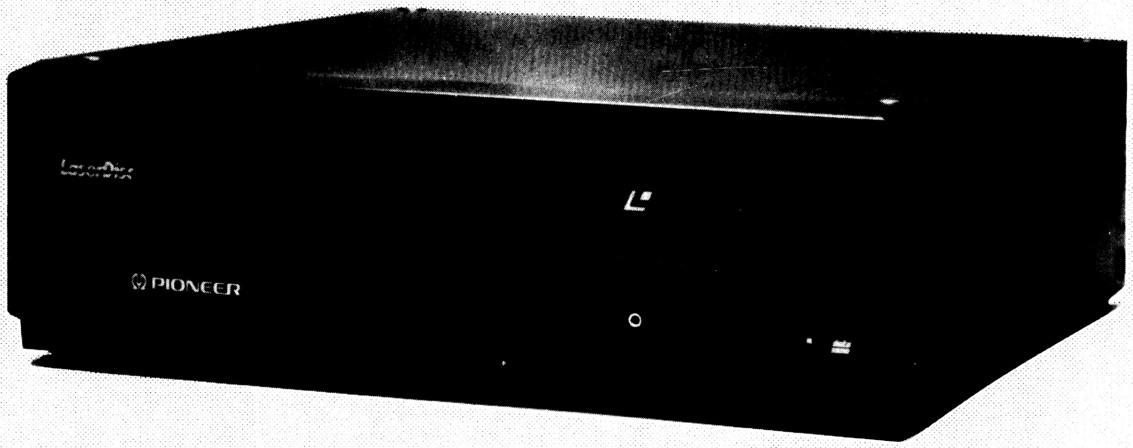
PIONEER®

PIONEER COMMUNICATIONS OF AMERICA, INC.



See inside this Reference Guide for:
Level II — Internal Program Control

LD-V8000

INDUSTRIAL LASERDISC PLAYER

LEVEL II

USER'S MANUAL

PROGRAMMER'S REFERENCE GUIDE

Pioneer Technical Publication DTP 114 V. 1.0 7/91

LD-V8000 Level II Documentation

For Internal Program Control

Note to Users

This manual is based on the most up-to-date information for Level II program development and delivery on the LD-V8000 available at the time of publication. It is subject to change without notice. Although every reasonable effort has been made to include accurate information, the statements in this document are not warranties.

Pioneer New Media Technologies, Inc., makes no warranty or claims as to the accuracy, completeness or fitness for any particular purpose of the technical information provided herein. Throughout this manual **NOTES** appear reflecting details of the particular player functions which may be different on future players. The **NOTES** are included to aid understanding, but should not be depended upon in designing applications.

Please fill out the Registration Form on the next page and return it to us to insure that you receive updated versions of the Level II Manual for the LD-V8000, and related support materials as they become available. Also, comments, observations, and/or corrections regarding this document would be appreciated.

For more information on Level I & III Program Control for the LD-V8000, please refer to the ***LD-V8000 Level I & III User's Manual /Programmer's Reference Guide***. The Level I & III Manual for the LD-V8000 is available from Pioneer New Media Technologies, Inc., Technical Support/System Integration, 310-952-2111

LD-V8000 Level II DOCUMENTATION
for Internal Program Control

CONTENTS

Appendices:

Accompanying Figures, by Chapter:

Chapter Two

Chapter Three

Chapter Four

1. Introduction

CHAPTER

1

LD-V8000

LEVEL II

USER'S MANUAL

Programmer's Reference Guide

1 Introduction

Before you use the LD-V8000, please read the safety information contained in the ***Operating Instructions*** packaged with the player. For an overview of the three player control methods: Level I, II and III, and for more details on player Operational Basics, refer to the Pioneer ***LD-V8000 Level I & III User's Manual / Programmer's Reference Guide***, Technical Publication 113 Version. 2.0 3/91.

1.1 Level II and the LD-V8000

Although several earlier Pioneer Industrial Videodisc Players contained programmable memory allowing for the execution of Level II programs, added features of the Pioneer LD-V8000 now provide even more advanced Level II capabilities. This ***LD-V8000 Level II User's Manual/Programmer's Reference Guide*** provides information to assist programmers in the development of Level II program applications for Pioneer's model LD-V8000 industrial videodisc player.

The LD-V8000 is a highly flexible, programmable playback system that employs a laser to read video and audio program material from a rotating videodisc. An internal microprocessor controls all phases of the Pioneer LD-V8000's operation, processing external commands, internally-stored commands, and internally generated control and status signals. The microprocessor makes possible the player's many "Play", "Search", and "Display" functions. The player includes 7K of user-accessible Random Access Memory (RAM), allowing for the presentation of significant player-controlled interactive videodisc applications.

Because the LD-V8000 is programmable, the exact sequence and display of information presented to the viewer can be predetermined by an interactive program designer and computer programmer. Audio-visual applications developed for a wide variety of uses may be executed by the player's internal microprocessor. Properly constructed "stand-alone" Level II programs allow a wide range of dynamic viewer interactions with the displayed material. This Level II program may be loaded onto the player manually, downloaded from a computer, or read from a Level II videodisc.

When a videodisc is manufactured with a properly formatted "Level II" program encoded in the first few seconds of its Audio Channel 2, it is referred to as a Level II disc. In addition to the normal Audio and Video, the Level II disc contains computer readable instructions that define all, or part of, an interactive application. Interactive programs stored within and executed by the LD-V8000 are referred to as Level II programs, even if they do not require a Level II disc.

When a Level II videodisc is spun-up on the LD-V8000, its Level II program can be automatically loaded into the player's memory. When executed, the program will tell the player what audio and video to present, and how to respond to user inputs. Using a Remote Control Unit (RCU) or other input device, the viewer provides the player with the inputs used by the Level II program to guide its logical path. In addition, the player can command external devices and can monitor some external inputs. Typical interactive programs are written to shape the presentation of audio-visual material to the user's unique requirements — without the need for an external computer to be attached to the videodisc player. Properly designed programs for the LD-V8000 can bring outstanding performance, flexibility, and interaction to applications developed for industry, business, education, entertainment, and other uses.

This manual is intended to be a reference guide for programmers. It is *not* intended to be an instructional course in Level II programming. It contains explanations of concepts, terms, and Level II commands. If you are new to Level II program development and/or plan to produce a Level II videodisc, we strongly recommend working closely with an experienced Level II computer programmer,

Additional information may be obtained from Pioneer New Media Technologies, Inc., West Coast Engineering Support, 310/952-2111 or East Coast Engineering Support 201-327-6400.

1.2 Chapter Highlights

This manual is divided into chapters providing the following information:

Chapter One — Introduction

This chapter provides an overview of Level II videodiscs and the Pioneer LD-V8000 videodisc player. It also includes a summary of what information is included in each chapter.

Chapter Two — Level II Basics

This chapter provides the basic concepts required for understanding Level II programming and an overview of loading and executing Level II programs. This is baseline information for new Level II application developers and is intended as a reference for Level II application designers and programmers. This chapter contains a CAUTION section which highlights the different hardware capabilities and Level II commands available on the LD-V8000 as compared to the LD-V6000A. It also provides an overview of the following essential subjects: Random Access Memory; Addressable Program Areas and Registers; Program Formats; Arguments and Commands; Level II Program Structure; and Command Execution Speed.

Chapter Three — Using Level II

This chapter explains how to enter and execute short Level II demonstration and test programs using either the remote control unit or an external computer. The computer is attached to the player's RS232C port. The chapter also provides information about preparing Level II programming for encoding onto a videodisc.

Chapter Four — Level II Commands for LD-V8000

This chapter presents definitions and explanations of all LD-V8000 Level II commands, both those previously available on the LD-V6000A and the new commands that take advantage of the special capabilities of the LD-V8000. Commands are presented by category: Program Load Control, Audio/Video Control, Display Control, Player Control, Register Control, Input Processing, Program Execution, Flag Set, Transmit, and Memory Control. Often, examples are included for educational purposes, usually with comments explaining command usage.

The user will also find **Notes** referring to details which may be different on future (or past) players. The information in the **Notes** is included to aid understanding but should not be depended upon in designing applications.

Note: Most Level II program applications developed using the LD-V6000A command set will run on the LD-V8000. However, some new hardware features and accompanying Level II commands available on the LD-V8000 are not available on the LD-V6000 or LD-V6000A.

See ***Appendix A*** for a chart comparing the Level II commands available on different Pioneer programmable players: the LD-V8000, the LD-V6000A, the LD-V6000, the LD-V3000, and the PR7820-3. For a complete alphabetical listing of the Level II commands available for the LD-V8000, see ***Appendix B***. Also, refer to the ***LD-V8000 Level I & III User's Manual/Programmer's Reference Guide, Appendix C***: ***LD-V8000 Interface Cable Specifications***, for the RS-232C port specifications and some cable configurations useful in attaching various computers.

2. Level II Basics

CHAPTER

2

LD-V8000

LEVEL II

USER'S MANUAL

Programmer's Reference Guide

2 Level II Basics

This chapter is intended to familiarize Pioneer LD-V8000 users with basic concepts, terms, and procedures associated with developing and delivering Level II program applications. A Level II program consists of a series of commands that, when stored in the player's RAM and interpreted by the microprocessor, cause the player to operate in a pre-defined way.

This chapter explains how Level II programs can be written into the player's memory. It contains a CAUTION section detailing the differences between the LD-V8000 and the LD-V6000A. It describes the player's 7K Random Access Memory — both the Program Area and the Registers. It also describes the elements of a Level II Program: Arguments, Commands, variables , and data. Finally, the chapter covers the structure of Level II Program code in memory and instruction Execution Speed.

2.1 What is Level II?

As with the authoring languages used for other interactive video productions, the Level II programming language is responsive to new hardware capabilities. Thus, it continues to evolve. Sophistication of Level II applications depends upon the increasing knowledge of programmers and developers who work with the system.

Any successful interactive videodisc production requires meticulous planning and Level II applications are no exception to the rule. A well-planned, carefully programmed Level II application can provide a very complex and highly interactive application that is extremely easy to work with — requiring no prior computer knowledge on the part of the viewer. The Level II system allows both simple and complex interactive programming to be delivered with only a Level II videodisc player, a remote keypad, a monitor, and, of course, a videodisc. It does not require an external computer to send commands to the player.

In most cases, a Level II program is prepared and tested in advance of disc production. The program's object code is encoded into Audio Channel 2 when the videodisc is manufactured. When the start-up parameters on the LD-V8000 are set for automatic Level II execution, the disc's Level II program is loaded into the player's memory just after the disc spins up. However, Level II discs are not required. For some applications, the "Level II" program is simply entered into the player's memory with the RCU or downloaded into the videodisc player's memory from a computer.

For trade shows, museums, and other situations where the program and the videodisc do not change, the program might simply reside in the player. However, when there is a library of training videodiscs to choose from, it is quite convenient to select the one disc (lesson) you want, put it into the player, and have the whole lesson ready to run. No floppy disks or complex startup procedure. Likewise, when a network of point of purchase kiosks must be updated monthly, it is again most convenient to simply mail out a new videodisc for the store manager to put into the player. The simplicity and overall low cost of updating the entire interactive network in this manner is noteworthy.

Level II applications have saved developers and system integrators thousands of dollars in hardware costs by allowing applications to be used in multiple settings without an expensive computer at every workstation. Level II applications provide, in effect, “stand-alone systems”. When Level II programs are carefully planned and efficiently developed, companies find that cost savings using Level II program delivery are substantial, primarily because a computer is not required to control each videodisc player kiosk or station. Level II delivery systems are often comprised entirely of “off the shelf” components. This can lower the cost, allow for faster system delivery and simplify set-up for customers.

When a Level II encoded videodisc spins-up on the LD-8000, program code on Audio Channel 2 of the disc can be automatically loaded or “dumped” into the LD-V8000’s 7K of RAM. The program information is written into one of seven “pages”, where each page can contain 1022 bytes of information.

Earlier Pioneer Industrial Videodisc Players (the LD-V6000 series, the LD-V3000, and the PR-V7820-3) are also capable of loading and executing Level II programs. However, they all have slightly different hardware capabilities. Thus, the available Level II commands also vary from player to player to reflect these differences.

(See ***Section 2.4, CAUTION: Differences Between the LD-V8000 and the LD-V6000A.*** See also ***Appendix A, Comparison of Level II Commands Available on Different Pioneer Industrial Laserdisc Players***.)

The succession of players has been generally upward compatible. Usually discs for the PR-V7820-3 run on the LD-V6000, LD-V6000A, and on the LD-V8000. In order to take advantage of the latest hardware features, we urge developers to write Level II programs for use with specific players. Note that other players in the current Pioneer Industrial Videodisc product line: the LD-V2000, LD-V2200, CLD-V2400, LD-V4400, and the LC-V330 (Autochanger), are ***not*** “Level II” players and do not contain an internal microprocessor that will execute Level II programs.

When new LD-V8000 players are purchased, Level II programs written for and tested on LD-V8000 players containing older EPROMs (Erasable Programmable Read Only Memory chips) should be thoroughly tested with the newer EPROMs that may contain improvements and additional features. (See ***Technical Bulletin #131, LD-V8000 Version Upgrade***.)

All Level II programs should be thoroughly tested, preferably with a proof disc, before the master stamper is made and replicates are pressed. A proof disc is a pre-replication test disc containing all program video, audio, and Level II program information. It is used to verify and confirm both the video and audio material and the disc’s interactive programming functions.

Caution: Pioneer Level II programs will not execute on "Level II players" produced by other manufacturers, because Level II languages vary between manufacturers.

Level II programming code is usually developed and tested using an external computer to edit, compile, download, and test the application. Then, a properly formatted object code version of the program is submitted to a specific videodisc manufacturer for encoding into Audio Channel 2 of the videodisc. Although hundreds of dumps can be placed on a single disc, ***typically, from one to fifteen program dumps (1022 bytes each) are encoded onto a Pioneer videodisc.***

2.2 Loading and Executing Level II Programs

The LD-V8000's microprocessor, in addition to directly controlling the videodisc player's operations, provides the user with seven pages and one extra register (7156 bytes) of RAM. This memory space is available for the storage of user-designed Level II program instructions, associated data, and variables. Level II program instructions stored in RAM are executed by an interpreter program that is resident in the player's EPROMs.

2.2.1 Loading Level II Program Code into RAM

Programming code may be entered directly into the player's memory to allow execution of stand alone programs that do not require a Level II disc. More commonly, however, interactive Level II programs are generated on a computer, downloaded to the player for testing, and then sent to videodisc manufacturing to be encoded onto a videodisc. Then, the ***On-Screen Function Switch Settings*** of the LD-V8000 can be set to automatically load a Level II program from a videodisc into the player's memory. The program read from disc is executed to control the player.

Thus, Level II programs are loaded into RAM from three different sources:

- ***Automatically from Disc***, by reading programs encoded on Audio Channel 2 of a Level II videodisc (Program Dumps).
- ***Manually with the RCU***, entering program argument, command, and data codes individually, using the RU-V6000T remote control unit.
- ***Via the RS-232C port from an External Computer***, downloading each page of codes in just a few seconds.

To prepare the LD-V8000 to automatically load a Level II program from a Level II videodisc, make sure that the player's ***Level II Auto Start*** parameter is set to ***Load From Disc***, as described below:

- Power-On the player while simultaneously pressing the player's DISPLAY button. This allows customization of any of the player's ***On-Screen Function Switch*** settings.

- Press the SCAN FORWARD button four times to select Page 4 of the ***On-Screen Function Switch Setting*** menus.
- Press the STEP FORWARD button to select ***Level II Auto Start***.
- Press the STEP REVERSE button to choose the option, ***Load from Disc***.
- Press the DISPLAY button to store the chosen options.

For more about the available options, see the ***LD-V8000 Level I & III User's Manual/Programmer's Reference Guide: On-Screen Function Switches*** and ***On-Screen Status Displays in Manual Mode, Audio Status Display***.

Then, make sure that Audio Channel 2 is ON as the disc spins up. In general it is not necessary for Audio 2 to be ON for the player to read dumps, but Audio 2 OFF during spinup tells the player to skip its initial dump load detection sequence.

For more information about loading Level II programming codes to the player's RAM using the RCU, see ***Section 3.1***. For information about sending Level II code to the player's RAM from a computer via the RS-232C port, see ***Section 3.2.***

2.2.2 Executing Level II Program Code from RAM

Level II programs loaded from videodisc automatically begin execution when they are loaded. However, one may wish to automatically execute a program that is already in the player's memory independent of the type of disc or the original source of the program. In this case, set ***Level II Auto Start***, as described above, but choose the ***Back-Up Memory*** option. This will cause the player to automatically begin execution of the code stored in memory, as soon as any videodisc is spun up. **CAUTION**: When running programs automatically with the ***Back-Up Memory*** setting, make sure that the program code begins at program address 0.

The Level II program remains in the player's RAM indefinitely, until it is overwritten. The LD-V8000 contains a lithium battery so that a Level II program can be held in memory up to 5 years, even when the player is not plugged in.

To begin execution of a program manually, press the RUN button on the RCU.
To use an external computer to command execution to begin, send a "RUN" (*R) command.

2.2.3 Stopping Level II Program Execution

To stop execution of a Level II program, press the CLEAR/HALT button on the remote control unit, or send a “HALT” command (*H) from an external computer. When the program is “halted”, the player changes from *Automatic Mode* to *Manual Mode*.

NOTE: If the HALT is sent during the execution of an AUTO STOP command, the player will continue execution of the AUTO STOP even though it is in *Manual Mode*.

2.3 Caution: Differences Between the LD-V8000 and the LD-V6000A

Level II developers must be aware of both hardware and Level II language command differences between the LD-V8000 and the LD-V6000A. It is advised that any Level II program be prepared, tested, and then used with specific players. Programs can be carefully written to work "properly" on several player types. Programs using the new LD-V8000 features may not execute "properly" on other players.

Although sales of the LD-V6000A were discontinued in March 1991, Pioneer New Media Technologies, Inc., Engineering Support continues to provide technical support for that player, as well as for other discontinued Pioneer Industrial Videodisc players.

The LD-V8000 has new features and increased capabilities that make it a more advanced player than the LD-V6000A. New features of the LD-V8000 can be accessed with new or modified Level II commands. Some of the new player features are:

- *Four Channels of Audio*: The player can simultaneously read and process two channels of Analog Audio and two channels of Digital Audio. Thus, program designers have access to four independent audio tracts. For example, these could be used as four different interpretations or languages, all relating to the same video material.
- *Video Buffer*. The player's ability to capture and hold any frame of video on the disc provides "seamless" searches, CLV "still-frames", and "Sound-Over-Still". With new Level II commands to control this buffer, special user-programmed effects are possible.
- *CLV Frame Access*. Most Level II commands now apply to CLV videodiscs, facilitating highly interactive CLV applications.
- *Rapid Search Time*. The player provides one-half second access across an entire CAV disc. Three seconds access across an entire CLV disc.

In the past, Level II applications were prepared only for CAV discs because CLV discs did not allow the full range of interactive capabilities (Still Frame, Step Forward and Reverse, Multi-Speed Forward and Reverse, etc.) The LD-V8000's video memory buffer makes possible "CLV interactive", with "frame accurate" CLV searches. Thus, it is now possible to create significant Level II applications for CLV discs.

Special care must be taken when programming Level II applications for CLV discs. CLV "frame access" uses up to seven-digit arguments: for example, 1231514 to represent 1 hour, 23 minutes, 15 seconds, and frame number 14 (of frame 0 to 29 within the second). Usually 6 digits are sufficient, but Level II registers contain a maximum value of 65535 and cannot hold six-digit values for arguments.

Four new Video Buffer control commands have been added to the LD-V8000 command set. For detailed explanations see ***Section 4.2.10***. They are:

- *Set (Video) Memory Switch:*: Allows user control over the Video Buffer, inhibiting the normal automatic use of the Video Buffer by the player. A Field Mode or Frame Mode may be selected for the buffer.
- *Select Read Memory*: For Field Mode, this command selects the video buffer field (0 or 1) to be used for generating the player's video output signal.
- *Memory Write Enable*: Allows disc-generated video signals to be written into the video buffer. In Field Mode, the command selects which video buffer field (0 or 1) will be written.
- *Memory Write Disable*: Disables writing into the video buffer.

Both the LD-V8000 and the LD-V6000 provide Binary Output, but the LD-V8000 does not support the 6000's Ascii-Hex Output mode. The Binary Output is most useful in controlling external serial devices via the RS-232C port, for example, a serial printer used to provide feedback to viewers via a scorecard or coupon print-out.

Note: The Level II Transmit Register commands available on earlier player models have been eliminated, along with a number of the Transmit Status commands. Since RF Output and Antenna Input are not available on the LD-V8000, the Antenna Input Enable and Disable commands have been eliminated. Since CX control is automatic, the CX Enable and CX Disable commands have also been eliminated. (Refer to ***Appendix A, Comparison Chart of Level II Commands available on various Pioneer Videodisc Players***.)

The LD-V8000 offers the best of the LD-V6000A — its programmable memory and command set for Level II programming, along with the best of the LD-V4200 — its standard mnemonic command set for Level III computer control of the player.

The LD-V8000 may receive viewer input during Level II program execution from either RCU, the RU-V6000T or the RU-V103. With the RU-V6000T, one can do manual Level II programming. See ***Section 3.1 Sending Level II Code to the LD-V8000 with the RCU***. Also see ***Appendix F, Flow Chart and Sample Level II Code.***

Note: Since the RU-V103 remote control unit lacks the "Program" button, it cannot be used to write Level II code into the player's RAM.

2.4 Random Access Memory

The LD-V8000 includes a 8086 microprocessor, two EPROMs that contain the basic operating system of the player, and seven kilobytes of Random Access Memory (RAM), of which 7156 bytes are available for Level II programming. The RAM holds program codes, the registers, and other data. This coexistence requires RAM to be addressed in two ways. One addressing method is used to store single-bytes of program code and data. The second method is used to manipulate the values stored in two-byte registers.

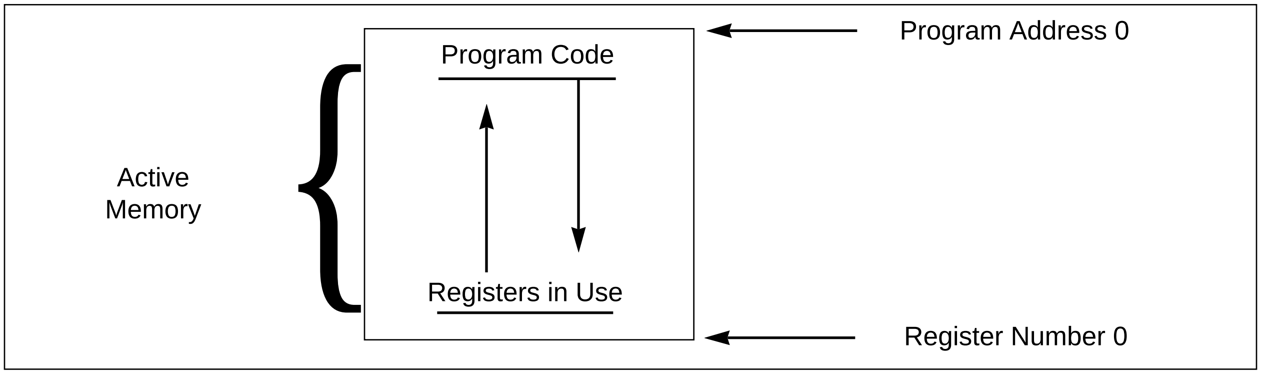


Figure 2-A

Because program code and registers may overlay each other, the programmer must understand how each is structured and addressed. (See **Figure 2-A,** above.) Since there is rarely a good reason to allow program code to overlap the registers in use, the programmer should usually consider such an overlap to be an error, and make every effort to avoid it.

Generally, the program code begins at the lowest address in the active memory and the register area begins at the highest address. This is not always required, but it is usually convenient and sufficient. The size of the program and the number of registers to be used must be managed so that the total size of the program data and the register data does not exceed the total size of the active memory.

Active Memory Size		Memory Locations		
Pages	**Bytes**	**Range**	**Hi Byte Reg N**	**Low Byte Reg N**
1	1024	0-1023	1022 - 2*N	1023 - 2*N
2	2046	0-2045	2044 - 2*N	2045 - 2*N
3	3068	0-3067	3066 - 2*N	4067 - 2*N
4	4090	0-4089	4088 - 2*N	4089 - 2*N
5	5112	0-5111	5110 - 2*N	5111 - 2*N
6	6134	0-6133	6132 - 2*N	6133 - 2*N
7	7156	0-7155	7154 - 2*N	7155 - 2*N

Figure 2-B

"PAGE" Command	Active Memory Size		Register Numbers	
	Pages	**Bytes**	**Range**	**Reg # at Location M**
0 PAGE	1	1024	0-511	INT ((1023-M)/2)
1 PAGE	2	2046	0-1022	INT ((2045-M)/2)
2 PAGE	3	3068	0-1533	INT ((3067-M)/2)
3 PAGE	4	4090	0-2044	INT ((4089-M)/2)
4 PAGE	5	5112	0-2555	INT ((5111-M)/2)
5 PAGE	6	6134	0-3066	INT ((6133-M)/2)
6 PAGE	7	7156	0-3577	INT ((7155-M)/2)

Figure 2-C

2.4.1 Active Memory

The player's RAM is divided into seven 1022-byte blocks called Pages, and one 2-byte block called Register 0. Each RAM memory location is one eight-bit byte. Memory locations begin at Address 0 and continue to Address 7155. Depending upon the argument of the most recent Page (PAG) command, the active memory may consist of one to seven pages, and Register 0. This gives an Active Memory Size of 1024, 2046, ..., 7156 bytes. Register 0 is not changed by any program load from disc and it occupies two fixed bytes of memory that are separate from all of the pages. However, the two bytes of Register 0 can usually be addressed as the last two program locations of Active Memory.

2.4.2 Program Area

The program area is a part or all of the player's Active Memory. Program instructions (arguments and commands) are written into the program area, along with other data (characters, etc.). A program is usually loaded from a videodisc in units of one page, or, in unusual circumstances, as a partial page.

The size of the active program area can be set by the Page command, allowing from 1 to 7 pages to be active. The actual size of the active program, including Register 0, can be calculated as follows:

Number of active pages x 1022 + 2 bytes

At power-on, there is only one page active (1022 bytes plus Register 0). This provides for compatibility with earlier players and program dumps which do not use the Page command. Beginners will often write program dumps for the LD-V8000 in one of two ways:

- Without using the PAG command, for small programs or when compatibility with the oldest players is desired.
- Using the "6 PAG" command to make the active memory as large as possible.

Program instructions, (arguments and commands) are stored in coded format. Each digit of an argument and each command is a one-byte code which occupies one memory location. For example, the instruction, 1536 Search, consists of a 4-byte argument and the one-byte SC command, represented in memory by the following five bytes of Hex code: 0F, AF, 4F, 6F, and F7.

See ***Chapter 4, Level II Commands the LD-V8000*** for an explanation of specific program commands and ***Appendix B, Alphabetical Listing of Level II Commands Available on the LD-V8000.*** Both ***Chapter 4*** and ***Appendix B*** include the commands' corresponding one-byte Hex codes.

2.4.3 Registers

All or part of any page in active memory can be used to hold register data. Each register occupies two bytes (two memory locations). The most significant byte of a register is at an even program address, the least significant byte is at the next higher location (odd address). There are 511 registers in each active page. Register storage begins at the highest program address and proceeds downward: When only one page is active, Register 0 occupies program addresses 1022 and 1023, while Register 511 occupies program addresses 0 and 1. Since this correspondence of register number to program addresses changes with the use of the Page command, the two program addresses corresponding to a particular register number can be computed with the aid of **Figure 2-B** of this chapter.

Registers store data in a 16-bit binary unsigned integer format. For example, the value 1536 can be stored in a register as the hexadecimal value ‘0600’. Since a register is 16 bits long, it can contain a value of 0 through 65535. There are no negative numbers, and larger numbers are usually taken modulo 65536.

The Active Register Pointer holds a number designating which register is currently considered to be “Active”. Of the available registers, one will always be designated as the current “Active Register”. Register 0 is designated as “active” at power-on. Any register in RAM can be designated as active by specifying it as the argument of a Recall command. In addition, the Active Register Pointer is increased by one whenever one of the following commands are executed: Autostop, Search, Store, and Recall. These commands may use the contents of the current active register. Then, they always activate the next highest register.

The following charts show the relationship between active memory size, program addresses, register numbers, and memory addresses in the active program area. These “Memory Addresses” are the addresses used when reading or writing Level II program code through the RS-232C port.

Note: The relationship between program addresses and register numbers changes if the size of the active memory (number of active pages) is changed. Reading or writing program codes through the RS232C port always uses the memory addresses, and all of the pages can be accessed, independent of the size of active memory.

The whole RAM memory area is shown below.

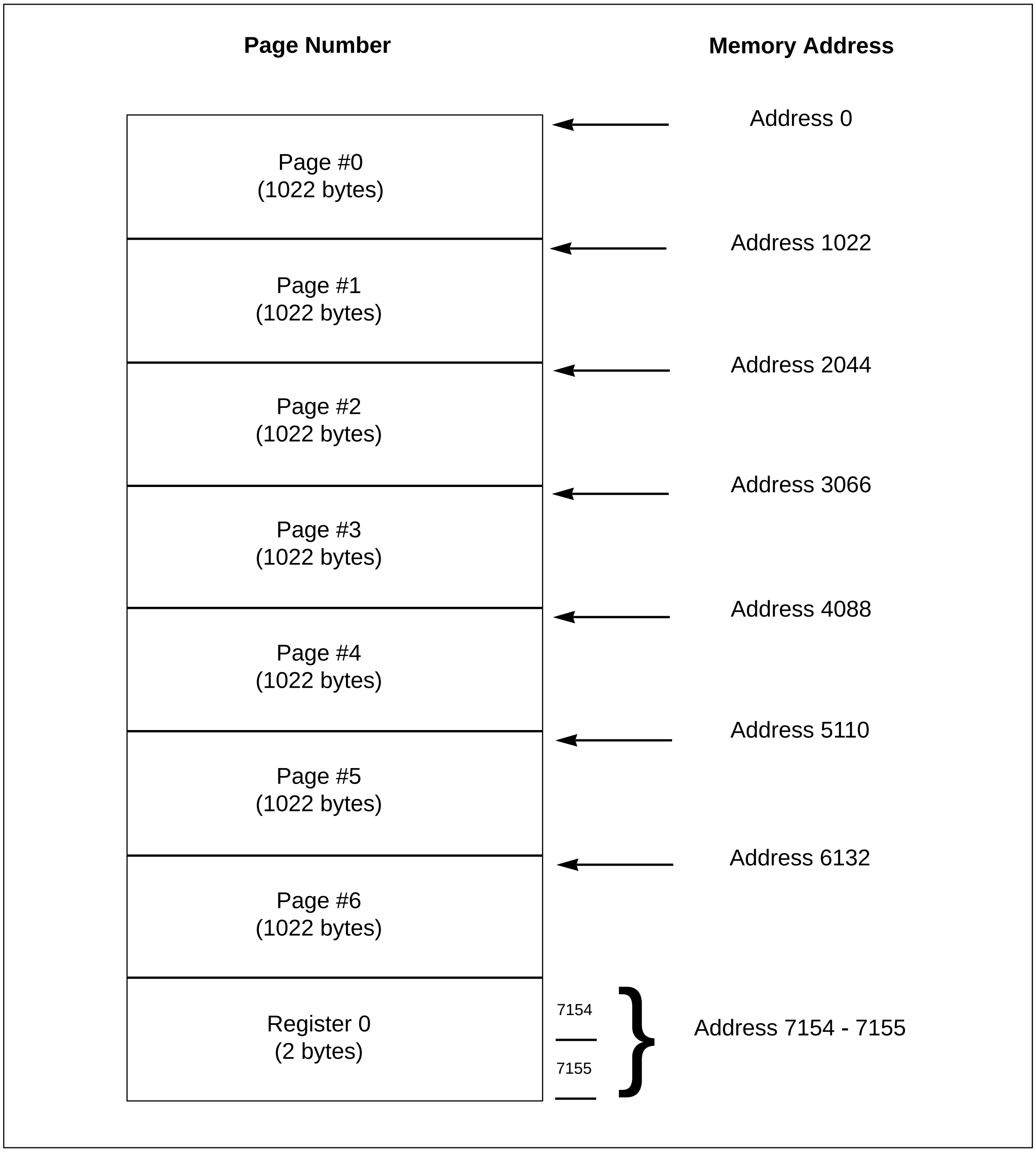


Figure 2-D

The relationship between program addresses, register numbers, and memory addresses is shown in the following figures, for one, two, and all seven pages active.

One page active (by using the 0 PAG command):

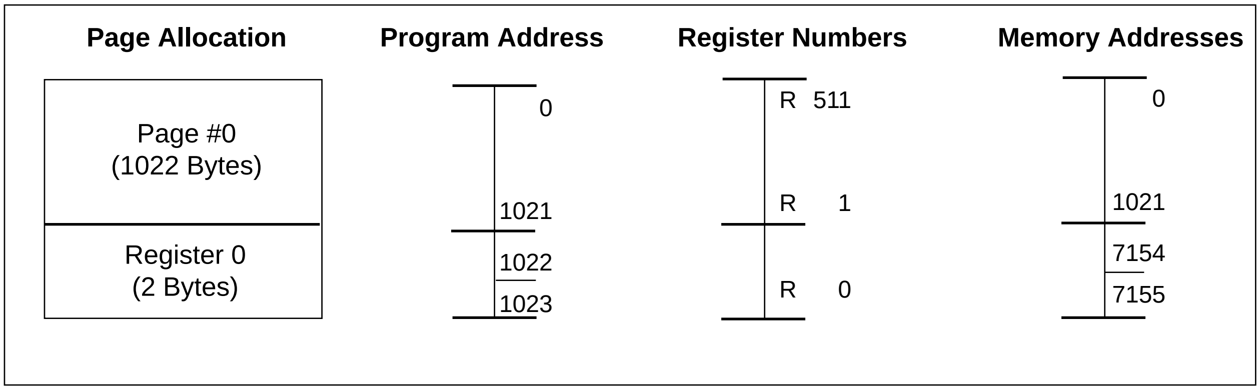


Figure 2-E

Two pages active (by using the 1 PAG command):

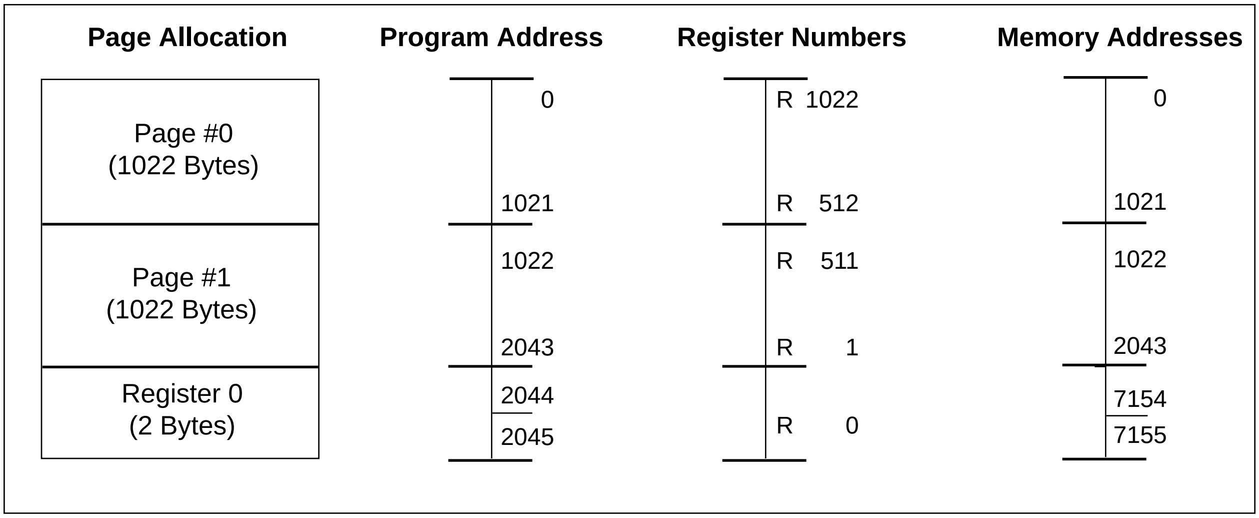


Figure 2-F

Seven pages active (by using the 6 PAG command):

Page Allocation	Program Address	Register Numbers	Memory Addresses
Page #0 (1022 Bytes)	0 1021	R 3577 R 3067	0 1021
Page #1 (1022 Bytes)	1022 2043	R 3066 R 2556	1022 2043
Page #2 (1022 Bytes)	2044 3065	R 2555 R 2045	2044 3065
Page #3 (1022 Bytes)	3066 4087	R 2044 R 1534	3066 4087
Page #4 (1022 Bytes)	4088 5109	R 1533 R 1023	4088 5109
Page #5 (1022 Bytes)	5110 6131	R 1022 R 512	5110 6131
Page #6 (1022 Bytes)	6132 7153	R 511 R 1	6132 7153
Register 0 (2 Bytes)	7154 7155	R 0	7154 7155

Figure 2-G

2.5 Program Format

The following is a brief description of the two parts of a Level II instruction used to control the LD-V8000 videodisc player — the arguments and the command. Also included is an overview of Level II program code structure and of command execution speed. Specific Level II commands and arguments are described in ***Chapter 4**, **Level II Commands for the LD-V8000.***

2.5.1 Arguments

An argument is attached to a command to provide a numeric parameter useful for the command's execution. Arguments represent integer data, CAV or CLV frame numbers, time codes, chapter numbers, program addresses, register numbers, time delays, or other values. In Level II Programs, the argument, if any, is always placed before the command.

Any number of digits can be placed before the command to form the argument. However, only the lower-order seven digits are used for a CLV frame number, the lower two digits for a chapter number, and the lower five digits for most other parameters.

In addition to the ten digits (0 - 9), several other program codes (ARG, DIN, DRP, etc.) are also considered to be argument codes, because they generate argument digits for the command that immediately follows them. For example, 123 ARG DRP ARG 12 ARG is a nine-code argument that creates argument digits for a following command (such as “Search”).

NOTE: Usually, the arguments generated in this manner are five digits (they can be more) and they may be taken modulo 65536. Usually extra high-order digits are ignored. But Beware, the instruction 90000 DRP SC does not search to frame 9000 but the instructions 12345 GET 0 ARG DRP 7 SC may indeed find frame 23457.

Some commands don’t require arguments; others do, often because the default argument (usually zero) does not make sense. When the argument is optional, there is usually a meaningful default or an implied argument can be taken from the active register. Unless otherwise specified, no argument is equivalent to a zero argument.

Each numeric digit of an argument is internally represented as a one-byte code. Thus, each digit (or other argument code) occupies one memory location.

Note: Registers can only hold the values 0 through 65535.

2.5.2 Commands

The Level II command set represents the functions available for development of a Level II program. Many of the commands are direct counterparts of buttons on the RCU (e.g., SEARCH, AUDIO1, DISPLAY, etc.) and they cause corresponding operations to be performed by the player. Other commands are used for controlling program interpretation, directing the path of execution, managing registers, etc. A command is stored as a one-byte code in the active memory.

Any argument must be placed before the command. An argument, if any, and the following command make an instruction. See ***Chapter 4, Level II Commands for the LD-V8000***, for a description of each Level II command. Refer also to ***Appendix B, Alphabetical Listing of Level II Commands Available on the LD-V8000***.

Many commands can be executed directly by the player or entered into RAM from the RCU with a single button press. All codes and any data byte can be entered into RAM as a hexadecimal code, with three button presses on the RCU. See the procedure described in ***Section 3.1.3, Entering Level II Code with the RCU***.

When the programmer enters arguments, commands, and data from the RCU, the video display shows the byte codes on-screen as command or digit mnemonics whenever possible. Unrecognized codes are displayed as two-character hex values.

2.5.3 Program Structure

A Level II program segment, when stored in memory, is a continuous string of one-byte codes. The string is processed by the player's Level II program interpreter beginning at the location specified or implied by the RUN command. As each byte is examined, argument codes are accumulated until an executable command code is found. Some commands have explicit arguments, others have implied arguments, default arguments, or no arguments.

As an example, the two instructions 1000 SC 2000 AS are internally represented as codes 0F, 3F, 3F, 3F, F7, 8F, 3F, 3F, 3F, F3. Starting with the first byte, 0F, the argument is accumulated while the codes are scanned for a command code. In this example, the Search command code, F7, is detected. The SC command, using the currently accumulated argument (0F, 3F, 3F, 3F), instructs the player to position the laser read head at frame 1000.

When the player is executing Level II program code (in *Automatic Mode*), succeeding commands from memory are not processed until the function specified by the “current” command has been completed. The Play command and the INN command are the exceptions. A Play command instructs the videodisc player to begin playing audio-video material and continue until instructed to do something else by another command.

3. Entering Level II Code into RAM

CHAPTER

3

LD-V8000

LEVEL II

USER'S MANUAL

Programmer's Reference Guide

3 Entering Level II Code into RAM

Either the RCU or an external computer attached to the player's RS-232C port may be used to enter simple or complex Level II programs into the player's memory. Complete Level II applications (usually short) are sometimes entered with the RCU and retained in the LD-V8000's memory by it's 5-year battery. However, the RCU is most often used for interactive input, examining variables, or patching and examining code during the testing of larger programs. These are usually written and edited on an external computer, compiled with a Level II computer utility, and downloaded into the player for thorough testing. This procedure is highly reliable and is strongly recommended. ***Section 3.1*** and ***Section 3.2*** of this chapter explain both ways of entering Level II codes into the player's memory.

Sometimes, a Level II simulator utility program is used for testing, instead of downloading code into a player. Since the best simulator may not exactly duplicate a particular model of videodisc player, it is best to test applications on the player actually intended for use, including a final check using a proof disc. No application should be considered ready for mass production until a proof disc has been extensively tested and approved.

Section 3.3 of this chapter describes how Level II programs are encoded in the Audio Channel 2 of a Pioneer videodisc.

Caution: This manual is intended to be a *reference guide*, not an instructional manual in the fundamentals of computer programming. Actual programming procedures and methods are not explained anywhere in this manual. If you decide to create your own Level II videodisc application and you are not familiar with computer programming, we strongly suggest you become familiar with programming concepts. This will help you to make best use of the considerable power of the Level II videodisc programming language. In any case, it is highly recommended that you contract an experienced Level II programmer before you begin the project. Pioneer New Media Technologies, Inc., Engineering Support, can answer questions or refer you.

3.1 Entering Level II Code with the RCU

Here are the steps for entering *Programming Mode* and entering Level II codes into the LD-V8000 using the RU-V6000T remote control unit (RCU):

3.1.1 Entering Programming Mode

Pressing the PROGRAM button on the RCU when the player is ON, or when it is in *Manual Mode* (i.e. when a disc has been spun-up, played and stopped) puts the player into *Programming Mode*, ready to receive Level II program code input. You will see an on-screen program address indicator appear on the video monitor attached to the LD-V8000. If no argument is specified before the PROGRAM button is pressed, programming will begin at address 0. If an argument is used, programming will begin at the specified program address.

3.1.2 Screen Display

When the player is in *Programming Mode*, the monitor displays a four digit (decimal) program address at the upper left of the screen. Mnemonics representing one or more sequential program codes are displayed on a second line. The displayed program address is the address of the rightmost byte of code (command, argument digit, or data) displayed on the second line.

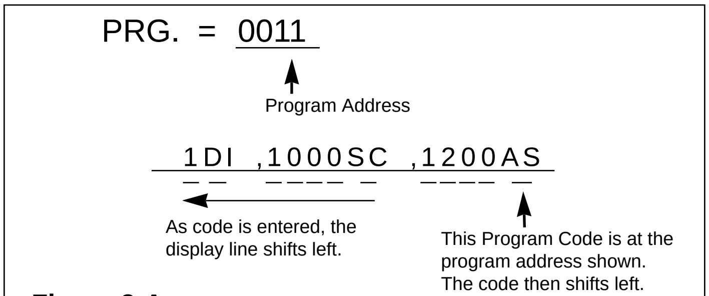
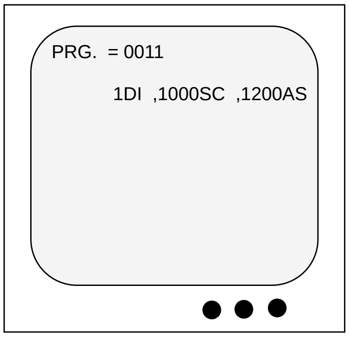

PRG. = 0011
Program Address
1DI ,1000SC ,1200AS
As code is entered, the display line shifts left.
This Program Code is at the program address shown. The code then shifts left.

Figure 3-A



Notice: Each argument digit or command takes up one byte.

The rightmost code byte is special, it's value may be replaced by a code entry from the RCU. When a new code replaces an old code, the display shifts to the right so that the next code byte in active memory is displayed as the rightmost mnemonic. While in *Programming Mode*, the PROGRAM button does not enter a replacement code, it preserves the codes displayed and just shifts the second line left to display the next program code. By pressing the PROGRAM button repeatedly, entire program segments can be reviewed and verified.

The displayed mnemonics differ depending upon the contents of each byte. Each argument digit code is displayed as a one-digit numeric character. Other codes are displayed in a three-character area followed by a comma. Most commands are displayed as a 1, 2, or 3-character mnemonic. Other program codes are displayed as a hexadecimal value preceded by an asterisk (i.e. *BA). A code value being entered using the “Hex Code Entry” method starts to appear on screen in Hex, but the code's mnemonic, if any, is used in the display as the code byte shifts left.

For example, the Set Frame Mode command has no single corresponding RCU button. ***Appendix B, Alphabetical List of Level II Commands*** indicates that the Hex code is 8E and the mnemonic is SFM. As the first Hex digit (the “8”) is entered, one can see an “*8” on the screen. However, as the second Hex digit (the “E”) is pressed, the display will shift to the left showing the SFM command mnemonic and the displayed program address will be incremented by one.

3.1.3 Entering and Changing Program Code

When the RU-V6000 RCU is used to enter program codes, bytes of code in memory are changed one byte at a time. The program address displayed on the screen indicates the memory location of the code byte that will be changed by an entry. The entry is a simple one-for-one replacement - a code cannot be inserted between other codes in memory. If an erroneous code is discovered, a correct code can be rewritten over the offending code. However, if a code is omitted, a whole section of codes may have to be re-entered.

As codes are entered into RAM with the RCU, the program address is incremented by one each time a byte of code is entered. During entry, press the PROGRAM button instead of entering a code value to “skip over” a byte of code without replacing it.

RU-V6000T Remote Control Unit

Buttons used for Level II Programming

In Programming Mode, the following codes can be sent directly to the Player’s memory from the RCU with one button press:

STOP
AUDIO 1/L
INPUT
SEARCH
RECALL
BRANCH

DISPLAY
AUDIO 2/R
DEC REG
AUTOSTOP
STORE
HALT

MULTI-SPEED SET Slow Fast
MULTI-SPEED PLAY Fwd. Rev.
STEP Fwd. Rev.
The Digits 0-9

Pressing **PLAY** prepares the player to receive a two-digit Hex Code entry, using 0-9 and A-F.

Buttons A-F can be used for Hex Entries.

Buttons 0-9 are used for most arguments.

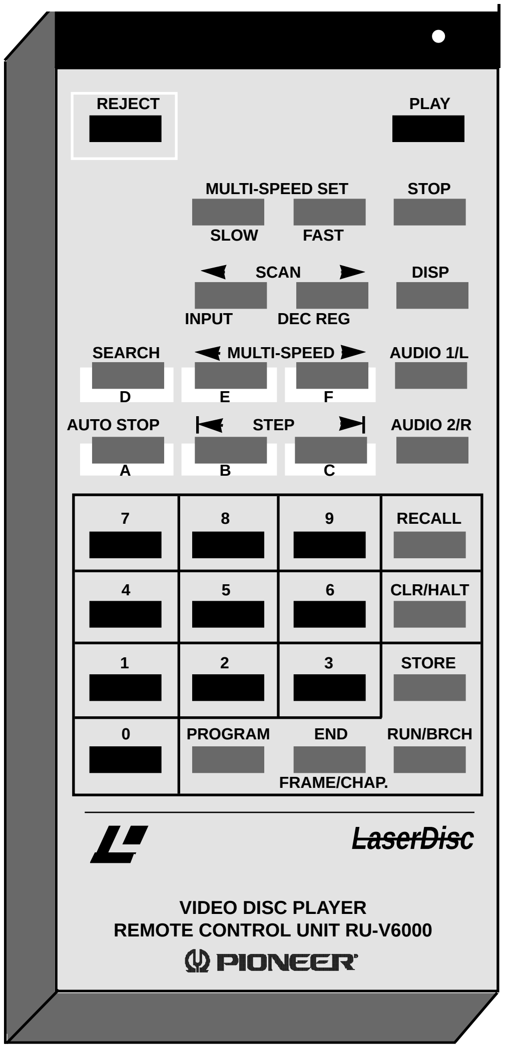


Figure 3-B

Press **PROGRAM** to put the player into *Programming Mode*, ready to receive Level II code. Press **END** to exit *Programming Mode*.

Press **RUN/BRANCH** in *Manual Mode* to execute Level II program code stored in the player’s memory. in *Programming Mode* use it to enter a BRANCH command to loop back to a specific program address.

Press **CLEAR/HALT** to stop Level II program execution. In *Programming Mode*, it enters a HALT command into Level II code.

Press **RECALL** to examine register data. Use STORE to load data into registers.

For descriptions of specific buttons on the RU-V6000T for Level I control, please see the ***LD-V8000 Level I & III User’s Manual/Programmer’s Reference Guide.***

Level II code can be entered into the player's RAM with the RCU by using either the ***Direct Code Entry*** method or the ***Hex Code Entry*** method, as described below:

- ***Direct Code Entry***:

  This is the one-button press method usually used for entering the command codes which have a single RCU button assigned to them (except Play). Using these convenient one-press entries, the user can enter most arguments and the most commonly used commands with just the RCU's numeric and command keys. The RCU buttons other than REJECT, PLAY, PROGRAM, and END can be used for one button-press entries. If the PROGRAM button is pressed, new data is not written into the current byte - the old value is retained and the next byte of the program memory is displayed.

Example #1: Use the ***Direct Code Entry*** method. At program address 100, enter the following program:

250 Search, 350 Auto Stop, Halt

To enter and run the program use the following sequence of RCU button presses:

100	PROGRAM	Start entering code at program location 100.
250	SEARCH	Search to frame 250.
350	AUTOSTOP	Play to frame 350.
	HALT	Stops execution of Level II program.
	END	Exits programming mode.
100	RUN	The player will begin to execute the program code at program address 100.

- ***Hex Code Entry***:

  Commands not represented by an RCU button and most data codes must be entered using a three-button press method. Any and all code values (0 - 255, or 00 to FF) can be entered using this method. After the PLAY button is pressed, use the 0 through 9 digit buttons and the A through F function buttons on the RCU to input a two-digit hexadecimal code value as explained below.

While in *Programming Mode*, press the PLAY button to enable the player to accept entry of a single byte of Hex code. The rightmost mnemonic displayed on the screen changes to *00. Enter a two-digit hexadecimal code using the 0 to 9 and A to F buttons. (For example, the Step Forward button becomes the Hex digit C). Refer to the information in ***Chapter 4*** and/or ***Appendix B*** for Hex code equivalents. When the two-digit code has been entered, the corresponding one-byte code is written into the program memory and the program address is increased by one. To enter another byte of code using the Hex Code Entry method, the PLAY button must be pressed again.

Example #2: At Program location 300 enter the following program, using ***Direct Code Entry*** (one button press) and ***Hex Code Entry*** (three button presses) as necessary:

Set Frame Mode, AUDIO OFF
1200 SEARCH, 1350 AUTO STOP
50 WAIT, 300 BRANCH

Use the RCU button presses below:

300	PROGRAM	Player is put into *Programming Mode*, beginning at program address 300.
	PLAY 8 E	Set Frame Mode, Hex 8E. (SFM is displayed on screen.)
	PLAY A 0	AFF command turns both Audio channels OFF
1200	SEARCH	Searches to frame 1200.
1350	AUTO STOP	Plays the video segment 1200 to 1350.
50	STOP	Waits for five seconds.
300	BRANCH	Loops to location 300 to repeat the video segment.
	END	Exits *Programming Mode*.
300	RUN	Begin Level II execution at Program Address 300.

In the previous example, when the 300 RUN command is given, the player will execute the Level II program (*Automatic Mode*), repeating the video sequence over and over because of the 300 BRANCH command. To stop the program execution, press the CLEAR/HALT button (the HALT command) on the RCU. A HALT changes the player's mode from *Automatic Mode* to *Manual Mode*. If the HALT occurs while the AUTO STOP command is being executed, the player will continue playing to the target frame (unless it is subsequently told to do otherwise).

3.1.4 Exiting *Programming Mode*

Press the END button on the remote control unit to exit *Programming Mode*, usually returning to *Manual Mode*.

3.2 Entering Level II Code via the RS-232 Port

The user can enter (download) Level II code into the player's memory from an external computer by using the RS-232C port. This downloading of data is accomplished by using new Level III commands. Thus, the player must be ON or in *Manual Mode* (such that Level III commands will control the player) before entering Level II code using this *Downloading Mode*.

3.2.1 Downloading Level II Codes

To use *Downloading Mode* to send data (as Hex codes) from a computer to the player's memory, use the following two Level III commands:

• 1. Set the Memory Address pointer

Before sending any data to the player, it is usual to specify the memory location (address) where the first byte of program code is to be written. Use the *S command, as shown in the example below, to start writing data at memory location 100. The argument specifies the memory location as a decimal number. If the *S command is successful, an "R" will be sent by the player. If the memory location has already been specified by other means, this step may be omitted.

```
100*S<CR>
```

Note that memory locations are almost identical to program addresses. See ***Section 2.5, Random Access Memory*** for details about memory locations.

• 2. Download data

Specify a data length (the number of code bytes to be sent to the player) and then send the data. This is a two-step Level III command.

Step 1: Use the *W command to specify a maximum number of data bytes to be written into the player's memory by the next step. Without this, the player would try to interpret the following data as Level III commands. If the *W command is successful, the player will transmit an “R”. The computer should wait for this response from the player before proceeding.

Step 2: Send the data, two Hex digits for each code byte, followed by a carriage return. A maximum of 64 bytes may be specified in Step 1. Thus, a maximum of 128 Hex characters should be transmitted in Step 2.

For example, transfer 8 bytes of data:

```
8*W<CR>
                R
AFBFCFDF01020304<CR>
                R
```

Since a maximum of 64 bytes of Level II code can be sent to the player in one download operation, it will usually be necessary to download many times. Since the memory location pointer is incremented by one each time one byte of program code is written, it is usually not necessary to use the *S command except at the beginning of a sequence of download operations.

Example: The Level II program codes for1000 SEARCH, 1200 AUTOSTOP, HALT are sent to the player’s RAM beginning at address 120. Obtain the Hex codes for each code byte - use ***Chapter 4*** or ***Appendix B*** and ***E***. This 11-byte program is represented by the following Hex codes:

```
1000  SEARCH     -->   0F, 3F, 3F, 3F, F7
1200  AUTOSTOP   -->   0F, 8F, 3F, 3F, F3
      HALT       -->   BF
```

Set the program memory pointer to 120:

```
>     120*S<CR>
<                      R<CR>
```

Set the data length to 11 bytes:

```
>       11*W<CR>
<                         R<CR>
```

Send the Hex codes for the 11 bytes:

```
>       0F3F3F3FF70F8F3F3FF3BF<CR>
<                         R<CR>
```

(The memory location pointer will now be 131.)

When downloading Level II code into the player's memory, the code can be written to any byte of any page, even if the page is not active. Even though they are essentially identical (except for referring to Register 0), Memory Locations (Memory Addresses) are used instead of Program Addresses. See **Figure 2-D**

If one wishes to download data into specific registers, the user must take into account the number of pages that will be active when the registers are accessed by the Level II program. For example, Register 2 would be at Program Address 1018 and 1019 when there is only one page active. However, it will be at 2040 and 2041 if there are just two pages active. Register 0 is always accessed as Memory Locations 7154 and 7155.

Recall that each register uses two bytes of memory. The most significant byte of the register data is written at an even-numbered address (lower program or memory address). The least significant byte is written at an odd-numbered address (higher program or memory address). The formula to calculate the memory locations that a register occupies is shown below.

Memory Location of Register Y, assuming it is accessed by a Level II program after an X PAGE command was executed (with $0 \leq X \leq 6$, and $0 \leq Y \leq (511 \text{ x } (X + 1))$):

If Y = 0, then (for R 0):

a.) The most significant byte of R 0 is at memory location 7154.
b.) The least significant byte of R 0 is at memory location 7155.

If Y >0, then (for R Y):

a.) The most significant byte is at memory location $(((511 \text{ x } (X +1)) - Y) \text{ x } 2$.
b.) The least significant byte is at memory location $(((511 \text{ x } (X +1)) - Y) \text{ x } 2 + 1$.

3.2.2 Reading Level II Codes

Program codes can be read from the player’s memory by an external computer in a manner similar to that used to send them to the player.

First, set the memory location pointer with the *S command just as you would for sending code to the player. The location specified will be the first code byte read.

Second, use the *D command to tell the player now many code bytes to transmit from it's memory. The maximum transmission is 64 bytes of code, followed by a carriage return. Each byte is sent as two Hex characters, so a maximum of 128 characters are transmitted.

To aid reading sequential sections of the player’s memory, the memory location pointer is increased by one every time a byte of data is output.

Example: Read the first 9 bytes of the program downloaded in the previous section.

• Set the memory location pointer.

As before, set the memory location pointer to 120:

```
>       120*S<CR>
<                           R<CR>
```

• Transfer Data.

Ask the player to transmit 9 bytes of code:

```
>       9*D<CR>
<       0F3F3F3FF70F8F3F3F<CR>
```

The memory location pointer will be at 129 when the player’s transmission is complete.

At this point the programmer should note that the player’s Program Address Pointer is used for two different purposes:

First, it is used during the execution of a Level II program to indicate the instruction being processed. While being used in this manner, the Program Address counter will point to a location in active memory.

Second, it is set by the *S command and is used as a Memory Location Pointer by the Level III downloading and code reading commands. At this time, it may point to any of the byte locations (addresses) in memory, even if only one page is active.

Since it might be useful for an external computer to monitor the execution of a Level II program, the player has a new *P command (Level III) to ask the player to transmit the value of the Program Address Pointer.

• Transmit the value of the Program Address Pointer
The player transmits four decimal digits followed by a carriage return. For example:

```
        >       *P<CR>
        <       0129<CR>
```

3.3 Level II Programs Encoded on Videodiscs

As mentioned at the beginning of this chapter, programmers preparing a Level II program to be encoded onto a videodisc will most likely use an authoring utility program of some sort. Programmers might develop their own compilers and other utilities or use commercially available authoring support tools. You may contact Pioneer New Media Technologies, Inc., Engineering Support for more information. The West Coast number is (310) 952-2111; the East Coast number is (201) 327-6400.

It is highly recommended that the entire Level II application be carefully structured before any programming begins. Like scriptwriting and storyboarding, the interactive program should be well thought out and reasonably well documented before any video or audio production begins. This will help eliminate the need for the costly re-do of video sequences that don't quite fit into the intended interactive framework.

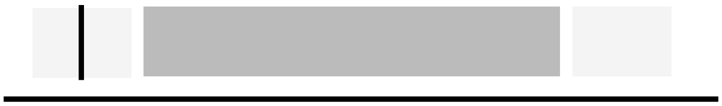
A flow chart provides a symbolic roadmap for any interactive application. The intent of the flowchart is to clearly document the intended interactions and sequence of visual and audio events. The specific symbols used are not too important. Consistency and clarity are very important. For your convenience, we have provided example flow chart symbols in ***Appendix F, Sample Flow Chart and Level II Code***. Additional information on flow charts is also available from Multimedia Engineering/Technical Support.

The actual Level II program code is encoded on Audio Channel 2 of the videodisc as a brief series of audio tones. Each "burst" of tones lasts about 2 seconds and contains one page of data (1022 bytes). It is sometimes referred to as a Level II program “Dump”.

Caution: When planning to place program dumps one after another on the videodisc, remember to allow for sufficient space between the dumps. PVMI specifies 3 seconds (90 frames) per dump, allowing for 0.5 seconds of leader tone (30 fields), almost 2 seconds for the dump itself (approximately 100 fields) and about another 0.5 second of buffer zone. Refer to **Figure 3-C**.

Note: The vertical line below in the leader represents the target frame for the dump.

Structure of a Pioneer Level II "Dump"



Leader tone.	Level II Program Code	Buffer Zone
0.5 sec.	1.64 sec.	0.5 sec.
15 Frames	50 Frames	15 Frames

90 frames on Audio Channel 2

Figure 3-C

When a Level II videodisc is spun up on the LD-V8000 and the ***Level II Auto Start*** *On-Screen Function Switch Setting* is set to ***Load from Disc***, the player searches to frame 1, squelches the video and audio, and looks for Level II leader tone on the disc's Audio Channel 2.

When the player finds leader tone, it loads the first dump into memory page one and then begins executing the Level II program from program address zero. As the program in that first page of memory is being executed, the program can command the player to search to any other frame on the videodisc and attempt to load another page of program code (another dump). If a dump is found, it will be read and either overlay previously loaded memory or fill unused memory, as commanded by the Level II program instructions.

If the player finds no Level II leader tone at frame 1, it continues into the player's normal *Manual Mode*, to be controlled by either Level I or Level III commands. Later, one of those commands may initiate Level II operations, executing any Level II program codes that might be in the player's memory.

Preparing Level II Programs for Transmission to PVMI

When a Level II videodisc is manufactured by Pioneer Video Manufacturing, Inc. (PVMI), each Level II program dump is sent to PVMI as an "Object File" in Pioneer Level II Object Code Format. Along with comments, the target frame number, and possibly other information, the file contains a series of Hex codes, one two-character Hex code for each byte in the program dump. The file also contains a check sum value of object code at the end of the dump.

The Object File is a simple ASCII text file. Call Pioneer New Media Technologies, Inc., Engineering Support for documentation of Pioneer's Level II Object File syntax.

3.4 Player Initialization

Since different videodisc players might initialize their internal parameters to different settings, it is best to write programs that do not expect specific settings. When possible, force the settings that you want. Make sure to test any that you expect.

Some programmers will detect initial program load by finding a one in Register 0 when the program starts from program address 0. Loading a dump does not change Register 0. Concerning all other parameters, be careful.

Although the figure below shows the expected behavior of some parameters, there may exist peculiar circumstances where the expected does not occur.

Parameter	At Power On (at Run)	At Program Halt
AUDIO 1/L & 2/R	Both On (Both On)	These remain in the state they were in before HALT
Register 0 contents	1 (no change)	
Frame Display	Off	
Active Register Pointer	0 (1)	
Character Generator	Enabled	Depends on REG. A Status

Figure 3-D

4. Level II Commands for the LD-V8000

CHAPTER

4

LD-V8000

LEVEL II

USER'S MANUAL

Programmer's Reference Guide

4 Level II Commands for the LD-V8000

This section provides a detailed description of each Level II command available for use with the LD-V8000. The program commands are grouped by function:

- Program Load Control Commands
- Audio Control Commands
- Video Control Commands
- Player Control Commands
- Program Execution Commands
- Register Management Control Commands
- Input Processing Commands
- Flag Set Commands
- Transmit Commands
- Video Buffer Control Commands

(See ***Appendix B*** for an Alphabetical Listing of Level II Commands for the LD-V8000. This list also includes page numbers where detailed command descriptions are found.) The command descriptions are accompanied by a header in the corner of each page to aid in locating the commands.

Most of the Level II commands available on the LD-V8000 were available on the LD-V6000A. They may execute somewhat differently on the LD-V8000 due to command and hardware enhancements or modifications. New Level II commands are available on the LD-V8000 to better use the player's new features. These new commands are highlighted by the symbol ***NEW*** preceding the command title and description. We urge programmers to thoroughly test Level II applications developed for the LD-V8000 (or any other player) prior to disc manufacturing to assure that the program runs as intended.

4.1 Format Used to Describe Commands

Individual commands are described in the following format: There is a short ***Function*** statement; a table showing the command's argument type, Hex code, mnemonic, and RCU button(s); a detailed ***Explanation*** of the command; and usually an ***Example*** showing how the command might be used.

4.1.1 Functions

Function: This part provides a short statement describing the basic command function.

4.1.2 Table

The table indicates how a programmer refers to a command.

Argument	Hex Code	Mnemonic	RCU Button(s)
(1)	(2)	(3)	(4)

^

- **1.) The Argument**

  The command's argument type is specified, for example as an integer, register number, line number, program address, or disc location.

  Integer: A decimal number from 0 to 65535

  Disc Location: Frame number, time code (minutes and seconds), or chapter number

  An argument enclosed in brackets is optional; it can be omitted. Unless stated otherwise, the default value for an omitted argument, is zero. Sometimes, the value in the active register is used as the default argument.

- **2) The Hex Code**

  The command's two-digit hexadecimal (Hex) code is sometimes used in programming, or when code is entered using the RCU's Level II Hex Entry mode. (See page 3-5 for Hex Code Entry)

- **3) The Mnemonic**

  The command mnemonic produced is displayed on the screen by the LD-V8000 in *Programming Mode* when Level II commands are sent from the RCU. As a shorthand in examples or in programming, a command is sometimes referred to by it's mnemonic rather than it's full name.

- **4) Remote Control Unit Buttons (RCU)**

  Level II commands can be entered into memory in *Programming Mode* by pressing RCU buttons. Many frequently used commands may be input by pressing a single button. Other codes require three button presses. Any code (0 - 255, Hex 00 - FF) can be entered. (See pages 3-3 through 3-6.)

4.1.3 Explanation

Explanation: A detailed description of the command’s execution is provided.

4.1.4 Notes

Special information is sometimes presented as a **Note** to further explain player operation. However, the programmer should not depend upon this information, since it may not be true of other model players, future players, or some versions of this player.

4.1.5 Examples

A program example is often shown as an aid in understanding the use of the command in a programming sequence. The example usually includes a short statement of the program's intended function and a chart containing program address, argument, command, Hex code for the command, and a comment for each command used. Depending upon the compiler used, the source language format and the command mnemonics used in the source code may differ greatly from the code shown. These examples are intended to be instructional in nature, not examples of source code format.

Example: Play from frame 1000 to frame 1500, displaying frame numbers.

Address	Argument	Command	Hex Code		Comment
100		SFM	8E	;	Set Frame Mode
101	1	DISPLAY	F1	;	Turn Display ON
103	1000	SEARCH	F7	;	Search to frame 1000
108	1500	AUTOSTOP	F3	;	Play to frame 1500

This sample program segment just happens to start at address 100. Notice that each digit of the argument uses one byte of memory and that the command also uses one byte. The second instruction begins at program address is at 101, the next at 103, and the last at 108. If you enter this code from the RCU, you will see command mnemonics on the monitor instead of these command names.

To enter the example above using the RCU, press the following buttons: 100 PROGRAM (This puts the player into *Programming Mode* and code will be entered beginning at address 100); PLAY, 8, E (This is the three button press: PLAY prepares the player to receive Hex code, and 8E is the Hex code to SET FRAME MODE); 1 DISPLAY (1 enables DISPLAY, the DISPLAY button is a single button press sending Hex code directly to the player); 1000 SEARCH button; 1500 AUTO STOP button.

To enter this sample code from a computer, the following commands must be sent 100 *S (indicating the program address) and 64 *W (allowing up to 64 bytes of information to be written into the player’s memory). Then the following Hex code may be sent: 8E (SET FRAME MODE), 0FF1 (1 DISPLAY= DISPLAY ON), 0F3F3F3FF7 (1000 SEARCH), and 0FAF3F3FF3 (1500 AUTO STOP).

Note: When sending sample code from the computer to the player’s memory, refer to Hex code for commands included in the sample charts, and refer to ***Appendix E, Numbers and Their Hex Code Equivalents***. All arguments and commands must be in Hex code when sent from the computer via the RS-232C.

Review **Chapter 3** for details on entering Level II Code into RAM via RCU, RS-232C or from a Level II videodisc.

4.2 Level II Command Descriptions

The following are descriptions of the Level II commands available on the LD-V8000.

4.2.1 Program Load Control Commands

The PAGE command is used to set the size of active memory. The other four commands described in this section are used to load program dumps from a Level II videodisc. In general, they specify that 1022 (or fewer) bytes are to be loaded into a particular page of the player's memory. The commands are: LOAD (loads page zero only), PLOAD (Partial Load, for page zero only), MLOAD (Moving Load, for any page), and MPLOAD (Moving Partial Load, for any page).

1) PAGE

Function: The Page command sets the active memory size, from 1 to 7 pages active.

Argument	Hex Code	Mnemonic	RCU Buttons
Integer	11	PAG	Play, 1, 1

Explanation: The PAGE command sets the current size of the Program Area. Up to seven pages (1022 bytes each) of program area can be selected. Specify the number of pages minus one in the argument. If N PAGE (0 ≤ N ≤ 6) is executed, an area of (N + 1) pages is made active.

As explained earlier in ***Section 2.4, Random Access Memory***, an increase or decrease in the size of the active program area causes a change in the correspondence between register numbers and program addresses. When the PAGE command is used, it is recommended that it be executed just once, before most other significant code.

After power-on or a REJECT, the active memory size is one page. After the initial program load, Register 0 contains a 1 and Registers 1, 2, ... contain "data" loaded from the disc. Then, if a 6 PAGE is executed, Registers 1, 2, ... "move". They then "contain" data left in RAM from previous use of the player. *This feature may be useful, but must be carefully understood for proper programming.*

Note: Execution of the PAGE command without an argument makes one page active. If the argument exceeds six, an argument of six is assumed.

Number of Active Pages	Active Memory Size	Number of Registers	Specified Command
1	1024 Bytes (1022 bytes, and 2 bytes for Reg 0)	512 (511 Regs and Reg 0)	POWER ON 0 PAGE
2	2046	1023	1 PAGE
3	3068	1534	2 PAGE
4	4090	2045	3 PAGE
5	5112	2556	4 PAGE
6	6134	3067	5 PAGE
7	7156	3578	6 PAGE

2) LOAD (Load information from disc)

Function: This command loads a 1022-byte dump from the videodisc into Page 0.

Argument	Hex Code	Mnemoni	RCU Buttons
Not Allowed	CC	L	Play, C, C

Explanation: An argument is not permitted — see MLOAD. Video and both audio channels are temporarily squelched, the current frame is used as the dump's "target frame", and the player tries to detect dump leader tone on Audio Channel 2. It is not required that the user have Audio 2 "ON".

If leader tone is detected, the player reads 1022-bytes of dump data from the disc into program memory Page 0. It does not affect the contents of Register 0 or the Active Register Pointer. Thus, Register 0 may be used for passing data between different program "overlay" segments.

There are no prohibited byte values in a dump. Any of the Hex codes ('00' through 'FF') may be loaded into RAM via a program dump.

LOAD (cont.); MLOAD

Program dump data is stored as a specially constructed "tone" on Audio Channel 2. Data bits are recorded at 5000 bits per second and span approximately 50 frames of Audio Channel 2. The combined leader tone, data, and buffer zone may occupy 2 to 3 seconds of Audio Channel 2. Thus, approximately 2 to 3 seconds are required to load a 1022-byte "page" of data.

Information regarding the recommended positioning and spacing of multiple program dumps may be obtained from Pioneer Video Manufacturing, Inc. Also, refer to **Section 3.3, *Level II Programs Encoded on Videodisc*** on page 3-10 of this manual.

Upon successful completion of the load operation, the audio and video squelch status are returned to the state they were in before the load was executed. Then, program execution continues at program address 0.

If the load is unsuccessful (for example, a checksum error), the player retries the load operation up to eight times. If the load operation still fails, or no leader tone was detected at the target frame, the player returns to *Manual Mode* operation.

Note: The player may not properly process commands received while it is loading a program dump from a videodisc.

Example: A dump that has it’s target frame (covered by leader tone) at frame 450 is loaded into Page 0 and the program execution is to resume at Address 0. The player searches to frame 450, finds the leader tone, and then loads the data. Since there is no argument before the Load command, the player will automatically resume executing program instructions at program address 0. Also show a program segment which passes a parameter (the value "13") in Register 0 to the loaded program.

Address	Argument	Command	Hex Code
100	450	SEARCH	F7
104		LOAD	CC
200		VOFF	1C
201	13	GET	08
204	100	BRANCH	CF

3) MLOAD (Moving Load)

Function: This command loads a 1022-byte dump into the indicated page.

Argument	Hex Code	Mnemonic	RCU Buttons
Integer	CC	L	Play, C, C

Explanation: One or several dumps can be loaded while a motion sequence is playing. In effect, loading the program is hidden under a motion segment, providing an “invisible load”. Like LOAD, the MLOAD command loads a 1022-byte program dump from the videodisc. However, the data may be put into any of the currently active pages of memory. Thus, the command **must** have an argument specifying the page (0 - 6) into which the "dump" information will be loaded. The command does not affect the contents of Register 0 or the Active Register Pointer.

When the MLOAD command is executed, AUDIO 2 is temporarily muted. Valid program leader must be detected for the MLOAD to continue. It is not necessary for AUDIO 2 to be turned "ON". While “listening” for data, the player is forced to PLAY forward if it is not already doing so. The VIDEO and AUDIO 1 squelch status are not changed. Thus, dumps could be loaded while a motion sequence is playing. After the MLOAD, the player reverts to the mode it was in before the MLOAD was executed. Program execution continues at the program address immediately following the MLOAD command. Thus, an MLOAD would usually be used to load data only into another page, not the page that is currently executing code.

If dump leader is not detected, or the load is not successfully completed, the player does not retry the MLOAD. Branch on Failure (BRF) is used to detect the failure.

Note: The argument is taken modulo 256. The command is ignored if the argument specifies a page that is not active. The player may not properly process commands received while it is loading a program dump from a videodisc.

Example: Load the dump at frame 100 into Page 3 (of pages 0 - 6). “Invisible Load” the dump at frame 450 into Page 2 while playing from 450 to 600. In both cases, if loading fails, Branch to program address 200.

Address	Argument	Command	Hex Code
50		VOFF	1C
51		AFF	A0
52	100	SEARCH	F7
56	3	MLOAD	CC
58		STOP	FB
59	200	BRF	07
100	450	SEARCH	F7
104		PLAY	FD
105	2	MLOAD	CC
107	200	BRF	07
111	600	AUTOSTOP	F3

4) PLOAD (Partial Load)

Function: This command loads a specially prepared "partial dump" into Page 0.

Argument	Hex Code	Mnemonic	RCU Buttons
Not Allowed	0C	PLD	Play, 0, C

Explanation: The command gives the ability to pass more data from one overlay of Page 0 to the next. Since the LD-V8000 has multiple pages of memory, this command is rarely used in programming the LD-V8000.

Like the LOAD command, the PLOAD command loads program data from the videodisc into program memory Page 0, an argument is not allowed, and there are no prohibited byte values. When the command is executed, valid leader must be detected on Audio Channel 2, which does not need to be "ON".

Unlike LOAD, PLOAD can load less than 1022 bytes into Page 0. The number of bytes to be loaded is calculated from the value stored in Register 0, (a value of 2 through 1023). If Register 0 contains "N", then (1024 - N) bytes of data are loaded. The data is loaded starting at Address (1023 - N), proceeding down to Address 0. The remaining (N - 2) bytes in Page 0 are not rewritten. Thus, new program code or data can be loaded into the low-address part of Page 0 without changing the information stored in the high-address end.

CAUTION: Do not use a value of N that is less than 2 or greater than 1023.

In execution, the PLOAD command is almost the same as the LOAD command. The player reverts to *Manual Mode* if the PLOAD does not successfully complete the load operation. After a PLOAD is completed, the instruction at program address 0 is executed.

Example: The dump leader starts before frame 950, the dump data starts after frame 950. The user wishes to overlay the first 500 bytes of page 0 (addresses 0 - 499) and resume program execution at program address 0. Page 0 addresses 500 through 1021, and Register 0, remain unchanged.

Address	Argument	Command	Hex Code	Comment
300	524	GET	08	Place 524 into R 0
304	950	SEARCH	F7	Search to frame 950
308		PLOAD	0C	Load 500 Bytes (1024 - 524)

5) MPLOAD (Moving Partial Load)

Function: This command loads a special "partial dump" into any active page.

Argument	Hex Code	Mnemonic	RCU Buttons
Integer	0C	PLD	Play, 0, C

Explanation: This command loads a partial page of information into the indicated active page. Since the LD-V8000 has multiple pages of memory, this command is rarely used in programming the LD-V8000. However, the command might possibly be used to load small amounts of data quickly.

Like the MLOAD command, the MPLOAD command loads information from the videodisc into any active page, an argument (0 - 6) is required (see MLOAD), and there are no prohibited byte values. Neither Register 0 nor the Active Register Pointer is changed by any load.

Like PLOAD, MPLOAD can load less than 1022 bytes and the number of bytes is calculated from the value stored in Register 0, (a value of 2 through 1023). If Register 0 contains "N", then (1024 - N) bytes of data are loaded. The data is loaded starting at the (1024 - N)th byte of the page, proceeding down to the first byte of the page. The remaining (N - 2) bytes at the high address end of the page are not rewritten.

CAUTION: Do not use a value of N that is less than 2 or greater than 1023.

In execution, the MPLOAD command is almost the same as the MLOAD command, except that the shorter load may finish faster. As with MLOAD, valid leader must be detected on Audio Channel 2, which does not need to be "ON". The load operation is not automatically retried, even if it fails. After the MPLOAD command, the next sequential program instruction is executed regardless of whether the MPLOAD command was successful.

Note: The argument is taken modulo 256. The MPLOAD command is ignored if the argument does not specify a currently active page.

MPLOAD (cont.)

Example #1: Load a "604-byte" partial dump at frame 2500 into Page 2.

Address	Argument	Command	Hex Code	Comment
100	420	GET	08	Place 420 into Register 0
104		VOFF	1C	Turn Video OFF
105		AFF	A0	and Audios OFF
105	2500	SEARCH	F7	Search to frame 2500
110	2	MPLOAD	0C	Load into page 2
112		STOP		Force Still mode

Example #2: Overlay the first 500 bytes of Page 1. Retry until successful.

Address	Argument	Command	Hex Code	Comment
598		VOFF	1C	Turn Video OFF
599		AFF	A0	and Audios OFF
600	524	GET	08	Place 524 in R0
604	450	SEARCH	F7	Search to dump Target frame
608	1	MPLOAD	0C	Load 500 bytes (1024 - 524)
610	604	BRF	07	Retry, if load failed.
614	1000	SEARCH	F7	Perhaps show a menu ...
619		VON	1B	Turn Video back ON

4.2.2 Audio Control Commands

The audio control commands set the switches and status registers that pass or block the disc's audio information.

6 & 7) AUDIO 1 and AUDIO 2

Function: These commands can be used to control the Audio Channel Select switches.

Argument	Hex Code	Mnemonic	RCU Button
[Integer]	F4	A1	AUDIO 1/L
[Integer]	FC	A2	AUDIO 2/R

Explanation: The AUDIO 1 command opens, closes, or toggles Audio Channel Select Switches 1 and 3 (see **Figure 4-1** below). The AUDIO2 command opens, closes, or toggles Switches 2 and 4. The affected switches are both opened (turned OFF) if the argument is 0. The switches are both closed (turned ON) if the argument is 1. The switches are both turned ON (closed) at power on and program RUN.

Also, the player automatically closes (turns ON) Audio Switch 5 whenever only one of Audio Switches 1 or 2 is closed (ON), otherwise, Switch 5 is open (OFF). Likewise, Audio Switch 6 is automatically closed (ON) whenever only Audio Switch 3 or 4 is closed (ON), otherwise, it is open (OFF).

If the command is executed without an argument, it toggles the controlled switches. In other words, the switches are opened (OFF) if they were closed (ON), and closed (ON) if they were open (OFF). **Note:** An odd argument is equivalent to a "1" and an even argument is equivalent to a "0".

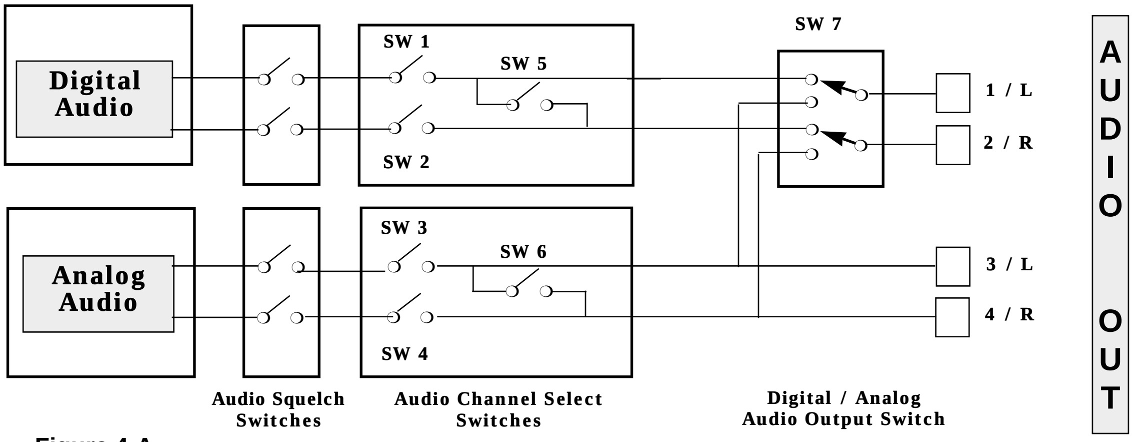


Figure 4-A

AUDIO 1 & AUDIO 2 (cont.)

Example: Play from frame 1000 to 3000 with Audio 1 OFF and Audio 2 ON. Play from frame 3000 to 5000 with Audio 1 ON and Audio 2 OFF.*

Address	Argument	Command	Hex Code	Comment
0013	0	AUDIO1	F4	Turn off Audio 1
0015	1	AUDIO2	FC	Turn on Audio 2
0017	1000	SEARCH	F7	Search to frame 1000
0022	3000	AUTOSTOP	F3	Play to frame 3000
0027		AUDIO1	F4	Toggle Audio 1 (ON)*
0028		AUDIO2	FC	Toggle Audio 2 (OFF)•
0029	5000	AUTOSTOP	F3	Play to frame 5000

*This would actually be poor programming practice if the programmer intended to **force** AUDIO 1 ON and AUDIO 2 OFF. The viewer might have toggled either audio channel during the AUTOSTOP from 1000 to 3000.

8) AXX (Audio Channel Select)

Function: These single-byte commands can be used to efficiently control the Audio Channel Select switches.

Argument	Hex Code	Mnemonic	Audio Ch 1/L	Audio Ch 2/R
(Ignored)	A0	AFF	OFF	OFF
(Ignored)	A1	AFN	OFF	ON
(Ignored)	A2	ANF	ON	OFF
(Ignored)	A3	ANN	ON	ON
(Ignored)	A4	AFT	OFF	Toggled
(Ignored)	A5	AFI	OFF	"Ignored"
(Ignored)	A6	ANT	ON	Toggled
(Ignored)	A7	ANI	ON	"Ignored"
(Ignored)	A8	ATF	Toggled	OFF
(Ignored)	A9	ATN	Toggled	ON
(Ignored)	AA	AIF	"Ignored"	OFF
(Ignored)	AB	AIN	"Ignored"	ON
(Ignored)	AC	ATT	Toggled	Toggled
(Ignored)	AD	ATI	Toggled	"Ignored"
(Ignored)	AE	AIT	"Ignored"	Toggled

F = OFF = Open N = ON = Closed T = Toggled I = "Ignored" = no change

Explanation: The Audio Channel Select switches can be turned ON (N), turned OFF (F), toggled (T), or ignored (I) — left alone. Audio Channel 1/L is activated by closing switches 1 and 3. Audio Channel 2/R is activated by closing switches 2 and 4. The player automatically controls switches 5 and 6.

Note: Even though any arguments are ignored, it is preferred that they be omitted.

Example: Frames 1000 to 1200 are played with both AUDIO 1 and AUDIO 2 ON. Frames 1200 to 1400 are played with AUDIO 1 ON and AUDIO 2 OFF. Frames 1400 to 1600 are played with AUDIO 1 OFF and AUDIO 2 ON. Frames 1600 to 1800 are played with both AUDIO 1 and AUDIO 2 OFF.

Address	Argument	Command	Hex Code	Comment
100		SFM	8E	Frame Mode
101	1000	SEARCH	F7	Search to frame 1000
106		ANN	A3	Audio 1/L and Audio 2/R ON
107	1200	AUTOSTOP	F3	Stop at frame 1200
112	1	AUDIO1	F4	Audio 1/L ON
114	0	AUDIO2	FC	Audio 2/R OFF
116	1400	AUTOSTOP	F3	Stop at frame 1400
121		AFN	A1	Audio 1/L ON and Audio 2/R OFF
122	1600	AUTOSTOP	F3	Stop at frame 1600
127		AFF	A0	Audio 1/L and Audio 2/R OFF
128	1800	AUTOSTOP	F3	Stop at frame 1800

9) DAD (Digital Audio)

Function: This command selects Digital Audio Output or Analog Audio Output.

Argument	Hex Code	Mnemonic	RCU Buttons
[Integer]	82	DAD	Play, 8, 2

Explanation: When a videodisc has Digital Audio encoded on it, the player will normally connect the Digital Audio channels to audio outputs 1/L and 2/R (instead of the disc's Analog Audio channel). When playing a Digital Audio disc, either the Digital Audio channels or the Analog Audio channels can be connected there, by controlling Switch 7 (see SW 7 in **Figure 4-A**). The Analog Audio channels are always connected to Audio Outputs 3/L and 4/R.

With this command, an argument of 0 selects Analog Audio output on 1/L and 2/R. An argument of 1 selects Digital Audio output there (if Digital Audio is present on the videodisc). Switch 7 is just toggled if there is no argument. However, if the videodisc does not have Digital Audio, then Analog Audio is output regardless of the attempt to select Digital Audio with the DAD command.

Example: On a four-language disc, encoded with different languages on each of two independent Analog Audio channels and on each of two independent Digital Audio channels, play a sequence (frames 1050 through 1836) four times. Each time the sequence is played, accompany it with a different language output on Audio Output 1/L.

Address	Argument	Command	Hex Code	Comment
200		SFM	8E	Frame Mode
201		ANF	A2	Select Left Channel
202	1	DAD	82	Select Digital Audio
204	1050	SEARCH	F7	Search to frame 1050
209	1836	AUTOSTOP	F3	AutoStop at frame 1836
214		AFN	A1	Select Right Channel (still Digital)
215	1050	SEARCH	F7	Search to frame 1050
220	1836	AUTOSTOP	F3	AutoStop at frame 1836
225	0	DAD	82	Select Analog Audio (still Right Channel)
227	1050	SEARCH	F7	Search to frame 1050
232	1836	AUTOSTOP	F3	AutoStop at frame 1836
237		ANF	A2	Select Left Channel (still Analog)
238	1050	SEARCH	F7	Search to frame 1050
243	1836	AUTOSTOP	F3	AutoStop at frame 1836

4.2.3 Video Control Commands

These commands control the selection of the video source that provides the player's output. They also control the operation of the player's internal character generator and allow specification of the information it overlays on the video signal. The video control commands set the switches and registers that display, mute, squelch, and overlay the video information read from the videodisc. In *Manual Mode*, these commands may be executed with or without arguments, but in *Interrupt Mode* any arguments are ignored.

All video information from the disc is either ignored or written into a digital video buffer inside the player. The player's output video always comes from one of two sources: The Video Buffer or a Blue/Black Squelch Generator. Either may be overlayed with text from the player’s Text Overlay Generator. The typical power-on configuration allows the disc's video to be written into the video buffer, with the video buffer providing the player’s video output. During a SEARCH, when video would be squelched, the last information in the video buffer is used to generate a “still” while the laser read head is moving.

10) VOFF (Video Off)

Function: The player's Video Output is squelched to Blue or Black.

Argument	Hex Code	Mnemonic	RCU Buttons
[Ignored]	1C	VFF	Play, 1, C

Explanation: This command substitutes the player's internally generated video Blue (or Black) background for the normal video output (from the Video Buffer). Thus, it blocks the buffered video image (captured from the videodisc) from being displayed on the screen, providing, instead, an entire screen of blue (or black). Characters generated by the character generator (if it is enabled) can still be seen superimposed on the squelch screen.

11) VON (Video On)

Function: Video output previously “squelched” by VOFF is turned back on.

Explanation: This command routes the output of the Video Buffer to the player's Video Output terminals. If the Blue or Black “squelch video" was being output, that video will be replaced by whatever image is stored in the video buffer. Character generator overlay, if any, continues to be seen. The player's power-on default is Video ON.

Argument	Hex Code	Mnemonic	RCU Buttons
[Ignored]	1B	VON	Play, 1, B

Example: Wait for the user to press button "0". Then, squelch the video during a Search so that the viewer gets an immediate "response" to the input.

Address	Argument	Command	Hex Code	Comment
0	1	INPUT	F8	Wait for Input from User
2		VOFF	1C	Button 0 pressed, squelch video
3	1000	SEARCH	F7	Search to Frame 1000
8		VON	1B	Turn video back ON
9	14	BRANCH	CF	Continue program at 14
12	0	BRANCH	CF	Button 1-9, ignored

12 & 13) CGE and CGD (Character Generator Enable and Disable)

Function: These commands enable or disable the overlay of character generator output on the player's Video Output signal.

Argument	Hex Code	Mnemonic	RCU Buttons
[Ignored]	E0	CGE	PLAY, E, 0
[Ignored]	E1	CGD	PLAY, E, 1

Explanation: The CGE command permits the overlay of characters produced by the player's internal character generator on the video being routed to the player's Video Output. The CGD command inhibits the overlay process of even the Frame Number Display information. At program RUN, the character generator overlay is enabled, but usually there is no text being generated.

The internal character generator is used to display a variety of information. While in *Automatic Mode*, the following information can be displayed:

- Frame or Time Number in the upper left corner, on line 0.
- Chapter Number in the upper left corner, on line 0.
- A user message on any of lines 0 through 11, of 20 characters each.

Note: Descriptive names used in this section differ from the names used in explaining the external control of the player. CGE is equivalent to the external “DISPLAY ON” and CGD is equivalent to the external “DISPLAY OFF”.

14) DISPLAY

Function: This command controls the generation of a Disc Location display.

Argument	Hex Code	Mnemonic	RCU Button
[Integer]	F1	DI	DISPLAY

Explanation: The player's internal character generator can produce a Disc Location display on line 0 of its overlay. On a CAV disc, the five-digit Frame Number is generated. On a CLV disc, the Time Code 3-, 5-, 0r 7-digit time number is generated — Hours, Minutes, Seconds, and Frame numbers, depending upon the encoding of the disc. If a CAV or CLV disc is encoded with chapters, the two digit Chapter Numbers will also be generated preceeding the frame or time number.. The character generator must be "enabled" for character overlay or the Disc Location display will not be visible.

DISPLAY (cont.)

An argument of 0 turns OFF the generation of the Disc Location display. An argument of 1 turns ON the generator. If there is no argument, DISPLAY simply toggles the On/Off state of the Disc Location display generator. At program RUN, the Disc Location display generator is forced OFF.

Note: The terminology used in this section differs from that used in the description of external player control. This DISPLAY control is equivalent to externally setting Register A to 7 for Display ON and to 4 for Display OFF.

Note: An odd argument is equivalent to a 1, and an even argument is equivalent to a 0. During program execution (in *Interrupt Mode*), the user may toggle this Disc Location display ON or OFF at any time, usually undetected by the program. The CGD command turns off all display from the character generator, regardless of the Disc Location display's ON or OFF status.

Example #1: Display the Disc Location information for one second as the sequence from 1000 to 2000 begins playing.

Address	Argument	Command	Hex Code	Comment
000	1000	SEARCH	F7	Search to Frame 1000
005	1	DISPLAY	F1	Disc Location display ON
007	10	WAIT	FB	Wait for one second
010	0	DISPLAY	F1	Disc Location display OFF
012	2000	AUTOSTOP	F3	Finish the sequence

Example #2: Display frame numbers during the first video sequence.

Address	Argument	Command	Hex Code	Comment
0050	1000	SEARCH	F7	Search to Frame 1000
0055		CLD	2C	Clear all lines of the Character Generator
0056		CGE	E0	Force Character Generator overlay ON
0057	1	DISPLAY	F1	Turn ON "Frame display" generation
0059	1500	AUTOSTOP	F3	Play to Frame 1500 with Display ON
0064	0	DISPLAY	F1	Force Disc Location generation OFF
0066	2000	AUTOSTOP	F3	Continue to Frame 2000

15) SUD (Set User Display)

Function: This command loads data into one line of the character generator.

Argument	Hex Code	Mnemonic	RCU Buttons
[Line Number]	2B	SUD	Play, 2, B

Explanation: When enabled, the character generator overlays its lines of text on the output video. Each line (lines 0 - 11 in the LD-V8000 player) contains twenty "ASCII" characters. An argument of 0 through 11 is used to specify the line to be loaded. The SUD command reads and loads the 20 bytes of data starting at the program address stored in the Active Register. After loading is complete, the active register pointer is incremented by one.

Note: The topmost overlay line is line 0 and the bottommost line is line 11. Since other players access different lines, use care in selecting lines if compatibility is desired. The command is ignored if the argument is other than 0 through 11. The 20 bytes of data to be loaded should be byte values in the range of '20' through '9F'. Individual characters do not blink. Refer to ***Appendix D, Character Generator: Table of Hex Codes.***

16) CLD (Clear Display)

Function: This command erases one (or all) character generator lines.

Argument	Hex Code	Mnemonic	RCU Buttons
[Line Number]	2C	CLD	Play, 2, C

Explanation: The argument (0 - 11) selects one line of the character generator's display. The data stored in the character generator for that line is "erased" (that display line is "cleared"). If there is no argument, all of the character generator lines are cleared.

17) BLINK

Function: All the characters on the selected line start to blink.

Argument	Hex Code	Mnemonic	RCU Buttons
[Line Number]	2D	BLK	Play, 2, D

Explanation: The character generator is directed to begin blinking all of the characters on a line (0 - 11) specified by the argument. All lines blink if the argument is missing.

18) CLB (Clear Blink)

Function: All the characters on the selected line cease blinking.

Argument	Hex Code	Mnemonic	RCU Buttons
[Line Number]	2E	CLB	Play, 2, E

Explanation: The character generator is directed to stop blinking all of the characters on a line (0 - 11) specified by the argument. All lines stop if the argument is missing.

Example: "HELLO there." is overlayed on line 7 for 5 seconds. Then, the line blinks for 3 seconds and the overlay is cleared. After 4 more seconds, the cycle repeats.

```
  Address   Argument    Command   Hex Code          Comment
   0100        200          GET       08
   0104         10          PUT       09
   0107         10       RECALL       7F
   0110          7          SUD       2B      Display the 20 character message
   0112         50         WAIT       FB         and Wait for 5 seconds
   0115          7        BLINK       2D      Start blinking
   0117         30         WAIT       FB         for 3 seconds
   0120                     CLD       2C      Then clear all lines
   0121                     CLB       2E         and all blinking
   0122         40         WAIT       FB      Wait for 4 seconds
   0125        107       BRANCH       CF         and repeat
    ...
   0200     *20   *20   *20   *20   *48       4 spaces and "H"
   0205     *45   *4C   *4C   *4F   *20       "ELLO" and 1 space
   0210     *74   *68   *65   *72   *65       "there"        (data representation will vary)
   0215     *2E   *20   *20   *20   *20       period and 4 spaces
```

19) SBC (Set Background Color)

Function: This command selects the color for the video squelch generator.

Argument	Hex Code	Mnemonic	RCU Buttons
[Integer]	88	SBC	Play, 8, 8

Explanation: This command selects Blue or Black as the "background" color used for the output video signal when the normal video is “squelched”. Blue is selected when the argument is 1 and Black is selected when the argument is 0. The background color toggles between Blue and Black if an argument is not supplied.

Note: An odd argument is equivalent to a 1, and an even argument is equivalent to a 0.

Example: The player's video is squelched and and the "background" screen color is changed from blue to black and back every 6 seconds.

Address	Argument	Command	Hex Code	Comment
0005	1	SBC	88	Blue selected
0007		VOFF	1C	Video squelched
0008	30	WAIT	FB	and Wait for three seconds
0011	0	SBC	88	Black selected
0013	30	WAIT	FB	and Wait for three seconds
0016	5	BRANCH	CF	Branch to repeat

4.2.4 Player Control Commands

These commands control video playback by specifying how the player accesses the information on the videodisc.

20) REJECT

Function: REJECT stops disc rotation and returns the player to the PARK position.

Argument	Hex Code	Mnemonic	RCU Button
[Ignored]	F9	RJ	Play, F, 9

Explanation: REJECT forces the player to stop disc rotation and enter *Park Mode*. It also terminates *Automatic Mode*, returning the player to *Manual Mode*. In addition, Video, Audio1, and Audio 2 are turned ON; Disc Location display generation is turned OFF; the Character Generator is Enabled; Multi-Speed is set to 1/4 speed; the Active Register Pointer is set to 0; Register 0 is set to 1; Active Memory Size is set to one page; and the RCU is enabled.

Note: A programmed Level II REJECT command does not open the disc drawer.

21) PLAY

Function: The videodisc is played at normal speed.

Argument	Hex Code	Mnemonic	RCU Buttons
[Disc Location]	FD	P	Play, F, D

Explanation: When there is no argument, the videodisc starts playing forward at normal speed from the current location. Then, the next command is executed. When there is an argument, the player proceeds forward or reverse as necessary to reach the specified disc location (frame, time numbers, or chapter). The next command is executed when the specified disc location is reached. When the disc is playing in reverse, audio is automatically squelched.

Note: If Level II programs attempt to play into leadout, the player immediately stops and re-positions itself on the last frame before leadout. Other players may not properly execute the PLAY command with an argument.

Example: Play from Frame 1000 forward at normal speed with AUDIO 1 and 2 ON for five seconds. With AUDIO 2 OFF, continue playing for five more seconds. Finally, play backward to Frame 1000.

Address	Argument	Command	Hex Code	Comment
100		SFM	8E	Frame Mode
101	1	DISPLAY	F1	Disc Location Display ON
103	1000	SEARCH	F7	Search to Frame 1000
108		ANN	A3	Audio 1 and 2 ON
109		PLAY	FD	Normal Play
110	50	PAUSE	0D	for 5 seconds
113	0	AUDIO2	FC	Turn Audio 2 OFF, playing continues
115	50	PAUSE	0D	and Wait for 5 more seconds
118	1000	PLAY	FD	Play backward to Frame 1000

22) AUTOSTOP

Function: The player plays forward at normal speed to a specific Disc Location.

Argument	Hex Code	Mnemonic	RCU Button
[Disc Location]	F3	AS	AUTOSTOP

Explanation: If an argument is not specified, the value in the Active Register is used as an argument. The argument is a "target" disc location (frame, time code, or chapter number), which is compared to the current disc location. If the target is ahead, the player plays at normal speed to the target and stops there, in still mode. If it is behind, the player just searches to the target. Then, the Active Register Pointer is incremented by one (even when there is an explicit argument). The next command is executed after the target is reached.

NOTE: In Level II, an Autostop takes precedent over any picture stops encoded on the videodisc.

Example: Play frames 2000 to 3000, delay five seconds, and play the next 1500 frames. Assume: R50 = 2000 and R51 = 3000.

Address	Argument	Command	Hex Code	Comment
0010	50	RECALL	7F	Activate Register 50
0013		SEARCH	F7	Search to Frame 2000, activate R51
0014		AUTOSTOP	F3	Play to Frame 3000, activate R52
0015	50	WAIT	FB	and wait five seconds
0018	4500	AUTOSTOP	F3	Play to Frame 4500, activate R53

23) SEARCH

Function: This command is used for high speed access to the specified disc location.

Argument	Hex Code	Mnemonic	RCU Button
[Disc Location]	F7	SC	SEARCH

Explanation: If an argument is not specified, the value in the Active Register is used as an argument. The argument is a "target" disc location (frame, time code, or chapter number), and the player performs a high speed search for the target. Then, the Active Register Pointer is incremented by one (even when there is an explicit argument). The next command is executed after the search is finished. Search sets the Success / Fail Flag.

Note: During a search, good playback video from the disc itself is not available, so the Video Buffer usually acts as the video output source during the search. Since the Video Buffer contains the last "picture" seen before the search, the user may not be aware that a search is in progress.

Example: Assume Register 20 contains the value 1500. Search to Frame 1500 three different ways. (First enter the following code: 20 RECALL 1500 STORE END CLEAR)

Address	Argument	Command	Hex Code	Comment
100	1500	SEARCH	F7	Explicit argument
200	20	RECALL	7F	
203		SEARCH	F7	Use value in Active Register
300	20 ARG	SEARCH	F7	Use value from Register 20

24) WAIT

Function: WAIT forces a still frame and then delays execution for a period of time.

Argument	Hex Code	Mnemonic	RCU Button
[Integer]	FB	W	STOP

Explanation: The command forces any motion to stop (*Still Frame Mode)* and then the player waits for the designated time before executing the next instruction. The argument specifies the delay in tenths of a second. No argument means no delay.

Note: The maximum delay is 1000 seconds (10,000 tenth seconds, or 10000 WAIT).

Example: The 52 WAIT instruction will force a still frame and delay 5.2 seconds.

25) PAUSE

Function: The PAUSE command just delays program execution for a period of time.

Argument	Hex Code	Mnemonic	RCU Buttons
[Integer]	0D	PAU	Play, 0, D

Explanation: The command tells the player to wait for the designated time before executing the next instruction. Motion in progress may continue. The argument specifies the delay in tenths of a second. No argument means no delay.

Note: The maximum delay is 1000 seconds (10,000 tenth seconds, or 10000 PAUSE).

Example: Play the first 13.5 seconds of material after Frame 1000 and stop.

Address	Argument	Command	Hex Code	Comment
0100	1000	SEARCH	F7	Go to Frame 1000
0105		PLAY	FD	and begin playing the disc
0106	135	PAUSE	0D	Delay program execution for 13.5 sec.
0110		WAIT	FB	and Still Mode at the current frame

26) SLOW (Slow Speed)

Function: SLOW selects a "slow" speed to be used by the MULTI-SPEED commands.

Argument	Hex Code	Mnemonic	RCU Button
[Integer]	ED	SS	SLOW

Explanation: This command selects one of the several predefined "slow" playback speeds to be used by the next MULTI-SPEED FWD or MULTI-SPEED REV. It usually selects a slower than normal speed. An argument from 1 to 127 selects a speed from the following table.

Note: A missing or zero argument will select "zero" speed. Then, a following MSF or MSR command will be ignored. The argument is taken modulo 128.

The relationship between the SLOW argument and the selected speed is shown below:

Argument	Speed	Fr / Sec	Argument	Speed	Fr / Sec
1	1 x	30	2	1/2 x	15
3	1/3 x	10	4	1/4 x	7.5
5	1/5 x	6	6	1/6 x	5
7	4/30 x	4	8	7/60 x	3.5
9 - 10	3/30 x	3	11 - 12	5/60 x	2.5
13 - 15	2/30 x	2	16 - 20	3/60 x	1.5
21 - 30	1/30 x	1	31 - 127	1/60 x	0.5

27) FAST (Fast Speed)

Function: FAST selects a "fast" speed to be used by the MULTI-SPEED commands.

Argument	Hex Code	Mnemonic	RCU Button
[Integer]	EC	FS	FAST

Explanation: This command selects one of the several pre-defined "fast" playback speeds to be used by the next MULTI-SPEED FWD or MULTI-SPEED REV Command. It usually selects a faster than normal speed. An argument from 1 to 3 selects a speed from the following table.

Note: A missing or zero argument will select "zero" speed. Then, a following MSF or MSR command will be ignored. If an argument over 3 is used, the selected speed will be three times normal speed.

The relationship between the FAST argument and the selected speed is shown below.

Argument	Speed	Fr / Sec
1	1 x	30
2	2 x	60
3	3 x	90

28 & 29) MSF and MSR (Multi-Speed Forward and Multi-Speed Reverse)

Function: Forward or Reverse silent motion video is produced, at a selected speed. The player enters *Still Mode* when the "target" disc location is reached.

Explanation: The argument is a "target" disc location (frame, time code, or chapter number), which is compared to the current disc location. If the direction required to reach the target is compatible with the command's direction, play begins at the most recently selected speed ("slow" or "fast"). Audio is squelched while the Multi-Speed motion is in progress. When the player reaches the target, the player enters *Still Mode* at the target and then the next command is executed.

If the player is instructed to play in the wrong direction to reach the target, the player will just search to the target.

Note: The command will be ignored unless it has an argument and the selected speed is non-zero. It is best to select the desired speed with SLOW or FAST rather than rely upon any default speed.

Argument	Hex Code	Mnemonic	RCU Button
Disc Location	F2	MF	MULTI-FWD
Disc Location	FA	MR	MULTI-REV

Example: Frames 1000 to 1200 are played at one-half speed. Reverse play, returning to Frame 1000, is at triple speed.

Address	Argument	Command	Hex Code	Comment
0100		SFM	8E	Frame Mode
0101	1000	SEARCH	F7	Search to Frame 1000
0106	2	SLOW	ED	Set 1/2 speed
0108	1200	MSF	F2	Multi-speed forward to Frame 1200
0113	3	FAST	EC	Set triple speed
0115	1000	MSR	FA	Multi-speed reverse to Frame 1000

Example: Play Frame 100 to Frame 500 at twice normal speed.

Address	Argument	Command	Hex Code	Comment
0000	2	FAST	EC	Select two times normal speed
0002	100	SEARCH	F7	Search to Frame 100
0006	500	MSF	F2	Plays to Frame 500 at 2 x speed

30 & 31) STEP F and STEP R (Step Forward and Step Reverse)

Function: These commands force *Still Mode* and step forward or backward one frame.

Argument	Hex Code	Mnemonic	RCU Button
[Ignored]	F6	SF	STEP FWD
[Ignored]	FE	SR	STEP REV

Explanation: Both commands force the player into *Still Mode* if it is not already there. In Still mode, each STEP F command will advance the player to the next video frame, and each STEP R will access the preceding frame on the videodisc. These commands can be used in a programmed loop to display a series of still frames or to provide programmed slow-motion effects.

NOTE: For maximum compatibility, the argument should be omitted. **CAUTION:** The time required for the execution of these commands has changed.

Example #1: From Frame 1000 go forward 2 frames and return, at 1 frame/second.

Address	Argument	Command	Hex Code	Comment
0100	1000	SEARCH	F7	Search to Frame 1000
0105	10	WAIT	FB	and Wait 1 second
0108		STEP F	F6	Step Forward, to 1001
0109	10	WAIT	FB	and Wait 1 second
0112		STEP F	F6	Step Forward, to 1002
0113	10	WAIT	FB	and Wait 1 second
0116		STEP R	FE	Step Reverse, to 1001
0117	10	WAIT	FB	and Wait 1 second
0120		STEP R	FE	Step Reverse, to 1000

Example #2: Beginning at the current frame, display the next ten frames as a series of 8-second stills, then return to a menu at frame 2000. (First enter: 20 RECALL, 10 STORE END CLEAR.)

Address	Argument	Command	Hex Code	Comment
0053	10	GET	08	Place the value 10 in Register 0
0056		STEPF	F6	Step to the next frame
0057	80	WAIT	FB	and wait 8 seconds
0060	0	DECREG	F0	Decrement Register 0, and test it for 0
0062	56	BRANCH	CF	R0 > 0: Continue at address 56
0065	2000	SEARCH	F7	R0 = 0: Show the menu

32) SFM (Set Frame Mode)

Function: This command puts the player into *Frame Mode*.

Argument	Hex Code	Mnemonic	RCU Buttons
[Ignored]	8E	SFM	Play, 8, E

Explanation: Any argument representing a Disc Location is treated as a Frame Number on a CAV disc and as a Extended Time Number on a CLV disc. A Frame Number is 1 to 5 decimal digits, from 1 through 54000. The Extended Time Number is 1 to 7 decimal digits (HMMSSFF), from 0 onward. H is the hour digit, if any. MM is minutes, 00 to 59. SS is seconds, 00 to 59. FF is frames, 00 to 29. Leading zeros are not required in these Disc Location numbers.

33) STM (Set Time Mode)

Function: For CLV discs, this command puts the player into *Time Code Mode*.

Argument	Hex Code	Mnemonic	RCU Buttons
[Ignored]	8D	STM	Play, 8, D

Explanation: If the disc has time numberss (a CLV disc), any argument representing a Disc Location is treated as a Time Number. The Time Number is 1 to 5 decimal digits (HMMSS), from 0 onward. H is the hour digit, if any. MM is minutes, 00 to 59. SS is seconds, 00 to 59. Leading zeros are not required in these Disc Location numbers.

34) SCM (Set Chapter Mode)

Function: If Chapter Numbers are encoded on the disc, the player enters *Chapter Mode*.

Argument	Hex Code	Mnemonic	RCU Buttons
[Ignored]	8C	SCM	Play, 8, C

Explanation: If the disc has chapter numbers, any argument representing a Disc Location is treated as a Chapter Number. The Chapter Number is a 1- or 2-digit decimal number, from 0 through 79. If the disc does not have chapter numbers, the command is ignored.

NOTE: In *Manual Mode*, the END button on the RCU can be used to cycle through the modes that are valid for a specific disc. All discs encoded with Frame Numbers can be accessed in *Frame Mode;* CLV discs can be accessed in *Time Mode*.; and any CAV or CLV disc with chapters encoded can be accessed in *Chapter Mode*.

Example: A chapter-coded CLV disc is being used. The frame at 0 hours, 20 minutes, 30 seconds, and 21 frames is displayed for one second. Then, the frame at 11 minutes, 30 seconds, and 0 frames is displayed for one second. Finally, the first frame of Chapter 16 is displayed.

Address	Argument	Command	Hex Code	Comment
0100		SFM	8E	Use Frame Mode (CLV Disc)
0101	203021	SEARCH	F7	Search to 20 min., 30 sec., 21 frames
0108	10	WAIT	FB	and Wait for 1 second
0111		STM	8D	Use Time Code Mode
0112	1130	SEARCH	F7	Search to start of 11 min., second 30
0117	10	WAIT	FB	and Wait for 1 second
0120		SCM	8C	Use Chapter Mode
0121	16	SEARCH	F7	Search to start of Chapter 16

35) SSM (Set Still Mode)

Function: This command allows the user to select a special *4-field Still Mode*.

Argument	Hex Code	Mnemonic	RCU Buttons
[Integer]	8B	SSM	Play, 8, B

Explanation: When the player is in normal Still Mode, it repeatedly accesses 2 fields - a "normal" still frame. For special applications, it may be desirable to access 4 fields instead, displaying a special "4-field still". The normal 2-field still mode is selected if the argument is 0, the 4-field mode is selected if the argument is a 1. The 2-field/4-field mode is toggled if there is no argument. SSM does not make the player enter Still Mode.

Note: An odd argument is equivalent to a 1 and an even argument is equivalent to a 0.

36 & 37) TJF and TJR (Track Jump Forward and Track Jump Reverse)

Function: The player jumps forward or in reverse quickly, by the number of tracks designated by the argument, up to 100 "tracks".

Argument	Hex Code	Mnemonic	RCU Buttons
Integer	80	*80	Play, 8, 0
Integer	81	*81	Play, 8, 1

Explanation: The argument is an integer from 1 to 100. If a value greater than 100 is given, 100 is used instead. If no argument is supplied, the command is ignored. The player jumps "quickly" to the new disc location. After the jump, the player continues in its previous operating mode..

Note: Instead of jumping actual "tracks", the LD-V8000 probably jumps "frames". On a 2-2 pulldown CAV disc there is one frame per track, but 3-2 pulldown and other disc configurations are possible. **CAUTION:** the time required for the execution of this command has changed substantially.

4.2.5 Program Execution Control Commands

These commands modify the normal sequence of program instruction execution. Usually, after one instruction has finished execution, the next sequential instruction is fetched and executed. In addition to BRANCH, BRF, and JUMP described below, see DECREG and COMPARE in the next section and the Input commands. The command NE (No Entry) is rarely used in a program.

38) BRANCH

Function: Take the next instruction from the specified program address.

Argument	Hex Code	Mnemonic	RCU Button
[Program Address]	CF	BR	RUN/BRANCH

Explanation: BRANCH directs the player to continue Level II execution at the program address specified by the argument. If no argument is present, address 0 is assumed. The BRANCH command is only executed from memory. It does not alter the status of the Audio outputs and has no effect on the disc location display or the active register pointer. BRANCH is generally used under the following conditions:

1. Unconditional branch, to cause immediate transfer of control to another location.
2. After an INPUT type command, where it ends each of the command “groups”.
3. Following DECREG, for conditional branching or loop control.
4. Following COMPARE, for conditional control of the execution sequence.

Note: The RUN/BRANCH button on the Remote Control Unit doubles as the RUN button, in *Manual Mode*, pressing RUN tells the player to execute a Level II program.

Example: A variety of BRANCH instructions are shown. (First, press 1 RECALL 10 STORE to store number 10 in register 1.)

Address	Argument	Command	Hex Code	Comment
0051	256	BRANCH	CF	Branch to program address 256
0055	1 ARG	BRANCH	CF	Branch to the address in Register 1
0058	0613	BRANCH	CF	Branch to program address 613
0063		BRANCH	CF	Branch to program address 0

39) BRF (Branch on Failure)

Function: The program branches only if the most recent “failure-reporting” command failed in its execution.

Argument	Hex Code	Mnemonic	RCU Buttons
[Program Address]	07	BRF	Play, 0, 7

Explanation: Some commands, such as SEARCH, AUTOSTOP, MSF and MSR, and the MLoad commands, set the player’s “Success / Fail” flag as they finish execution. If the flag indicates “Fail” when BRF is executed, program execution continues at the program address specified by its argument. Otherwise, the next sequential instruction is executed.

Example: If MLOAD fails, try again. Otherwise, continue program execution.

Address	Argument	Command	Hex Code	Comment
0021		VOFF	1C	Video squelched
0022	100	SEARCH	F7	
0026	2	MLOAD	CC	
0028	22	BRF	07	On failure, try the load again
0031		. . .		

40) JUMP

Function: "Branches" to a subroutine, recording a return address in Register 1.

Argument	Hex Code	Mnemonic	RCU Buttons
[Program Address]	0B	JMP	Play, 0, B

Explanation: Like a BRANCH command, JUMP causes the player to continue instruction execution at the program address specified by the argument. It also stores, in Register 1, the program address of the code byte that immediately follows the JUMP command. This stored address can be used as a return address by a subroutine exit.

Example: Call subroutine “A” (at 60) which calls subroutine “B” (at 200). Notice the different code implementing the two types of subroutine return.

Address	Argument	Command	Hex Code	Comment
0010	60	JUMP	0B	To subroutine “A”, and R1 <-- 13
0060	1 ARG	GET	08	A: Get the return address from R1
0063	5	PUT	09	and save it in Register 5
0065	200	JUMP	0B	go to another subroutine ("B"), & R1 <-- 69
0089	5 ARG	BRANCH	CF	Return from subroutine "A"
0200				B: ...
0220	1 ARG	BRANCH	CF	Bottom level return, from subroutine "B"

41) HALT

Function: This command makes the player exit *Automatic Mode* and enter *Manual Mode*.

Argument	Hex Code	Mnemonic	RCU Button
[Ignored]	BF	H	CLEAR/HALT

Explanation: This command stops program execution and returns the player to *Manual Mode*. **Note:** If the player is not in *Still Mode*, it continues to “play” after a HALT.

42) NE (No Entry)

Function: NE instructs the player to “do nothing”.

Argument	Hex Code	Mnemonic	RCU Buttons
[(*)]	FF	NE	Play, F, F

Explanation: NE’s might be put into programs to reserve space for future program modifications. Leading (non-significant) zeros in command arguments might be better.

Note: (*) NE is the one defined command that does not “eat” preceding digits as an argument, but leaves them for the following command, just as if the NE was not there. However, it is best not to rely on this. Also, the player takes 16.7 milliseconds (one field time) to execute each NE command.

4.2.6 Register Commands

Each of the following commands affect the contents of at least one of the user registers: ADD, SUBTRACT, MULTIPLY, DIVIDE, GET, PUT, DECREG, STORE, RRS, and CLOCK. The COMPARE command uses the contents of a register. RECALL affects the Active Register Pointer. The ARG, RND, and DROP "commands" are only used as part of another instruction's argument.

43) ADD (Addition)

Function: This command adds the argument value to Register 0.

Argument	Hex Code	Mnemonic	RCU Buttons
[Integer]	02	ADD	Play, 0, 2

Explanation: The integer argument (taken modulo 65536) is added to the value stored in Register 0. The resultant sum is also taken modulo 65536 and the result then replaces the value in Register 0.

Example: Add 1020 to the number stored in Register 0 and use it as a frame number for a search. Assume Register 0 initially contains 8400.

Address	Argument	Command	Hex Code	Comment
0015	1020	ADD	02	R 0 <-- R 0 + 1020
0020	0 ARG	SEARCH	F7	Search to Frame 9420 (from R0)

44) SUBTRACT

Function: This command subtracts the argument value from Register 0.

Argument	Hex Code	Mnemonic	RCU Buttons
[Integer]	03	SUB	Play, 0, 3

Explanation: The integer argument (taken modulo 65535) is subtracted from the value in Register 0. The resultant difference (also taken modulo 65536) is then stored in Register 0. Thus, if the subtraction result is negative, 65536 is added to the result. before it is stored in Register 0. For example, if R0 contains a value of 10, then 12 SUB would store 65534 in R0. (For example, 10 - 12; 65536 - 2 = 65534 in Register 0.)

Example: Assume R0 contains 1200. Subtract 50 from R0 and BRANCH to the calculated program address.

Address	Argument	Command	Hex Code	Comment
0015	50	SUBTRACT	03	R0 <-- R0 - 50
0018	0 ARG	BRANCH	CF	Branch to 1150 (from R0)

45) MULTIPLY

Function: This command multiplies Register 0 by the argument, with overflow to R3.

Argument	Hex Code	Mnemonic	RCU Buttons
[Integer]	22	MUL	Play, 2, 2

Explanation: This command takes the argument (modulo 65536) and multiplies it by the value in R0. The two lower-order bytes of the 4-byte result are stored in R0. The two high-order bytes are stored in Register 3. Thus, if the result is less than 65535, Register 3 becomes 0. For example, 358 x 450 = 161100, so R0 <-- 30028 and R3 <-- 2.

Example: Assume R0 contains 1200. Multiply R0 by 5 and use the new number as a frame number for a SEARCH.

Address	Argument	Command	Hex Code	Comment
0015	5	MULTIPLY	22	R0 <-- R0 x 5
0017	0 ARG	SEARCH	F7	Search to Frame 6000

46) DIVIDE

Function: This command divides R0 by the argument, with the remainder in R3.

Argument	Hex Code	Mnemonic	RCU Buttons
Integer	21	DIV	Play, 2, 1

Explanation: The 2-byte contents of Register 0 are divided by the integer argument (taken modulo 65536). The quotient is stored in R0 and the remainder is stored in R3. If 37 is divided by 5, R0 <-- 7 and R3 <-- 2.

Note: An attempt to divide by zero does not change R0 or R3.

Example: Assume R0 contains 62. Take Register 0 modulo 12 and use the result times 100 as a frame number for a SEARCH.

Address	Argument	Command	Hex Code	Comment
0015	12	DIVIDE	21	R3 <-- (R0 modulo 12) = 2
0018	0 ARG 00	SEARCH	F7	Search for Frame 200

47) GET

Function: This command sets Register 0 to the argument value.

Argument	Hex Code	Mnemonic	RCU Buttons
[Integer]	08	GET	Play, 0, 8

Explanation: This command replaces the contents of Register 0 by the value specified by the argument (taken modulo 65536). R0 is set to 0 if the argument is omitted.

Example: Assume Register 2 contains 25.

Address	Argument	Command	Hex Code	Comment
0100	100	GET	08	Set R0 to 100
0104	12 ARG	GET	08	Copy the contents of R12 into R0
1008	2 ARG ARG	GET	08	Copy the contents of R25 into R0

48) PUT

Function: This command copies the contents of Register 0 into a specified register.

Argument	Hex Code	Mnemonic	RCU Buttons
Register Number	09	PUT	Play, 0, 9

Explanation: This command copies the contents of Register 0 into the register specified by the argument.

Example: Make R10 and the Active Register equal to R0.

Address	Argument	Command	Hex Code	Comment
500	10	PUT	09	Copy the contents of R0 into R10
503	ARG	PUT	09	Copy R0 into the Active Register

49) RECALL

Function: This command sets or manipulates the Active Register Pointer, which specifies which register is the current Active Register.

Explanation: When there is an argument, RECALL sets the Active Register Pointer to the register number specified by the argument. Without an argument, one of two things happens: Usually the Active Register Pointer is incremented by one. If, since the last RECALL, another command (such as STORE, SEARCH, AUTOSTOP, etc.) has already incremented the Active Register Pointer, then this RECALL does nothing.

Argument	Hex Code	Mnemonic	RCU Button
[Register Number]	7F	RC	RECALL

Note: HALT does not change the Active Register Pointer. RUN sets the pointer to 1.

Example:

Address	Argument	Command	Hex Code	Comment
50	10	RECALL	7F	R10 Active
53		RECALL	7F	R11 Active
54	100	SEARCH	F7	R12 Active
58		RECALL	7F	Nothing happens
59		RECALL	7F	R13 Active

50) ARG (Argument)

Function: ARG acts as part of an argument. It generates a numerical value which is used as an argument by the command which follows it.

Argument	Hex Code	Mnemonic	RCU Buttons
[Register Number]	0A	ARG	Play, 0, A

Explanation: When ARG has an argument, that argument is a register number. The argument value generated for the next command is the value stored in the specified register. When ARG has no argument, the value generated is equal to the value of the Active Register Pointer (not the value in the Active Register).

Example #1: The contents of the Active Register, multiplied by 10, is used as the argument for a SEARCH. Assume that the Active Register contains 38.

Address	Argument	Command	Hex Code	Comment
0015		SFM	8E	Set Frame Mode
0016	ARG ARG 0	SEARCH	F7	Search to 3800

Example #2: In the following instruction examples assume that: R0 = 13; R8 = 10; R10 = 208; and R3 = 8, where R3 is the Active Register.

	Argument	Command	Hex Code	Comment
00	10 ARG	BRANCH	CF	Go to address 208
04	8 ARG ARG	BRANCH	CF	Go to address 208
08	ARG	BRANCH	CF	Go to address 3
10	ARG ARG	BRANCH	CF	Go to address 8
13	ARG ARG ARG	BRANCH	CF	Go to address 10
17	0 ARG	RECALL	7F	R13 becomes active
20	8 ARG	PUT	09	R10 <-- 13
23	10 ARG	GET	08	R0 <-- 208

51) COMPARE

Function: This command compares R0 to the argument for conditional branching.

Argument	Hex Code	Mnemonic	RCU Buttons
[Integer]	04	COM	Play, 0, 4

Explanation: The COMPARE command compares the contents of Register 0 to the argument value. Sequential execution continues if Register 0 is greater than the argument. If Register 0 is equal to the argument, program code is skipped until exactly one BRANCH command is skipped, then sequential execution continues (with the command just after that BRANCH). If Register 0 is less than the argument, two BRANCH commands are skipped, and then command execution continues (with the command immediately following the second BRANCH).

Note: No argument is equivalent to a zero argument.

Example: If R0 > R21, SEARCH to Frame 1000. If R0 = R21, set R0 to 36. If R0 < R21, do neither. Then wait 2 seconds and SEARCH to Frame 1234.

Address	Argument	Command	Hex Code	Comment
0019	21 ARG	COMPARE	04	Compare R0 and R21
0023	1000	SEARCH	F7	Search when R0 > R21
0028	37	BRANCH	CF	
0031	36	GET	08	When R0 = R21, R0 <-- 36
0034	37	BRANCH	CF	
0037	20	WAIT	FB	Skip to here if R0 < R21
0040	1234	SEARCH	F7	

52) DECREG (Decrement Register)

Function: This command tests and decrements a register, usually for loop control.

Argument	Hex Code	Mnemonic	RCU Button
[Register Number]	F0	DR	DEC REG

Explanation: If the value in the specified register is greater than zero, the value is decremented by one. Then, if the resultant value in the register is greater than zero, program execution proceeds with the instruction immediately following the DECREG. If the value is equal to zero, all following commands are skipped until one BRANCH is skipped, then program execution resumes with the instruction immediately following that BRANCH. DECREG does not change the Active Register Pointer. No argument is equivalent to a zero argument.

Example: Odd numbered registers 21 through 29 hold the start frames and the even numbered registers 22 through 30 hold the end frames for five video motion sequences. Play all five sequences, then show a "menu" at frame 6500.

Register Contents

R 21	=	1001	R 26	=	13000
R 22	=	2000	R 27	=	5100
R 23	=	4000	R 28	=	5800
R 24	=	6000	R 29	=	6001
R 25	=	10000	R 30	=	6011

Address	Argument	Command	Hex Code	Comment
0500	21	RECALL	7F	Make Register 21 the Active Register
0503	5	GET	08	Put count of 5 into Register 0
0505		SEARCH	F7	Search to a start frame
0506		AUTOSTOP	F3	Play to an end frame
0507	0	DECREG	F0	Test and decrement the R0 counter
0509	505	BRANCH	CF	Loop to play next sequence
0513	6500	SEARCH	F7	Show the menu

53) DROP

Function: The DROP command acts as part of an argument. It "drops" the low-order decimal digit from the argument it receives, passing the result on as an argument value.

Argument	Hex Code	Mnemonic	RCU Buttons
[Integer]	1D	DRP	Play, 1, D

Explanation: The argument value (perhaps taken modulo 65536) is divided by 10, with any remainder discarded. The resulting value is used as the argument for the following command. With no argument or arguments less than 10, 0 is passed on as the new argument value.

CAUTION: In some cases 90000 DROP SEARCH will not search to frame 9000 due to a modulo 65536 operation performed on the 90000 argument.

Example: Set R0 to 123. Then, replace the low digit with a 7.

Address	Argument	Command	Hex Code	Comment
0014	123	GET	08	Put 123 into R0
0018	0 ARG DROP 7	GET	08	Puts 127 into R0*

54) RND (Generate Random Number)

Function: RND generates a random number which is used as an argument for the following command.

Argument	Hex Code	Mnemonic	RCU Buttons
[Ignored]	05	RND	Play, 0, 5

Explanation: RND acts as part of an argument. It ignores its own argument and generates a "random" value in the range of 0 through 255. That value becomes the argument for the following command.

* Recalls contents of R0 =123; divides by 10 with no remainder =12. Multiplies previous argument by 10 and adds 7. Puts 127 into R0.

Example: Display frame 1000 or frame 3000. Choose frame 1000 about 30 percent of the time.

Address	Argument	Command	Hex Code	Comment
0014	RND	GET	08	Set R0 to a random number 0-255
0016	77	COMPARE	04	77 = 30% of 256
0019	3000	SEARCH	F7	R0 > 77, Show frame 3000
0024	35	BRANCH	CF	and continue at address 35
0027	19	BRANCH	CF	R0 = 77, go show frame 3000
0030	1000	SEARCH	F7	R0 < 77, Show frame 1000
0035		...		Continue program execution

55) STORE

Function: This command stores a value in the Active Register.

Argument	Hex Code	Mnemonic	RCU Button
[Integer]	F5	ST	STORE

Explanation: If there is no argument, the numerical value of the Disc Location currently being accessed by the player (for example, a frame number) is used as the argument. Then, STORE writes the argument value (modulo 65536) into the Active Register. The Active Register Pointer is incremented by one after the value is stored.

NOTE: With a CLV disc, only the hours, minutes, and seconds of the Disc Location are used, the frame (0 - 29) is not.

Example #1: Place the current frame number in Register 20 and the value 4000 in Register 21.

Address	Argument	Command	Hex Code	Comment
0010	20	RECALL	7F	Activate Register 20
0013		STORE	F5	R20 <-- current frame #, activate R21
0114	4000	STORE	F5	Store 4000 in R21, activate R22

Example #2: Assume a video segment is being played, but we want to interrupt it to see a "Help" frame (FR 5000) for 3 seconds; then continue playing from the interrupt point.

Address	Argument	Command	Hex Code	Comment
0100		WAIT	FB	Stop the player
0101	19	RECALL	7F	Make R19 active
0109		STORE	F5	Store current frame # in R19
0110	5000	SEARCH	F7	Show the "Help" frame (FR 5000)
0114	30	WAIT	FB	for 3 seconds
0117	19 ARG	SEARCH	F7	Return to the interrupt frame *
0118		PLAY	FD	and continue playing

* Recall contents of R19 to return to the interrupt frame.

56) RRS (Read Rear Switch)

Function: RRS puts the sixteen user function switch settings into R0.

Argument	Hex Code	Mnemonic	RCU Buttons
[Ignored]	10	RRS	Play, 1, 0

Explanation: This command reads the sixteen user-function switches (1 - 16), counting an "ON" switch as a "1" bit and an "OFF" switch as a "0" bit. The 16-bit value is stored in Register 0.

Note: On the LD-V8000, these on-screen parameter "switches" are accessed by pressing the player's "Display" button while turning on the player's power. Then, the user-parameter switch settings can be viewed using the player's Scan Forward or Reverse buttons. They can be changed using the Step Forward and Step Reverse buttons. Switch 1 is the least significant and Switch 16 is the most significant. See ***LD-V8000 Level I & III User's Manual/Programmer's Reference Guide*** *TP 113 v.2.0 3/91*, **Section 2.4.**

Example: If user function Switch 12 is ON (1), access Frame 1000, otherwise access Frame 2700.

Address	Argument	Command	Hex Code	Comment
101		RRS	10	Set R0 to Switches
102	2048	DIVIDE	21	Remove SW 1 - 11 bits
107	2	DIVIDE	21	SW 12 bit to Register 3
109	0	GET	08	R0 <-- 0
111	3 ARG	COMPARE	04	Test SW 12 ON or OFF?
114	132	BRANCH	CF	R0 > R3 (not possible here)
118	2700	SEARCH	F7	R0 = R3, so show frame 2700
123	132	BRANCH	CF	and continue
127	1000	SEARCH	F7	R0 < R3, so show frame 2000
132				and continue

57) CLOCK (Clock Read and Reset)

Function: This command reads and then resets the user tenth-second timer.

Argument	Hex Code	Mnemonic	RCU Buttons
[Ignored]	16	CLK	Play, 1, 6

Explanation: This command sets Register 2 to the current value of the tenth-second timer. Then, it resets the timer to zero. The timer is incremented every 1/10 second, counting to 65535, where it overflows to zero again. Thus, it increases for 6553.5 seconds.

Example: Measure the time required to search from Frame 100 to 4000. Then, display the time required as a frame number, showing the number of 0.1-second time periods.

Address	Argument	Command	Hex Code	Comment
0000	100	SEARCH	F7	Search to Frame 100
0004		CLOCK	16	and Clear the timer
0005	4000	SEARCH	F7	Search to Frame 4000
0010		CLOCK	16	R2 <-- current timer value
0011	2 ARG	SEARCH	F7	Use R2 as search target
0014	1	DISPLAY	F1	and display the "elapsed time"

4.2.7 Input Processing Commands

The commands described in this section allow the Level II program to respond to external inputs. The subsequent path of program execution may be changed, depending upon the received input.

The commands are usually used to process viewer "button presses" from the hand-held Remote Control Unit (RCU) or an equivalent keypad. The commands can also respond to various data bytes sent to the player by an external controller or computer on the RS232C port. These commands permit a wide range of interactivity between the viewer/user and a Level II program.

58) INPUT (Input from Digit Keys)

Function: INPUT waits for a 0 - 9 digit code, and then modifies the program execution sequence.

Argument	Hex Code	Mnemonic	RCU Button
[Integer]	F8	IN	INPUT

Explanation: INPUT puts the player into *Input Mode*, where it waits indefinitely for one of the 0 - 9 digit codes. The digit code is usually generated as the result of a viewer's pressing one of the digit buttons on the RCU. This is often a response to a “menu” of choices displayed on the video screen.

The command’s argument is normally a single digit "N" (1 - 9). High-order digits, if present, are ignored. If the argument is missing or 0, the value 9 is used for N. The argument allows a separate block of code to be executed for each of the N “expected” viewer inputs - the codes for the digits 0 through (N-1). The other digit codes (N - 9) are treated in one single category, as “other digits”. For example, when N = 3, each of the responses 0, 1, or 2 causes its own block of code to be executed. The "other" entries (codes for the digits 3 through 9) are not ignored - they cause the player to skip over the three reserved blocks of code and continue program execution.

While waiting for this numeric input, the player does not accept most other "commands" from the RCU. REJECT, HALT, AUDIO1 and AUDIO2, and various Transmit commands are exceptions.

When digit M's code is received, if M < N, the player skips M blocks of code. Otherwise, N blocks are skipped over. Then, program execution continues. For each block, code is skipped until exactly one BRANCH is skipped.

The following table shows the relationship between argument values, “expected” input digits, and “other” digits.

Argument Value (N)	Range of “Expected” Numeric Input	Range or “Other” Numeric Input
1	0 Only	1 through9
2	0 or 1	2 through 9
3	0 through 2	3 through 9
4	0 through 3	4 through 9
5	0 through 4	5 through 9
6	0 through 5	6 through 9
7	0 through 6	7 through 9
8	0 through 7	8 and 9
9	0 through 8	9 only

Example #1: The player waits for numeric button input and execution proceeds in one of four ways, depending upon the input value.

Address	Argument	Command	Hex Code	Comment
0100	1000	SEARCH	F7	Search to Menu Frame (1000)
0105	3	INPUT	F8	Wait for input 0, 1, 2, (3 - 9)
0107	1500	AUTOSTOP	F3	0-digit: play 1000 to 1500
0112	147	BRANCH	CF	<end of 0-digit code block>
0116	3500	SEARCH	F7	1-digit: Search to 3500
0121	4000	AUTOSTOP	F3	and play to 4000
0126	147	BRANCH	CF	<end of 1-digit code block>
0130	4500	SEARCH	F7	2-digit: Search to 4500
0135	100	WAIT	FB	and wait 10 seconds
0139	147	BRANCH	CF	<end of 2-digit code block>
0143	105	BRANCH	CF	digits 3-9, ignore these
0147		...		

Example #2: In response to the "menu" at frame 1000, branch to one of 4 different locations (400, 500, 600, and 700) when an input of 1, 2, 3, or 4 is received.

Address	Argument	Command	Hex Code	Comment
0100	1000	SEARCH	F7	Search to FRAME 1000
0105	5	INPUT	F8	Allow 0 - 4 and “other (5 - 9)
0107	105	BRANCH	CF	0-digit: ignore it
0111	400	BRANCH	CF	1-digit: go to address 400
0115	500	BRANCH	CF	2-digit: go to address 500
0119	600	BRANCH	CF	3-digit: go to address 600
0123	700	BRANCH	CF	4-digit: go to address 700
0127	105	BRANCH	CF	digits 5-9: ignore

59) FIN (Input with Function Keys)

Function: FIN waits for a digit or function key code, and then modifies the program execution sequence.

Argument	Hex Code	Mnemonic	RCU Buttons
[Integer]	18	FIN	Play, 1, 8

Explanation: Operation is the same as the INPUT command, except that function key codes are allowed. As with INPUT, the "other" digits skip N blocks of code. All of the function keys cause (N+1) blocks of code to be skipped. A function key input is "processed" when the byte code for one of the buttons listed below is recognized.

See **Figure 3-B**, on page 3-3, for a detailed diagram of the RU-V6000T remote control. The function buttons are indicated and corresponding function button numbers are listed below:

Remote Control Buttons	Function Key Number
AUTOSTOP	10
STEP REV	11
STEP FWD	12
SEARCH	13
SLOW REV	14
SLOW FWD	15
SCAN REV	16
SCAN FWD	17
STOP	18
PLAY	19

The function key number of the button that was pressed is made available to the program by the DIN command. The digits 0 - 9 have "function key" numbers of 0 - 9.

Example: The player waits for digit or function key input. Processing continues when Function Key 15 (SLOW FWD) is pressed. Ignore all other inputs

Address	Argument	Command	Hex Code	Comment
0000	1	FIN	18	Wait for digit or function key
0002		BRANCH	CF	digit-0: ignore
0003		BRANCH	CF	digits 1-9: ignore
0004	DIN	GET	08	Function Key: R0 <-- function #
0006	15	COMPARE	04	Was it Function Key #15?
0009		BRANCH	CF	Key over 15, ignore
0010	14	BRANCH	CF	Key 15: Go on to address 14
0013		BRANCH	CF	Key less than 15, ignore
0014		...		Processing continues

60) TIN (Input with Timeout)

Function: TIN waits for a digit code or a timeout, and then modifies the program execution sequence.

Argument	Hex Code	Mnemonic	RCU Buttons
[Integer]	0E	TIN	Play, 0, E

Explanation: TIN behaves like a FIN command where the only "function key" allowed is the timeout - "function key" 20. The least significant digit of the argument ("N") is still used as in the INPUT command, but the whole argument is used to specify a timeout period, expressed in tenths of a second. Thus, an argument of 278 specifies a 27.8 second timeout and sets "N" = 8. The maximum timeout period is 10,000 tenth-seconds.

If a response is not received before the timeout period expires, a pseudo-function key response is generated internally and the player process it just like any other function key response. Thus, TIN will only wait the specified amount of time for an input, then processing will continue (skipping (N+1) blocks of code).

Thus, there should be one BRANCH instruction following TIN for each of the N "allowed" viewer inputs, and one additional BRANCH for the “Other” inputs. Then, as with the FIN command, the following code is for the timeout condition.

Example: Display Frame 100 and wait for viewer input. If the viewer enters a 1, show Frame 300. If the viewer enters any other numeric digit, or does not respond within 10.2 seconds, show Frame 200.

Address	Argument	Command	Hex Code	Comment
0000	100	SEARCH	F7	Search for Frame 100
0004	102	TIN	0E	Wait for Input, with timeout
0008	21	BRANCH	CF	0-digit: go show frame 200
0011	300	SEARCH	F7	1-digit: Show frame 300
0015	25	BRANCH	CF	and continue
0018	21	BRANCH	CF	digit 2-9: go show frame 200
0021	200	SEARCH	F7	Timeout: Show frame 200
0025		...		Program Continues

61) FTI (Input with Function and Timeout)

Function: FTI waits for a digit code, function code, or timeout, then it modifies the program execution sequence.

Argument	Hex Code	Mnemonic	RCU Buttons
[Integer]	19	FTI	Play, 1, 9

Explanation: The FTI command is a combination of Tin and FIN. Its operation is like that of FIN except that it also allows the timeout "function key" (see TIN). As with TIN, the timeout duration is specified by the argument in tenths of seconds. The maximum timeout is 10,000 tenth seconds.

Example #1: Frame 100 is displayed and the player waits for input. A frame number is calculated by multiplying the input's equivalent "function key number" by 20, then that frame is displayed. Thus, frame 400 is displayed if the 20.1 second timeout occurs.

Address	Argument	Command	Hex Code	Comment
0000	100	SEARCH	F7	Search for Frame 100
0004	201	FTI	19	Wait for input or timeout
0008	14	BRANCH	CF	0-digit: go calculate frame #
0011	14	BRANCH	CF	digit 1-9: go calculate frame #
0014	DIN	GET	08	Function or Timeout: R0 <-- "function #"
0016	20	MULTIPLY	22	calculate frame #, R0 <-- R0 x 20
0019	0 ARG	SEARCH	F7	Search to calculated frame
0022		HALT	BF	Program end

Example #2: In response to a digit or function key input, play from frame 100 to Frame 200. If the 10.1 second timeout occurs, just display frame 300.

Address	Argument	Command	Hex Code	Comment
0050	100	SEARCH	F7	Search to frame 100
0054	101	FTI	19	Accept viewer input
0058	100	SEARCH	F7	0-digit: Play the sequence
0062	200	AUTOSTOP	F3	to the end
0066	90	BRANCH	CF	and continue
0069	58	BRANCH	CF	digit 1-9: go play the sequence
0072	DIN	GET	08	Function: R0 <-- function #
0074	20	COMPARE	04	Is it a timeout (#20)?
0077	58	BRANCH	CF	> 20, go play the sequence
0080	300	SEARCH	F7	= 20 (timeout), so show frame 300
0084	90	BRANCH	CF	and continue
0087	58	BRANCH	CF	< 20, go play the sequence
0090		...		program continues

62) DIN (Data Input)

Function: DIN provides information about the viewer's response to the last Input-type command (INPUT, FIN, TIN, FTI, or IIN).

Argument	Hex Code	Mnemonic	RCU Buttons
[Ignored]	1E	DIN	Play, 1, E

Explanation: DIN generates a value to be used as an argument by the command which follows it. DIN generates a value of 0 through 20, depending on the last input received. A 0 - 9 numeric-digit button generates the corresponding value, 0 - 9. A function key input generates a value of 10 through 19 (see FIN). A timeout generates the value 20. It can be used after any of the input commands to determine which input was received.

Example #1: Digit 6 input shows frame 200. Other inputs are ignored.

Address	Argument	Command	Hex Code	Comment
0000	1	INPUT	F8	Make the player wait for input
0002		BRANCH	CF	0-digit: ignore, go to address 0
0003	6	GET	08	Set R0 = 6
0005	DIN	COMPARE	04	Compare R0 with input #
0007		BRANCH	CF	6 > #, ignore
0008	12	BRANCH	CF	6 = #, go show frame 200
0011		BRANCH	CF	6 < #, ignore
0012	200	SEARCH	F7	Show frame 200

Example #2: Following a button press of a 0 - 9 digit button, place the associated numerical value in Register 0.

Address	Argument	Command	Hex Code	Comment
0000	1	INPUT	F8	Accept viewer input
0002	4	BRANCH	CF	0-digit
0004	DIN	GET	08	digits 1-9: R0 <-- digit number

63) BIN (Binary Input)

Function: BIN generates an argument for the following command from any input data.

Argument	Hex Code	Mnemonic	RCU Buttons
[Ignored]	17	BIN	Play, 1, 7

Explanation: The player can receive button-press inputs from the RCU and single-byte inputs from the RS232 port. The RCU button press is converted by the player to a one-byte code (0 - 255). The last byte received is saved by the player as the "last input" code. BIN reads this "last input" code, converts it to a decimal argument value for the next command, and then sets the "last input" code to 255. If a button is being held down, it's code is immediately stored again as the current "last input" code.

Example: The program does search until there are no button being pressed, then it does not continue until some button is pressed.

Address	Argument	Command	Hex Code	Comment
0006	255	GET	08	R0 <-- 255
0010	BIN	COMPARE	04	watch for no-input code (255)
0012	10	BRANCH	CF	255 > #, button being or was pressed
0013	2000	SEARCH	F7	# = 255Search
0018	24	BRANCH	CF	and go watch for any button press
0021	10	BRANCH	CF	255 < #, abnormal, try again
0024	BIN	COMPARE	04	Wait loop, for some key to be pressed
0026	35	BRANCH	CF	255 > #, button pressed, continue
0029	24	BRANCH	CF	# = 255, no button, try again
0032	24	BRANCH	CF	255 < #, abnormal, try again
0035		...		continue program

64) IIN (Interrupt Input)

Function: This command sets up conditions so that certain external inputs will cause the normal sequence of instruction execution to be interrupted.

Argument	Hex Code	Mnemonic	RCU Buttons
[Integer]	20	IIN	Play, 2, 0

Explanation: Unlike other Input-type commands, this command does not wait for external input but rather tells the player to monitor external inputs while program execution continues. Digits 0 - 9 and function keys 10 - 19 can be monitored.

The argument is used to determine which viewer inputs (0 - 19) will be monitored. If the argument is set to N (0 < N < 21), inputs from 0 through (N - 1) are monitored. For example, if the argument is 3, then 0, 1, and 2 will be monitored. If an argument is not specified or the argument is set to 0, monitoring is discontinued.

When IIN with a non-zero argument is executed, the following commands up through the next BRANCH command are skipped, With no argument or a zero argument, no commands are skipped.

If a monitored viewer input occurs, Register 1 is set to the program address of the next command that normally would have been executed, monitoring is disabled, the normal sequence of program instructions is “interrupted”, and execution commences with the command immediately following the IIN command. The input’s “function key” value can be obtained with the DIN command.

Only the last IIN command remains in effect if two or more IIN commands are executed. The other Input-type commands (IN, FIN, TIN, or FTI) also terminate monitoring.

Example: Frames 1000 through 1600 are played repeatedly while inputs are monitored. If 0 through 11 is pressed, Frame 2000 + DIN is shown for 2 seconds.

Address	Argument	Command	Hex Code	Comment
0000	12	IIN	20	Allow 0 to 11 keys to interrupt
0003	DIN	GET	08	Interrupt: R0 <-- key value
0005	2000	ADD	02	R 0 = R 0 x 20000
0010	0 ARG	SEARCH	F7	Search
0013	20	WAIT	FB	Wait two seconds
0016	32	BRANCH	CF	continue program
0019	1000	SEARCH	F7	Search to Frame 1000
0024	1600	AUTOSTOP	F3	and play to Frame 1600
0029	18	BRANCH	CF	and Repeat
0032		...		

CAUTION: Register 1 is also used by the JUMP command for subroutine return addresses. Thus, an IIN interrupt may invalidate the return address in R1, limiting the usefulness of subroutines while using IIN. Also, continuation of the interrupted code using the Register 1 address should only be done with care, since the previous command may not have "finished" its execution when the interrupt occurred (as AUTOSTOP).

4.2.8 Flag Set Commands

The following commands select non-standard operational modes of the videodisc player.

65 & 66) RCE and RCD (Remote Control Enable and Disable)

Function: These commands can block (disable) or allow input from the RCU.

Argument	Hex Code	Mnemonic	RCU Buttons
[Ignored]	12	RCE	PLAY, 1, 2
[Ignored]	13	RCD	PLAY, 1, 3

Explanation: Normally the Remote Control Unit input is enabled. The RCD command disables that input. RCE enables the input again. Both the infrared and the wired RCU inputs are affected.

Note: A HALT does not re-enable a disabled RCU input. However, the RCU is re-enabled at power-on.

67) SCS (Special Control Switches)

Function: This command turns the player's special control switch bits ON or OFF.

Argument	Hex Code	Mnemonic	RCU Buttons
Integer	8A	SCS	Play, 8, A

Explanation: All four of the LD-V8000 player's special control switch bits are set to ON (1) or OFF (0) when this command is executed. The low order 4 bits of the argument (bits 0, 1, 2, and 3) are used to set the 4 switches. The power-on value for all 4 bits is OFF (zero). A description of the switch functions is provided below.

Note: The meaning and detailed functioning of these bits may be specific to this player. Level II programs to be used on other players should be carefully tested.

Special control switch Functions

Bit 0: Audio Squelch Disable
This bit disables the automatic audio squelch mechanism. If this bit is set ON, audio is not automatically squelched as usual. If this bit is OFF, the player automatically squelches the audio to match the current the playback mode.

Bit 1: Video Squelch Disable
This bit disables the automatic video squelch mechanism. If this bit is set ON, video is not automatically squelched as usual. If this bit is OFF, the player automatically squelches the video to match the current the playback mode.

Bit 2: SEARCH-End Mode
This bit determines whether the player enters *Still Mode* or *Play Mode* after a SEARCH is finished execution. If this bit is set ON, the player enters *Play Mode* after a SEARCH. If this bit is OFF, the player enters *Still Mode* after a SEARCH, as usual.

Bit 3: AUTOSTOP-End Mode
This bit determines whether the player enters *Still Mode* or *Play Mode* in one of the following situations: an AUTOSTOP, PLAY (with an argument), or a MULTI-SPEED command finishes execution, an IIN interrupt occurs, or the chapter / frame / time mode is changed. If this bit is set ON, the player enters *Play Mode* after the relevant operation. If this bit is OFF, the player enters *Still Mode* after the relevant operation, as usual.

4.2.9 Transmit Commands

Three transmit commands can be used to send bytes of information from the player to external devices via the RS232 port. Three other commands provide support.

68, 69, & 70) TM, ITM, and DTM (Transmit Memory commands)

Function: These commands send one byte of data out the RS232 port.

Argument	Hex Code	Mnemonic	RCU Buttons
[Ignored]	DC	TM	PLAY, D, C
[Ignored]	DD	ITM	PLAY, D, D
[Ignored]	DE	DTM	PLAY, D, E

Explanation: The TM command sends the byte stored at the program address indicated by the Transmit Pointer. The Transmit Pointer is set by the STP command.

The ITM command increments the Transmit Pointer (+1), then performs like a TM.

The DTM command decrements the Transmit Pointer (-1), then performs like a TM.

Note: CAUTION: ASCII output mode is not available on the LD-V8000. Binary output mode is the only output mode available. It is the power-on default.

71) STP (Set Transmit Pointer)

Function: STP sets the Transmit Pointer to the specified program address.

Argument	Hex Code	Mnemonic	RCU Buttons
[Program Address]	E8	STP	PLAY, E, 8

Explanation: STP sets the Transmit Pointer to the argument value, which should be a program address in active memory. The pointer is set to zero if there is no argument.

Note: Power-on and REJECT initialize the Transmit Pointer to zero.

Example: Transmit the 10 bytes at program address 1016 through 1025.

Address	Argument	Command	Hex Code	Comment
0040	1015	STP	E8	Set Transmit Pointer to 1015
0045	10	GET	08	Set a counter to 10
0048		ITM	DD	Increment pointer & transmit data
0049	0	DECREG	F0	decrement counter & test
0051	48	BRANCH	CF	continue loop

72 & 73) ITP and DTP (Increment and Decrement Transmit Pointer)

Function: These commands increment or decrement the Transmit Pointer.

Argument	Hex Code	Mnemonic	RCU Buttons
[Ignored]	D9	ITP	PLAY, D, 9
[Ignored]	DA	DTP	PLAY, D, A

Explanation: The ITP command increases the value stored in the Transmit Pointer by one. The DTP command decreases the value by one.

Example: Transmit the contents of program address11 followed by program address 13.

Address	Argument	Command	Hex Code	Comment
0000	11	STP	E8	Set Transmit Pointer to address 11
0003		TM	DC	Transmit contents of location 11
0004		ITP	D9	Increment pointer to location 12
0005		ITM	DD	Increment pointer to 13 and transmit

4.2.10 Video Buffer Control Commands

Four new commands are available in the LD-V8000 to change the default operation of the player's Video Buffer memory.

These commands are available on the LD-V8000 because the player has a full-frame Video Buffer which can be used as two independent one-field buffers. These Video Buffer Control commands cannot be used with other players because they manipulate the unique video buffer hardware capabilities of the LD-V8000 videodisc player.

NEW 74) SMS (Set Video Memory Switches)

Function : SMS is used to set 3 of the player's video buffer control switches.

Argument	Hex Code	Mnemonic	RCU Buttons
[Integer]	84	SMS	PLAY, 8, 4

Explanation: This command sets three switches in player control Register G, as specified by the argument (see following table). When a HALT is executed, these switches return to the settings they had before Level II program execution.

There are 3 video buffer control switches which can be set:

- ***Program Buffer Control switch***
  This switch must be set to allow the program to control operation of the video buffer. When this switch is ON, the other 3 Video Buffer Control commands can be used to control the video buffer. If this switch is set to OFF, the video buffer is controlled automatically by the player.

- ***Buffer Field Mode Select switch***
  This switch determines whether video buffer is used as two one-field buffers or as one two-field (one frame) buffer. When the switch is ON, *Buffer Field Mode* is selected and each buffer field can be controlled independently. Then, video field 0 or field 1 may be selected for video storage and video output. When *Buffer Frame Mode* is selected, both field buffers are used and controlled together. Then, both video fields 0 and 1 are automatically selected.

- ***Blue / Black Search-Squelch switch***
  This switch selects the video source to be used to generate the player's "search-squelch" output. When the switch is ON, the player's internal Blue / Black screen generator is selected. When the switch is OFF, the player's internal video buffer is selected, so that the image held there continues being displayed.

ARGUMENT	Program Buffer Control	Field / Frame Mode	Blue / Black Search-Squelch
0	OFF	FRAME	OFF
1	ON	FRAME	OFF
16	OFF	FIELD	OFF
17	ON	FIELD	OFF
32	OFF	FRAME	ON
33	ON	FRAME	ON
48	OFF	FIELD	ON
49	ON	FIELD	ON
CAUTION: Use **only** the arguments listed in this table.			

NEW 75) SRM (Select Read Memory)

Function: SRM selects which field buffer to use to produce video output.

Argument	Hex Code	Mnemonic	RCU Buttons
Integer	85	SRM	PLAY, 8, 5

Explanation: In *Buffer Field Mode*, the argument values 0 or 1 select field buffer 0 or field buffer 1 as the video buffer's output source. Only the values 0 or 1 may be specified. In *Buffer Frame Mode*, this 1 / 0 setting is ignored.

NEW 76) MWE (Memory Write Enable)

Function: MWE allows disc playback video to be written into the selected video buffer.

Argument	Hex Code	Mnemonic	RCU Buttons
[Integer]	86	MWE	PLAY, 8, 6

Explanation: This function enables the writing of disc playback video information into the video buffer. In *Buffer Field Mode*, buffer field 0 or 1 is independently enabled with a 0 or 1 argument. Only 0 or 1 may be specified. In *Buffer Frame Mode*, both buffer fields are enabled, independent of the command's argument.

NEW 77) MWD (Memory Write Disable)

Function: MWD inhibits disc playback video writing into the selected video buffer.

Argument	Hex Code	Mnemonic	RCU Buttons
[Integer]	87	MWD	PLAY, 8, 7

Explanation: This function disables the writing of disc playback video information into the video buffer. In *Buffer Field Mode*, buffer field 0 or 1 is independently disabled with a 0 or 1 argument. Only 0 or 1 may be specified. In *Buffer Frame Mode*, both buffer fields are disabled, independent of the command's argument.

Example: Use the new Video Buffer Control commands to create a Sound-Over-Still effect. Set the Program Buffer Control switch ON. Then, hold the video image at frame 1000 and play Audio Channel 1 from frame 3000 to 4000. Finally, display frame 2000, hold that image, and play Audio Channel 2 from frame 5000 to 6000.

Address	Argument	Command	Hex Code	Comment
0000	1	SMS	84	Set Program Buffer Control ON
0002	1000	SEARCH	F7	Search to frame 1000
0007		ANF	A2	Turn ON Audio Channel 1 only
0008	0	MWD	87	Hold the video image
0010	3000	SEARCH	F7	Search to frame 3000
0015	4000	AUTOSTOP	F3	and play to frame 4000
0020	2000	SEARCH	F7	Search to Frame 2000
0025	0	MWE	86	capture another image in video buffer
0027	0	MWD	87	and hold the image
0029		AFN	A1	Turn ON Audio Channel 2 only
0030	5000	SEARCH	F7	Search to frame 5000
0035	6000	AUTOSTOP	F3	and play to frame 6000

4.3 LD-V8000 EPROM Upgrades

The Audio Sync Lock/Video Delay Time Switch was implemented on LD-V8000 videodisc players manufactured since October 1990. All LD-V8000 players with EPROMs #1119 and #1120 and above contain this feature. If your player has older EPROMs and you want to upgrade, contact Pioneer Parts and Service. For an EPROM Upgrade Kit Dealers can call: 1-800-457-2881; End Users can call 1-800-228-7221. See ***Technical Bulletin #137*** or ***Appendix D*** of the ***LD-V8000 User's Manual/Programmer's Reference Guide*** for details.

4.3.1 Video Delay Time

The LD-V8000 always sends the video signals read from the videodisc through its video buffer before it generates an output video signal. Audio, on the other hand, is processed directly to the audio outputs. Therefore, the video signal is delayed, and the audio signal is not. The video signal may be delayed up to 16.2 milliseconds (refer to **Figure 4-1** below). Normally, this delay time is anything from 4.2 to 16.2 msec, depending upon how the disc happened to spin up. If directed to do so, the player can take a longer time to spin up (up to 20 seconds) and force the delay time into a narrower range, 15.2 to 16.2 msec.

A "VIDEO DELAY TIME" switch (also referred to as Audio Sync Lock) can be set to select the narrower delay time range. The switch option has been added to Page 4 of the *Function Switch Setting Mode* (hold down DISPLAY at player power-on). This feature is available only when CAV discs are used. This switch is ignored when CLV discs are used.

The LD-V8000 only adjusts the delay time at spin-up. Therefore, if external synchronization is done after spin-up, the delay time may change. To achieve proper external synchronization, attach the signals to the player before spin-up.

The "VIDEO DELAY TIME" switch is "switch" 7 (bit 7) of EPROM switch bank 6, which is read into Register H at power on.

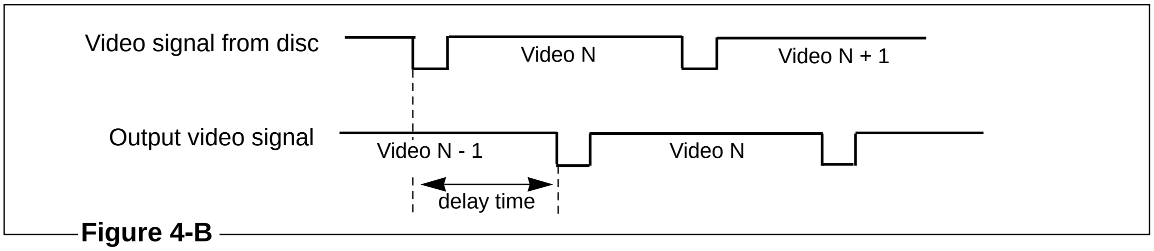


Figure 4-B

Appendix A: Comparison of Level II Commands Available on Different Pioneer Industrial LaserDisc Players

APPENDIX

A

LD-V8000

LEVEL II

USER'S MANUAL

Programmer's Reference Guide

Level II Command Comparisons by Player

A = Available
N/A = Not Available
Changed = Available, but meaning is changed

(p) = Performance substantially modified
(e) = "Eats" argument, for compatibility

Mnemonic	Command	LD-V8000	LD-V6000A	LD-V6000	LD-V3000	PR7820-3
	Mode Control Commands					
PGM	Programming Mode	Changed	A	A	A	A
END	End Programing Mode	Changed	A	A	A	A
RUN	Run Program	A	A	A	A	A
H	Halt Program	A	A	A	A	A
	Program Load Commands					
PAG	Page (Set Memory Page)	A	A	A	N/A	N/A
L	Load Program	A	A	A	A	A
L	Moving Load	A	A	A	N/A	N/A
PLD	Partial Load	A	A	A	N/A	A
PLD	Moving Partial Load	A	A	A	N/A	N/A
	Audio Control Commands					
CX	CX Control	N/A (automatic)	A	A	A	N/A
A1	Audio 1/L Out	A	A	A	A	A
A2	Audio 2/R Out	A	A	A	A	A
AXX	Set Audio Status	A	A	A	A	A
DAD	Digital Audio Out	A	A	N/A	N/A	N/A
	Video Control Commands					
VFF	Video Off	A	A	A	A	A
VON	Video On	A	A	A	A	A
	Character Generator Control Commands					
DI	Display	A	A	A	A	A
SUD	Set User Display	A	A	N/A	N/A	N/A
CLD	Clear Display	A	A	N/A	N/A	N/A
BLK	Blink	A	A	N/A	N/A	N/A
CLB	Clear Blink	A	A	N/A	N/A	N/A
SBC	Set Background Color	A	A	N/A	N/A	N/A
CGE	Character Generator Enable	A	A	A	N/A	Changed
CGD	Character Generator Disable	A	A	A	N/A	Changed

Level II Command Comparisons by Player

Mnemonic	Command	LD-V8000	LD-V6000A	LD-V6000	LD-V3000	PR7820-3
	Player Control Commands					
RJ	Reject	A	A	A	A	A
PAU	Pause Execution	A	A	A	A	A
P	Play	A	A	A	A	A
SC	Search	A	A	A	A	A
MF	Multi Speed Forward	A	A	A	A	A
MR	Multi Speed Reverse	A	A	A	A	A
W	Wait (Stop)	A	A	A	A	A
W	Wait (Stop and Delay)	A	A	A	A	A
SF	Step Forward	A (p)	A	A	A	A
SR	Step Reverse	A (p)	A	A	A	A
SCN	Scan to Target	N/A	A	N/A	N/A	N/A
-	Scan Fwd / Rev	N/A	N/A	N/A	N/A	N/A
TJF (*80)	Track Jump Forward	A (p)	A	N/A	N/A	N/A
TJR (*81)	Track Jump Reverse	A (p)	A	N/A	N/A	N/A
AS	AutoStop	A	A	A	A	A
SMK	Set Marker	N/A	A	N/A	N/A	N/A
SFM	Set Frame Mode	A	A	N/A	N/A	N/A
STM	Set Time Mode	A	A	N/A	N/A	N/A
SCM	Set Chapter Mode	A	A	N/A	N/A	N/A
LPD	Landing Pad	N/A	A	A	N/A	A
SS	Slow Speed Set	A	A	A	A	N/A
FS	Fast Speed Set	A	A	A	A	N/A
FSM	Field Step Mode	N/A	N/A	N/A	N/A	A
SSM	Set Still Mode	A	A	N/A	N/A	N/A
WFW	White Flag Wait	N/A	N/A	N/A	N/A	A
	Register Management Commands					
ADD	Addition	A	A	A	A	A
SUB	Subtract	A	A	A	A	A
MUL	Multiply	A	A	N/A	N/A	N/A
DIV	Divide	A	A	N/A	N/A	N/A
ARG	Argument	A	A	A	A	A
COM	Compare	A	A	A	A	A
DR	Decrement Register	A	A	A	A	A
DRP	Drop	A	A	A	N/A	A
GET	Get	A	A	A	A	A
PUT	Put	A	A	A	A	A
RC	Recall	A	A	A	A	A
RND	Random	A	A	A	N/A	A
ST	Store	A	A	A	A	A
RRS	Read Rear Switch	A	A	N/A	N/A	N/A
CLK	Clock	A	A	A	N/A	A

Level II Command Comparisons by Player

Mnemonic	Command	LD-V8000	LD-V6000A	LD-V6000	LD-V3000	PR7820-3
	Input Processing Commands					
IN	Input	A	A	A	A	A
FIN	Input with Function Keys	A	A	A	N/A	A
TIN	Input with Timeout	A	A	A	A	A
FTI	Input with Function & Timeout	A	A	A	N/A	A
DIN	Digit Input	A	A	A	A	A
BIN	Binary Input	A	A	A	N/A	A
IIN	Interrupt Input	A	A	N/A	N/A	N/A
	Program Execution Control Commands					
BR	Branch	A	A	A	A	A
BRF	Branch on Failure	A	A	A	A	A
JMP	Jump	A	A	A	A	A
NE	No Entry	A	A	A	A	A
	Flag Set Commands					
BIE	Binary Out Enable	N/A (e)	A	A	N/A	Changed
BID	Binary Out Disable	N/A (e)	A	A	N/A	Changed
RCE	Remote Control Unit Enable	A	A	A	N/A	N/A
RCD	Remote Control Unit Disable	A	A	A	N/A	N/A
AIE	Antenna in Enable	N/A (e)	A	A	N/A	N/A
AID	Antenna in Disable	N/A (e)	A	A	N/A	N/A
SCS	Set Player Control Switch	A	A	N/A	N/A	N/A
	Transmit Commands					
TFN	Transmit Frame Number	N/A	A	A	N/A	A
TPA	Transmit Program Address	N/A	A	A	N/A	A
TS	Transmit Status	N/A	A	A	N/A	A
TES	Transmit Extended Status	N/A	A	A	N/A	N/A
STP	Set Transmit Pointer	A	A	A	N/A	A
ITP	Increment Transmit Pointer	A	A	A	N/A	A
DTP	Decrement Transmit Pointer	A	A	A	N/A	A
TM	Transmit Memory	A	A	A	N/A	A
ITM	Increment & Transmit Memory	A	A	A	N/A	A
DTM	Decrement & Transmit Memory	A	A	A	N/A	A
TP0	Transmit F8 Out Port 0	N/A	N/A	N/A	N/A	A
TP1	Transmit F8 Out Port 1	N/A	N/A	N/A	N/A	A
TP5	Transmit F8 Out Port 5	N/A	N/A	N/A	N/A	A
TIA	Transmit F8 In Port A	N/A	N/A	N/A	N/A	A

Level II Command Comparisons by Player

Mnemonic	Command	LD-V8000	LD-V6000A	LD-V6000	LD-V3000	PR7820-3
	Transmit Commands (cont.)					
TIB	**Transmit F8 In Port B**	**N/A**	**N/A**	**N/A**	**N/A**	**A**
TIC	**Transmit F8 In Port C**	**N/A**	**N/A**	**N/A**	**N/A**	**A**
TR	**Transmit F8 Register**	**N/A**	**N/A**	**N/A**	**N/A**	**A**
TID	**Transmit Player ID**	**N/A**	**A**	**N/A**	**N/A**	**N/A**
SRP	**Set Transmit Register Pointer**	**N/A**	**A**	**N/A**	**N/A**	**N/A**
TRG	**Transmit Register**	**N/A**	**A**	**N/A**	**N/A**	**N/A**
ITR	**Increment Transmit Register**	**N/A**	**A**	**N/A**	**N/A**	**N/A**
DTR	**Decrement Transmit Register**	**N/A**	**A**	**N/A**	**N/A**	**N/A**
IRR	**Increment TR Pointer**	**N/A**	**A**	**N/A**	**N/A**	**N/A**
DRR	**Decrement TR Pointer**	**N/A**	**A**	**N/A**	**N/A**	**N/A**
TDS	**Transmit Disc Status**	**N/A**	**A**	**N/A**	**N/A**	**N/A**
TLS	**Transmit Loading Status**	**N/A**	**A**	**N/A**	**N/A**	**N/A**
TAC	**Transmit Acknowledge Status**	**N/A**	**A**	**N/A**	**N/A**	**N/A**
TCN	**Transmit Chapter Number**	**N/A**	**A**	**N/A**	**N/A**	**N/A**
TSS	**Transmit Switch Status**	**N/A**	**A**	**N/A**	**N/A**	**N/A**
	Video Buffer Control Commands					
SMS	**Set Memory Control Switch**	**A**	**N/A**	**N/A**	**N/A**	**N/A**
SRM	**Select Read Memory**	**A**	**N/A**	**N/A**	**N/A**	**N/A**
MWE	**Memory Write Enable**	**A**	**N/A**	**N/A**	**N/A**	**N/A**
MWD	**Memory Write Disable**	**A**	**N/A**	**N/A**	**N/A**	**N/A**

Appendix B: Alphabetical Listing of Level II Commands Available on the LD-V8000

APPENDIX

B

LD-V8000

LEVEL II

USER'S MANUAL

Programmer's Reference Guide

Alphabetical Listing of Level II Commands for LD-V8000

Note: Some of these commands are specific to the LD-V8000, or a particulat version of the LD-V8000. For example, the Video Buffer commands are specific to the LD-V8000-01 with "later" EPROMS. In some cases, specific EPROM versions may be necessary. PIONEER makes no assurances of compatibility with any particular videodisc player model, past, present, or future. **Test all programs carefully.**

Command Name	Mnemonic	Hex Code	RCU Button	Argument	Page
Add	ADD	02	Play, 0, 2	(Integer)	4•36
Argument	ARG	0A	Play, 0, A	(Register #)	4•40 .
Audio 1	A1	F4	AUDIO 1/L	(Integer)	4•11
Audio 2	A2	FC	AUDIO 2/R	(Integer)	4•11
AutoStop	AS	F3	AUTOSTOP	(Disc Location)	4•23
AXX	AFF	A0	Play, A, 0	(Ignored)	4•13
"	AFI	A5	Play, A, 5	(Ignored)	"
"	AFN	A1	Play, A, 1	(Ignored)	"
"	AFT	A4	PLAY, A, 4	(Ignored)	"
"	AIF	AA	PLAY, A, A	(Ignored)	"
"	AIN	AB	PLAY, A, B	(Ignored)	"
"	AIT	AE	PLAY, A, E	(Ignored)	"
"	ANF	A2	PLAY, A, 2	(Ignored)	"
"	ANI	A7	PLAY, A, 7	(Ignored)	"
"	ANN	A3	PLAY, A, 3	(Ignored)	"
"	ANT	A6	PLAY, A, 6	(Ignored)	"
"	ATF	A8	PLAY, A, 8	(Ignored)	"
"	ATI	AD	PLAY, A, D	(Ignored)	"
"	ATN	A9	PLAY, A, 9	(Ignored)	"
"	ATT	AC	PLAY, A, C	(Ignored)	"
Binary Input	BIN	17	PLAY, 1, 7	(Ignored)	4•55
Blink	BLK	2D	PLAY, 2, D	(Integer)	4•20
Branch	BR	CF	BRANCH	(Address)	4•33
Branch on Failure	BRF	07	PLAY, 0, 7	(Address)	4•34
Character Generator Disable	CGD	E1	PLAY, E, 1	(Ignored)	4•17
Character Generator Enable	CGE	E0	PLAY, E, 0	(Ignored)	4•17

Alphabetical Listing of Level II Commands for LD-V8000 (cont.)

Command Name	Mnemonic	Hex Code	RCU Button	Argument	Page
Clear Blink	CLB	2E	PLAY, 2, E	(Integer)	4•20
Clear User Display	CLD	2C	PLAY, 2, C	(Integer)	4•19
Clock	CLK	16	PLAY, 1, 6	(Ignored)	4•47
Compare	COM	04	PLAY, 0, 4	(Integer)	4•41
Decrement Register	DR	F0	DEC REG	(Register #)	4•42
Decrement Transmit Pointer	DTP	DA	PLAY, D, A	(Ignored)	4•60
Decrement & Transmit Memory	DTM	DE	PLAY, D, E	(Ignored)	4•59
Digital Audio	DAD	82	PLAY, 8, 2	(Integer)	4•14
Divide	DIV	21	PLAY, 2, 1	Integer	4•37
Digit Input	DIN	1E	PLAY, 1, E	(Ignored)	4•54
Display	DI	F1	DISPLAY	(Integer)	4•17
Drop	DRP	1D	PLAY, 1, D	Integer	4•43
End Programming Mode	—	EF	END		
Fast Speed Set	FS	EC	SPEED SET (Fast)	(Integer)	4•27
Function Key Input	FIN	18	PLAY, 1, 8	Integer	4•50
Function Key Input &TimeOut	FTI	19	PLAY, 1, 9	Integer	4•52
Get	GET	08	PLAY, 0, 8	(Integer)	4•38
Halt Program	H	BF	HALT		4•35
Increment Transmit Pointer	ITP	D9	PLAY, D, 9	(Ignored)	4•60
Increment & Transmit Memory	ITM	DD	PLAY, D, D	(Ignored)	4•59
Input	IN	F8	INPUT	Integer	4•48
Interrupt Input	IIN	20	PLAY, 2, 0	(Integer)	4•56
Jump	JMP	0B	PLAY, 0, B	(Program Address)	4•34
Load Program	L	CC	PLAY, C, C		4•5
Memory Write Disable	MWD	87	PLAY, 8, 7	(Integer)	4•63
Memory Write Enable	MWE	86	PLAY, 8, 6	(Integer)	4•62
Moving Load	L	CC	PLAY, C, C	Page Number	4•6

Alphabetical Listing of Level II Commands for LD-V8000 (cont.)

Command Name	Mnemonic	Hex Code	RCU Button	Argument	Page
Moving Partial Load	PLD	0C	PLAY, 0, C	Page Number	4•9
Multi-Forward	MF	F2	MULTI-FWD	(Disc Location)	4•28
Multi-Reverse	MR	FA	MULTI-REV	(Disc Location)	4•28
Multiply	MUL	22	PLAY, 2, 2	Integer	4•37
No Entry	NE	FF	PLAY, F, F	— passed on —	4•35
Page	PAG	11	PLAY, 1, 1	(Page Number)	4•4
Partial Load	PLD	0C	PLAY, 0, C		4•8
Pause	PAU	OD	PLAY, O, D	Integer	4•26
Play	P	FD	PLAY, F, D	(Disc Location)	4•22
Programming Mode	—	DF	PROGRAM	(Program Address)	
Put	PUT	09	PLAY, 0, 9	Register #	4•38
Read Rear Switch	RRS	10	PLAY, 1, 0	—	4•46
Recall	RC	7F	RECALL	(Register #)	4•39
Reject	RJ	F9	PLAY, F, 9	(Ignored)	4•22
Remote Control Unit Disable	RCD	13	PLAY, 1, 3	(Ignored)	4•57
Remote Control Unit Enable	RCE	12	PLAY, 1, 2	(Ignored)	4•57
Random	RND	05	PLAY, 0, 5	(Ignored)	4•43
Run Program	—	CF	RUN	(Program Address)	
Search	SC	F7	SEARCH	(Disc Location)	4•24
Select Read Memory	SRM	85	PLAY, 8, 5	(Integer)	4•62
Set Background Color	SBC	88	PLAY, 8, 8	(Integer)	4•21
Set Chapter Mode	SCM	8C	PLAY, 8, C	—	4•31
Set Frame Mode	SFM	8E	PLAY, 8, E	—	4•30
Set Video Memory Switch	SMS	84	PLAY, 8, 4	(Integer)	4•61
Set Special Control Switches	SCS	8A	PLAY, 8, A	Integer	4•58
Set Time Mode	STM	8D	PLAY, 8, D	—	4•30
Set Transmit Pointer	STP	E8	PLAY, E, 8	(Progam Address)	4•59

Alphabetical Listing of Level II Commands for LD-V8000 (cont.)

Command Name	Mnemonic	Hex Code	RCU Button	Argument	Page
Set Still Mode	SSM	8B	PLAY, 8, B	(Integer)	4•32
Set User Display	SUD	2B	PLAY, 2, B	Integer	4•19
Slow Speed Set	SS	ED	SPEED SET (Slow)	(Integer)	4•26
Step Forward	SF	F6	STEP FWD	(Ignored)	4•29
Step Reverse	SR	FE	STEP REV	(Ignored)	4•29
Stop	Stop	FB	STOP	(Integer)	
Clear Blink	CLB	2E	PLAY, 2, E	(Integer)	4•20
Store	ST	F5	STORE	(Integer)	4•44
Subtract	SUB	03	PLAY, 0, 3	(Integer)	4•36
Timed Input	TIN	0E	PLAY, 0, E	Integer	4•51
Track Jump Forward	TJF (*80)	80	PLAY, 8, 0	Integer	4•32
Track Jump Reverse	TJR (*81)	81	PLAY, 8, 1	Integer	4•32
Transmit Memory	TM	DC	PLAY, D, C	(Ignored)	4•59
Video Off	VFF	1C	PLAY, 1, C	(Ignored)	4•16
Video On	VON	1B	PLAY, 1, B	(Ignored)	4•16
Wait	WAIT	FB	STOP	(Integer)	4•25

Appendix C: Hex Code Matrix of Level II Commands Available on the LD-V8000

APPENDIX

C

LD-V8000

LEVEL II

USER'S MANUAL

Programmer's Reference Guide

LD-V8000 Matrix of Level II Hex Codes

LD-V8000 Level II Command Table

High Order Hex Digit

Low Order Hex Digit

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0		RRS	IIN						TJF		AFF				CGE	DR	0
1		PAG	DIV						TJR		AFN				CGD	DI	1
2	ADD	RCE	MUL						DAD		ANF					MF	2
3	SUB	RCD									ANN					AS	3
4	COM								SMS		AFT					A1	4
5	RND								SRM		AFI					ST	5
6		CLK							MWE		ANT					SF	6
7	BRF	BIN							MWD		ANI					SC	7
8	GET	FIN							SBC		ATF				STP	IN	8
9	PUT	FTI									ATN			ITP		RJ	9
A	ARG								SCS		AIF			DTP		MR	A
B	JMP	VON	SUD						SSM		AIN					W	B
C	PLD	VFF	CLD						SCM		ATT		L	TM	FS	A2	C
D	PAU	DRP	BLK						STM		ATI			ITM	SS	P	D
E	TIN	DIN	CLB						SFM		AIT			DTM		SR	E
F	1	7	4	0	3	9	6	RC	2	8	5	H	BR			NE	F

NOTE: The High Order Hex Digit is entered first, the Low Order Hex Digit is entered second. For example, the Hex code for SEARCH (SC) is F7.

Appendix D: Character Generator: Table of Hex Codes

APPENDIX

D

LD-V8000

LEVEL II

USER'S MANUAL

Programmer's Reference Guide

LD-V8000 Character Generator: Table of Hex Codes

"ASCII" Character Codes for the User Display Lines of the LD-V8000

The Hex code and corresponding graphic for the LD-V8000's Character Generator:

Hex Code:	20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
Character:	sp	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
Hex Code:	30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
Character:	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
Hex Code:	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
Character:	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Hex Code:	50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
Character:	P	Q	R	S	T	U	V	W	X	Y	Z	←	¥	→	(*)	_
Hex Code:	60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
Character:	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
Hex Code:	70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
Character:	p	q	r	s	t	u	v	w	x	y	z	↑	|	↓	(*)	–
Hex Code:	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
Character:	Ç	ü	é	â	ä	à	å	ç	ê	ë	è	ï	î	ì	Ä	Â
Hex Code:	90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
Character:	É	æ	Æ	ô	ö	ò	û	ù	ÿ	Ö	Ü	¢	£	ß	Pt	ƒ

(*) Hex Code:	5E	7E
Character:	White Block	"Black" Block

For more information on using the LD-V8000 Character Generator Overlay in Level II programs, see ***Section 4.2.3, Video Control Commands***, in this manual. See the example provided with the CLB command on page 4-20. Also see **Appendix F**, **Example #3, *Three selectable motion segments with simple instruction menu, using character overlay.***

Appendix E: Numbers and Their Hex Code Equivalents

APPENDIX

E

LD-V8000

LEVEL II

USER'S MANUAL

Programmer's Reference Guide

Numeric Digit Hex Codes

Digit	Hex Code	Decimal Value
0	3F	63
1	0F	15
2	8F	143
3	4F	79
4	2F	47
5	AF	175
6	6F	111
7	1F	31
8	9F	159
9	5F	95

When downloading Level II instructions to the LD-V8000 via the RS-232 port, command argument digits are each sent as a two-character code, the Hex Code shown above.

Since the numeric digit buttons on the RCU cause the player to "receive" the hex codes shown above, they can be used directly in *Programming Mode* to enter digits as part of command arguments.

The Decimal Values are sometimes used in a program as arguments for a COMPARE command following the use of a BIN command.

See ***Chapter 3*** of this manual for more details on entering code and data into the player's memory.

Appendix F: Sample Flow Charts & Level II Program Examples — RCU entry

APPENDIX

F

LD-V8000

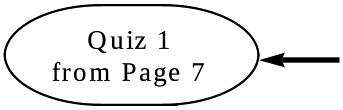
LEVEL II

USER'S MANUAL

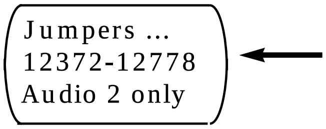
Programmer's Reference Guide

Example Flowchart Symbols

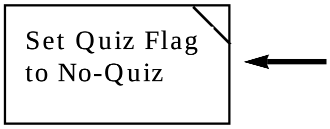
The symbols below (or similar ones) could be used in a flow chart to aid the design process and to document the interactive aspects of an Audio / Video presentation. The flowchart is a graphic representation of both the order of information presentation and of how the interactive control program responds to viewer inputs. The flowchart is the interactive "story board", which should be used to guide subsequent scripting and programming. Consistency, clarity, and completeness are more important than the symbols used. When in doubt, use text descriptions of your intentions.



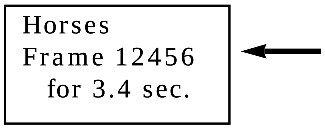
A terminal symbol, indicating the starting or ending point of a flowchart, or a continuation of the flowchart from another page.



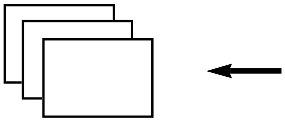
A motion sequence, with or without sound. Possibly, a still image with narration or music.



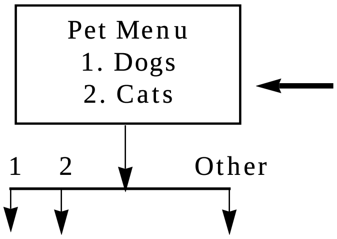
Special instructions, scoring, or any computations the control program may be required to do. Flags may be set, values recorded, etc.



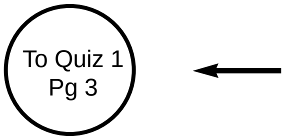
A still frame. Include how long to wait at the still before continuing to the next block in the flowchart.



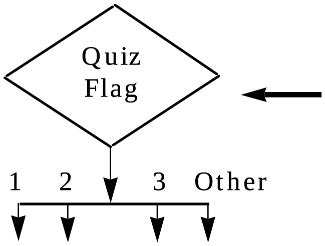
A series of stills. Specify how the user proceeds forward through the stills. Is it by time steps only? Can the user back up in the sequence? What happens when the user goes "past" the end still, or tries to back up "past" the beginning still?



A still frame Menu. All possible viewer inputs are accounted for, and a timeout, if any, is specified. The branch paths, and the input values that cause the path to be taken, are clearly shown.



Indicate a continuation of the flow chart on another page. Include the page number.



A decision point, indicating the value or condition to be tested, the branching that results, and the values that caused the branching.

Level II Program Examples

Here are four Level II examples for use with the LD-V8000 videodisc player. They were developed to serve several purposes:

1. With a flow chart and several sentence introduction describing each program and then the program itself in the format below, these examples explain how to enter Level II programs into the memory of the LD-V8000 using the remote control unit, the RU-V6000T.

2. These are actual working examples that can be used at trade shows or in training sessions by users, if they replace the frame numbers with their own frame numbers. We have used leading zeros to keep the program addresses the same no matter what frame numbers are entered.

We are assuming these examples will be developed with CAV videodiscs. An * indicates the Play button in the RCU input sequence section.

Format describing Input of Level II Programs using RCU button presses:

Address	Argument	Command	RCU Input Sequence of button presses	Comments

NOTE: The Frame numbers used in the following examples correspond to segments on the ***Pioneer LD-V8000 Demonstration Disc.***

Still Frame	frame 00210	Pioneer Logo
Segment One	frame 02113 to frame 03493	Introduction
Segment Two	frame 04450 to frame 05117	Frame Store
Segment Three	frame 05619 to frame 06570	Digital Audio
Segment Four	frame 06700 to frame 07820	Fast Search

These segments may be replaced with others for use with different CAV videodiscs.

The following examples can be expanded upon to add additional segments, etc. after the concepts illustrated in the flow charts are understood. Modifications to the programs must also take into account resulting changes to program locations. For more details about sending program code to the LD-V8000's RAM, Level II button press code entry or Level II command descriptions please refer to ***LD-V8000 Level II, User's Manual, Programmer's Reference Guide***, *Pioneer Technical Publication # 114 Ver. 2.0 7/92.*

Sample Level II Program — RCU Entry

The buttons "A" through "F" and "Input" are marked on the RU-V6000T keypad, but their equivalent buttons are noted below:

Play = * ; Pressing the Play Button prepares player to receive Hex input

RCU Button	Hex equivalent	RCU Button	Hex equivalent
Autostop	A	Search	D
StepRev	B	Multi-Rev	E
StepFwd	C	Multi-Fwd	F

Scan-Rev = "Input" command, in Hex mode.

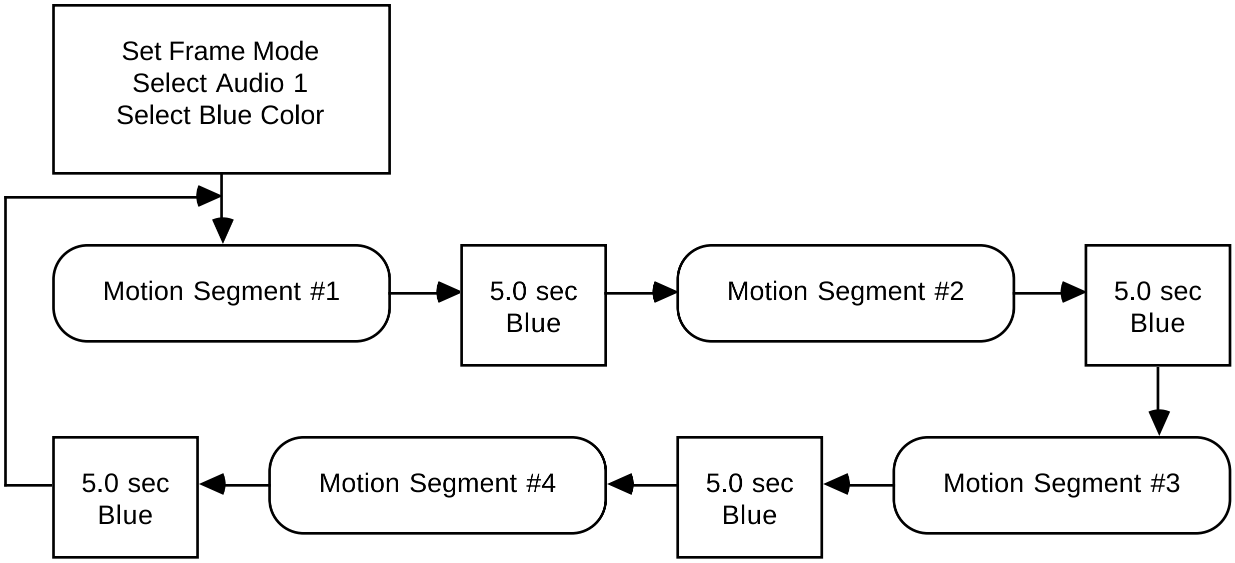
Level II Programming Examples:

Example #1

A continuously repeating series of four motion segments with a five second blue squelch screen between each segment. Start up, set frame mode, audio channel 1 on, search to the first motion segment and play it, flash a 5-second blue squelch screen, search to second motion segment and play it....and so forth through four segments. After segment four and the five-second blue squelch is completed, branch back to segment 1.

Segment One is Frame 02113 to 03493; Segment Two is 04450 to 05117. Segment Three is 05619 to 06570 and Segment Four is 06700 to 07820.

Flow Chart of Example #1



Sample Level II Program — RCU Entry

Example #1 (cont.)

Since the commands used here are fairly “simple” commands, this program is fairly easy to understand, even without reading the ***LD-V8000 Level II User’s Manual/Programmer’s Reference Guide***. Here is the program in a format as it might be made by a programmer for a compiler:

```
                         SFM                            ; Set Frame Mode
                       1 Audio1       0 Audio2          ; Use Audio 1 only
                       1 SBC                            ; Force Blue Color Background

Loop:       2113 Search     Von     3493 Autostop    Voff     50 Wait      ; Segment #1
            4450 Search     Von     5117 Autostop    Voff     50 Wait      ; Segment #2
            5619 Search     Von     6570 Autostop    Voff     50 Wait      ; Segment #3
            6700 Search     Von     7820 Autostop    Voff     50 Wait      ; Segment #4
                       Loop Branch
```

Example #1 in “RCU entry” format:

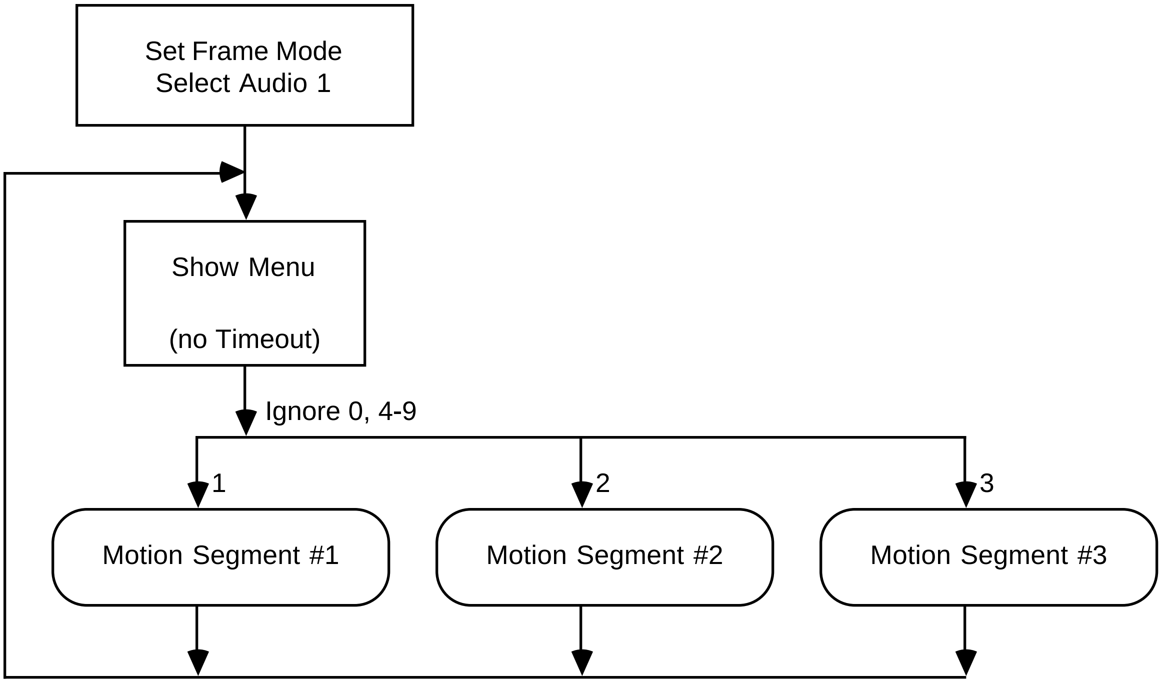
Address	Arg.	Command	RCU Input	Comments
0		SFM	*, 8, E	Set Frame Mode
1	1	Audio1	1, Audio1	Turn on Audio 1
3	0	Audio2	0, Audio2	Turn off Audio 2
5	1	SBC	1, *, 8, 8	Select Blue Color
7	000		000	Filler
10	02113	Search	02113 Search	Start Segment #1
16		VON	*, 1, B	Turn video On
17	03493	Autostop	03493 Autostop	Play to the segment end
23		VOFF	*, 1, C	Turn on Blue Screen
24	00050	Wait	00050 Stop	Wait for 5.0 seconds
30	04450	Search	04450 Search	Start Segment #2
36		VON	*, 1, B	Turn video On
37	05117	Autostop	05117 Autostop	Play to the segment end
43		VOFF	*, 1, C	Turn on Blue Screen
44	00050	Wait	00050 Stop	Wait for 5.0 seconds
50	05619	Search	05619 Search	Start Segment #3
56		VON	*, 1, B	Turn video On
57	06570	Autostop	06570 Autostop	Play to the segment end
63		VOFF	*, 1, C	Turn on Blue Screen
64	00050	Wait	00050 Stop	Wait for 5.0 seconds
70	06700	Search	06700 Search	Start Segment #4
76		VON	*, 1, B	Turn video On
77	07820	Autostop	07820 Autostop	Play to the segment end
83		VOFF	*, 1, C	Turn on Blue Screen
84	00050	Wait	00050 Stop	Wait for 5.0 seconds
90	10	Branch	10 Branch	Loop to Segment #1

Sample Level II Program — RCU Entry

Example #2

Three selectable motion segments. Start up, set frame mode, audio channel 1 on, hold a still frame until RCU button 1, 2 or 3 is pressed to select one of three motion segments. At the end of each selected segment, branch back to introductory still frame. Introductory still frame is at Frame 00210. Segment One is 4450 to 05117; Segment Two is 05619 to 06570 and Segment Three is 06700 to 07820.

Flow Chart of Example #2



It is necessary to understand the format of the INPUT command to be able to follow this program. The other commands used here are fairly “simple” commands, so this program is fairly easy to understand. Here is the program in a format as it might be made by a programmer for a compiler:

```
                         SFM                     ; Set Frame Mode
                       1 Aud1       0 Aud2       ; Use Audio 1 only

Menu:     2113 Search
Ignore:        4 Input          Ignore Branch              ; Key 0, ignore it

          4450 Search    5117 Autostop    Menu Branch          ; Key 1, Segment #1
          5619 Search    6570 Autostop    Menu Branch          ; Key 2, Segment #2
          6700 Search    7820 Autostop    Menu Branch          ; Key 3, Segment #3

                                Ignore Branch              ; Keys 4-9, ignore
```

Sample Level II Program — RCU Entry

Example #2 in "RCU entry" format:

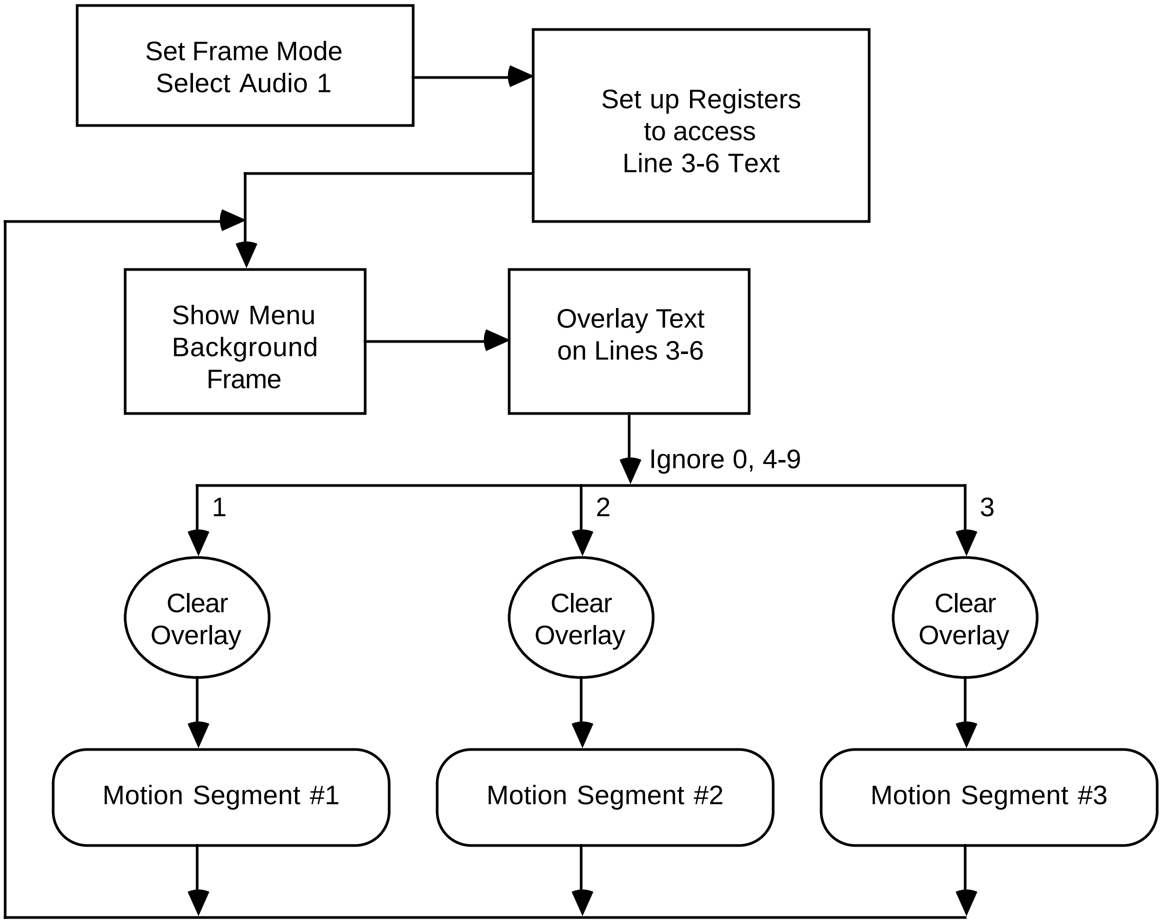
Address	Arg.	Command	RCU Input	Comments
0		SFM	*, 8, E	Set Frame Mode
1	1	Audio1	1, Audio1	Turn on Audio 1
3	0	Audio2	0, Audio2	Turn off Audio 2
5	02113	Search	02113 Search	Show Menu Frame
11	0004	Input	0004 Input	Wait for Input
16	011	Branch	011 Branch	Key 0: Ignore it
20	04450	Search	04450 Search	Key 1: Start Segment #1
26	05117	Autostop	05117 Autostop	Play to the segment end
32	05	Branch	05 Branch	Return to the Menu
35	05619	Search	05619 Search	Key 2: Start Segment #2
41	06570	Autostop	06570 Autostop	Play to the segment end
47	05	Branch	05 Branch	Return to the Menu
50	06700	Search	06700 Search	Key 3: Start Segment #3
56	07820	Autostop	07820 Autostop	Play to the segment end
62	05	Branch	05 Branch	Return to the Menu
65	11	Branch	11 Branch	Ignore buttons 4-9

Sample Level II Program

Example #3
Three selectable motion segments with simple instruction menu, using character overlay. Use same example as above only include this basic instruction menu on a still frame at frame 00765:

(Line 3)		Select a Topic
(Line 4)	1	Frame Store
(Line 5)	2	Digital Audio
(Line 6)	3	Fast Search

Flow Chart of Example #3



Sample Level II Program

Example #3   (cont.)

It is helpful to understand the format of the INPUT command, the use of CLD (Clear User Display) and SUD (Set User Display) to overlay text on the screen, and the use of RECALL and STORE to manipulate Registers to be able to follow this program. The other commands used here are fairly “simple” commands, so this program is moderately easy to understand. Here is the program in an format as it might be made by a programmer for a compiler:

```
                     SFM                       ; Set Frame Mode
                   1 Aud1       0 Aud2         ; Use Audio 1 only

                   5 Recall     ; Setup Registers 5 - 8 with Text addresses
                   100 Store   120 Store   140 Store   160 Store

Menu:       765 Search      CLD                           ; Show background frame
              5 Recall     3 SUD     4 SUD     5 SUD     6 SUD  ; Put up Text

Ignore:        4 Input          Ignore Branch             ; Key 0, ignore it

      CLD    4450 Search    5117 Autostop    Menu Branch     ; Key 1, Segment #1
      CLD    5619 Search    6570 Autostop    Menu Branch     ; Key 2, Segment #2
      CLD    6700 Search    7820 Autostop    Menu Branch     ; Key 3, Segment #3

                                Ignore Branch             ; Keys 4-9, ignore
```

Then, Example #3 in “RCU entry” format:

Address	Arg.	Command	RCU Input	Comments
0		SFM	*, 8, E	Set Frame Mode
1	1	Audio1	1, Audio1	Turn on Audio 1
3	0	Audio2	0, Audio2	Turn off Audio 2
5	5	Recall	5 Recall	Point to Register 5
7	100	Store	100 Store	Line 3 text address in Reg 5
11	120	Store	120 Store	Line 4 text address in Reg 6
15	140	Store	140 Store	Line 5 text address in Reg 7
19	160	Store	160 Store	Line 6 text address in Reg 8
23	00765	Search	00765 Search	Menu Background Frame
29		CLD	*, 2, C	Clear Character Overlay
30	5	Recall	5 Recall	
32	3	SUD	3, *, 2, B	Display Line 3 text
34	4	SUD	4, *, 2, B	Display Line 4 text
36	5	SUD	5, *, 2, B	Display Line 5 text
38	6	SUD	6, *, 2, B	Display Line 6 text

Sample Level II Program

Example #3 in “RCU entry” format (cont.) :

Address	Arg.	Command	RCU Input	Comments
40	4	Input	4 Input	Wait for Inputs 0-9
42		Branch	40 Branch	Key 0: Ignore it
45		CLD	*, 2, C	Clear Character Overlay
46	04450	Search	04450 Search	Key 1: Start Segment #1
52	05117	Autostop	05117 Autostop	Play to the segment end
58	23	Branch	23 Branch	Return to the Menu
61		CLD	*, 2, C	Clear Character Overlay
62	05619	Search	05619 Search	Key 2: Start Segment #2
68	06570	Autostop	06570 Autostop	Play to the segment end
74	23	Branch	23 Branch	Return to the Menu
77		CLD	*, 2, C	Clear Character Overlay
78	06700	Search	06700 Search	Key 3: Start Segment #3
84	07820	Autostop	07820 Autostop	Play to the segment end
90	23	Branch	23 Branch	Return to the Menu
93	40	Branch	40 Branch	Ignore buttons 4-9
96	0000		0000	
100		(space)	*, 2, 0	Line 3 text
101		S	*, 5, 3	“ Select a Topic ”
102		e	*, 6, 5	
103		l	*, 6, C	
104		e	*, 6, 5	
105		c	*, 6, 3	
106		t	*, 7, 4	
107		(space)	*, 2, 0	
108		a	*, 6, 1	
109		(space)	*, 2, 0	
110		T	*, 5, 4	
111		o	*, 6, F	
112		p	*, 7, 0	
113		i	*, 6, 9	
114		c	*, 6, 3	
115		(space)	*, 2, 0	
116		(space)	*, 2, 0	
117		(space)	*, 2, 0	
118		(space)	*, 2, 0	
119		(space)	*, 2, 0	

Example #3 in "RCU entry" format (cont.) :

Address	**Arg.**	**Command**	**RCU Input**	**Comments**
120		1	*, 3, 1	Line 4 text
121		(space)	*, 2, 0	"1 Frame Store "
122		(space)	*, 2, 0	
123		(space)	*, 2, 0	
124		F	*, 4, 6	
125		r	*, 7, 2	
126		a	*, 6, 1	
127		m	*, 6, D	
128		e	*, 6, 5	
129		(space)	*, 2, 0	
130		S	*, 5, 3	
131		t	*, 7, 4	
132		o	*, 6, F	
133		r	*, 7, 2	
134		e	*, 6, 5	
135		(space)	*, 2, 0	
136		(space)	*, 2, 0	
137		(space)	*, 2, 0	
138		(space)	*, 2, 0	
139		(space)	*, 2, 0	
140		2	*, 3, 2	Line 5 text
141		(space)	*, 2, 0	"2 Digital Audio "
142		(space)	*, 2, 0	
143		(space)	*, 2, 0	
144		D	*, 4, 4	
145		i	*, 6, 9	
146		g	*, 6, 7	
147		i	*, 6, 9	
148		t	*, 7, 4	
149		a	*, 6, 1	
150		l	*, 6, C	
151		(space)	*, 2, 0	
152		A	*, 4, 1	
153		u	*, 7, 5	
154		d	*, 6, 4	
155		i	*, 6, 9	
156		o	*, 6, F	
157		(space)	*, 2, 0	
158		(space)	*, 2, 0	
159		(space)	*, 2, 0	

Example #3 in "RCU entry" format (cont.) :

Address	Arg.	Command	RCU Input	Comments
160		3	*, 3, 3	Line 6 text
161		(space)	*, 2, 0	"3 Fast Search "
162		(space)	*, 2, 0	
163		(space)	*, 2, 0	
164		F	*, 4, 6	
165		a	*, 6, 1	
166		s	*, 7, 3	
167		t	*, 7, 4	
168		(space)	*, 2, 0	
169		S	*, 5, 3	
170		e	*, 6, 5	
171		a	*, 6, 1	
172		r	*, 7, 2	
173		c	*, 6, 3	
174		h	*, 6, 8	
175		(space)	*, 2, 0	
176		(space)	*, 2, 0	
177		(space)	*, 2, 0	
178		(space)	*, 2, 0	
179		(space)	*, 2, 0	

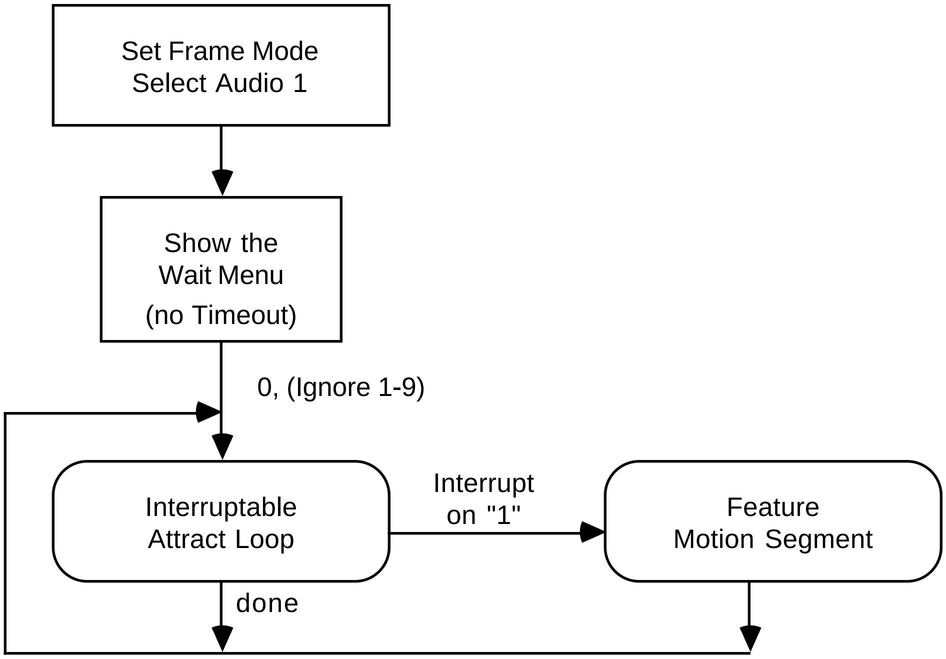
Sample Level II Program

Example #4

A selectable continuously repeating Attract Loop with a selectable Main Feature. Set on screen function switches for Load from Memory. Then, when the disc is inserted in the drawer, and PLAY is pressed, the Level II program in memory is automatically executed.

The Program starts by setting frame mode and audio channel 1 on, then searches to a still frame 00210, and waits for only one input to start attract loop....Press 0 to start Attract Loop (frame 02113 to 03493). The attract loop repeats continually until 1 is pressed to play the Main Feature Motion Segment (frame 05619 to 06564). At the end of the Main Feature, the program automatically branches back to the continuously repeating attract loop.

Flow Chart of Example #4



This flow chart looks simple, but actually the program is difficult to understand unless one well understands the IIN (Interrupt Input) and COM (Compare) commands.

Sample Level II Program

Example #4 (cont.)

Here is the program in a “high level” programming format as it might be made for a compiler:

```
                           SFM                            ; Set Frame Mode
                         1 Aud1          0 Aud2           ; Use Audio 1 only

Menu:          210 Search
Ignore:            1 Input               Attract Branch                   ; Key 0, start attract
                                         Ignore Branch                    ; Key 1-9, ignore

Attract:      2113 Search
Continue:          2 IIN                                                  ; Set Interrupt Mode
                         DIN Get         CheckKey Branch                  ; an interrupt happened

         3493 Autostop                                                    ; Play Attract to end
                               0 IIN                                      ; Cancel Interrupt Mode
                         Attract Branch                                   ; Loop the Attract

CheckKey:                1 Compare               Continue Branch          ; Key # > 1, continue

             5619 Search          6570 Autostop         Attract Branch            ; Key 1, Play Feature

                                                 Continue Branch          ; Key # < 1, continue
```

Sample Level II Program

Example #4 in "RCU entry" format:

Address	Arg.	Command	RCU Input	Comments
0		SFM	*, 8, E	Set Frame Mode
1	1	Audio1	1, Audio1	Turn on Audio 1
3	0	Audio2	0, Audio2	Turn off Audio 2
5	00210	Search	00210 Search	Wait to Start Frame
11	1	Input	1 Input	Wait for Input
13	19	Branch	19 Branch	Key 0: Start the Attract
16	11	Branch	11 Branch	Key 1-9: Ignore the key
19	02113	Search	02113 Search	Start of the Attract Loop
25	2	IIN	2, *, 2, 0	Interrupt on Key 0 or 1
27	DIN	GET	*, 1, E, *, 0, 8	Capture the pressed key #
29	43	Branch	43 Branch	Go play the Feature
32	03493	Autostop	03493 Autostop	Play to the Attract end
38	0	IIN	0, *, 2, 0	Turn off Interrupt Mode
40	19	Branch	19 Branch	Repeat the attract Loop
43	1	COM	1, *, 0, 4	Was the "1" key pressed ?
45	25	Branch	25 Branch	Key > 1, continue Attract
48	05619	Search	05619 Search	Key = 1: Start Feature
54	06564	Autostop	06564 Autostop	Play to the Feature end
60	19	Branch	19 Branch	Restart the Attract loop
63	25	Branch	25 Branch	Key < 1, continue Attract

Appendix G: Flow Charts and Level II Program Examples — Programming

APPENDIX

G

LD-V8000

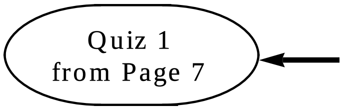
LEVEL II

USER’S MANUAL

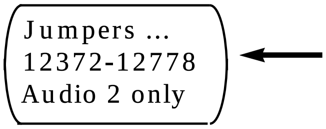
Programmer’s Reference Guide

Example Flowchart Symbols

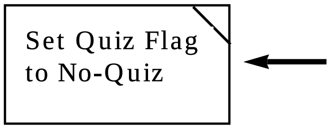
The symbols below (or similar ones) could be used in a flow chart to aid the design process and to document the interactive aspects of an Audio / Video presentation. The flowchart is a graphic representation of both the order of information presentation and of how the interactive control program responds to viewer inputs. The flowchart is the interactive "story board", which should be used to guide subsequent scripting and programming. Consistency, clarity, and completeness are more important than the symbols used. When in doubt, use text descriptions of your intentions.



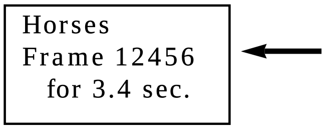
A terminal symbol, indicating the starting or ending point of a flowchart, or a continuation of the flowchart from another page.



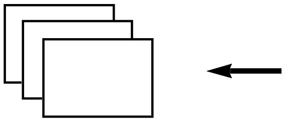
A motion sequence, with or without sound. Possibly, a still image with narration or music.



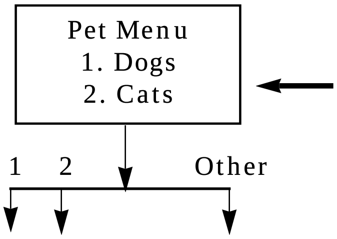
Special instructions, scoring, or any computations the control program may be required to do. Flags may be set, values recorded, etc.



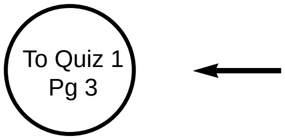
A still frame. Include how long to wait at the still before continuing to the next block in the flowchart.



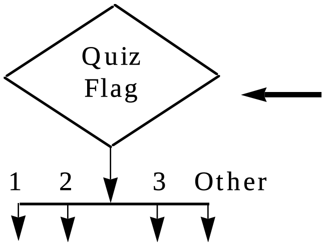
A series of stills. Specify how the user proceeds forward through the stills. Is it by time steps only? Can the user back up in the sequence? What happens when the user goes "past" the end still, or tries to back up "past" the beginning still?



A still frame Menu. All possible viewer inputs are accounted for, and a timeout, if any, is specified. The branch paths, and the input values that cause the path to be taken, are clearly shown.



Indicate a continuation of the flow chart on another page. Include the page number.



A decision point, indicating the value or condition to be tested, the branching that results, and the values that caused the branching.

Sample Level II Pogramming Code

Some Necessary Definitions

The Examples in this Appendix were written to help explain how to create Level II programs. To simplify these examples, they were written in symbolic assembler form. This means that a program called a symbolic assembler will be necessary to convert any of the examples to actual code that a Pioneer videodisc player can run. The input to a symbolic assembler is called the “source code file” and is in a syntax that is easy for a programmer to understand; the output of an assembler is called the “object code file” and contains the codes the videodisc player understands.

Symbolic assemblers offer various methods for a programmer to control the process of converting the symbolic source code file to the final object code file. The following is a description of the assembler control syntax used in the four examples of this Appendix. The assembler you use may require different syntax to accomplish the same operations.

```
RSEQ1$  20              ; Set the value of symbol RSEQ1 to 20
RSEQ2$  22              ; Set the value of symbol RSEQ2 to 22

$N 23                   ; $N sets the assembler’s internal register pointer to register 23

$R 2300 2000 1200 1000  ; $R loads the data into registers 23, 22, 21, and 20 respectively

RSTS1 EQU 24            ; Set the value of symbol RSTS1 to 24
RSTS2 EQU 26            ; Set the value of symbol RSTS2 to 26

$N 27                   ; $N sets the assembler’s internal register pointer to register 27

$R 3057 3050 3010 3001  ; $R loads the data into registers 27, 26, 25, and 24 respectively

$ADDR 0                 ; Set the assembler’s internal program counter to 0, all code
                        following this statement will be entered into successive locations
                        in memory.
```

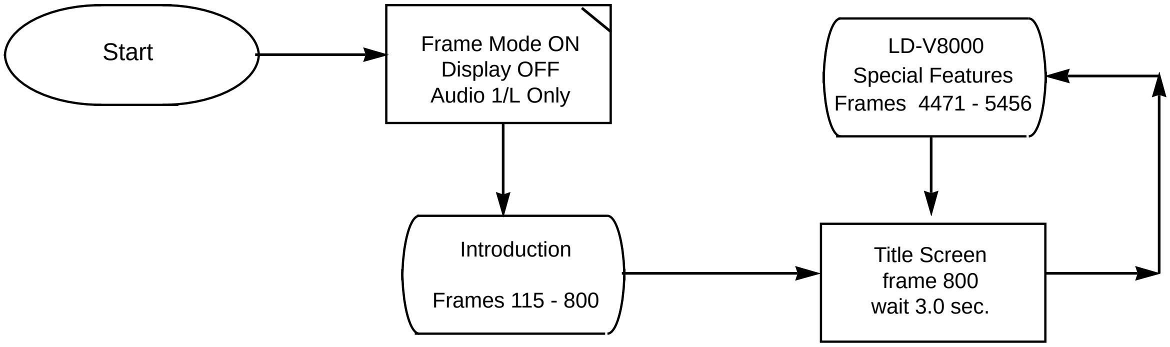
Sample Level II Program Code

Level II Example #1 - Flow Chart

For use with LD-V8000 Demo Disc — CAV Side

A Repeating Video Segment, preceded by an Introduction.

This flowchart details the introductory sequence and the motion segment to be looped.



Level II Example #1 - Program Code

Continuously Repeating Video Segment

The Level II source code shown below uses a mixture of command names and mnemonics. The intent is to show an operational program in an educational way. Some Level II compilers may require a slightly different syntax. With hand entry of the program, the Program Address of each label (TITLE below) must be noted and that address value substituted where necessary.

```
                $ADDR 0     ; START THIS PROGRAM AT PROGRAM ADDRESS 0000
                SFM
        0       DISPLAY     ; TURN OFF DISPLAY
                ANF         ; TURN AUDIO 1 AND 2 ON
        115     SEARCH      ; CUE BEGINNING OF MOTION
        800     AUTOSTOP    ; PLAY THE INTRODUCTION


TITLE:
        800     SEARCH      ; CUE BEGINNING OF MOTION
        30      WAIT        ; SHOW TITLE FRAME FOR 3 SEC
        4471    SEARCH      ; CUE BEGINNING OF MOTION
        5456    AUTOSTOP    ; SHOW LOOP MOTION SEGMENT
        TITLE   BRANCH      ; LOOP BACK
```

Level II Example #2 - Program Code

Menu with Timeout selects Video Segments

This Level II program uses a menu at frame 1000. The frame number is preloaded by the dump load into Register 10 so that it can be changed easily later. One of five segments is played, and the menu has a 15.6 second timeout to an Attract loop. Any key interrupts the Attract loop. Here, we use SC for SEARCH, AS for AUTOSTOP, and BR for BRANCH. Note the use of the symbol RMENU (to represent the value 10) in the program to improve readability.

```
                RMENU$ 10    ; Set "RMENU" to "10", so substitute "10" for
                             ;"RMENU" wherever it is found below.

                $N 10        ; Set the Program Address to the High Byte of Register 10
                $R 1000      ; Put frame number 1000 into Register 10

;-------------------------------------------------------------------------

                             $ADDR 0  ; START AT PROGRAM ADDRESS 0

MMENU:
               RMENU ARG     SEARCH   ; SHOW MENU AT FRAME RMENU (i.e. 1000 HERE)
               156           TINPUT   ; TIMED WAIT FOR A KEY PRESS, TIME OUT IN 15.6 SECONDS
               MMENU         BR       ; DIGIT 0 - IGNORE
               2000          SC       ; KEY "1" COMES HERE, SEARCH BEGINNING OF MOTION
               2200          AS       ; PLAY TO END OF MOTION SEGMENT
               MMENU         BR       ; DIGIT 1
               3000          SC       ; KEY "2" COMES HERE, SEARCH BEGINNING OF MOTION
               3200          AS       ; PLAY TO END OF MOTION SEGMENT
               MMENU         BR       ; DIGIT 2
               4000          SC       ; KEY "3" COMES HERE, SEARCH BEGINNING OF MOTION
               4200          AS       ; PLAY TO END OF MOTION SEGMENT
               MMENU         BR       ; DIGIT 3
               5000          SC       ; KEY "4" COMES HERE, SEARCH BEGINNING OF MOTION
               5200          AS       ; PLAY TO END OF MOTION SEGMENT
               MMENU         BR       ; DIGIT 4
               6000          SC       ; KEY "5" COMES HERE, SEARCH BEGINNING OF MOTION
               6200          AS       ; PLAY TO END OF MOTION SEGMENT
               MMENU         BR       ; DIGIT 5
               MMENU         BR       ; OTHER DIGITS 6-9, IGNORE

               ; TIMEOUT OF THE TINPUT EXECUTION COMES HERE AND CONTINUES BELOW
;--------------------------------------------------------------------
```

Sample Level II Program Code (cont.)

Level II Example #2 - Program Code (CONT.)

```
ATTRACT:
            100             SC       ; ATTRACT LOOP 100 - 700

PRESSING:                            ; WAIT FOR NO KEY PRESS
            BIN             GET      ; GET NUMBER OF LAST KEY PRESSED
            255             COMPARE; IF KEY NUMBER IS NOT 255 KEY WAS PRESSED
            PRESSING        BR
                            PLAY
            SPIN            BR       ; WAIT FOR NO-BUTTON
            PRESSING        BR       ; AND START PLAYING

SPIN:                                ; LOOP HERE AND CHECK FOR END OF ATTRACT LOOP
            0               RECALL
                            STORE    ; GET CURRENT FRAME #
            690             COMPARE; SEE IF ALMOST DONE
                                     ; WE NEED A LITTLE WARNING BEFORE THE REAL
MOTION END

DONE:                                ; ATTRACT LOOP NEAR END SO
            700             AS       ; FINISH LAST 10 FRAMES
            ATTRACT         BR
            DONE            BR       ; NOT YET TO FRAME 690, SEE IF ANY BUTTON PRESSED
            BIN             GET      ; CHECK IF USER PRESSED A KEY
            255             COMPARE
            MMENU           BR       ; ERROR, SHOW MENU -- CAN'T REALLY GET HERE
            SPIN            BR       ; NO BUTTON PRESS
            MMENU           BR       ; BUTTON PRESS, SHOW MENU

;----------------THE END ---------------------------------------------
```

Level II Example #3 - Program Code

Demonstrate menu, sub-menus, interruptable motion, and still-frame sets.

Load a second and third dump (not really needed here). Use sub-menus to select the material to be displayed. Show interruptable motion sequences and sets of still frames.

```
                RSEQ1 EQU 20   ; First motion sequence, frame 1000 - 1200, in registers 20 & 21
                RSEQ2 EQU 22   ; 2nd motion sequence, frame 2000 - 2300, in registers 22 & 23
                $N 23          ; Load data into registers 23, 22, 21, and 20

                $R 2300 2000 1200 1000

                RSTS1 EQU 24   ; Set of 10 stills, from 3001 through 3010
                RSTS2 EQU 26   ; Set of 8 stills, from 3050 through 3057

                $N 27

                $R 3057 3050 3010 3001

;---------------------------------------------------------------------

                            $ADDR 0 ; START OF PROGRAM AT ADDRESS 0

                2           PAGE    ; ENABLE 2 MEMORY PAGES IN PLAYER
                            VOFF    ; TURN OFF VIDEO WHILE OTHER DUMPS ARE LOADED

LOADA:                              ; LOAD 2ND DUMP
                500         SC      ; SEARCH TO 2ND DUMP LOCATION
                1           LOAD    ; LOAD DUMP INTO PAGE 1
                LOADA       BRF     ; TRY AGAIN IF AN ERROR OCCURS

LOADB:                              ; LOAD 3RD DUMP
                1000        SC      ; SEARCH TO 2ND DUMP LOCATION
                2           LOAD    ; LOAD DUMP INTO PAGE 2
                LOADB       BRF     ; TRY AGAIN IF AN ERROR OCCURS
                            WAIT    ; STOP PLAYBACK

MMENU:
                1000        SC      ; SEARCH MENU FRAME
                            VON     ; DISPLAY IT
                3           INPUT   ; WAIT FOR DIGIT KEY PRESS 0..2
                MMENU       BR      ; DIGIT 0 - IGNORE
                SMENU1      BR      ; DIGIT 1, TO SUBMENU 1
                SMENU2      BR      ; DIGIT 2, TO SUBMENU 2
                MMENU       BR      ; OTHER DIGITS 3-9, IGNORE
```

Level II Example #3 - Program Code (CONT.)

```
SMENU1:
            1010        SC      ; SEARCH SUB-MENU 1
            SMENU1      GET     ; GET THE LOCATION OF SMENU1
            3           PUT     ; SAVE IT IN REGISTER 3
            3           INPUT   ; WAIT FOR KEY PRESS 0..2
            MMENU       BR      ; DIGIT 0 - RETURN TO MAIN MENU
            RSEQ1       GET     ; GET THE REG. # CONTAINING MOTION SEQ1 RANGE
            MOTION      BR      ; DIGIT 1, MOTION SEG1
            RSEQ2       GET     ; GET THE REG. # CONTAINING MOTION SEQ2 RANGE
            MOTION      BR      ; DIGIT 2, MOTION SEG2
            SMENU1      BR      ; OTHER DIGITS 3-9, IGNORE

SMENU2:
            1020        SC      ; SEARCH SUB-MENU 2
            SMENU2      GET     ; GET THE LOCATION OF SMENU2
            3           PUT     ; SAVE IT IN REGISTER 3
            3           INPUT   ; WAIT FOR KEY PRESS 0..2
            MMENU       BR      ; DIGIT 0 - RETURN TO MAIN MENU
            RSTS1       GET     ; GET THE REG. # CONTAINING STILLS STS1 RANGE
            STILLS      BR      ; DIGIT 1, STILL GROUP 1
            RSTS2       GET     ; GET THE REG. # CONTAINING STILLS STS2 RANGE
            STILLS      BR      ; DIGIT 2, STILL GROUP 2
            SMENU2      BR      ; OTHER DIGITS 3-9, IGNORE

;--------------------------------------------------------------------------
```

Sample Level II Program Code (cont.)

Level II Example #3 - Program Code (CONT.)

```
MOTION:                     ; REG 0 HAS REGISTER # OF START FRAME, END FRAME IS IN NEXT REG
                            ; STOP BUTTON INTERRUPTS MOTION, RETURNS TO
                            ; CALLING MENU
                            ; 0-DIGIT INTERRUPTS AND RETURNS TO MAIN MENU

                0 ARG ARG           SC          ; SEARCH THE FRAME # THAT IS IN THE REG # THAT IS
                                                ; IN REG 0
                                    PLAY        ; LOCATE START FRAME AND PLAY
                1                   ADD         ; INCREMENT REG 0 TO POINT TO THE END FRAME
                                                ; REGISTER
                0 ARG ARG           GET         ; GET END FRAME
                4 PUT                           ; SAVE IT IN REG 4
                10                  SUB         ; SUBTRACT 10 FROM END FRAME
                5                   PUT         ; AND SAVE IT IN REG 5
                                                ; WE NEED A LITTLE WARNING BEFORE THE REAL
                                                ; MOTION END
SPIN:
                0                   RECALL
                                    STORE       ; GET CURRENT FRAME #
                5 ARG               COMPARE     ; SEE IF ALMOST DONE
DONE:
                4 ARG               AS          ; PLAY TO END OF MOTION
                3 ARG               BR          ; FINISH & RETURN TO LAST SUB-MENU
                DONE                BR

                            ; NOT TO END-TEST FRAME YET, SEE IF STOP OR 0 WAS PRESSED
                BIN                 GET         ; GET KEY # OF LAST KEY PRESSED
                251                 COMPARE     ; CHECK FOR STOP KEY
                SPIN                BR          ; NOT STOP KEY SO CONTINUE CHECKING FOR
                                                ; MOTION END
                                    WAIT        ; STOP MOTION PLAYBACK
                3 ARG               BR          ; STOP PRESSED, RETURN TO LAST SUB-MENU

                63                  COMPARE     ; CHECK FOR ZERO KEY PRESS
                SPIN                BR          ; NOT ZERO KEY SO CONTINUE CHECKING FOR
                                                ; MOTION END
                                    WAIT        ; STOP MOTION PLAYBACK
                MMENU               BR          ; 0 PRESSED, RETURN TO MAIN MENU
                SPIN                BR          ; NOT ZERO KEY SO CONTINUE CHECKING FOR
                                                ; MOTION END

;---------------------------------------------------------------------------
```

Level II Example #3 - Program Code (CONT.)

```
STILLS:                       ; REG 0 HAS REGISTER # OF START FRAME, END FRAME IS IN NEXT REG
                              ; USE STEP FWD AND REV TO NAVIGATE THE STILLS
                              ; STEP FWD ON END FRAME RETURNS TO CALLING MENU
                              ; 0-DIGIT INTERRUPTS THE STILLS AND RETURNS TO MAIN MENU

            0 ARG ARG      GET       ; GET THE START FRAME # IN THE REG # THAT IS IN REG 0
            4              PUT       ; SAVE START FRAME
            4 ARG          SC        ; DISPLAY START FRAME
            1              ADD
            0 ARG ARG      GET       ; GET THE END FRAME # IN THE REG # THAT IS IN REG 0
            5 PUT                    ; AND SAVE IT IN REG 5

NAVIGATE:
            1              FIN       ; WAIT FOR STEP FWD OR STEP REV KEY
            MMENU BR                 ; 0-DIGIT, RETURN TO MAIN MENU
            NAVIGATE BR              ; DIGITS 1-9, IGNORE

                                     ; HANDLE FUNCTION KEYS
            DIN            GET       ; FUNCTION KEY NUMBER TO REG 0
            12             COMPARE; CHECK FOR STEP FORWARD KEY
            NAVIGATE       BR        ; GO AND WAIT FOR SOME OTHER KEY
            FORWARD        BR        ; STEP FORWARD
            11             COMPARE; CHECK FOR STEP REVERSE KEY
            NAVIGATE       BR        ; GO AND WAIT FOR SOME OTHER KEY
            REVERSE        BR        ; STEP REVERSE

            NAVIGATE       BR        ; GO AND WAIT FOR SOME OTHER KEY

;---------------------------------------------------------
```

Level II Example #3 - Program Code (CONT.)

```
FORWARD:
            0               RECALL
                            STORE     ; GET THE CURRENT FRAME #
            5 ARG           COMPARE;  CHECK IF AT END FRAME OF STILLS
            3 ARG           BR
            3 ARG           BR        ; AT OR PAST END, --> SUB-MENU
                            STEPF     ; NOT AT END, STEP FWD
            NAVIGATE        BR        ; GO AND WAIT FOR SOME OTHER KEY

;--------------------------------------------------------
REVERSE:
            0               RECALL
                            STORE     ; GET THE CURRENT FRAME #
            4 ARG           COMPARE;  CHECK IF AT BEGINNING FRAME OF STILLS
                            STEPR     ; NOT AT START, STEP REVERSE
            NAVIGATE        BR
            NAVIGATE        BR        ; AT START FRAME, DO NOTHING
            4 ARG           SC
            NAVIGATE        BR        ; ERROR, BEFORE START


;------------------------- THE END -------------------------------------
```

Level II Example #4 - Program Code

"Sound-over-Still" selections from a multi-page menu.

Use a multi-page menu to select a non-interruptable "sound-over-still" sequence.

```
                 RSEQ1 EQU 20   ; 1st "sound (1000 - 1200) over still (5010)" sequence, Audio 1 only
                 RSEQ2 EQU 24   ; 2nd "sound (1000 - 1200) over still (5020)" sequence, Audio 2 only
                 RSEQ3 EQU 28   ; 3rd "sound (3000 - 3200) over still (5030)" sequence, Audio 1 & 2
                 RSEQ4 EQU 32   ; 4th "sound (4000 - 4200) over still (5040)" sequence, Audio 1 only

                                ; THE FOLLOWING SYNTAX WILL $N 35

                 $R 4200 4000 5040 1 3200 3000 5030 3 1200 1000 5020 2 1200 1000 5010 1

;---------------------------------------------------------------------

                                $ADDR 0   ; START OF PROGRAM AT ADDRESS 0
                 1              SMS       ; ALLOW PROGRAM CONTROL OF VIDEO BUFFER

MENUA:
                 7010           SC        ; SEARCH MENUA STILL
                                MWE       ; ENABLE FRAME BUFFER WRITE
                 MENUA          GET       ; GET MENUA LOCATION AND
                 3              PUT       ; SAVE IT IN REG 3
                 3              FINPUT    ; WAIT FOR DIGITS OR FUNCTION KEY PRESS
                 MENUA          BR        ; DIGIT 0 - IGNORE
                 RSEQ1          GET       ; GET TABLE REG LOCATION
                 SOSMOTION      BR        ; DIGIT 1, SOS SEG 1
                 RSEQ2          GET       ; GET TABLE REG LOCATION
                 SOSMOTION      BR        ; DIGIT 2, SOS SEG 2
                 MENUA          BR        ; OTHER DIGITS 3-9, IGNORE
                                          ; ANY FUNCTION KEY GOES TO NEXT MENU
```

Sample Level II Program Code (cont.)

Level II Example #4 - Program Code (CONT.)

```
MENUB:
               7020           SC          ; SEARCH MENUB STILL
                              MWE         ; ENABLE FRAME BUFFER WRITE
               MENUB          GET         ; GET MENUB LOCATION AND
               3              PUT         ; SAVE IT IN REG 3
               3              FINPUT      ; WAIT FOR USER KEY PRESS
               MENUB          BR          ; DIGIT 0 - IGNORE
               RSEQ3          GET         ; GET TABLE REG LOCATION
               SOSMOTION      BR          ; DIGIT 1, SOS SEG 3
               RSEQ4          GET         ; GET TABLE REG LOCATION
               SOSMOTION      BR          ; DIGIT 2, SOS SEG 4
               MENUB          BR          ; OTHER DIGITS 3-9, IGNORE
               MENUA          BR          ; ANY FUNCTION KEY GOES TO OTHER MENU

;---------------------------------------------------------------------------
                              ; THIS SOSMOTION ROUTINE DEPENDS ON A GROUP OF
                              ; REGISTERS CONTAINING THE
                              ; DATA IN THE FOLLOWING FORMAT:

SOSMOTION:                    ; REG 0 HAS REGISTER # OF AUDIO-SET FLAG (1, 2. or 3),
                              ; THE STILL FRAME NUMBER IS IN THE 2ND REG
                              ; THE START FRAME IS IN THE NEXT REG &
                              ; THE END FRAME IS IN THE 4TH REG

               2              PUT         ; SAVE REG 0
               1              ADD         ; INCREMENT REG. 0
               0 ARG          RECALL      ; GET SET FOR SEARCH TO STILL
                              SC          ; SHOW THE STILL

               2 ARG ARG      GET         ; GET THE AUDIO FLAG
               2              COMPARE
                              ANN
               MOVE           BR          ; FLAG > 2, USE BOTH AUDIOS
                              AFN
               MOVE           BR          ; FLAG = 2, USE AUDIO 2 ONLY
                              ANF         ; FLAG < 2, USE AUDIO 1 ONLY

MOVE:
                              MWD         ; ENABLE FRAME BUFFER WRITE
                              SC          ; SEARCH TO FRAME IN ACTIVE REGISTER
                              AUTOSTOP    ; LOCATE START FRAME AND PLAY AUDIO
               3 ARG          BR          ; RETURN TO MENU

;------------------------------ THE END -------------------------------------
```

PIONEER®

PIONEER COMMUNICATIONS OF AMERICA, INC.

EAST
Corporate Headquarters and
Multimedia Systems Division Sales Office
& Engineering/Technical Support
600 East Crescent Avenue
Upper Saddle River, NJ 07458-1827
TEL: (201)327-6400
FAX: (201)327-9379

CENTRAL
MSD Sales Office
1263 Hamilton Parkway
Itasca, IL 60143
TEL: (708)285-4500
FAX: (708)285-4570

WEST
MSD Sales Office
& Technical Support
915 E. 230th St.,
Carson, CA 907457
TEL: (310)522-8600
FAX: (310)522-8699

PIONEER ELECTRONIC (EUROPE) N.V.
Keetberglaan 1, 2740 Beveren, BELGIUM

PIONEER ELECTRONICS (AUSTRALIA) PTY LTD.
178-184, Boundary Road Braeside, Vic. 3195, AUSTRALIA

PIONEER ELECTRONIC CORPORATION
4-1, Meguro 1-Chome, Meguro-ku, Tokyo 153, JAPAN

Pioneer Technical Publication TP 113 V. 2.1 2/93