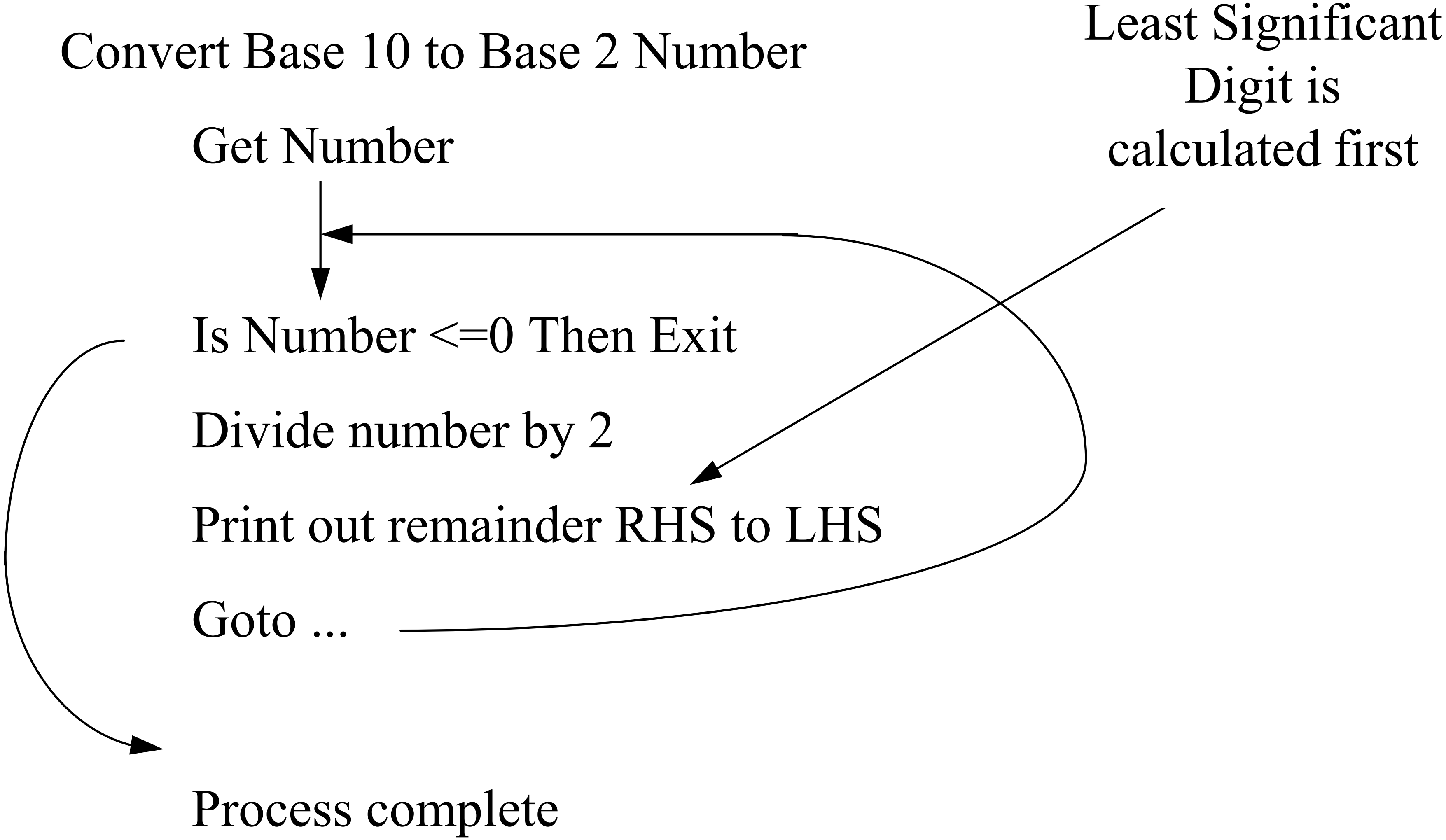
Programming Activities.

Iteration Processes

Number Base Conversion.

More Activities

Convert Base 10 to Base 2 Number



Least Significant Digit is calculated first

Get Number

Is Number <=0 Then Exit

Divide number by 2

Print out remainder RHS to LHS

Goto ...

Process complete

Number Base Conversion.

More Activities

```
REM Convert Base 10 to Base 2 Number
        Print “Enter Value” ;       ‘User prompt
        Input A                     ‘Get Input Value
        While A>0                   ‘Loop till value converted
                B=INT(A/2)          ‘Divide value by 2
                A1=A-(B+B)          ‘A1 = Remainder
                PRINT A1            ‘Display Remainder
                A= B                ‘Next value to Process
        WEND
```

Number Base Conversion.

More Activities

```
REM Convert Base 10 to Base 2 Number with Horizontal Output
        Vals$ = “““ : Dig$=“0123456789”
        Print “Enter Value” ;       ‘User prompt
        Input A                     ‘Get Input Value
        OldValue = A                ‘ Remember Input Value
        While A>0                   ‘Loop till value converted
                B=INT(A/2)          ‘Divide value by 2
                A1=A-(B+B)          ‘A1 = Remainder
                Vals$=Mid$(Dig$,A1+1,1)+Vals$
                PRINT A1            ‘Display Remainder
                A= B                ‘Next value to Process
        WEND
Print “Decimal Value =“;OldValue;” Binary Value =“;Vals$
```

Calculate Roots.

More Activities

Calculate a Cube Root

eg. $X^3 = J$ or $X^3 - J = 0$ Initial value Xn=J

$Xn+1 = Xn - ((Xn^3 - J) / (3 * Xn^2)$

Calculate a Square Root

eg. $X^2 = J$ or $X^2 - J = 0$ Initial value Xn=J

$Xn+1 = Xn - ((Xn^2 - J) / (2 * Xn)$

Newton Raphson Iterative Process.

Xn+1 = Xn - f(Xn)/f ‘(Xn) to solve f(x)=0

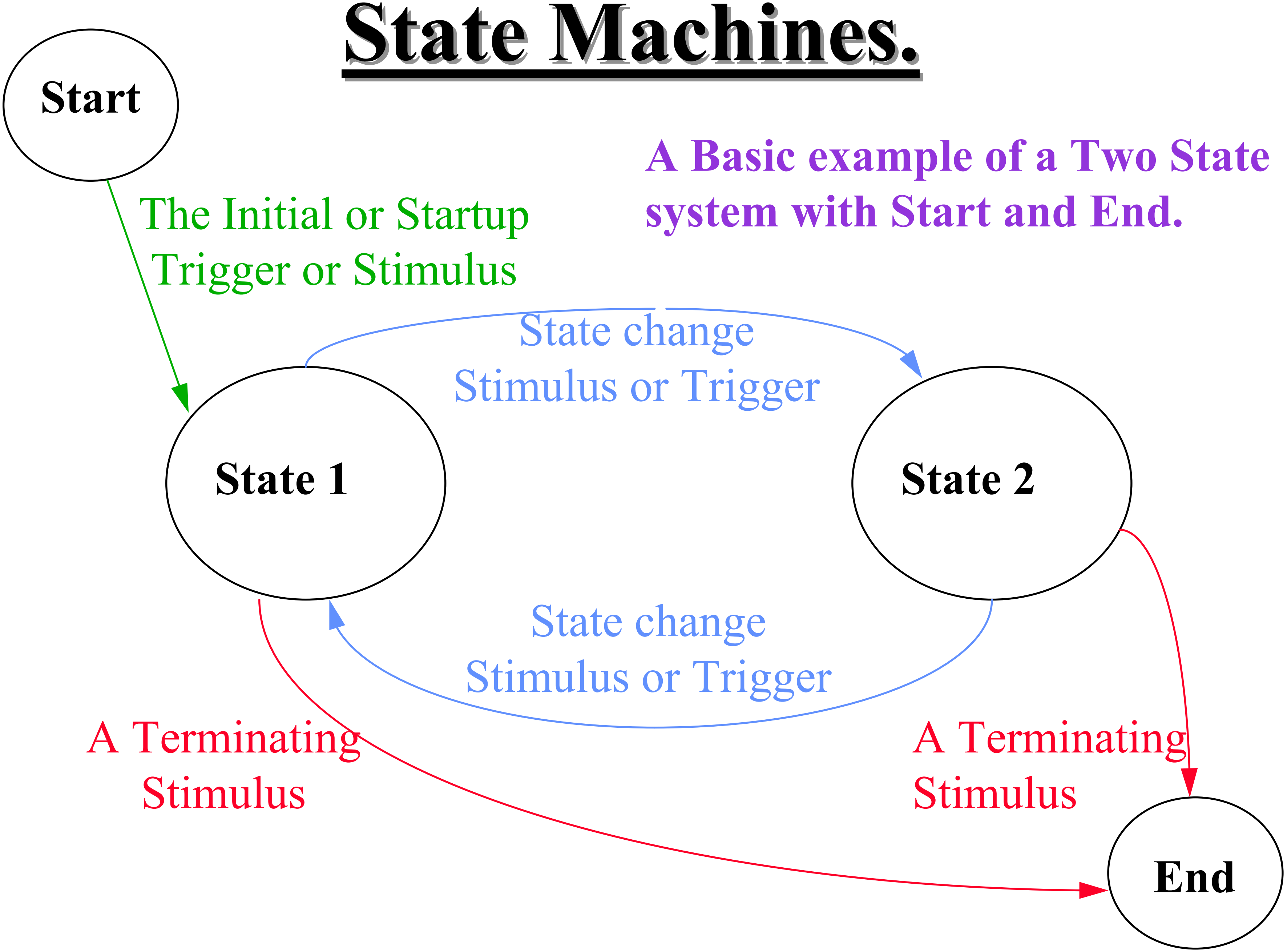
Xn+1 = Initial value - (function / first differential of function)

State Machines.

State Machines.

- What is a State Machine ?
- ***It is a system that moves from one stable condition to another dependent on an external stimulus.***
- Many systems or problems can be subdivided into a sets of stable conditions (or states) that respond to some trigger stimulus.
- We can produce diagrams that both define a problem or can be used to produce an analytical solution to the problem.
- States are represented by **Circles or Boxes** with text inserts.
- Trigger, Stimulus are **Lines** drawn between the states.

State Machines.



A Basic example of a Two State system with Start and End.

Start

The Initial or Startup Trigger or Stimulus

State 1

State change Stimulus or Trigger

State 2

State change Stimulus or Trigger

A Terminating Stimulus

A Terminating Stimulus

End

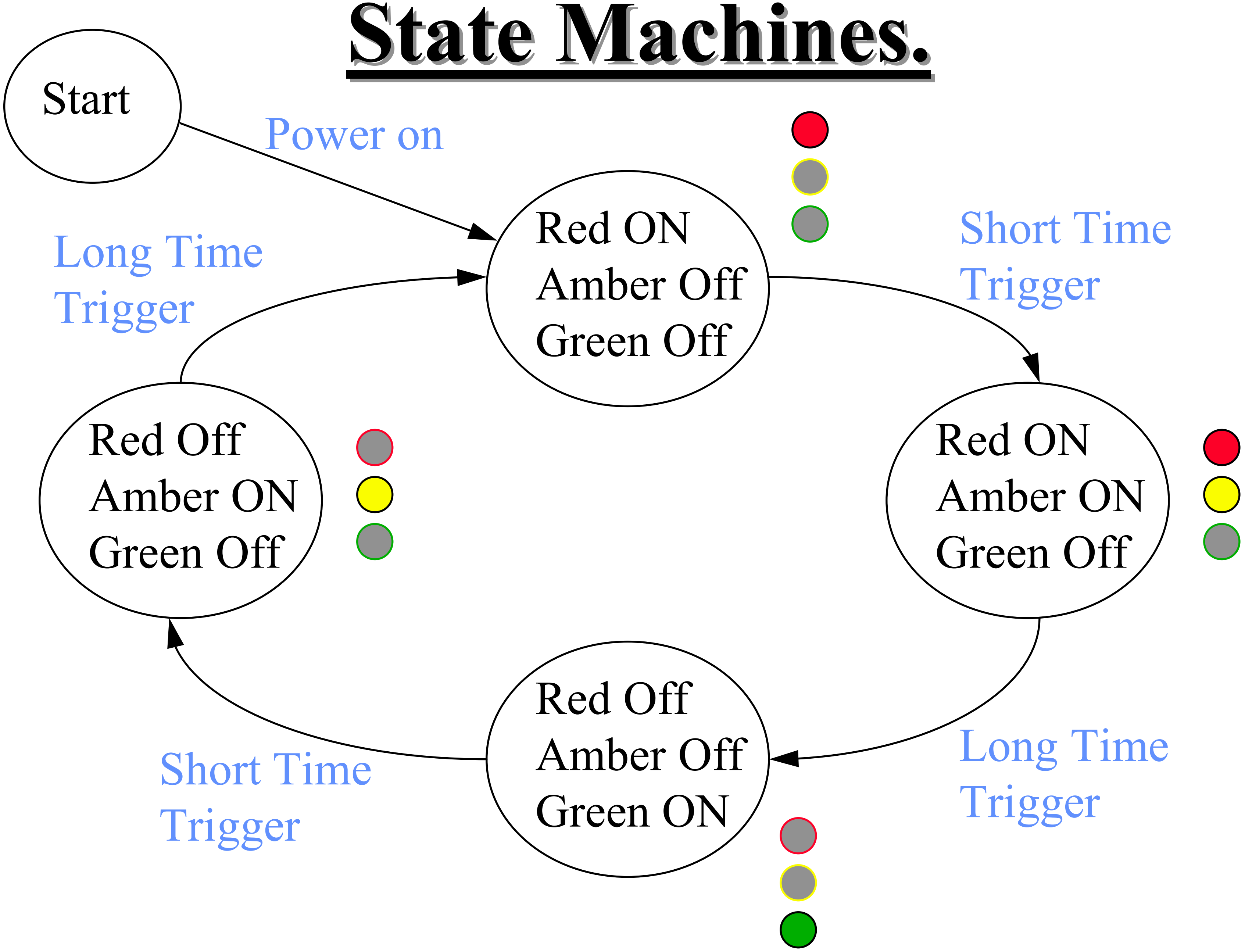
State Machines.

- How may we represent a set of traffic lights ?
- What states can the lights be in ?
- RED, RED + AMBER , GREEN , AMBER, and then back to RED (4 States).
- What sort of transitions signals do we have ?
- We have :-
- A Power on Transition (When they start)
- A Short Display transition that moves

  (RED → RED+AMBER) and (GREEN → AMBER)
- A Long Display transition that moves

  (RED+AMBER → GREEN) and (AMBER→ RED)

State Machines.



File Handling.

File Handling.

- Operations basically similar to the PRINT to Screen and INPUT from Keyboard.
- Before a file can be used it must be accessed or created. The command to start this process is the "OPEN" Statement.
- When the file processing is complete and to ensure your application programme terminates in a tidy manner (good practice) we use the "CLOSE" statement.
- All files are allocated a unique channel number when being opened.

File Handling.

- Filenames in QBASIC take use DOS format :-

  <Name>.<extension>

  where <Name> is eight characters or less

  and <extension> is three characters or less.

The filename must consist of at least one character.

- Typical file OPEN statement could be :-

  OPEN “FILE1.DAT” FOR INPUT AS #1

  or

  OPEN “FILE2.DAT” FOR OUTPUT AS #7

File Handling.

- Useful associated function is “EOF(filenumber)”
- When reading a file: This function is used to identify if the End of the file has been reached.
- Example of use:-

```
IF EOF(1) THEN ....
                    or
WHILE NOT EOF(3)
        process some file statements.
WEND
```

File Handling.

- Reading data from a file we use the command :-

  ```
  INPUT #filenumber, <variables>
  ```

- If we wish to read a whole line of data we can use

  ```
  LINE INPUT #filenumber, <string variable>
  ```

- Writing data to a file we use the command :-

  ```
  PRINT #filenumber, <variables>
  ```

Sorting Data.

Sorting Data.

Why do we need to Sort Data ?

Sorting Data.

- **Why sort data ?**
- To make locating of information easier.
- To make the reading of information easier.
- To highlight specific characteristics of the information set.

Sorting Data.

How do we Sort Data ?

Sorting Data.

- **How do we sort data ?**
- There are thousands of different sorting systems.
- The efficiency of most sorting system much depends upon the way the data is presented.
- Typically we

```
Cycle through data records
    Compare data entries and exchange the
    data when data appears in the incorrect
    order
repeat till data sorted.
```

Sorting Data.

Initially we will look at two simple sorting algorithms :-

(1) The Bubble Insertion Sort.

This sort is suitable for **small** data sets that need to be created.

(2) The Bubble or Ripple Sort.

This sort is suitable for **small** pre-existing data sets that need to be processed.

Both sorts can be easily modified to give either an Ascending and Descending sorted data sequence.

Sorting Data.

- Typical Ascending Insertion Sort Algorithm :-

```
Read Information into NewDataStore
FOR Pointer1 = 1 TO NumberOfDataRecords -1
    IF {contents of DataRecord(Pointer1)
              isGreaterThan   NewDataStore }
    THEN  Exchange DataRecord with NewDataStore
NEXT Pointer1
DataRecord(NumberOfDataRecords)= NewDataStore
Increment NumberOfDataRecords
```

Sorting Data.

- Typical Ascending Bubble Sort Algorithm :-

```
FOR Pointer1 = 1 TO NumberOfDataRecords -1
  FOR Pointer2 =2 TO NumberOfDataRecords
     IF {contents of DataRecord(Pointer1)
                        isGreaterThan
           contents of DataRecord(Pointer2)}
     THEN  Exchange Data Records
  NEXT Pointer2
NEXT Pointer1
```

Sorting Data.

- Typical Descending Bubble Sort Algorithm :-

```
FOR Pointer1 = 1 TO NumberOfDataRecords -1
  FOR Pointer2 =2 TO NumberOfDataRecords
     IF {contents of DataRecord(Pointer1)
                    isLessThan
           contents of DataRecord(Pointer2)}
     THEN  Exchange Data Records
  NEXT Pointer2
NEXT Pointer1
```

Sorting Data.

A few Sorting and File definitions.

Sorting Data.

Some common basic definitions.

- Data consists of blocks of information.
- The basic block of information is called a RECORD.
- The blocks of information may be subdivided into smaller unique sections and these are referred to as FIELDS.
- A collection of RECORDs is a FILE or it may also be referred to as a DATABASE.

Sorting Data.

What problems do we encounter when we Sort Data ?

Sorting Data.

Problems :-

- As the data set get larger the sorting time increases significantly.
- If the data is simple then the exchange process is simple, however if each data record is large then the exchange process also becomes very time consuming.
- What if each data record consisted of multiple fields then how could the data set be sorted.

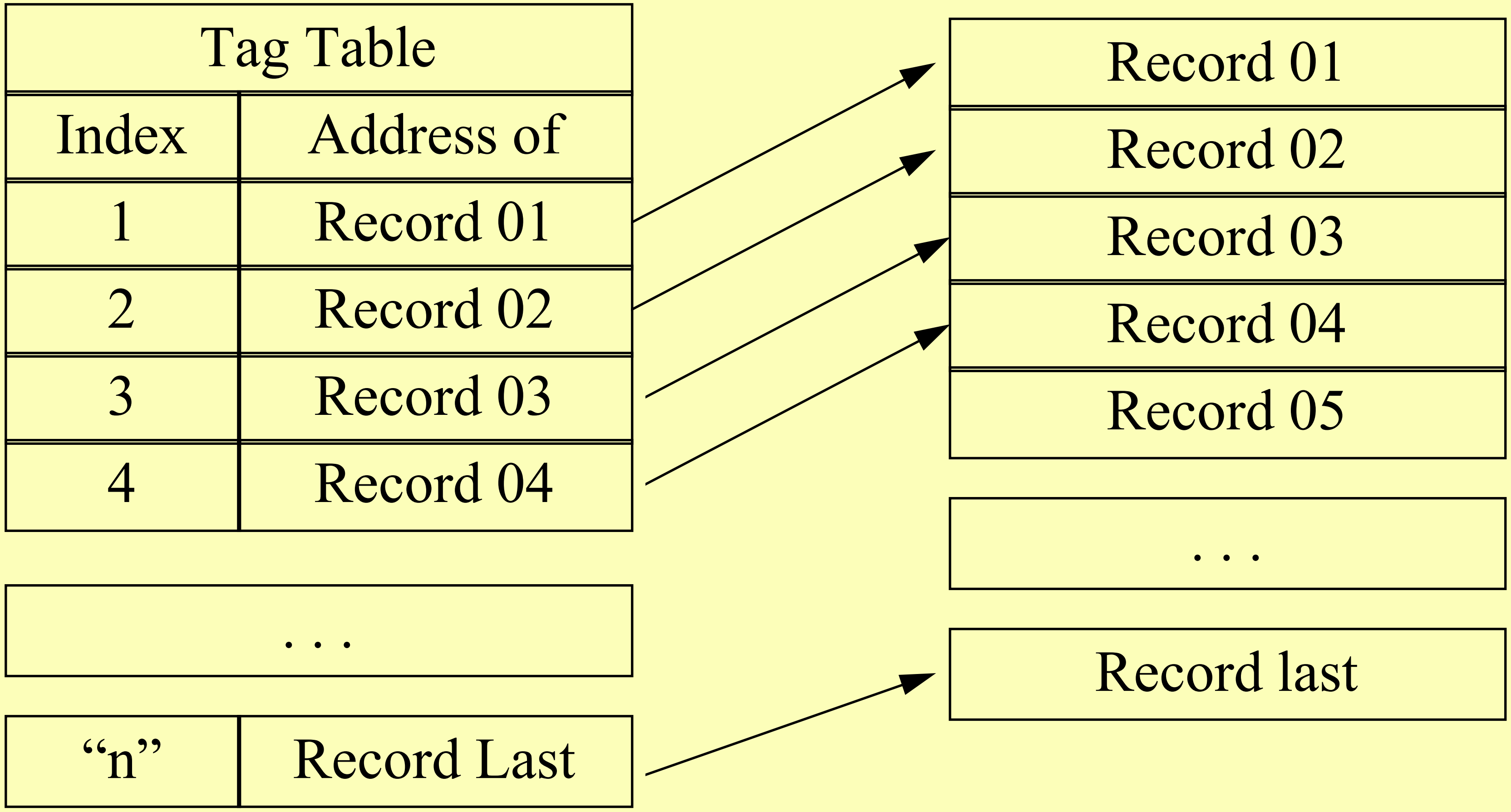
Sorting Data.

A common solution is to build a TAG list and move the tags rather than move the actual data.

However the same sorting methods are still used.

Sorting Data.

Tag Sorting before Sort.



Tag Table	
Index	Address of
1	Record 01
2	Record 02
3	Record 03
4	Record 04
. . .	
“n”	Record Last

Record 01

Record 02

Record 03

Record 04

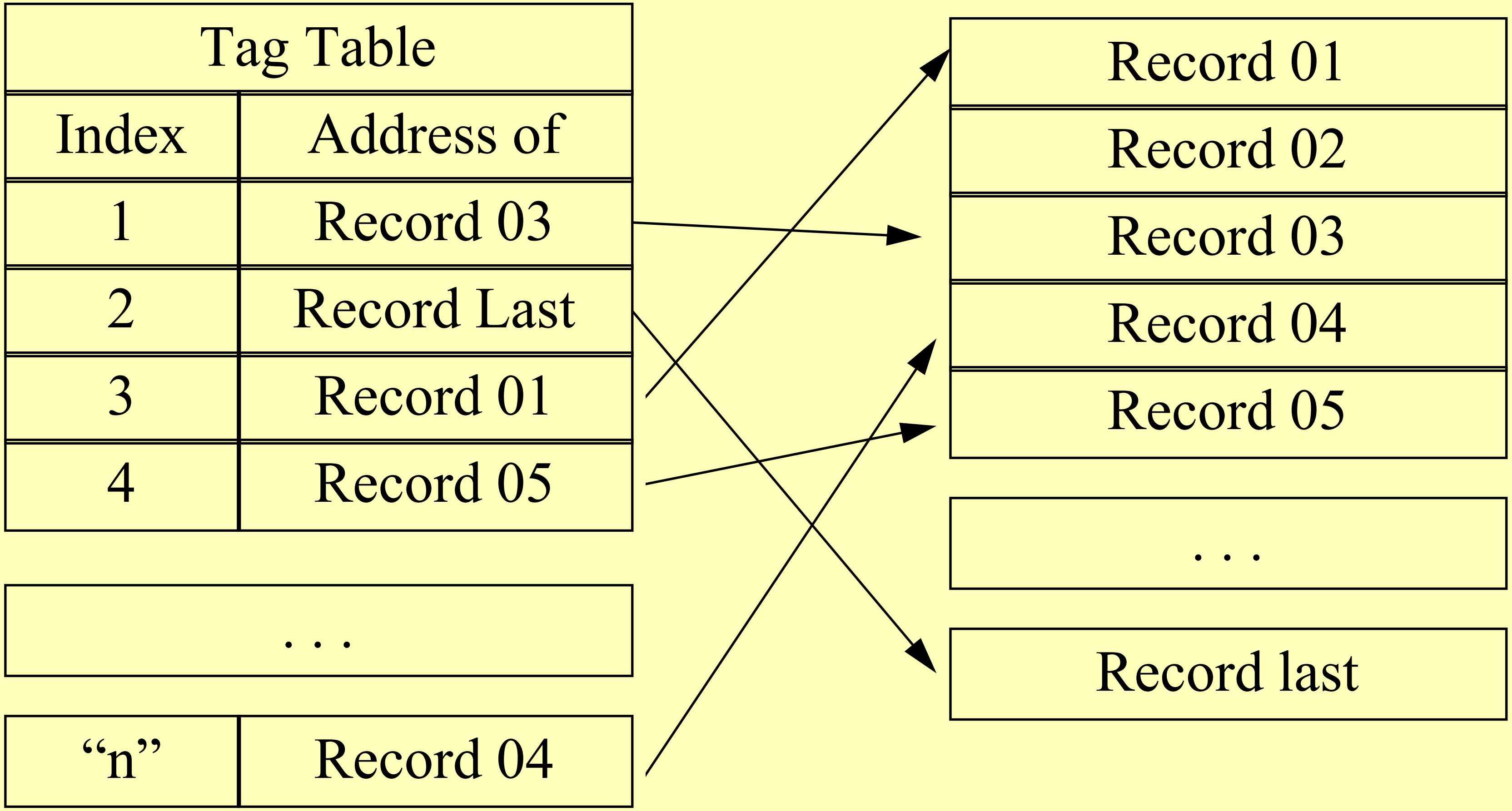
Record 05

. . .

Record last

Sorting Data.

Tag Sorting after Sort.



Tag Table	
Index	Address of
1	Record 03
2	Record Last
3	Record 01
4	Record 05
. . .	
“n”	Record 04

Record 01

Record 02

Record 03

Record 04

Record 05

. . .

Record last

Note the data remains unchanged.

Sorting Data.

Tag Sorting for multiple Sorts.

Tag Table			
Index	Sort One	Sort Two	Sort Last
1	Record 01	Record 01	Record 01
2	Record 02	Record 02	Record 02
3	Record 03	Record 03	Record 03
4	Record 04	Record 04	Record 04
. . .			
“n”	Record Last	Record Last	Record Last

Sorting Data.

Other Sort methods you may encounter :-

- Merge sort. (Suitable for very large data sets)
- Shell Sort.   (Fast general purpose)
- Tree Sort.   (Needs balanced data)
- Quick Sort.   (Common but rather complex)
- Shuttle Sort. (Variant bubble sort)

Visual BASIC

VISUAL BASIC.

- What is **Visual Basic** ?
- It is an advanced/ next generation version of the computer language called **QBASIC**
- Main differences or features are:
  - User interface is Graphical rather than Text
    - Uses WIMPs Windows Icons Menus Pointers.
    - i.e. you use the Mouse rather than the Keyboard that is to say it is event driven.
  - You produce WINDOWS style programmes.
  - Is the Macro language of Microsoft Office applications.
  - Input is via Text/List boxes, Buttons or Check boxes.
  - uses Object orientated programming concepts Oop.

VISUAL BASIC.

- Typical sequence of an event driven application.
  - The application starts and a form is loaded and displayed.
  - The form (or a control on the form) receives an event.
    - The event might be caused by the user (eg. a key press), or by the system (a timer expiring) or indirectly by your code (a load event when your code loads a form)
  - If code has been placed in the event procedure, it is executed. (eg. CommandButton_Click())
  - The application waits for the next event.

VB Variables.

VB Variables.

- Place where information can be stored.
- Variables types additional to QBASIC
  - Currency (Quick math's for Finance)
  - Boolean (True or false)
  - Unsigned Byte (8 bits of Data)
  - Date (Stores Date and Time information)
  - Object (can be used to shorten code)
    - Declare the variable: Dim *variable* As *class*
    - Assign object to it: Set *variable = object*
  - Variant (Can store anything (this is default))

VB Objects.

VB Objects.

- An Object is something real, tangible and is given a name to reference it.
- An Object may contain its own Data and may have the ability to process or manipulate that data.
- Objects have Properties that describe difference or characteristics of that object.
- Objects may also have Methods of how the data they contain may be processed, manipulated or used. (Common methods for data processing are PRINT, ADD, DELETE, INSERT, FIND etc.)
  - A dot is used to link an Object to its Properties or methods.

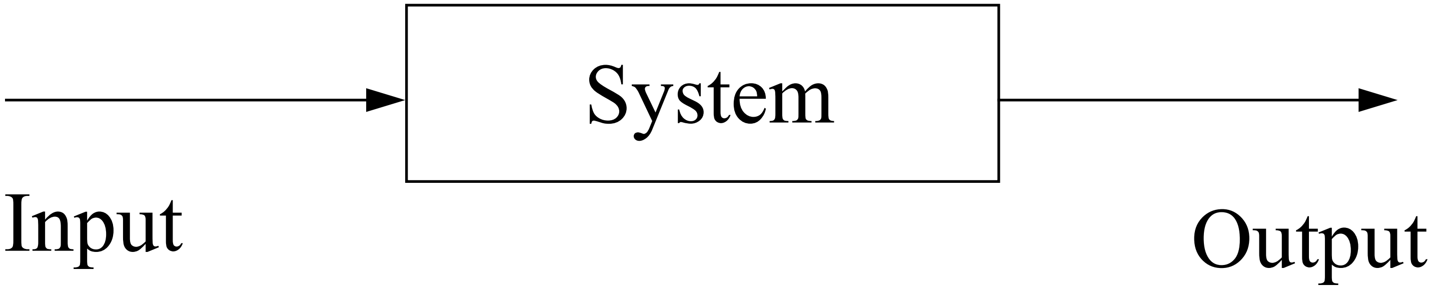
Control Systems and Robotics.

Control Systems.

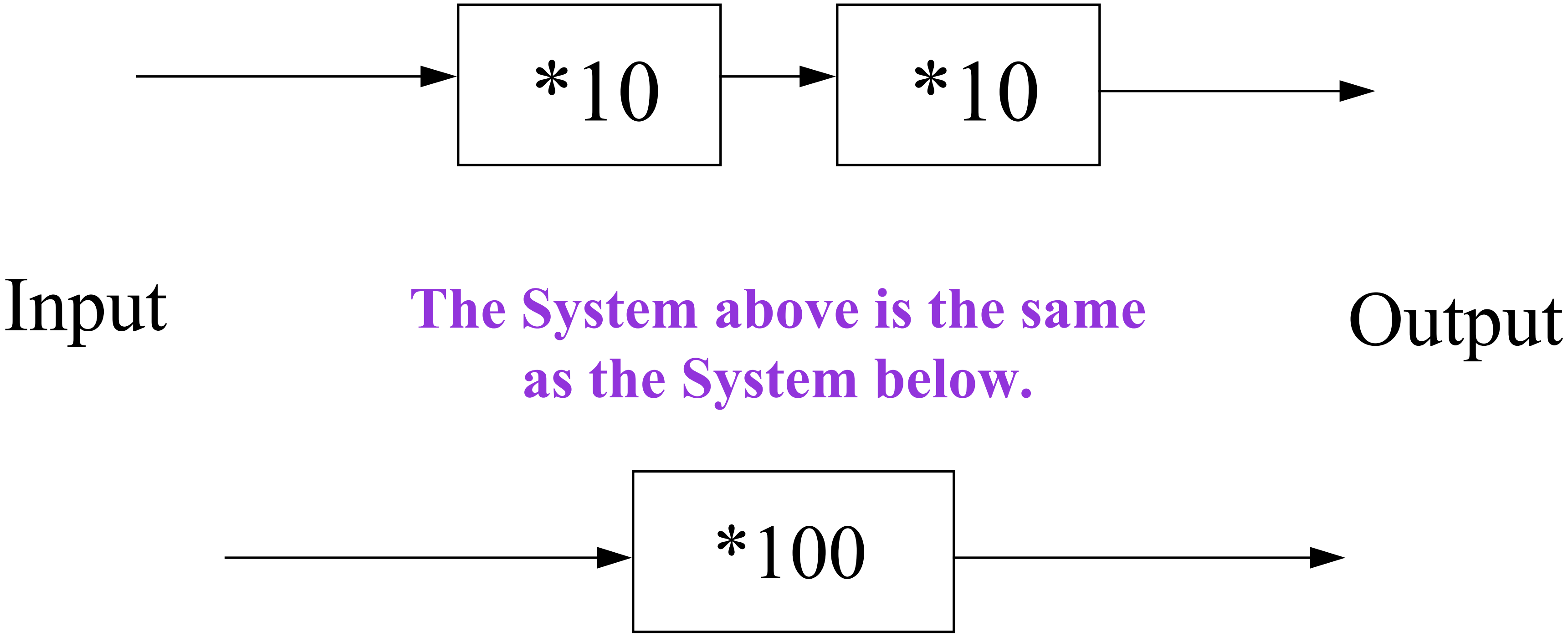
Control Systems.

- What is system?
  - A system is a set of connected items or devices which work together.
- What is special about a control system?
  - It is a system
  - It also has a feedback component.
  - It is a system which can have its behaviour modified as a result of feedback from its own output.

Control Systems.



Control Systems.



*10

*10

Input

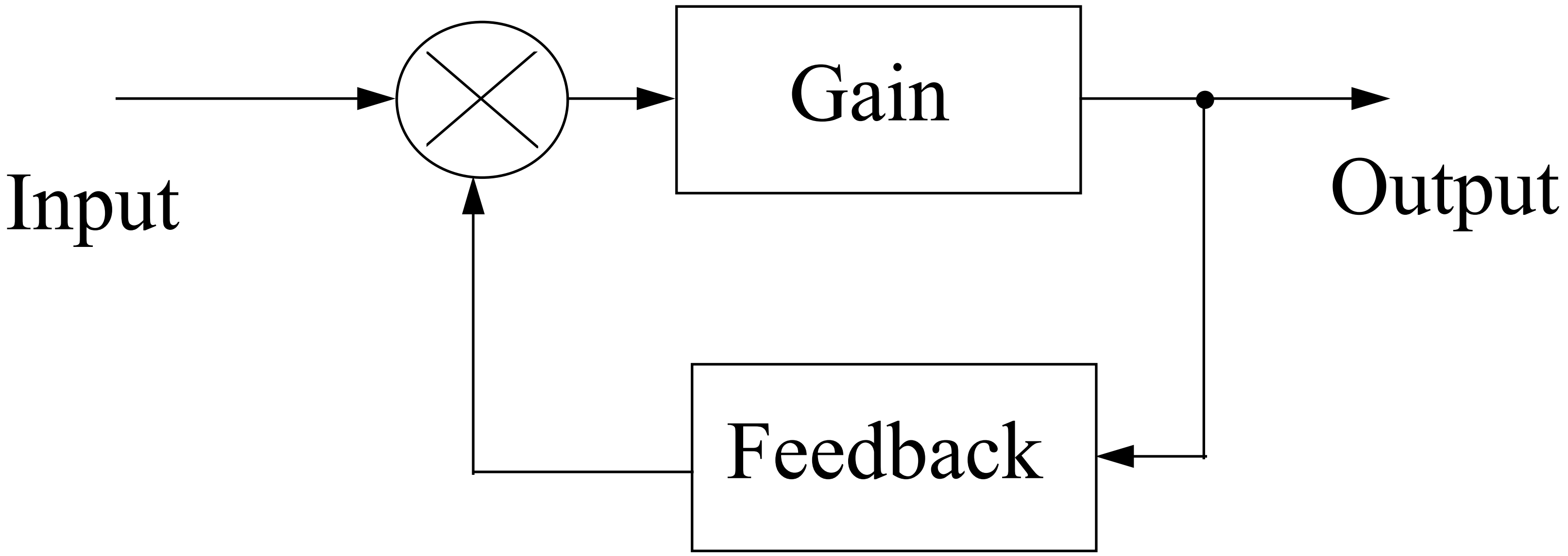
The System above is the same as the System below.

Output

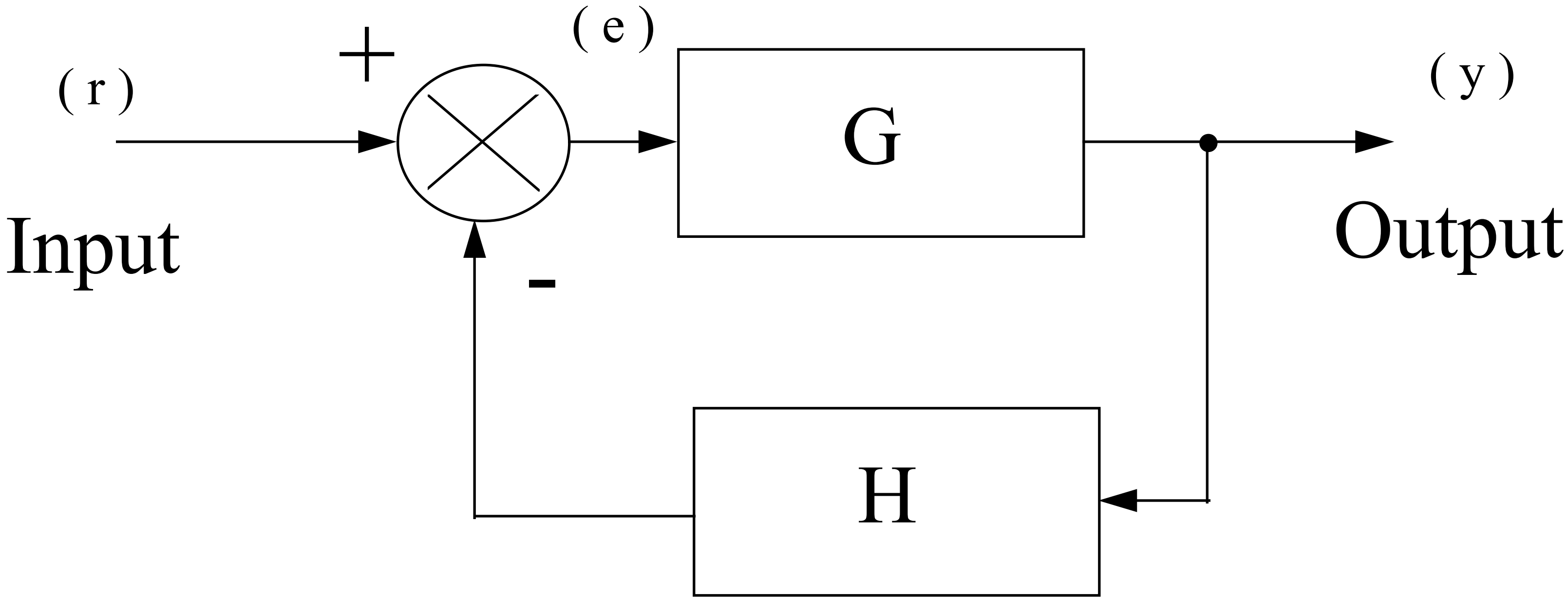
*100

Example of how one may combine two or more systems together.

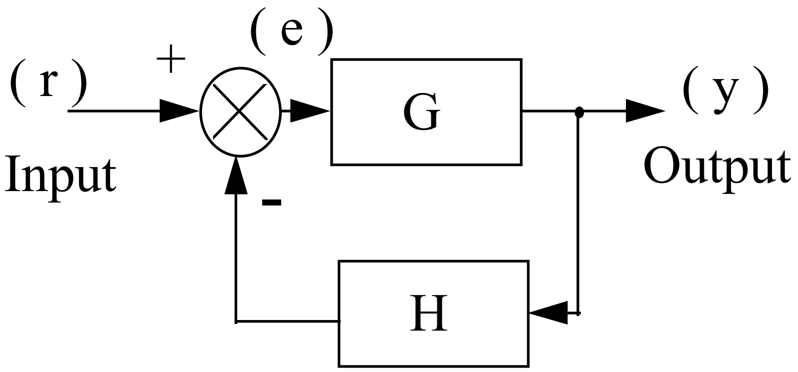
Control Systems.



Control Systems.



Control Systems.



$$(y) = G * (e)$$

$$(e) = (r) - (H * (y))$$

$$(y) = G * \{(r) - (H * (y))\}$$

$$(y) = G(r) - GH * (y)$$

$$(y) + GH * (y) = G(r)$$

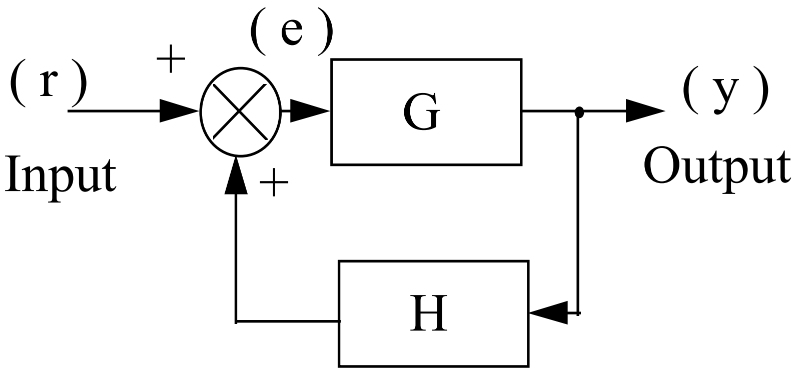
$$(y) (1 + GH) = G(r)$$

However

$$\text{Gain} = \frac{(y)}{(r)} \quad \text{or} \quad \text{Gain} = \frac{G}{(1 + GH)}$$

Negative Feedback Example System

Control Systems.



$$(y) = G * (e)$$

$$(e) = (r) + (H * (y))$$

$$(y) = G * \{(r) + (H * (y))\}$$

$$(y) = G(r) + GH * (y)$$

$$(y) - GH * (y) = G(r)$$

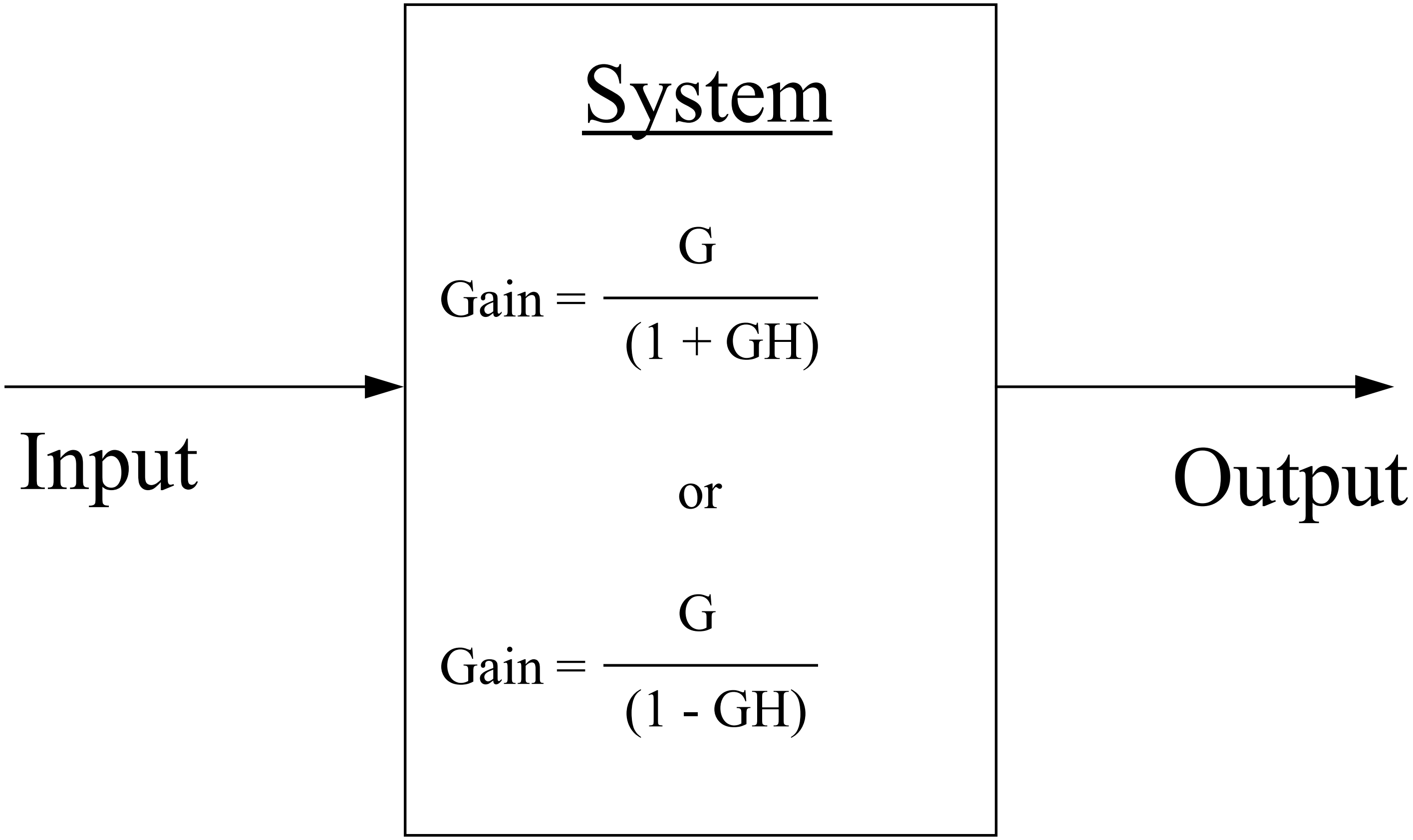
$$(y) (1 - GH) = G(r)$$

However

$$\text{Gain} = \frac{(y)}{(r)} \quad \text{or} \quad \text{Gain} = \frac{G}{(1 - GH)}$$

Positive Feedback
Example System

Control Systems.



System

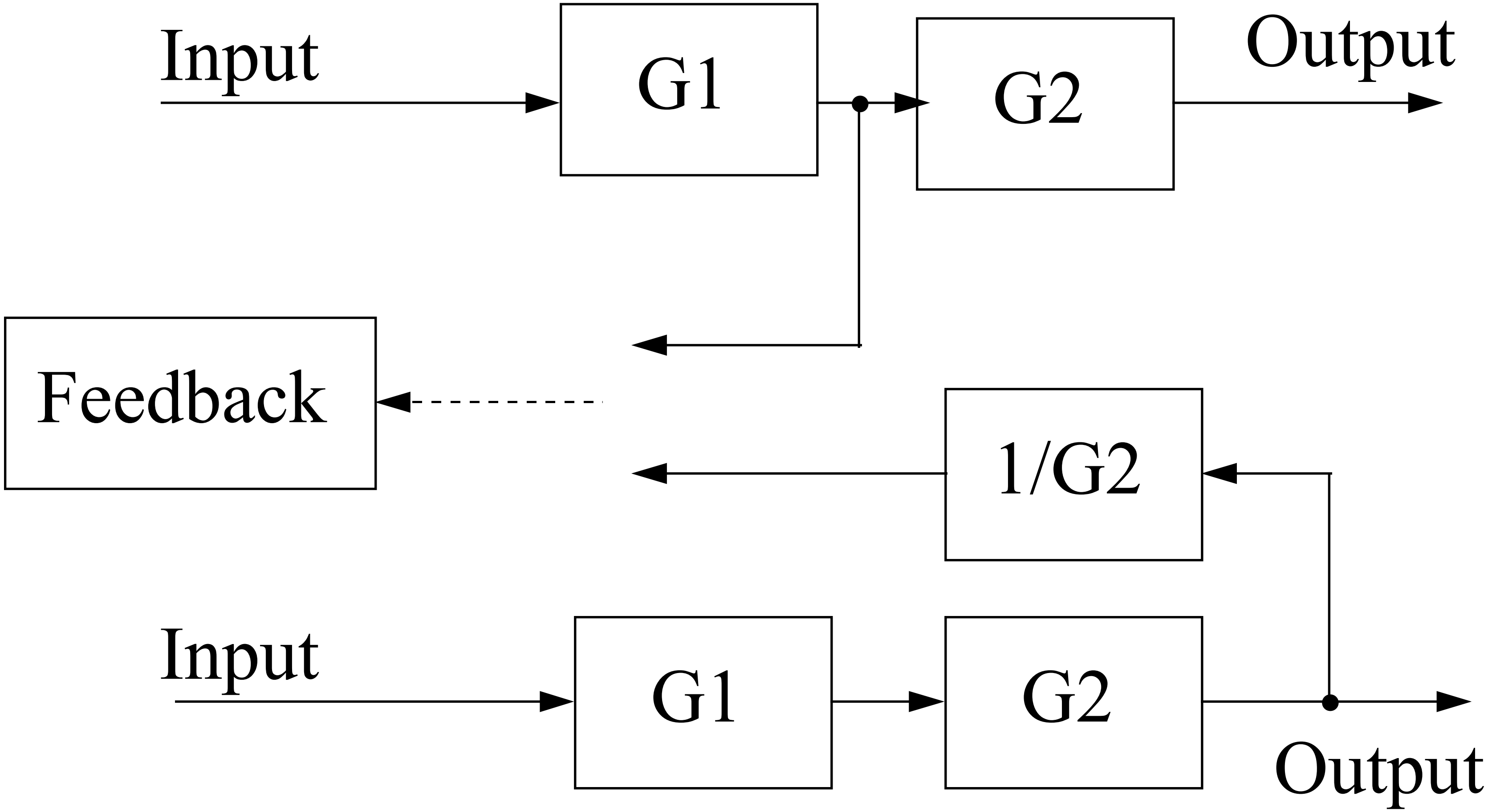
$$\text{Gain} = \frac{G}{(1 + GH)}$$

or

$$\text{Gain} = \frac{G}{(1 - GH)}$$

Control Systems.

Moving the Take off points



Data Logger.

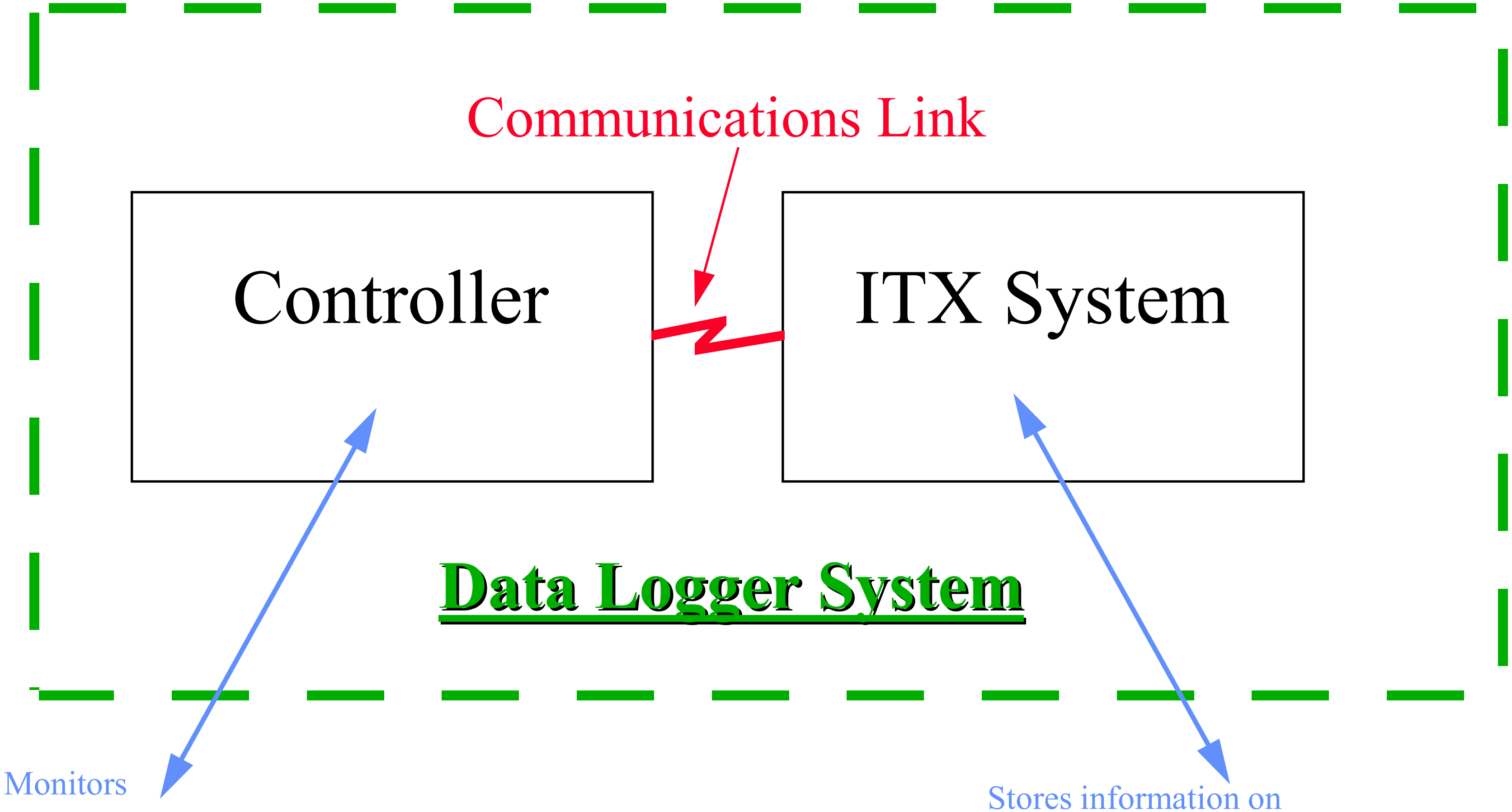
Data Logger.

- What is a Data Logger ?
  - An electronic system that can record physical parameters (Temperature, Voltage , Current, Events pH, Sound or Light levels, Pressure, Depth etc.) over a time period.
  - The physical measurement part of the logger is usually integral with the logger storage part of the system.
  - Usually the logger stored information can be uploaded to or monitored by other systems.

Data Logger.

- Operational modes of Data Loggers.
  - Capture many samples of data in or over a very short time period (Fast time)
    - eg. a crash test, impact and movement sensors.
  - Capture many samples of data over a long or very long time (Continuous , Real-time).
    - eg. Traffic analysis (counting cars per hour/day).
  - Capture samples of data after a change or significant event has occurred (Event, Compressed).
    - eg. Time and Date stamp measurements.
  - Combinations of the above.

Data Logger.



Communications Link

Controller

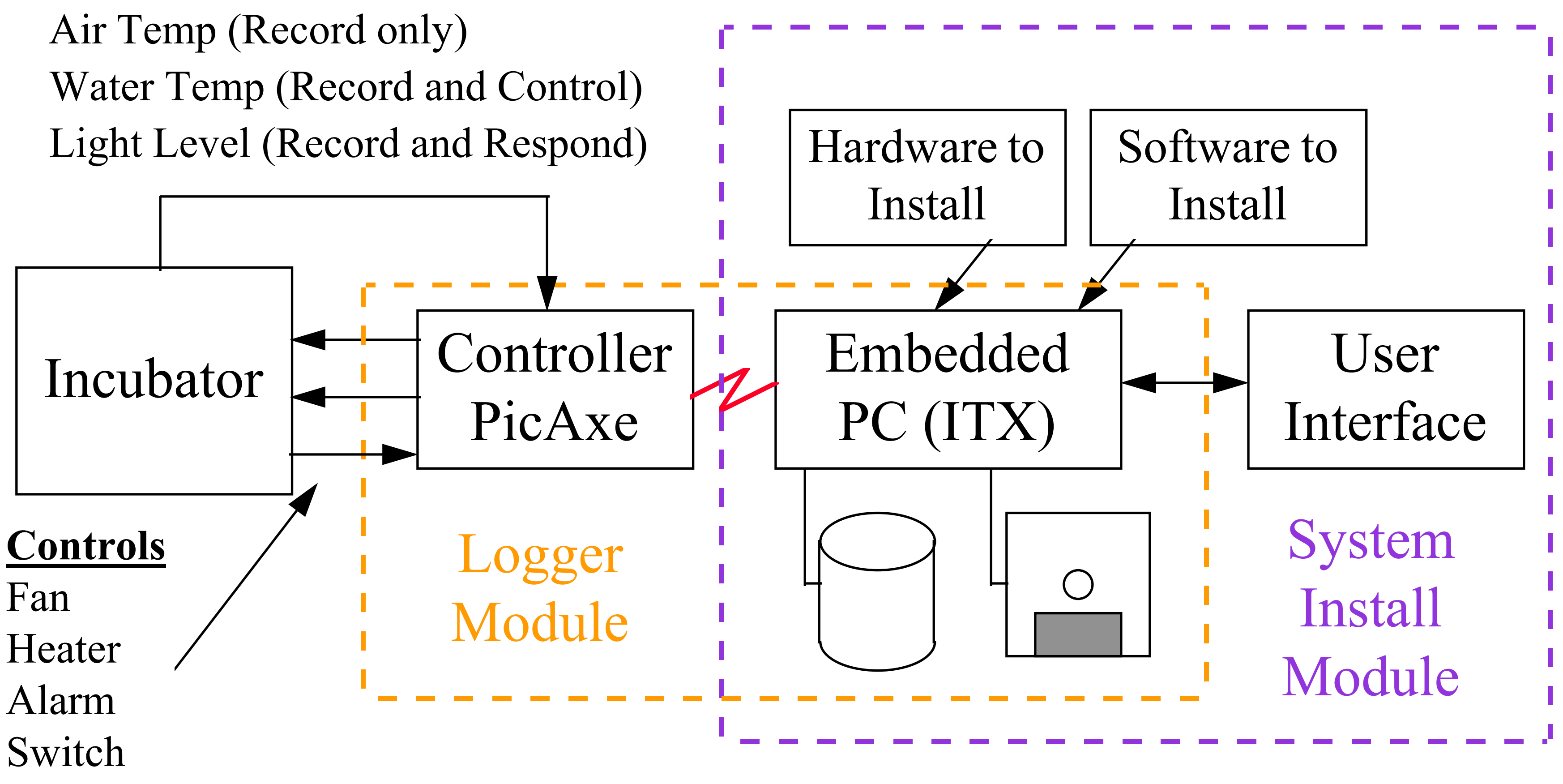
ITX System

Data Logger System

Monitors
Temperature
Depth
Light level

Stores information on
Hard Disk
Floppy disk
Validates messages.

Data Logger - System Installation.



Overall System Configuration

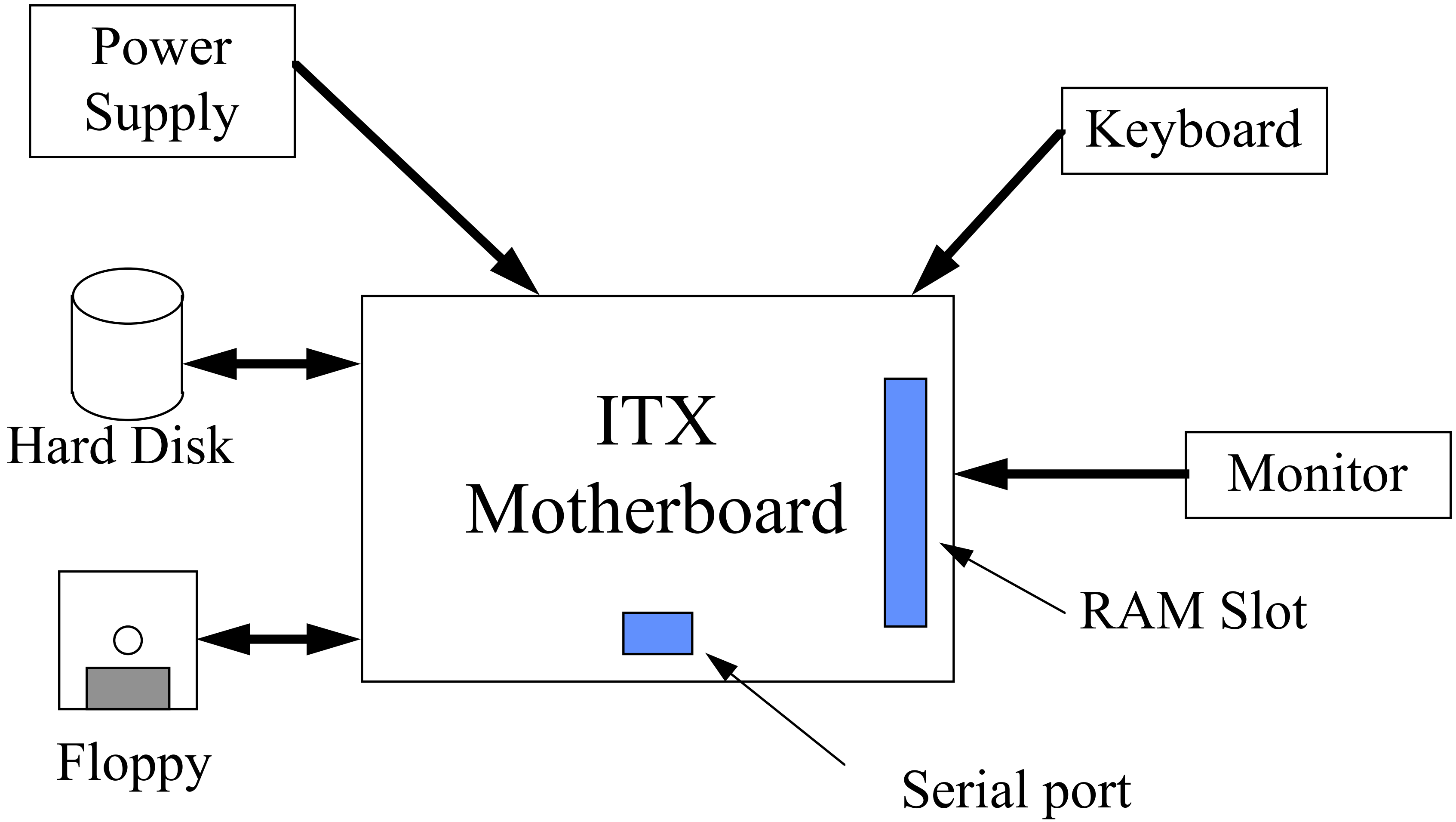
Data Logger.

- Our Logger requirements :-
- A PC based system
  - Motherboard, RAM, Keyboard, Monitor
- Storage Fixed and Removable
  - (Store a weeks worth of data)
- Communications port to controller.
- An operation system that will survive power fails and restart without user intervention. (DOS 6.2)
- Logging Software (Draw a flowchart)
  - Own Software (QBASIC) or use Logger.Exe

Data Logger.

- Resources we can use for Our Logger :-
- 200W ATX PC Power unit
- An ITX Motherboard with onboard
  - PS2 Keyboard, PS2 Mouse, Serial port, Parallel Port, USB, RJ45 LAN, VGA graphics, Sound in and out, TV Out.
- 64Mb RAM Cards (DOS needs max 640Kb)
- Hard Disk with 4Gb Storage.
- Floppy Disk 1.4Mb
- Keyboard and VGA Monitor

Data Logger.



ITX System in basic configuration

Specifications.

Specifications (Two Levels).

- Level 1 (Top) Specification (May be informal)
  - This is an analysis of the user requirement.
- This process will consist of :-
  - Identification of what the customer wants and why.
  - Offering suggestions and reasons that will meet all or most of the customers needs in a written form and with explanation diagrams as appropriate.
  - Explanation what might cause difficulties and how you are going to achieve a suitable outcome.
  - Use of prototypes, demonstrations simulation etc.
  - Agreement to a mutually acceptable solution.
  - Summary documentation explaining the solution.

Specifications (Two Levels).

- Level 2 (Technical) Specification (Formal)
  - This is a detailed analysis of agreed User requirements presented in writing.
- This process will formally define :-
  - What is going to be delivered in detail.
  - When and how it is going to be delivered in detail.
  - Constraint and limitation of use in detail.
  - User and other system interfaces.
  - References to test, commissioning and acceptance.
  - Special requirements that have been agreed.

Robotics.

Robotics.

- What is a Robot?
  - It is a construction of mechanical, electrical and electronic components which is capable of autonomous or semi autonomous operation.
- How does it differ from a control system?
  - It has movement capability.
  - Typically can complete more than one action.

Robotics.

- What **Sensors** might a Robot need and why?
- **Vision** / Light, Dark, edge detection, to locate objects, to monitor shape , size, colour.
- **Pressure** for Touch sensing, Picking objects, selecting items by weight, identifying when an end stop reached.
- **Heat** , **Radioactivity**, when working in hazardous locations.
- **Smell or Taste**, when working in Catering, location of explosives or drugs, product selection Perfumes .....
- **Sound** to identify or track other units if they are close by.
- **Position** to identify where the unit is.
- **Your choice** ...

Documentation and Projects

Projects.

Projects.

You need to produce a portfolio of evidence to show how you applied your skills and understanding to an engineering project. It should include:

- Logbook of what you have done while carrying out your project including:
  - Who you spoke to.
  - Meetings you were involved in.
  - The outcome of your project.
  - Suggestions for improvement.
  - Verbal and written presentation.

Projects.

To achieve a grade E your work must show:

- Effective project planning by regularly communicating with your supervisor.
- Project specification which describes the customer's needs.
- Two potential solutions to the project specification.
- Final design solution which has the potential to work.
- Implementation of your final design solution.
- Comment on the effectiveness of your solution to the problem.
- A product or procedure or system which should be able to meet the customer's needs.

Projects.

To achieve a grade C your work must show:

- Valid conclusions which describe what you achieved.
- Ways in which you could improve your project if you were to do it again.
- A product or procedure or system which adequately meets most of the customer's needs.

Projects.

To achieve a grade A your work must show:

- Critical evaluation of what you have achieved in your project.
- A product or procedure or system which fully meets the customer's needs.

Projects.

Standard ways of working.

- Plan your work to produce what is required to given deadlines.
- When amending software or configuration files save work regularly, particularly previous versions.
- Use file names that are sensible and that help to remind you of the contents.
- Store files where you can easily find them in the directory/folder structure.
- Keep a log of any ICT problems you encounter and how you solve them.

Projects.

Standard ways of working.

- Keep information secure (e.g. protection from theft, loss, viruses, fire).
- Protect confidentiality (e.g. prevention of illegal access to medical or criminal records).
- Observe copyright laws (e.g. not using the work of others without permission).
- Keep dated backup copies of files on another disk and in another location.
- Evaluate your work and suggest how it might be improved.

Projects.

Standard ways of working.

- Proof-read your products (on screen) to ensure accuracy and economic use of material.
- Work safely and recognise all of the health and safety legislation that applies.

Handout sheet ICT_SWW.doc

Projects.

The 80 , 20 rule

and

other hints.

Projects.

- Remember the 80 / 20 project rule.
- With most Projects the Documentation take 80% of the time with 20% being used for developing the actual project.
- With the actual project 80% of the work is usually completed in 20% of the time. It is the last 20% of the work that takes 80% of your effort.
- **A Project that is not planned, plans to fail.**
- Nearly all projects throw up some unexpected problem that takes a significant amount of time and effort to resolve. (Always leave some slack)

Design Tools

Projects.

Design Methods

- Black Art or Bottom Up Design.
- Flowcharts and Block Diagrams.
- HIPO Hierarchical Input Process Output.
- Top Down Design (Decomposition).
- Data Centered or Data Focused Design .
- SSADM Structured System Analysis and Design Methodology (or Jackson Method)
- YOURDON
  - Context, State Diagrams, Pseudo Code.
- UML Unified Modeling Language

Project Specific.

Project Specific.

- Identify what you want to do
  - This gives you the Project proposal.
- Planning and preparation
- Initial ideas
- Develop a final design
- Implement the design
- Test the design
- Report on your project

Project Specific.

- The Project Proposal will contain :-
  - What it is you want to do.
- The Project Proposal may also contain :-
  - The Project Title.
  - Name/s of Customer/s.
  - Any significant Budget constraints.
  - Any significant Time constraints.
  - Any special Resources that are needed.
- The Proposal **MUST BE AGREED** before you start working on the Project.

Project Specific.

- Planning and preparation:
  - Find a customer
  - Work out what the customer wants
  - This is the Customer Specification.
  - Gain your supervisor’s approval
  - Work out what tasks need to be carried out
  - Work out how much time is needed for each.
  - This is part of the Feasibility Study.

Project Specific.

- Initial ideas :
  - Convert the customer requirements into a technical specification.
  - Think of different ways of solving the problem
  - Work out two or three feasible ways of solving the problem
  - Chose the best way of solving the problem.
  - Produce a plan of your project.
  - This is the remainder of the Feasibility Study.

Project Specific.

- Develop a final design :
  - Work out the details of the design.
  - Produce engineering drawings and specifications.
  - Select materials and components.
  - Order any materials and components which are not immediately available.
  - Produce the test procedures from the specification.

Project Specific.

- Implement the design :
  - Build the design or solution to the problem.
- Test the design:
  - Produce a test plan
  - Test the design
  - Fault find and make it work
  - Work out how to improve the design.

Project Specific.

- Report on your project:
  - Write a report
  - Produce illustrations for the report
  - Word process the report
  - Prepare to give a presentation
  - Give a presentation to your class or supervisor.

Documentation.

Documentation.

- Project Documentation should consists of :-
- A Project Requirement Specification.
- A Project Log.
- A Feasibility, Risk and Cost analysis study.
- A Resources, Allocation Plan (Gantt Chart).
  - Time, Materials, Delivery, Milestones
- A Full set of Test procedures and Results.
- The Final Project Outcome.
- A Project Report (with conclusion).

Documentation.

- Programme Documentation consists of :-
- A Overall Specification consisting of :-
  - A Full customer requirements specification.
  - A full functional description which may also contain :-
    - Flowcharts. Block diagrams.
    - Pseudo code. State diagrams.
    - Decision tables. Structure charts (for data areas).
  - Specification for all Input and Output requirements.
- A Fully Modular Commented Programme Source.
- A Full set of Test procedures (System and Module).
- A Full set of Test procedure Results.

Documentation.

- The Project Report should consist of :-
  - Title Page followed by an optional Abstract
  - System Design and used Methodology
  - System Specification
  - Project Plan
    - Project Activities, Project Evaluation
    - Risk Analysis, Decision Analysis, Cost Analysis
  - Module Design (as appropriate)
  - User interface
  - System testing
  - Conclusion
  - References and Appendix

Documentation.

- Programme Documentation should consists of :-

  1. A programme introduction section.

     (**Why**) do we need the programme.

  2. A customer specification and requirements.

     (**What**) does the customer require or need.

  3. A design plan.

     (**How**) are we going to implement the requirements.

     (**When**) are we going to complete.

  4. A programme description which includes :-

     Resources used, calculations etc (all the **Details**).

Documentation.

- Programme Documentation should consists of :-

  5. A fully commented modular programme.

  6. A set of test procedures defining.

     (**How**) are the tests are to be performed.

     (**Why**) are the tests are needed

     (**What**) are we checking for.

  7. A set of test results showing.

     (**What**) has happened **and**

     (**Why**) did it happen.

  8. A final programme report and conclusion on how requirements and specification were met.

Documentation.

- Within a document it is often necessary to define "**Data Structures**" in detail.
- The definition is often called a Data Template and the following Syntax is in common usage.
  - **<item>**
    - Item enclosed in angle brackets is a single entity.
  - **[item]**
    - Items enclosed in square brackets are optional
  - **{item 1 ¦ item 2 ... }**
    - Braces and a broken vertical bar indicate a choice among two or more items. You must choose one of the items unless the choices are enclosed in square brackets.

Templates

Documentation.

- Example Template of a Month description
- <Date> = <Day>/<Month>/<Year>
- <Month> = [<Day Digit 1>]<Day Digit 2>
- <Day Digit 1> = { 0 ¦ 1}
- <Day Digit 2> = { 0 ¦ 1 ¦ 2 ¦ 3 ¦ 4 ¦ 5 ¦ 6 ¦ 7 ¦ 8 ¦ 9}
- Examples of **Valid** Template formats:-
  - dd/1/yy , dd/01/yy , dd/11/yy , dd/19/yy , dd/00/yy
  - **Note.** Just because the template format is correct does **Not** imply that the contents are correct.
- Examples of **Invalid** Template formats:-
  - dd/jan/yy , dd/20/yy

Templates

Testing

Documentation.

- Using Templates with Test procedures definitions.
- Points to **Remember**:-
- When testing validity of data you need to check that:-
  - The data meets the expected data format (Template).
  - The data is meaningful in the context:-
    - Numbers are within valid ranges (between +44 and -66).
    - Number of appropriate type
      - (Integers with decimal point ?)
      - Exponent / scientific format (2.3E+5 or 55.6E-2).
      - Using the correct number base (011% , AF2H, +5).
  - Sequences of information are appropriate.
    - The actual Messages sequence order is correct.
    - Start blocks come before End blocks.

Templates

Dealing with the Briefs.

Dealing with the Briefs.

- Basic Requirements for all Briefs :-
- Activity log or diary (especially for Brief 3)
- Initial draft design ideas showing methodical approach (Not just a final outcome).
- Flowcharts, Flow Diagrams or Block diagrams.
- Test procedures and how test data and result have been produced. Error Handling when appropriate.
- All applications fully commented and operational.
- Documentation with Introduction, body and conclusion.

Dealing with the Briefs.

Area	**Project**	**What is needed, Minimum requirements.**
Brief 1.	**The Picaxe Controller**	• Test procedure sheets * 2 (Yours plus Customer)
Brief 2.	**The Picaxe Serial Interface**	• Log example of data output. • Test procedure sheets * 2 • Templates of data structure.
Brief 3.	**The ITX Embedded system**	• A Log of all Activities , • Test Data , Utility Log Reports, • Test Procedures, Real Data, • Macro produced test data, • Witness Statement/s
Brief 4.	**The VB. Data Logging Application**	• Test Procedures, Full Documentation. • Screen shots. User guide.

Project Plan Structure.

Project Plan Structure.

- What activities might I need to develop a project ?
- A Project proposal.
  - The proposal will identify an outline or scope for the project.
  - I could identify who the sponsor is.
  - It could identify why the project is needed.
  - It could identify some typical outcomes.
  - It may be used to identify if the project is acceptable to be developed for the sponsor organization.

Project Plan Structure.

- What activities might I need to plan a project ?
- A Project proposal.
- Customer specification and its constraints.
  - What the customer requires.
    - Size, Weight, Performance, Cost, Features, Finish...
  - When the customer requires it.
  - How it is delivered.
  - Special constraints
  - Health and Safety, Quality management
    - BS5750, ISO9000, (*CE*)

Project Plan Structure.

- What activities might I need to plan a project ?
- A Project proposal.
- Customer specification and its constraints.
- Feasibility study which may include:-
  - Outline design plans to select best option.
  - Cost Analysis.
  - Resources Analysis.
  - Risk Analysis.
  - Project Plan
-

Project Plan Structure.

- What activities might I need to plan a project ?
- A Project proposal
- Customer specification
- Feasibility study
- Product Development
- Product Production
- Product Testing

Plans, Risk and Tests.

Plans.

Plans, Risk and Tests.

- **How do I plan a project ?**
- Stage sketch out activities you think you may need to perform

Plans, Risk and Tests.

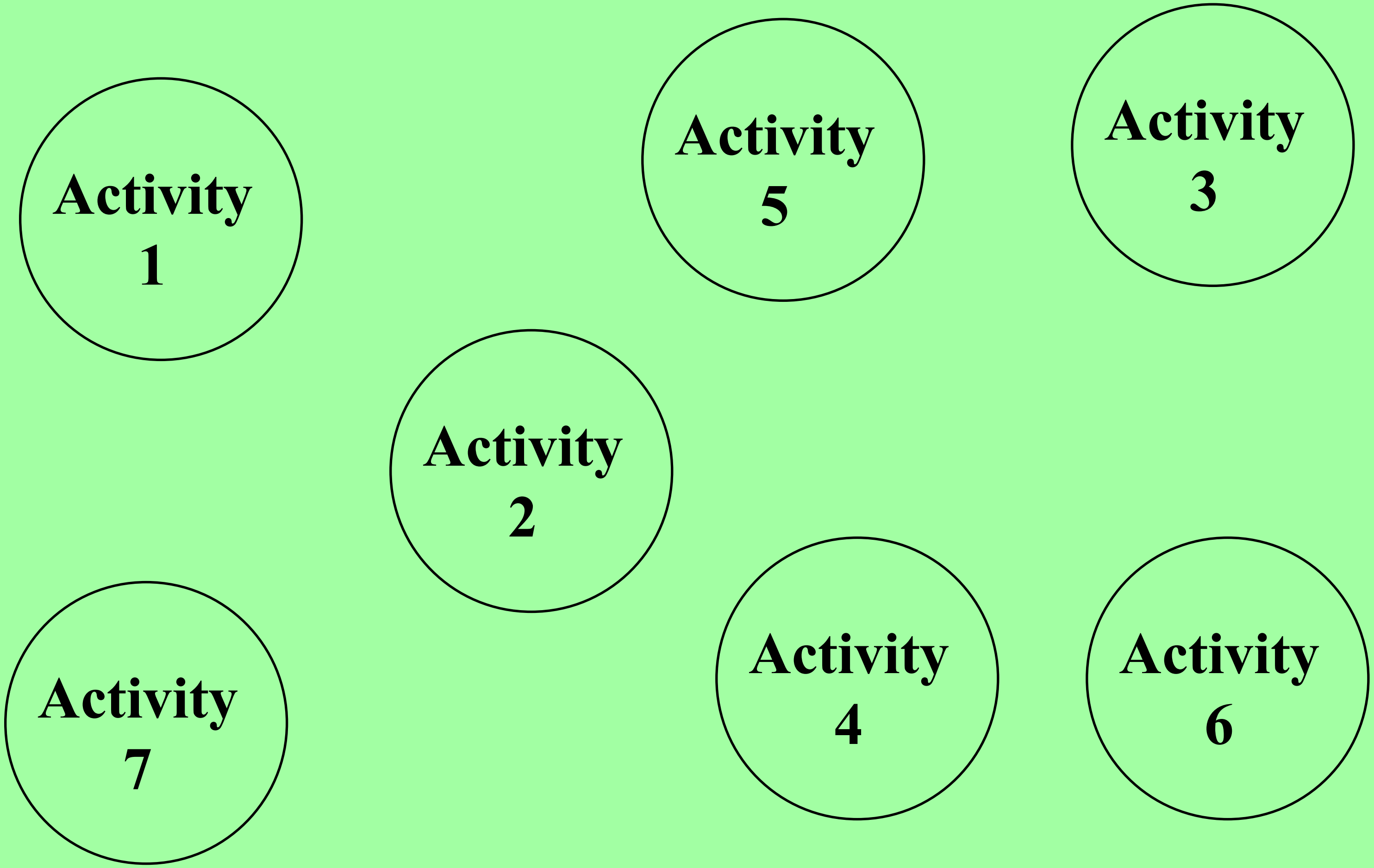


Diagram of Activities that need to be performed.

Plans, Risk and Tests.

- How do I plan a project ?
- Stage sketch out activities you think you may need to perform.
- Add relationships to the activities for example indication data flows, messages that might be passed or resources needed.

Plans, Risk and Tests.

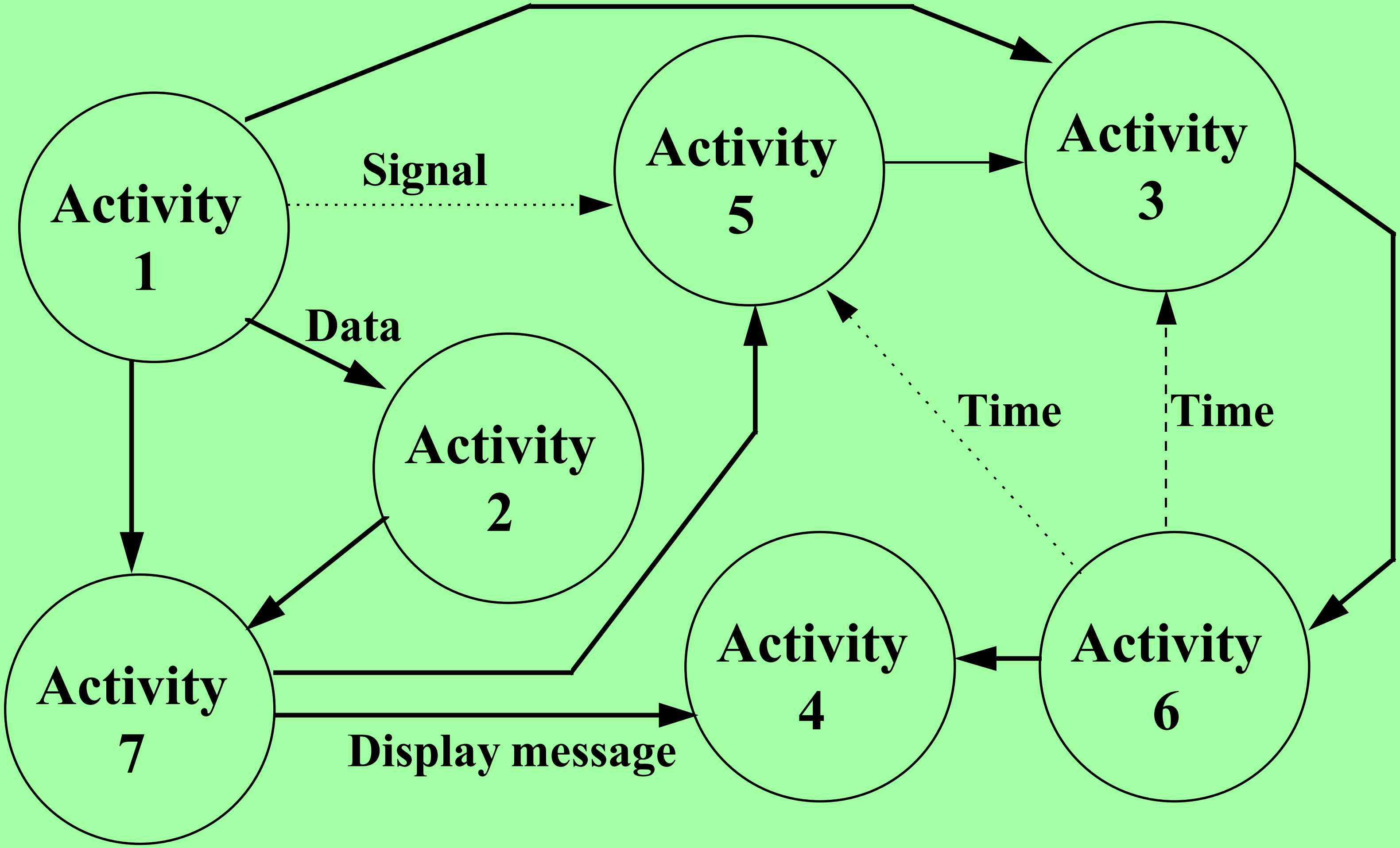


Diagram showing relationship between Activities.

Plans, Risk and Tests.

- How do I plan a project ?
- Stage sketch out activities you think you may need to perform.
- Add relationships to the activities for example indication data flows, messages that might be passed or resources needed.
- **Build a table of all activities that need to be performed with time estimates: Then ...**
- **Calculate Expected times ..**

  (1*Slow + 4*Average + 1*Fast)/6

Plans, Risk and Tests.

Activity		Fast	Avr	Slow	Expected	Comments
1	Documentation	3	4	7	4.33	
2	Order Parts	1	1	2	1.16	No Problem
3	Develop Code	5	9	25	11	Low Risk
4	Write report	2	5	16	6.33	Cut and Paste ?
5	Activity 1	1	4	5	3.66	
6	Activity 2	7	14	19	13.67	Research needed
7	Activity 3	2	3	5	3.167	Usually easy
8	Activity 4	1	3	7	3.33	Ditto
9	Activity 5	22	42	45	39.16	Could give problems
10	Activity 6	16	17	22	17.67	Tape player needed
11	Activity 7	4	6	17	7.5	
12	System Design	0	0	0	0	
13		0	0	0	0	
14		0	0	0	0	

Table of activities with time estimates.

Plans, Risk and Tests.

- How do I plan a project ?
- Stage sketch out activities you think you may need to perform.
- Add relationships to the activities for example indication data flows, messages or resources that might be passed.
- Build a table of all activities that need to be performed with time estimates.
- Finally decide dependency order the activities need to be “performed in” with any specific resources that need to be available.

Plans, Risk and Tests.

Activity		Fast	Avr	Slow	Expected	Dependence order
1	Documentation	3	4	7	4.33	12,4
2	Order Parts	1	1	2	1.16	12,7
3	Develop Code	5	9	25	11	10
4	Write report	2	5	16	6.33	12
5	Activity 1	1	4	5	3.66	12
6	Activity 2	7	14	19	13.67	3,5
7	Activity 3	2	3	5	3.167	5
8	Activity 4	1	3	7	3.33	2, 6
9	Activity 5	22	42	45	39.16	12
10	Activity 6	16	17	22	17.67	11
11	Activity 7	4	6	17	7.5	12
12	System Design	3	5	7	5	Start Activity
13		0	0	0	0	
14		0	0	0	0	

Table of activities with Dependencies.

Plans, Risk and Tests.

- How do I plan a project ?
- **With the basic information of the plan and the calculated time allocations you can now draw a chart to plot your progress.**

Plans, Risk and Tests.

Activity		Expected	Dependence order
1	Documentation	4.33	12,4
2	Order Parts	1.16	12,7
3	Develop Code	11	10
4	Write report	6.33	12
5	Activity 1	3.66	12
6	Activity 2	13.67	3,5
7	Activity 3	3.167	5
8	Activity 4	3.33	2, 6
9	Activity 5	39.16	12
10	Activity 6	17.67	11
11	Activity 7	7.5	12
12	System Design	5	Start Activity
13			

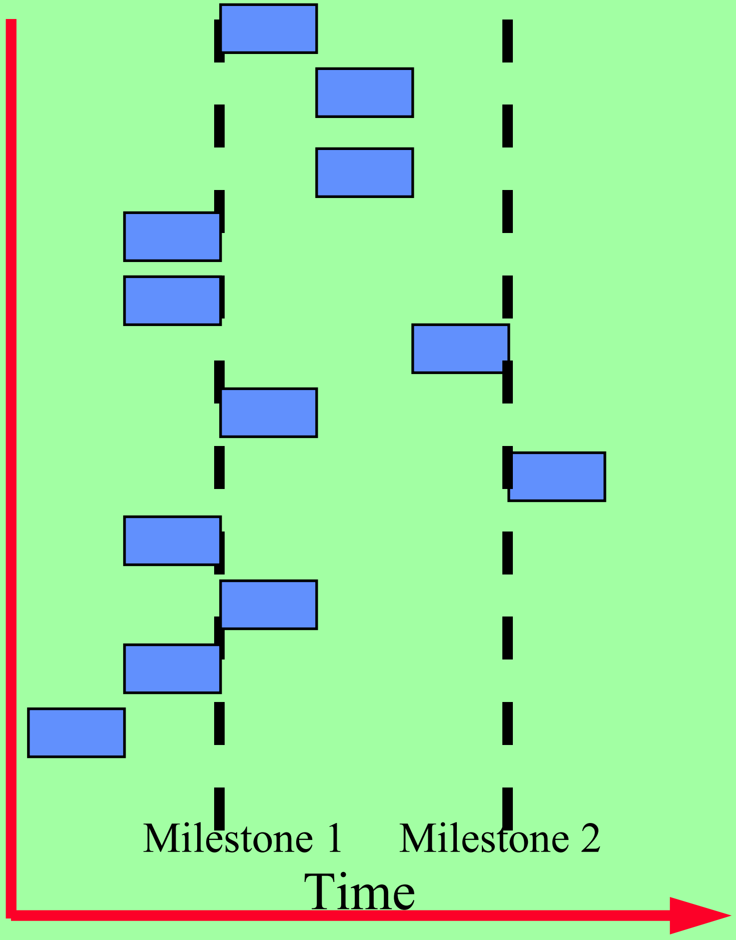


Table of activities with Dependencies.

Plans, Risk and Tests.

- How do I plan a project ?
- With the basic information of the plan and the calculated time allocations you can now draw a chart to plot your progress.
- If we scale the time allocation we can produce a Gantt chart to help monitor progress.
- Note we can allocate Milestones to our chart again to help us identify progress and any slippage within the project.

Plans, Risk and Tests.

Activity		Expected
1	Documentation	4.33
2	Order Parts	1.16
3	Develop Code	11
4	Write report	6.33
5	Activity 1	3.66
6	Activity 2	13.67
7	Activity 3	3.167
8	Activity 4	3.33
9	Activity 5	39.16
10	Activity 6	17.67
11	Activity 7	7.5
12	System Design	5
13		

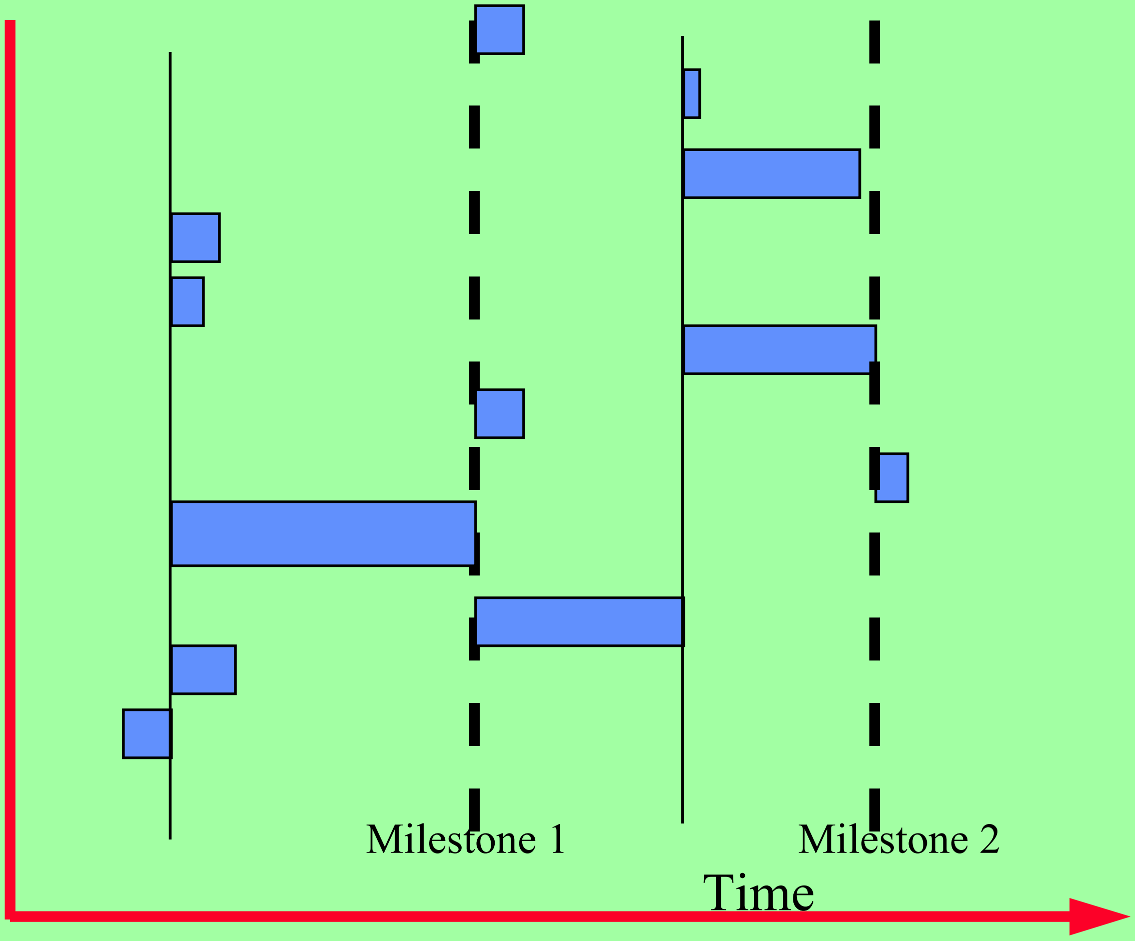
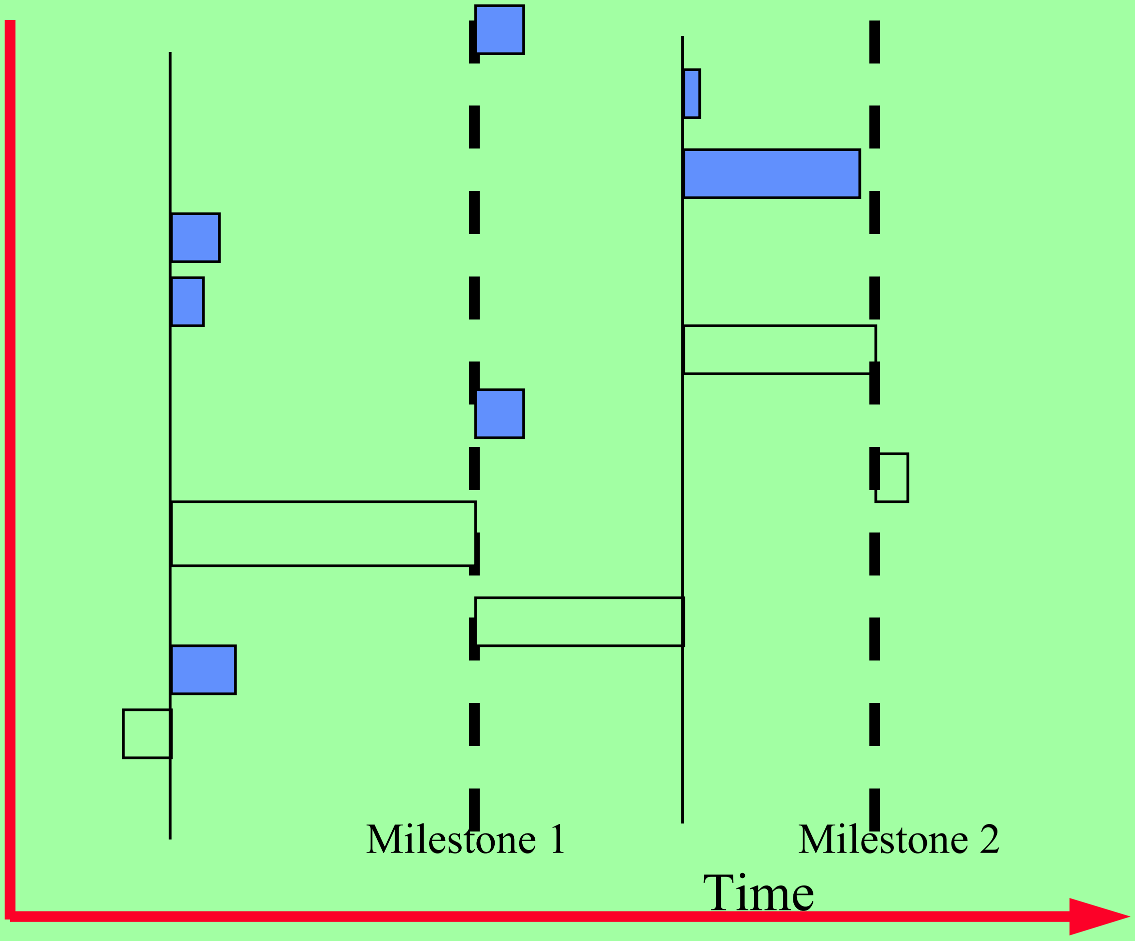


Table of activities with Gantt type display.

Plans, Risk and Tests.

Activity		Expected
1	Documentation	4.33
2	Order Parts	1.16
3	Develop Code	11
4	Write report	6.33
5	Activity 1	3.66
6	Activity 2	13.67
7	Activity 3	3.167
8	Activity 4	3.33
9	Activity 5	39.16
10	Activity 6	17.67
11	Activity 7	7.5
12	System Design	5
13		



Gantt type display showing Critical Path.

Risk.

Plans, Risk and Tests.

- How do I plan a project ?
- Risk Analysis
- It is considered that there are three categories of risk.
- Type 1 risk.
  - The events that can be easily to overcome.
  - They cause minor delays.
  - They need additional resources
  - They have a reasonable likelihood of occurring.

Plans, Risk and Tests.

- How do I plan a project ?
- **Risk Analysis**
- **It is considered that there are three categories of risk.**
- **Type 2 risk.**
  - The events that can be difficult to overcome.
  - They can cause major delays.
  - They may need significant additional resources
  - They have a low probability of occurring.

Plans, Risk and Tests.

- How do I plan a project ?
- **Risk Analysis**
- **It is considered that there are three categories of risk.**
- **Type 3 risk.**
  - The Total Disaster.
  - The Project fails.
  - Compensation due to failure to deliver.
  - They are a very rare occurrence.

Plans, Risk and Tests.

- **How do I plan a project ?**
- **How do I assess Risk.**
  - Step 1. Assume a numeric range for risk for example say 1 to 5.
  - Step 2. Assume a numeric range for risk modification for example say +2 to -2 (For the unknown elements).
  - Step 3. Allocate a numeric risk value to each activity.
  - Step 4. Allocate a numeric risk modification value to each activity if this is appropriate.
  - Step 5. Calculate the risk value for each activity.
  - Step 6. Re-evaluate all activities that are above a threshold of acceptable risk for the project.

Plans, Risk and Tests.

- How do I plan a project ?
- **How do I assess Risk.**
  - **Possible risk value assignments.**

– Risk value =1	No Problem.
– Risk value =2	Should be OK.
– Risk value =3	Acceptable.
– Risk value =4	Cause for concern.
– Risk value =5	May not succeed.
– Risk Modify value = -1	Experienced at doing this.
– Risk Modify value = 0	Situation normal.
– Risk Modify value = +1	Unusual option.

Plans, Risk and Tests.

Activity		Risk	Mod	Sum	Comments
1	Documentation	1	0	1	
2	Order Parts	1	0	1	No Problem
3	Develop Code	4	-2	2	Low Risk
4	Write report	1	0	1	Cut an Paste ?
5	Activity 1	1	1	2	**RISK Concern**
6	Activity 2	2	+1	3	Research needed
7	Activity 3	2	-1	1	Usually easy
8	Activity 4	1	1	2	Ditto
9	Activity 5	3	+1	4	Could give problems
10	Activity 6	2	-1	1	Tape player needed
11	Activity 7	4	+1	5	
12	System Design	1	1	2	
13		0	0	0	
14		0	0	0	

25 = Sum , Average = 25/12=2.083

Overall project Risk
Should be OK (2.083)

Table of activities with Risk estimates.

Tests.

Plans, Risk and Tests.

- **How do I Test a project ?**
- **Identify from the specification all the constraints of the system.**
  - Verify the **System** works when all parameters are within the defined constraints.
  - Verify the **System** works when parameters approach any of the defined constraints.
  - Verify the **System** responds correctly when any or all parameters drift beyond the defined constraints. (Ensure non Destructive settings)

Conclusion.

Plans, Risk and Tests.

- How do I plan a project ?
- **In Conclusion**
  - – Plan to succeed.
  - – Monitor your progress (Log Book).
  - – Log all activities so you can identify with hindsight where the project could have been improved.
  - – Use the skills developed to improve future projects.

Templates

Templates.

- Templates are a formal method or way of defining structures or shapes.
- A mechanical **template** may be used to identify positions where holes are to be drilled or that an item must fit a specific shape.
- In a similar manner a message **templates** identify the exact structure of a messages.
- With messages we use a hierarchical type structure starting with the whole message and breaking it down into smaller and smaller segments until we define all aspects of the message.

Templates.

Consider the message :-

Message Number nnn, Temp = nnn <line end>

- Hopefully we can see that the message has space for two numbers. (nnn and nnn)
- It has two sections of Plain text (Message Number and , Temp =)
- And the message is terminated (<line end>)
- We could divide the message into the following sections <Prefix><Temp><Terminator>
- We can then subdivide these sections.

Templates.

Consider the message :-

Message Number nnn, Temp = nnn <line end>

- <Prefix> consists of :-

  < Message Number ><Number>
- <Temp> consists of :-

  < , Temp = ><Number>
- <Terminator> normally consists of :-
  two characters that enable the print position to start on a new line and at left hand side of page
- We can then subdivide the common <Number> section using the same definition.

Templates.

Consider the message :-

Message Number nnn, Temp = nnn <line end>

- < Number > consists of :-

  <Digit><Digit><Digit>

- <Terminator> consists of :-

  <CR><LF>

- <Digit> consists of :-

  One of the following characters

  0 1 2 3 4 5 6 7 8 9

- Note: Code for CR = 13 and Code for LF = 10.

Templates.

Template Structure Description Definition Syntax.

- <item> Anything enclosed in angle brackets is a single entity.
- [item] One or more entities enclosed in square brackets are optional
- {item1 ¦ item2 } Braces and a broken vertical bar indicate a choice among two or more items. You must choose one of the items unless the choices are enclosed in square brackets.
- Plain text Must exist as written.

Templates.

Formal definition of message :-

Message Number nnn, Temp = nnn <line end>

Message	= <Prefix><Temp><Terminator>
<Prefix>	= Message Number <Number>
<Temp>	= , Temp = <Number>
< Number >	= <Digit><Digit><Digit>
<Terminator>	= <CR><LF>
<Digit>	= {0 ¦ 1 ¦ 2 ¦ 3 ¦ 4 ¦ 5 ¦ 6 ¦ 7 ¦ 8 ¦ 9}
<CR>	= 13.
<LF>	= 10.

MACRO

Processing

MACRO Processing.

- The term **Macro** has many meanings in the computer world.
- It can mean a process that records a sequence of keyboard commands or mouse clicks that can be replayed when required. The recorded sequence may also be automatically converted into a programme script. (ie. to VB in Excel)
- It is also a loosely defined block structured computer style language. It is overlaid on many Assembler Languages and also present in some High level languages like “C”.

MACRO Processing.

- The **Macros** are quite often presented as Subroutines or Sub processes.
- The main strength of the Macro is that parameters can be passed to the macro which can be expanded within the body of the macro.
- The Macro permits a single line of information to be expanded into many lines.
- The Macro enables a level of intelligence to be applied to a text or programme source.

A Macro Language.

A MACRO Language.

- “MAC” This language is based on Digital Electronics Corporation DEC MACRO_11 computer language.
- It is one of the more complete Macro languages with some extended and unique features that can aid programme application development.
- “MAC” can also be used in a stand alone mode that allows the user to develop Macros to process standard text or HTML files.
- “MAC” supports the following features :-

A MACRO Language.

- The language recognizes numbers with the following bases :-

– Base 2 (**%** Binary)	10110%
– Base 8 (**O**ctal)	177O
– Base 10 (**.** Decimal)	128.
– Base 16 (**H**exadecimal)	1AFH

- The language recognizes two type of Symbols :-
  - The Number Symbol (Range 0 to FFFFH).
  - The String Symbol (Max 50 Characters).

A MACRO Language.

- Symbol names consist of 1 to “n” characters.
- The default setting is 6 characters wide. However the INIT programme can adjust the Symbol name width up to 24 characters wide.
- The Symbol name must start with an Alpha then the remainder of the name can be any combination of both Alphas and Numerics.
- Example
  - AName , Bref01 , MyVal
- Note Character case is insignificant. “ABC” and “abc” symbol names are assumed to be the same.

A MACRO Language.

- The language also supports conversion between the two types of Symbols :-
  - The **.DEFNUM** directive converts an expression to a Number Symbol.
  - The **.STR_SYM** directive converts a numeric value into a String Symbol.
- The language also supports expressions, however these are evaluated left to right without any hierarchical precedence.

A MACRO Language.

- The .VOID directive.
- This directive can be used to :-
  - Make a gap in the input source that will not appear in the output image.
  - Store a comment.
  - Erase another directive during debug process.
- Example

```
.VOID
.VOID Your comments could go here
.VOID .DEFNUM MySym,123.
```

A MACRO Language.

- To define a Symbol :-
  - we use the .DEFINE directive.
  - .DEFINE P1,P2
  - Where :-
  - P1 = Symbol name.
  - P2 = Text String.
- Example

```
.Define SParm1,Hello
.Define SParm2,”Hello with Spaces”
.Define SParm3,’Hello with Spaces’
.Define SParm4,<Hello with Spaces>
```

A MACRO Language.

- Symbol representation examples :-

.Define SParm1,Hello

SParm1 = HELLO

.Define SParm2,”Hello with Spaces’ and a Quote”

SParm2 = Hello with Spaces ’ and a Quote

.Define SParm3,’Hello Spaces “ and a Double Quote’

SParm3 = Hello Spaces “ and a Double Quote

.Define SParm4,<Hello with Spaces>

SParm4 = HELLO WITH SPACES

- Note Unprotected strings are always automatically converted to Uppercase characters.

A MACRO Language.

- To define a number :-
  - we use the .DEFNUM directive.
  - .DEFNUM P1,P2
  - Where :-
  - P1 = Symbol name.
  - P2 = Value or Numeric expression.
- Example

```
        .DefNum NParm1,128.
        .DefNum NParm2,1001%
        .DefNum NParm3,123O
        .DefNum NParm4,0AAH
```

A MACRO Language.

- The .REPT and .ENDR directive.
- This directive is used to :-
  - Repeat a block of lines a fixed number of times.
- Example

```
 .REPT 20.
          Repeat this line 20 decimal times
 .ENDR
```

A MACRO Language.

- The .IF_?? , .IF_??? and .ENDIF directives.
- These directives are used to :-
  - Create a visible block of lines of text which are controlled by the status of a Symbol.
  - The visible block of lines of text are always enclosed between the .IF... directive and the .ENDIF directive
- The .IF directives come in two forms:-
  - The Symbol definition group.
  - The Symbol contents comparison group.

A MACRO Language.

- The . DEF_MACRO and .ENDM directive.
- These directives are used to :-
  - Define a Macro definition.
  - Define the default parameters associated with the Macro definition.
- Format

```
.DEF_MACRO {macro name}{Macro parameters}
            { Macro Body Contents }
 .ENDM
```

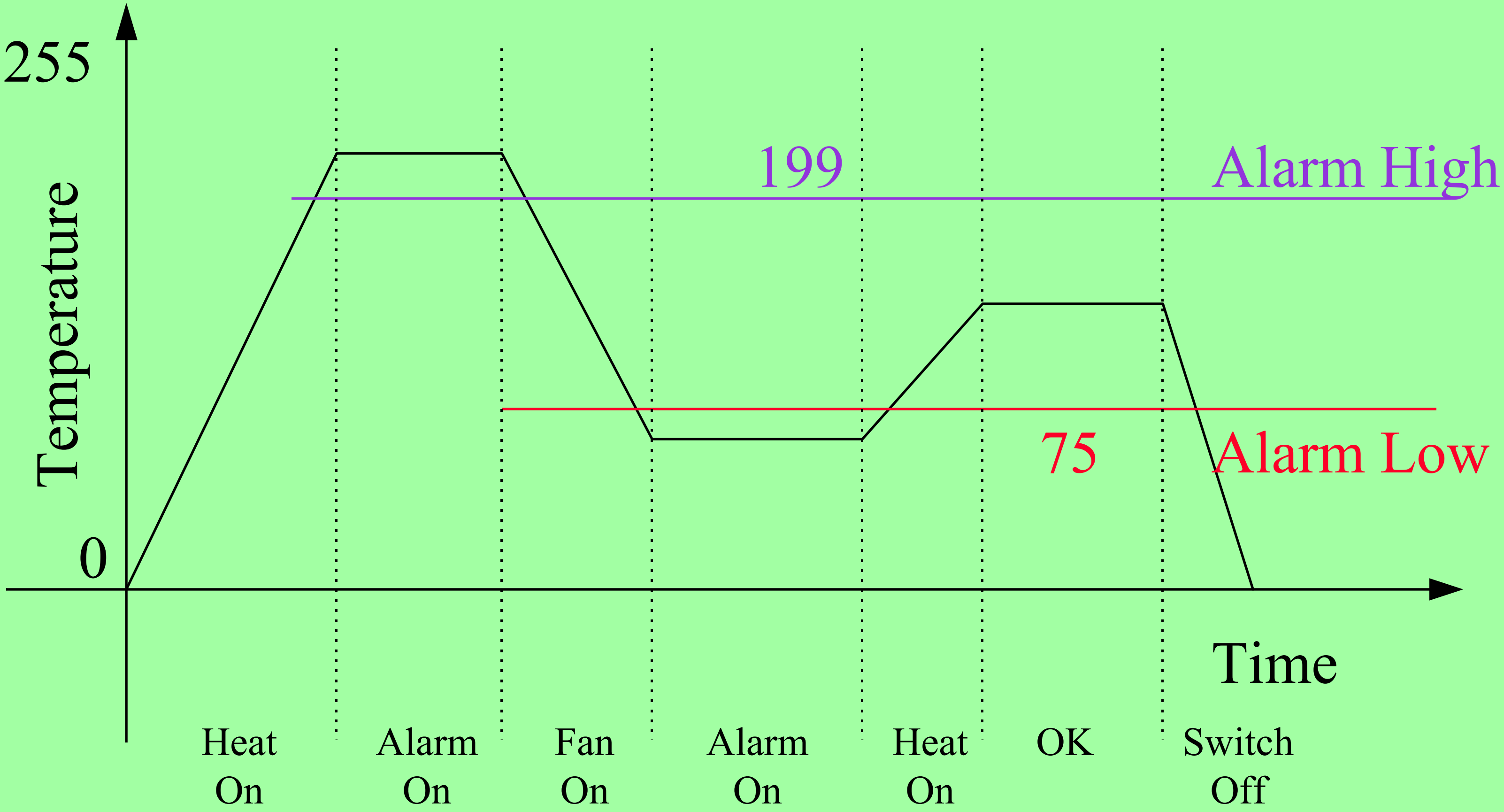
A Macro Language.

Creating test Data.

Creating Test Data.

- The Macro processor is an very effective method of creating large quantities of test data for checking the reliability of computer applications.
- The Macro processor can also add a level of intelligence into the production of the data.

Creating Test Data.



Let us assume we wish to create the data for the above situation.

Creating Test Data.

- The details
  - “Heat On” temperature rise 0 to 199.
  - “Heat On” “Alarm On” temperature rise 199 to 210.
  - “Alarm On” 50 time units.
  - “Fan On” “Alarm On” temperature fall 210 to 200.
  - “Fan On” temperature fall 199 to 76.
  - “Fan On” “Alarm On” temperature fall 75 to 51.
  - “Alarm On” 50 time units.
  - “Heat On” “Alarm On” temperature rise 50 to 74.
  - “Heat On” temperature rise 76 to 100.
  - “OK” 50 time units.
  - “Off“ temperature fall 100 to 0.

Creating Test Data.

- The Template
  - <System message><Message Terminator>
  - <Message Terminator> = <Carriage Return Code><Line Feed Code>
  - <Carriage Return Code> = 13
  - <Line Feed Code> = 10
  - <System message> = <User message> : <Temp Message>
  - <User message> = any user text you wish to enter
  - <Temp message> = TEMP=<Three Digits>
  - < Three Digits > = <Digit> <Digit> <Digit>
  - <Digit> = { 0 ¦ 1 ¦ 2 ¦ 3 ¦ 4 ¦ 5 ¦ 6 ¦ 7 ¦ 8 ¦ 9 }

Creating Test Data.

- Example messages :-
  - Heat On : TEMP=025
  - Heat On , Alarm On : TEMP=202
  - Alarm ON : TEMP=210
  - Fan On , Alarm On : TEMP=205
  - Fan On : TEMP=165
  - Alarm On : TEMP=062
  - Heat On , Alarm On : TEMP=068
  - Heat On : TEMP=082
  - OK : TEMP=100
  - OFF : TEMP=022

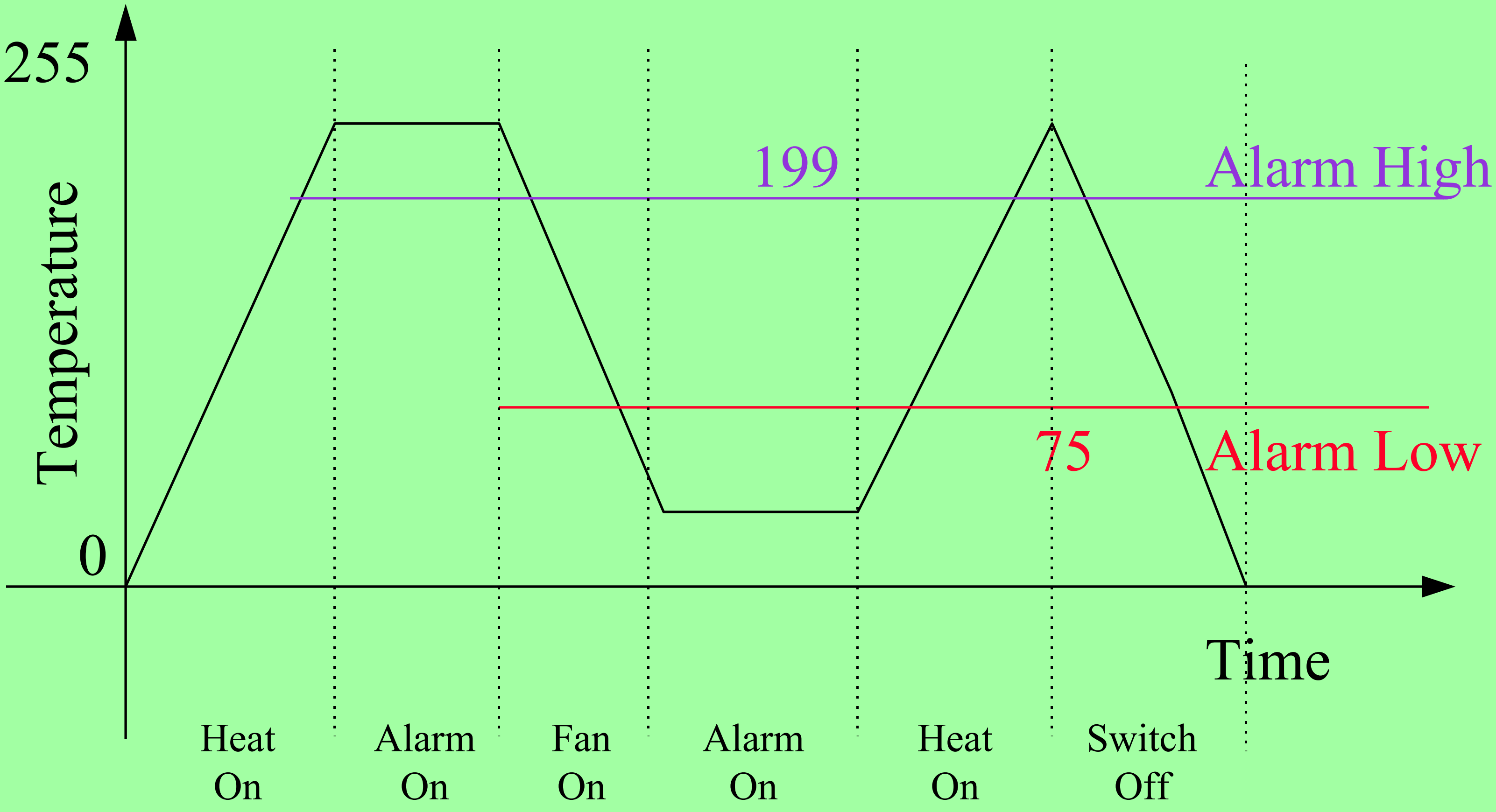
Creating Test Data.

- **Macro Requirements** :-
- The ability to generate a repeats of similar messages but with Temperature Code able to changed (incremented or decremented) if required.
- Some indicator to show if Temperature Code needs to Rise, Fall or Stay the same.
- A counter to hold the changing Temperature Code.

Creating Test Data.

- **Macro Requirements Practice**:-
- Use the definitions on the next slide as an example of a required test data wave shape.
- Create a list of expected values.
- Use the Macro Processor to create the required data.
- Demonstrate using programme Brief_4.exe that you have created the correct data.

Creating Test Data.



Let us assume we wish to modify the data to the above situation.

End Slide

Revision Page

Title	Micro Processors and Related Information
Author	R. J. Spriggs
Last Update	10/April/2006
Version	2.14
Edit	0135