

UNIVERSITY HONORS PROGRAM

SENIOR PROJECT - APPROVAL

Name: Luke Strobel

College: Arts & Sciences Department: Computer Science

Faculty Mentor: Dr. Pederson

PROJECT TITLE: Software Design of Virtual Manuscript  
Paper Module for use in Aural Training Program

I have reviewed this completed senior honors thesis with this student and certify that it is a project commensurate with honors level undergraduate research in this field.

Signed: Donald M. Pederson, Faculty Mentor

Date: 1-8-04

Comments (Optional):

# **Specification and Requirements of the Virtual Manuscript Paper Module**

**By Luke Strobel**

**For**

**the University of Tennessee Knoxville Music Department**

## **Introduction:**

This paper is the documentation of the development of the Virtual Manuscript Paper module for the University of Tennessee, Knoxville Music Department. In this process, I, Luke Strobel, am working with Dr. Pederson and Dr. Boling to develop a module that will be utilized by various educational programs developed by the university, the aural training software in particular, to have a standard interface by which the user of the program will enter in musical notation onto the screen. This notation will then be used by the program that calls the module to assess whether or not the notes entered were those desired (depending on the nature of the calling program), and assess a grade. This module is only concerned with the input and output of notational data and does not seek to run any tests. The actual testing will be done by the calling programs. The program provides an intuitive input interface while making the output appear as music is conventionally notated. It is written in a new language called Revolution, which is based on the stack/card model as used by Meta-card and Hyper-card.

## **Initial Prognosis:**

Upon the commencement of developmental activities, the project is extremely open ended. The initial requirements for the program were very few and the UI was to be built from scratch. Since there were no real initial specifications, a lengthy requirements gathering process had to be done in order to come up with a complete specification of the project. The inherent difficulty of this process was further compounded by the fact that the program was to be written in an untried rapid applications development (RAD) type language that loosely follows current modeling paradigms to an extent that it can be said to have a programming rubric of its own.

## **Language Concerns:**

The language the program is written in is called Revolution. A history and information not related to this program can be found at <http://www.runrev.com>. The language, as stated before, is a RAD type language that very loosely follows OOP. Normally, since RADs are mainly used for small uncomplicated programs or prototyping, a program of this type would not be written in such a language, however, due to other constraints the program will be written in Revolution.

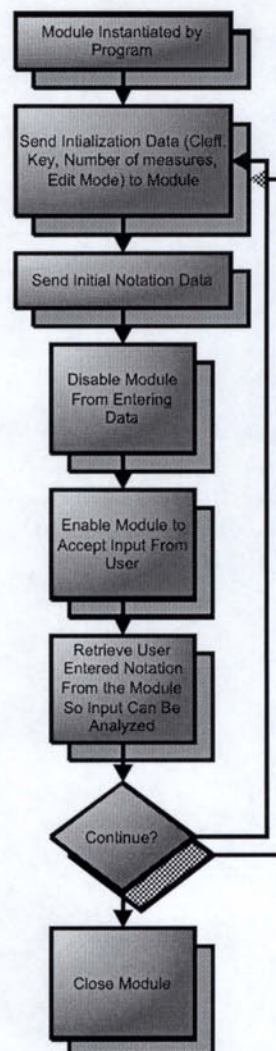
The language has several strengths. The first is that, unlike most RADs it is cross platform, which will allow the program to run on both the Windows and Macintosh platforms, thus increasing the number of possible users. Also, it has some useful built in functions to handle images, mouse input, keyboard input, and interrupts. Also, it is based on an event driven model, instead of a functional model, which is better suited to this program.

However, this language has many difficulties that need special attention in the development phase. First, there are no real programs of note written with this language and no documentation or books about the design model of this language. Therefore, using this language increases the design time since much prototyping has to be done to test the various possibilities. Also, since the language is a RAD, it is inherently slow and therefore speed must always be considered an issue in the design process.

## Specification and Prototyping:

Since this project started with the vague idea of creating a general notation I/O interface that could be used by other programs to perform tests, much research and prototyping had to be done in order to reach the desired design. While there are many notational I/O's models out there, there were special considerations for this design since this was a module that was to be used by other programs for testing purposes, and not just notating music on a computer. The program was designed to follow the following flowchart.

### VMP Flowchart



The first design model that was examined was the Finale notational program. From this program, the desirable features that it had are that it allows the insertion and deletion of notes, hotkeys for note input selection, and correct note formatting. However, this model could not be used for this specific module, since it groups the notes by measure, which is not fine-grained enough for the purposes of testing a student.

The next design that was examined was that of MacGAMUT. The features of this program were that it allowed you to enter measures while entering the other notation. Also, it stores notation, element by element, which gives a finer grained look at the notation entered than with finale. After experimenting with the approach, it was decided that we did not want our design to have either of these features since they did not facilitate grading and that they did not follow good transcribing practices. Having the fine grained approach of MacGAMUT and incorporating the measure input method that it has makes giving partial credit very difficult. Also, the standard practice of transcribing is such that one should take an approach where one blocks off the number of measures before putting notes in the measure. By having the module specify the number of measures along with the key, clef, and time signature, this procedure can be enforced.

After this approach was examined, the design that was considered was having the user enter notation into the measure, which was divided up into beats. While this approach had a better appearance than the MacGAMUT program (in that it followed standard notational practices as far as spacing and other formatting issues were concerned), it still was not suitable for an instructional purposes.

Using all of this information, the design was finalized to have a separate input and output interface. By doing this, the design would allow that information could be grouped by the beat and that the output would be formatted according to common practice. It also was decided that the design idea that would govern the decisions would be that of making "Virtual Manuscript Paper" (VMP), in that the process by which one entered notes into this module would try to imitate that of writing notes on a piece of manuscript paper. The requirements that were decided upon are delineated thoroughly in the next section.

## **Requirements:**

Through the design process, the following requirements were gathered. They have been split into two sections. The first section is the UI requirements. These requirements deal with how the interface is to appear on the screen. The second set of requirements is the program requirements, which deal with the data and how the module is to interact within its environment.

### **UI Requirements:**

1. Note size big enough for easy input but small enough for aesthetics.
2. User can scroll if the score is longer than the screen.
3. Soft color will be used for paper as to avoid undue eyestrain.
4. Two interfaces, one for input and the other for output.
5. Note formatting to follow standard formatting practices in terms of spacing, beaming, etc.

### **Program Requirements:**

1. Module to interact with various programs to get notational input from students for various musical exercises using a separate module for input and output through the mouse and keyboard.
2. Use single staff or Grand Staff.
3. Can use Treble, Bass, Alto, Tenor Clefs.
4. Meter can be 2/4, 4/4, 3/4, 6/8, 9/8, or 12/8.
5. Can enter in notes from 2 ledger lines below the staff to 2 lines above.
6. Key Signature can be up to 6 flats or 6 sharps.
7. User can enter notes and rest values: Whole, half, quarter, 1/8, and 1/16 notes or rests.
8. When entering notes, you can re-enter that note value without having to select the note value button again.
9. User can triple three quarter notes or three eighth notes.
10. User can add one dot, a sharp, a flat, a natural, a double sharp, or a double flat to a note and one dot to a rest.
11. User enters/erases the beaming on the notes (but the computer formats the beaming and the notes so that the music is readable) and can tie (but not slur) notes.
12. Finale interface to select note values from keyboard. (and up and down arrows raise and lower pitch respectively.) Use + or – to change accidental. Use 9 key to toggle enharmonics (for example – F# to Gb).
13. Input can converted to MIDI format so that the answer can be played by the user for comparison if answer is wrong.

14. The note-formatting/output module will be able to also be used in a “conductor” type program where the user hears a melody and addresses how it deviates from the written notes (or some variant of this type of program) and in a rhythmic dictation where the pitches are given and the user just adds the appropriate time values to the notes.
15. Up to 8 measures will be used.
16. No double bar at end of music.
17. Will allow user to enter in more notes than in one beat and/or measure without warning. Will allow the user to enter 4 notes into one simple time beat (or six in one complex time beat).
18. Allow VMP to be set multiple times so that error information can be displayed.

# Design Development of Virtual Manuscript Paper

University of Tennessee, Knoxville  
Department of Music  
Luke Strobel

## **Introduction:**

This document contains the design of the VMP module. The main part of this documentation is the artifacts following the Cleanroom software development method. Using the Cleanroom method, concise, mathematically complete descriptions of the operation of the software is produced, which can be directly translated into code. This paper also contains information such as data structure descriptions, coding style, etc. which is pertinent to the documentation and description of the design of this module. Also, in some instances, the thought process behind design decisions is discussed.

## **Virtual Manuscript Paper Requirements:**

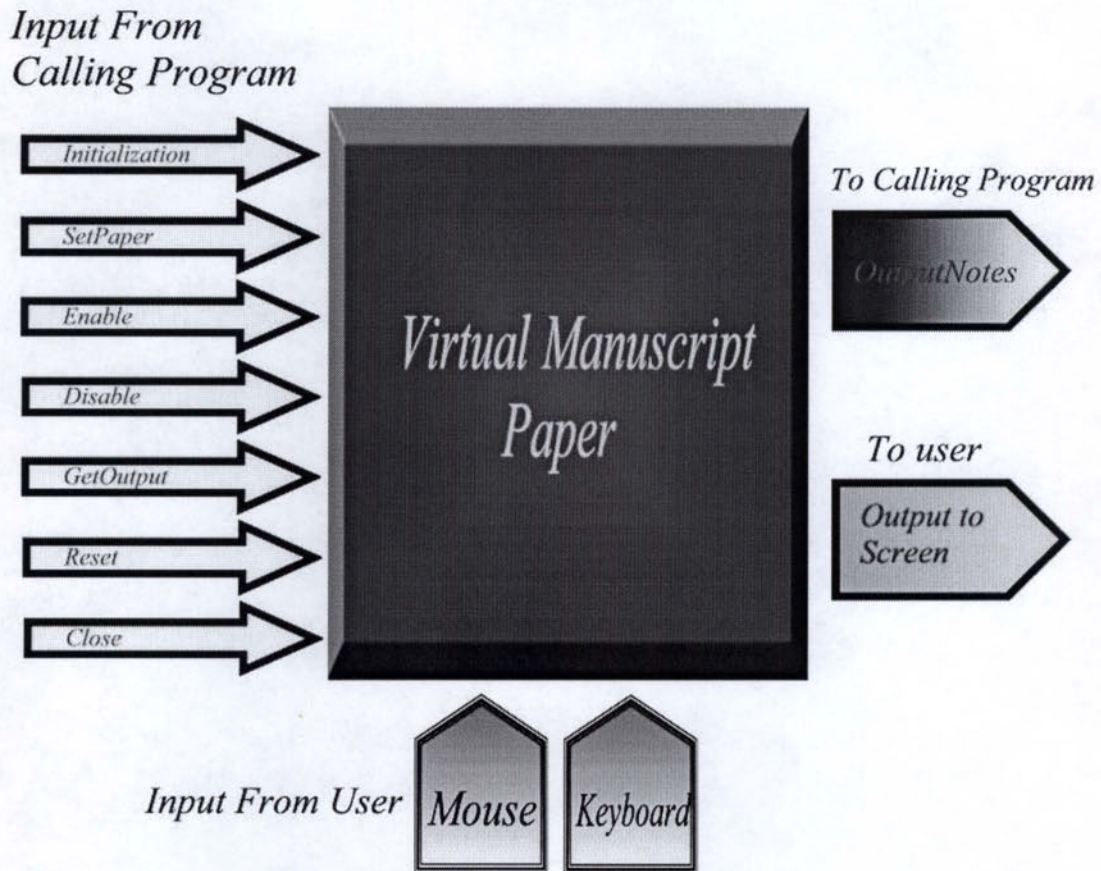
1. Module to interact with various programs to get notational input from students for various musical exercise by mouse and keyboard.
2. Use single staff or Grand Staff.
3. Can use Treble, Bass, Alto, Tenor Clefs.
4. Meter can be 2/4, 4/4, 3/4, 6/8, 9/8, or 12/8.
5. Can enter in notes from 2 ledger lines below the staff to 2 lines above.
6. Key Signature can be up to 6 flats or 6 sharps.
7. User can enter notes and rest values: Whole, half, quarter, 1/8, and 1/16 notes.
8. When entering notes, you can re-enter that note value without having to select the note value button again.
9. User can triple three quarter notes or three eighth notes.
10. User can add one dot, a sharp, a flat, a natural, a double sharp, or a double flat to a note and one dot to a rest.
11. User enters/erases the beaming on the notes (but the computer formats the beaming and the notes so that the music is readable) and can tie (but not slur) notes.
12. Finale interface to select note values from keyboard. (and up and down arrows raise and lower pitch respectively.) Use + or – to change accidental. Use 9 key to toggle enharmonics (for example – F# to Gb).
13. Input can converted to MIDI format so that the answer can be played by the user for comparison if answer is wrong.
14. The note-formatting/output module will able to also be used in a “conductor” type program where the user hears a melody and addresses how it deviates from the written notes (or some variant of this type of program) and in a rhythmic dictation where the pitches are given and the user just adds the appropriate time values to the notes.
15. Up to 8 measures will be used.
16. No double bar at end of music.

17. Will allow user to enter in more notes than in one beat and/or measure without warning. Will allow the user to enter 4 notes into one simple time beat (or six in one complex time beat).

## **Derived Requirements:**

- D1. Output will be received after the input from the mouse and keyboard has been disabled for the VMP.
- D2. Information can only be received from one source at a time. The program cannot set the VMP while keyboard and mouse input are enabled and vice-versa.
- D3. A None edit mode will allow displaying of errors and a flag attached to each note allows the specification of pitch and rhythm errors.
- D4. The input and output modules start out disabled from user input.
- D5. All I/O modules must be disabled before they can be set.

## System Boundary:



### Inputs:

- **From Calling Program:**
  - Initialize – Occurs when the calling program calls the VMP stack.
  - Set Paper – When the program sets the initial data to be displayed on VMP display.
  - Enable – Enables keyboard and mouse input to be processed by the VMP.
  - Disable – Disables input from keyboard and mouse to the VMP.

- GetOutput – Sent when the calling program wants to retrieve the output from the VMP.
  - Reset – Sets all the parameters of VMP to their initial values.
  - Close – Closes and exits VMP.
- **From User:**
    - Mouse – Interaction between VMP objects and mouse pointer.
    - Keyboard - Interaction between VMP and keyboard.

## **Outputs:**

- Gives encoded note information to calling program
- Displays formatted note output to user.

## **Stimulus Definitions:**

| <b>Stimulus</b> | <b>Description</b>                                                                           | <b>Requirements Trace</b> |
|-----------------|----------------------------------------------------------------------------------------------|---------------------------|
| INIT            | Initialization of VMP                                                                        | 1                         |
| SET             | The calling program setting the data for the VMP                                             | 1                         |
| EN              | The calling program sending signal to enable the VMP to accept input from keyboard and mouse | 1                         |
| USER            | Mouse or Keyboard input received by VMP                                                      | D1, 13                    |
| DIS             | The calling program sending a disable signal to the VMP                                      | 1                         |
| OUTPUT          | The calling program sending a signal to make the VMP dump its data to program                | 1                         |
| RESET           | Sets all the parameters of VMP to their initial values                                       | 1                         |
| CLOSE           | Closing the VMP                                                                              | 1                         |

## **Response Definitions:**

| <b>Response</b> | <b>Description</b>                                                            | <b>Requirements Trace</b> |
|-----------------|-------------------------------------------------------------------------------|---------------------------|
| START           | Do instantiation process for VMP                                              |                           |
| SETUP_VMP       | Setup VMP according to input                                                  |                           |
| ACCEPT          | The VMP module allowing user input.                                           |                           |
| PROC_INPUT      | Process mouse input received by VMP and update display                        |                           |
| BLOCK           | The VMP disallowing user interaction.                                         |                           |
| DUMP            | The calling program sending a signal to make the VMP dump its data to program |                           |
| CLEAR           | Sets all parameters to their initial values                                   |                           |
| END_VMP         | Closing the VMP                                                               |                           |
| ERROR           | Reporting misuse of the module.                                               |                           |

## Sequence Enumeration:

### Length Zero

| Sequence | Response | Equivalence | Req. Trace No. |
|----------|----------|-------------|----------------|
| —        | 0        |             |                |

### Length One

| Sequence | Response   | Equivalence | Req. Trace No. |
|----------|------------|-------------|----------------|
| INIT     | START      |             |                |
| SET      | $\omega^1$ |             |                |
| EN       | $\omega$   |             |                |
| USER     | $\omega$   |             |                |
| DIS      | $\omega$   |             |                |
| OUTPUT   | $\omega$   |             |                |
| RESET    | $\omega$   |             |                |
| CLOSE    | $\omega$   |             |                |

### Length Two

| Sequence    | Response  | Equivalence | Req. Trace No. |
|-------------|-----------|-------------|----------------|
| INIT INIT   | START     | INIT        |                |
| INIT SET    | SETUP_VMP | INIT        |                |
| INIT EN     | ACCEPT    |             |                |
| INIT USER   | 0         | INIT        |                |
| INIT DIS    | ERROR     | INIT        |                |
| INIT OUTPUT | DUMP      | INIT        |                |
| INIT RESET  | CLEAR     | INIT        |                |
| INIT CLOSE  | END_VMP   | —           |                |

---

<sup>1</sup>  $\omega$  = illegal/impossible value

**Length Three**

| Sequence       | Response   | Equivalence | Req. Trace No. |
|----------------|------------|-------------|----------------|
| INIT EN INIT   | START      | INIT        |                |
| INIT EN SET    | ERROR      | INIT EN     |                |
| INIT EN EN     | ERROR      | INIT EN     |                |
| INIT EN USER   | PROC_INPUT | INIT EN     |                |
| INIT EN DIS    | BLOCK      | INIT        |                |
| INIT EN OUTPUT | ERROR      | INIT EN     |                |
| INIT EN RESET  | ERROR      | INIT EN     |                |
| INIT EN CLOSE  | END_VMP    | —           |                |

## Canonical Sequence Analysis:

### Finding Orthogonal State Variables:

| Canonical Sequence | Initialized | InputMode |
|--------------------|-------------|-----------|
| —                  | False       | None      |
| INIT               | True        | Program   |
| INIT EN            | True        | User      |

### State Variables:

| Canonical Sequence | Range                 | Initial Value |
|--------------------|-----------------------|---------------|
| Initialized        | {True, False}         | False         |
| InputMode          | {Program, User, None} | Program       |

### Canonical Sequence Detail:

| Canonical Sequence                                      | State Variables          | Value Before Current Stimulus | Value After Current Stimulus |
|---------------------------------------------------------|--------------------------|-------------------------------|------------------------------|
| —                                                       | --                       | --                            | --                           |
| INIT<br>The program has instantiated VMP                | Initialized              | False                         | True                         |
| INIT EN<br>The program has enabled input from the user. | Initialized<br>InputMode | True<br>Program               | True<br>User                 |

## **Specification Functions:**

### **Specification Function: Initialized(h)**

| Stimulus | Prefix Condition   | Value          |
|----------|--------------------|----------------|
| —        | h = empty          | False          |
| INIT     | Any                | True           |
| s        | Otherwise (h = ps) | Initialized(p) |

### **Specification Function: InputMode(h)**

| Stimulus | Prefix Condition       | Value        |
|----------|------------------------|--------------|
| —        | h = empty              | Program      |
| EN       | InputMode(p) = Program | User         |
| s        | Otherwise (h = ps)     | InputMode(p) |

## **Black Box Tables:**

### **Current Stimulus: INIT**

| Prefix Condition | Response | Trace |
|------------------|----------|-------|
| Any              | START    |       |

### **Current Stimulus: SET**

| Prefix Condition                                | Response  | Trace |
|-------------------------------------------------|-----------|-------|
| Initialized(p) = True<br>InputMode(p) = Program | SETUP_VMP |       |
| Initialized(p) = True<br>InputMode(p) = User    | ERROR     |       |

### **Current Stimulus: EN**

| Prefix Condition                                | Response | Trace |
|-------------------------------------------------|----------|-------|
| Initialized(p) = True<br>InputMode(p) = Program | ACCEPT   |       |
| Initialized(p) = True<br>InputMode(p) = User    | ERROR    |       |

### **Current Stimulus: USER**

| Prefix Condition                                | Response   | Trace |
|-------------------------------------------------|------------|-------|
| Initialized(p) = True<br>InputMode(p) = Program | PROC_INPUT |       |
| Initialized(p) = True<br>InputMode(p) = User    | ERROR      |       |

**Current Stimulus: DIS**

| Prefix Condition                                | Response | Trace |
|-------------------------------------------------|----------|-------|
| Initialized(p) = True<br>InputMode(p) = Program | ERROR    |       |
| Initialized(p) = True<br>InputMode(p) = User    | BLOCK    |       |

**Current Stimulus: OUTPUT**

| Prefix Condition                                | Response | Trace |
|-------------------------------------------------|----------|-------|
| Initialized(p) = True<br>InputMode(p) = Program | DUMP     |       |
| Initialized(p) = True<br>InputMode(p) = User    | ERROR    |       |

**Current Stimulus: RESET**

| Prefix Condition                                | Response | Trace |
|-------------------------------------------------|----------|-------|
| Initialized(p) = True<br>InputMode(p) = Program | CLEAR    |       |
| Initialized(p) = True<br>InputMode(p) = User    | ERROR    |       |

**Current Stimulus: CLOSE**

| Prefix Condition | Response | Trace |
|------------------|----------|-------|
| Any              | END_VMP  |       |

## State Box Table:

### Current Stimulus: INIT

| Current State | Response | New State                                   |
|---------------|----------|---------------------------------------------|
| Any           | START    | Initialized := True<br>InputMode := Program |

### Current Stimulus: SET

| Current State        | Response  | New State |
|----------------------|-----------|-----------|
| InputMode := Program | SETUP_VMP | No change |
| InputMode := User    | ERROR     | No change |

### Current Stimulus: EN

| Current State        | Response | New State         |
|----------------------|----------|-------------------|
| InputMode := Program | ACCEPT   | InputMode := User |
| InputMode := User    | ERROR    | No change         |

### Current Stimulus: USER

| Current State        | Response   | New State |
|----------------------|------------|-----------|
| InputMode := Program | Null       | No change |
| InputMode := User    | PROC_INPUT | None      |

### Current Stimulus: DIS

| Current State        | Response | New State            |
|----------------------|----------|----------------------|
| InputMode := Program | ERROR    | None                 |
| InputMode := User    | BLOCK    | InputMode := Program |

**Current Stimulus: OUTPUT**

| Current State        | Response | New State |
|----------------------|----------|-----------|
| InputMode := Program | DUMP     | No change |
| InputMode := User    | ERROR    | No change |

**Current Stimulus: RESET**

| Current State        | Response | New State |
|----------------------|----------|-----------|
| InputMode := Program | CLEAR    | No change |
| InputMode := User    | ERROR    | No change |

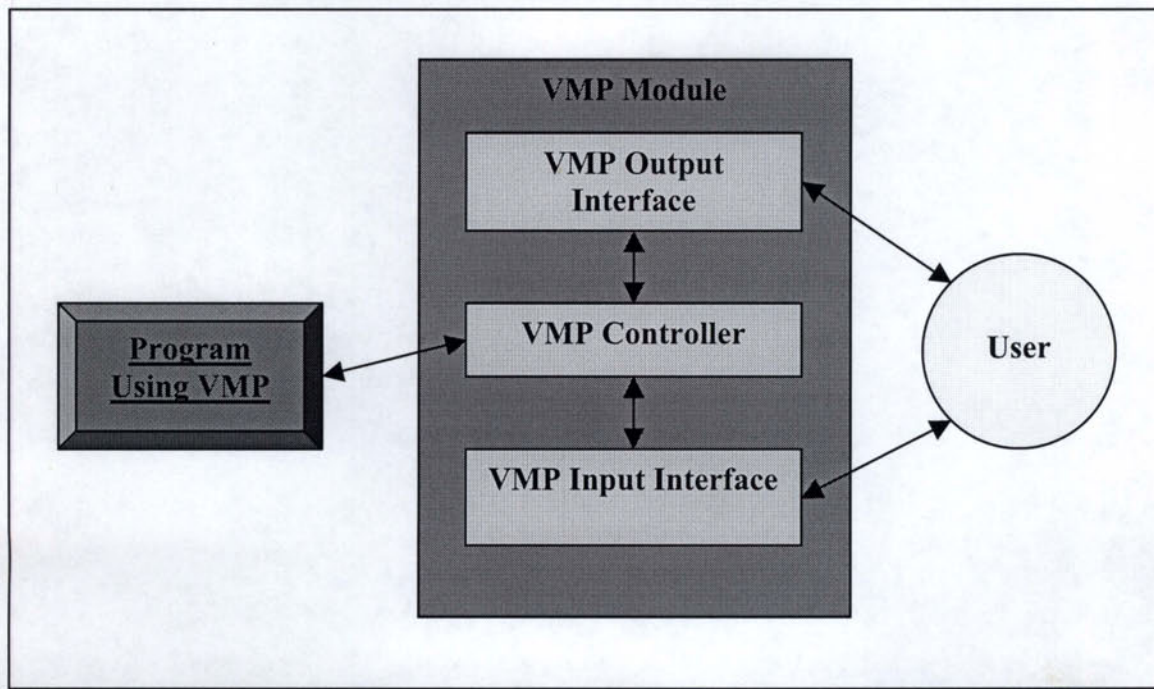
**Current Stimulus: CLOSE**

| Current State       | Response | New State            |
|---------------------|----------|----------------------|
| Initialized := True | END_VMP  | Initialized := False |

## Clear Box Design:

The preceding describes the behavior of this program as it is seen by the system boundary, which includes the programmer and the user. To implement this design, this module will be broken up into three major components: a control module that will interact with the program which utilized this module, an input interface, and a separate output interface. The input and output interface are separated due to the information gathered in the requirements interface. The topology of the program is illustrated in the following diagram.

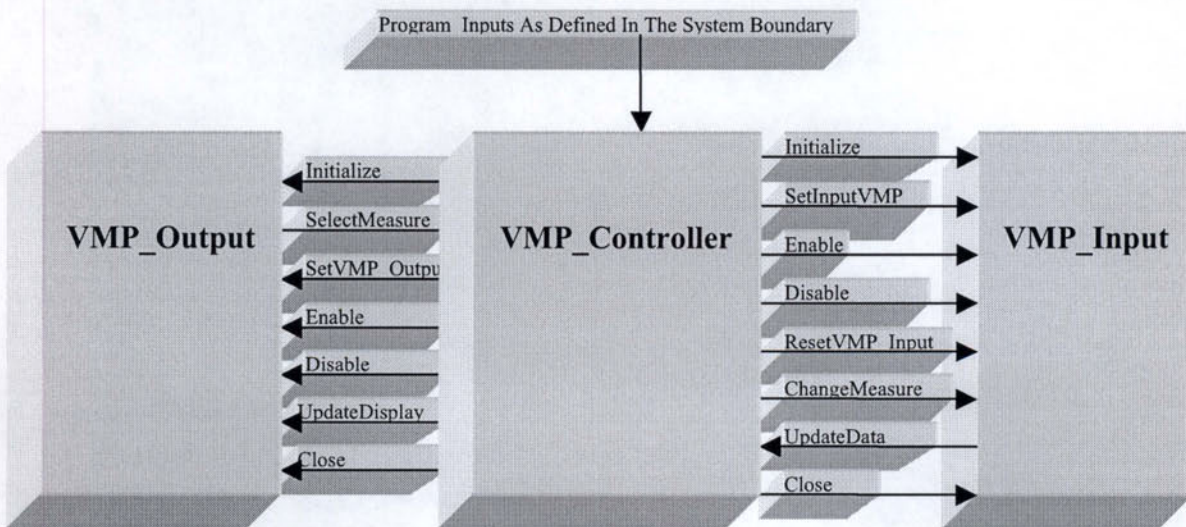
### VMP Component Diagram



The information is pushed into the module and the user interacts with the I/O modules via monitor, keyboard, and mouse, after which it will be pulled back to get the results. For implementation purposes, each component will be implemented as a main stack. The programmer will only interact with the VMP controller and the User will interact with the two on-screen interfaces. The VMP

controller will be known to the programmer as the VMP and will have the same behaviors defined by the previous sequence enumeration, except that it will also have internal inputs and outputs to the output interface and the input interface.

The overall internal behavior will be such that the main data will be stored in the VMP controller and it will be pulled by the other two components on an as needed basis. The VMP controller will also take care of the internal signals that will be sent in between the input interface and the output interface. An example GUI has been created to give a visual representation of the ways the programmer can manipulate the VMP. The interfaces will be independent from each other so that, if in the future, a different I/O interface is needed, one can be easily added without restructuring the whole system. The internal components diagram, with all program signals, is shown in the following diagram.



## **UI Design:**

The UI will be designed such that the interfaces for the input and output are separate. Due to the amount of space this will need on the screen, the controlling program's screen display will have to occupy a slot on the side. There are several main features in the design. First, since this is a program that will be used by the user for extended periods of time, much attention will be given to using dimmer colors and aesthetic layouts that will not be tiresome to the eyes. Also, the input module will facilitate easy entering of notes so that the user will not have to strain to place the note in the desired position. Thirdly, the output interface will try to implement an algorithm that closely resembles how music is commonly notated. There is to be a scrollbar, however, it will only have to be used in cases in which the musical example has a very large number of notes.

In respect to the program, the UI layout will be implemented so that every object's size is relative to the screen resolution. If this causes the appearance to be unsatisfactory, the screen resolutions can be hard coded in the variable constants defined in each main stack. The reason the screen is to be set relative to screen size is in hopes that it can be used with similar appearance (without enlarging or shrinking) across several screen resolutions. The images will be optimized for screen resolutions with 4/3 aspect ratios, with a resolution of 1024x768 in particular. If the appearance is unsatisfactory on other resolutions, the program will be able to be updated fairly easily by designing the images for the desired resolution and perhaps editing several segments of code.

## Coding Style:

Throughout the program the coding style will be consistent to make the coding style readable and maintainable. The program will use a pseudo-Hungarian notation that is defined as follows.

| Notation                     | Meaning                       |
|------------------------------|-------------------------------|
| <b>G_</b> variableName       | Global variable               |
| <b>SL_</b> variableName      | Script Local Variable         |
| <b>STATE_</b> variableName   | State Variable (Script Local) |
| <b>L_</b> variableName       | Local Variable                |
| <b>P_</b> variableName       | Parameter Variable            |
| <b>STACK_</b> stackName      | Stack ID                      |
| <b>CARD_</b> cardName        | Card ID                       |
| <b>MSG_</b> handlerName      | Handler ID                    |
| <b>GRP_</b> groupName        | Group ID                      |
| <b>IMG_</b> imageName        | Image ID                      |
| <b>PBTN_</b> pushButtonName  | Push Button ID                |
| <b>RBTN_</b> radioButtonName | Radio Button ID               |

### Commenting:

The sections of code will be commented according to the following scheme.

#### ----- **Setting Constants**

This comment marks the large sections of a script, i.e the script local definition section, handler section, etc.

#### **##### SETTING PROPERTIES #####**

This comment marks the sections within the script, such as variable initialization, main function body, etc.

## **-- Setting display properties**

This comment explains each step of the process within a script

## **--NOTE: xxxxxxxx**

This comment marks a special note that explains special concerns and reasoning behind implementation decisions that are not readily apparent.

## **Coding Approach:**

In all instances, the coding will try to group similar operations, variables, procedures, etc. within the code as to make understanding easier. Also, most of the code will try to be contained in the highest level as possible and will only allow duplication in areas in which it will make maintenance easier and when it will make for better performance. Also, the preOpenStack message will be treated as a constructor and the closeStack message will be treated as a destructor. Most initializations and setup will be done here as to try to let the other handlers focus on specific functions.

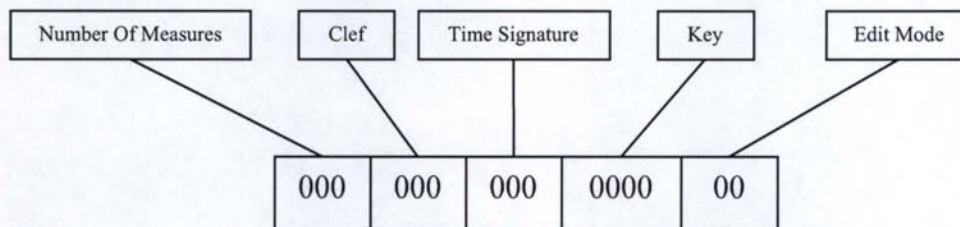
Most of the specification of the appearance will be done using scripts as opposed to using the GUI. There are several reasons for this. First, by using scripts, this allows better documentation of how the screen is set up. It also allows flexibility between resolutions. Secondly, during the prototyping phase, it was found that the GUI was very buggy and had much unpredictable behavior. Thirdly, this will allow the appearance to be more easily changed when adapting the program to different resolutions.

## Input Format:

The information will be set in the VMP through the SetPaper function by calling the following handler with the parameters as follows.

### **Set Paper PaperProperties, NotationArray**

Where PaperProperties is an integer that is set by the bits as follows.



**1. Number of Measures** – This is the binary representation of the number of measures to be edited in the paper.

**2. Clef** – This specifies which clef is to be used by the example.

| Binary Representation | Meaning                       |
|-----------------------|-------------------------------|
| 000                   | Single Voice Soprano Clef     |
| 001                   | Single Voice Bass Clef        |
| 010                   | Single Voice Alto Clef        |
| 011                   | Single Voice Tenor Clef       |
| 100                   | Grand Staff (Bass and Soprano |

|            |                          |
|------------|--------------------------|
|            | Voices)                  |
| 101 to 111 | Available for future use |

**3. Time Signature** – This specifies the time signature used for all voices.

| Binary Representation | Meaning  |
|-----------------------|----------|
| 000                   | 4/4      |
| 001                   | 3/4      |
| 010                   | 2/4      |
| 011                   | 3/8      |
| 100                   | 6/8      |
| 101                   | 9/8      |
| 110                   | 12/8     |
| 111                   | Not used |

**4. Key** – This defines the Key signature that all voices will have.

| Binary Representation | Meaning            |
|-----------------------|--------------------|
| 0000                  | 0 sharps / 0 flats |
| 0001                  | 1 sharp            |
| 0010                  | 2 sharps           |
| 0011                  | 3 sharps           |
| 0100                  | 4 sharps           |
| 0101                  | 5 sharps           |
| 0110                  | 6 sharps           |
| 0111                  | 6 flats            |
| 1000                  | 5 flats            |
| 1001                  | 4 flats            |

|      |         |
|------|---------|
| 1010 | 3 flats |
| 1011 | 2 flats |
| 1100 | 1 flat  |
| 1101 | unused  |

**5. Edit Mode** – These bits define how the user can edit the notes.

| Binary Representation | Meaning                                   |
|-----------------------|-------------------------------------------|
| 00                    | Allows editing for pitch and rhythm       |
| 01                    | Allows editing for pitch only             |
| 10                    | Allows editing for rhythm only            |
| 11                    | Allows no editing, notes for display only |

And NotationArray is an array indexed as follow.

**NotationArray[Voice, Measure, Beat, Element]**

Where the indexes are,

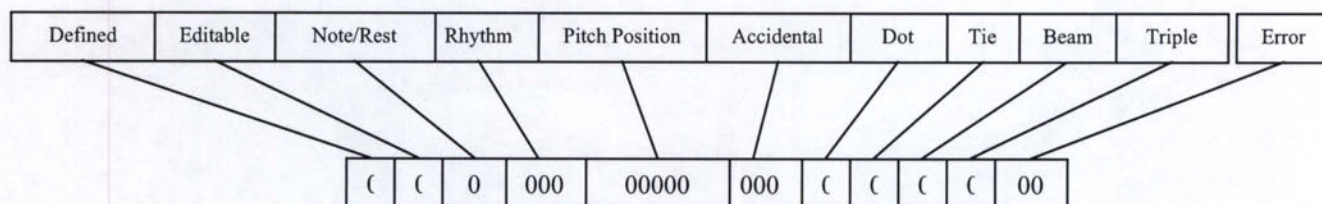
**Voice** – The number of the voice (used when notating multiple voices, such as in Soprano Bass)

**Measure** – The measure number being indexed.

**Beat** – The beat being accessed.

**Element** – The number of the element within the beat that is being accessed.

With each note defined bitwise as shown in the following diagram.



**1. Defined** – This bit defines whether or not the note has been defined, where 0 is no and 1 is yes.

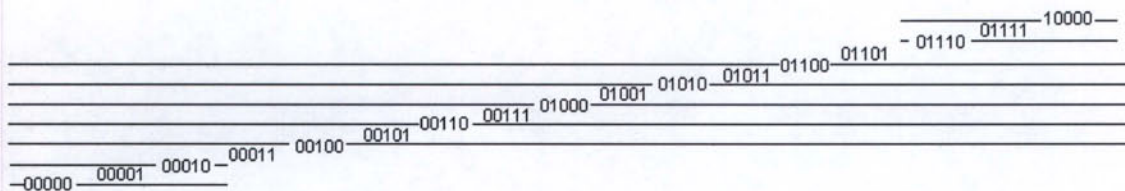
**2. Editable** – This specifies whether or not the user can edit this note using the specified edit mode with 0 being no and 1 being yes.

**3. Note/Rest** – This denotes whether or not the note is a rest (1) or a note (0).

**4. Rhythm** – This denotes the rhythm value the element is given.

| Binary Representation | Meaning                  |
|-----------------------|--------------------------|
| 000                   | Whole                    |
| 001                   | Half                     |
| 010                   | Quarter                  |
| 011                   | Eighth                   |
| 100                   | Sixteenth                |
| 101 to 111            | Available for future use |

**5. Pitch Position** – This defines the vertical position on the staff that a note will be. They represent the position only, not the pitch which is dependent upon the clef. The diagram shows how the positions are represented.



**6. Accidental** – Defines what accidental, if any, a note has.

| Binary Representation | Meaning      |
|-----------------------|--------------|
| 000                   | None         |
| 001                   | Double Flat  |
| 010                   | Flat         |
| 011                   | Natural      |
| 100                   | Sharp        |
| 101                   | Double Sharp |
| 110 to 111            | Not Used     |

**7. Dot** – Defines the absence (0) or presence (1) of a dot on a note or rest.

**8. Tie** – Defines whether or not a note is tied to the following note, 1 if yes and 0 if no.

**9. Beam** – Defines whether or not the note is beamed to the following note, 1 if yes and 0 if no.

**10. Triple** – Defines whether or not a note is part of a tripled group, 1 if yes and 0 if no.

**11. Error** – Used when errors are fed into the VMP to display to the user.

| Binary Representation | Meaning |
|-----------------------|---------|
|-----------------------|---------|

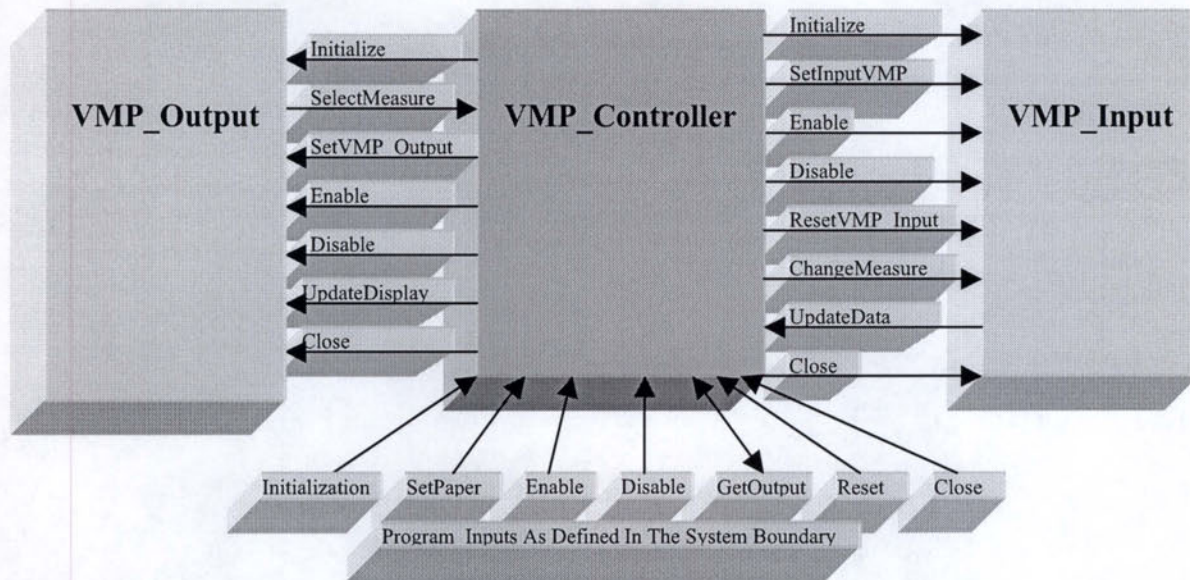
|    |                        |
|----|------------------------|
| 00 | No Error               |
| 01 | Pitch Error            |
| 10 | Rhythm Error           |
| 11 | Pitch and Rhythm Error |

# Component Design: VMP\_Controller

## **Introduction:**

This section of the documentation deals with the design of the VMP\_Controller component of the VMP module. This component's responsibilities are to interact with calling program and to drive the input and output modules. Also, it will be the location where the notation data is stored. The data will be updated from the input by pulling the data from the input module. This was done to keep the data centralized in one area and prevent possible the possible problems of having the data acted directly upon by the input module. It will be a main stack, but it will not be visible on the screen. All of the operations will be contained in the script. The requirements referenced are the same as those for the system boundary for the VMP. The behavior of this module, and therefore the analysis, will be similar to the overall behavior of the VMP as described in the preceding analysis. However, the controller has several extra inputs and outputs which are internal to the VMP, and it excludes some of the VMP's inputs and outputs, so they will not be exactly alike.

## System Boundary:



## Inputs:

- **From Calling Program:**

- Initialize – Occurs when the calling program instantiates the VMP module.
- Set Paper – When the program sends the data to initialize the display of the VMP.
- Enable – Enables keyboard and mouse input to be processed by the VMP module.
- Disable – Disables input from keyboard and mouse to the VMP module.
- GetOutput – Sent when the calling program wants to retrieve the output from the VMP.
- Reset – Sets all the parameters of VMP to their initial values.
- Close – Closes and exits VMP.

- **From VMP\_Input Component:**
  - UpdateData – Sent to VMP\_Controller to cause the notation array to be updated according to the information entered into VMP\_Input.
- **From VMP\_Output Component:**
  - SelectMeasure – Sent to VMP\_Controller when the user clicks on the measure he/she wants to edit in the output module.

## **Outputs:**

- GetOutput - Gives encoded note information to calling program.
- SetVMP Output – This sends a signal for the output module to update its display.
- Enable/Disable – Sent to both the input and output modules to enable user interaction with each.
- ResetVMP Input – Causes Input module to clear all of its contents and return to its initialized state.
- Change Measure – Message sent to VMP\_Input to cause it to change the current measure being edited.
- UpdateDisplay – Sent to cause the output module to update the display after new input is entered.

## **Stimulus Definitions:**

| <b>Stimulus</b> | <b>Description</b>                                                                                                                      | <b>Requirements Trace</b> |
|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------|---------------------------|
| INIT            | Initialization of VMP_Controller                                                                                                        | 1                         |
| SET             | The calling program setting the data for the VMP_Controller                                                                             | 1                         |
| EN              | The calling program sending signal to enable the input and output interfaces to begin interacting with user.                            | 1                         |
| DIS             | The calling program sending a signal to cease the input and output interfaces from interacting with the user.                           | 1                         |
| OUTPUT          | The calling program sending a signal to make the VMP_controller dump its data to program.                                               | 1                         |
| RESET           | Sent to cause the whole VMP module to reset its data.                                                                                   | 1                         |
| CLOSE           | Closing the VMP                                                                                                                         | 1                         |
| UPDATE          | Causes the VMP_Controller to update the notation array with the information retrieved from the user that is stored in the input module. |                           |
| SELM            | Sent from output module to cause the VMP_Controller to relay to the input module which measure to edit.                                 |                           |

## **Response Definitions:**

| <b>Response</b> | <b>Description</b>                                                                                                 | <b>Requirements Trace</b> |
|-----------------|--------------------------------------------------------------------------------------------------------------------|---------------------------|
| START           | Do instantiation process for VMP module                                                                            |                           |
| SETUP_VMP       | Setup VMP according to input.<br>VMP_Controller sets up internally and signals the other components to do the same |                           |
| ACCEPT          | The VMP allowing interaction with the user by sending enable signals to other components.                          |                           |
| BLOCK           | The VMP disallowing interaction with the user by sending a disable signal to other components.                     |                           |
| DUMP            | The calling program sending a signal to make the VMP dump its data to program.                                     |                           |
| CLEAR           | Setting all parameters to their initial values and causing the other components to do likewise.                    |                           |
| END_VMP         | Closing the VMP and doing garbage collection.                                                                      |                           |
| ERROR           | Reporting misuse of the module.                                                                                    |                           |
| UPD_VMP         | Updating the notation array and signaling VMP_Output to update by sending the message UpdateDisplay.               |                           |
| CHGM            | Send message to VMP_Input to change the measure currently being edited.                                            |                           |

## Sequence Enumeration:

**Note:** This enumeration is, in essence, a modified version of the sequence enumeration of the whole system. Therefore, the sequence enumeration of VMP\_Controller will begin from the canonical sequences of the whole system's enumeration.

### Length Zero

| Sequence | Response | Equivalence | Req. Trace No. |
|----------|----------|-------------|----------------|
| —        | 0        |             |                |

### Length One

| Sequence | Response | Equivalence | Req. Trace No. |
|----------|----------|-------------|----------------|
| INIT     | START    |             |                |
| SET      | $\omega$ |             |                |
| EN       | $\omega$ |             |                |
| DIS      | $\omega$ |             |                |
| OUTPUT   | $\omega$ |             |                |
| RESET    | $\omega$ |             |                |
| CLOSE    | $\omega$ |             |                |
| UPDATE   | $\omega$ |             |                |
| SELM     | $\omega$ |             |                |

### Length Two

| Sequence    | Response  | Equivalence | Req. Trace No. |
|-------------|-----------|-------------|----------------|
| INIT INIT   | START     | INIT        |                |
| INIT SET    | SETUP_VMP | INIT        |                |
| INIT EN     | ACCEPT    |             |                |
| INIT DIS    | ERROR     | INIT        |                |
| INIT OUTPUT | DUMP      | INIT        |                |
| INIT RESET  | CLEAR     | INIT        |                |

|             |         |   |  |
|-------------|---------|---|--|
| INIT CLOSE  | END_VMP | — |  |
| INIT UPDATE | ω       |   |  |
| INIT SELM   | ω       |   |  |

### Length Three

| Sequence       | Response | Equivalence | Req. Trace No. |
|----------------|----------|-------------|----------------|
| INIT EN INIT   | START    | INIT        |                |
| INIT EN SET    | ERROR    | INIT EN     |                |
| INIT EN EN     | ERROR    | INIT EN     |                |
| INIT EN DIS    | BLOCK    | INIT        |                |
| INIT EN OUTPUT | ERROR    | INIT EN     |                |
| INIT EN RESET  | ERROR    | INIT EN     |                |
| INIT EN CLOSE  | END_VMP  | —           |                |
| INIT EN UPDATE | UPD_VMP  | INIT EN     |                |
| INIT EN SELM   | CHGM     | INIT EN     |                |

## Canonical Sequence Analysis:

### Finding Orthogonal State Variables:

| Canonical Sequence | Initialized | InputMode |
|--------------------|-------------|-----------|
| —                  | False       | None      |
| INIT               | True        | Program   |
| INIT EN            | True        | User      |

### State Variables:

| Canonical Sequence | Range                 | Initial Value |
|--------------------|-----------------------|---------------|
| Initialized        | {True, False}         | False         |
| InputMode          | {Program, User, None} | Program       |

### Canonical Sequence Detail:

| Canonical Sequence                                      | State Variables          | Value Before Current Stimulus | Value After Current Stimulus |
|---------------------------------------------------------|--------------------------|-------------------------------|------------------------------|
| —                                                       | --                       | --                            | --                           |
| INIT<br>The program has instantiated VMP                | Initialized              | False                         | True                         |
| INIT EN<br>The program has enabled input from the user. | Initialized<br>InputMode | True<br>Program               | True<br>User                 |

## **Specification Functions:**

### **Specification Function: Initialized(h)**

| Stimulus | Prefix Condition   | Value          |
|----------|--------------------|----------------|
| —        | h = empty          | False          |
| INIT     | Any                | True           |
| s        | Otherwise (h = ps) | Initialized(p) |

### **Specification Function: InputMode(h)**

| Stimulus | Prefix Condition       | Value        |
|----------|------------------------|--------------|
| —        | h = empty              | Program      |
| EN       | InputMode(p) = Program | User         |
| s        | Otherwise (h = ps)     | InputMode(p) |

## **Black Box Tables:**

### **Current Stimulus: INIT**

| Prefix Condition | Response | Trace |
|------------------|----------|-------|
| Any              | START    |       |

### **Current Stimulus: SET**

| Prefix Condition                                | Response  | Trace |
|-------------------------------------------------|-----------|-------|
| Initialized(p) = True<br>InputMode(p) = Program | SETUP_VMP |       |
| Initialized(p) = True<br>InputMode(p) = User    | ERROR     |       |

### **Current Stimulus: EN**

| Prefix Condition                                | Response | Trace |
|-------------------------------------------------|----------|-------|
| Initialized(p) = True<br>InputMode(p) = Program | ACCEPT   |       |
| Initialized(p) = True<br>InputMode(p) = User    | ERROR    |       |

### **Current Stimulus: DIS**

| Prefix Condition                                | Response | Trace |
|-------------------------------------------------|----------|-------|
| Initialized(p) = True<br>InputMode(p) = Program | ERROR    |       |
| Initialized(p) = True<br>InputMode(p) = User    | BLOCK    |       |

**Current Stimulus: OUTPUT**

| Prefix Condition                                | Response | Trace |
|-------------------------------------------------|----------|-------|
| Initialized(p) = True<br>InputMode(p) = Program | DUMP     |       |
| Initialized(p) = True<br>InputMode(p) = User    | ERROR    |       |

**Current Stimulus: RESET**

| Prefix Condition                                | Response | Trace |
|-------------------------------------------------|----------|-------|
| Initialized(p) = True<br>InputMode(p) = Program | CLEAR    |       |
| Initialized(p) = True<br>InputMode(p) = User    | ERROR    |       |

**Current Stimulus: CLOSE**

| Prefix Condition | Response | Trace |
|------------------|----------|-------|
| Any              | END_VMP  |       |

**Current Stimulus: UPDATE**

| Prefix Condition                                | Response | Trace |
|-------------------------------------------------|----------|-------|
| Initialized(p) = True<br>InputMode(p) = Program | ERROR    |       |
| Initialized(p) = True<br>InputMode(p) = User    | UPD_VMP  |       |

**Current Stimulus: SELM**

| Prefix Condition                                | Response | Trace |
|-------------------------------------------------|----------|-------|
| Initialized(p) = True<br>InputMode(p) = Program | ERROR    |       |
| Initialized(p) = True<br>InputMode(p) = User    | CHGM     |       |

## **State Box Table:**

### **Current Stimulus: INIT**

| Current State | Response | New State                                   |
|---------------|----------|---------------------------------------------|
| Any           | START    | Initialized := True<br>InputMode := Program |

### **Current Stimulus: SET**

| Current State        | Response  | New State |
|----------------------|-----------|-----------|
| InputMode := Program | SETUP_VMP | No change |
| InputMode := User    | ERROR     | No change |

### **Current Stimulus: EN**

| Current State        | Response | New State         |
|----------------------|----------|-------------------|
| InputMode := Program | ACCEPT   | InputMode := User |
| InputMode := User    | ERROR    | No change         |

### **Current Stimulus: DIS**

| Current State        | Response | New State            |
|----------------------|----------|----------------------|
| InputMode := Program | ERROR    | No change            |
| InputMode := User    | BLOCK    | InputMode := Program |

### **Current Stimulus: OUTPUT**

| Current State        | Response | New State |
|----------------------|----------|-----------|
| InputMode := Program | DUMP     | No change |
| InputMode := User    | ERROR    | No change |

**Current Stimulus: RESET**

| Current State        | Response | New State |
|----------------------|----------|-----------|
| InputMode := Program | CLEAR    | No change |
| InputMode := User    | ERROR    | No change |

**Current Stimulus: CLOSE**

| Current State       | Response | New State            |
|---------------------|----------|----------------------|
| Initialized := True | END_VMP  | Initialized := False |

**Current Stimulus: UPDATE**

| Current State        | Response | New State |
|----------------------|----------|-----------|
| InputMode := Program | ERROR    | No change |
| InputMode := User    | UPD_VMP  | No change |

**Current Stimulus: SELM**

| Current State        | Response | New State |
|----------------------|----------|-----------|
| InputMode := Program | ERROR    | No change |
| InputMode := User    | CHGM     | No change |

## **Clear Box Design:**

The clear box design for this module is straight forward. Each input is handled by a dedicated message handler with the names as specified in the VMP\_Controller's system boundary. As stated before, the controller is a stack that is not visible on the screen. The data is stored in a value that contains the page properties and in an array which contains the notation information (as described on pages 22-26). The state box tables will be transferred as is to branch statements within the respective handler.

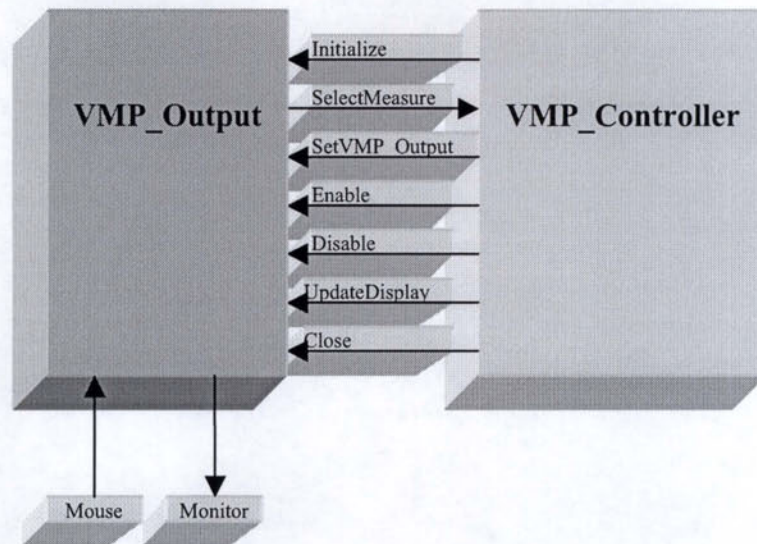
# Component Design:

## VMP\_Output

## **Introduction:**

This section of the documentation deals with the design and implementation of the output interface for the VMP, VMP\_Output. This component is basically a graphic representation of the data stored in VMP\_Controller. This part will contain the sequence analysis and it will contain a description of the algorithm used to format the notes on the screen. The difficulties of this component have to do with the problem of coming up with an algorithm that will format the notes on the screen in a pleasing matter, yet not having the algorithm be too complex.

## System Boundary:



### Inputs:

- **From VMP\_Controller:**
  - *Initialize* – Sent to component on instantiation.
  - *SetVMPOutput* – This is sent to initialize the output module according to initial data sent to VMP\_Controller.
  - *Enable* – Sent to component to allow input from the user.
  - *Disable* – Sent to component to disallow input from the user.
  - *UpdateDisplay* – Sent when user inputs notation information and the output interface is to be updated.
  - *Close* – Sent when the VMP module is closed.
- **From User:**
  - *ClickMeasure* – Measure clicked that user wants to edit.

## **Outputs:**

- SelectMeasure – Output sent to change the measure being edited by VMP\_Input.
- Displays formatted note output to user.

## **Stimulus Definitions:**

| <b>Stimulus</b> | <b>Description</b>                                                                                                                      | <b>Requirements Trace</b> |
|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------|---------------------------|
| INIT            | Initialization of VMP_Output                                                                                                            |                           |
| SETOUT          | Sent to cause VMP_Output to setup according to data in VMP_Controller.                                                                  |                           |
| EN              | The calling program sending signal to enable the input and output interfaces to begin interacting with user.                            |                           |
| UPDATE          | Causes the VMP_Controller to update the notation array with the information retrieved from the user that is stored in the input module. |                           |
| CLICKM          | User clicks on measure the he/she desires to edit.                                                                                      |                           |
| DIS             | The calling program sending a signal to cease the input and output interfaces from interacting with the user.                           |                           |
| CLOSE           | Closing the VMP                                                                                                                         |                           |

## **Response Definitions:**

| <b>Response</b> | <b>Description</b>                                   | <b>Requirements Trace</b> |
|-----------------|------------------------------------------------------|---------------------------|
| START           | Do instantiation process for VMP_Output; Constructor |                           |
| SETUP           | Setup VMP according to input                         |                           |
| ACCEPT          | The VMP module allowing user input.                  |                           |
| UP_DIS          | Update output interface to reflect user input.       |                           |
| SENDM           | Sends message to of the measure to be edited.        |                           |
| BLOCK           | The VMP disallowing user interaction.                |                           |
| END_VMPOUT      | Closing the VMP; Destructor                          |                           |
| ERROR           | Report misuse of module.                             |                           |

## Sequence Enumeration:

### Length Zero

| Sequence | Response | Equivalence | Req. Trace No. |
|----------|----------|-------------|----------------|
| —        | 0        |             |                |

### Length One

| Sequence | Response | Equivalence | Req. Trace No. |
|----------|----------|-------------|----------------|
| INIT     | START    |             |                |
| SETOUT   | $\omega$ |             |                |
| EN       | $\omega$ |             |                |
| UPDATE   | $\omega$ |             |                |
| CLICKM   | $\omega$ |             |                |
| DIS      | $\omega$ |             |                |
| CLOSE    | $\omega$ |             |                |

### Length Two

| Sequence    | Response   | Equivalence | Req. Trace No. |
|-------------|------------|-------------|----------------|
| INIT INIT   | START      | INIT        |                |
| INIT SETOUT | SETUP      | INIT        |                |
| INIT EN     | ACCEPT     |             |                |
| INIT UPDATE | UP_DIS     | INIT        |                |
| INIT CLICKM | 0          | INIT        | D3             |
| INIT DIS    | ERROR      | INIT        |                |
| INIT CLOSE  | END_VMPOUT | —           |                |

### Length Three

| Sequence       | Response | Equivalence | Req. Trace No. |
|----------------|----------|-------------|----------------|
| INIT EN INIT   | START    | INIT        |                |
| INIT EN SETOUT | ERROR    | INIT EN     | D5             |

|                |         |         |  |
|----------------|---------|---------|--|
| INIT EN EN     | ERROR   | INIT EN |  |
| INIT EN UPDATE | UP_DIS  | INIT EN |  |
| INIT EN CLICKM | SENDM   | INIT EN |  |
| INIT EN DIS    | BLOCK   | INIT    |  |
| INIT EN CLOSE  | END_VMP | —       |  |

## Canonical Sequence Analysis:

### Finding Orthogonal State Variables:

| Canonical Sequence | Initialized | Enabled |
|--------------------|-------------|---------|
| —                  | False       | None    |
| INIT               | True        | False   |
| INIT EN            | True        | True    |

### State Variables:

| Canonical Sequence | Range         | Initial Value |
|--------------------|---------------|---------------|
| Initialized        | {True, False} | False         |
| Enabled            | {True, False} | False         |

### Canonical Sequence Detail:

| Canonical Sequence                                  | State Variables        | Value Before Current Stimulus | Value After Current Stimulus |
|-----------------------------------------------------|------------------------|-------------------------------|------------------------------|
| —                                                   | --                     | --                            | --                           |
| INIT<br>VMP_Controller<br>has been<br>instantiated. | Initialized            | False                         | True                         |
| INIT EN<br>User interaction<br>has been enabled.    | Initialized<br>Enabled | True<br>False                 | True<br>True                 |

## **Specification Functions:**

### **Specification Function: Initialized(h)**

| Stimulus | Prefix Condition   | Value          |
|----------|--------------------|----------------|
| —        | h = empty          | False          |
| INIT     | Any                | True           |
| s        | Otherwise (h = ps) | Initialized(p) |

### **Specification Function: Enabled(h)**

| Stimulus | Prefix Condition   | Value      |
|----------|--------------------|------------|
| —        | h = empty          | Program    |
| EN       | Enabled(p) = False | True       |
| s        | Otherwise (h = ps) | Enabled(p) |

## **Black Box Tables:**

### **Current Stimulus: INIT**

| Prefix Condition | Response | Trace |
|------------------|----------|-------|
| Any              | START    |       |

### **Current Stimulus: SETOUT**

| Prefix Condition                            | Response | Trace |
|---------------------------------------------|----------|-------|
| Initialized(p) = True<br>Enabled(p) = False | SETUP    |       |
| Initialized(p) = True<br>Enabled(p) = True  | ERROR    |       |

### **Current Stimulus: EN**

| Prefix Condition                            | Response | Trace |
|---------------------------------------------|----------|-------|
| Initialized(p) = True<br>Enabled(p) = False | ACCEPT   |       |
| Initialized(p) = True<br>Enabled(p) = True  | ERROR    |       |

### **Current Stimulus: UPDATE**

| Prefix Condition      | Response | Trace |
|-----------------------|----------|-------|
| Initialized(p) = True | UP_DIS   |       |

**Current Stimulus: CLICKM**

| Prefix Condition                            | Response | Trace |
|---------------------------------------------|----------|-------|
| Initialized(p) = True<br>Enabled(p) = False | Null     |       |
| Initialized(p) = True<br>Enabled(p) = True  | SENDM    |       |

**Current Stimulus: DIS**

| Prefix Condition                            | Response | Trace |
|---------------------------------------------|----------|-------|
| Initialized(p) = True<br>Enabled(p) = False | ERROR    |       |
| Initialized(p) = True<br>Enabled(p) = True  | BLOCK    |       |

**Current Stimulus: CLOSE**

| Prefix Condition | Response | Trace |
|------------------|----------|-------|
| Any              | END_VMP  |       |

## **State Box Table:**

### **Current Stimulus: INIT**

| Current State | Response | New State                               |
|---------------|----------|-----------------------------------------|
| Any           | START    | Initialized := True<br>Enabled := False |

### **Current Stimulus: SETOUT**

| Current State    | Response | New State |
|------------------|----------|-----------|
| Enabled := False | SETUP    | No change |
| Enabled := True  | ERROR    | No change |

### **Current Stimulus: EN**

| Current State    | Response | New State       |
|------------------|----------|-----------------|
| Enabled := False | ACCEPT   | Enabled := True |
| Enabled := True  | ERROR    | No change       |

### **Current Stimulus: DIS**

| Current State    | Response | New State        |
|------------------|----------|------------------|
| Enabled := False | ERROR    | None             |
| Enabled := True  | BLOCK    | Enabled := False |

### **Current Stimulus: CLOSE**

| Current State       | Response | New State            |
|---------------------|----------|----------------------|
| Initialized := True | END_VMP  | Initialized := False |

## **Clear Box Design:**

VMP\_Output will be a main stack with no substacks. There will be a group that will act as the manuscript paper and it operate as a “background” as defined by the Revolution language. This group, which will be called OutputPaper, will be comprised of 3 types of objects: a staff(of which there will be multiple instances), a CKT (Cleff, Key Signature, Time Signature) object, and notational images. There will be 6 cards, one for each key signature. Each card will have the appropriate formatting handler for each respective time signature. The card will be set during the SetOutput handler, but will default at 4/4. The notation will be formatted by the following process.

1. The UpdateDisplay handler will pass the value of the measure that was edited along with the notation array.
2. The formatting process will begin on the staff containing that measure. This will save time by not formatting previous staves which are unaffected by the new input.
3. The formatter will first parse the measures among the staves according to how many notes are in each measure or group of measures.
4. Each notational element will be assigned a spacing value based on its rhythm value and the rhythm value of the other notes in the staff.
5. The notes will then be assigned an orientation, depending on there vertical position in the staff and, if part of a beamed group, the vertical positions of the notes it is grouped with.
6. The notes will then be beamed.
7. Ties will be added to the notes.
8. Triplets will be added

The objects will each have their respective handlers as defined here.

**1. IMG\_CKT**

Handlers:

MSG\_Set Cleff, Key, Time Signature

Sets the display of the image according to parameters.

Output:

Visual display of appropriate Cleff, Key and Time Signature.

**2. IMG\_Staff\_x**

Where x is the staff number.

Handlers:

Reset

Set NumMeasures, NotationArray

Output:

Formatted notation.

**3. IMG\_x**

Where x is the number of the image starting at 1 from left to right.

Handlers:

Output:

Appropriate notation represented by data in notation array.

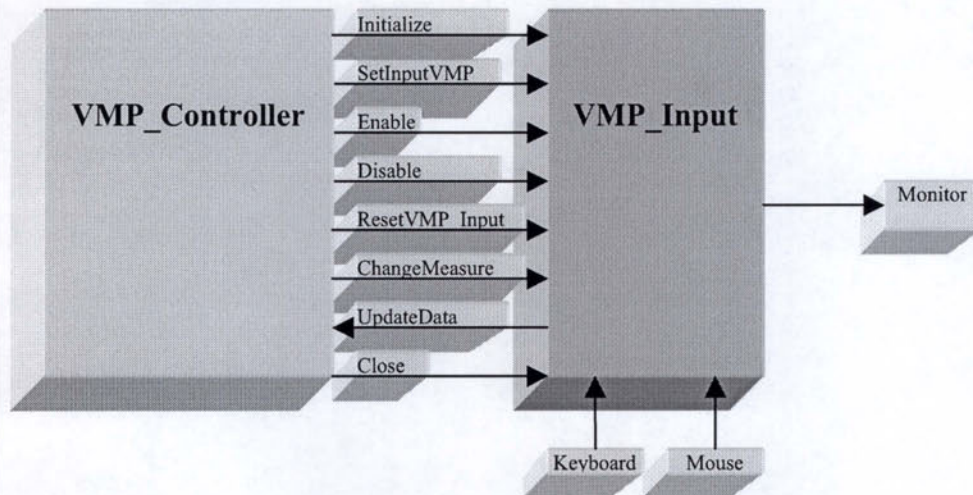
# Component Design:

## VMP\_Input

## **Introduction:**

This section contains the design of the VMP\_Input component. This is the most complex component of the VMP module. Almost all of the user interaction will be done in this component. This interface will attempt to balance the “transparent” interface of notation editors like Finale, but also give the input system required for a testing module. To do this, it will accept input in sections that are separated by the beat. The maximum number of notes allowed in the beat will be the number of sixteenth notes per beat (4 for simple and 6 for complex time).

## System Boundary:



### Inputs:

- **From Calling Program:**
  - Initialize – Occurs when the module is initialized.
  - SetupInputVMP – Sent to module to set the input module to the data sent from program to VMP\_Controller.
  - Enable – Signal to allow user interaction with component.
  - Disable – Signal to disallow user interaction with component.
  - ResetVMP Input – Signal to clear all data stored in VMP\_Input.
  - ChangeMeasure – Sent to change the measure being edited by module.
  - Close – Occurs when module is closed by VMP\_Controller.
- **From User:**
  - Keyboard – Input from keyboard.

- Mouse Click – Mouse clicking on certain elements in interface.

### **Outputs:**

- UpdateData – Sent to update data stored in VMP\_Controller so that it is consistent with user input.
- Displays output of measure being edited to monitor.

## **Stimulus Definitions:**

| <b>Stimulus</b> | <b>Description</b>                                                                      | <b>Requirements Trace</b> |
|-----------------|-----------------------------------------------------------------------------------------|---------------------------|
| INIT            | Initialization of VMP_Input component.                                                  |                           |
| SETINP          | Sent to cause VMP_Input to setup according to data in VMP_Controller.                   |                           |
| EN              | The calling program sending signal to enable the UI to begin interacting with user.     |                           |
| USER            | Either a keystroke or a mouse click that is to be processed by the Input module.        |                           |
| DIS             | The calling program sending a signal to cease the UI from interacting with the user.    |                           |
| RESET           | Sent from VMP_Controller to begin reset procedure.                                      |                           |
| CHANGE          | A signal sent from VMP_Controller to change the measure that is currently being edited. |                           |
| CLOSE           | Closing the VMP_Input component.                                                        |                           |

## **Response Definitions:**

| <b>Response</b> | <b>Description</b>                                                             | <b>Requirements Trace</b> |
|-----------------|--------------------------------------------------------------------------------|---------------------------|
| START           | Do instantiation process for VMP_Input; Constructor                            |                           |
| SETUP           | Setup VMP_Input according to input from VMP_Controller.                        |                           |
| ACCEPT          | The VMP_Input component allowing user input.                                   |                           |
| PROC            | The VMP_Input component processing user input.                                 |                           |
| CHGM            | Changing the measure currently being updated to the one specified by stimulus. |                           |
| BLOCK           | The VMP_Input component disallowing user interaction.                          |                           |
| CLEAR           | The VMP_Input component clearing all internal variables.                       |                           |
| END_VMPINP      | Closing the VMP; Destructor                                                    |                           |
| ERROR           | Report misuse of module.                                                       |                           |

## Sequence Enumeration:

### Length Zero

| Sequence | Response | Equivalence | Req. Trace No. |
|----------|----------|-------------|----------------|
| —        | 0        |             |                |

### Length One

| Sequence | Response | Equivalence | Req. Trace No. |
|----------|----------|-------------|----------------|
| INIT     | START    |             |                |
| SETINP   | $\omega$ |             |                |
| EN       | $\omega$ |             |                |
| USER     | $\omega$ |             |                |
| DIS      | $\omega$ |             |                |
| RESET    | $\omega$ |             |                |
| CHANGE   | $\omega$ |             |                |
| CLOSE    | $\omega$ |             |                |

### Length Two

| Sequence    | Response   | Equivalence | Req. Trace No. |
|-------------|------------|-------------|----------------|
| INIT INIT   | START      | INIT        |                |
| INIT SETINP | SETUP      | INIT        |                |
| INIT EN     | ACCEPT     |             |                |
| INIT USER   | $\omega$   |             |                |
| INIT DIS    | ERROR      | INIT        |                |
| INIT RESET  | CLEAR      | INIT        |                |
| INIT CHANGE | CHGM       | INIT        |                |
| INIT CLOSE  | END_VMPINP | —           |                |

**Length Three**

| Sequence       | Response   | Equivalence | Req. Trace No. |
|----------------|------------|-------------|----------------|
| INIT EN INIT   | START      | INIT        |                |
| INIT EN SETINP | ERROR      | INIT EN     |                |
| INIT EN EN     | ERROR      | INIT EN     |                |
| INIT EN USER   | PROC       | INIT EN     |                |
| INIT EN DIS    | BLOCK      | INIT        |                |
| INIT EN RESET  | ERROR      | INIT EN     |                |
| INIT EN CHANGE | CHGM       | INIT EN     |                |
| INIT EN CLOSE  | END_VMPINP | —           |                |

## Canonical Sequence Analysis:

### Finding Orthogonal State Variables:

| Canonical Sequence | Initialized | Enabled |
|--------------------|-------------|---------|
| —                  | False       | None    |
| INIT               | True        | False   |
| INIT EN            | True        | True    |

### State Variables:

| Canonical Sequence | Range         | Initial Value |
|--------------------|---------------|---------------|
| Initialized        | {True, False} | False         |
| Enabled            | {True, False} | False         |

### Canonical Sequence Detail:

| Canonical Sequence                                  | State Variables        | Value Before Current Stimulus | Value After Current Stimulus |
|-----------------------------------------------------|------------------------|-------------------------------|------------------------------|
| —                                                   | --                     | --                            | --                           |
| INIT<br>VMP_Controller<br>has been<br>instantiated. | Initialized            | False                         | True                         |
| INIT EN<br>User interaction<br>has been enabled.    | Initialized<br>Enabled | True<br>False                 | True<br>True                 |

## **Specification Functions:**

### **Specification Function: Initialized(h)**

| Stimulus | Prefix Condition   | Value          |
|----------|--------------------|----------------|
| —        | h = empty          | False          |
| INIT     | Any                | True           |
| s        | Otherwise (h = ps) | Initialized(p) |

### **Specification Function: Enabled(h)**

| Stimulus | Prefix Condition   | Value      |
|----------|--------------------|------------|
| —        | h = empty          | Program    |
| EN       | Enabled(p) = False | True       |
| s        | Otherwise (h = ps) | Enabled(p) |

## **Black Box Tables:**

### **Current Stimulus: INIT**

| Prefix Condition | Response | Trace |
|------------------|----------|-------|
| Any              | START    |       |

### **Current Stimulus: SETINP**

| Prefix Condition                            | Response | Trace |
|---------------------------------------------|----------|-------|
| Initialized(p) = True<br>Enabled(p) = False | SETUP    |       |
| Initialized(p) = True<br>Enabled(p) = True  | ERROR    |       |

### **Current Stimulus: EN**

| Prefix Condition                            | Response | Trace |
|---------------------------------------------|----------|-------|
| Initialized(p) = True<br>Enabled(p) = False | ACCEPT   |       |
| Initialized(p) = True<br>Enabled(p) = True  | ERROR    |       |

### **Current Stimulus: USER**

| Prefix Condition                           | Response | Trace |
|--------------------------------------------|----------|-------|
| Initialized(p) = True<br>Enabled(p) = True | PROC     |       |

**Current Stimulus: DIS**

| Prefix Condition                            | Response | Trace |
|---------------------------------------------|----------|-------|
| Initialized(p) = True<br>Enabled(p) = False | ERROR    |       |
| Initialized(p) = True<br>Enabled(p) = True  | BLOCK    |       |

**Current Stimulus: RESET**

| Prefix Condition                            | Response | Trace |
|---------------------------------------------|----------|-------|
| Initialized(p) = True<br>Enabled(p) = False | CLEAR    |       |
| Initialized(p) = True<br>Enabled(p) = True  | ERROR    |       |

**Current Stimulus: CHANGE**

| Prefix Condition                            | Response | Trace |
|---------------------------------------------|----------|-------|
| Initialized(p) = True<br>Enabled(p) = False | ERROR    |       |
| Initialized(p) = True<br>Enabled(p) = True  | CHGM     |       |

**Current Stimulus: CLOSE**

| Prefix Condition | Response | Trace |
|------------------|----------|-------|
| Any              | END_VMP  |       |

## **State Box Table:**

### **Current Stimulus: INIT**

| Current State | Response | New State                               |
|---------------|----------|-----------------------------------------|
| Any           | START    | Initialized := True<br>Enabled := False |

### **Current Stimulus: SETINP**

| Current State    | Response | New State |
|------------------|----------|-----------|
| Enabled := False | SETUP    | No Change |
| Enabled := True  | ERROR    | No Change |

### **Current Stimulus: EN**

| Current State    | Response | New State       |
|------------------|----------|-----------------|
| Enabled := False | ACCEPT   | Enabled := True |
| Enabled := True  | ERROR    | No Change       |

### **Current Stimulus: USER**

| Current State   | Response | New State |
|-----------------|----------|-----------|
| Enabled := True | PROC     | No Change |

### **Current Stimulus: DIS**

| Current State    | Response | New State        |
|------------------|----------|------------------|
| Enabled := False | ERROR    | No Change        |
| Enabled := True  | BLOCK    | Enabled := False |

**Current Stimulus: RESET**

| Current State    | Response | New State |
|------------------|----------|-----------|
| Enabled := False | Clear    | No Change |
| Enabled := True  | ERROR    | No Change |

**Current Stimulus: CHANGE**

| Current State    | Response | New State |
|------------------|----------|-----------|
| Enabled := False | ERROR    | No Change |
| Enabled := True  | CHGM     | No Change |

**Current Stimulus: CLOSE**

| Current State       | Response | New State            |
|---------------------|----------|----------------------|
| Initialized := True | END_VMP  | Initialized := False |

## **Clear Box Design:**

The Input\_VMP module will be implemented as main stack with 6 substacks, one for each meter. The mainstack will choose which substack will be in use, depending on the parameters in SetInputVMP. For each substack, one measure will be stored on each card. During SetInputVMP, the initial notation will be entered into each measure. The approach will be to maximize by speed by keeping everything in memory instead of setting up the different measures on an as needed basis.

The display will feature a group containing control buttons and a group that is the measure. The buttons in the button group will control the input state, will allow scrolling between measures, and other needed functions. The measure group will be a collection of beat groups. A beat group will be a collection of buttons that will process mouse input depending on where the mouse is clicked and the input state. The notation elements will be made up of groups that contain the appropriate notation image with a transparent button placed above it that will interpret mouse clicks.

The beat objects will allow the maximum number of the shortest notes per beat. Since the lowest note is the sixteenth note, a simple meter will allow four notes a beat and a complex will allow six. The only two instances where this is not followed are for the cases of the whole rest and half rest. In order to maintain some semblance of proper formatting, the whole note and half note will erase the contents of the beats that the rhythm overlaps and it will be placed within the proper place in the measure.