NAME: NURUL AIN BT MAIZAN

MATRIK NUMBER: WEK020195

PROJECT TITLE:
WEB STIM (FLASH MEMORY SIMULATOR)

SUPERVISOR: En. NOORZAILY BIN MOHAMED NOOR

MODERATOR: En. MOHD YAMANI IDNA BIN IDRIS

NAME: NURUL AIN BT MAIZAN

MATRIK NUMBER: WEK020195

PROJECT TITLE:
WEB STIM (FLASH MEMORY SIMULATOR)

SUPERVISOR: En. NOORZAILY BIN MOHAMED NOOR

MODERATOR: En. MOHD YAMANI IDNA BIN IDRIS

ABSTRACT

Web STIM is a Web-sensor by means of interior data structure and communication functionalities and a smart sensor able to interface itself with Internet through an Ethernet network. Web STIM is designed to satisfy the requirements of low cost, continuous acquisition and data processing, and wide remote communication capability as describe in the IEEE 1451 standard. It is use HTTP interface that replace the TII (Transducer Independent Interface) hardware, which has been inspired by IEEE 1451.2 standard. This paper describes the Transducers Electronic Datasheet (TEDS) directly interfaced to internet through an HTTP based procedure. Web STIM is a web based system that can be access directly through internet connection by diffuse software on the world.

ACKNOWLEDGEMENT

First and foremost, I would like to thank Encik Noorzaily bin Mohamed Noor as my project supervisor who has been giving advices and guidelines for every questions and inquiries I asked to better understand the TEDS and STIM concept. I'm honestly appreciates his comments and suggestions to my project development. Secondly, A very big thanks to Encik Mohd Yamani Idna bin Idris as my project moderator that spend her time to moderate my project. Her suggestion and opinion during presentation give me more idea to build my project.

I would also like to thank to my parents that give a support and encourage me to do a project. To all my friends especially Yogeswari Jayaraman, Syazana Suhatta, and Marina Haryati Mohammad and also my fellow friends from FSKTM who directly or indirectly helped me to complete my project, thank you very much for the never ending support.

Last but not least I have to thank the creator and administrator of building a web site that concern with my topic that I refer. I must admit that I got a lot of ideas referring to their websites mostly regarding the understanding of a Smart Transducer Interface Module (STIM).

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1.0 INTRODUCTION

Sensor technology show that smart sensors are replacing traditional analog sensor and it is arranged into networks to share information. Currently, internet is choosing to share information at any level, it is fully supported by the market than IEEE 1451.1 Network Capable Application (NCAP). THE IEEE 1451 standard aims at simplifying transducer connectivity to existing networks, therefore the most essential improvement that smart sensors recommend is the network ability.

IEEE 1451 comes out as a set of four complete sub-standards 1451.1, 1451.2, 1451.3 and 1451.4. Each standard focused on a detailed area of the smart networked sensor signal path and be able to be used independently or as piece of an in general IEEE1451 networked system solution. The aim of this standard which is composed of four path are to enable plug and play at transducer level, standardizing data structures and communication, and to simplify the creation of networked sensor over a network independent system. Up to now, only two of the sub-standards, 1451.1 and 1451.2 have been properly adopted by IEEE, nevertheless P1451.3 and P1451.4 are still under development, thus the prefix 'P' denoting a proposed document. The 1451.1, are defines a Network Capable Application Processor (NCAP) model and also can adapt to a variety of networks. The 1451.2 is about the Smart Transducer Interface Module (STIM), which includes the transducers and the signal conditioning that may be required, a nonvolatile memory called Transducer Electronic Data Sheet

(TEDS), and the Transducer Independent Interface (TII) that enables communication between the STIM and the Network Capable Application Processor (NCAP).

Based on IEEE1451 network architecture describe that the transducers are connected to a STIM by its own port. TII (Transducer Independent Interface) joint the STIM and NCAP by using the point-to-point connection. The STIM is not connected to the internet instead of NCAP is connected.

Web STIM has become popular with the presence of cheaper, miniature and smart sensors, abundant fast and ubiquitous computing devices, wireless and mobile communication network, and autonomous and intelligent software agents. In brief it is offers full dimensional, full scale and full sensing and monitoring of Earth at all level. The web STIM is a revolutionary concept toward achieving collaborative, coherent, and consistent and consolidates sensor data collection, fusion and distribution. It is directly accessible by a browser such as Internet Explorer, Netscape Navigator. More than one client can be reached and they can share information easily if sensors are interfaced to internet. The characteristic of web STIM are interoperable that allowed different kinds of sensor need to be linked, must be intelligent that each sensor will communicate each other, and flexible that each sensors will handle various modes of data transmission and easily added new sensors.

The match of TEDS and STIM directive with powerful characteristics, simplicity and cost-effectiveness of recent web STIM are the purpose of this work.

1.1 PROJECT SCOPE AND EXPECTED OUTCOME

The scope of this project is implementing the only two of TEDS field, namely the mandatory Meta and Channel TEDS. The Meta-TEDS consists of common manufacture and specification data related to the general operation of the STIM and the Channel TEDS contain channel information specific to each of the implemented channels on the STIM. TEDS are directly connected to internet, it does not communicate with the true TII but through an HTTP based protocol.

The outcome of the project should be a web-based Java application. End users can interact with Web-STIM from anywhere in the world without the need for special equipment, communications facilities, or software where is the TEDS collected data are displayed in a binary.

1.2 PROJECT OBJECTIVE

The main objectives of the web STIM are:-

i) Study the Smart Transducer Electronic Data Sheet (STIM).

ii) Two mandatory TEDS data block will be implement are Meta and Channel TEDS.

iii) TEDS are directly connected to internet through HTTP based protocol.

iv) All the information that user require about transducer can access through web server such as Internet Explorer.

1.3 MOTIVATION

Basically, there are two main factors that motivate the study in research of web STIM. Firstly is to understand the concept of Smart Transducer Interface Module (STIM) which is it replace the traditional sensor including how the smart transducer process and also to find out why the smart sensors become smart compare to traditional sensor.

Secondly is about to realize the advantage and profit by using smart sensor in the real world. The advantages of smart sensors are a common transducer interface will lower the cost to design transducer to a set standardized digital interfaces, and having transducer electronic data sheet (TEDS) with the sensors will enable self-description of sensors and actuators, and it also simplify field installation, upgrade, and maintenance of sensors by simply "plug and play" devices to instruments and networks.

1.4 MOTIVATION LIMITATION

The limitations of this project are only implemented the TEDS part in Smart Transducer Interface Module, which is have another part are Signal Conditioning and Address Logic. TEDS are divided into eight fields, but in this project only focus on two mandatory fields are META and Channel TEDS.

1.5 CONCLUSION

In this Smart Transducer Interface Module (STIM) prototype implements the Transducer Electronic Data Sheet (TEDS) functionality defined in the 1451.2 specification. It should also be noted that my aim was to develop a basic STIM process. To this end, this project shows only the mandatory TEDS definitions are META and Channel TEDS.

2.0 LITERATURE REVIEW

2.1 INTRODUCTION

This chapter is divided into five parts, a brief introduction of IEEE1451 family, the details of IEEE1451.2 which is sub-standards of IEEE1451, a specification of Smart Transducer Functional Specification, Data type and the difference between IEEE network architecture and Web-STIM network architecture.

2.2 OVERVIEW IEEE1451

The transducer market is very diverse, so that the transducer manufacturers are looking for ways to build low-cost, networked smart transducers. The large number of networks on the market today is one of the problems faced by transducer manufacturer. It is too costly for transducer manufacturers to make unique smart transducers for each network on the market. For that reason the IEEE 1451 standard is proposed to be developed to tackle these issues.

The objective of the IEEE 1451 standard is to develop a smart transducer interface which is to make it easier for transducer manufacturers to develop smart devices and to interface those devices to networks.

The National Institute of Standards and Technology (NIST) and the Institute of Electrical and Electronics Engineers (IEEE)'s Technical Committee on Sensor Technology of the Instrumentation and Measurement Society are responsible to discus about the standard that is proposed and response to establish a common communication interface for smart transducer in September 1993.

IEEE 1451 is a family of standards and proposed standards for connecting smart transducers to networks. IEEE 1451 is divided into four sub-standards. IEEE 1451.1 defines a network independent common object model for networked smart transducers. IEEE 1451.2 defines a digital interface and communication protocol for the connection of transducers and a microcontroller. Furthermore, IEEE 1451.2 defines transducer electronic data sheet (TEDS) that describe the smart transducer (STIM) properties in a machine-readable format. The other two parts of the standard, IEEE P1451.3 and IEEE P1451.4 are not yet ratified, the prefix 'P' denoting a proposed document. IEEE P1451.3 specifies properties for distributed multi-drop systems in hazardous environments, while IEEE P1451.4 covers a mixed-mode service of analog and digital transducers, whereas analog transducers shall be enhanced with self-identification and configuration capabilities.

2.3 INTRODUCTION OF IEEE1451.2

This was the first published standard in the family. However, it has not been widely accepted in industry because of discontent with the digital communication interface and the complex software features for its interface with an NCAP.

This standard defines a STIM, TEDS, and Transducer Independent Interface (TII) for NCAP communication. Each Transducer is denoted in the STIM as a channel, and there can be up to 255 channels within one STIM. Individual channels and the STIM as a whole can be accessed by the NCAP. The different transducers that are interfaced to the STIM are triggered (sampled or set) by a command that is sent from the NCAP to the STIM. The STIM decodes this information and then sends the results back to the NCAP.

The TII is a ten wire serial connection for NCAP/STIM communication and has been widely criticized by industry because of its complexity, so as part of this standard's revision; the TII may be eliminated in HTTP based connection. Figure 1.1 shows the structure of the STIM. The TEDS supports a variety of transducers and is accessed by the digital interface (TII). The TEDS can be written by the NCAP or it can also be set at manufacture time. It resides in non-volatile memory, and contains fields that describe the type, attributes, operation, and calibration of the transducers. The TEDS is the core of the standard since it provides a method for self-identifying transducers.

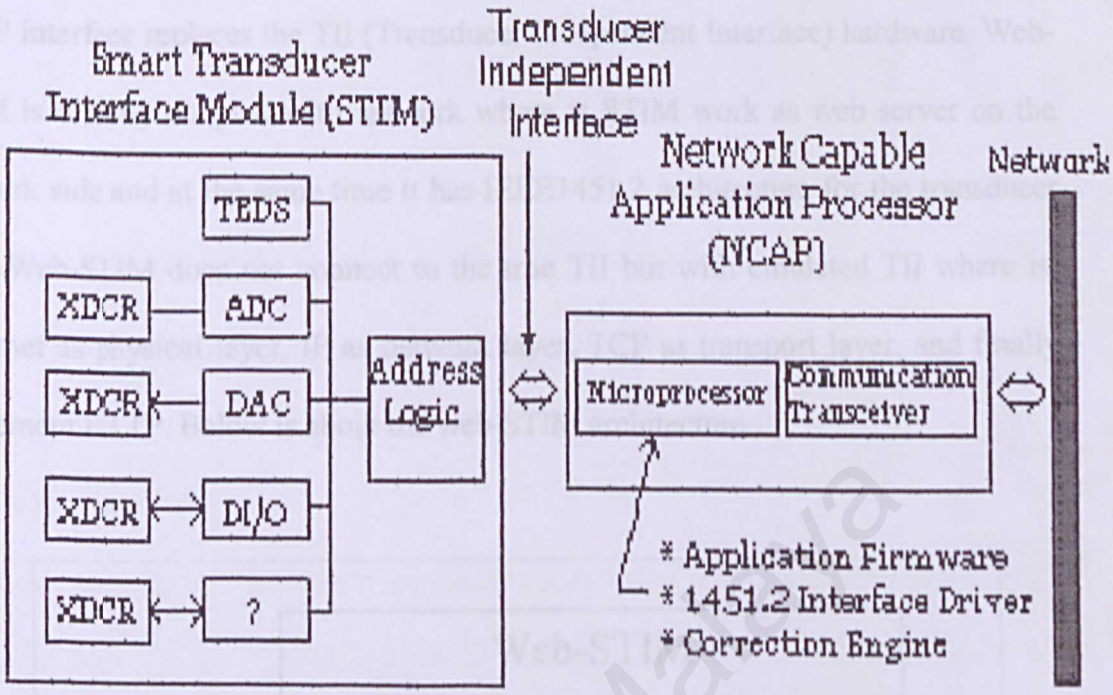


Figure 1.1: STIM block Diagram

Base on the STIM block diagram in figure 1 show that the STIM is connected with network via Network Capable Application Processor (NCAP). A STIM is controlled by a NCAP module by means a dedicated digital interface, and this interface is not a network. Physically, STIM is connected with NCAP using Transducer Independent Interface (TII) which is a 10 pin wire I/O. Base on the STIM developers to build a low cost smart sensor, they have several problems by using this communication such as the NCAP is not fully supported by market and it is expensive. So, to overcome this problem, the vendor introduces Web-STIM. Web-STIM is a Web-sensor

compatible with access procedures specified in the IEEE1451.2 standard, where HTTP interface replaces the TII (Transducer Independent Interface) hardware. Web-STIM is directly coupled with network where is STIM work as web server on the network side and at the same time it has IEEE1451.2 architecture for the transducer side. Web-STIM does not connect to the true TII but with emulated TII where is, Ethernet as physical layer, IP as network layer, TCP as transport layer, and finally implement HTTP. Below is about the web-STIM architecture.

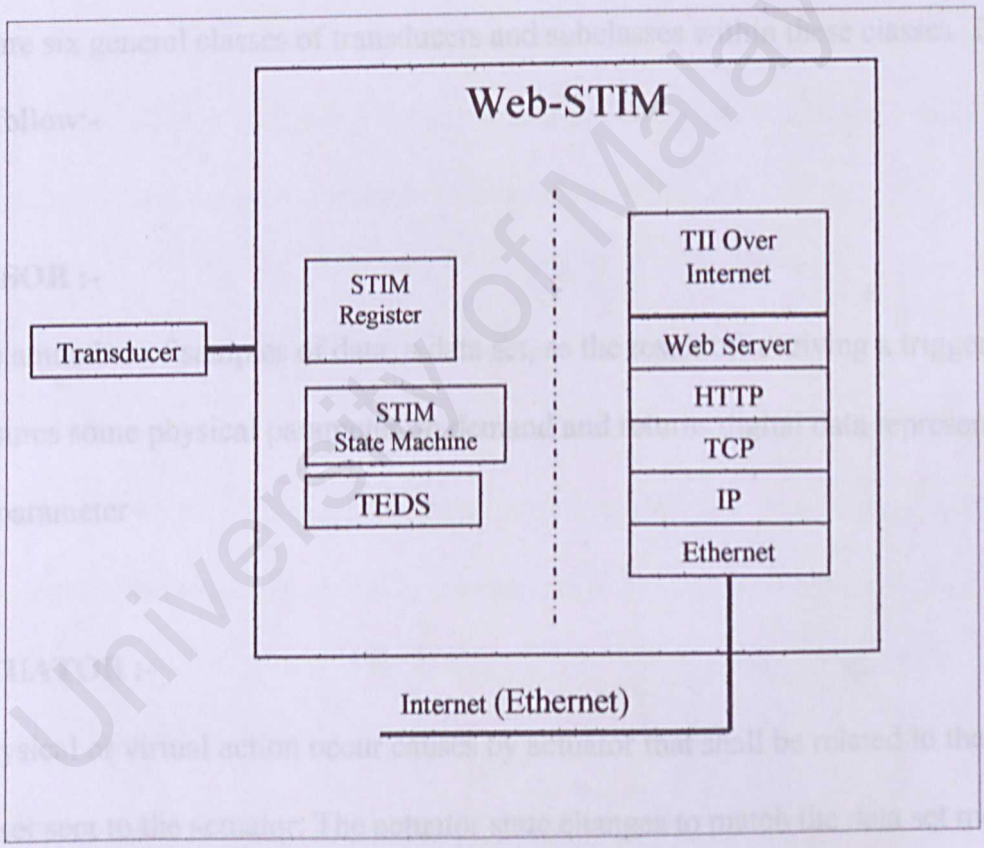


Figure 1.2: Web-STIM network architecture

2.4 SMART TRANSDUCER FUNCTIONAL SPECIFICATION

STIM is a module that consists of:-

- Transducer
- Transducer Electronic Data Sheet
- Interface logic / 1451.2 interfaces
- Signal conditioning and analog to digital conversion

2.4.1 TRANSDUCER

There are six general classes of transducers and subclasses within these classes. They are as follow:-

- **SENSOR :-**

  Take a number of samples of data, a data set, as the result of receiving a trigger. It measures some physical parameter on demand and returns digital data representing that parameter

- **ACTUATOR :-**

  A physical or virtual action occur causes by actuator that shall be related to the data set sent to the actuator. The actuator state changes to match the data set most recently sent to it when a triggering event occurs.

- **BUFFERED SENSOR :-**

  A buffered sensor has a single level of data buffering on the output channel. A new data set shall be sampled once for each triggering event. The data set available to be read shall be the data set acquired on the second most recent trigger event. The number of data points in the data set shall be determined by the Channel Data Repetitions field.

- **DATA SEQUENCE SENSOR :-**

  A data sequence sensor acquires data continuously, with sampling times under control of the STIM. Triggering selects a data set from this stream of continuous measurements and makes it available to be read by the NCAP.

- **BUFFERED DATA SEQUENCE SENSOR :-**

  A buffered data sequence sensor acquires data continuously, with sampling times under control of the STIM. Triggering selects a data set from this stream of continuous measurements and makes it available to be read by the NCAP.

- **EVENT SEQUENCE SENSOR :-**

  .A signal produced by event sequence sensor whenever a specific event occurs. The signal shall be the same signal used by sensors and actuators to acknowledge triggering events. The event may be a digital signal transition or an analog signal crossing a set point.

- **GENERAL TRANSDUCER :-**

  It allows for the presence of channels that behave on it and shall implement the same functions required for all channels

2.4.2 TRANSDUCER ELECTRONIC DATA SHEET

Transducer Electronic Data Sheet also known as TEDS. TEDS electronically correspond to information about the sensors and actuators attached to a STIM. It is to identify and describe itself to the network, thereby easing automatic system configuration. The specific technical details for data acquisition, system deployment, and on-going system maintenance are the reason why the transducer need for self-identification. The goal in developing TEDS was to offer a generalized data sheet representation of several key sensor and actuator features in a standard format that users could retrieve electronically and upgrade remotely in some cases. TEDS are divided into eight fields, each of which is used to describe different aspects of the STIM and transducer channels. Only two of the eight TEDS data block are mandatory and must remain with the STIM for the duration of its lifetime. The other six are optional and are human readable and are store as strings in one of several different character set. The two mandatory TEDS and two of the optional ones are defined as binary data format and is machine readable only. The TEDS types are:-

Table 2.1: TEDS types

Type of TEDS	TEDS Function
Meta TEDS	Contains the mandatory machine-readable data that describes the entire STIM. The data may include information such as the revision of the IEEE standard, version of the TEDS, number of channels and timing restriction.
Channel TEDS	Contains the mandatory machine-readable data that describe each transducer channel in the STIM. The data may include information such as the transducer type, calibration model, physical units, limits range, data format and the timing restriction for the relevant transducer channel.
Meta-Identification TEDS	Provides the optional human-readable (Text/ASCII) data for the overall STIM. Data may include information such as manufacturer's name, model number, serial number, version codes, date codes and product description.
Calibration TEDS	Contains the optional machine-readable data when a correction engine is used in the STIM. The data may include information such as the calibration

	coefficients, intervals, date and time for the each transducer channel that requires calibration.
Channel-Identification TEDS	Provides the optional human-readable data similar to Meta-Identification TEDS, except that it is for an individual channel. This data is used when a STIM is built with multi-channel transducers from different manufacturers.
Calibration-Identification TEDS	Provides the optional human-readable data when a correction engine is used in the STIM. The data may include information such as the calibrator id and the calibration standard used.
End-User's Application-Specific TEDS	Provides the additional human-readable data not covered by the specific TEDS described above. The data may include information such as the location of the STIM and the contact information for the technical inquiry.
Generic Extensions TEDS	Allows an option for the future extension to the TEDS described above

2.5 EXISTING SYSTEM

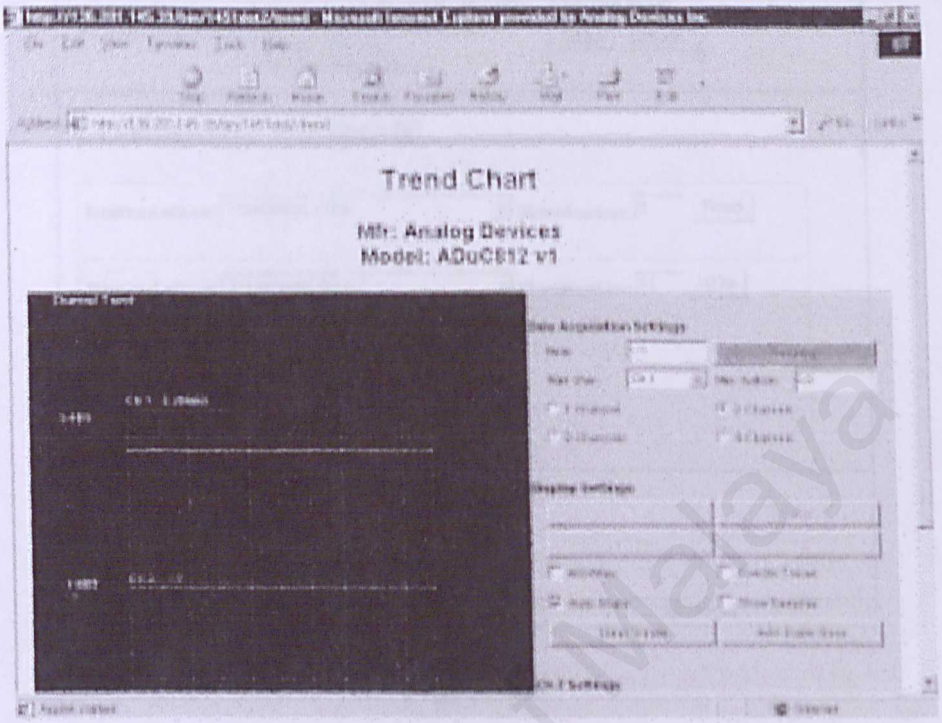


Figure 2.1

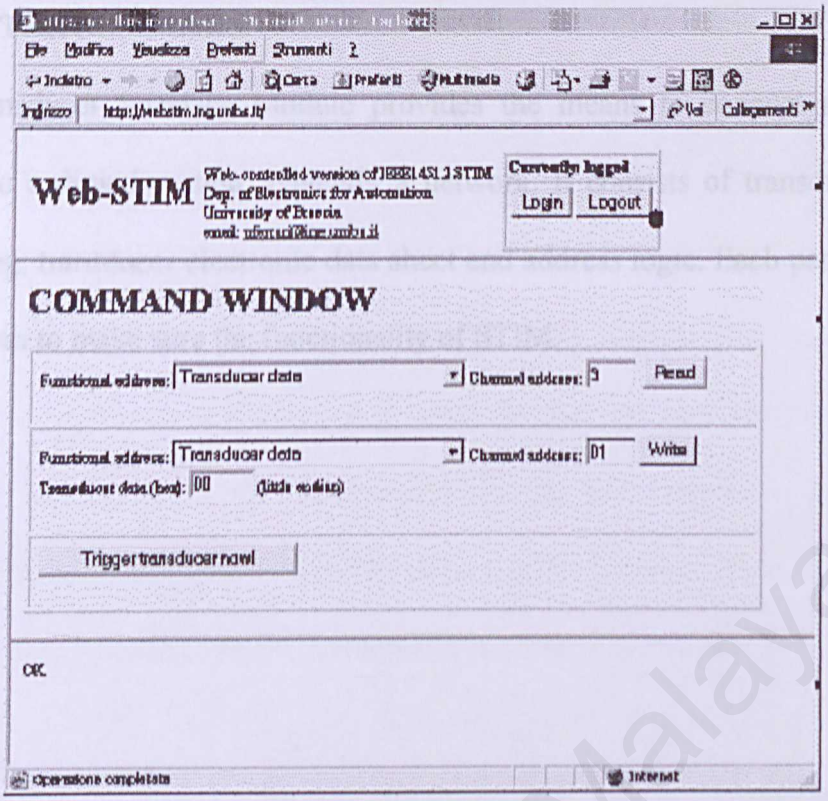


Figure 2.2

Figure 2.1 and figure 2.2 are the existing STIM system that can be view using internet. The differences between these STIM systems are how they are connected with the network. The STIM system in figure 2 use Transducer Independence Interface (TII) to connect with Network Capable Application Processor then the NCAP is connected to network, and the STIM system in figure 3 use the Web-STIM concept to connect with internet. They also use the different language where is, the figure 1 use the C language and the figure 2 use the JAVA language. Both of the STIM system implements the mandatory Transducer Electronic Data Sheet are META and Channel TEDS to develop a basic STIM process.

2.6 CHAPTER SUMMARY

Smart Transducer Interface Module provides the means to connect sensors and actuators to a digital system, typically a network. It consists of transducer, signal conditioning, transducer electronic data sheet and address logic. Each part have their own function to make sure the functionality of STIM.

3.0 METHODOLOGY

3.1 OBJECT ORIENTED PARADIGM

Donald Firesmith in his book Dictionary of Object Technology (SIGS Books, 1995), explain that Object Oriented analysis is "the discovery, analysis and specification of requirements in terms of objects with identity that encapsulate properties and operations, message passing, classes, inheritance, polymorphism and dynamic binding." Firesmith also states that OO design is "the design of an application in terms of objects, classes, clusters, frameworks and their interactions."

This project used Object Oriented Analysis and Design methodologies. The methodologies has two basic types are ternary (three-pronged) and unary. The ternary type is the natural evolution of existing structured methods and has three separate notations for data, dynamics, and process. The unary type asserts that because objects combine processes (methods) and data, only one notation is needed. The unary type is considered to be more object-like and easier to learn from scratch, but has the disadvantage of producing output from analysis that may be impossible to review with users.

There are almost two dozen major object-oriented programming languages in use today such as C++, Smalltalk, and Java. I prefer to use Java language. Object oriented programming gives us a natural and intuitive way to view the programming

process, namely, by modeling real world objects, their attributes and their behavior. It also provides for communication among objects. The main advantages of OO programming is its ease of modification which is objects can easily be modified and added to a system there by reducing maintenance costs. OO programming is also considered to be better at modeling the real world than is procedural programming. It allows for more complicated and flexible interactions.

3.2 JAVA PROGRAMMING PLATFORM AND LANGUAGE

Java is an object-oriented programming language with a built-in application programming interface (API) that can handle graphics and user interfaces and that can be used to create applications or applets.

The following are the version of Java:-

- JDK 1.02 (obsolete)
- JDK 1.1.x (obsolete)
- J2SE v1.2.x (also called Java2 SDK, includes Swing)
- J2SE v1.4.x (the latest, no big changes vs. 1.2)

3.2.1 J2SE V1.4.2

The java 2 platform, standard edition is at the core of Java technology, and version 1.4 raises the Java platform to a higher standard. From client to server, from desktop to supercomputer, improvements have been made to J2SE across the board. With the

version 1.4, enterprise can now use Java technology to develop more demanding business applications with less effort and in less time.

Version 1.4 builds upon the current J2SE platform and provides even more features for developers to build into their applications. More functionality in 1.4 developers can now spend less time writing custom code to accomplish what is now part of the core J2SE platform. The result is faster application programming with more consistency for enterprise development initiatives. New features in J2SE 1.4 also reduce the developer's reliance on other technologies such as C, C++, PERL, or SSL and DOM implementation in browsers. This allows the developers to use a single technology to develop, test, and deploy end-to-end enterprise development

3.3 SYSTEM ARCHITECTURE

System architecture of this project is web-based application.

3.3.1 WEB-BASED APPLICATION

Web based applications are developed and being executed on the internet. It involves the participations of servers and clients. Clients can access to permitted information from the server even though he is in a distance using the World Wide Web.

Advantages of web-based applications are:-

1. accessible from almost anywhere with internet access
2. information updating is easy as it is done on the server side

3. requires common software on the client side, web browsers
4. various type of information can be stored with various methods

The advantages of web based applications are:-

1. hard to be maintained with complicated programming
2. have to maintain a server
3. vulnerable to hackers even though password protected
4. information loaded to the web is very limited in term of size

3.3 CHAPTER SUMMARY

Object oriented approach offers a new and powerful model for developing web-based which is objects is "black box" which send and receive messages. This approach speeds the development of new programs, and, if properly used, improves the maintenance, reusability, and modifiability of software.

The Java language provides a powerful addition to the tools. Java makes programming easier because it is object-oriented and has automatic garbage collection. In addition, because compiled Java code is architecture-neutral, Java applications are ideal for a diverse environment like the Internet.

4.0 ANALYSIS

4.1 FUNCTIONAL REQUIREMENT

Functional requirements are statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situation. In some cases, it also stated what the system should not do.

4.1.1 TRANSDUCER ELECTRONIC DATA SHEET (TEDS)

The TEDS contains field that fully describe the type, operation, attributes, specific technical details useful for data acquisition, system deployment, and on-going system maintenance and stores all the information a user might need to know about a particular transducer. A transducer must be self-aware to be really smart, and this is where the TEDS come in.

Each STIM must have an area of memory written in a defined electronic format that describes the STIM itself and any transducer channels associated with it.

TEDS are divided into eight fields, each of which is used to describe different aspects of the STIM and transducer channels. In this paper, only the mandatory TEDS are supported in this implementation (one META and two channel TEDS). The channels are META TEDS and Channel TEDS for transducers (sensor and

actuator). The data type for each Channel TEDS is channel 1 for a sensor and channel 2 for actuator. Thus the data model for each channel is integer.

4.1.2 META TEDS DATA BLOCK

META TEDS function contains an overall description needed to gain access to any transducer and information common to all transducer such as version of the TEDS, number of channels and timing parameters. The bytes of Meta-TEDS are constant and read only.

Field types

U8, U16, U32 – are unsigned integers of length 8, 16, and 32 bits respectively.

F32 – is a single precision IEEE floating point number.

STRING – is an array of character bytes

UNITS – is the representation.

L is for length, E is for enumeration and C for counting.

The data structure of the **META TEDS** is shown in Table 4.1 based on IEEE standard.

Table 4.1: META TEDS Data Structure

Field No.	Description	Type	No. of byte
TEDS version constant related data sub-block			
1	Meta-TEDS Length	U32L	4
2	IEEE 1451 Standards Family Working Group Number	U8E	1
3	TEDS Version Number	U8E	1
Identification related data sub-block			
4	Globally Unique Identifier	UUID	10
Data structure related data sub-block			
5	CHANNEL_ZERO Industry Calibration TEDS Extension Key	U8E	1
6	CHANNEL_ZERO Industry Nonvolatile Data Fields Extension Key	U8E	1
7	CHANNEL_ZERO industry TEDS extension key	U8E	1
8	CHANNEL_ZERO End-Users' Application-Specific TEDS key	U8E	1
9	Number of Implemented Channels	U8C	1
10	Worst-Case Channel Data Model Length	U8C	1
11	Worst-Case Channel Data Repetitions	U16C	2
12	CHANNEL_ZERO writable TEDS length	U32C	4
Timing related data sub-block			
13	Worst-Case Channel Update Time (twu)	F32	4
14	Global Write Setup Time (tgws)	F32	4
15	Global Read Setup Time (tgrs)	F32	4
16	Worst-Case Channel Sampling Period (twsp)	F32	4
17	Worst-Case Channel Warm-Up Time	F32	4
18	Command Response Time	F32	4
19	STIM Handshake Timing (ths)	F32	4
20	End-Of-Frame Detection Latency (tlat)	F32	4
21	TEDS Hold-Off Time (tth)	F32	4
22	Operational Hold-Off Time (toh)	F32	4
23	Maximum Data Rate	U32C	4
Channel grouping related data sub-block			
24	Channel Groupings Data Sub-block Length	U16L	2
25	Number of Channel Groupings = G	U8C	1

The following is about the explanations of each data field in the data structure of the META TEDS. The first 6 bytes or the first three fields of the META TEDS data structure shall never modify in any subsequent TEDS version. The data structure consists of:-

4.1.2.1 META-TEDS LENGTH

Meta-TEDS data field number 1

Data type: unsigned integer used for field length (U32L, 4 bytes)

This field specifies the total number of bytes in the Meta-TEDS data block excluding this field.

4.1.2.2 IEEE 1451 STANDARDS FAMILY WORKING GROUP NUMBER

Meta-TEDS data field number 2

Data type: unsigned byte integer used for enumeration (U8E, 1 byte)

This field shall be set to two for devices conforming to this standard. This field shall be used by other members of the IEEE 1451 standards family to indicate to an NCAP that a different data structure follows.

4.1.2.3 TEDS VERSION NUMBER

Meta-TEDS data field number 3

Data type: unsigned byte integer used for enumeration (U8E, 1 byte)

The version number of the TEDS that corresponds to the particular IEEE 1451 standard of the working group that specifies the TEDS data structure as shown in Table 4.2.

Table 4.2: Enumeration of TEDS Version Numbers

TEDS version	IEEE 1451.2 standard version
0	Reserved
1	This will correspond to the first official version of the standard: IEEE Std 1451.2-1997.
2 – 225	Reserved

4.1.2.4 GLOBALLY UNIQUE IDENTIFIER

Meta-identification TEDS data field number 4

Data type: Universally unique identification (UUID, 10 bytes)

The UUID field is provided to allow better management of STIM components in a distributed system (e.g., tracking and traceability of STIMs for operational and maintenance purposes). The UUID must be guaranteed to be unique in the universe of all STIMs.

4.1.2.5 CHANNEL_ZERO INDUSTRY CALIBRATION TEDS EXTENSION KEY

Meta-TEDS data field number 5

Data type: unsigned byte integer used for enumeration (U8E, 1 byte)

The value in this field indicates the highest functional address for writing the industry-implemented Calibration TEDS extension that is available in the STIM for CHANNEL_ZERO. Acceptable values and their meanings are defined in Table 4.3.

Table 4.3: Enumeration of CHANNEL_ZERO Industry Calibration TEDS Extension Keys

Key Value (K)	Meaning
0	No extensions implemented in STIM
1 – 79	Invalid
80 – 95	Valid, TEDS extension(s) implemented for: • Functional addresses used for writing: between 80 and (K); and • Functional addresses used for reading: between 208 and (K+128)
96 – 255	Invalid

4.1.2.6 CHANNEL_ZERO INDUSTRY NONVOLATILE DATA FIELDS EXTENSION KEY

Meta-TEDS data field number 6

Data type: unsigned byte integer used for enumeration (U8E, 1 byte)

The value in this field indicates the highest functional address for writing the industry-implemented nonvolatile data field extensions that is available in the STIM for CHANNEL_ZERO. Acceptable values and their meanings are defined in Table 4.4.

Table 4.4: Enumeration of CHANNEL_ZERO Industry Nonvolatile Data Fields Extension Keys

Key value (K)	Meaning
0	No extensions implemented in STIM
1 - 111	Invalid
112 - 127	Valid, TEDS extension(s) implemented for: • Functional addresses used for writing: between 112 and (K); and • Functional addresses used for reading: between 240 and (K+128)
128 - 255	Invalid

4.1.2.7 CHANNEL_ZERO INDUSTRY TEDS EXTENSION KEY

Meta-TEDS data field number 7

Data type: unsigned byte integer used for enumeration (U8E, 1 byte)

The value in this field indicates the highest functional address for writing the industry-implemented TEDS extensions that is available in the STIM for CHANNEL_ZERO. Acceptable values and their meanings are defined in Table 4.5.

Table 4.5: Enumeration of CHANNEL_ZERO Industry TEDS Extension Keys

Key value (K)	Meaning
0	No extensions implemented in STIM
1 - 175	Invalid
176 - 191	Valid, TEDS extension(s) implemented for: • Functional addresses used for reading: between 176 and (K)
192 - 255	Invalid

4.1.2.8 CHANNEL_ZERO END-USERS' APPLICATION-SPECIFIC TEDS KEY

Meta-TEDS data field number 8

Data type: unsigned byte integer used for enumeration (U8E, 1 byte)

This field indicates the presence of End-Users' Application-Specific TEDS function in CHANNEL_ZERO as defined in Table 4.6.

Table 4.6: Enumeration of End-Users' Application-Specific TEDS Keys

Key value (K)	Meaning
0	End-Users' Application-Specific TEDS is not implemented on CHANNEL_ZERO.
1	End-Users' Application-Specific TEDS is implemented on CHANNEL_ZERO.
2 - 255	Reserved.

4.1.2.9 NUMBER OF IMPLEMENTED CHANNELS

Meta-TEDS data field number 9

Data type: unsigned byte integer used for counting (U8C, 1 byte)

Specifies the number of channels implemented in the STIM. If the number is one, this shall be a single variable transducer. Numbers greater than one identify multiple variable transducers, perhaps consisting of both sensor and actuator elements. There can be up to 255 channels on a STIM, thus the value of this field shall be M such that 1 □M□255.

A STIM can give TEDS without having to produce data. This is specified by setting the following Channel

TEDS fields:

- Channel Data Model to "N-byte"
- Channel Data Model Length to zero
- Channel Model Significant Bits to zero
- Channel Data Repetitions to zero

For details on these Channel TEDS fields, see channel data structure.

4.1.2.10 WORST-CASE CHANNEL DATA MODEL LENGTH

Meta-TEDS data field number 10

Data type: unsigned byte integer used for counting (U8C, 1 byte)

This field specifies the maximum value of the Channel Data Model Length for all the implemented channels.

4.1.2.11 WORST-CASE CHANNEL DATA REPETITIONS

Meta-TEDS data field number 11

Data type: unsigned 16 bit integer used for counting (U16C, 2 bytes)

This field specifies the maximum value of the Channel Data Repetitions for all the implemented channels.

4.1.2.12 CHANNEL_ZERO WRITABLE TEDS LENGTH

Meta-TEDS data field number 12

Data type: unsigned 32 bit integer used for counting (U32C, 4 bytes)

This field specifies the length in bytes available for each CHANNEL_ZERO user-writable TEDS. The only structure currently defined in this standard is the CHANNEL_ZERO End-Users' Application-Specific TEDS. An entire writable TEDS, including the length field and checksum, must fit within this maximum length.

4.1.2.13 WORST-CASE CHANNEL UPDATE TIME

Meta-TEDS data field number 13

Data type: single-precision real (F32, 4 bytes)

This field specifies the maximum value of the Channel Update Time (twu) for all the implemented channels in seconds. For a STIM without enabled event sequence channels, this parameter is used to determine if a STIM is failing to respond to a global trigger. If such a STIM is fully functional, the time between trigger and trigger acknowledge shall never exceed this time. For a STIM with at least one enabled event sequence sensor, this parameter indicates the additional time that must pass after a global trigger acknowledgment before all other channels may be assumed to have acknowledged the virtual triggering associated with the event.

4.1.2.14 GLOBAL WRITE SETUP TIME

Meta-TEDS data field number 14

Data type: single-precision real (F32, 4 bytes)

This field specifies the minimum time (tgws), in seconds, between the end of a global write frame and the application of a global trigger.

4.1.2.15 GLOBAL READ SETUP TIME

Meta-TEDS data field number 15

Data type: single-precision real (F32, 4 bytes)

This field specifies the minimum time (tgrs), in seconds, between the receipts of a global trigger acknowledge and the beginning of a global read frame. For STIMs with enabled event sequence sensors, the NCAP shall wait for the duration of the Worst-Case Channel Update Time plus the Global Read Setup Time before beginning a global read frame.

4.1.2.16 WORST-CASE CHANNEL SAMPLING PERIOD

Meta-TEDS data field number 16

Data type: single-precision real (F32, 4 bytes)

This field specifies the maximum value (twsp), in seconds, of the channel sampling period for all implemented channels.

4.1.2.17 WORST-CASE CHANNEL WARM-UP TIME

Meta-TEDS data field number 17

Data type: single-precision real (F32, 4 bytes)

This field specifies the minimum time, in seconds, that is necessary between application of power to the STIM and instigation of the first transducer data transfer. This is the maximum value of all the Channel Warm-Up Times.

4.1.2.18 COMMAND RESPONSE TIME

Meta-TEDS data field number 18

Data type: single-precision real (F32, 4 bytes)

This field specifies the longest time, in seconds.

4.1.2.19 STIM HANDSHAKE TIME

Meta-TEDS data field number 19

Data type: single-precision real (F32, 4 bytes)

This field specifies the longest time (ths), in seconds, for the STIM to remove the trigger acknowledge signal after the trigger signal is removed by the NCAP, or for the STIM to remove the data transport acknowledge signal after the data transport is inactivated by the NCAP.

4.1.2.20 END-OF-FRAME DETECTION LATENCY

Meta-TEDS data field number 20

Data type: single-precision real (F32, 4 bytes)

This field specifies the longest time (tlat), in seconds, that a STIM shall take to detect the removal of the data transport enable signal.

4.1.2.21 TEDS HOLD-OFF TIME

Meta-TEDS data field number 21

Data type: single-precision real (F32, 4 bytes)

This field specifies the maximum individual hold-off time, in seconds, imposed by the STIM before the first byte, or between bytes, of any data transfer addressed to TEDS functions, (i.e., functional addresses in the ranges of 32-127 or 160-255, inclusive).

4.1.2.22 OPERATIONAL HOLD-OFF TIME

Meta-TEDS data field number 22

Data type: single-precision real (F32, 4 bytes)

This field specifies the maximum individual hold-off time, in seconds, imposed by the STIM before the first byte, or between bytes, of any data transfer addressed to operational functions, (i.e., functional addresses in the ranges of 1-31 or 129-159, inclusive).

4.1.2.23 MAXIMUM DATA RATE

Meta-TEDS data field number 23

Data type: unsigned 32 bit integer used for counting (U32C, 4 bytes)

This field specifies the maximum data rate, in bits per second, supported by the STIM interface.

4.1.2.24 CHANNEL GROUPINGS DATA SUB-BLOCK LENGTH

Meta-TEDS data field number 24

Data type: unsigned 16 bit integer used for field length (U16L, 2 bytes)

This field specifies the total number of bytes in the Channel Grouping data sub-block. The Channel Groupings Data Sub-Block Length field shall not include the length of the length field itself.

4.1.2.25 NUMBER OF CHANNEL GROUPINGS

Meta-TEDS data field number 25

Data type: unsigned byte integer used for counting (U8C, 1 byte)

This field specifies the number of discrete channel groupings defined in this STIM's Meta-TEDS. The subsequent fields in the channel grouping data sub-block shall be repeated in that order for the Number of Channel Groupings.

4.1.2.26 GROUP TYPE

Meta-TEDS data field number 26

Data type: unsigned byte integer used for enumeration (U8E, 1 byte)

The relationship between the channels comprising the specific group shall be defined by the enumeration in Table 4.7. The arbitrary relation, when the enumeration value is equal to zero, shall be used to convey grouping semantics not specifically enumerated but deemed necessary by the transducer manufacturer, for the correct operation or interpretation of the data related to the channels that are members of the group. The arbitrary relation may be used to redundantly convey, in a more compact form than the Calibration TEDS fields. That correct behavior of channels with coupling terms in the calibration assumes that all channels involved are triggered at the same time.

Table 4.7: Enumeration of Group Types

Value	Meaning
0	An arbitrary relation
1	x, y, z right-hand rectangular spatial coordinates
2	r, f, z, right-hand cylindrical spatial coordinates
3	r, q, f right-hand spherical spatial coordinates
4	Latitude, longitude, altitude planetary coordinates
5	In-phase, quadrature temporal coordinates
6	Red, green, blue color coordinates
7	Analog event sequence sensor channel, analog input sensor channel, upper threshold virtual actuator channel,

	hysteresis virtual actuator channel
8	Sensor channel (any type), high-pass filter virtual actuator channel, low-pass filter virtual actuator channel, scale factor virtual actuator channel
9	Transducer (any type), sample interval virtual actuator channel
10	Digital event sequence sensor channel, digital input sensor channel, event pattern virtual actuator channel
11 - 127	Reserved for future expansion
128 – 255	Open to industry

To identify the virtual actuator channels, enumerations 7 and 10 may be used to set up an event sequence sensor. Besides, they also identify a sensor channel that may be used to read the level of the signal in an analog event sequence sensor or the current pattern input to a digital event sequence sensor.

Enumeration 8 may be used to identify the virtual actuator channels used to set the high-pass filter, low-pass filter, and scale factor associated with a sensor of any type.

Enumeration 9 may be used to identify the virtual actuator channels used to set the channel sampling period in a data sequence sensor or buffered sequence sensor. It may also be used to set the channel sampling period in sensors, buffered sensors, and actuators with Channel Data Repetitions greater than zero and Series Increment and Series Units indicating that a time sequence of samples is to be processed on a trigger.

4.1.2.27 NUMBER OF GROUP MEMBERS

Meta-TEDS data field number 27

Data type: unsigned byte integer used for counting (U8C, 1 byte)

This field specifies the number of channels comprising the specific group.

4.1.2.28 MEMBER CHANNEL NUMBERS LIST

Meta-TEDS data field number 28

Data type: an array of unsigned byte integers used for enumeration (U8E, 0 to 255 bytes)

This field specifies a one-dimensional array (list) of 1 byte elements. Each element is the channel address for a member channel in the specific group. The values of the elements in this list shall be in the sequence specified by the group type. An element with value zero shall indicate that the transducer does not implement this particular element of the enumerated relationship. For example, a two-axis vector measurement implemented by channels 1, x, and 2, y, may be specified by designating the Group Type (5.1.3.26) as 1 (x, y, z) with the Member Channel Numbers List (1, 2, 0). The value zero shall not appear in the Member Channel Numbers List for a group of group type "arbitrary relation".

Note that a channel can be represented in multiple groups.

4.1.2.29 CHECKSUM FOR META-TEDS

Meta-TEDS data field number 29

Data type: unsigned 16 bit integer used for counting (U16C, 2 bytes)

This field contains the checksum for the complete Meta-TEDS data block. The checksum shall be the one's complement of the sum (modulo 216) of all the data structure's preceding bytes, including the initial length field and excluding the checksum field.

4.1.3 CHANNEL TEDS DATA BLOCK

Channel TEDS function is to make available at the interface all of the information concerning the channel being addressed to enable the proper operation of the channel. Channel TEDS bytes are constant and read-only.

Table 4.8: Data structure of Channel TEDS data block

Field No.	Description	Type	No. of bytes
Data structure related data sub-block			
1	Channel TEDS Length	U32L	4
2	Calibration Key	U8E	1
3	Channel Industry Calibration TEDS Extension Key	U8E	1
4	Channel Industry Nonvolatile Data Fields Extension Key	U8E	1
5	Channel Industry TEDS Extension	U8E	1

	Key		
6	Channel End-Users' Application-Specific TEDS Key	U8E	1
7	Channel Writable TEDS Length	U32C	4
Transducer related data sub-block			
8	Channel Type Key	U8E	1
9	Physical Units	UNITS	1
10	Lower Range Limit	F32	4
11	Upper Range Limit	F32	4
12	Worst-Case Uncertainty	F32	4
13	Self-Test Key	U8E	1
Data converter related data sub-block			
14	Channel Data Model	U8E	1
15	Channel Data Model Length	U8C	1
16	Channel Model Significant Bits	U16C	2
17	Channel Data Repetitions	U16C	2
18	Series Origin	F32	4
19	Series Increment	F32	4
20	Series Units	UNITS	10
Timing related data sub-block			
21	Channel Update Time (tu)	F32	4
22	Channel Write Setup Time (tws)	F32	4
23	Channel Read Setup Time (trs)	F32	4
24	Channel Sampling Period (tsp)	F32	4
25	Channel Warm-Up Time	F32	4
26	Channel Aggregated Hold-Off Time (tch)	F32	4
27	Timing Correction	F32	4
28	Trigger Accuracy	F32	4
Event sequence options field			

29	Event Sequence Options	U8E	1
Data integrity data sub-block			
30	Checksum for Channel TEDS	U16C	2

Each data field in the data structure is describing below:-

4.1.3.1 CHANNEL TEDS LENGTH

Channel TEDS data field number 1

Data type: unsigned 32 bit integer used for counting (U32L, 4 bytes)

This field specifies the total number of bytes in the channel TEDS data block excluding this field.

4.1.3.2 CALIBRATION KEY

Channel TEDS data field number 2.

Data type: unsigned byte integer used for enumeration (U8E, 1 byte)

Table 4.9 defined the calibration capabilities of the STIM

Table 4.9: Enumeration of Calibration Keys

Value	Name	Function
0	CAL _NONE	No calibration information needed or provided. No correction is performed by the NCAP on transducer data associated with this channel. If the value is CAL_NONE, this implies that there is no calibration

		TEDS associated with this block. If the Calibration TEDS is accessed, the Calibration TEDS Length shall be zero.
1	CAL_FIXED	Fixed calibration information provided. This information cannot be modified. Correction is performed in the NCAP or elsewhere in the system.
2	CAL_MODIFIABLE	Calibration information provided. This information can be modified by writing to the Calibration TEDS. Correction is performed in the NCAP or elsewhere in the system.
3	CAL_SELF	Calibration information provided. Adjusted by a self-calibration capability. Correction is performed in the NCAP or elsewhere in the system.
4	CAL_CUSTOM	Calibration information is provided through an industry extension. Correction is performed in the NCAP or elsewhere in the system.
5	STIM_CAL_FIXED	Fixed calibration information is provided to be applied in the STIM. This information cannot be modified. See 5.2.3.2.2 for additional details.
6	STIM_CAL_MODIFIABLE	Calibration information is provided to be applied in the STIM. This information can be

		modified by writing to the Calibration TEDS. See 5.2.3.2.2 for additional details.
7	STIM_CAL_SELF	Calibration information is provided to be applied in the STIM. Adjusted by a self-calibration capability. See 5.2.3.2.2 for additional details.
8 – 225	Reserved	Reserved for future expansion.

4.1.3.2.1 NCAP CORRECTIONS

Calibration key enumerations CAL_FIXED, CAL_MODIFIABLE, CAL_SELF, and CAL_CUSTOM are to be used when the correction is performed in the NCAP or elsewhere in the system.

4.1.3.2.2 STIM CORRECTIONS

Calibration key enumerations STIM_CAL_FIXED, STIM_CAL_MODIFIABLE, and STIM_CAL_SELF are to be used when the correction is performed in the STIM using the correction method and information stored in the Calibration TEDS.

4.1.3.3 CHANNEL INDUSTRY CALIBRATION TEDS EXTENSION KEY

Channel TEDS data field number 3.

Data type: unsigned byte integer used for enumeration (U8E, 1 byte)

The value in this field indicates the highest functional address for writing the industry-implemented Calibration TEDS extension that is available in the STIM for this channel. Acceptable values and their meanings are defined in Table 4.10.

Table 4.10: Enumerations of Channel Industry Calibration TEDS Extension Keys

Key Value (K)	Meaning
0	No extensions implemented in STIM
1-79	Invalid
80-95	Valid, TEDS extension(s) implemented for: Functional addresses used for writing: between 80 and (K); and Functional addresses used for reading: between 208 and (K+128)
96-255	Invalid

4.1.3.4 CHANNEL INDUSTRY NONVOLATILE DATA FIELDS EXTENSION KEY

Channel TEDS data field number 4

Data type: unsigned byte integer used for enumeration (U8E, 1 byte)

The value in this field indicates the highest functional address for writing the industry-implemented nonvolatile data field extensions that is available in the STIM for this channel. Acceptable values and their meanings are defined in Table 4.11.

Table 4.11: Enumerations of Channel Industry Nonvolatile Data Fields Extension Keys

Key Value (K)	Meaning
0	No extensions implemented in STIM
1-111	Invalid
112-117	Valid, TEDS extension(s) implemented for: • Functional addresses used for writing: between 112 and (K); and • Functional addresses used for reading: between 240 and (K+128)
128-125	Invalid

4.1.3.5 CHANNEL INDUSTRY TEDS EXTENSION KEY

Channel TEDS data field number 5

Data type: unsigned byte integer used for enumeration (U8E, 1 byte)

The value in this field indicates the highest functional address for writing the industry-implemented TEDS extensions that is available in the STIM for this channel. Acceptable values and their meanings are defined in Table 4.12.

Table 4.12: Enumerations of Channel Industry TEDS Extension Keys

Key Value (K)	Meaning
0	No extensions implemented in STIM
1-175	Invalid
176-191	Valid, TEDS extension(s) implemented for: • Functional addresses used for reading: between 176 and (K)
192-255	Invalid

4.1.3.6 CHANNEL END-USERS' APPLICATION-SPECIFIC TEDS KEY

Channel TEDS data field number 6

Data type: unsigned byte integer used for enumeration (U8E, 1 byte)

This field indicates the presence of End-Users' Application-Specific TEDS function for this channel as defined in Table 4.13.

Table 13: Enumeration of End-Users' Application-Specific TEDS Keys

Key Value (K)	Meaning
0	End-Users' Application-Specific TEDS Function Is Not Implemented On This Channel.
1	End-Users' Application-Specific TEDS function is implemented on this channel.
2-255	Reserved

4.1.3.7 CHANNEL WRITABLE TEDS LENGTH

Channel TEDS data field number 7

Data type: unsigned 32 bit integer used for counting (U32C, 4 bytes)

This field specifies the length in bytes available for each individual user-writable TEDS associated with this channel, such as Calibration TEDS, Calibration Identification TEDS, or End-User's Application-Specific TEDS. An entire writable TEDS, including the length field and checksum, must fit within this maximum length.

4.1.3.8 CHANNEL TYPE KEY

Channel TEDS data field number 8

Data type: unsigned byte integer used for enumeration (U8E, 1 byte)

This field specifies the channel transducer type. The values for Channel Type Key are defined in Table 4.14.

Table 14: Enumeration of Channel Type Keys

Key Value (K)	Meaning
0	Sensor
1	Actuator
2	Event sequence sensor
3	Data sequence sensor
4	General transducer
5	Buffered sensor
6	Buffered data sequence sensor
7-255	Reserved for future expansion

4.1.3.9 PHYSICAL UNITS

Channel TEDS data field number 9

Data type: Physical units (UNITS, 10 bytes)

This field defines the physical units that apply to the transducer data, however, if the Calibration Key is CAL_FIXED, CAL_MODIFIABLE, CAL_SELF, or CAL_CUSTOM the physical units apply only to the transducer data *after* correction for the case of sensors, or *before* correction for the case of actuators.

4.1.3.10 LOWER RANGE LIMIT

Channel TEDS data field number 10

Data type: single-precision real (F32, 4 bytes)

For sensors, this shall be the lowest valid value for transducer data after correction is applied, interpreted in the units specified by the Physical Units field of the Channel TEDS. If the corrected transducer data lies below this limit, it may not comply with STIM specifications set by the manufacturer.

For actuators, this shall be the lowest valid value for transducer data before correction is applied, interpreted in the units specified by the physical units field of the channel TEDS. Writing corrected transducer data below this limit may result in behavior outside the STIM specifications set by the manufacturer.

In cases where no correction is applied and the data is expressed in a different format than single-precision real, conversion to single-precision real is necessary before making the comparison.

An example of this is data from a channel whose Calibration Key is CAL_NONE and whose Data Model is N-byte integer. Note that this conversion may limit the practical range or precision of the converted transducer data.

When this parameter is not applicable it shall be NaN. An example of a number for which Range Limits do not apply is N-byte data representing a bank of switches. In this case the field shall be set to NaN. On the other hand, Range Limits may apply to N-byte data that represents a 12 bit integer with no expressed units, such as raw analog-to-digital convertor (ADC) output. In either case, the physical units will be

"digital data." If the Channel Data Repetitions field of this channel is nonzero, then the value of this field shall be interpreted to apply to all of the repetition instances.

4.1.3.11 UPPER RANGE LIMIT

Channel TEDS data field number 11

Data type: single-precision real (F32, 4 bytes)

For sensors, this shall be the highest valid value for transducer data after correction is applied, interpreted in the units specified by the Physical Units field of the Channel TEDS. If the corrected transducer data lies above this limit, it may not comply with STIM specifications set by the manufacturer.

For actuators, this shall be the highest valid value for transducer data before correction is applied, interpreted in the units specified by the Physical Units field of the Channel TEDS. Writing corrected transducer data above this limit may result in behavior outside the STIM specifications set by the manufacturer.

In cases where no correction is applied, and the data is expressed in a different format than single-precision real, conversion to single-precision real is necessary before making the comparison. An example of this is data from a channel whose Calibration key is CAL_NONE and whose Data Model is N-byte integer. Note that this conversion may limit the practical range or precision of the converted transducer data. When this parameter is not applicable it shall be NaN.

An example of a number for which Range Limits do not apply is N-byte data representing a bank of switches. In this case the fid shall be set to NaN. On the other

hand, Range Limits may apply to N-byte data that represents a 12 bit integer with no expressed units, such as raw ADC output. In either case, the physical units will be "digital data." If the Channel Data Repetitions field of this channel is nonzero, then the value of this fielshall is interpreted to apply to all of the repetition instances.

4.1.3.12 WORST-CASE UNCERTAINTY

Channel TEDS data field number 12

Data type: single-precision real (F32, 4 bytes)

The value of this field shall be expressed in the same units as the transducer data as specified in the Physical Units field of the Channel TEDS, 4.1.3.9.

4.1.3.13 SELF-TEST KEY

Channel TEDS data field number 13

Data type: unsigned byte integer used for enumeration (U8E, 1 byte)

This field defines the self-test capabilities of the transducer as shown in Table 4.15.

Table 4.15: Enumeration of Self-Test Keys

Key Value (K)	Meaning
0	No self-test function needed or provided
1	Self-test function provided
2-255	Reserved for future expansion

4.1.3.14 CHANNEL DATA MODEL

Channel TEDS data field number 14

Data type: unsigned byte integer used for enumeration (U8E, 1 byte)

This field describes the data model used when addressing read transducer data or write transducer data for this channel as shown in Table 4.16.

There are two differences between N-byte-integer (enumeration zero) and N-byte-fraction (enumeration three), as follows:

a) The radix point (which divides integer from fractional bits) is to the right of the lsb for N-byte-integer.

It is immediately to the right of the msb for N-byte-fraction.

b) Justification of the significant bits differs (see 4.1.3.16).

The N-byte fraction type may be used to keep the multinomial coefficients within represent able bounds.

Table 4.16: Enumeration of Channel Data Models

Value	Model	Length
0	N-byte integer (unsigned)	0□ N□ 255
1	Single-precision real	4 bytes
2	Double-precision real	8 bytes
3	N-byte fraction (unsigned)	0□ N□ 255
4-255	Reserved for future expansion	-

4.1.3.15 CHANNEL DATA MODEL LENGTH

Channel TEDS data field number 15

Data type: unsigned byte integer used for counting (U8C, 1 byte)

This field specifies the number of bytes in the representation of the selected Channel Data Model.

For N-byte integer the value in this field shall be N, where 0 £ N £ 255.

For N-byte fraction the value in this field shall be N, where 0 £ N £ 255.

For single-precision real the value in this field shall be 4.

For double precision real the value in this field shall be 8.

4.1.3.16 CHANNEL MODEL SIGNIFICANT BITS

Channel TEDS data field number 16

Data type: unsigned 16 bit integer used for counting (U16C, 2 bytes)

When the Channel Data Model is N-byte integer (enumeration zero) or N-byte fraction (enumeration three), the value of this field is the number of bits that are significant. The value of this field shall be between zero and 2040.

For example, if data from a transducer channel comes from a 12-bit ADC, then

Channel Data Model = N-byte integer (field enumeration value of zero)

Channel Data Model Length = 2 (the number of bytes to hold 12 bits)

Channel Model Significant Bits = 12

When the Channel Data Model is N-byte integer or N-byte fraction, the Channel Model Significant Bits shall not exceed eight times the Channel Model Data Length.

When the Channel Data Model is N-byte integer, the significant data bits shall be right-justified within the byte stream.

When the Channel Data Model is N-byte fraction, the significant data bits shall be left-justified within the byte stream.

When the Channel Data Model is single- or double-precision real (enumerations one or two), the value of this field is the number of bits in the STIM's signal converter.

4.1.3.17 CHANNEL DATA REPETITIONS

Channel TEDS data field number 17

Data type: unsigned byte integer used for counting (U16C, 2 bytes)

The number *L* of repetitions of the transducer value produced or required by a single trigger. Each repetition represents an additional measurement or actuation value produced or consumed by the transducer at each trigger, which shall be spaced apart from the initial value associated with the trigger along some axis (for example, time) by an amount defined by the Channel TEDS fields Series Increment and Series Units, respectively.

When *L* is zero, the values of Series Origin, Series Increment, and Series Units may be ignored. The purpose of this structure shall be to enable the specification of transducers that produce an array of data with the application of a single trigger such as a time series or a mass spectrum. When reading or writing data with Channel Data Repetitions greater than zero, the order of transmittal shall be with the 0th data sample transmitted first, the first repetition transmitted second, etc.

4.1.3.18 SERIES ORIGIN

Channel TEDS data field number 18

Data type: single-precision real (F32, 4 bytes)

For the case where the Channel Data Repetitions is greater than zero, the Series Origin represents the value of the independent variable associated with the first datum returned in a data set. The Series Origin is expressed in units defined by the Series Units field in the Channel TEDS, 4.1.3.20.

4.1.3.19 SERIES INCREMENT

Channel TEDS data field number 19

Data type: single-precision real (F32, 4 bytes)

For the case where the Channel Data Repetitions is greater than zero, the series increment represents the spacing between values of the independent variable associated with successive members of the data set. The series increment is expressed in units defined by the Series Units field in the Channel TEDS, 4.1.3.20.

4.1.3.20 SERIES UNITS

Channel TEDS data field number 20

Data type: Physical units (UNITS, 10 bytes)

This field specifies the physical units associated with the series origin, 4.1.3.18, and series increment, 4.1.3.19 fields in the Channel TEDS.

4.1.3.21 CHANNEL UPDATE TIME

Channel TEDS data field number 21

Data type: single-precision real (F32, 4 bytes)

This field specifies the maximum time (tu), in seconds, between the receipt of a trigger and the issue of trigger acknowledge for this channel. This parameter allows NCAPs to determine time-out values, if appropriate.

For data sequence and buffered data sequence sensors, this parameter only applies when they are enabled. For event sequence sensors, this parameter shall be NaN.

4.1.3.22 CHANNEL WRITE SETUP TIME

Channel TEDS data field number 22

Data type: single-precision real (F32, 4 bytes)

This field specifies the minimum time (tws), in seconds, between the end of a write frame and the application of a trigger. (For most devices this will be a setup time characteristic of the transducer electronics. For more complex transducers, particularly those with a microprocessor, there could be additional time needed that can be specified by this constant.)

4.1.3.23 CHANNEL READ SETUP TIME

Channel TEDS data field number 23

Data type: single-precision real (F32, 4 bytes)

This field specifies the minimum time (trs), in seconds, between the trigger acknowledge and the beginning of a read frame. (For most devices this will be a setup time characteristic of the transducer electronics. For more complex transducers, particularly those with a microprocessor, there could be additional time needed that can be specified by this constant.)

4.1.3.24 CHANNEL SAMPLING PERIOD

Channel TEDS data field number 24

Data type: single-precision real (F32, 4 bytes)

The Channel Sampling Period (tsp) shall be the minimum sampling period of the channel transducer unencumbered by read or write considerations (since there is no requirement to read or write with each trigger). Typically, for sensor, buffered sensor, and actuator channels this time will be limited by A/D or D/A conversion times, STIM processor speed, etc., but in more complex transducers it may reflect transducer or sample handling times as well (e.g., a pH sensor that on each trigger extracts a new fluid sample using a pump). If reads or writes are involved, then the actual sampling rates will be further limited by setup and data transfer times depending on the transducer type.

In the case of data sequence and buffered data sequence transducers, this parameter shall represent the sequence sampling time determined by the STIM implementation.

In the case of event sequence transducers, this parameter shall represent the

minimum event resolution time. The Channel Sampling Period shall be expressed in seconds.

4.1.3.25 CHANNEL WARM-UP TIME

Channel TEDS data field number 25

Data type: single-precision real (F32, 4 bytes)

This field specifies the period of time, in seconds, in which the device stabilizes its performance to predefined tolerances after the application of power to the transducer.

4.1.3.26 CHANNEL AGGREGATED HOLD-OFF TIME

Channel TEDS data field number 26

Data type: single-precision real (F32, 4 bytes)

This field specifies the maximum aggregated time (tch) that the STIM will spend holding off the data transport during a complete data transfer addressed to *read transducer data* or *write transducer data* and this channel, assuming the Maximum Data Rate is used. This time shall include the time between the NCAP activating the data transport and the STIM acknowledgment.

4.1.3.27 TIMING CORRECTION

Channel TEDS data field number 27

Data type: single-precision real (F32, 4 bytes)

This field specifies the time offset, in seconds, between the issue of global trigger acknowledge and when this channel actually sampled the sensor or updated the actuator.

4.1.3.28 TRIGGER ACCURACY

Channel TEDS data field number 28

Data type: single-precision real (F32, 4 bytes)

This field specifies the accuracy, in seconds, of the Timing Correction.

4.1.3.29 EVENT SEQUENCE OPTIONS

Channel TEDS data field number 29

Data type: unsigned byte integer used for enumeration (U8E, 1 byte)

An event sequence sensor has the option of changeable pattern, upper threshold, and/or hysteresis. It also has the option of detecting inconsistencies in settings of these parameters as described in 4.6.9. This enumeration defines, for the NCAP, the ability of the STIM to detect and report these inconsistencies. The options are defined in Table 4.17.

Table 4.17: Event sequence options

Value	Meaning
0	Not applicable
1	Pattern/threshold/hysteresis not changeable
2	Changeable and inconsistencies detected
3	Changeable and inconsistencies not detected
4-255	Reserved

4.1.3.30 CHECKSUM FOR CHANNEL TEDS

Channel TEDS data field number 30

Data type: unsigned 16 bit integer used for counting (U16C, 2 bytes)

This field contains the checksum for the complete Channel TEDS data block. The checksum shall be the one's complement of the sum (modulo 216) of all the data structure's preceding bytes, including the initial length field and excluding the checksum field.

4.1.4 FLASH MEMORY

Flash memory is a type of constantly-powered nonvolatile memory that can be erased and reprogrammed in units of memory called blocks. It is a variation of electrically erasable programmable read-only memory (EEPROM) which, unlike flash memory, is erased and rewritten at the byte level, which is slower than flash memory updating. Flash memory is often used to hold control code such as the basic input/output system (BIOS) in a personal computer. When BIOS needs to be changed (rewritten), the flash memory can be written to in block (rather than byte) sizes, making it easy to update. On the other hand, flash memory is not useful as random access memory (RAM) because RAM needs to be addressable at the byte (not the block) level.

4.2 NON-FUNCTIONAL REQUIREMENT

A non-functional requirement does not describe what a system or software will do or process, but how it does. For example, software performance requirements, some external interface requirements, software design constraints, and software quality attributes. Non-functional requirements are difficult to test, they are usually, or most of the time, evaluated subjectively. The non-functional requirements are listed below:-

- Availability is about the rate of hardware and software component failure (mean time between failures)
- Maintainability Ability of the support programming staff such as the web master to keep the system in steady running state, including enhancement
- Data integrity Tolerance for loss, corruption, or duplication of data
- Extensibility is to accommodate increased functionality
- Flexibility is to handle requirement changes, such as add new channel.
- Functionality Number, variety, and breadth of user-oriented features
- Portability Ability to move application to different platforms or operating systems

Quality Reduced number of severe defects

- Robustness Ability to handle error and boundary conditions while running
- Scalability Ability to handle a wide variety of system configuration sizes
- Install ability Ease of system installation on all necessary platforms

4.3 DEVELOPMENT REQUIREMENTS

HARDWARE REQUIREMENTS:-

Table 4.18 Hardware Requirement for development side

	Hardware Requirement	Description
Processor	Pentium II 300 MHz or equivalent	The minimum requirements in order to run Java
Operating System	Microsoft Windows XP	This platform is chosen because most of the computer in FSKTM is currently installed with Microsoft Windows XP.
Memory	128 MB RAM	This amount required to run Java
Hard Disk space	500 MB Hard disk	This amount of hard disk spacerequired to store the database of the database, documentation and the entire system component.
External device	Printer	Printer needs to print all the documents that needs for the system development

4.4.1 HARDWARE REQUIREMENTS:

- For Windows
- Intel Pentium 200MHz processor or equivalent processor running
- Windows 98, Windows ME, Windows NT, Windows 2000 or Windows XP
- 32 MB RAM (128 MB is recommended)
- 20 MB of available disk space

- Internet Connection
- A 16-bit color monitor capable of 1024 x 768 resolutions
- For Macintosh
- Macintosh PowerPC with System 8.6 or later
- Mac OS 9.1, Mac OS 9.2.1, or Mac OS X 10.1 or later
- 32 MB RAM (128 is recommended)
- 20 MB of available disk space

4.5 CHAPTER SUMMARY

When we make the requirements measurable, we have the basis for testing and negotiating the solutions. Whenever we discover a new requirement, we test it to ensure that it conforms to the minimum standard. That is:

- It has an understandable, no ambiguous description.
- It has fit criteria that will guarantee that the implemented solution conforms to, or fits, the original requirement.
- It is correctly identified.
- It carries the appropriate cross-references.

5.0 DESIGN

5.1 DATA FLOW DIAGRAM OF TEDS

The data flow diagram is about how the TEDS data structure is write and read from flash memory. The TEDS data structures begin with setup the TEDS data structure first such as Setup META, Setup Channel 1 and Setup Channel 2 TEDS. It is followed by loading TEDS data structure into random access memory (RAM) known as buffer memory. The loading process is converting the data structures from text structure into binary to store in buffer memory. After that the TEDS data structures are write and read to flash memory using the same process. The data flow diagram is shown in figure 5.1 below.

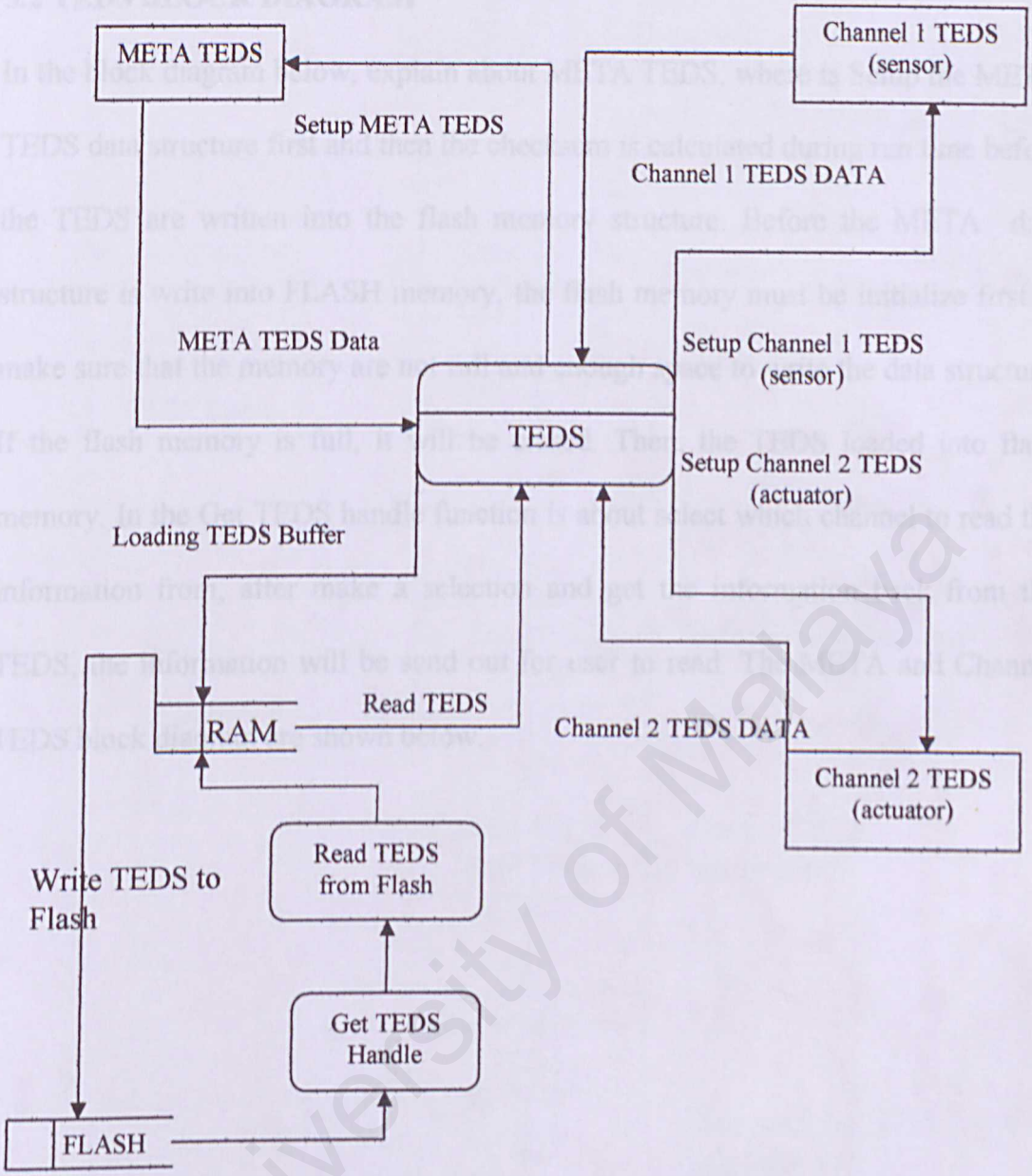


Figure 5.1: Data Flow Diagram for TEDS

5.2 TEDS BLOCK DIAGRAM

In the block diagram below, explain about META TEDS, where is Setup the META TEDS data structure first and then the checksum is calculated during run time before the TEDS are written into the flash memory structure. Before the META data structure is write into FLASH memory, the flash memory must be initialize first to make sure that the memory are not full and enough space to write the data structure. If the flash memory is full, it will be erased. Then, the TEDS loaded into flash memory. In the Get TEDS handle function is about select which channel to read the information from, after make a selection and get the information back from the TEDS, the information will be send out for user to read. The META and Channel TEDS block diagram are shown below.

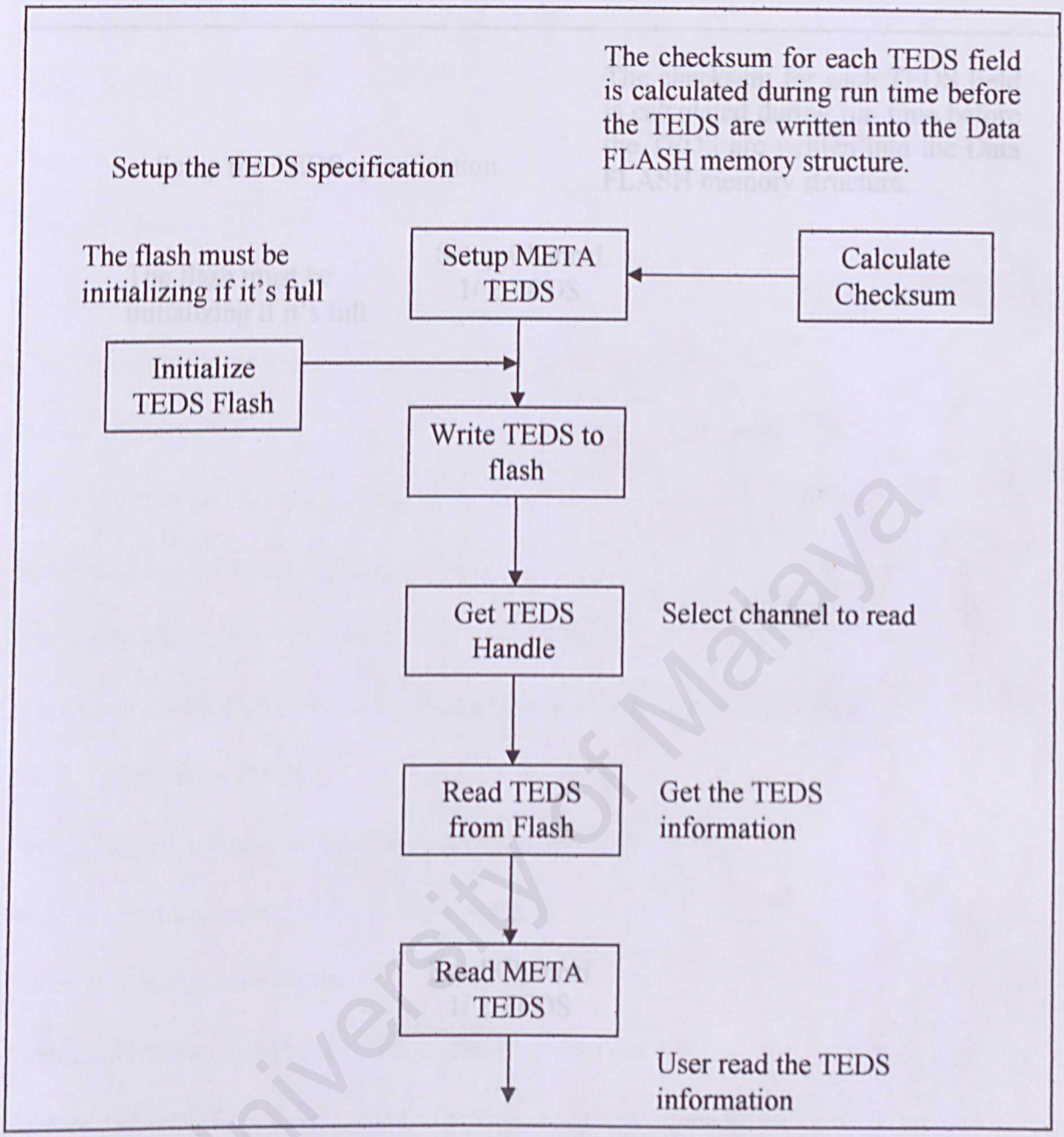


Figure 5.2:META TEDS Block Diagram

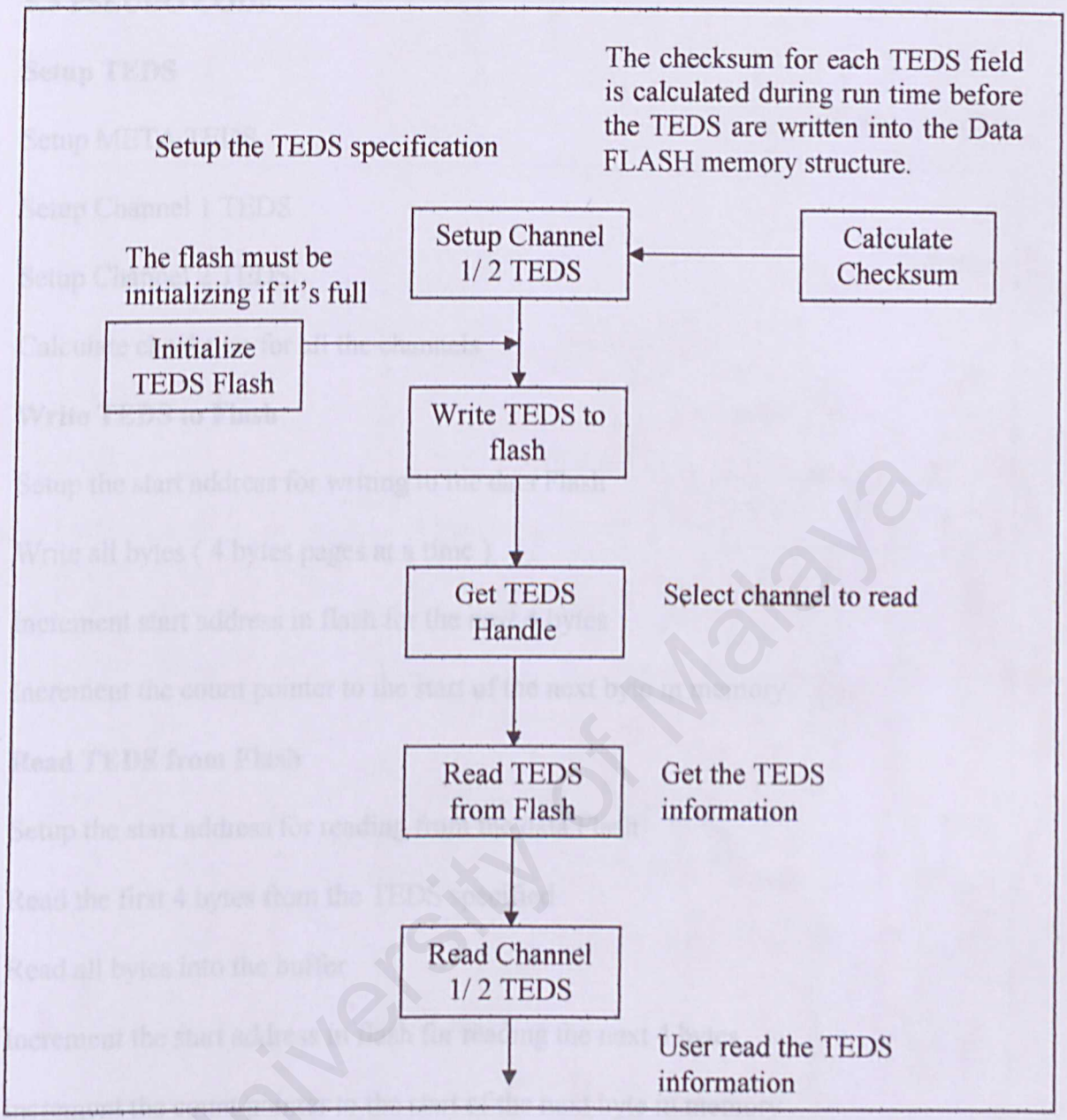


Figure 5.3: Channel TEDS Block Diagram

5.3 PSEUDO CODE

Setup TEDS

Setup META TEDS

Setup Channel 1 TEDS

Setup Channel 2 TEDS

Calculate checksum for all the channels

Write TEDS to Flash

Setup the start address for writing to the data Flash

Write all bytes (4 bytes pages at a time)

Increment start address in flash for the next 4 bytes

Increment the count pointer to the start of the next byte in memory

Read TEDS from Flash

Setup the start address for reading from the data Flash

Read the first 4 bytes from the TEDS specified

Read all bytes into the buffer

Increment the start address in flash for reading the next 4 bytes

Increment the count pointer to the start of the next byte in memory

Get TEDS Handle

Select Channel

Get Data from Channel

5.4 FLOW CHART OF TEDS INTERFACE

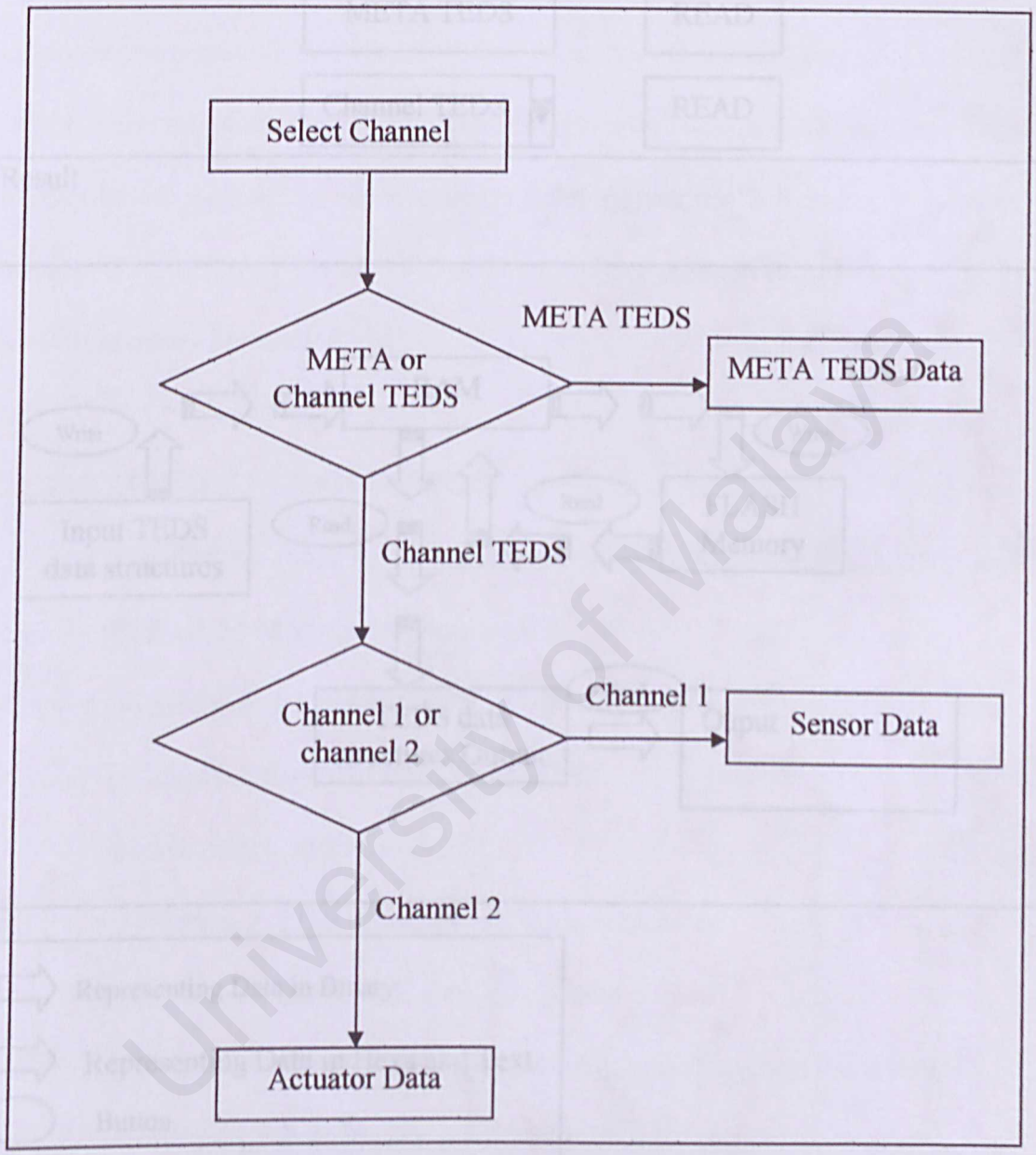
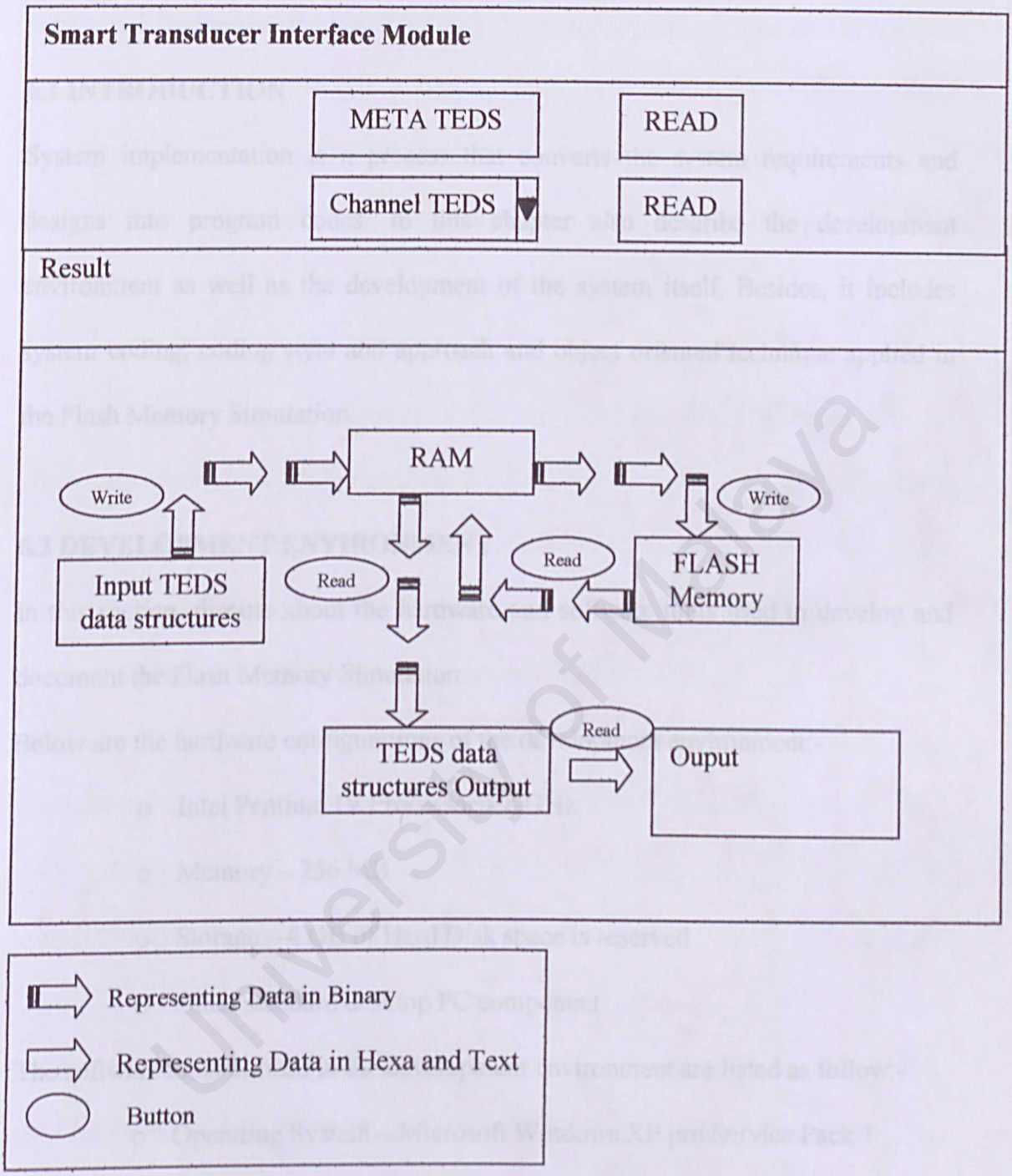


Figure 5.4: Flow Chart of TEDS Interface

5.5 INTERFACE PROTOTYPE



6.0 SYSTEM IMPLEMENTATION

6.1 INTRODUCTION

System implementation is a process that converts the system requirements and designs into program codes. In this chapter also describe the development environment as well as the development of the system itself. Besides, it includes system coding, coding style and approach and object oriented technique applied in the Flash Memory Simulation.

6.2 DEVELOPMENT ENVIRONMENT

In this section, discuss about the hardware and software tools used to develop and document the Flash Memory Simulation.

Below are the hardware configurations of the development environment:-

- Intel Pentium IV Processor 1.8 GHz
- Memory – 256 MB
- Storage – 4 GB of Hard Disk space is reserved
- Other standard desktop PC component

The software tools utilized in the development environment are listed as follow:-

- Operating System – Microsoft Windows XP pro Service Pack 1
- Web Browsers – Internet Explorer 6.0
- J2SE (J2SE1.4.2_07)
- Xinox software JCreator LE v2.50

6.3 DEVELOPMENT OF THE SYSTEM

First step taken before start develops the system is study and get the experiences with the java programming language. Below is the explanation of Object Oriented program approach, classes that are defined and created for the simulation system and the related coding parts of the entire simulation program.

6.3.1 OBJECT ORIENTED PROGRAM

In this section will discuss about an object, class and also the relationship between objects and class. Firstly, discuss about the HTML markup for Applets. In this code, use the <APPLET> tag to include applets within HTML files. The syntax of the <APPLET> tag is shown below;

```
<HTML> <TITLE>Flash Memory Simulation</TITLE>
<BODY> <H1><center>Flash Memory Simulation</center></H1>
<center><APPLET CODE = "Box.class" WIDTH = "500" HEIGHT ="450">
</APPLET></center>
</BODY></HTML>
```

The <APPLET> tag specifies where and how to display an applet within the HTML document. The CODE, WIDTH, and HEIGHT parameters are required. Parameters within the <APPLET> tag are separated by spaces, not by commas. </APPLET> used for closes the <APPLET> tag. The following parameters may appear inside the <APPLET> tag. ALIGN, alignment is optional which is specifies the applet's

alignment on the Web page. Valid values are: left, right, top, texttop, middle, absmiddle, baseline, bottom, absbottom, and center. CODE is about applet-filename. This parameter or the OBJECT parameter is required. Name of applet *.class* file. The *.class* extension is not required in the <APPLET> tag but is required in the class's actual filename. The filename has to be a quoted string only if it includes whitespace. WIDTH is applet-pixel-width, required. Initial width of applet in pixels. Many browsers do not allow applets to change their width.

The AWT classes are about draw strings, images, and shapes via the Graphics class in Java program. The Graphics class is an abstract class that provides the means to access different graphics devices. It is the class that lets users draw on the screen, display images, and so forth. Graphics is an abstract class because working with graphics requires detailed knowledge of the platform on which the program runs. Below are the lists for the graphics methods that used in flash memory system;

Drawing string - These methods allow draw text strings on the screen. The coordinates refer to the left end of the text's baseline.

Drawing shapes - The location of the object's upper left corner, plus its width and height.

The example of the code is shown below:-

```
/*1-3 rectadd vertica*/
        g.drawRect(205,200,40,30);
        g.drawLine(225,190,225,200);//line pendek atas ke rect beza 60
```

```
g.drawLine(225,230,225,240);//line bwh lepas rect
g.drawLine(190,215,205,215);//line kiri rect
g.drawLine(245,215,260,215);//line kanan rect
```

By using this method (AWT) also can create a simple animation that uses a thread to run the application. Below is the example of code that using threads method;

```
for (x1 = 35; x1 <= 68; x1+=4)
 {      try { Thread.sleep(10); }
        repaint();
        catch (InterruptedException e) { }
 }
```

From the code shown above, show that the code creates a new thread to control the animation. This thread calls sleep(10), followed by repaint(), to display a new image every 10 milliseconds.

6.3.2 SYSTEM CODING

After studies the java language have been done, the development of the simulation system which is able to run the simulation as a web-based application which can be executed using any java enabled web browsers. The coding phase was done using the Xinox Software's JCreator Ligh Edition (LE) v 2.50 integrated development environtment.

6.4 PROGRAM DEVELOPMENT AND CODING

In this section will discuss about the process of creating the programs needed to satisfy an information system's processing requirements.

1. Review the program documentation – the first and foremost step to be taken in this chapter is to review the program documentation that was prepared during the earlier phases.
2. Designing the program – for this phase, is about recognize how the program can accomplish the features and functions that are described in the program documentation and developing a logical solution to the programming problem.
3. Coding approaches – there are two approaches in coding which is known as top down and bottom up approach. For this simulation system was developed using both the top down and bottom up approach.
4. Coding style – coding style is about how to manage the source code. It is important because it will make the system easier to maintain and enhance. Besides that, there are some elements must be consider such as standard naming convention and standard graphical user interface for better understanding. This can be done by using comments which is providing a clear guide to programmers for future enhancement.

6.5 CHAPTER SUMMARY

System implementation is a process that translates a detail design representation of software into a programming language realization. Coding convention is about program labeling, naming, and comments. Besides that, each code must be easy to understand and also must be easy to modify or corrected. The code also should be

able to handle user error by responding appropriately, perhaps with a diagnostic error message and system failure should not result. In this chapter assures that the system being developed is operational and then allowing the users to take over its operation for use.

7.0 SYSTEM TESTING AND SYSTEM EVALUATION

7.1 INTRODUCTION

Software testing is an important activity to check either the Flash Memory simulation system is operational based on its criteria. The objective of testing is to detect the presence of errors in systems, which are the errors that have not been discovered yet. In this case, a good test case is one that has a high probability of finding an undiscovered error. A successful test is one that discovers error whereas an unsuccessful test is one that discovers no errors. The goal of testing is to design test that will uncover the greatest number of errors or classes or errors with the minimum amount of time and effort. Successful testing will result in quality system will less errors and which work according to specification and performance requirements. It will lead to dependable and reliable system.

7.2 TESTING STAGES

The testing process is carried out in stages to suit the system, as the system itself is composed of modules integrated together. System development usually involves several stages of testing, consisting of unit testing, integration testing and system testing.

7.2.1 UNIT TESTING

Unit testing is the first stage of testing where each program component is tested on its own, isolated from the other components in the system. Unit testing verifies the correctness of the smallest unit of the application. The testing is generally carried out by programmers, and not the users, as detailed understandings of internal system coding and design are required for the testing. It verifies that the component functions work properly with the types of input and output expected from studying the component's design. After each component has been tested, the interaction between these components can be integrated.

Five modules have been tested, in which the author is in charge of functionalities of each module are tested for errors. Testing of each module is discussed as following:-

1. **Interface and navigation**

   This testing to ensure that the each page of the system has a navigation menu for users to return navigates in and round the system.

   Users are clearly told which page they are in at the moment, such as a title on top of each page to inform them that they are in which section.

2. Simulate the system

Testing performed to ensure that the users are able to enter input address and data to simulate the system and ensure that the system simulate correctly.

7.2.2 INTEGRATION SYSTEM

After performing the unit tests, the modules are integrated or combined into a working system. The testing is necessary because problems might occur only when the modules are integrated together, although the modules have been individually tested to be functioning properly. This integration is planned and coordinated so that when a failure occurs, it can be solved immediately.

7.2.3 SYSTEM TESTING

System testing is the final stage, where the whole integrated system is tested as one single unit, and it is to ensure that the whole system works according to user's specification. Developers will join the users to perform this stage of testing where the system is checked against the users' requirement description. If there is a need for a change, system modification will then be carried out. If the users are satisfied with the system's performance, the system is ready to be deployed. There are several

steps in testing the proposed system, such as functional testing, performing testing and acceptance testing.

Functional testing

Initially, functions performed by the proposed system are tested. This begins with a set of components that were tested. This begins with a set of components that were tested individually. Functional checks that whether the integrated system performs its functions as specified in the requirement.

Performance Testing

System performance is measured using performance objectives set by potential users as highlighted in the non-functional requirements section a guidelines.

Acceptance testing

The purpose of acceptance testing is to demonstrate that a system is ready for operational use.

7.3 SYSTEM EVALUATION

System evolution is processes that encounter technical and non technical problems during the development stage. The detected problems will be list out and recognize and try to solve it. The most important part in this chapter is to determine whether

the expected output true or not. The strength and weakness of the system also will be discussed in this chapter. So the system can be evaluated by the criteria that have been list out. When all the testing mentioned above are complete, we are convinced that the system meets all the requirements specified during the initial stages of software development.

7.4 CHAPTER SUMMARY

In this chapter, all the testing carried out on this system which include unit, integration, system, acceptance, performance, usability and security testing were discussed and explained in detail. System testing is a process of executing a program with the intent of finding errors and runtime program errors. System testing phase aims to uncover as many errors as possible in the system. The objectives of a system can only be achieved after a thorough testing is carried out. After test the system then the system can be evaluate based on the criteria that have been mention in system testing.

8.0 DISCUSSION AND CONCLUSION

8.1 DISCUSSION

8.1.1 PROBLEM ENCOUNTERED AND SOLUTIONS

There are various problems encountered during the development process of the proposed system. The following section highlights the problems come across and the solutions taken.

Lack of experience in programming language

Lack of knowledge in the new programming language has leads for a lot of difficulties during the early stage of system coding. The programming language used to develop the simulation system is Java programming language. Before this, I am not familiar with this programming language so the first step taken to get the knowledge and experience with this language is study the language. Besides that I also refer to many sources such as from internet, e-books, and java book to expert in this language. There are some problems encountered in this stage of system development, which is discussed below:-

Difficulty in coding

Problems were encountered when the languages had to be learning from scratch. When errors occurred, it was hard to detect and spend much time to debug the errors. Besides that, the main problem that I faced is to do the animation of the system. The solution of these problems was seeking advices from course mates and friends.

Besides, many tutorials and references was being referred and downloaded from internet. Programming books were in order to learn the basic concepts.

Difficulties in determining the system scope

The system scope is one of the main problem that I faced and also hard to determine. It is because I am not very clear what actually the simulation system that will be design. After get the guideline from the supervisor and moderator and they agree with the scope that I must do, the development of the system begin.

Lack of time

The development of the simulation system is done in the limited time. It is because I took other subjects that mostly are the third year course. In addition, each subject that I took has their own assignment either by group or individual. The subjects are quite difficult to understand and need more attention same as thesis project. Even though I have many assignment or subject to learn but I still spend much time to develop the simulation system. Easy to said that I try to develop the system everyday. To settle down these problems, I have managed my time properly so that I am able to finish my work on time.

When the system is done, I am satisfied with my system because the animation that I should do succeed. Even though the systems are not good enough and may be it looks easy for other people, I am happy with the system. It is because I start

developing the system using java language from the basic knowledge and I am able to make it.

8.2 SYSTEM STRENGTHS

a) Web Enabled

The simulation system was built based on the web technology. It means that the current implementation is deployable over the internet.

b) Informative Messages

The users will be prompted with appropriate message when trying to enter button without enter address or data.

c) Animation

Using this system for better understand about the flash memory rather than read it from books because this system explains the concept with animation.

d) Help provided

In this simulation system provides a help function that will explain to the users how to fully utilize the system.

8.3 SYSTEM CONSTRAINTS

The proposed system has a number of constraints, which are listed below.

a) Hard code

In this system, require user to enter address and data to be store in the flash memory, but only certain address and data are allowed. The users are not allowed

to enter random address and data to view the animation of this system. The value of the address and data can be view at main window. It is because, I have a problem with "code too large" while code the address and data.

b) Integer value

Only integer value both for address and data are allowed in this system. If the user enters floating point or character value, the system does not perform for the animation flash memory.

8.4 FUTURE ENHANCEMENT

Some functionalities of the system can be enhanced in order to improve the quality of the system, as well as reduce its constraints mentioned previously. The following section discussed about some enhancements that are possible to be incorporated into the system.

a) Various types of input

This system allow integer input for address and data, for the future enhancement the various of input such as floating point, hexadecimal, text and binary are allowed in this system.

b) Dynamic code

The input of this system were program in hard code, it means that only data that was program can be used. In the future, the data should be program in dynamic code means that the system will auto generate the input of the data.

8.5 KNOWLEDGE AND EXPERIENCE GAINED

Throughout the whole period of this system development life cycle, a lot of exciting and valuable experience was gained. There has been an improvement in searching information and solving problems, being able to work cooperatively in a team, as well gaining the capability to work independently. The benefit that gained throughout this project is the chance to understand the concept of software development process. There was a golden opportunity to learn additional programming language, which are not familiar before. Besides that, skills in time management were also improved when the system was compulsory to finish before the deadline. The way to handle a project under time constraints was learn. Indirectly, the experience to plan a system and solve a problem had been enriched.

8.6 CONCLUSION

System Overview

Finally the flash memory simulation system has achieved most of the aims and objectives stated in the introduction of the report. Even though the system has a number of constraints and limitation, but with the implementation further enhancement, it can be a powerful version of the same system, which is more efficient and effective. The system can be divided into three modules, first module is about the structure of flash memory, and second module is about flash memory cell and the last module is about view the flash memory.

Achieved Objective:-

After analyzing the project, objectives that have been met are listed below:-

1) Animate how the flash memory works.
2) The system can be view over internet.

9.0 REFERRENCE

9.1 Article and Journal Source:

[1] Ferrari P, Flammini A, Marioli D, Sisinni E, Taroni A, (Novembar, 2002), A Low-cost Smart Sensor with Java Interface.

[2] Ferrari P, Flammini A, Marioli D, Taroni A, (2002), A Low-cost Internet-enabled Smart Sensor.

[3] Institute of Electrical and Electronis Engineers, IEEE Standard for a Smart Transducer Interface for Sensors and Actuators - Transducer to Microprocessor Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats, 1997.

6.2 Internet Source:

[1] http://www.sensorsmag.com/articles/0800/46/main.shtml), 02/08/2004

[2] http://www.techonline.com/community/ed_resource/course/13399), 04/08/2004.

[3] www.telemonitor.com/doc/stim.pdf), 09/08/2004

[4]http://www-2.cs.cmu.edu/~sensing-sensors/S2004/L2004-03-data_acquisition/L2004-03-IEEE_1451-Anna_Liao.pdf), 12/08/2004

[5] http://www.analog.com/UploadedFiles/Application_Notes/4142732043878uC003__The_ADuC812_as_an_IEEE_1451.2_STIM.pdf , 23/08/2004

[6] http://bsing.ing.unibs.it/~label/activity/Web_Demo/How_to_use_WebSTIM.htm, 27/07/2004.

[7] http://jddac.labs.agilent.com/server/docs/transducer.htm, 18/18/2004

[8] http://ieee1451.nist.gov/intro.htm, 05/09/2004

[9] http://www.google.com.my/search?q=cache:WTFdtktsJ1UJ:sensorsynergy.com/Application_IEEE_1451_2.pdf+1451.2+STIM&hl=en, 07/09/2004

[10] http://www.manufacturing.net/ctl/article/CA191594, 08/09/2004

[11] http://www.big-dipper.com.cn/Network/STIM_Mobus/Ch2Data&Function.htm, 09/09/2004

[12] www.its.hk-r.se/staff/BLO/BT/BTreport_MEE-99-10.pdf, 12/09/2004

[13] http://jas2.eng.buffalo.edu/applets/education/system/memory/index.html

[14] http://java.sun.com/docs/books/tutorial/

[15] http://www.ibiblio.org/javafaq/javatutorial.html

[16] http://www.javacoffeebreak.com/tutorials/

[17] http://www.javaworld.com/javaworld/jw-03-1996/jw-03-animation.html

[18] http://www.rgagnon.com/javadetails/java-0262.html

[19] http://www.oberle.org/procsimu-index.html

[20] http://media.pearsoncmg.com/aw/aw_carpinel_compsys_1/rscpu/web.html

[21] http://www.freehandsource.com/_frames/_tips/_archive/tip_week020.html

[22] http://www.academicsolutions.com/workshop/animations/flash/animationflash.htm

[23] http://javaboutique.internet.com/javasource/

[24] http://java.sun.com/applets/jdk/1.1/demo/Animator/

[25] http://www.realapplets.com/tutorial/

[26] http://www.dgp.toronto.edu/~mjmcguff/learn/java/

[27] http://www.her.itesm.mx/academia/profesional/cursos/fisica_2000/FisicaII/PHYSENGL/physengl.htm

[28] http://www-fp.mcs.anl.gov/otc/Guide/CaseStudies/simplex/applet/source.html

6.3 Books reference:

[1] Daniel D. Gajski, University of California, , Principles of Digital Design, Prentice Hall, 1997.

[2] Roger L. Tokheim, Digital Electronics Principles and Applications, McGRAW-HILL International Editions, Fifth Edition, 1999.

[3] Deitel & Deitel, JAVA How To Program, Prentice Hall, Fifth Edition, 2003.

APPENDIX

USER MANUAL

This part is guidance for the first time user of this system. Below are the instructions to install J2SE 1.4.2_07 SDK.

Step1. Download and install the J2SE 1.4.2_07 SDK at http://java.sun.com/ web site.

Step2. Choose and click at DOWNLOAD WINDOWS J2SE SDK link and above figure appear and then click open button.

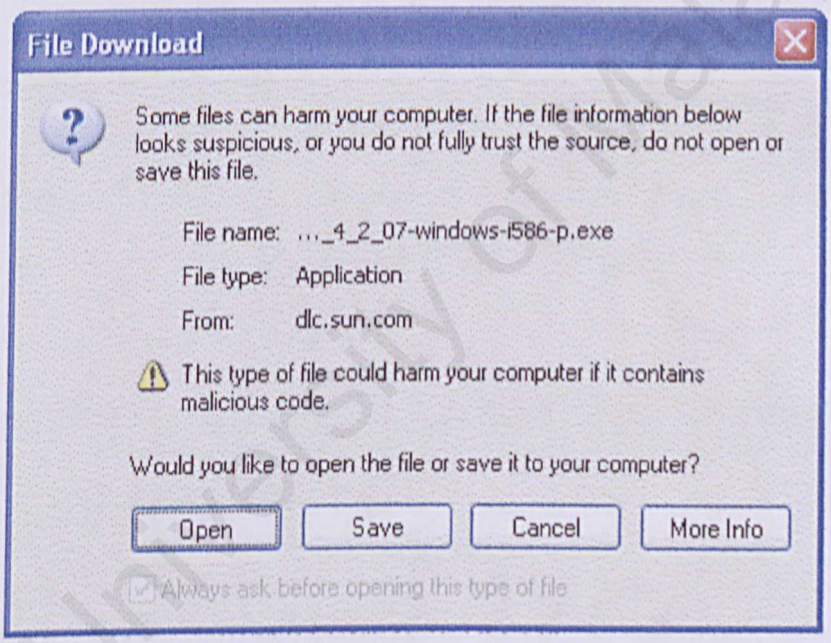


Figure 9.1: File Download box

Step 3: After that the figure below appear.

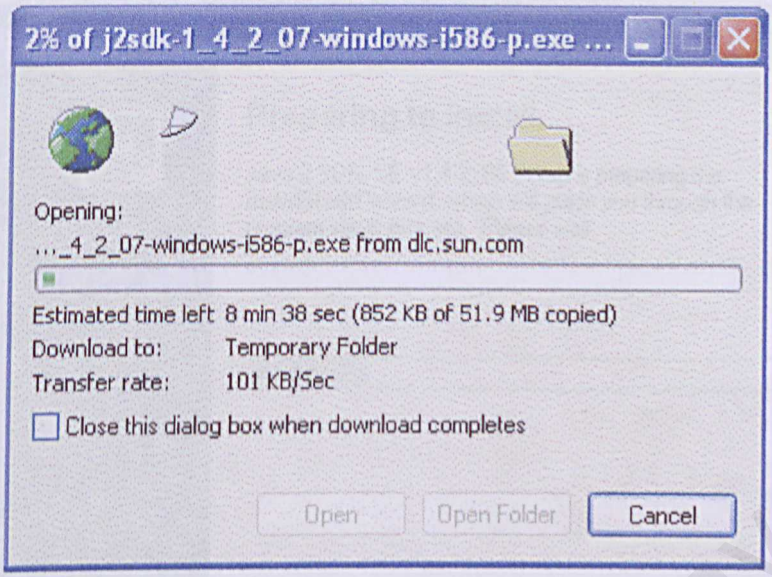


Figure 9.2: j2sdk-1_4_2_07 box

Step 4: Wait for the preparing to install

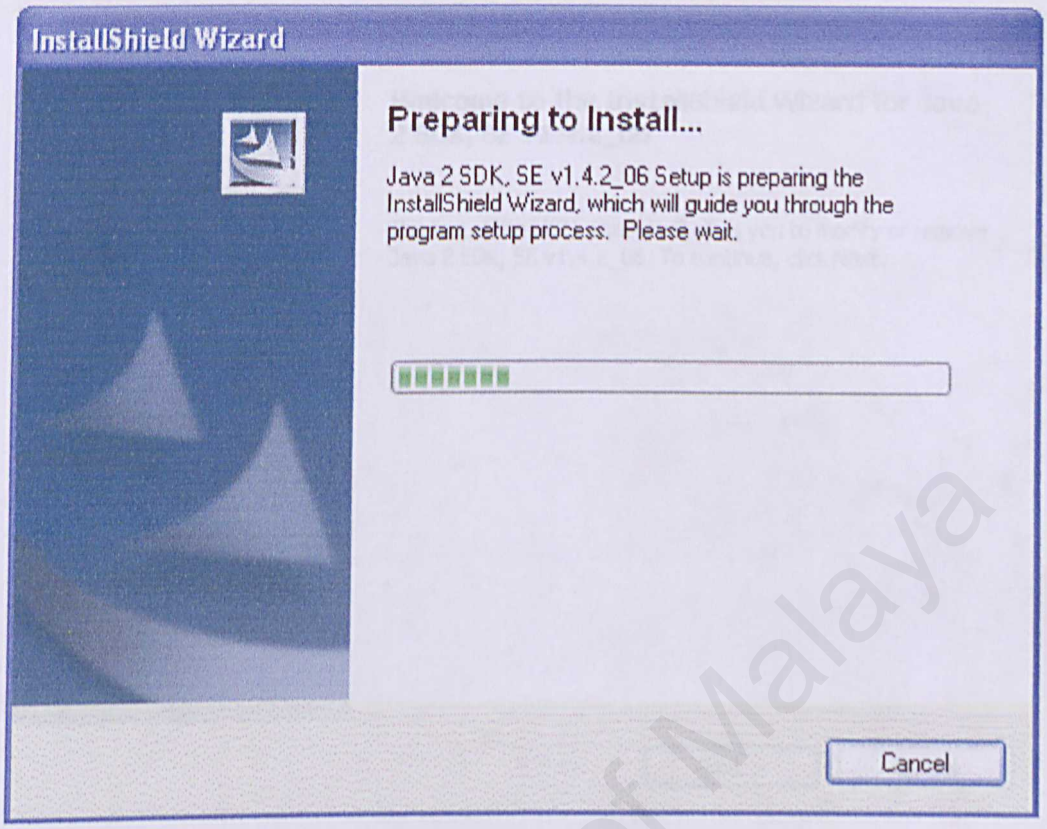


Figure 9.3 : InstallShield Wizard box

Step 5: Wait for the Windows Installer

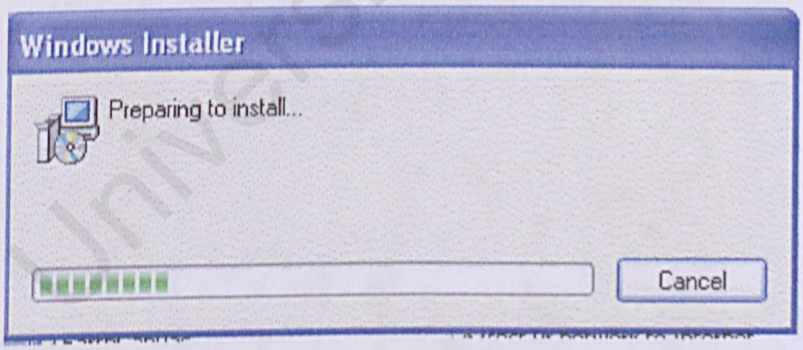


Figure 9.4: Windows Installer

Step 6: Click Next for this box

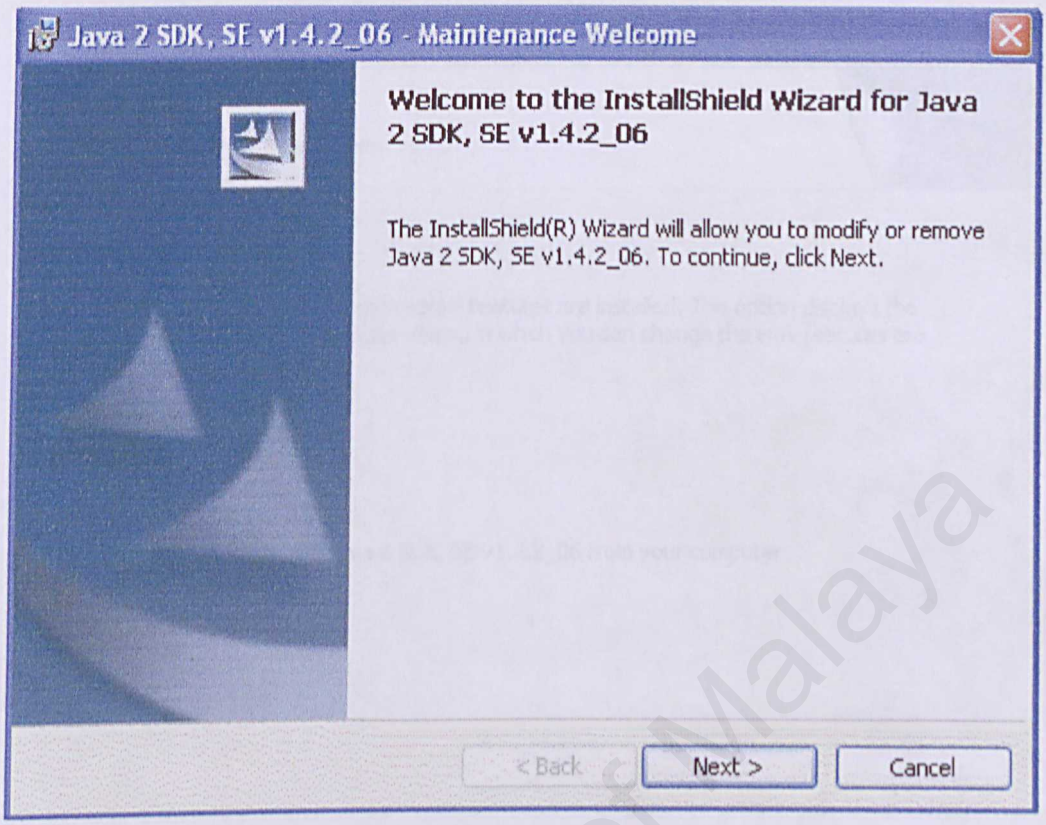


Figure 9.5 : Java 2 SDK, SE v1.4.2_06 box

Step 7: Select modify and click Next for this box

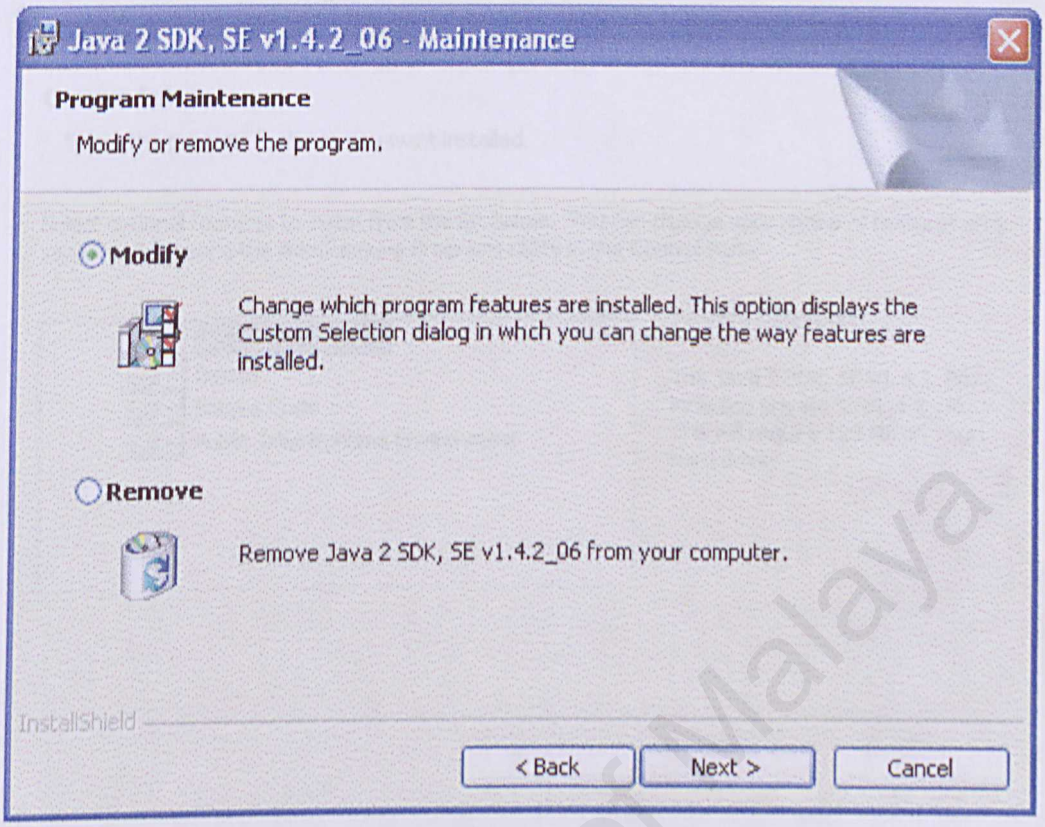


Figure 9.6 : Java 2 SDK, SE v1.4.2_06 - Maintenance

Step 8: Click Next for this box

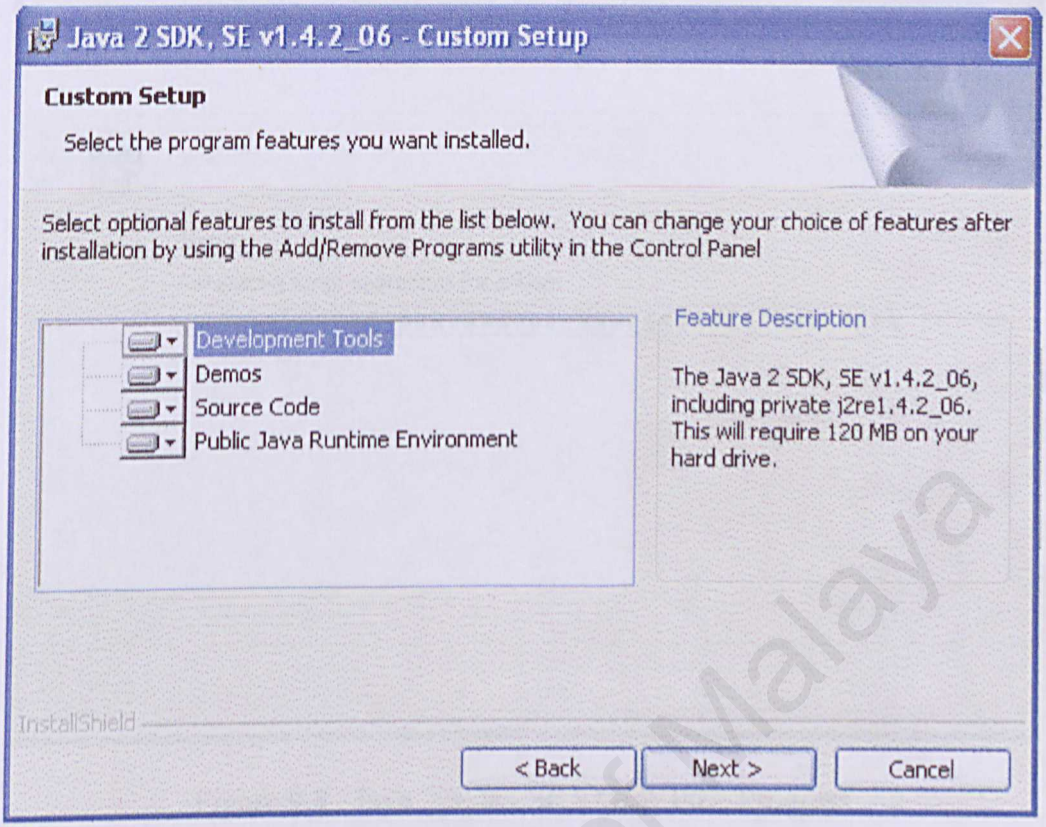


Figure 9.7 : Java 2 SDK, SE v1.4.2_06 – Custom Setup

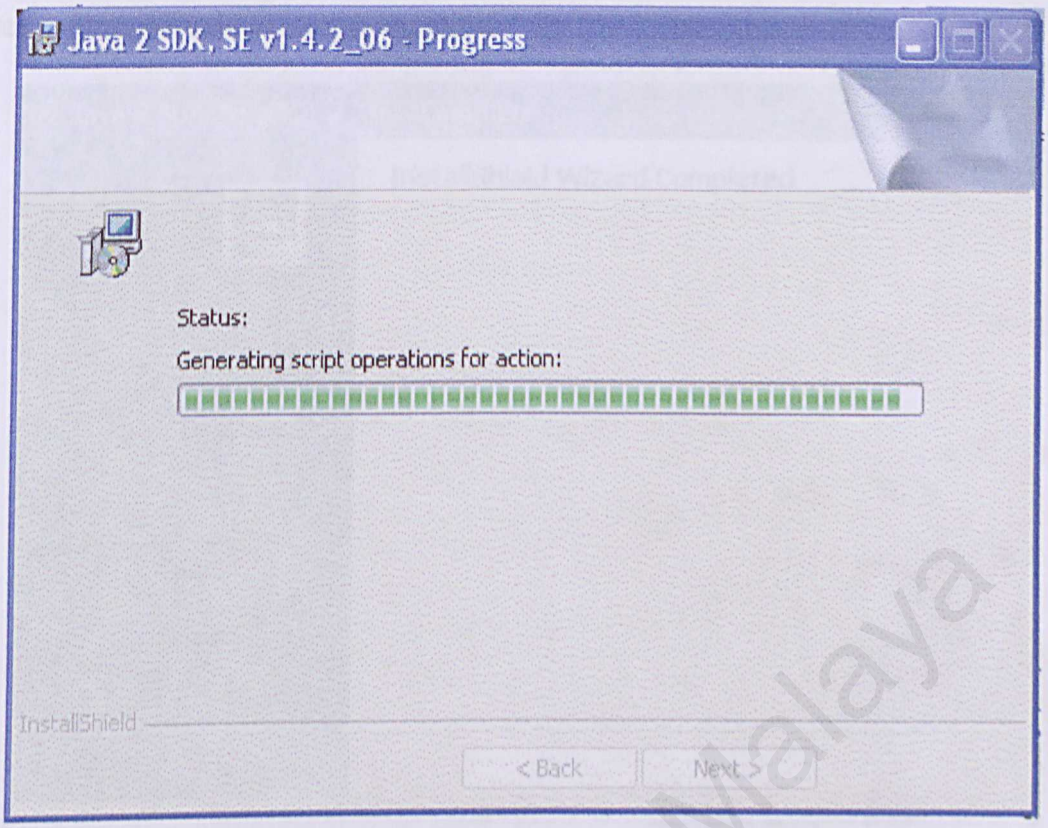


Figure 9.8 : Java 2 SDK, SE v1.4.2_06 – Progress

Step 9: InstallShield Wizard complete

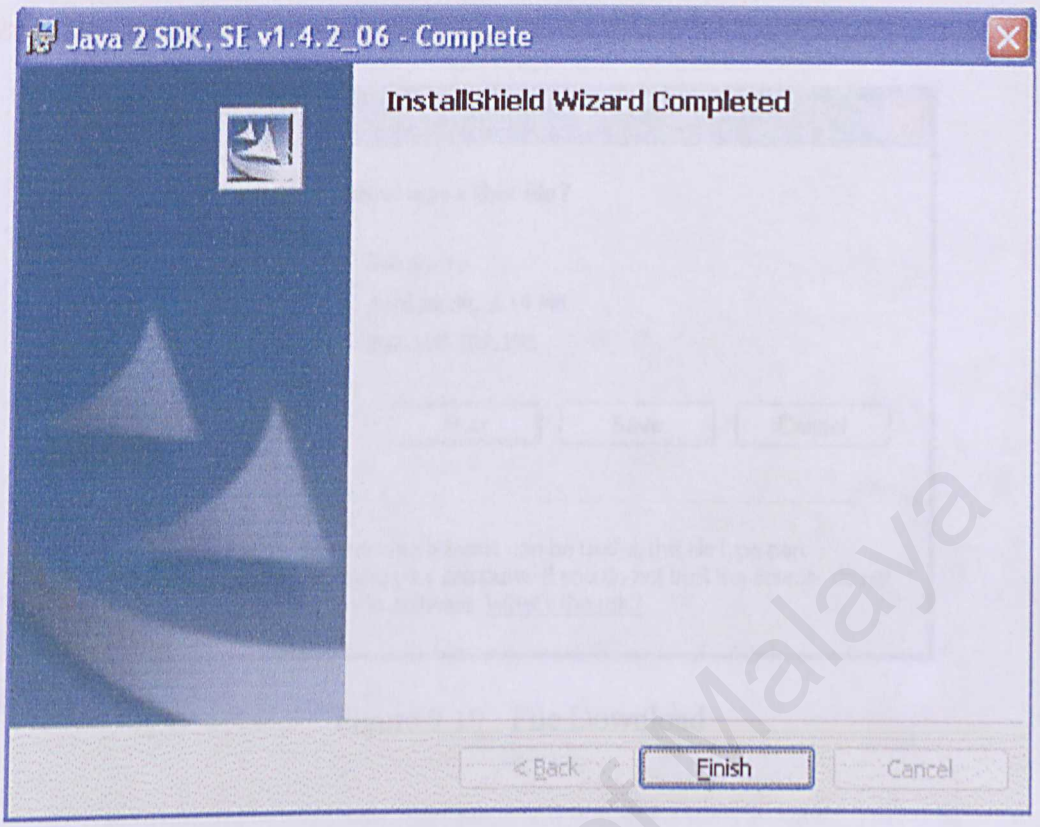


Figure 9.9 : Java 2 SDK, SE v1.4.2_06 – Progress

Below are the instructions to install Xinox software JCreator LE v2.50

Step1. Download and install the JCreator LE v2.50.

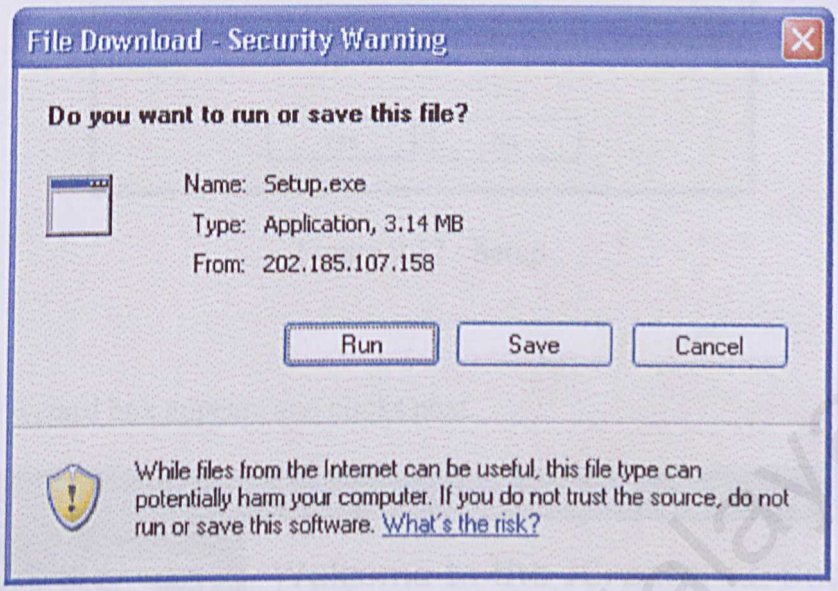


Figure 9.10 : File Download

Step2. Run the file download and the file will download to computer

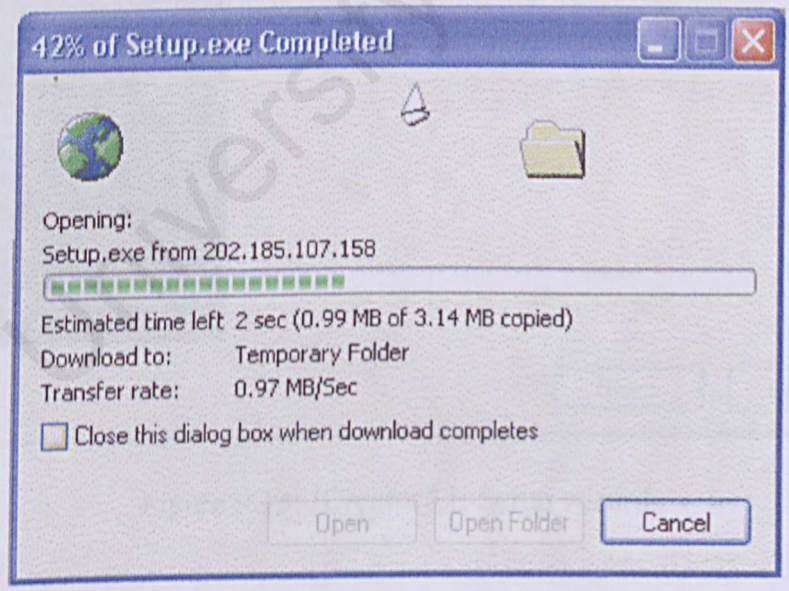


Figure 9.11 : Setup.exe Completed

Step3. Click yes to install JCreator LE

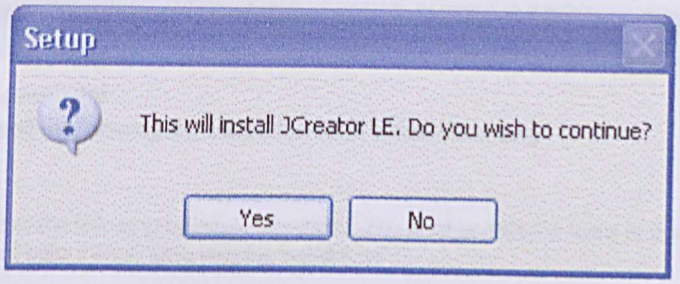


Figure 9.12 : Setup

Step4. The wizard box appears and clicks next

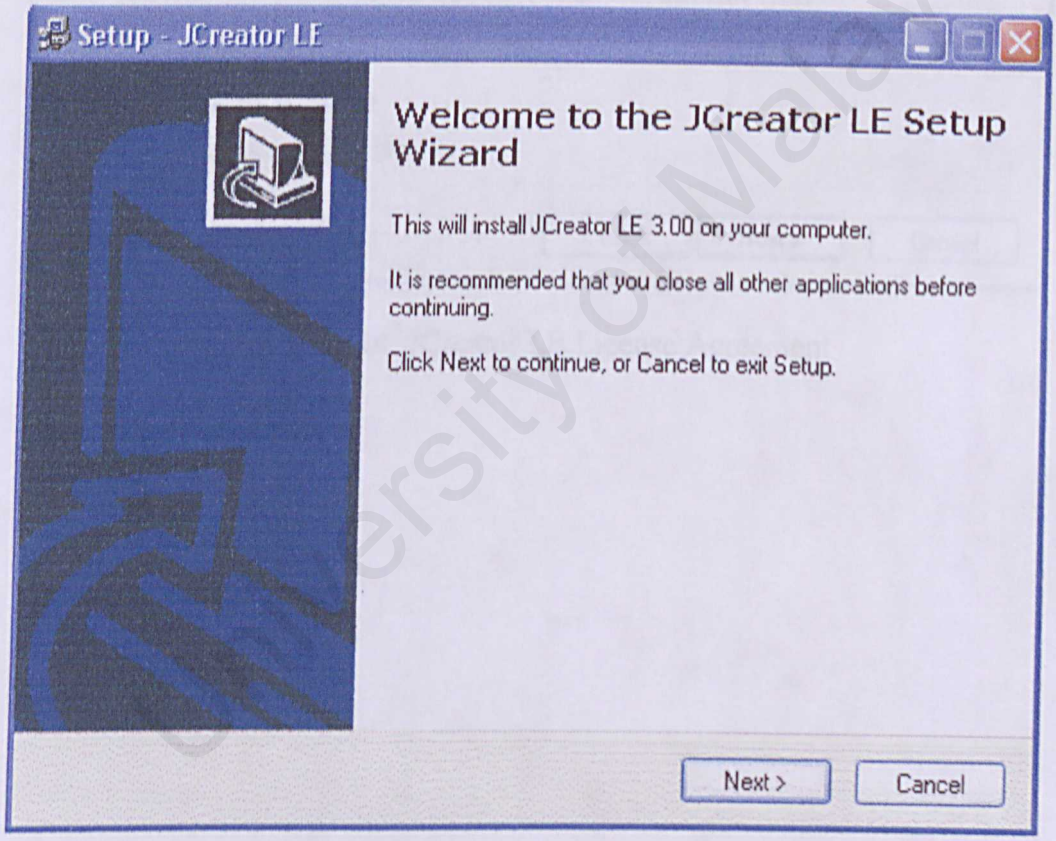


Figure 9.13: JCreator LE Setup wizard

Step5. Accept the agreement and click next

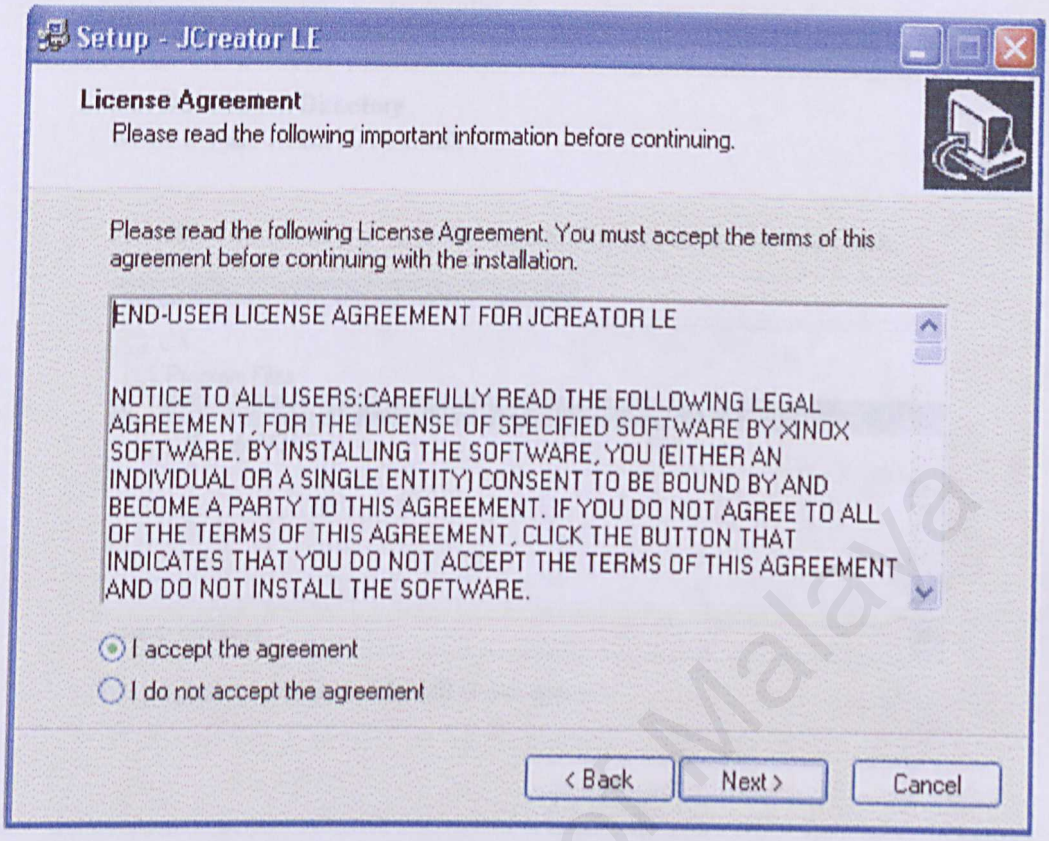


Figure 9.14: JCreator LE License Agreement

Step6. Select the destination directory to store the software

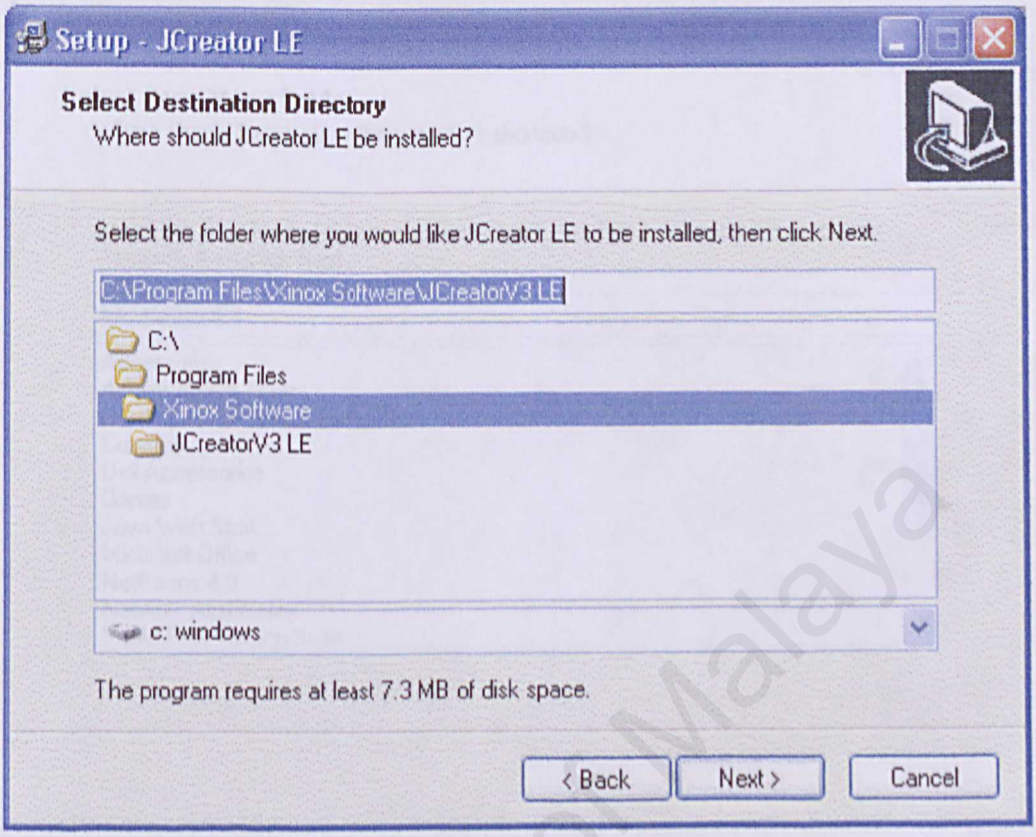


Figure 9.15: JCreator LE Directory

Step7. Click next for this box

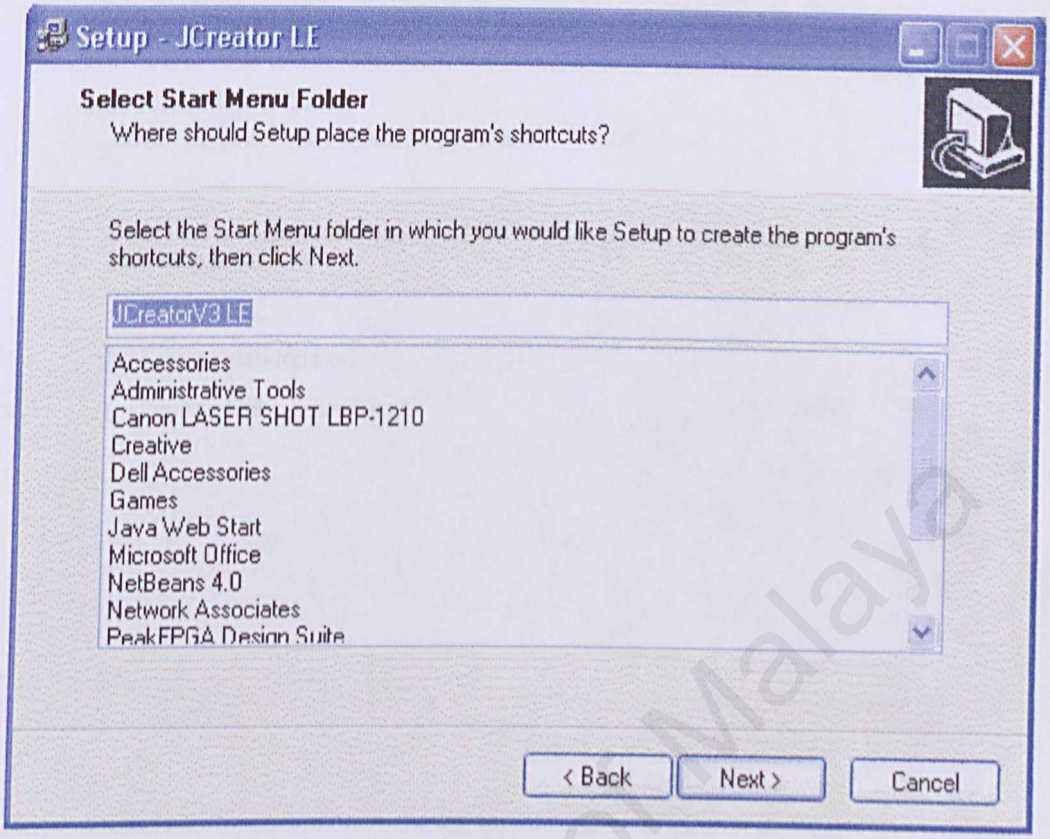


Figure 9.16: JCreator LE Select folder

Step8. Select Additional tasks and click next button

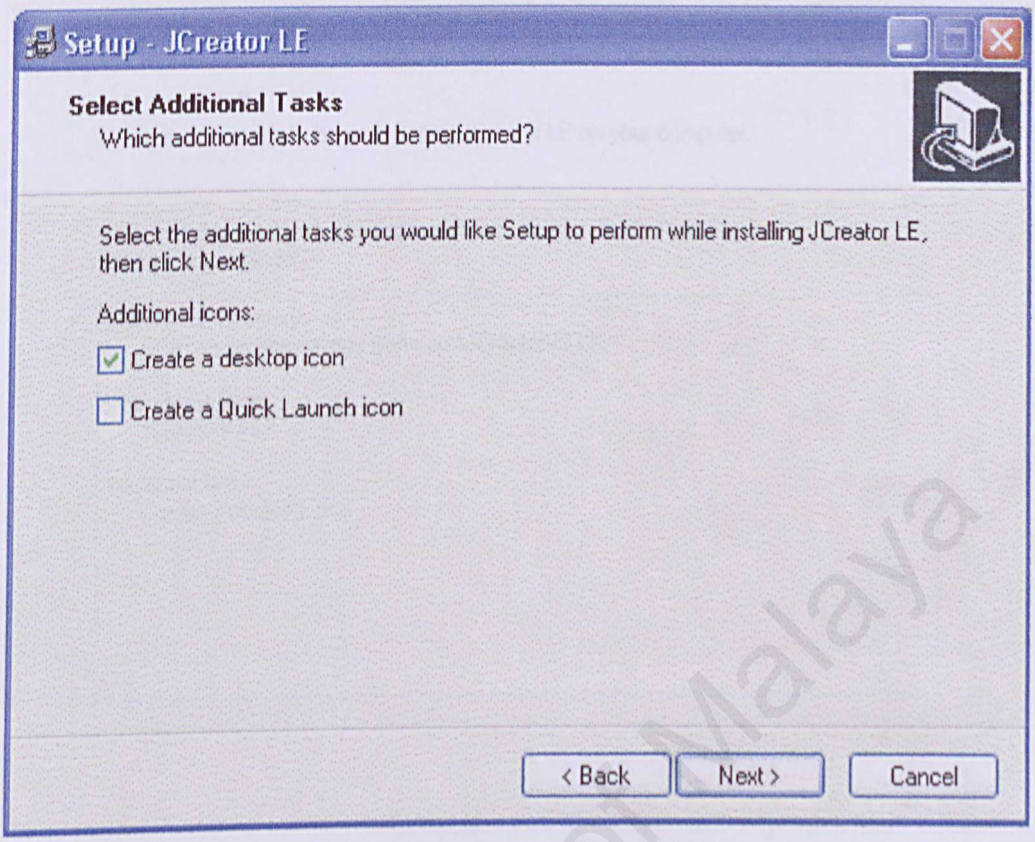


Figure 9.17 : JCreator LE Additional Tasks

Step9. JCreator is ready to install and click install button

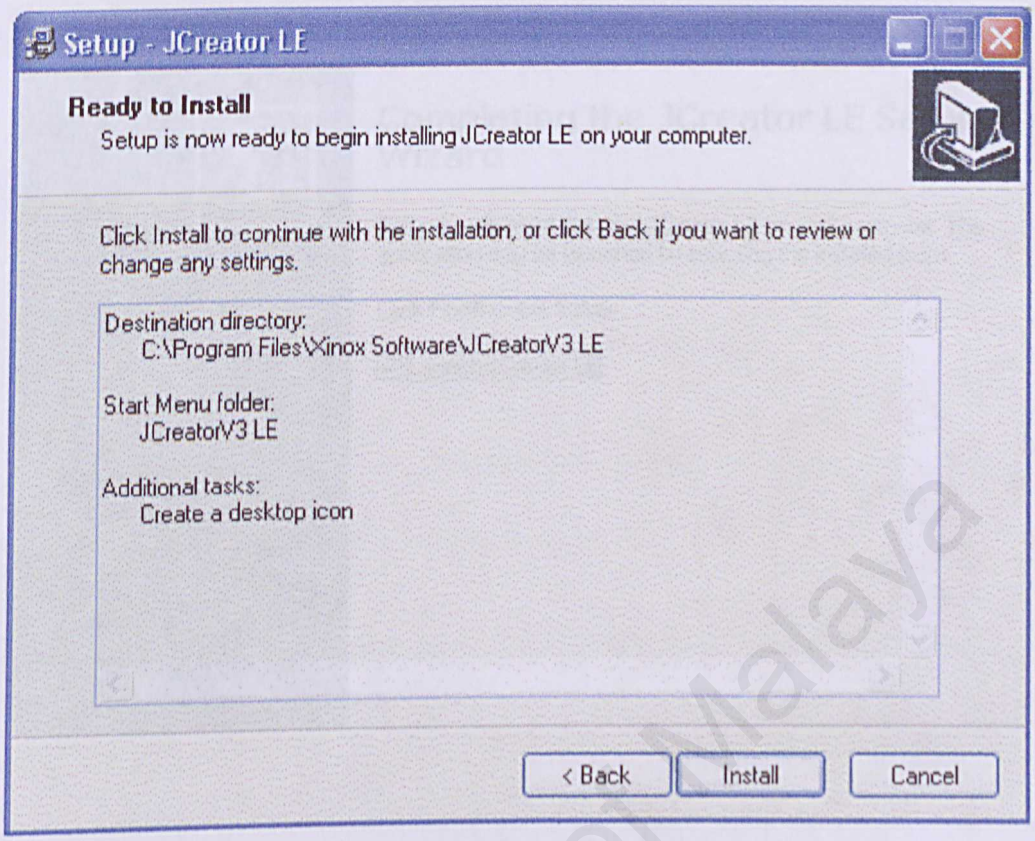


Figure 9.18 : JCreator LE Ready to install

Step10. JCreator LE Setup complete and ready to be used

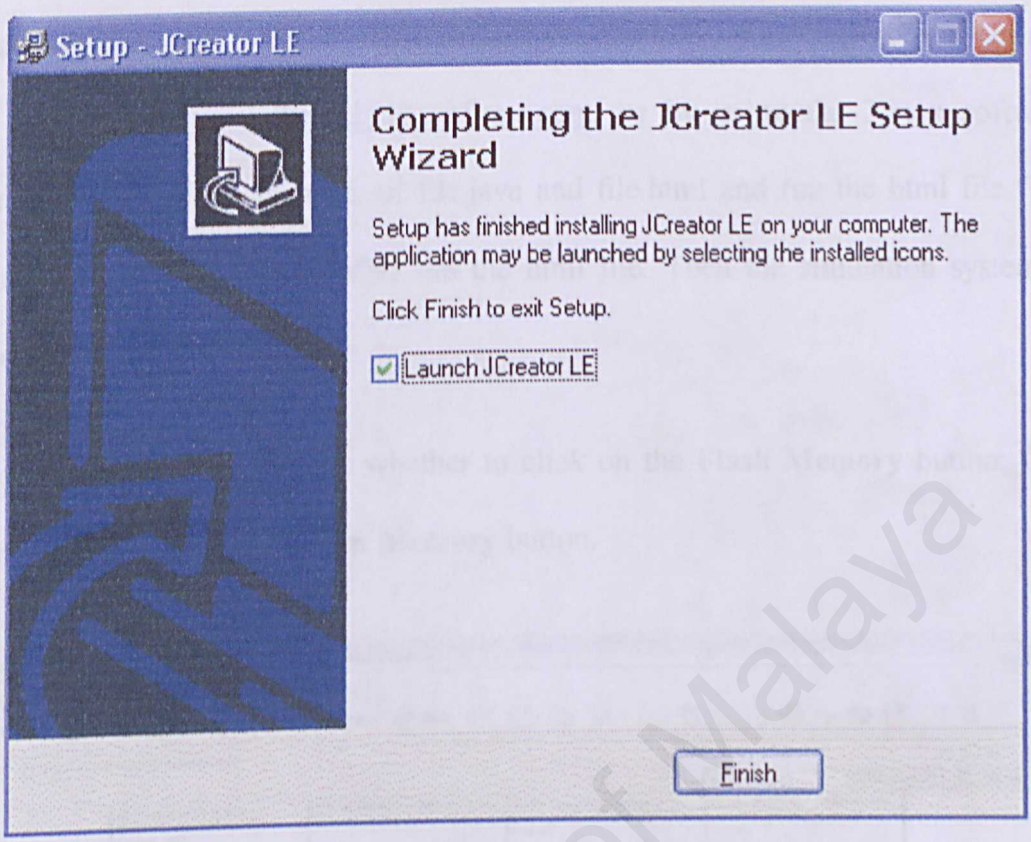


Figure 9.19: JCreator LE setup complete

Opening the Flash Memory Display

There are two ways to display this simulation system by using the web browser and go to the http://www.thesis3182.netfirms.com, or by using the Xinox software JCreator LE v2.50 with open all file.java and file.html and run the html file. The applet window will appear after run the html file. Then the simulation system is ready to use.

At the main window, choose whether to click on the **Flash Memory** button, **Cell Flash Memory** button or **View Memory** button.

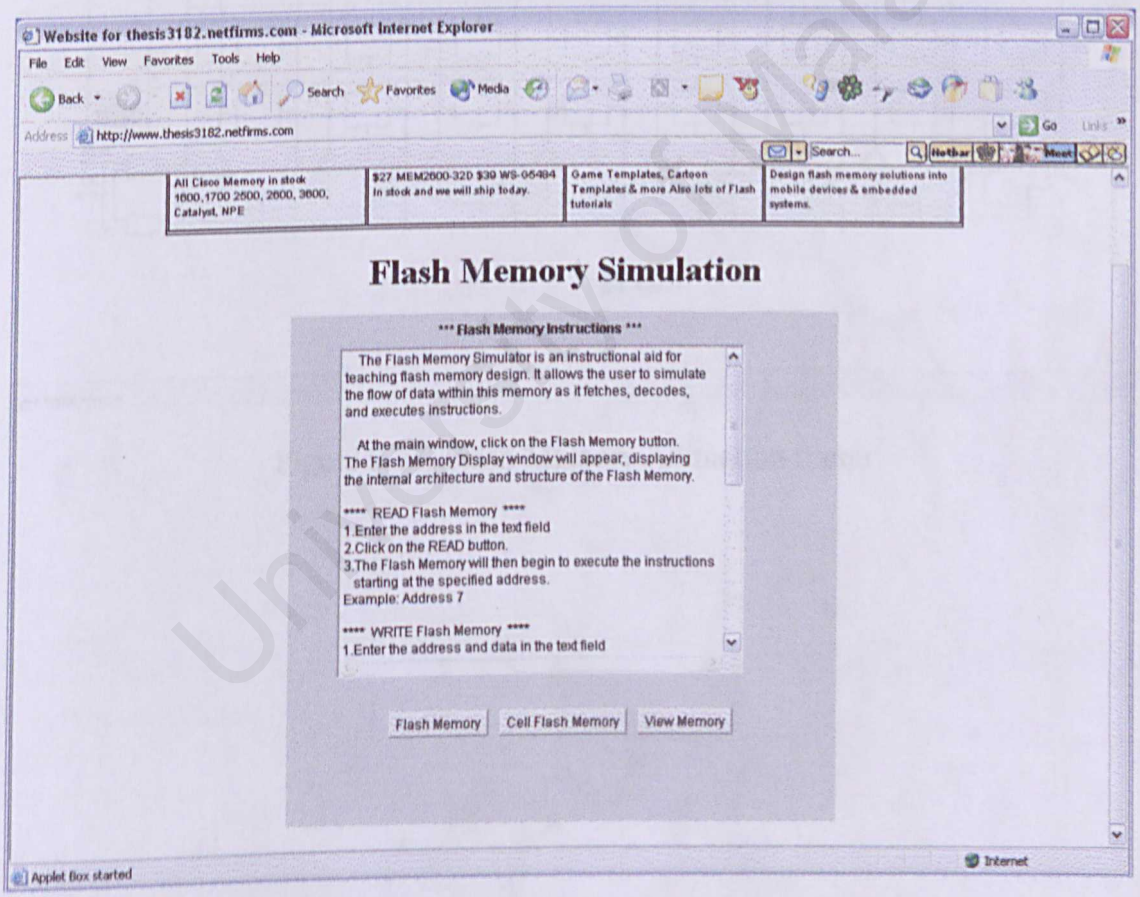


Figure 9.20 : Main window

Click on the Flash Memory button make a **Flash Memory Display** window appear and displaying the internal architecture and structure of the Flash Memory.

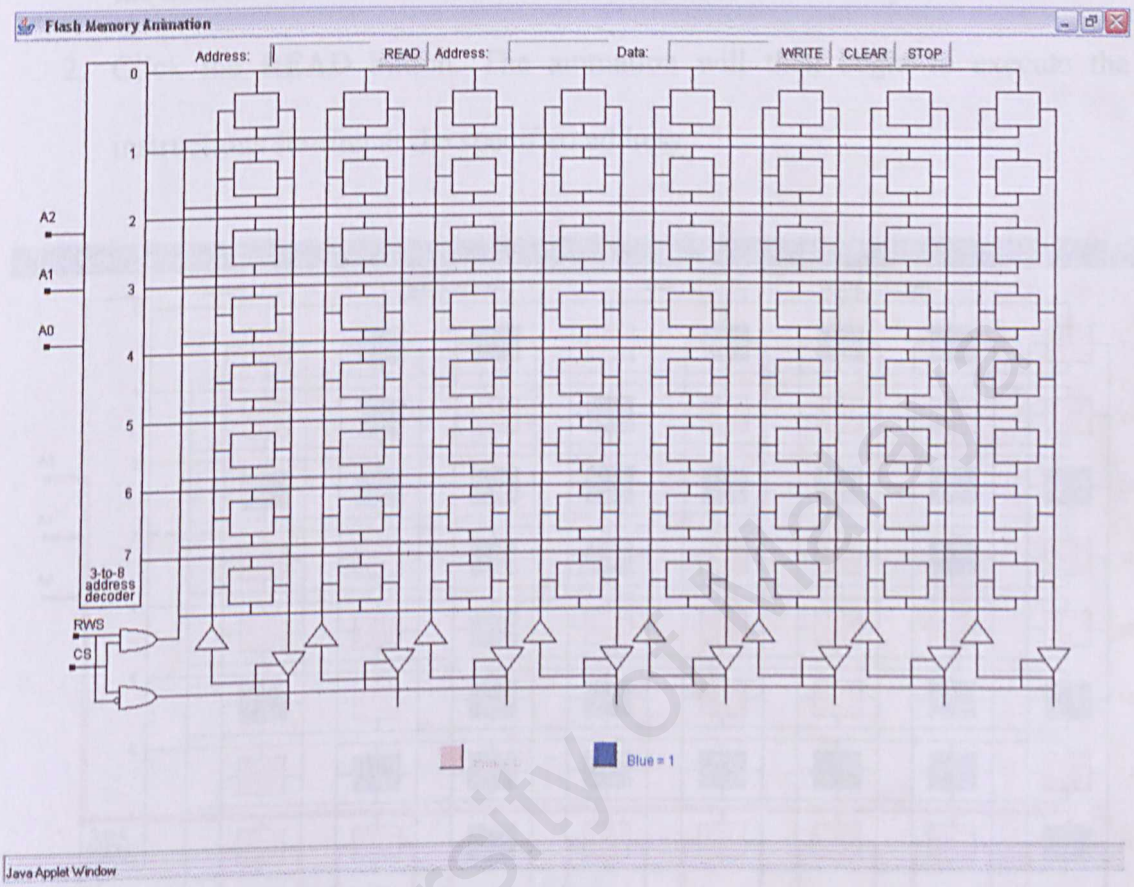


Figure 9.21: Flash Memory Animation frame

Starting a Read Flash Simulation

1. At the **Flash Memory Display** window, enter the address from (0 to 7) in the address text field to read the content of the flash memory.
2. Click the **READ** button. The animation will then begin to execute the instructions starting at the specified address.

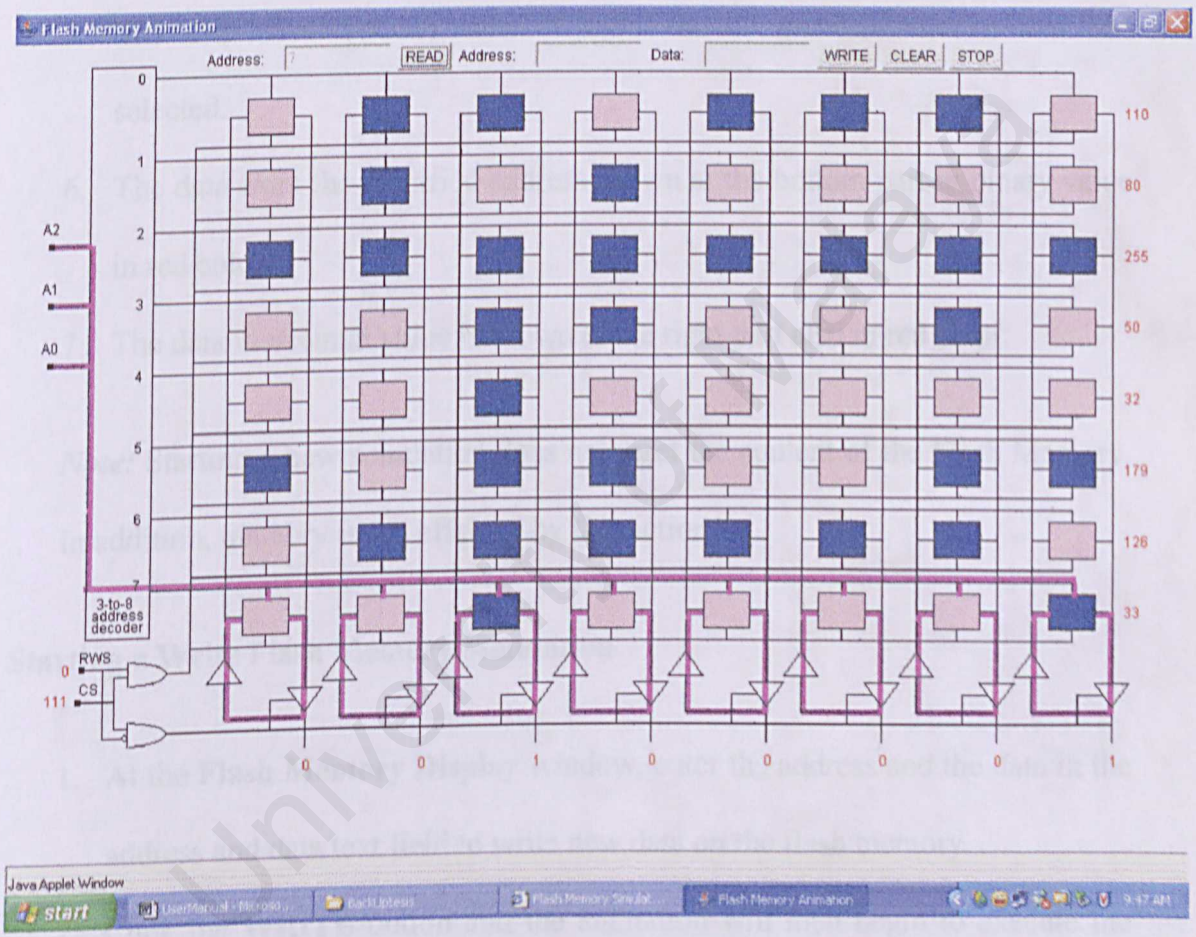


Figure 9.22: Flash Memory Animation frame

3. The animation in the magenta color, show how the flash memory work. It begins from decoder and moves to the each cell that stores the value either 0 in pink color or 1 in blue color.
4. The Read / Write select (RWS) status is 0 value means that the memory will read its content at the location specified by the address lines and make it available at its output ports.
5. The chip select (CS) status is based on the specified address that was selected.
6. The data from the specified address shown at the bottom part in binary value in red color.
7. The data in decimal value is shown at the right part also in red color.

Note: Starting a new simulation does not reset the content of the Flash Memory. In addition, memory is not affected by this action.

Starting a Write Flash Memory Simulation

1. At the **Flash Memory Display** window, enter the address and the data in the address and data text field to write new data on the flash memory.
2. Click the **WRITE** button and the animation will then begin to execute the instructions starting at the specified address and data.
3. The animation in green color will start from decoder and move to the each cell to store data.

4. The value of data that the user entered will start from processor in binary value and move to the flash memory cell based on its value.
5. At this event, Read / Write select status is equal to 1 show that the memory will write the content presented on its input ports into the location specified by the address lines.
6. The chip select status is equal to the selected address.

Figure 9.23 : Flash Memory Animation frame

Overwrite the Existing Data

1. The existing data in the flash memory can be change by using the same function read and write button.
2. Firstly, enter the address value to read the existing data in flash memory then enter the same address and new data to change the value of flash memory.
3. Finally, the existing data in flash memory will be change to the new data, as shown below.

Stopping a Running Animation

At the Flash Memory Display window, click on the STOP button will immediately stop animation.

Clear the Flash Memory Content

At the Flash Memory Display window, click on the CLEAR button will immediately stop instruction execution and the contents of the flash memory will be cleared immediately.

Opening the Flash Memory Cell Display

Click on the Flash Memory Cell button make a Flash Memory Cell Display appear without any instruction.

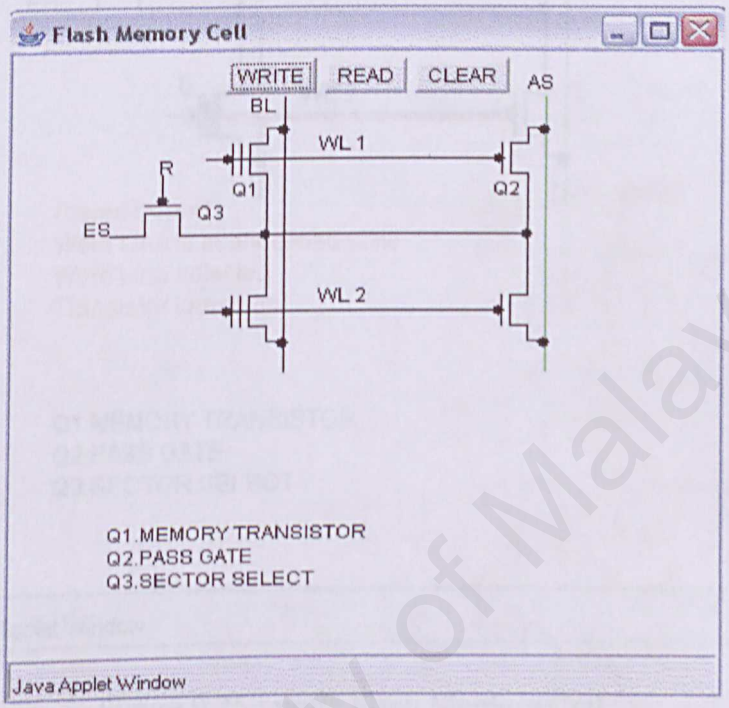


Figure 9.24 : Flash Memory Cell

Starting a Write Flash Memory Cell Simulation

1. Click WRITE button will show the animation how the flash memory cell work.
2. The stored data that the value is equal to one

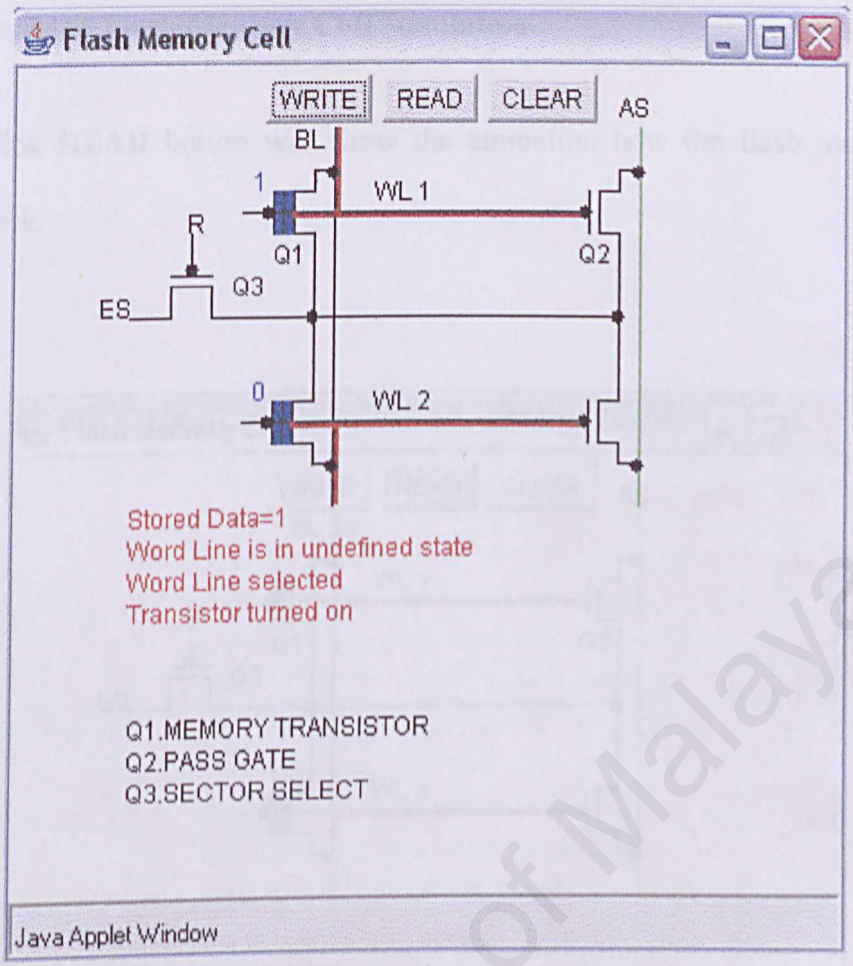


Figure 9.25 : write Flash Memory Cell

Starting a Read Flash Memory Cell Simulation

1. Click **READ** button will show the animation how the flash memory cell work.

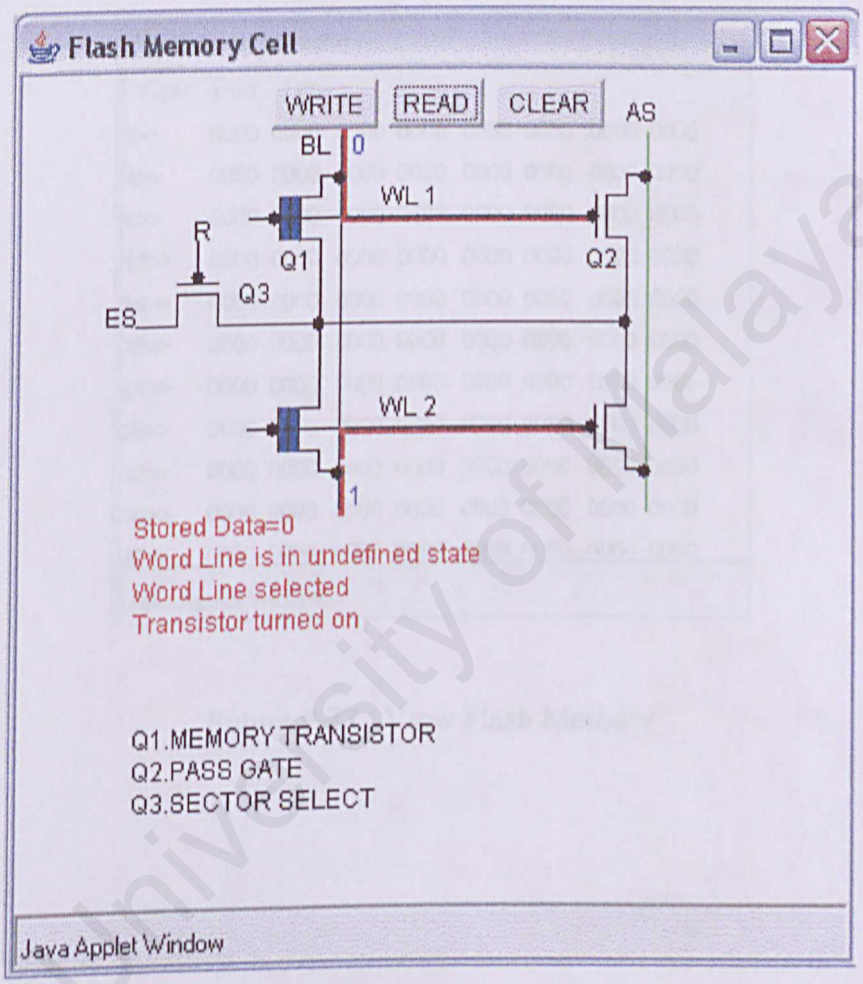


Figure 9.26 : Read Flash Memory Cell

Opening the view Flash Memory

Click on the view Flash Memory button make a view Flash Memory appear without data. Below are the figure shows that flash memory in database format in binary value.

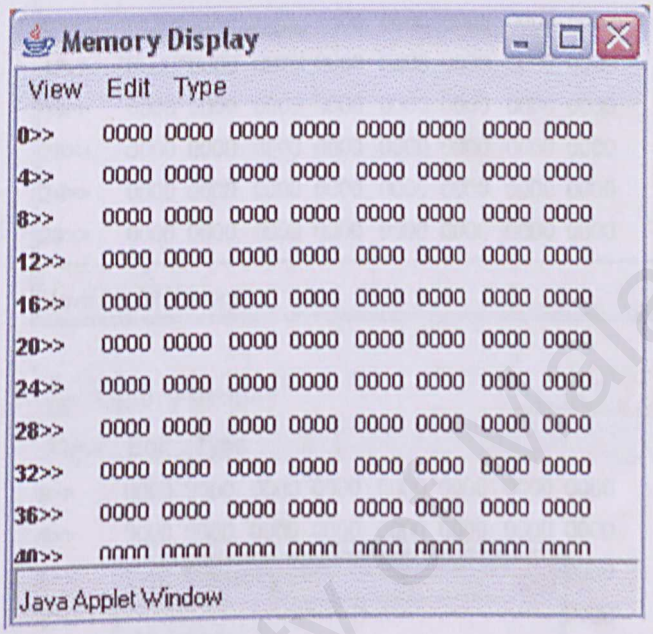


Figure 9.27 : View Flash Memory

Editing a Byte of Memory

1. At the Memory Display window, click Edit Memory on the Edit menu. The Edit Memory dialog box will appear.
2. Enter the address of the byte in the Address and Data text field.

3. Click the OK button to continue with the change operation. Otherwise, click the Cancel button.

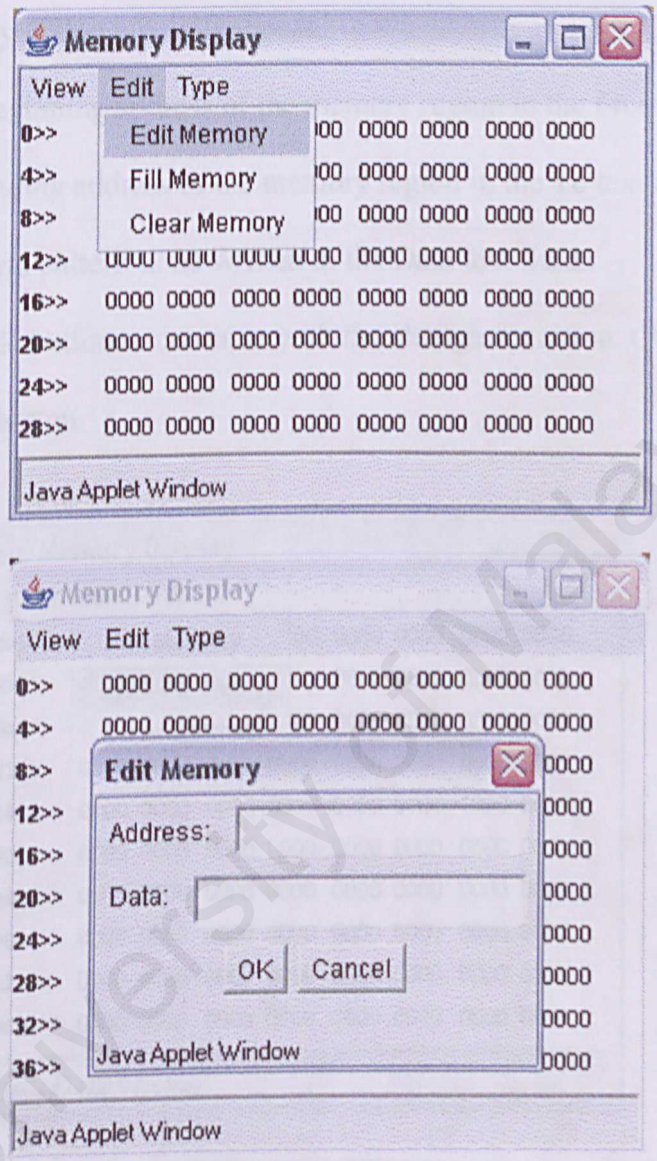


Figure 9.28 : View Edit Memory

Filling a Region of Memory

1. At the Memory Display window, click Fill Memory on the Edit menu. The Fill Memory dialog box will appear.
2. Enter the beginning address of the memory region in the From text field.
3. Enter the ending address of the memory region in the To text field.
4. Enter the byte pattern to be written in the Data text field.
5. Click the OK button to continue with the change operation. Otherwise, click the Cancel button.

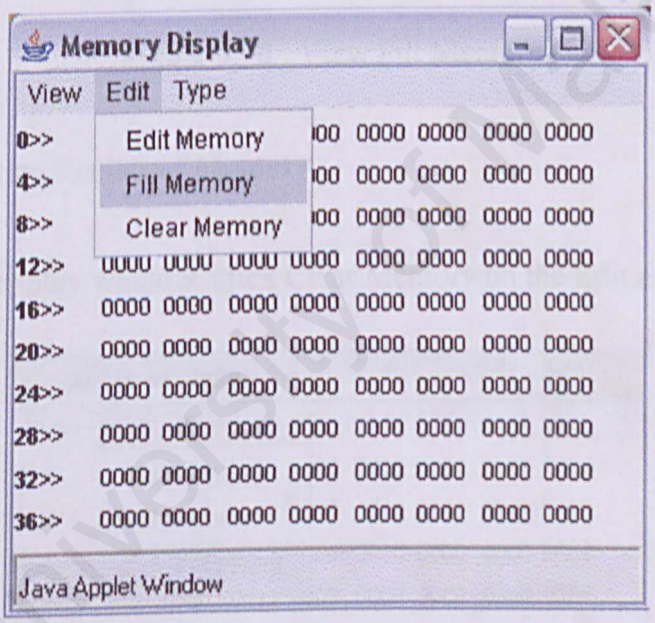


Figure 9.30 : View Fill Memory

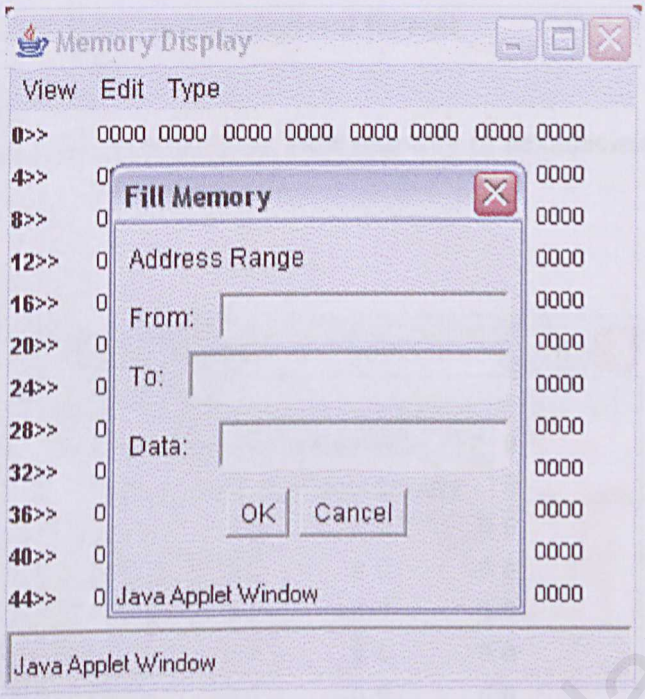


Figure 9.30 : View Fill Memory

Clearing the Entire Region of Memory

At the Memory Display window, click Clear Memory on the Edit menu.

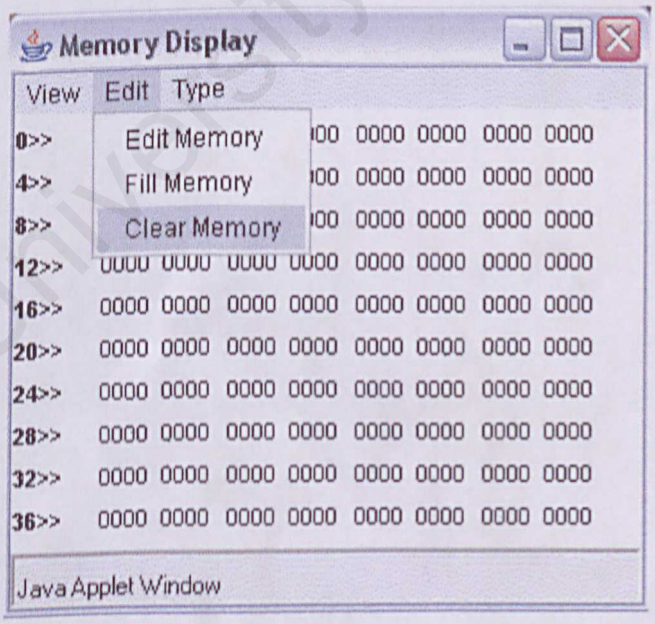


Figure 9.31 : Clear View Memory

View the value of Memory in hexadecimal format

At the Memory Display window, click view memory in hexadecimal on the type menu.

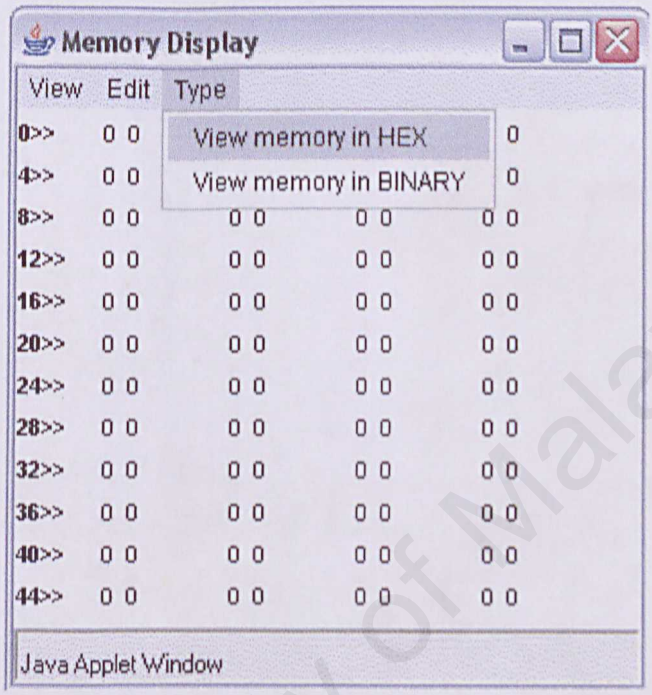


Figure 9.31 : View Memory in Hexadecimal