Document Nbr: CSI426/kayproIIt-TP1.01
Date: November 14, 1998
Copy Nbr: ____

TEST PLAN

FOR THE

KAYPRO II EMULATOR

Distribution List

Name	**Organization**
Dr. Charles Howerton	Metropolitan College

Table of Contents

1. Introduction

This section includes the overview and scope of the Test Plan as well as benefits of implementing a test program.

1.1 Overview

Each project evolves its own software reliability standards. This document defines the specific test program to be followed by the Kaypro II Emulator project to ensure quality software is delivered. The goal of all test programs is to provide enough testing to ensure that the probability of unintended results due to hibernating bugs is low enough to accept. Software is delivered when requirements are met and the team and customer have a level of confidence that the software works correctly (consistently).

1.2 Benefits

Benefits to implementing a structured test program include:

- Defining test requirements early in the software development life cycle (SDLC)
- Ensuring tests are written to satisfy the conditions defined in the Requirements Document
- Isolating design discrepancies early in the SDLC
- Involving the customers in the planning and requesting their approval of the acceptance criteria prior to conducting system and acceptance testing
- Educating the customers on the SDLC process and training them on the system early in the development life cycle
- Providing an independent source for verifying the developed system
- Baselining the criteria for accepting the system.

1.3 Scope

The Kaypro II project is a Java implementation of a classic Kaypro II computing system. The project is a team effort. As such, it will be necessary to unit test, integration test, and system test the project.

The Test Plan includes

- Narrative sections describing test activities, approach, methodology and phases
- Outline for the system test procedures

The Test Plan establishes the test methodology and the set of requirements to be tested during system test and acceptance test. One objective is for the customer to agree with the proposed project acceptance criteria. It also serves to provide customer visibility into the planned test activities. Customer acceptance of the Test Plan represents a commitment to use the set of requirements defined in the Test Plan as the acceptance criteria during system or acceptance testing.

2. Test Activities During the SDLC

This section defines test-related activities performed by the Kaypro II emulator project development team and the test team. It presents the test approach and methodology to be employed to ensure system compliance to the requirements and design documents.

2.1   Test Team

The test team will include the same team members used to develop the code. Traditionally, this represents a conflict of interest. Because this is a university class with limited resources, it will be necessary to circumvent some traditional testing practices.

2.2 Testing Schedule

Because of the nature of the class, testing will be accomplished in conjunction with development, and at the completion of the development process. The amount of testing will be limited to the available time at the end of development.

2.3 Goals for Testing

The first goal of testing is bug prevention. Prevented bugs reduce costs, cause no code to be corrected, and cannot wreck a schedule. Bug prevention is a state of mind. Developers need to make software testable. Good programmers test early and often to keep their bugs to a minimum. This is why thorough unit testing is so crucial.

The second goal of testing is to discover symptoms caused by bugs by using many small detailed tests. To make these tests repeatable, test cases must be designed, tested, and documented. Only then, can useful regression testing take place.

Finally, bugs must be clearly diagnosed so they can be easily corrected.

2.3.1 Requirements Validation (Define Requirements Phase)

The project team must influence the language used in the Requirements Document to ensure the requirements are testable.

During the requirements creation, each team member is responsible for assuring consistency and completeness of requirements design.

Each team member is responsible for a given area. Each team member is responsible for defining interfaces to and from their assigned areas, and assuring compliance for that interface.

2.3.2 Design Validation (High-Level and Detailed Design Phases)

There are two types of testing, functional and structural. The project team performs structural testing (implementation details) by validating the system, software, and database designs. The project team also performs functional testing (the users point of view) by validating the inputs and outputs against the requirements and their knowledge of the customer operational environment.

Each team member is responsible for contributions to the overall design

2.3.3 Unit Test and User Interface Usability (Construct Phase)

The construct phase consists of code and unit test performed by the development group. The development software engineer tests the functionality and usability of each unit. A unit is the smallest testable piece of software; in other words, it can be compiled or assembled, linked, loaded, and put under the control of a test harness. A Unit Test Condition form is used to document the conditions that are verified. These verified conditions are grouped into test cases. Each test case is documented on a Unit Test Case form. To pass unit test, the unit must satisfy its functional specification or intended design structure.

At this phase in the life cycle, the tester is concerned with structural testing. This includes implementation details such as programming style. A stylistic criteria is developed to define what is meant by a "good" program. Examples of this criteria are testability and openness and clarity. Such styles can do much to prevent bugs.

why (objective) -- verify software quality; prove the software meets requirements as specified in the design
when -- after the program has been coded, but before it is migrated to integration test
who -- the development programmer responsible for that particular module

where -- programmer's development work station *(identify specific hardware and software, including version number, to be used)*

what -- exercise all program functions (within a module), logic conditions, and user interfaces. Includes:

- ◊ Logic statements
- ◊ Edit conditions
- ◊ Error/abort condition testing
- ◊ Screen handling
- ◊ Report generation
- ◊ Calculation verification

deliverables -- module documentation to include test cases, test results, etc.

exit criterion -- team lead review and approval of module documentation; modules checked into the software library

2.3.4 Integration Testing (Integrate and Test Phase)

Integration is a process by which components are aggregated to create larger components. Integration testing involves integrating the modules into subsystems and ensuring the modules work together. Integration testing is conducted to show that even though the components were individually satisfactory, the combination of components are correct or consistent. Examples of possible inconsistencies include inconsistent data validation criteria and inconsistent handling of data objects. As each subsystem is successfully verified, it is checked into the software library.

The test engineer generates system test scenarios during this phase of the SDLC. Software is grouped by logically-related programs. Functionally-related test scenarios are then developed. For each test scenario, the test objectives are identified and one or more test cases are written. Each test case/script includes the detailed steps necessary to execute the test case.

When integration testing is complete, the System Test Readiness Review (TRR) is conducted. The expanded Test Plan is reviewed at the TRR.

why (objective) -- to provide early identification of problems; secondary goal: familiarize the test engineers with the software in order to draft the formal acceptance test procedures

when -- after the program has been unit tested

who -- development lead and the test team

where -- integration test environment: test-dedicated work stations and test server *(identify specific hardware and software, including version numbers, to be used)*

what -- exercise all functions; follow a "test and fix" pattern. Include testing:

- ◊ Data checks:
  - ⇒ nominal values
  - ⇒ minimum values
  - ⇒ maximum values

- ⇒ erroneous input values
- ◊ Functionality
- ◊ User Interface tests (usability and meeting standards)
- ◊ Regression testing after fixes are implemented: rerun all module tests at the unit test level; then, rerun all subsystem tests that use the revised modules.

deliverables -- subsystem test results; successfully tested components checked-in to the software library

exit criterion -- customer and management acceptance of subsystem components and test procedures, and successful completion of the Test Readiness Review

2.3.5 System Testing (Integrate and Test Phase)

The system test is an end-to-end test that ensures all of the subsystems are integrated correctly. System testing concerns issues and behaviors that can only be exposed by testing the entire integrated system or a major part of it. It includes testing for security, configuration sensitivity, start-up and recovery. It also involves demonstrating that performance meets or exceeds expectations when there is a load on the system.

Data flow anomalies, the unreasonable things that can happen to data, are also explored. The tester selects paths through the program's control flow to explore sequences of events related to the status of data objects. Enough paths are exercised to assure that every data object has been initialized prior to use or that all defined objects have been used for something. The data flow testing will show each path as one of the following:

- *successful path*: the path works and is "clean" (straightforward)
- *unsuccessful path*: the path does not work as it is supposed to or the path is not needed
- *bizarre path:* the path does what it is supposed to do but it does it in a convoluted way that would make the program difficult to maintain and test

From this phase forward in the life cycle, the system is managed via change procedures. When discrepancies are identified, a Discrepancy Report/Enhancement Request (DR/ER) is filled out and submitted by the team lead. The software is fixed only after the DR/ER is properly dispositioned according to change procedures.

The System Test and Acceptance Test Checklist is useful for ensuring all test activities are conducted.

System testing includes the following test methods:

- Inspection (I) - a visual examination to empirically establish that a requirement has been met. An example is comparing the output printed to the screen to the screen layout in the Requirements Document.

- Analysis (A) - a technical and/or mathematical evaluation using mathematical representations (e.g., math models, algorithms, equations, etc.), charts, graphs, data reduction, or representative data to show that a requirement has been met. An example is entering data to produce a graph and then analyzing the graph to determine if it accurately depicts the data.
- Demonstration (D) - the comprehensive exercising of a software function under all applicable conditions, the observed results of which show that a requirement has been met. An example is navigating through a series of screens showing that the screen hierarchies are in the correct sequence and that they make sense.
- Test (T) - a systematic exercising of the applicable item under all appropriate conditions with instrumentation followed by collection, analysis, and evaluation of quantitative data. An example is inputting data values (nominal, maximum, minimal, erroneous), comparing the output to the expected results, analyzing the unexpected results, and evaluating whether those unexpected results are acceptable.

why (objective) -- demonstrate compliance to all system functional, performance, and user interface requirements

when -- after successful completion of the Test Readiness Review

who -- the test team with management-level customers witnessing critical tests

where -- dedicated test environment using the new production equipment *(identify specific hardware and software, including version number, to be used)*

what -- exercise step-by-step test procedures with expected results criteria.
Includes:

◊ Installation check-out
◊ Start-up check
◊ Data checks:
 ⇒ minimum values
 ⇒ nominal values
 ⇒ maximum values
 ⇒ erroneous input values.
◊ Functionality tests
◊ User Interface tests
◊ Performance/Stress Testing
◊ Regression testing after fixes are implemented. In other words, rerun all module tests at the unit test level; then, rerun all subsystem tests that use the revised modules. Finally, rerun the test cases that had unexpected results and all related test cases that may have been affected (in other words, test cases that exercise the revised modules).

deliverables -- Test Plan/Test Procedures with test results (pass, fail, or suspend and comments as needed) and baselined software (product baseline)

exit criterion – Team Sign-off.

2.3.6 Acceptance Testing (Deliver Phase)

The system is managed via change procedures during acceptance testing. When discrepancies are identified, a DR/ER is filled out and submitted by the team lead. The software is fixed only after the DR/ER has been properly dispositioned according to change procedures. The system development effort culminates with acceptance of the system by the customer; the customer and Network Operations Director complete the Acceptance Test form.

why (objective) -- obtain customer approval; ensure that the system installation process has been completed successfully and that the system is ready for initial operations
when -- after system test sign-off
who -- the test team with management-level customers witnessing all tests
where -- dedicated test environment using the new production equipment *(identify specific hardware and software, including version number, to be used)*
what -- exercise subset of system test procedures that were agreed to by the customer
deliverables -- production system
exit criterion -- Acceptance Test sign-off (customer acceptance of the production system)

3. Test Procedures

Test cases are grouped into operational scenarios, also referred to as "procedures". Each test case is described and further broken down into steps that include expected results.

3.1 Emulation

The Kaypro II emulation shall be tested primarily by inspection
Platform shall be verified visually
Emulation shall be accomplished by viewing the Kaypro emulator and actual machine side-by-side. A representative set of programs will be run on each one. The tester will verify similar operation.
Modularity shall be confirmed by viewing the actual code. The code shall be written in a modular way.
Pure Java, JDK version, interface conformity shall be confirmed visually

Areas of emulation test

- The emulation program shall emulate the Kaypro II model of the Kaypro product line
- The user shall manipulate the emulated version just as they would the original
- The emulation shall contain debugging features, in addition to the original functionality
- The Kaypro II emulation shall be coded in such a way as to facilitate easy additions and expansions to the system.
- The emulation shall be coded in a modular way; such as logical Java classes
- The Kaypro II emulation shall be implemented in Java

- The implementation shall be pure Java (e.g. No Microsoft extensions).
- The Kaypro II emulation shall be implemented as an applet.
- The Kaypro II emulation shall be made available via the world wide web
- The Java interface shall be kept minimal, to afford quicker load times
- The Implementation shall use JDK 1.1.6

3.2 Z-80 CPU Emulation

The CPU shall be tested in three ways: 1. Unit test 2. Integration test 3. Kaypro II software code test
Each command shall be coded, and visually inspected. A representative command(s) from each set may be tested via a small developer generated program. A representative command or commands will be all that is tested, since it will be unpractical to test each command given the length of the class.
Integration testing will involve placing this module within the overall system. The test will assure that all functions are supported. Test code will be created where necessary to test external function interaction. The debug mode will be very useful in this stage.
Once integration testing is complete, actual Kaypro II code will be executed. The goals are: 1. Execute Z-80 code while stepping through with a debugger 2. Execute through RAM/ROM page transitions 3. Execute ROM code 4. Load floppy image 5. Load CP/M 6. Load/save a simple program 7. Execute a standard Kaypro program

Areas of CPU test

- The Kaypro II utilizes the Z-80 Processor. The CPU emulated shall be the Z-80
- The CPU instruction set shall conform to those presented in the Zilog Z80 Microprocessor Family User's Manual, Part number Q1/95 DC 8309-1.
- The RLA command as documented in the Zilog user's manual contains an error in the rotate
- The non-maskable interrupt may not be disabled.
- Maskable interrupts may be disabled via CPU instruction.
- Mode 0 interrupts shall be implemented.
- Mode 1 interrupts shall be implemented.
- Mode 2 interrupts shall be implemented.
- The Z-80 CPU shall support implementation of the RESET line. When this line is activated, the CPU shall force a jump to location 0x00 upon completion of the current command.

3.3 Hardware Emulation

The hardware functions will be tested

1. Unit test
2. Integration test
3. Kaypro II software code test

The hardware object codes will be carefully inspected and viewed for any errors or bugs. Most functions within the hardware are used to send and receive data to other objects, therefore this will be tested with input and output variables to make proper communication between objects. All flags and primary hardware settings that is required by other objects will be tested and made sure that they are set correctly.

Since the hardware is the central nervous system within the kaypro II we must test each individual component to insure proper communication between devices and components. The goals are:

1. Initializing the Video memory must be tested for proper display and screen alignment after the kaypro II starts.
2. Settings of the memory bank switch will also be tested with the needed devices.
3. All Interrupts and NMIs will be tested to insure proper settings with the needed component.
4. Sending and receiving memory and memory location will be tested for proper usage.
5. Writing and reading memory will be tested accordingly to its usage and its contents within the memory.
6. Testing on the passing of RAM/ROM/Video memory will be required for proper content and memory location.
7. Sending and receiving CPU in and out commands will be tested for proper usage and settings of modes
8. Sending and receiving SIO, PIO, Floppy ports and data will be tested for proper and correct settings on ports.

Areas of hardware test

- Keyboard
- Screen
- Memory (Ram/Video/RAM)
- Bank switching
- I/O Ports
- Disk Drives
- Motor control, indicator lights, etc
- NMI
- INT
- Reset
- Read memory
- Write memory
- Port commands

3.4 Port Emulation

The hardware functions will be tested 1. Unit test 2. Integration test 3. Kaypro II software code test
A write and read from each channel of each port will be tested for appropriate values and functionality.
The hardware and Ports will be plugged into the overall system including the CPU and GUI and the entire running emulator will be the test for the hardware ports.
Same as above.

Areas of port emulation test

- Floppy disk controller/Drive
- Serial I/O
- Keyboard
- Beeper
- Parallel port/printer
- Baud rate generators

3.5 Video and Memory

The hardware functions will be tested 1. Unit test 2. Integration test 3. Kaypro II software code test
Each memory address and content shall be coded and visually inspected in the two-dimensional memory array contained in the hardware.
Integration test for the memory will be tested through the debugging modes for assurance of correct contents. With the debug mode it is possible to see each step that is being received and sent.
Testing for proper usage and handles for the memory will also be a critical point for all other components that would need to obtain information within the memory. This will be tested in the hardware with conjunction with the necessary needed devices.

Areas of Video and memory test

- The emulated video RAM shall be 4K bytes
- The emulator shall support an 80 character per line by 24 line text screen
- The emulator shall support memory-mapped video RAM banking

3.6 Fast and Real Mode Display

The hardware functions will be tested

1. Unit test 2. Integration test 3. Kaypro II software code test
Reading a known set of values in an array the exact size of the video memory will do the fast and real mode unit tests.
Integration testing will involve placing this module within the overall system. The test will assure that the fast and real mode drawing functions are reading the video memory and correct output. The debug mode will be very useful in this stage.
Once integration testing is complete, actual Kaypro II code will be executed. The goals are: 1. Start the Kaypro II emulator testing the fast mode. 2. Testing the real mode by switching to real mode and restarting the emulation. 3. Fill the text area with text to determine if 80X24 characters are being output to the text area or screen.

Areas of fast and real mode test

- Fast mode will use a typical Java text box to display characters. Fast mode will not attempt to emulate the exact Kaypro II character generator ROM.
- Real mode will display characters as they are defined in the Kaypro II character generator ROM. Real mode will require extracting data from the character generator ROM, converting it to programmatic form, and inserting it in the emulator.

3.6.1.1 Debug

The hardware functions will be tested 1. Unit test 2. Integration test 3. Kaypro II software code test
The debug functions will split between the UI and the CPU
The CPU will be unit tested to determine if the debug functions operate correctly
Once the UI is connected (integrated), it will exercise the CPU functions
The tester will confirm each debug function

Areas of debug test

- The Debug Mode Off is default when the emulator is started.
- Toggle Step Mode/Run Mode
- Single Step
- Set Breakpoint
- Generate NMI
- Display Memory
- Options

3.7 User Interface

The hardware functions will be tested 1. Unit test 2. Integration test
All buttons will be tested for correct function. Test user input from all text boxes for correct input processing.
Integration testing will involve placing this module within the overall system. The test will assure that all functions are supported. Test code will be created where necessary to test external function interaction. The debug mode will be very useful in this stage.

Areas of user interface test

- Reset button
- Change Disk A and B
- Step and Run Mode
- Set Breakpoint
- Generate NMI
- Memory Dump
- View options

3.8 Serial I/O

The Individual Port functions will be tested 1. Unit test 2. Integration test 3. Kaypro II software code test
The SIO object will test that all bit patterns written and read are valid. A replication of how a program writes and reads from all ports of the SIO will be tested.
The SIO will be intergrated with the hardware object then a test suite will be designed to test the SIO with the hardware. Then the SIO and hardware will be plugged into the CPU and UI objects.
The SIO works if the console echos the keys.

Areas of Serial I/O test

- The SIO chip channel B provides for keyboard input and speaker output. These functions shall be implemented on a function level. That is, specific configuration of the SIO channel A shall be ignored, except where it influences functional operation.
- SIO baud rate may be programmed via command and data registers. The Kaypro II emulation shall accept all values for baud rate, parity, start word, and stop words, but shall operate at system speed. That is, these values shall be ignored.
- Synchronous mode shall not be supported.

3.9 Floppy Emulation

The hardware functions will be tested 1. Unit test 2. Integration test 3. Kaypro II software code test
All registers shall be tested for correct read and write bit patterns. All possible modes will tested meaning, Read, write, seek, step in, step out. The array that holds data shall be tested for out of bounds problems.
The floppy will be plugged into the hardware and tested with the hardware to ensure port connectivity. The hardware and floppy will then be plugged into the entire system and tested.
If the kaypro boots up the floppy works.

Areas of floppy emulation test

- The emulation shall contain two virtual floppy disk drives.
- Electronic copies of actual disks shall be obtained and translated into emulator format
- Disk data shall be copies of individual sector data
- Track and sector numbering shall be maintained
- Data alignment shall be maintained

3.10 Parallel/Printer Port

The hardware functions will be tested 1. Unit test 2. Integration test 3. Kaypro II software code test
The PIO port will be tested by coding a make shift gui to simulate a printer screen then writing characters to the PIO chip via a test suite and seeing if this works properly.
The PIO port will be opened up in the hardware object and simulated writes to the printer port will be made.
The Kaypro system uses a CTRL+P to turn on the printer, if after this is done characters are echoed to the screen, the printer is working.

Areas of parallel/printer port test

- PIO 1 shall act as the printer port. It shall accept output via its data channel, and print the data to a separate window as ASCII text. The size of the printer buffer shall be limited only by the Java control used.
- Configuration commands sent to PIO 1 shall be accepted but ignored. Instead, PIO 1 shall always function as a printer port.
- PIO 2 shall function as an interface to internal and external devices. Although output to these devices shall be supported, some may not be needed for emulation operation. These devices include:
    - Disk motor control

- Printer strobe
- Printer busy
- Floppy drive select (A or B)
- Floppy side select (side of dual density floppy to read or write)

3.11 Baud Rate Port

Baud rate generators are not implemented in the emulation. They shall not be tested

3.12 Operating System

The operating system operation shall be confirmed visually
The operating system will be contained on disk images. These images are loaded into the emulation via virtual floppy drives. The tester shall confirm the CP/M 2.2 is indeed executing
The tester shall confirm that the OS performs • File read • File write • Disk directories • File deletion • Other OS functions

Areas of operating system test

- CP/M 2.2 shall be supported
- CP/M OS shall be supplied on track 1 of each virtual floppy disk.
- The emulator shall support loading of the OS from floppy drive A
- The CP/M operating system shall actually be run at the software level via an obtained copy of the CP/M operating system

4. Applicable Documents, Reference, and Glossary

This section contains title, author, and publication information for documents referred to or having an impact on the test program for this project. It also contains a glossary of applicable terms and acronyms.

4.1 References

Requirements Definition For The CSI426/Kaypro II Emulator
Requirements Specification Document For The CSI426/Kaypro II Emulator
Design Document For The CSI426/Kaypro II Emulator
Zilog Z80 Microprocessor Family User's Manual, Part number Q1/95 DC 8309-1
Z80.DOC, opcode reference, compiled by Sean Young (syoung@cs.vu.nl)
Synertek Data Book, 1983

4.2 Glossary

Applet	- An internet application that runs inside an internet browser.
Bank	- A Computer science term used to describe a specific chunk of Random Access Memory (See Random Access Memory).
BIOS	- Basic Input and Output System. A set of programs, addresses or routines inside RAM (See Random Access Memory) that provide certiain functionality for the computer system.
Bit	- The smallest value used to represent computer data or memory in a base 2 binary numbering system having a value of 1 or 0.
Buad Rate	- A term used to describe the ability of two devices ports (See Port) to establish a communication between them at a certain speed of data transfer.
Buffer	- A permanent or temporary area of storage used to hold data.
Byte	- A standard unit of measurement for computer data or RAM (See Random Access Memory)
CPU	- The Central Processing Unit.

DOS	- Disk Operating System.
I/O	- Input and Output.
Interrupt	- A term used to describe the need for a device or software program that must send a message to the Processor (See CPU) in order to gain its attention for useage.
Java	- A programming language with internet and platform independent capibility.
Memory	- Term used to describe an area of storage in a computer system. (See Random Access Memory, Bank, Buffer)
Memory Mapped	- A term used to describe how a computer system connects it I/O (See I/O) to RAM (See Random Access Memory).
OP Code	- The basic unit of instruction in a computer system. This is what is executed when a computer program is running.
Operating system	- The software program that manages low level hardwareand software management inside a computer system.
Parallel	- Data transmission that occurs in a side by side manor using multiple data lines to transmit data across a specific line.
Port	- A term used to describe the means for I/O (See I/O) internally and externally in a computer system.
RAM	- See Random Access Memory.
Random Access Memory	- The second fastest form of storage used inside a computer system commonly used for application execution and data storage.
Register	- The fastest form a storage inside a computer system usually constrained to a finite size depending on a particular system.
ROM	- Read Only Memory. Usually contains useful programs or data for Operating System (See Operating System), hardware and program useage.
Serial	- A term used to describe inline communication or data that is sent one after another either interanally or externally.

Virtual Machine	**-** A term used to describe a software or hardware program that emulates a given environment in which its executing applications are thought to be running.