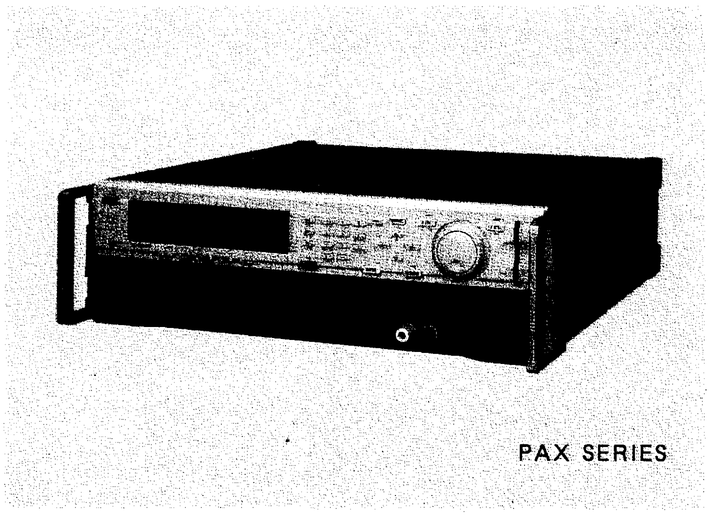
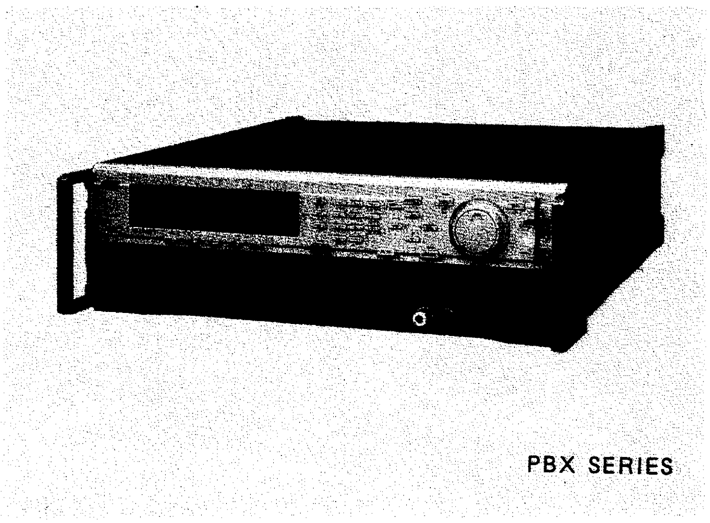
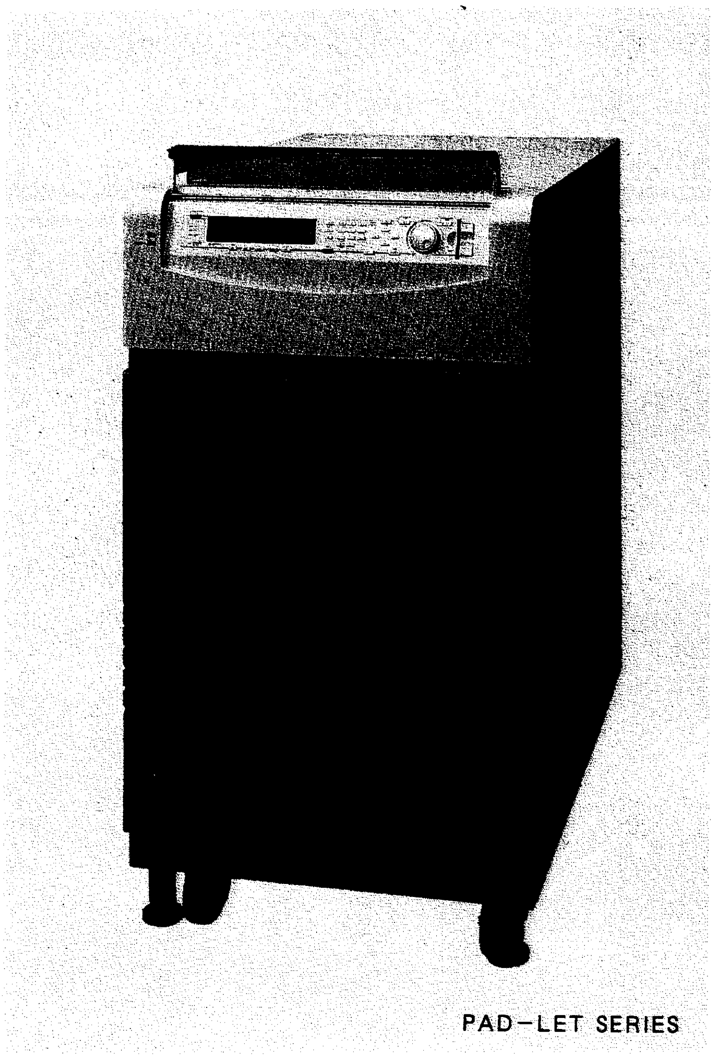
SEQUENCE OPERATION GUIDEBOOK

APPLICABLE MODELS

PAX SERIES
PBX SERIES
PAD-LET SERIES


PAX SERIES


PBX SERIES


PAD-LET SERIES

KIKUSUI

Use of Guidebook

- If you find any incorrectly arranged or missing pages in this guidebook, they will be replaced. If the guidebook it gets lost or soiled, a new guidebook can be purchased. In either case, please contact your Kikusui agent, and provide the "Kikusui Part No." given on this page.
- This guidebook has been prepared with the utmost care; however, if you have any questions, or note any errors or omissions, please contact your Kikusui agent.

Kikusui Part No. Z1-002-562, IB003081
Printed in Japan.

This book will let you understand sequence operations of the PAX, PBX, and PAD-LET series power supplies. In general, most people think that creating a program often seems difficult because there are various things that should be understood beforehand, and therefore it is not easy to start. So, we created this book that urges you to make sequence programs anyway and to understand what the sequence operation is. Creating a sequence program is the same as constructing a Sequence File.

At any rate, let's try it, and then you will realize the sequence programs are unexpectedly easy to create.

Composition of this book

TABLE OF CONTENTS

Chapter 1

SEQUENCE FUNCTION

This chapter summarizes about the sequence function. When you use the sequence function, utilize it as well as the operation manual.

Contents Page

A sequence is a function that executes a specific output waveform pattern automatically. Since what you will be practicing here are the sequence functions for the PAX, PBX, and PAD-LET series DC power supplies, the main output values automatically executed are the output voltage and the output current. You will program these values by a regular procedure, after which they will be automatically executed.

There are, roughly, two types of sequence modes. (Figure 1 and Figure 2)

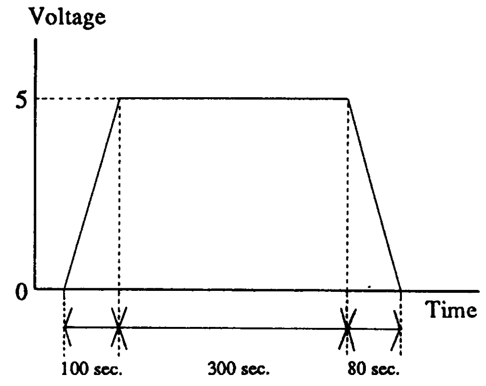


Figure 1 Normal Sequence Mode

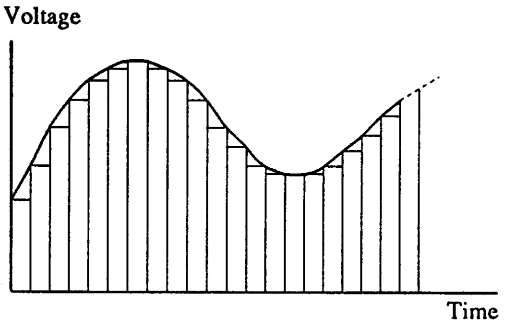


Figure 2 Fast Sequence Mode

The ***Normal Sequence*** is used for a long sequence such as figure 1. It gently rises from 0V to 5V in 100 seconds, then holds this value for 300 seconds whilst performing various measurements, and finally falls gently from 5V to 0V in 80 seconds.

In the ***Fast Sequence***, the target waveform should be approximated by a bar chart of min. 100 microsecond units. Therefore, by dividing a complex waveform, illustrated in Figure 2, in up to 1024 indentical intervals, and making the output vary rapidly, you can simulate the target waveform.

NOTE *Fast Sequence is unavailable on PAD-LET.*

Now, let's study sequences in more detail.

1.1 Normal Sequence

Consider the case of constructing a program for the sequence pattern shown in Figure 3.

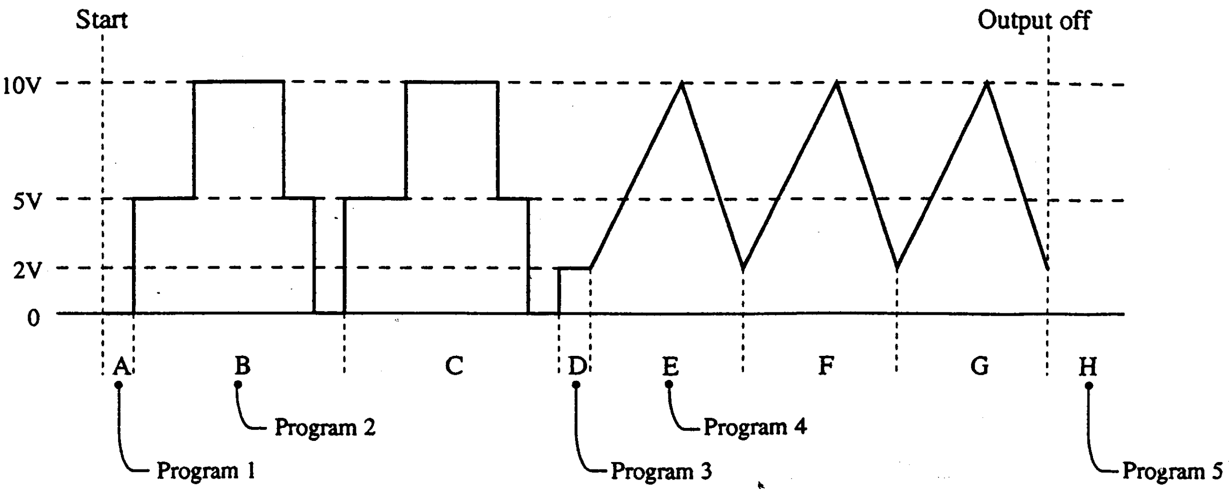


Figure 3 An Example of a Sequence

1.1.1 About the Program

Decomposing the sequence pattern in sections A to H, as in Figure 3, it seems that B and C have an identical pattern and E, F, and G are also repetitions of a unique pattern. Considering the repetitions of the pattern and picking out any unique pattern, you will notice that this sequence has only five patterns — A, B, D, E, and H. Call each section a ***Program*** with a numbering suffix. Then you will get:

Pattern A :	Program 1
Pattern B :	Program 2
Pattern D :	Program 3
Pattern E :	Program 4
Pattern H :	Program 5

Continuing, we will decompose the sequence. In Normal Sequence, since the values of output setup and execution time can be specified as a pair, for instance in Pattern B, you can decompose it into the following four:

1 5V for 2 seconds

2 10V for 2 seconds

3 5V for 1 second

4 0V for 1 second

The minimal element obtained in this manner is called a ***Step***. Therefore, we can say "Program 2 (pattern B) consists of four Steps."
What about pattern E? In pattern E, the voltage rises in a straight oblique line from 2V to 10V, in three seconds. Such an oblique transition is what we call a Ramp (see column), and it consists of a single Step. Therefore, Program 4 (pattern E) can be composed of only two Steps as follows:

Step 1: The voltage rises from 2V to 10V, taking 3 seconds.

Step 2: The voltage falls from 10V to 2V, taking 2 seconds.

Needless to say, remaining Programs 1, 3, and 5 are composed of one Step respectively.

Program 1: Step 1 0V for 1 second

Program 3: Step 1 2V for 1 second

Program 5: Step 1 0V and output off

Here, we tell you an important topic of how you can master the sequence function well. Programs 1, 3, and 5 work as boundary values at the beginning of or the end of big clusters such as Program 2 or 4. There are a lot of functions in the power supply instrument besides the sequence function, and the settings of those are intricately related. Therefore, the sequence might show an unexpected operation. In the beginning, although you will think it is useless to set an explicit initial value, make sure to do so. A concrete example will be introduced in Chapter 4.

Column:

With the old manner of sequence capability, it was necessary to prepare many steps to imitate such an operation (Ramp transition), for example --- 2V for 0.375 seconds, 3V for 0.375 seconds, and so on. However, the new sequence capability can simulate them in only one step, with a one millisecond resolution.

For instance, if you make the voltage increase 8 volts in 3 seconds, theoretically there are 3000 steps raised with 2.7mV for each step. (However, the number of actual steps will be fewer because the output setup resolution isn't so fine.)

1.1.2 About the Sequence

You have probably understood about the construction of a Program, with reading the above explanations.

By the way, to get the output waveform of Figure 3 using five Programs explained above, it's of no use that everything from Program 1 through Program 5 is simply executed one by one. Program 2 must be run twice, consecutively, and also Program 4 must be executed three times consecutively. What specifies the flow of programs is what we call a ***Sequence***.

In the sequence function, you must assign one Sequence to one Program. The Sequence specifies what number of Program is to be executed, how many times it is to be repeated, and what following number of Sequence is to be invoked. So, you can interpret Figure 3 as listed below:

Sequence 1: Run Program 1 once, and invoke Sequence 2.

Sequence 2: Run Program 2 twice, and invoke Sequence 3.

Sequence 3: Run Program 3 once, and invoke Sequence 4.

Sequence 4: Run Program 4 three times, and invoke Program 5

The function that specifies the following Sequence number to be invoked after repetition of the specified Program is, in the meaning of connecting Sequence to Sequence, called a Chain. In other words, if you decide "Set the Chain Sequence to 2 at Sequence 1," Sequence 2 will be invoked after completion of Sequence 1.

However, the situation differs a little for Sequence 4. If the above explanation were applied simply, although it will be usual to execute Sequence 5 after the Program 4, actually the Program 5 must be executed. The reason for this we will discuss in the next section.

1.1.3 About the End Program

Imagine that you have to interrupt the sequence execution for some reason, while the sequence is running. Usually, you would think you 'd like to generate zero voltage and safely turn the output off. (Of course, zero voltage is not always correct.) In the example of Figure 3, because what the final Program 5 delivers is "Voltage 0, Output off," you will do so naturally when the execution is interrupted. Then, the Program 5 is treated as an exception, the ***End Program***, and is used as shown in Figure 4.

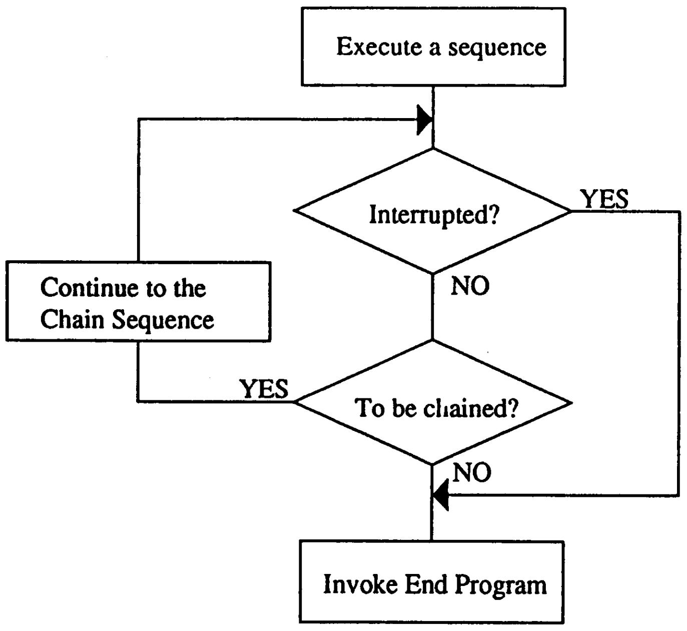


Figure 4 Relationship between Chain Sequence and End Program

1.2 Fast Sequence

NOTE *PAD-LET doesn't have this capability.*

Fast Sequence is a function that can simulate a waveform by varying setup values rapidly and it is shown in Figure 2. For instance, assume that you simulate a 50Hz sine waveform with the Fast Sequence. At 50Hz, one period of the waveform is 20 milliseconds. When the waveform is divided into 100 points with identical intervals, each interval will be 200 microseconds. So in the Fast Sequence, you can simulate the sine waveform with 100 steps changing setup values per 200 microseconds. (See Figure 5.)

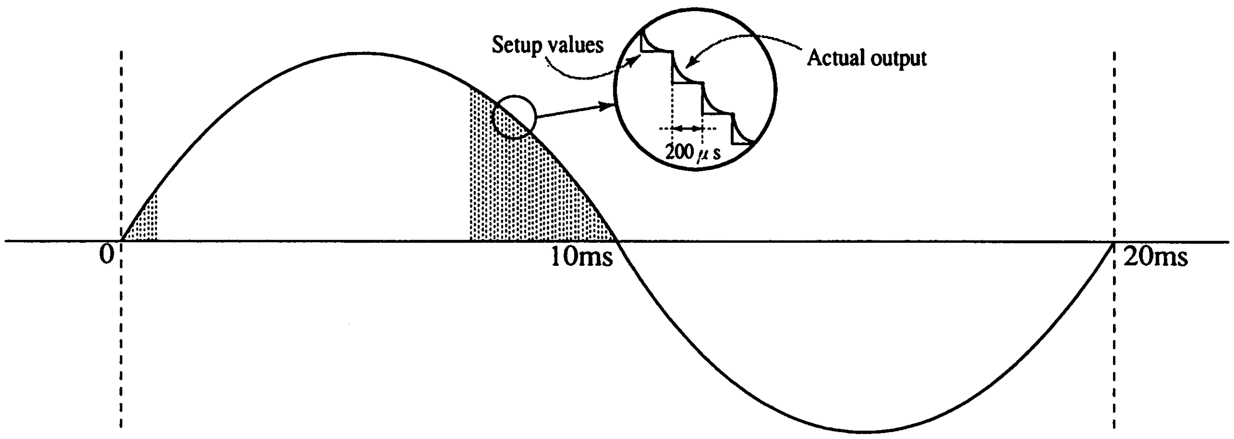


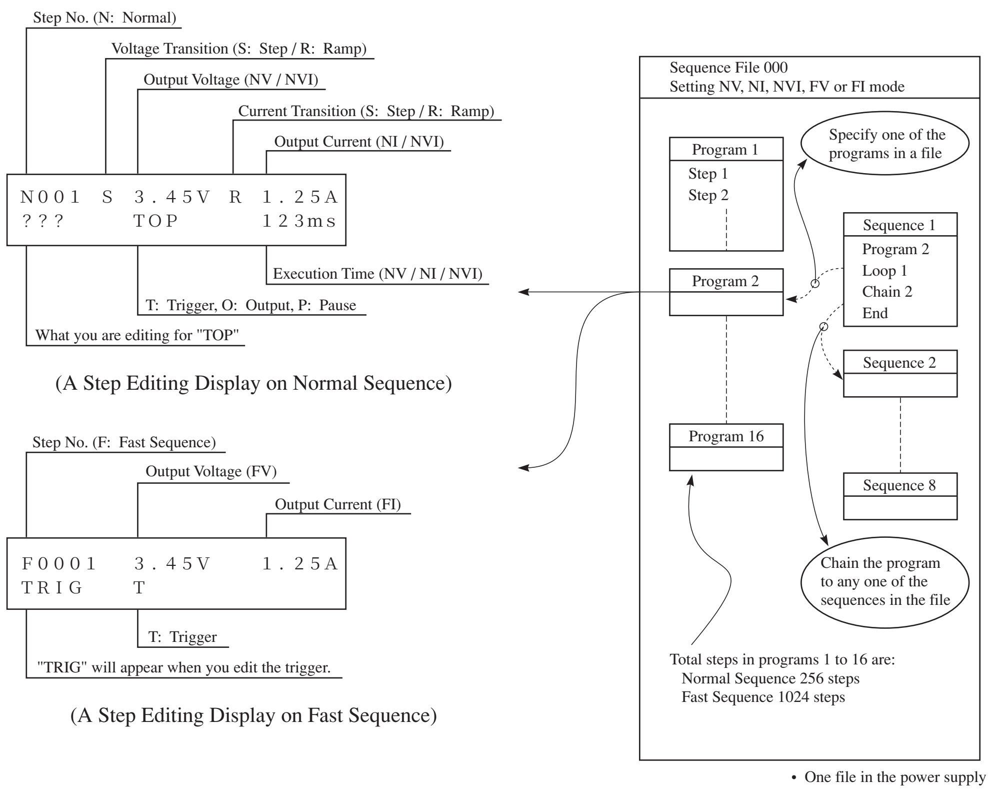
Figure 5 Approximation By Fast Sequence

The Fast Sequence has a maximum capability of showing setup variances every 100 microseconds. You might think, "Ah, it is said to be high-speed, though, 100 microseconds is not so fast." However, it is still in a high-speed category in the world of (more than hundred-watt) Power Supplies! Although how to program the Fast Sequence is almost similar to the Normal Sequence, identical execution time must be applied for all the steps. Therefore, the execution time of each step in a Program cannot be set and, you have to set it at the side of editing Sequence that specifies how Programs will be executed. In addition, you also cannot operate the Ramp transition in Fast Sequence.

During operation, it is possible to enter the sequence program from the instrument panel. However, if many steps are required, it will be easier to transfer the program data calculated with a personal computer via GPIB or RS232C.

1.3 Construction of a Sequence

So far, various keywords such as ***Step***, ***Program***, and ***Sequence*** appeared. To help you understand, we will now explain more about them. First, look at Figure 6.



Step No. (N: Normal)
Voltage Transition (S: Step / R: Ramp)
Output Voltage (NV / NVI)
Current Transition (S: Step / R: Ramp)
Output Current (NI / NVI)
N001 S 3.45V R 1.25A
??? TOP 123ms
Execution Time (NV / NI / NVI)
T: Trigger, O: Output, P: Pause
What you are editing for "TOP"
(A Step Editing Display on Normal Sequence)
Step No. (F: Fast Sequence)
Output Voltage (FV)
Output Current (FI)
F0001 3.45V 1.25A
TRIG T
T: Trigger
"TRIG" will appear when you edit the trigger.
(A Step Editing Display on Fast Sequence)
Sequence File 000
Setting NV, NI, NVI, FV or FI mode
Program 1
Step 1
Step 2
Specify one of the programs in a file
Sequence 1
Program 2
Loop 1
Chain 2
End
Program 2
Sequence 2
Program 16
Sequence 8
Chain the program to any one of the sequences in the file
Total steps in programs 1 to 16 are:
Normal Sequence 256 steps
Fast Sequence 1024 steps
• One file in the power supply

Figure 6 Structure of a Sequence File

1.3.1 About the Sequence File

Now, you will find NV, NI, NVI, FV, and FI in the column of the Sequence File in Figure 6. These are notations that indicate the kinds of sequence modes, N denotes the Normal Sequence and F denotes the Fast Sequence. V and I mean the sequence for which voltage and current setup values respectively, can be changed. For instance, NV makes the voltage and current setup values vary simultaneously. (The NVI mode doesn't exist in the PBX series. Also there are no FV and FI modes in the PAD-LET series.) When you create a new Sequence File, you must decide which mode to use. Also, only one mode can be utilized in a file. So, for example “A Sequence File of NV mode.”

A Sequence File is composed of ***Steps***, ***Programs***, and ***Sequences***, and can be preserved in the non-volatile memory (File #0) built in the instrument.

1.3.2 About a Step

Decomposing the sequence pattern, a minimal element is called a ***Step***. See the examples of Program Editing Display of Normal and Fast Sequences shown in Figure 6. In the Normal Sequence, since the execution time for each Step can be specified, you can treat the part where the output value increases or decreases in a straight oblique line, or remains the same, as one Step. However, in the Fast Sequence, there's no parameter to specify the execution time for Step. This means that the pattern of the Fast Sequence can be decomposed only into identical intervals. You can use up to 256 Steps in Normal Sequence and up to 1024 Steps in Fast Sequence in a Sequence File.

1.3.3 About the Program

Looking upon the cluster that repeatedly uses the identical waveform in a sequence pattern as a ***Program***, decompose the pattern. Each Program is a collection of ***Steps***, and up to sixteen Programs can be made up in a Sequence File.

1.3.4 About the Sequence

As explained above, a ***Program*** is a cluster from which the sequence is constructed. To provide the sequence pattern you want, you must specify in which order the Programs are to be executed, and how many times each is repeated. What specifies this is what we call a ***Sequence***. Up to eight Sequences can be made up in a Sequence File.
A concrete example of above explanation was introduced in the very beginning of this chapter. Read it once more to review.

1.3.5 Edit Memory

Once you have created a Sequence File, it should be saved in File #0 (non-volatile memory built in the instrument). (You must do this yourself. The file is never automatically preserved.) However, if not yet preserved, the contents of the file are still in the ***Edit Memory*** and can easily be lost when the instrument's power is turned off. It is shown in Figure 7.

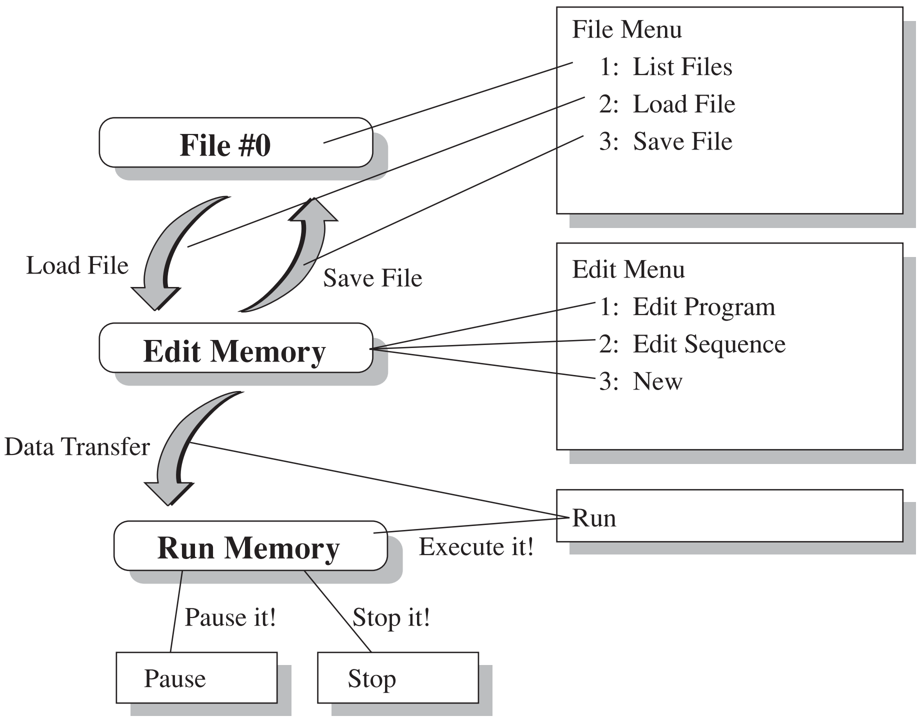


Figure 7 Conceptual Diagram of Sequence Operations

For instance, the Sequence File is loaded from File #0 (non-volatile memory) to the Edit Memory in the upper column on the right, named File Menu. Then in Edit, you edit the contents of Edit Memory, and then execute it with Run.

Chapter 2

PROCEDURE FOR CREATING A SEQUENCE

This chapter explains what should be confirmed before you get started on creating a sequence file, and a flowchart of the procedure for creating is also described.

Contents Page

2.1 Before Creation

At first, you should confirm what is written in the existing sequence memory built in the instrument. Then, shutdown the instrument power and turn it on again.

```
OUT   0.000V   0.00A
```

Shown above, the status immediately after turning power on is called the ***Root State***, and various setup items branch from there. In the above display example, what the numeric parts indicate are the existing output voltage and current setups. Note that these parts are not always as above.

Confirm the File List.

1) Press the **SEQ** key to invoke the Sequence Menu, and then by entering the number from the numeric keys choose 1:File.

```
>1:File
 2:Edit
```

2) And, choose 1:List Files.

```
>1:List Files
 2:Load File
```

3) Then, the following list display will appear.

```
List:000  SAMPLE
 NVI Step:0009
```

Like above, when the message SAMPLE is on the display, it is possible that an example sequence from the factory-default is still written in File #0. If not so, there's no problem, because somebody might have preserved a sequence in File #0. Then, you'd better consult the person who created it. After the confirmation, press the **ESC** key several times and return to the Root State.

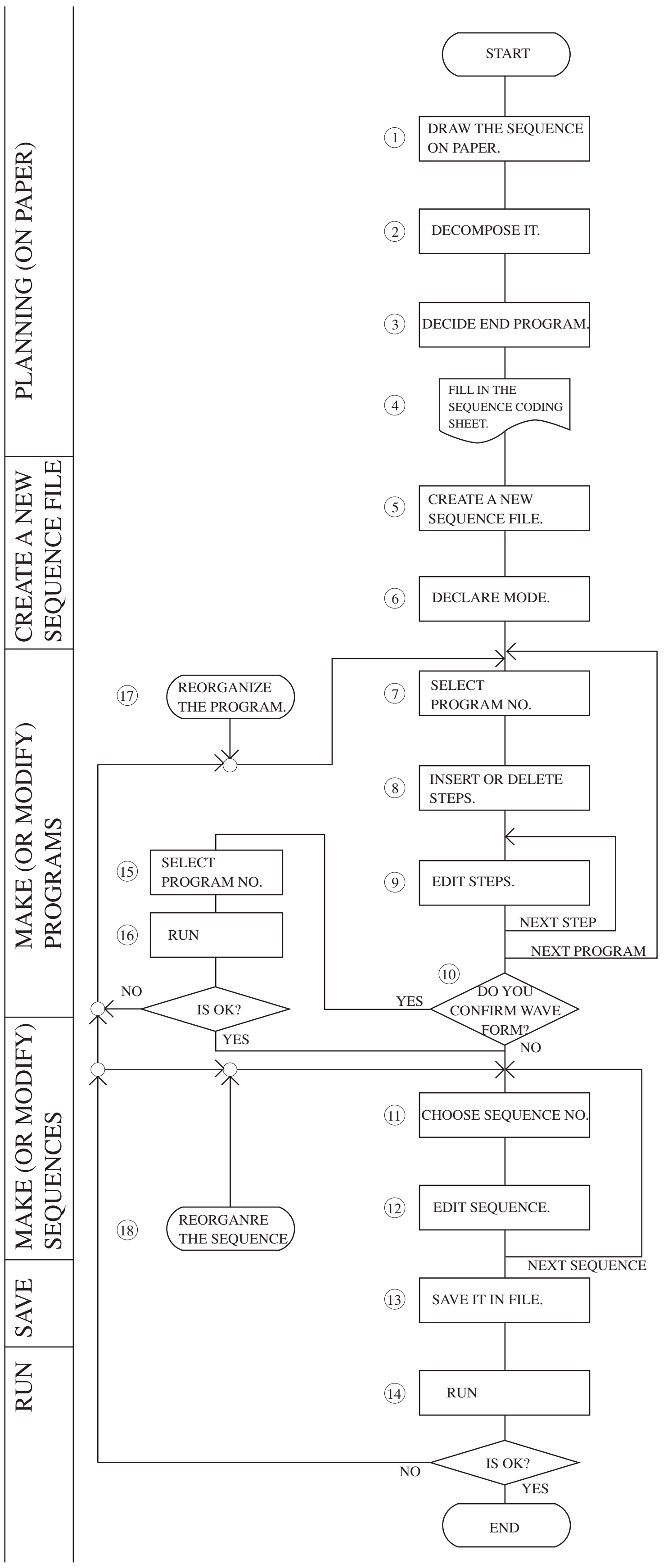
2.2 Flow of Creating a Sequence File

Here, we introduce a flowchart for the procedure of creating a Sequence File (Figure 9). Also, the menu hierarchy of the sequence function is given in Figure 8 and should be understood before you create actual sequences.

From Chapter 3 on, you will practice how to create sequences referring to Figure 8 and 9. When you actually use the sequence function, Figures 8 and 9 will surely be useful.

```
SEQUENCE MENU
├─ 1: [File] (File Management Menu)
│   │
│   ├─ 1: [List Files]      : Displays the sequence file contents.
│   ├─ 2: [Load File]       : Loads a sequence file into the edit memory.
│   ├─ 3: [Save File]       : Saves the contents of edit memory to a sequence file.
│   └─ 4: NULL
│
├─ 2: [Edit] (Edit Menu)
│   │
│   ├─ 1: [Edit Program]    : Inserts, modifies, or deletes steps in the edit memory.
│   ├─ 2: [Edit Sequence]   : Edits sequence parameters.
│   └─ 3: [New]             : Creates a new sequence file.
│
├─ 3: [Run] (Run Menu)      : Run the sequence in the edit memory.
│
└─ 4: [Configuration] (Configuration Menu)
    │
    ├─ 1: [TRIG Direction]  : Defines input/output direction of trigger.
    └─ 2: [Auto Run]        : Enables or disables the automatic execution of a sequence when starting
                              up the instrument.
```

Figure 8 Menu Hierarchy and Descriptions of Sequence Operations



① Draw the waveform you'd like to get on paper, deciding what mode of sequence you will use.

② Decomposing it into Steps and Programs, construct Sequences. If Normal Sequence, also decide the execution time of each step.

③ Consider how the output is to be set after finished or interrupted.

④ Write concrete setup values on it.

⑤ Confirming that an existing sequence may be lost, initialize it by 3: New under Edit Menu.

⑥ Select a mode using up/down direction keys. When using Normal Sequence, also choose execution time.

⑦ Choose 1: Edit Program under Edit Menu. Scrolling program display with up/down direcrion keys, select a Program No. you want to edit by the ENTER key.

⑧ After newly created, you will see only a "New" message. Select 2: Insert under Edit Program Sub Menu and then insert required step area. If you wish to delete it, choose 3: Delete. To edit existing Steps, you can select 1: Modify.

⑨ Set all Step data according to the Coding Sheet.
You can move to other Step No. with up/down keys.
To move to other Program No., return to Program Display with the ESC key and then follow the procedure 7 thru 9.

⑩ You can confirm whether the Program is okay or not.

⑪ Return to Edit Menu with the ESC key then choose 2: Edit Sequence. Select Sequence No. with up/down keys.

⑫ Pressing ENTER key, edit the Sequence. You can move to other Sequence No. with up/down keys.
Note that you must set End Program to every Sequence.

⑬ Return to Sequence Menu with ESC, choose 1: File and 3: Save File. Save the sequence file to the internal memory(#00) with the ENTER key.
Note that existing contents will be lost.

⑭ Press the RUN key and select a Sequence No. you want to run. Again press RUN. The sequence will run.

⑮ From the Program Display, select a Program No. with up/down keys.

⑯ With RUN key, the Program runs only once.

⑰ Pressing EDIT key, select 1: Edit Program under Edit Menu.

⑱ Pressing EDIT key, select 2: Edit Sequence under Edit Menu.

Figure 9 Flowchart for Creating a Sequence File

Chapter 3

A Petit Program

It is possible to generate a little waveform, without completing a real sequence file. First, try it.

Contents Page

In this chapter, you create and execute a very simple waveform pattern with only a ***Program*** of the Normal Sequence. Think of it as practice for the subsequent chapters.

3.1 Decomposing the Petit Pattern

Here, create the simple sequence pattern shown in Figure 10.

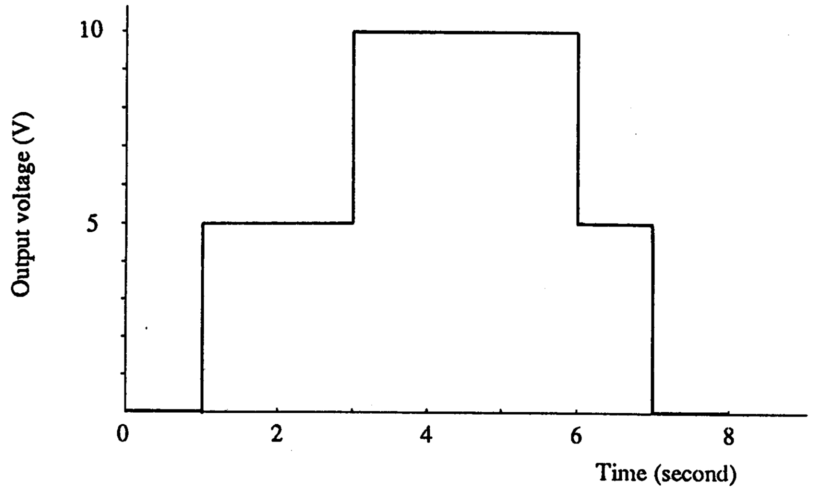
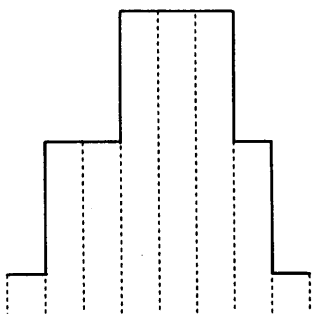


Figure 10 A Simple Sequence Pattern

Figure 10 can be interpreted in two ways illustrated in Figure 11 and Figure 12.



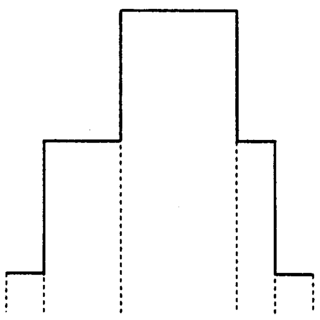


Figure 11 Decomposed in One Second Intervals Figure 12 Decomposed with Interval Variance

In Figure 11, the pattern is decomposed in one-second intervals. With this method, each step execution time has an identical interval and there are a total of eight steps. Thinking simply, a Program that deals with eight steps explained in Table 1 should be applied. This idea corresponds to the Fast Sequence. (But, one second is too long for the Fast Sequence.)

Table 1

1 second for each step execution	
Step 001	0V
Step 002	5V
Step 003	5V
Step 004	10V
Step 005	10V
Step 006	10V
Step 007	5V
Step 008	0V

Then, decompose the pattern as in Figure 12 paying attention to the voltage variance. In this method, a step is composed of a voltage value and the execution time, as a pair. Therefore, simply five steps are required. The number of steps is fewer than in the case of Figure 11.

Table 2

Step 001	0V for 1 second
Step 002	5V for 2 seconds (in Step Transition)
Step 003	10V for 3 seconds (in Step Transition)
Step 004	5V for 1 second (in Step Transition)
Step 005	0V for 1 second (in Step Transition)

Now, let's complete the Program section of the Sequence Coding Sheet, an accessory of this book. An example is shown in Table 3. Because the Ramp transition isn't used, every column of S/R must be S, meaning the Step Transition (variance a lot like a staircase) in this example.

Table 3 Entering a Sequence Creation Sheet, Example

Program:

No.	Step No.	S/R	CV[V]	S/R	CC[A]	Trig	Out	Pause	Time[s]	Comments
01	*001*	*S*	*0*			*0*	*1*	*0*	*1*	*My*
	002	*S*	*5*			*0*	*1*	*0*	*2*	*Training*
	003	*S*	*10*			*0*	*1*	*0*	*3*	
	004	*S*	*5*			*0*	*1*	*0*	*1*	
	005	*S*	*0*			*0*	*1*	*0*	*1*	

3.2 A Petit Normal Sequence

3.2.1 Creating a New Sequence File

Operate the key panel according to the following procedure.

1) Press the **EDIT** key and invoke the Edit Menu.

```
>1:Edit Program
 2:Edit Sequence
```

2) Press the down-arrow key to scroll, and then choose 3:New (Create New Sequence). To select the item, it is possible to either move the marker (>) and then press the **ENTER** key, or choose immediately with a numeric key.

```
 2:Edit Sequence
>3:New
```

Then, the following confirmation message will appear as the contents of the existing memory can be lost if carried out.

```
Create New Sequence
      Sure?
```

3) As it can be carried out here, press the **ENTER** key.

```
Mode:NV
```

4) Next, you have to select which sequence mode (Normal/Fast) is used. Because you choose the NV (Normal Sequence / Voltage variance) here, simply press the **ENTER** key and advance.

```
Mode:NV
Unit:msec
```

5) Here, you are asked for the time unit of Step. You can use the down-arrow key to change the unit to sec. Pressing the **ENTER** key, the following message will appear for a couple of seconds and you will return to the Edit Menu. Then, you're ready to continue.

```
Mode:NV
 Completed
```

3.2.2 Making the Petit Program

After the file is newly created, you are ready to make the Program that has five Steps, shown in Table 2. Apply the name Program 01 to it. As so far the Sequence File you are making has only been initialized, the number of Steps the Sequence is composed of is not specified yet. That is, no steps can be edited in the Program 01 under this status. Program 01 will consist of five Steps and you need to prepare the area where the steps can be written. This is called, "Securing step area." Now, secure the area for five steps by the following procedure.

Securing Step Area

```
>1:Edit Program
 2:Edit Sequence
```

1) First, choose 1:Edit Program, and then the following display will appear.

```
Program:01 New
 000
```

On the above display, the 01 indicates the Program number, and 000 on the second line shows the number of Steps currently owned by Program 01. At this point in time, the step area is zero and a message "New" is displayed besides the program number. Here, by pressing the **ENTER** key, you can see the setting of each Step.

```
N001
   New
```

2) The Program 01 has no steps yet, only a message "New" was displayed above. Therefore, you have to secure the step area and insert it in the Program. Press the **ENTER** key to invoke the Edit Menu,

```
N001
    1:Modify
```

```

   >2:Insert
```

press the down-arrow key and then choose 2:Insert.

3) Since you are prompted to enter how many steps to be inserted,

```
Insert:001
 How many steps?
```

Insert five Steps by pressing **5 + ENTER**.

```
Insert:005
 Completed
```

The area for the five steps has been secured and you will see the following message.

```
N001 ■ 0.00V
      .O.       0.1s
```

From now on, it is possible to edit Steps.

3.2.3 Editing Steps

Now, shown in the above display are the contents of a Step used in Normal Sequence. There are six items in the case of NV sequence, left-arrow and right-arrow keys allow you to move the cursor (■) among the items, and you can edit where the cursor is placed.

In the above example, since the cursor is blinking on the term "S," here you can specify transition type of output (Step/Ramp transition). Look at the Step Editing Display a little more in detail.

```
N001 S 0.00V
      .O.       0.1s
```

1) N denotes it is a Normal Sequence. The subsequent 001 indicates that it is the first of the five Steps, in other words, this Step is to be executed first when the Program 01 is invoked. If you want to move to another step number, use the up-arrow and down-arrow keys.

2) S (or R) specifies which transition type (Step or Ramp) is to be applied to the output transition. (**1** for Ramp; **0** for Step)

3) 0.00V is the voltage this Step generates. Enter it with the numeric keys.

4) The subsequent three will show messages Trig, Out, and Pause, depending on the migration of the cursor. These have the following meanings:

```
Trig   .O.
```

```
Out    .O.
```

```
Pause  .O.
```

Trig:	When this Step is executed, a trigger signal is generated at the Trigger Output terminal.
Out:	When this Step is executed, the output is turned on.
Pause:	Holds the output status, whilst temporarily interrupting the execution of this Step.

These are implemented in every step as additional functions and you can enable them by pressing the **1** key and can disable them with the **0** key. (The indication at the cursor becomes a dot if disabled; the indication at the cursor becomes initial for each item if enabled.) For the Out item, since the output should be usually enabled, "O" is applied as default. Also see the operation manual for more information about these functions. Let's procede.

5) Finally, 0.1s on the display is the execution time for the Step. Enter it with the numeric keys.

Well, try to set each step as listed below. Saying again, left-arrow and right-arrow keys can move the cursor among items, settings of S/R, Trig, Out, and Pause are with the **1** and **0** keys, and confirm a numeric entry with the **ENTER** key. The up-arrow and down-arrow keys also allow you to jump to an other step.

Step 001

```
N001 S 0.00V
      .O.        1.0s
```

Step 002

```
N002 S 5.00V
      .O.        2.0s
```

Step 003

```
N003 S10.00V
      .O.        3.0s
```

Step 004

```
N004 S 5.00V
      .O.        1. 0s
```

Step 005

```
N005 S 0.00V
      .O.        1. 0s
```

After data entry, press the **ESC** key and return to the Program Display as below. You will see that five steps have been inserted.

```
Program:01
 005
```

Program 01 is completed by the work so far. Now, you'd better confirm whether the input work is successful. Tentatively, press the **RUN** key, then the waveform you created will be generated once. Like this trial, when you press the **RUN** key at the Program Display in the Edit Program under the Edit Menu, you can run a half-finished Program alone. It is convenient to remember this operation.

To finish the practice, create a little more complex pattern in Chapter 4 and Chapter 5.

Chapter 4
Practicing Normal Sequence File

You will learn how to make a Normal Sequence file here with a more complex example.

Well, the sequence you create in this chapter is a little complex. It is to execute the following patterns:

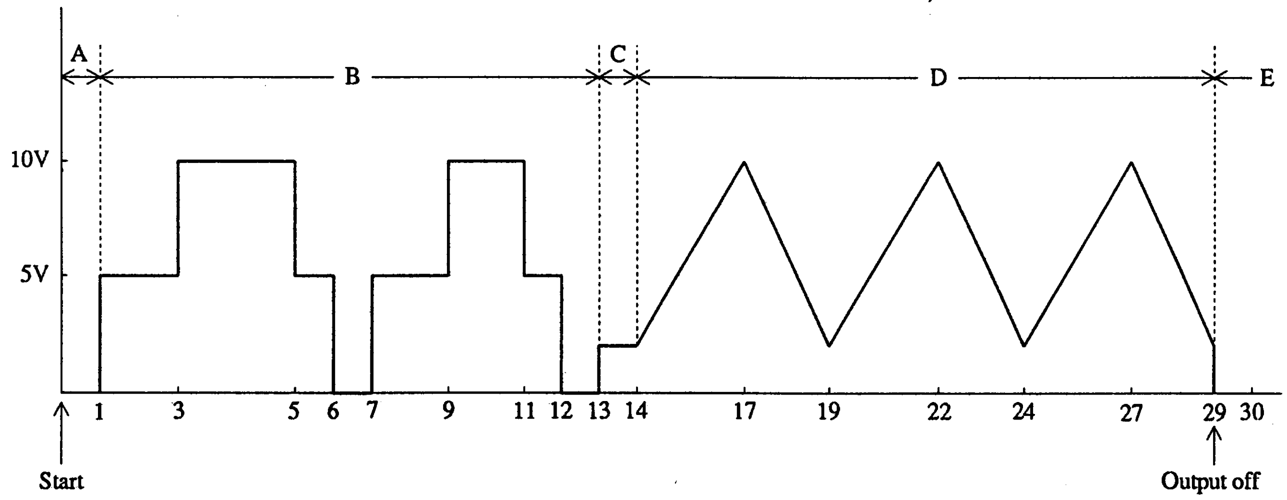


Figure 13 Decomposing a Complex Pattern

4.1 Decomposing the Pattern

In the beginning, you decompose the pattern. Although you may do it as you please, it is important to divide it into Start, Loop (repetition), Chain, and End sections. In this example, you can decompose it into five components A to E as illustrated in Figure 13, and each component is numbered as a ***Program***. Expressed in words, they are as in Table 4.

Table 4

A	Step 001:First, 0V for 1 second	Program 01	Sequence 1
B	Step 001: 5V for 2 seconds (Step transition) Step 002: 10V for 3 seconds (Step transition) Step 003: 5V for 1 second (Step transition) Step 004: 0V for 1 second (Step transition) (Repeat Step 001 thru 004 twice.)	Program 02	Sequence 2
C	Step 001: 2V for 1 second (Step transition)	Program 03	Sequence 3
D	Step 001: 2V to 10V taking 3 seconds (Ramp transition) Step 002: 10V to 2V taking 2 seconds (Ramp transition) (Repeat Step 001 thru 002 three times.)	Program 04	Sequence 4
E	Step 001: 0V and output off	Program 05	End Program

There are nine steps in total. Usually, the contents of Table 4 are to be written in the supplement Sequence Coding Sheet. Try to fill it in according to the examples in the operation manual.

NOTE

Because the Sequence File under editing is on the volatile Edit Memory, the file can be lost if the instrument power is turned off during operation. From now on, you'd better create the Sequence File at a stretch following the procedure until the file is preserved onto the non-volatile memory (File #0). If you have mistaken the operation while editing the sequence, it is recommended to return to the last position saved, and then edit it over again.
Also, if you get confused, edit it all over again from Creating a New Sequence File. Since you will return to the Root State by pressing the **ESC** *key several times, you can restart from there.*

4.2 Creating a New Sequence File

In addition to the practice of the previous chapter, you will create a Sequence File here. Therefore, the program previously created will be lost. How to save the file is described in this chapter.

Create New File

1) From the Edit Menu, choose 3:New.

```
  2:Edit  Sequence
>3:New
```

2) Here, you create a file in a similar way to Chapter 3. What you create here is a sequence of "NV mode, sec unit."

```
Create New Sequence
    Sure ?
```

Press the **ENTER** key, then choose mode NV and time unit sec.

```
Mode:NV
Unit:sec
```

The Editing Displays of the Normal Sequence are shown in Figure 14 and 15; refer to them if necessary. At this point, we will end the practice of the Normal Sequence.

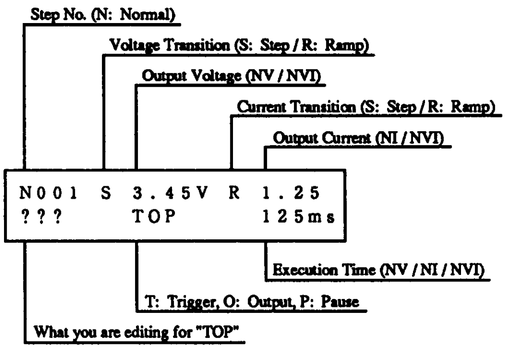

Step No. (N: Normal)
Voltage Transition (S: Step / R: Ramp)
Output Voltage (NV / NVI)
Current Transition (S: Step / R: Ramp)
Output Current (NI / NVI)
N001 S 3.45V R 1.25
??? TOP 125ms
Execution Time (NV / NI / NVI)
T: Trigger, O: Output, P: Pause
What you are editing for "TOP"

Figure 14 Step Editing Display

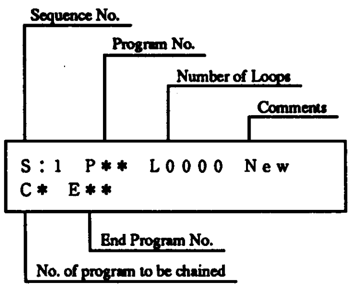


Figure 15 Sequence Editing Display

4.3 Creating a Program

4.3.1 Securing the Step Area for Program 01

After declaring a new file, create the first Program shown in Table 4. According to Table 4, since Program 01 is composed of one step, what you should do next is to secure the area for a single step. The procedure here is the same as the practice of 3-2-2. So, we will not explain how to operate in detail here.

1) From the Edit Menu, choose 1:Edit Program.

```
Program:01  New
 000
```

Pressing the **ENTER** key, the following message will appear.

```
N001
 New
```

2) "New", because no step is inserted yet, so secure the step area.

```
N001
    1:Modify
```

```

   >2:Insert
```

```
Insert:001
 How many steps?
```

Insert the area for one step by pressing **1 + ENTER**.

4.3.2. Editing Step

As the secured area is for a single step, edit it here. You practiced editing five Steps in Chapter 3. Therefore, since only one step is to be edited here, it should't be too dificult.

Step 001

```
N001  S  0.00V
       .O.        1.0s
```

If you have mistaken the setup values and you'd like to alter them afterwards, follow the procedure of reorganization described hereafter. If not required, it's enough just to skim through.

4.3.3 Reorganizing a Program

Here, we will explain how to modify the contents of Step owned by the existing Program, how to insert new Steps in the Program, and how to delete Steps from the Program.

First, return to the Root State by pressing the **ESC** key several times, and then choose 1:Edit Program under the Edit Menu.

```
Program:01
 001
```

Now, remember that you can move the current work space to other Program numbers with the up-arrow and down-arrow keys. It is possible to edit them by this operation. The 001 in the display indicates that the number of Steps of the Program is 1. So, 001 isn't always displayed here.

After the **ENTER** key is pressed, the contents of the beginning step, that is Step 001, will be displayed.

```
N001  S  0.00V
                    1.0s
```

1) Modifying Steps

Move to the Step number you want to edit with the up-arrow and down-arrow keys, and then press the **ENTER** key to invoke the Edit Program Sub Menu. After you select ***Modify*** and the Step Editing Display has appeared, you can reorganize the contents with a procedure similarly to 3.2.3.

2) Inserting Steps

Move to the Step number immediately behind the position you want to insert at with the up-arrow and down-arrow keys. (For instance, if you would like to insert after the forth, this is the fifth. And if you want to append as the last step, it is EOS.) Press the **ENTER** key, and then invoke the Edit Program Sub Menu. After choosing ***Insert***, since you are asked how many steps are to be inserted, enter the number of steps to be newly inserted. The new steps will be numbered sequentially and, you are ready to edit the new steps.

3) Deleting Steps

Move to the top number of the Steps you want to delete by using the up-arrow and down-arrow keys. (For instance, if you would like to delete the forth and fifth steps, this is at the forth.) Press the **ENTER** key, and then invoke the Edit Program Sub Menu. After choosing ***Delete***, since you are asked how many steps to be deleted, enter the number of steps to be deleted. The remaining steps will be renumbered and, you are ready to edit the existing steps.

4.4 Creating a Sequence

Next, you will create the Sequence that specifies how Program 01 is to be executed.

So far, the procedure didn't have any changes compared to the practice of Chapter 3. However, the Sequence explained hereafter is an important item that specifies what in order the Programs run, and it is always required when you construct a complex Sequence File.

First, to set the Sequence, return to the Root State, and, proceed as below.

1) Press the **EDIT** key to invoke the Edit Menu.

```
>1:Edit Program
 2:Edit Sequence
```

2) Scrolling the menu with the down-arrow key, choose 2:Edit Sequence. Then the following Sequence Editing Display will appear.

```
S:1 P01 L0000 New
C* E**
```

4.4.1 Setting Sequence Items

Shown in the above display are the contents of the ***Sequence***. There are four items and you can move among these items with the left-arrow and right-arrow keys. The item where the cursor is placed is what you can edit. Now, look at the Sequence Editing Display in more detail.

```
S:1 P01 L0000 New
C* E**
```

1) The leading S:1 is the Sequence number. There are up to eight Sequences, the up-arrow and down-arrow keys may be used for moving to other Sequence numbers.

2) The P specifies the Program number from which the Sequence starts. The 01 means "Begin from Program 01."

3) The L specifies how many times the Program is to be repeated (number of Loops). As for 0000, the sequence doesn't actually run because zero loop is merely an initial value, and it will generate an error. Therefore, it is necessary to set at least 0001 or more to the Loop. To set it, use the numeric keys.

4) The C specifies a Sequence number to which this Sequence is to be chained after execution (or after completion of the loop specified by P). This capability is what we call Chain and is set with the numeric keys. The asterisk (*) means "Don't Chain anywhere" and then the sequence will be chained to the End Program described later. To specify the asterisk (*), move the cursor to the C item and then enter **0** with the numeric key. Also, specifying a Sequence number identical to the current Sequence number will produce an Endless Sequence.

5) The E specifies a Program number to be invoked when the sequence has been interrupted by the **STOP** key. This Program is called ***End Program*** and it specifies how the supply output should behave (for instance, turning output off). (However, if the End Program contains two or more Steps, only the leading Step will be executed.) Figure 16 shows the relationship between End Program and Chain.

   Here, note that what is specified here isn't a Sequence number but a ***Program*** number. Also, the Sequence jumps and comes to the End Program when the item of C (Chain) is set to " * " (No Chain).

   The doubled asterisk (* *) means "No End Program," then no Program will be invoked. In such case, when the sequence has been interrupted the output currently set will be held and, when the sequence has been normally completed the final output state will be kept.

   Well, let's try to set the Sequence 1 as follows. Saying again, the left-arrow and right-arrow keys are for moving among items, numeric keys for numeric entries, and the **0** key specifies " * " for "none" settings.

```
S:1  P01  L0001
C2  E**
```

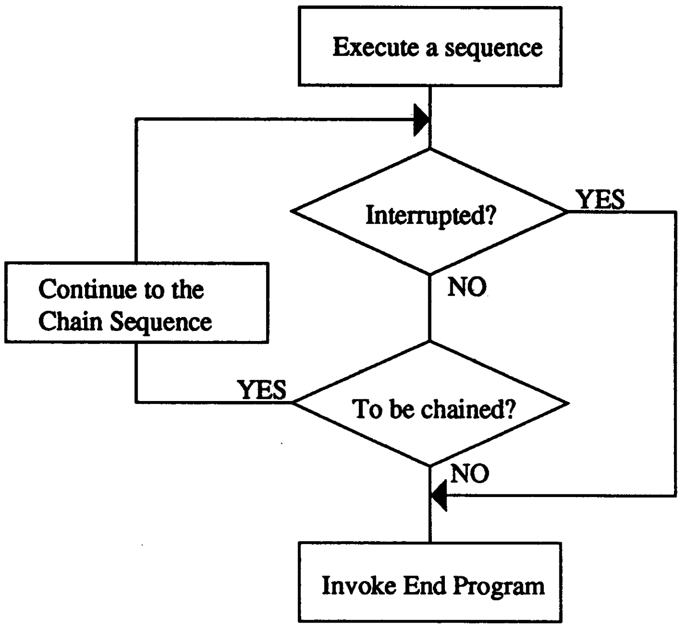


Figure 16 Relationship between Chain Sequence and End Program

4.4.2 Reorganizing a Sequence

If you need to edit Sequences again, press the **ESC** key several times to return to the Root State and then choose 2:Edit Sequence under the Edit Menu. You can reorganize as well as create in a similar manner to reorganizing and creating Programs.

4.5 Preserving a Sequence File

The Sequence created so far is still written in the volatile Edit Memory. Therefore, the contents of the Sequence File constructed with a toiling job can be easily lost if the instrument power is shut down. (Of course it will never be lost if you keep the instrument turned on.) To avoid loss, you should preserve it in the non-volatile memory (File #0) built in the instrument.

NOTE *If the existing contents of File #0 are the example sequence there's no problem with overwriting it with a new sequence, but otherwise, the existing contents will be lost. In this case you'd better consult the person who has created the Sequence File.*

Saving a File

1) Invoking the Sequence Menu from the root state with the SEQ key and choosing 1:File and then 3:Save File,

```
 1: List Files
 2: Load File
```

```

>3: Save File
```

the following display will appear.

```
Save:000  SAMPLE
NVI Step:009
```

Displayed here are the contents of the file already written. The number 000 indicates the File #0 built in the instrument.

2) In this situation, since you save the sequence in File #1, press the **ENTER** key.

```
Save:000
 Completed
```

Now, the contents of the sequence you have created so far has been saved onto the non-volatile File #0, from now on, turning off the instrument won't cause losing the contents.

4.6 Creating Remaining Programs and Sequences

4.6.1 Creating Program 02

In a similar way to how you have learned, create the Program 02 shown in Table 4 and register it as the ***Start Program*** of the Sequence 2. According to Table 4, the Program 02 is will have four Steps.

```
Program:02 New
 000
```

1) First, secure the area for four steps and begin to edit.

   Step 001: 5V for 2 seconds (Step Transition), output on

```
N001 S 5.00V
     .O.          2.0s
```

   Step 002: 10V for 3 seconds (Step Transition), output on

```
N002 S10.00V
     .O.          3.0s
```

   Step 003: 5V for 1 second (Step Transition), output on

```
N001 S 5.00V
     .O.          1.0s
```

   Step 004: 0V for 1 second (Step Transition), output on

```
N001 S 0.00V
     .O.          1.0s
```

2) Next, register this Program to Sequence 2.
   Sequence 2: (Start 2, Loop 2, Chain 3, and End 5)

```
S:2 P02 L0002
C3 E05
```

Note that the Step numbers are allocated locally to each Program. That is, although the Step 001 exists in above-mentioned Program 01, the Steps are also numbered from the beginning in Program 02.

4.6.2 Editing Programs 03 through 05

Continuing, edit Programs 03, 04, and 05.

1) Program 03

The Program 03 has one Step and is registered as the Start Program of Sequence 3.

Step 001: 2V for 1 second (Step transition), output on

```
N001  S  2.00V
        .O.          1.0s
```

Sequence 3: (Start 3, Loop 1, Chain 4, and End 5)

```
S:3  P03  L0001
C4  E05
```

2) Program 04

The Program 04 has two Steps and is similarly registered as the Start Program of the Sequence 4.

Step 001: To 10V for 3 seconds (Ramp transition), output on

```
N001  R10.00V
        .O.          3.0s
```

Step 002: To 2V for 2 seconds (Ramp transition), output on

```
N001  R  2.00V
        .O.          2.0s
```

Sequence 4: (Start 4, Loop 3, no Chain, and End 5)

```
S:4  P04  L0003
C*  E05
```

In Program 04, the output shows a Ramp transition. Having already understood this replace the item "S" with "R" in the Step Editing Display. (Pressing **1** for R, **0** for S) In Ramp transition, the output changes gently from the output setup value of the previous Step, to the setup value of the Step where "R" is specified, taking specified elapsing time. (See Figure 17.)

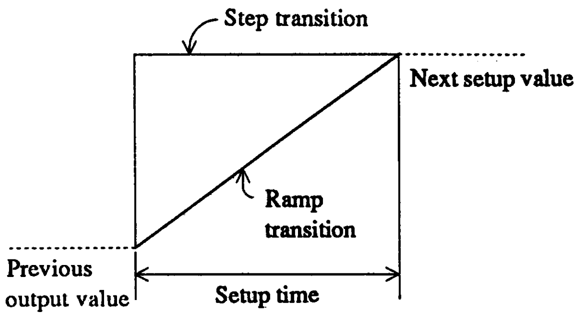


Figure 17 Step vs. Ramp

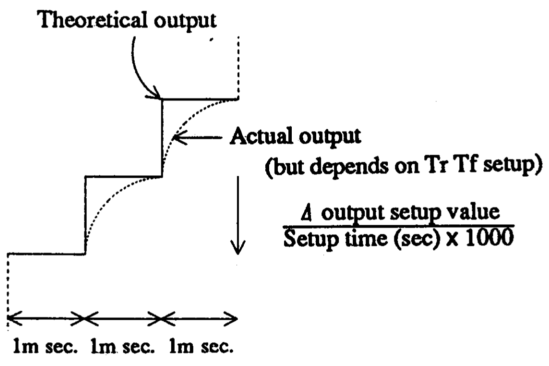

Theoretical output
Actual output
(but depends on Tr Tf setup)
Δ output setup value / Setup time (sec) x 1000
1m sec. 1m sec. 1m sec.

Figure 18 Ramp Transition Enlarged

Actually, in Ramp transition operation, the output is basically approximated by digital values. Therefore, the resolution of the output is limited, even though before it was said to vary gently, and strictly speaking, it changes as illustrated in Figure 18.

NOTE *In a Step specified as a Ramp transition, the output changes gently to the setup value of the Step, starting from the previous setup value. Therefore, if you assign the Ramp transition to the leading Step of the executive sequence, you have to confirm the voltage and current setups applied as defaults before execution. Especially, if you plan to perform the AutoRun function, you must pay attention to the contents of Setup Memory #00 that will be set as the power-on default. For instance, imagine that the contents of Setup Memory #00 are 10V and the leading Step is specified as "Ramp/5V/3sec." In this case, the AutoRun function will make the output voltage change from 10V to 5V, taking 3 seconds.*

3) Editing Program 05

For the final Program 05, since it is used as the End Program in this example, it's enough to prepare only one Step. Here, note that it is not needed to register Program 05 to Sequence 5.

Program 05 is the End Program. As explained previously Program 05 should be invoked when the **STOP** key is pressed during sequence execution, or should be invoked as a termination of the sequence. Therefore, the Sequence that has an End Program as a Start Program or Chain never exists.

Then, create the End Program.

Step 001: 0V (Step Transition), output off

```
N001  S  0.00V
        ...          0.1s
```

Since setting the execution time doesn't make sense in this situation, you may leave it at 0.1 seconds or any other value. Now, editing Program 03 through 05 has finished.

According to the above, the Sequence File that generates the operation shown in Table 4 has been completed. However, the contents entered since Sequence 2 (or Program 02) are still placed in the volatile Edit Memory. So, you'd better save it again in File #0 where Sequence 1 is already preserved.

To save the file, press the **SEQ** key from the Root State, invoke the Sequence Menu, and 1:File, and then choose 3:Save File. After preservation, confirm the content of File,#0 under 1:List Files. It is okay if the following message is visible.

```
List:000
NV Step:009
```

4.7 Execution of a Sequence

Now, let's execute the sequence you created. Return to the Root State pressing the **ESC** key several times. (Restarting the instrument power also returns it to the Root State.) As it's only practice, it is needless to connect a load to the output terminal on the instrument. The execution result will appear as a voltage waveform output, so it will be a good idea to connect a digital storage scope to the output terminals to watch it.

Execute The Sequence

1) Press the **RUN** key.

```
S:1  P01  L001
C2  E05
```

If you see a different number of the Sequence, choose Sequence 1 with the up-arrow and down-arrow keys. After that, press the **RUN** key once more.

```
OUT    0.000V    0.00A
RU  S2, P02, L0002, 0004
```

The Sequence number, Program number, number of Loops, and the Step number currently running will be displayed one after another. Although it's difficult to ascertain the variance of the display when an individual step interval is short, watch it running.

The output waveform should be like as shown in Figure 13 if the sequence is correctly programmed. If wrong, hold out until you get the correct output with doing the reorganization. (It's also practice!)

Practicing for creating Normal Sequence is done here. It has probably taken much time to come this far from the beginning of the practice. But, once you have understood the procedure for creating a file (Figure 9) and the construction of the Sequence File (Figure 6), you can probably complete this example within approximately five minutes.

Chapter 5

Practicing Fast Sequence

You will learn how to make a Fast Sequence file that can simulate a high-speed waveform.

In this chapter, you will learn how to create the Fast Sequence file. For the Fast Sequence, the method of making a file is almost the same as for the Normal Sequence. It's potentially simpler than the Normal Sequence because of the fewer items you need specify.

Here, we will describe how to create a sequence pattern illustrated in Figure 19 with using the Fast Sequence.

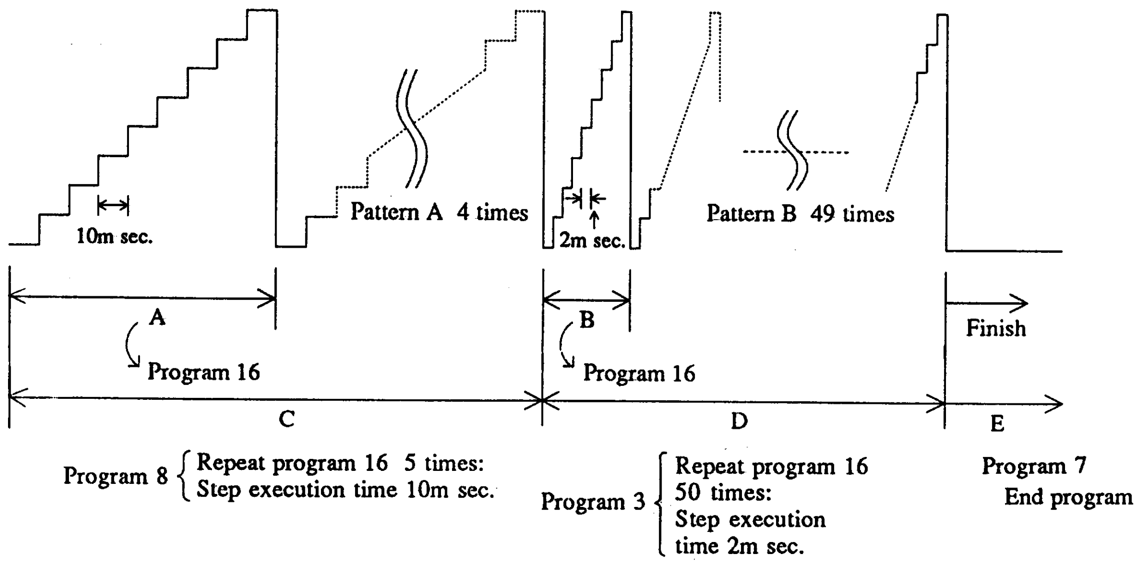

10m sec.
Pattern A 4 times
2m sec.
Pattern B 49 times
A
Program 16
B
Program 16
Finish
C
D
E
Program 8 { Repeat program 16 5 times: Step execution time 10m sec.
Program 3 { Repeat program 16 50 times: Step execution time 2m sec.
Program 7 End program

Figure 19 An Example of Fast Sequence

5.1 Decomposing the Pattern

First, look at the parts shown by A and B in Figure 19. Although each Step uses an independent execution time, every step conforms to an identical pattern in that the output increases from 0V to 8V in 1V steps. (As explained in Chapter 1, the Program specifies only the peak output value and the interval of a Step is decided by the Sequence in Fast Sequence.) This entire pattern is repeated in Figure 19, therefore assume this cluster to Program 16.

In the range specified by C, Program 16 is repeated five times with 10 milliseconds execution time for each step. Assign it to Sequence 8. Whereas in the range indicated by D, Program 16 is repeated fifty times with a 2 millisecond execution time for each step. Assign this to Sequence 3.

Finally, E indicates a status after this sequence done. When Sequence 3 finishes, the voltage output value is still identical to the final step of Sequence 3 (8V), and it must be reduced to zero. This final part should be written in Program 07 as the End Program. Therefore, the settings of Program and Sequence are as follows.

Program:

Program 16	Increases 0V to 8V by 1V steps (9 Steps)
Program 07	End Program of 0V output (1 Step)

Sequence:

Sequence 8	Execute Program 16 in 10 millisecond step execution, 5 times repetition, and chain to Sequence 3.
Sequence 3	Execute Program 16 in 2 millisecond step execution, 50 times repetition, and invoke the End Program.

We assigned random orders to the Sequences and Programs here. It is not so important to allocate well-ordered numbers to them, and we hope you understand you can combine eight Sequences and sixteen Programs arbitrarily in any order.

5.2 Creating a New Sequence File

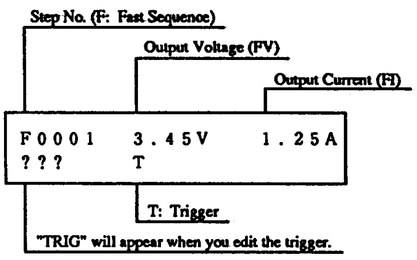
Now, you will first create a Sequence File for Program 16 and Program 07 (End Program). Because you have already practiced how to create a new file from Chapter 3 onwards, we need not describe this here in detail again.

Practice here is on the Fast Sequence, so create a file in the FV mode. (The mode can be chosen with the up-arrow and down-arrow keys.)

```
Mode:FV
```

Since Fast Sequence fixes the time unit in msec, there's no item for the time unit.

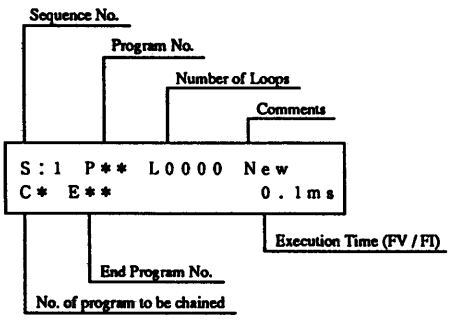
The Editing Displays of the Fast Sequence are shown in Figure 20 and 21; refer to them if necessary. At this point, we will end the practice of the Fast Sequence.



```
F0001   3.45V    1.25A
???     T
```

Step No. (F: Fast Sequence)
Output Voltage (FV)
Output Current (FI)
T: Trigger
"TRIG" will appear when you edit the trigger.

Figure 20 Step Editing Display



```
S:1  P**  L0000  New
C*  E**          0.1ms
```

Figure 21 Sequence Editing Display

5.3 Making a Program

5.3.1 Making Program 16

To begin with, create Program 16. As the Program 16 is composed of nine steps, secure the area for these nine steps.

1) Invoke Edit Menu with the **EDIT** key, and choose 1:Edit Program. (Then jump to the Program 16 with the down-arrow key.)

```
Program:16 New
 0000
```

2) Pressing the **ENTER** key twice, invoke the Edit Program Sub Menu, and then select 2:Insert.

```
F0001
    1:Modify
```

```

   >2:Insert
```

3) And, secure the area for the nine steps pressing the **9 + ENTER** keys.

```
Insert:0009
 Completed
```

The area has been secured.

5.3.2 Editing Steps

Here you edit the nine Steps secured above. How to edit Steps is basically similar to Normal Sequence. However, because several items are omitted in Fast Sequence, (there are no items for Pause and Out), it is easier.

The Pause function is unavailable in the Fast Sequence. Also, whether the output is delivered or not depends on the output on/off setting of the instrument before the sequence execution. Therefore, you must turn on the instrument output explicitly, before starting the sequence. Moreover, there's no item for the execution time of each Step, as specified in the Normal Sequence.

Then, set the nine Steps as described below.

Step 0001

```
F0001  0.00V
          .
```

Step 0002

```
F0002 1.00V
        .
```

Step 0003

```
F0003 2.00V
        .
```

Step 0004

```
F0004 3.00V
        .
```

Step 0005

```
F0005 4.00V
        .
```

Step 0006

```
F0006 5.00V
        .
```

Step 0007

```
F0007 6.00V
        .
```

Step 0008

```
F0008 7.00V
        .
```

Step 0009

```
F0009 8.00V
        .
```

The Program 16 created here will be doubly registered as a leading program of Sequence 8 and Sequence 3. However, you will do this registration work (setting Sequences) later.

5.3.3 Making the End Program

What you will make next is Program 07, used for the End Program. This Program has one step area and it should be set as follows.

Step 0001

```
F0001  0.00V
          .
```

The value to be set for this Step is 0V. Immediately after the area is secured, you have nothing to do since the initial value is always 0V.

5.4 Making Sequences

Here, register the Program 16 created above to two Sequences with independent time units.

In Normal Sequence, the execution speed of the sequence is decided in each Step (that is, in each Program that has steps). Whereas in the Fast Sequence, the speed is decided within a Sequence. Therefore, such a trick can be performed.

How to edit the Sequence is quite the same as for Normal Sequence. Set Sequence 8 and Sequence 3 as follows.

Sequence 8

```
S:8  P16  L0005
C3  E07          10.0ms
```

Sequence 3

```
S:3  P16  L0050
C*  E07          2.0ms
```

Constructing the file for Fast Sequence is completed here. You'd better preserve it in the non-volatile memory (File #0).

5.5 Execution of a Sequence

Well, let's try to execute the completed sequence. Return to the Root State by pressing the **ESC** key several times, and remove any loads from the output terminals of the instrument.

Execution of the sequence

1) Press the **RUN** key

Since the Sequence to be executed first is the eighth, use the up-arrow and down-arrow keys to choose Sequence 8.

```
S:8  P16  L0005
C3  E07
```

By the way, don't forget to turn on the output of the instrument. The output won't be delivered unless you turn the output on in advance, because the Fast Sequence has no capability of turning the output on/off.

After turning the output on with the **OUTPUT** key, press the **RUN** key.

```
OUT      0.000V     0.00A
RU  S8, P16, L0005, 0005
```

On the display, while the sequence is running, the part of each item varies round and round and the "RU" message is blinking. After the sequence is done, the display will hold. Because 0V is specified in the End Program, the output voltage will show zero.

NOTE *This chapter describes an example of the chained sequence for the Fast Sequence. The Fast Sequence has priority in the high-speed operation steps. Thus, care must be taken when using the Fast Sequence since the final step execution time of a program differs from the set value if sequences are chained.*
In the example of Figure 19, the pattern A final step execution time differs from the set value when sequences C and D are chained whereas the pattern B final step execution time differs from the set value when sequences D and E are chained.

Index

Alphabetic order