20 Tabular Statistics

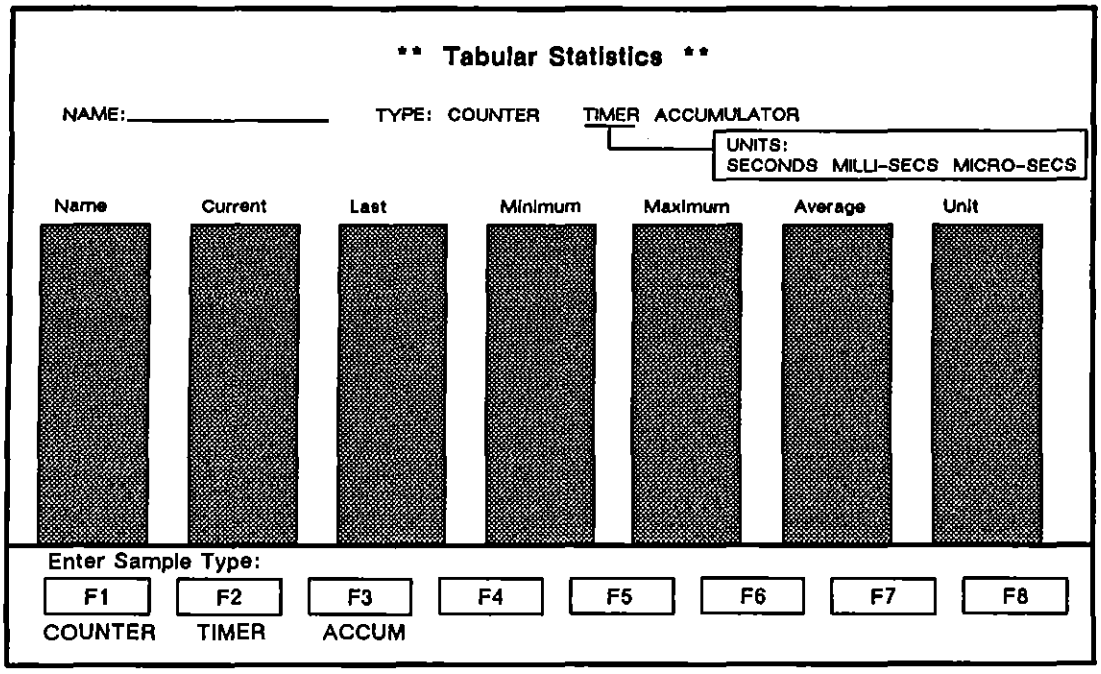


Figure 20-1 Menu fields, Tabular Statistics screen.

20 Tabular Statistics

The user of the INTERVIEW can assign tasks easily to an almost unlimited number of software counters and timers. When these incrementing counters and timers are sampled—that is, when they are read and cleared—their current totals are factored into a statistical breakout on the Tabular Statistics screen. This breakout is a real-time reading of current, last, minimum, maximum, and average values for the counter or timer.

At any one moment, the Tabular Statistics screen displays a maximum seventy-five values for fifteen counters, timers, and accumulators. (Accumulators are defined below in Section 20.4). The Graphic Statistics screen, treated in the next section, displays less information (sixteen values total) at any one time, but in a graphic format. Both statistics screens can be scrolled up or down to display additional rows of values.

The role of triggers in creating, operating, sampling, and accumulating various counters and timers is common to both screens and will be discussed here under Tabular Statistics.

20.1 Counters and Timers

Counters and timers are operated as *actions* on trigger menus and in Protocol Spreadsheet tests. In the example below, two different counters are made to increment as spreadsheet actions.

In this Bisync example, polling address A represents a drop on a multipoint circuit. The string "%A" on the DCE side is the beginning of a text block originating at remote drop A. When the spreadsheet program sees this string, it moves to a state called drop_a, where the end of every text block (DCE STRING "$^{F}_{F}$") increments one counter (allblk_a) and only blocks that end with a bad BCC increment another counter (badbcc_a).

```
LAYER: 1
     STATE: stx
        CONDITIONS: DCE STRING "%A"
        NEXT_ST: drop_a
     STATE: drop_a
        CONDITIONS: DCE BAD_BCC
        ACTIONS: COUNTER badbcc_a INC
        CONDITIONS: DCE STRING "^F^F"
        ACTIONS: COUNTER allblk_a INC
        NEXT_ST: stx
```

The current value of a counter also can be used as a *condition* either on a trigger menu or in a Spreadsheet test. Here is an example of a counter performing this "countdown" function:

```
CONDITIONS: COUNTER allblk_a EQ 1000
ACTIONS: COUNTER badbco_a SAMPLE
```

(The SAMPLE action is explained in Section 20.3, below.)

Timers are not used as trigger conditions, since timeouts serve this function.

20.2 Preparing the Tabular Statistics Screen

Current values of counters and timers are read on the Tabular Statistics and Graphic Statistics screen. Both statistics screens are always accessible by softkey during Run mode. A counter or timer that is named in a trigger must be identified by name on the statistics screen. This naming is done in Program mode prior to the run.

Press [PROGRAM] and then [F4] for the Statistics Menu screen. Press [F1] (or [RETURN]) to enter the Tabular Statistics screen. In Program mode, the screen shows fifteen tabular rows beneath a single line of menu fields. There are two cursors, one on the menu line and one in the table. See Figure 20-2. The fields on the menu line (second line at the top of the screen) always refer to the row in the results table that has the lower cursor.

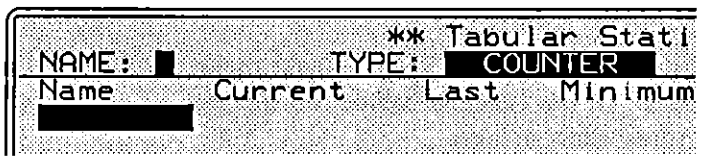


Figure 20-2 In Program mode, this screen has two cursors.

When you enter the Tabular Statistics screen, the upper cursor is in the **Name** field on the menu line, while the lower cursor is in Row One of the table.

Press [DONE] or [RETURN] (or [→] and [←]) to move the upper cursor from field to field in the menu area of the screen. Press [ROLL FWD] and [ROLL BACK] to change the selections in rotating-window fields.

Press [↓] and [↑] to move the lower cursor from display line to display line. The menu selections at the top of the screen will change as you cursor down the screen, since they are always keyed to the display line that has the cursor.

Each time the cursor advances one row down the table, the information that the user has entered on the menu line is written to the previous row in the table. In the top half of the sequence in Figure 20-3, the user has entered a name on the menu line

above a blank table. The cursor is now in the Type field, where COUNTER was the default selection. The bottom of the figure shows the resulting table after the user presses ↓ .

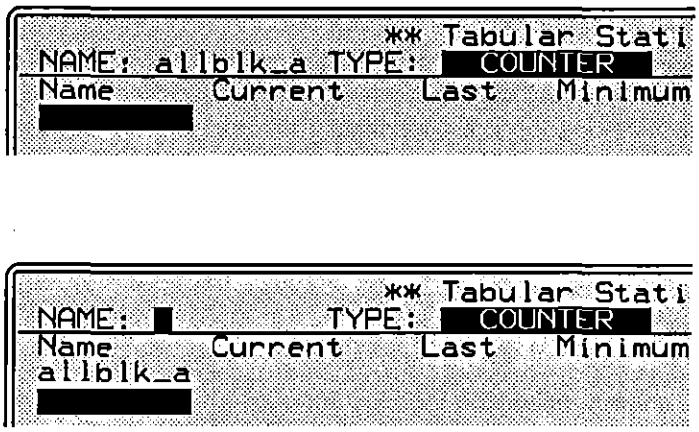


Figure 20-3 When lower cursor moves, user data is written to the vacated display line.

↑ reverses the direction of the lower cursor. The user may name or revise counters, timers, and accumulators by moving up the table as well as down.

The tabular area of the screen is a scrolling display of variable length that sets a very high limit (100) on the number of counters, timers, and accumulators that can be named by the user. To scroll down the directory, position the cursor on the last line of the listings and press ↓. This keystroke will display new lines one at a time. Or press NEXT PAGE to display fifteen new lines of counters, timers, and accumulators. SHIFT and ↓ together move the cursor to the end of the listings.

Position the cursor at the top of the listings and press ↑ to expose lines that have scrolled off the screen at the top. Or press PREV PAGE to retrieve an entire previous page of listings. SHIFT–↑ will always move you to the top of the listings.

INSERT LINE and DELETE LINE are operative keys on the scrolling statistics tables.

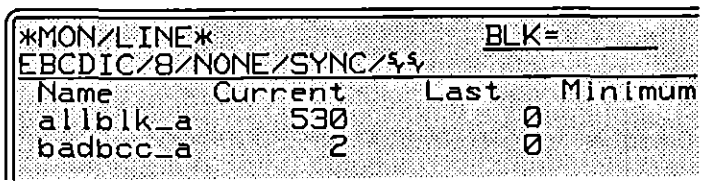


Figure 20-4 All counting and timing is performed in the Current column.

When [NUM] is pressed, a counter or timer that has been named in a trigger action will show its current value next to its name on the statistics screen. Figure 20-4 is a Run-mode display of two counters, one that is incrementing with each text block sent by a particular remote drop, and a second that is incrementing with each bad BCC from the same source.

If you have named a counter (or timer or accumulator) in a trigger action but forgotten to identify it on a statistics screen, the statistics will still be available in Program mode following the run (provided you have *sampled* the counter or timer at some point during the run). To view the statistics, simply identify the counter (timer, accumulator) by name on the statistics screen and move the lower cursor. The statistics from the previous run will appear on the screen next to the name.

20.3 Sampling Current Values

In addition to current value, the Tabular Statistics screen has columns for last, minimum, maximum, and average values. See Figure 20-5. Unit is not a value column. It applies to timers only, and reflects the unit of time—second, millisecond, or microsecond—selected by the user for that timer on the menu line during Program mode.

Last, Minimum, Maximum, and **Average** are statistical columns, based on previous samplings of the Current column. Sampling is a trigger action that reads the current value of the counter or timer and then resets it to zero. The **Last** column receives the sampled value. The other columns—**Minimum, Maximum,** and **Average**—compare the sampled value with previous samples.

```
*MON/LINE*                    BLK=_____   06/16/89 13:17
EBCDIC/8/NONE/SYNC/%%
 Name      Current     Last   Minimum Maximum  Average     Unit
 allblk_a      999        0
 badbcc_a        4        0
```

```
*MON/LINE*                    BLK=_____   06/16/89 13:18
EBCDIC/8/NONE/SYNC/%%
 Name      Current     Last   Minimum Maximum  Average     Unit
 allblk_a        0     1000
 badbcc_a        0        4         4       4        4.00
```

Figure 20-5 The current count ends when the counter is sampled.

We have already seen a counter that incremented with every bad BCC. A Spreadsheet trigger that *sampled* this incrementing counter every 1000 blocks would maintain a statistical record of errored blocks per thousand:

```
LAYER: 1
    STATE: stx
       CONDITIONS: DCE STRING "%A"
       NEXT_ST: drop_a
       CONDITIONS: COUNTER allblk_a EQ 1000
       ACTIONS: COUNTER badbcc_a SAMPLE
          COUNTER allblk_a SET 0
    STATE: drop_a
       CONDITIONS: DCE BAD_BCC
       ACTIONS: COUNTER badboo_a INC
       CONDITIONS: DCE STRING "F"
       ACTIONS: COUNTER allblk_a INC
       NEXT_ST: stx
```

In Run mode, zero appears in the **Last** column prior to the first sampling of a counter or timer, and nothing appears in **Minimum**, **Maximum**, and **Average** columns. See the top of Figure 20-5. The bottom of the same figure illustrates the effect of the first sampling. The counter named **badboo_a** is cleared automatically but its sampled value is retained in the Last column. Since this is a first sampling, the sampled value is carried over unchanged to the Minimum, Maximum, and Average columns also. Note that the counter named **allblk_a** was not sampled, so it had to be reset manually (**COUNTER allblk_a set 0**).

The next example uses a *timer* in an X.25 environment. A pair of triggers start and sample a timer called t2. Each sample is a measurement of the timeout observed by an X.25 PAD before it responds with an RR to a DCE Info frame. (This timeout is called T2 in X.25.) INFO and RR are spreadsheet conditions in the protocol package for X.25 Layer 2 (see Section 36.)

```
CONDITIONS: DCE INFO GDBCC
ACTIONS: TIMER t2 RESTART
CONDITIONS: DTE RR
ACTIONS: TIMER t2 SAMPLE
```

Figure 20-6 shows a set of results that might be generated by these two triggers.

```
*MON/LINE*                    BLK=_____   06/16/89 13:19
ASCII/8/NONE/BOP
 Name      Current     Last    Minimum Maximum  Average     Unit
 t2              6      110         15     120   111.83 MSECS
```

Figure 20-6 This timer has been sampled several times.

20.4 Accumulators

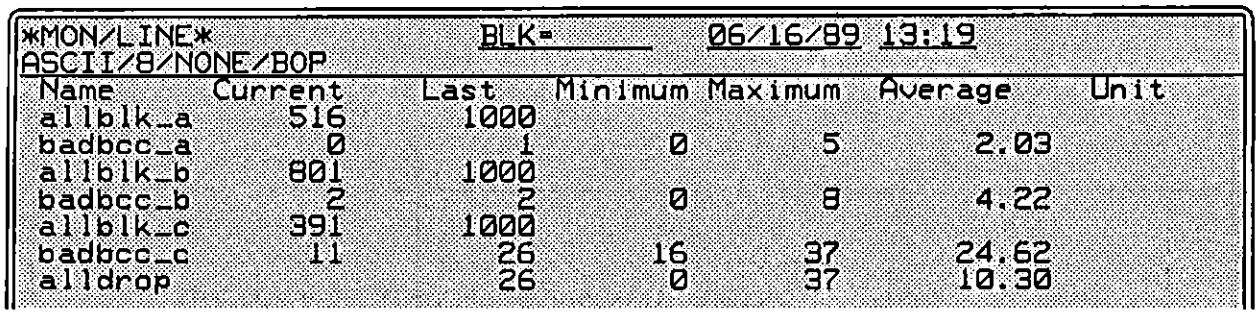
Accumulators look like counters and timers on the statistics screens but they do not increment or reset counters, nor do they start or stop timers. Rather, they *accumulate selected samplings* of these counters and timers without interfering with the counting and timing functions. Thus they enhance the performance of counters and timers by empowering them to work for several accumulators at the same time.

For example, we have already designed a pair of counters that counted bad BCCs per thousand blocks with respect to one drop on a multipoint circuit. We will enlarge the program with a pair of counters for each of two additional drops, drop B and drop C. Then we will add an accumulator to generate error-per-thousand statistics for *the three drops taken together.*

Accumulating is a trigger action found on spreadsheet softkeys but not on trigger menus. The ACCUMULATE actions in our spreadsheet program might look like this:

```
LAYER: 1
    STATE: stx
      CONDITIONS: DCE STRING "%A"
      NEXT_ST: drop_a
      CONDITIONS: COUNTER allblk_a EQ 1000
      ACTIONS: COUNTER badbcc_a SAMPLE
        COUNTER allblk_a SET 0
      CONDITIONS: DCE STRING "%B"
      NEXT_ST: drop_b
      CONDITIONS: COUNTER allblk_b EQ 1000
      ACTIONS: COUNTER badbcc_b SAMPLE
        COUNTER allblk_b SET 0
      CONDITIONS: DCE STRING "%C"
      NEXT_ST: drop_c
      CONDITIONS: COUNTER allblk_c EQ 1000
      ACTIONS: COUNTER badbcc_c SAMPLE
        COUNTER allblk_c SET 0
    STATE: drop_a
      CONDITIONS: DCE BAD_BCC
      ACTIONS: COUNTER badbcc_a INC
        ACCUMULATE alldrop COUNTER badbcc_a CURRENT
      CONDITIONS: DCE STRING "^F"
      ACTIONS: COUNTER allblk_a INC
      NEXT_ST: stx
    STATE: drop_b
      CONDITIONS: DCE BAD_BCC
      ACTIONS: COUNTER badbcc_b INC
        ACCUMULATE alldrop COUNTER badbcc_b CURRENT
      CONDITIONS: DCE STRING "^F"
      ACTIONS: COUNTER allblk_b INC
      NEXT_ST: stx
    STATE: drop_c
      CONDITIONS: DCE BAD_BCC
      ACTIONS: COUNTER badbcc_c INC
        ACCUMULATE alldrop COUNTER badbcc_c CURRENT
      CONDITIONS: DCE STRING "^F"
      ACTIONS: COUNTER allblk_c INC
      NEXT_ST: stx
```

The statistics table now can show results for six counters and one accumulator (Figure 20-7). The accumulator gives last, minimum, maximum, and average error-per-thousand counts based on all the drops on the circuit.



MON/LINE BLK= 06/16/89 13:19
ASCII/8/NONE/BOP

Name	Current	Last	Minimum	Maximum	Average	Unit
allblk_a	516	1000				
badbcc_a	0	1	0	5	2.03	
allblk_b	801	1000				
badbcc_b	2	2	0	8	4.22	
allblk_c	391	1000				
badbcc_c	11	26	16	37	24.62	
alldrop		26	0	37	10.30	

Figure 20-7 The accumulator at the bottom of the table is consolidating errors-per-thousand values from three separate drops.

Not only current values but also last, minimum, and maximum values can be accumulated and broken out statistically. Values in the Maximum column, for example, often are significant *limit* values: time limits, size limits, and so forth. An accumulator might be assigned to sample only this maximum value for several counters or timers running concurrently. The resultant tabular row would be a comparison of these maximum values.

20.5 Keeping a Statistical Log

The sampling action can be used to log statistics at regular time-intervals. In the example that follows, the program counts data packets on an X.25 link and sends a line of date- and time-stamped statistical values every hour on the hour to a serial printer attached to the INTERVIEW. DTE DATA and DCE DATA are packet-level conditions in the protocol package for X.25 Layer 3 (see Section 37). PRINT COUNTER (and PRINT TIMER) is a layer-independent action described in Section 30.4.

```
LAYER: 3
  TEST: paks_per_hr
    STATE: six_am
      CONDITIONS: TIME 0600
      ACTIONS: TIMEOUT sixtysec RESTART 60
      NEXT_ST: hourly
    STATE: hourly
      CONDITIONS: TIMEOUT sixtysec
      ACTIONS: TIMEOUT sixtysec RESTART 60
        COUNTER minutes INC
      CONDITIONS: COUNTER minutes EQ 60
      ACTIONS: COUNTER minutes SET 0
        COUNTER datapaks SAMPLE
        PRINT COUNTER datapaks
      CONDITIONS: DTE DATA
      ACTIONS: COUNTER datapaks INC
      CONDITIONS: DCE DATA
      ACTIONS: COUNTER datapaks INC
```

After several hours, the resulting printout might look like this:

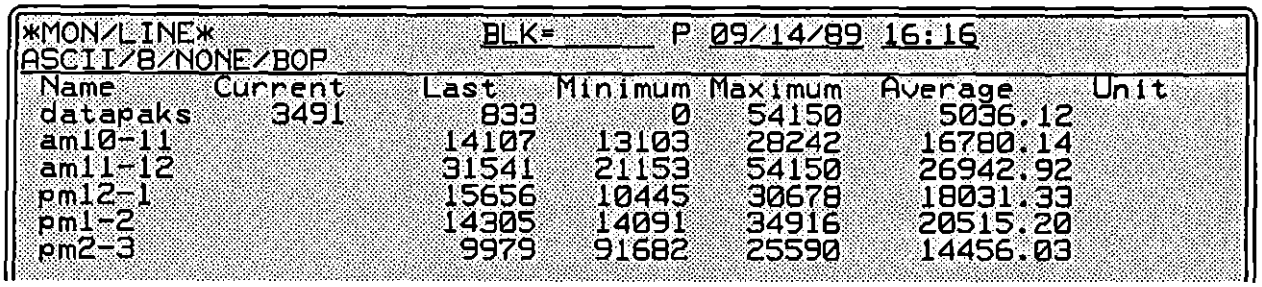
Time	Name	Current	Last	Minimum	Maximum	Average	Unit
09/14 07:00	datapaks	0	0				
09/14 08:00	datapaks	0	0	0	0	00.00	MSECS
09/14 09:00	datapaks	0	820	0	820	410.00	MSECS
09/14 10:00	datapaks	0	3388	0	3388	1402.67	MSECS

Figure 20-8 A counter is sampled every hour and its values are logged to a serial printer.

Accumulators might be added to the original program to gather statistics for a certain hour each day over a period of days or weeks:

```
TEST: time_of_day
   STATE: times
      CONDITIONS: TIME 1105
      ACTIONS: ACCUMULATE am10-11 COUNTER datapaks LAST
      CONDITIONS: TIME 1205
      ACTIONS: ACCUMULATE am11-12 COUNTER datapaks LAST
      CONDITIONS: TIME 1305
      ACTIONS: ACCUMULATE pm12-1 COUNTER datapaks LAST
      CONDITIONS: TIME 1405
      ACTIONS: ACCUMULATE pm1-2 COUNTER datapaks LAST
      CONDITIONS: TIME 1505
      ACTIONS: ACCUMULATE pm2-3 COUNTER datapaks LAST
```

The resulting tabular screen could tell you, for example, the average number of data packets traveling over the link between the hours of 11 A.M. and 12 noon for a given Monday through Friday:



MON/LINE BLK= P 09/14/89 16:16
ASCII/8/NONE/BOP

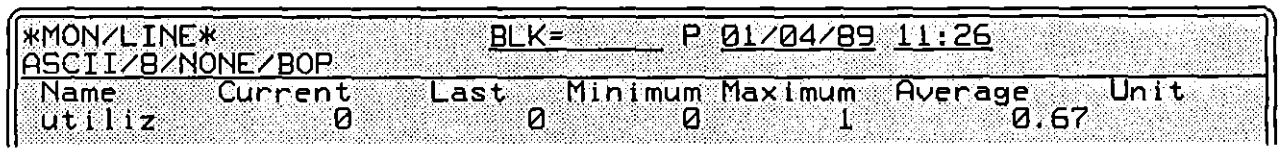
Name	Current	Last	Minimum	Maximum	Average	Unit
datapaks	3491	833	0	54150	5036.12	
am10-11		14107	13103	28242	16780.14	
am11-12		31541	21153	54150	26942.92	
pm12-1		15656	10445	30678	18031.33	
pm1-2		14305	14091	34916	20515.20	
pm2-3		9979	91682	25590	14456.03	

Figure 20-9 Here statistics are accumulated for specific hours of the day over a period of days or weeks.

20.6 The Sampling Action as Divisor

The sampling action can be used to divide the sum of all sampled current values on a counter by another value. To divide X by Y, count the events that add up to X and sample the counter Y times. The quotient or proportion will appear in the **Average** column of the counter.

Suppose, for example, that you want to divide the number of information frames on a link by *all* frames, as an indicator of how efficiently the link is being utilized. Count all Info frames on a counter called **Info**. Sample the **Info** counter whenever *any* frame is seen. Reference the **Info** counter on the Tabular Statistics screen. If 1000 out of 1500 frames are Info frames, your results will look like those in Figure 20-10.



MON/LINE BLK= P 01/04/89 11:26
ASCII/8/NONE/BOP

Name	Current	Last	Minimum	Maximum	Average	Unit
utiliz	0	0	0	1	0.67	

Figure 20-10 The value in the *Average* column for this counter is the sum of all sampled values divided by the number of samples.

21 Graphic Statistics

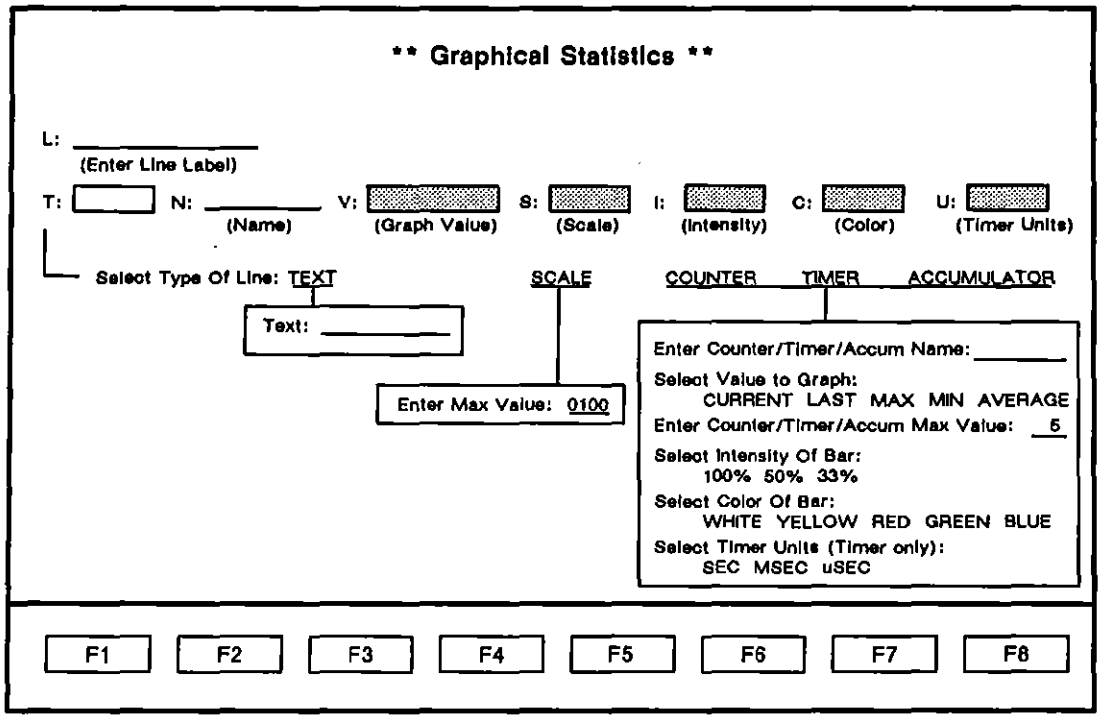


Figure 21-1 Setup selections on the Graphical Statistics screen.

21 Graphic Statistics

The operator of the INTERVIEW will find it easy to design a bar-graph display on the Graphical Statistics screen, with color parameters that can be mapped to a color monitor. Counters, timers, and accumulators that are referenced on the Graphical Statistics screen display their values on this screen as horizontal bars that are drawn in real time and retained in Freeze and Program modes. Variously shaded bars for up to sixteen counters, timers, and accumulators may be displayed.

Any of the sixteen horizontal lines in the graphics display may be reserved for explanatory text or scale numbers instead of a graphic bar. The bars themselves can represent statistics chosen from the entire pool of counter and timer values, grouped and renamed on the graphics display for clarity of overall presentation.

21.1 Enabling the Graphic Display

Both of the statistics screens, tabular and graphics, are enabled at all times during Run mode and can be entered via softkey. Both statistics displays are named on the second bank of softkeys in Run mode. See Figure 21-2. (On the first bank of Run mode softkeys, [F5] is labeled STATS and will call up whichever screen is the current or dormant entry—TABULAR or GRAPHIC—in the **Statistics Type** field on the Display Setup menu.)

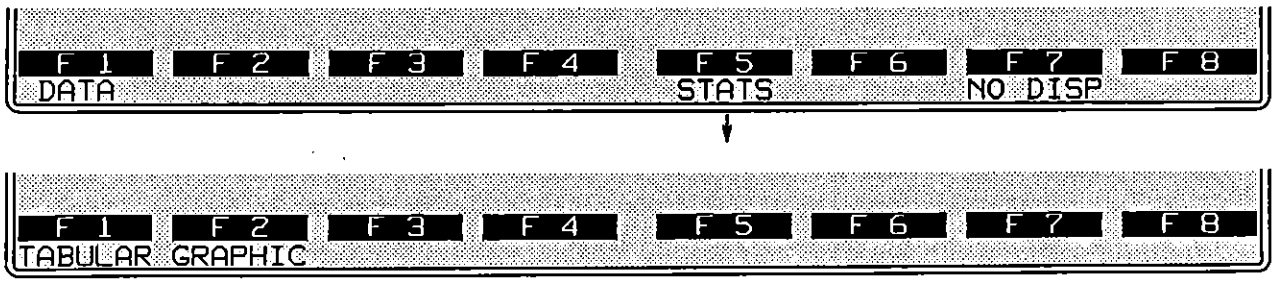


Figure 21-2 Both statistics displays can be entered via the second bank of softkeys during Run mode.

No graphic bar is drawn until a counter or timer has been named on the Graphical Statistics screen in Program mode and then put in motion by the program during Run mode. Examples of triggers that control counters and timers are given in the preceding section, Tabular Statistics. After you have created your counters and timers in the trigger-menu or spreadsheet program, enter the Graphical Statistics screen by pressing [PROGRAM], [F4] for Stats, and [F2] for Graphical Statistics.

21.2 Cursor Movement on Graphical Statistics Menu

There are always two cursors in the Graphical Statistics menu in Program mode. When you enter the screen, one cursor is in the L(abel) field in the menu area at the top of the screen and the lower cursor is on the top line of the sixteen-line display area. See Figure 21-3. The menu-field area always applies to the horizontal display line that has the lower cursor. In Figure 21-3, the display area is blank and the L(abel) field and the other menu fields are in default condition.

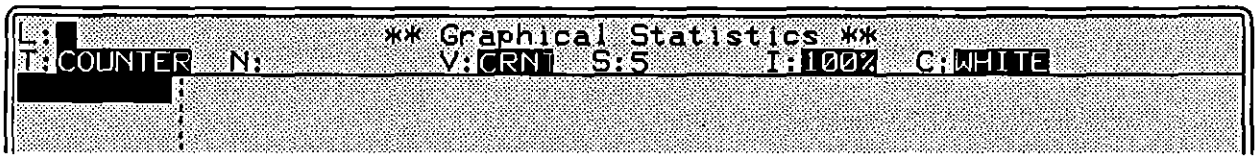


Figure 21-3 Two cursors on default graphics setup screen: the fields in the menu area always pertain to the display line that has the lower cursor.

Press [CONT] or [RETURN] (or → and ←) to move the upper cursor from field to field in the menu area of the screen. Press [ROLL FWD] and [ROLL BACK] to change the selections in rotating-window fields.

Press ↓ and ↑ to move the lower cursor from display line to display line. The menu selections at the top of the screen will change as you cursor down the screen, since they are always keyed to the display line that has the cursor. New menu-area selections and data entries are written to the display line that has the lower cursor as soon as that cursor is moved up or down.

The graphics area of the screen is a scrolling display of variable length that sets a very high limit (48) on the number of bar, scale, and text lines that may be created by the user. To scroll down the directory, position the cursor on the last line of the listings and press ↓. This keystroke will display new bar lines one at a time. Or press [NEXT PAGE] to display sixteen new lines of bar lines. [SHIFT] and ↓ together move the cursor to the end of the listings.

Position the cursor at the top of the listings and press ↑ to expose lines that have scrolled off the screen at the top. Or press [PREV PAGE] to retrieve an entire previous page of listings. [SHIFT]-↑ will always move you to the top of the listings.

21.3 Menu Fields

(A) Label

L is the label field. The horizontal bar lines on the Graphical Statistics screen have labels at the far left. Referring to Section 21.3(B), below, give each bar a name that is compatible with the *text* line at the top of the chart and with the *scaling numbers* above or below the chart.

The label does not have to correspond to the *name* of the counter or timer (or accumulator) on the trigger menus or in the Protocol Spreadsheet program. For example, the label *PHILA* might be used for a counter named badbcc_a, if the counter is tracking errored blocks sent from multidropped device A in Philadelphia.

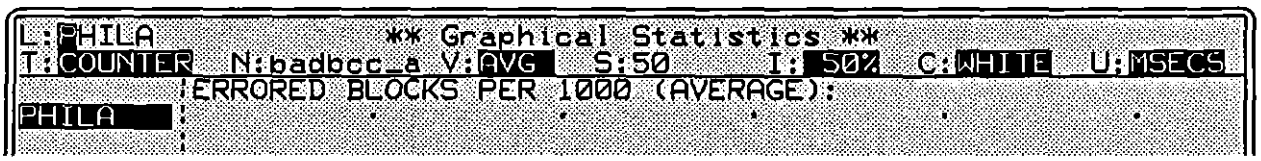


Figure 21-4 The label *PHILA* will appear alongside a horizontal bar representing the average value of the counter named "badbcc_a."

The L field is eight columns wide. Any ASCII entry may be made or the field may be left blank, as in the line of text *ERRORED BLOCKS PER 1000 (AVERAGE):* in Figure 21-4. A label in the L field is written to the display line that has the lower cursor as soon as that cursor is moved up or down.

(B) Type

T designates the type of horizontal line that will be created at the lower cursor. Line types are TEXT, SCALE, COUNTER, TIMER, and ACCUM. Counters, timers, and accumulators will be represented in Run mode by horizontal bars of various shadings that lengthen and shorten as the values for the counters, timers, and accumulators increase and decrease.

T: TEXT devotes a display line to explanatory text. The text line shown in Figure 21-4 was created by the T: TEXT entry in Figure 21-5. The text entry may be 54 characters long. This is the full width of the display area.

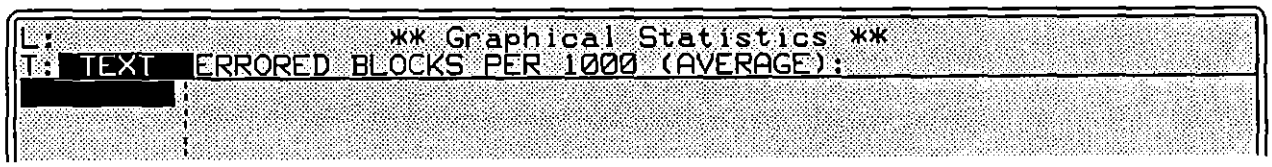


Figure 21-5 Explanatory text will be written to the display line that has the lower cursor as soon as that cursor is moved up or down.

T: SCALE creates a line with five scaling numbers. Enter a number in the S field that represents the highest number of units you will want to display on the graph. The entry may be placed anywhere in the S field. An example of an S entry and the scale line that results is given in Figure 21-6.

The logic will distribute the other four scaling numbers on the scaling line. It will scale the value you enter directly; or else it will raise your value to the next value that fits the scaling algorithm.

To be scaled directly, your S number should be expressible by a single digit of precision in scientific notation. The number 40, for example, will be applied directly to the scaling line, since in its scientific expression—4 x 10^1—4 is a single digit. 45 (4.5 x 10^1) will be raised to 50. Here is the beginning of the series of valid S numbers: 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 2000, etc., up to and including 90,000. (If you enter a value between 90,001 and 99,999, the scaling logic will raise the value to 100,000.)

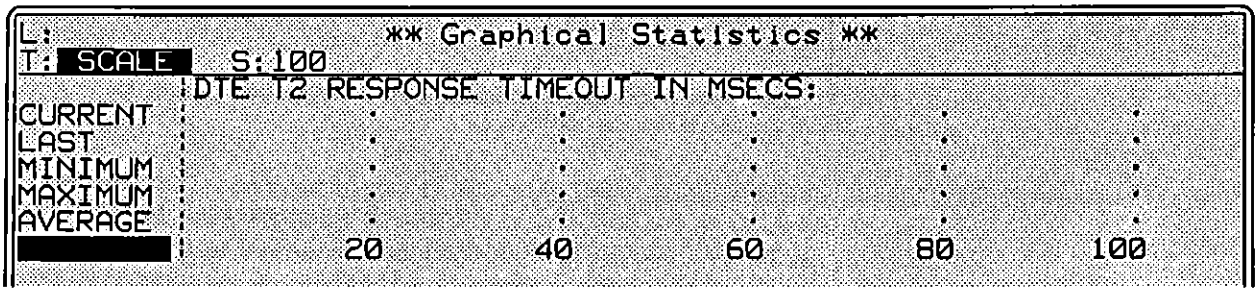


Figure 21-6 The scale line at the bottom of the figure was created by the menu selections at the top.

(C) Name

The N field appears when you have selected COUNTER, TIMER, or ACCUM in the T(ype) field. Enter the name of a counter, timer, or accumulator that you have created in your program. In Figure 21-6, the labels all pertain to values called out under the name t2.

(D) Value

Counters, timers, and accumulators have a set of statistical values associated with them. Any of these values can be represented by a bar on the graphics display. Select one of these values in the V field. Selections under V are AVG, CNNT, LAST, MIN, and MAX. In Figure 21-6, each label references a different value of timer t2.

Note that accumulators do not have a current value: see Section 20.4.

(E) Scale

Each bar line is 54 columns wide. The scale field allows you to pick a number that will display a bar that is *fifty* columns wide. When the statistical value you have selected for graphic display attains this number, its bar will *almost* fill the width of the line.

The S(cale) entry for a counter or timer value has no direct relation to any scale *line* (see Type, above) that may be drawn above or below it. The scale line merely writes numbers on the screen. The S selection for a counter, timer, or

accumulator will scale the actual bar to your estimate of what the maximum value will be. If your estimate is good, the bar will have some magnitude without overflowing the width of the screen.

If your bar is drawn to too small a scale and it overflows, go back to the Graphical Statistics screen in Program mode and increase the **S** value for the counter or timer. The statistical values are kept during Program mode (until you hit [RUN] again), and the bar will be redrawn to the new scale as soon as you move the cursor up or down.

(F) Intensity

Three degrees of intensity are selectable for any horizontal bar. In the **I** field, select 100% for full intensity (white against a dark background), 66% for half intensity (medium gray), and 33% for low intensity (light gray). All three bar intensities are shown in the Run-mode graphics display in Figure 21-7. (Remember that whites and blacks are inverted in the screen illustrations in this manual.)

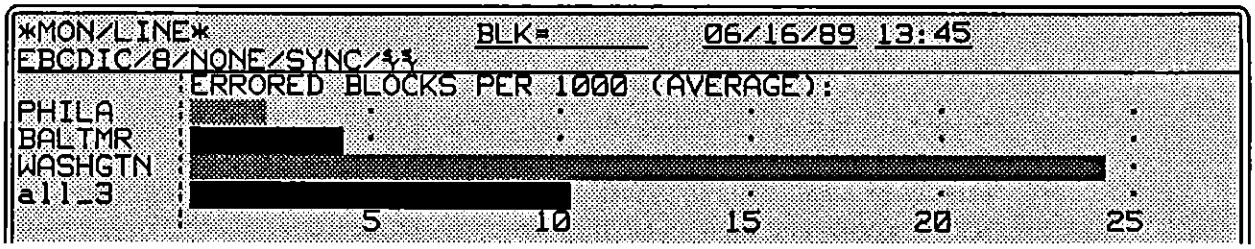
(G) Color

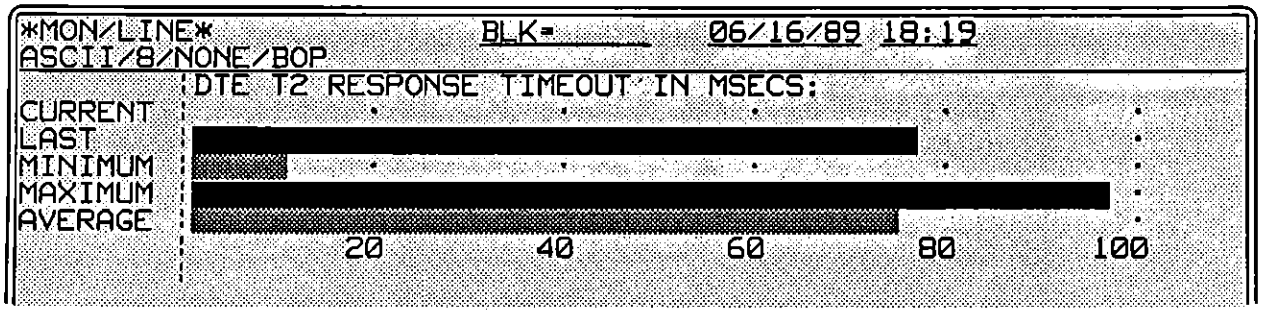
Selections in the **C**(olor) field are WHITE, RED, GREEN, BLUE and YELLOW. The selection in this field has no effect on the screen of the INTERVIEW, but it does affect the signal transmitted on the RGB interface at the rear of the unit. If a color monitor is attached at this interface, horizontal bars on the color-graphics display will be white, red, green, blue or yellow, according to the color selected for each bar (subject to the intensity selected for that bar).

(H) Unit

The **U**(nit) field appears whenever TIMER is the Type selected. Selections in this field are SECS, MSECS, and USECS. The scale numbers on timing graphs relate directly to these units.

Do not select a timer unit that is smaller than the tick interval (**Tick Rate**) selected on the Front-End Buffer Setup menu. This rule of thumb is explained in Section 9.1(C), Time Ticks in Relation to Timer Units.





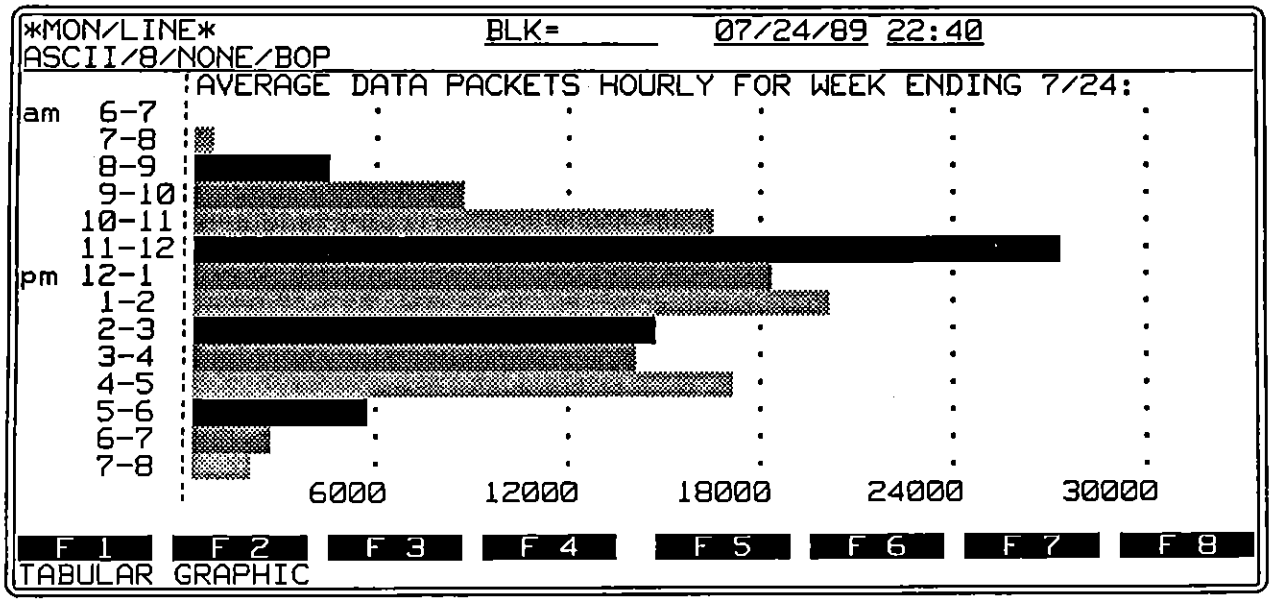


Figure 21-7 Three graphics displays.

22 Three-Tiered Programming

The INTERVIEW 7000 Series is designed to provide programming solutions for problems of varying complexity and for users with different levels of programming skill (see Figure 22-1). The simplest programming tool is the Trigger Setup screen. The setup screen guides the user through a fundamental set of programming selections.

The Protocol Spreadsheet is a more sophisticated and flexible programming method. While based on the same principles as the Trigger Setup screens, the Protocol Spreadsheet provides free-form programming options and a more advanced set of conditions and actions. The spreadsheet allows branching from program routine to program routine as well as simultaneous testing for different sets of conditions. In addition, the structure of the Protocol Spreadsheet is modeled after OSI layered architecture described in Section 23.

A third programming method is present in the INTERVIEW 7000 Series: C programming language, accessible from the spreadsheet, allows the advanced user to write code for test situations outside the scope of standard spreadsheet test selections.

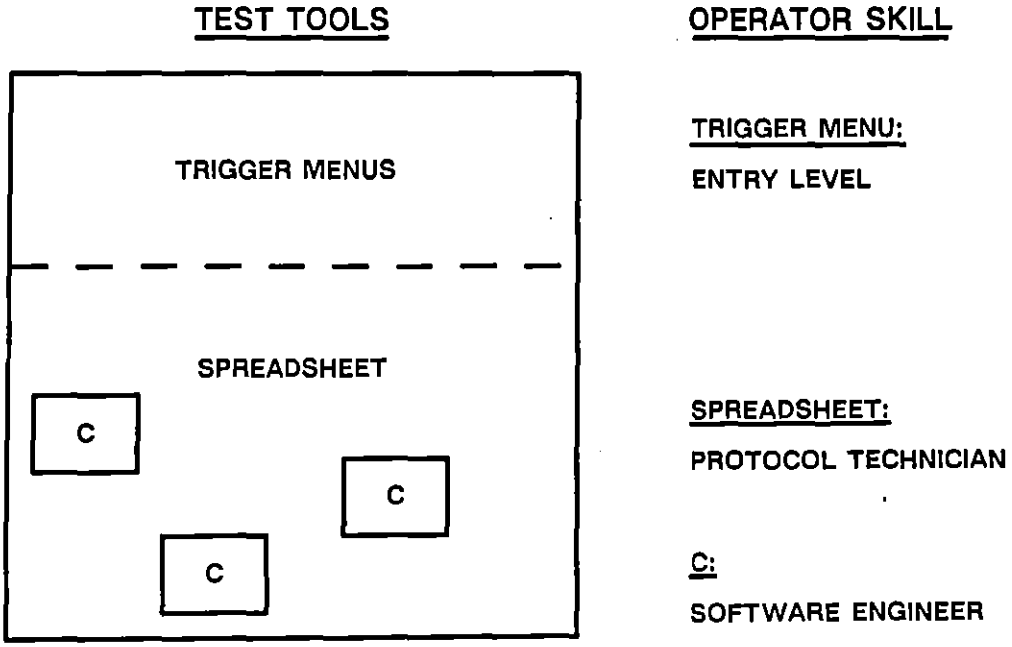


Figure 22-1 There are three separate, *integrated* user-interfaces for programmers of the INTERVIEW 7000 Series.

22.1 Trigger Setup Screens

Triggers are the basic programming tools behind all of the INTERVIEW's test activity. The operator creates each trigger in one of two ways: by using a pre-existent Trigger Setup screen or by keying in trigger conditions and actions on the INTERVIEW's Protocol Spreadsheet.

A trigger is a distinct set of conditions (input) and actions (output). That is, a trigger waits for a specified event or group of events. (These events might include, for example, receipt of a certain data string or change in an internal counter.) When all conditions are met, the trigger responds with a specified action or group of actions. (Trigger actions might include transmission of a data string, sounding of an alarm, or setting of an internal flag.)

There are 16 Trigger Setup screens available in the INTERVIEW 7000 Series. A sample Trigger Setup screen is shown in Figure 22-2. These preconfigured screens provide a simplified approach to programming. Possible conditions are grouped at the top of the menu, and actions potentially taken in response to those conditions are grouped at the bottom of the menu. Trigger conditions contained on the Trigger Setup screens are described in Section 24; Trigger Setup actions are described in Section 25.

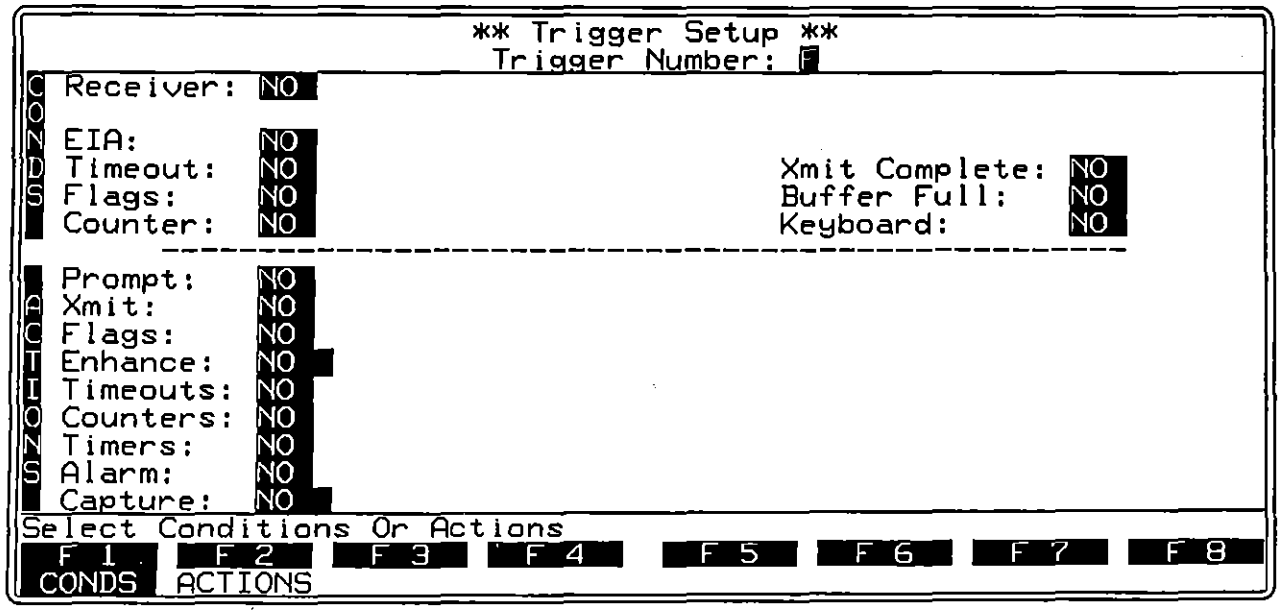


Figure 22-2 There are 16 predefined Trigger Setup screens in the INTERVIEW 7000 Series.

22.2 The Protocol Spreadsheet

The Protocol Spreadsheet, while not a prefabricated menu, contains and extends the set of programming options available on the Trigger Setup screens. As explained in Section 23, the Protocol Spreadsheet program is divided into layers, which are in turn subdivided into smaller components. At each layer, a different protocol is applicable. Depending on the protocol packages which you load, the set of trigger conditions and actions is enlarged to include automatic selections tailored to the protocol and layer you are programming. (Protocol packages are loaded from the Layer Setup screen as described in Section 8.) Figure 22-3 shows the beginning of a spreadsheet test.

Trigger conditions and actions which are always available on the Protocol Spreadsheet are described in Section 30. Conditions and actions available at Layer 1 are discussed in Section 31. Primitives used at different layers are discussed in detail in Section 33.

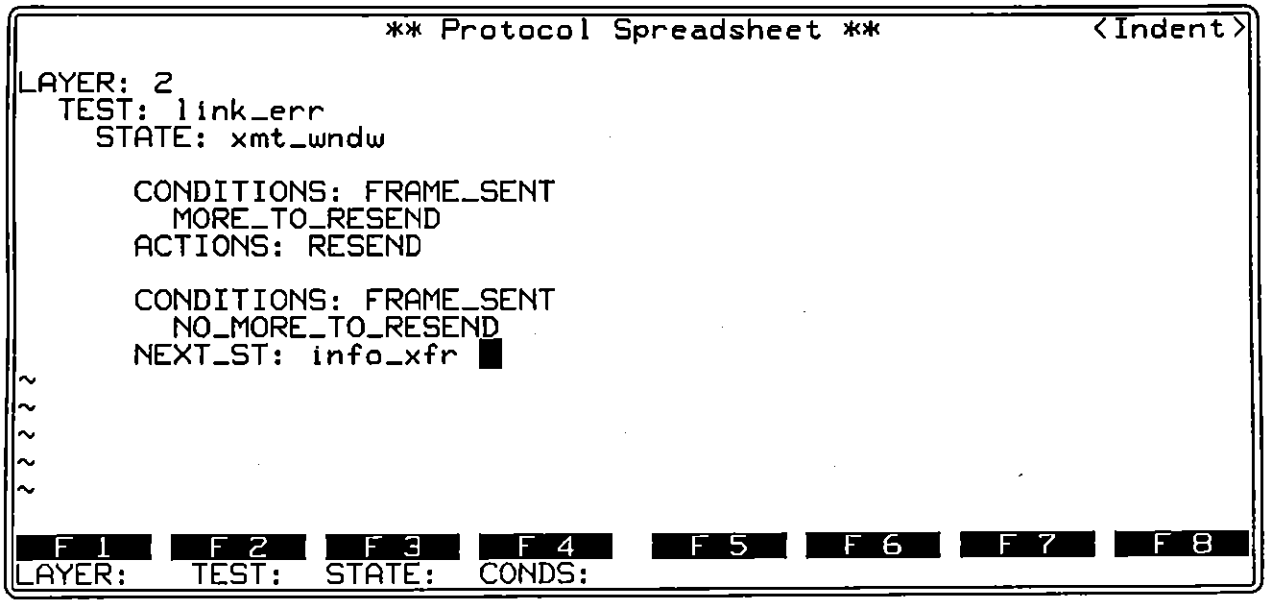


Figure 22-3 The Protocol Spreadsheet conditions and actions shown here are part of the X.25 personality package loaded in at Layer 2.

(A) Automatic Protocol Selections: Personality Packages

Standard data units for a protocol which has been loaded from the Layer Setup screen are available by name so that it is not necessary to enter long hexadecimal strings as conditions or transmit actions. You are also spared the calculation of sequence numbers, poll bits, parity, block checking, and certain other variables that must be included in a received or transmitted string.

Timeouts, window sizes, calling sequences, transmission paths, and other protocol-specific parameters can be modified on a sub-menu which accompanies the personality package.

Protocol-specific conditions and actions are discussed at the end of this manual in a section devoted to the protocol and layer.

(B) Creating and Editing Spreadsheet Programs

Protocol Spreadsheet triggers are created by the operator through the use of indexed softkeys. The entries you make become visible on the screen only after you have made a selection. You also have the option of typing your program from the regular keyboard, as long as keyed entries match the text keywords which are displayed on the screen once you press the function keys. Syntax errors are indicated by a strike-through as you type or make function-key entries.

Press [EDIT] to invoke an alternate bank of spreadsheet keys which provide advanced editing functions. All editing functions are described in Section 29.

22.3 C Programming Language

The INTERVIEW version of C is based on the current ANSI recommendations for C programming language, with extensions to provide multitasking. C is intended as an aid to users who have advanced programming knowledge.

C statements can be incorporated in the spreadsheet as conditions or actions. Figure 22-4 shows C included as a trigger action which displays a prompt at the top of the screen and incorporates a counter value as part of the message. This gives you the ability to extend existing spreadsheet selections or to construct an entire test from scratch using C. C allows you the freedom, for example, to create a customized protocol or program trace display or to manipulate variable data strings anticipated within a user-specific protocol.

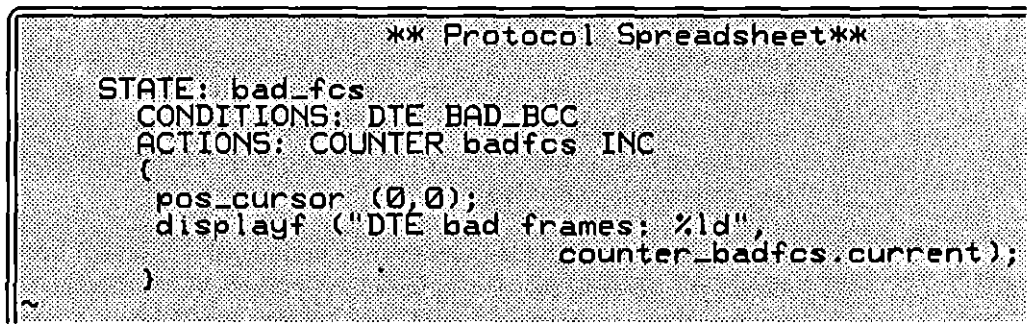


Figure 22-4 The *displayf* function in a C window allows you to write a variable such as a counter to the top of the data screen during Run mode.

22.4 Integrating Programming Methods

The three tiers of programming, Trigger Setup screens, Protocol Spreadsheet, and C programming language, can be integrated to match the needs of each user. The 16 preconfigured Trigger Setup screens can, for example, be employed as a simple line-monitoring test operating at Layer 1. The Protocol Spreadsheet program can later add more complex tests to this, so that several tests are operating simultaneously at a number of layers (see Section 23 for a discussion of layered architecture). Within the Protocol Spreadsheet program, unique test situations or test problems of particular complexity can be resolved by including C programming statements.

(A) Variables Shared Between Trigger Menus and the Spreadsheet

Certain internal program controls are shared between spreadsheet and Trigger Setup screens, in order to allow communication and interdependency between the two types of testing. Internal counters and program timeouts which have the same name can be monitored and controlled both from trigger screens and from the spreadsheet.

There is limited sharing of internal flag bits between Trigger Setup screens and the Protocol Spreadsheet. Trigger Setup flag bits are shared between all trigger screens. They can also be monitored or changed on the spreadsheet, where they are referenced as a flag named **trig_flag**.

(B) Saving and Loading Program Segments

The INTERVIEW's filing system provides a means for storing entire programs or portions of programs for later use by operators of any skill level. On the File Maintenance Screen, you may specify which group of menus you wish to save. For example, if you specify a "Setup," only the five setup menus (Line Setup, Interface Control, BCC Control, Front-End Buffer Setup, and BERT Setup) are saved. This allows you the freedom to create new trigger or spreadsheet tests without continually reconfiguring all menus.

If you save a "Program" on the File Maintenance screen, you are saving the configuration of all menus, including Trigger Setup screens, Layer Setup, and the Protocol Spreadsheet. The one menu that is not saved is the Printer Setup. A program may be a simple test ready to be enlarged, or it may be a highly complex group of tests that can be loaded and run by an operator with little or no programming knowledge.

As a complement to file maintenance options, the Protocol Spreadsheet editor allows you to save only the spreadsheet portion of a program. An advanced programmer can create a set of tests on the spreadsheet or text files containing C code, then use the editor to write his work to a file. Later, a non-programmer may load a setup or a partial program from the File Maintenance screen, call up the spreadsheet screen, and use the editor to read in the advanced programmer's file in order to complete his own program.

> **NOTE:** The File Maintenance Compile command also can be used to save the contents of the Protocol Spreadsheet. The linkable-object file which results contains the compiled object-code version of the program. See Section 27.4.

23 The Layered Program Model

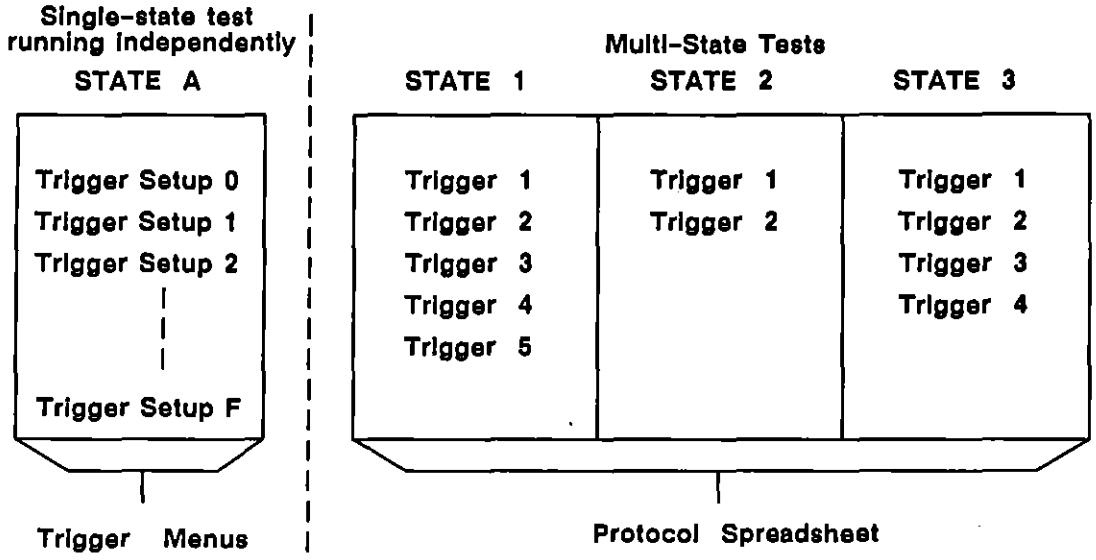


Figure 23-1 Triggers on the Protocol Spreadsheet are grouped into States which can be called in varying order.

23 The Layered Program Model

The trigger, described at the beginning of the previous section, is the fundamental component of all INTERVIEW programs. On the Protocol Spreadsheet, the trigger is grouped to form larger programming blocks, referred to as states. States are grouped to form tests. And tests are divided into layers. The largest component of the INTERVIEW program, the layer, is patterned after the Open Systems Interconnection (OSI) model.

23.1 States

It is a useful programming procedure to group triggers so that some are inactive while others are active. This is possible to a limited extent on Trigger Setup screens, using counters, timeouts, or internal flags to sequence the triggers.

On the Protocol Spreadsheet, triggers can be more easily grouped by separating them into States (Figure 23-1). A state is an independent group of simultaneously active triggers called into play as required by the test. Within a test, only one state is active at one time. That is, all the triggers in this state are awaiting input. All other triggers in the test are dormant.

(When Trigger Setup screens are compiled into the Protocol Spreadsheet, they may be thought of as a single-state test constantly running at the Layer 1 interface.)

When one of its triggers receives the right input, the active state passes control to another state and itself becomes inactive. Transitions between states are always controlled by (spreadsheet) triggers. These trigger-controlled transitions between active and inactive states make branching possible. That is, a more complex set of conditions can be set up inside of a state and certain actions can ensue. Then, at a decision point (for example, "Did the receiving party respond to my transmission?"), the test can choose the correct path from several potential paths. (A: "Yes, he answered, so transmit next message"; or B: "No, he didn't answer, so resend previous message.")

23.2 Tests

Even further flexibility is possible in INTERVIEW programs, because states can be grouped into tests. So, not only does the INTERVIEW move back and forth laterally between the groups of triggers contained in various states, but it can also use different

sets of states to perform several different tests at the same time (Figure 23-2). As an example, two simultaneous tests might be used to check the different set of exchanges expected on either side of a full-duplex line.

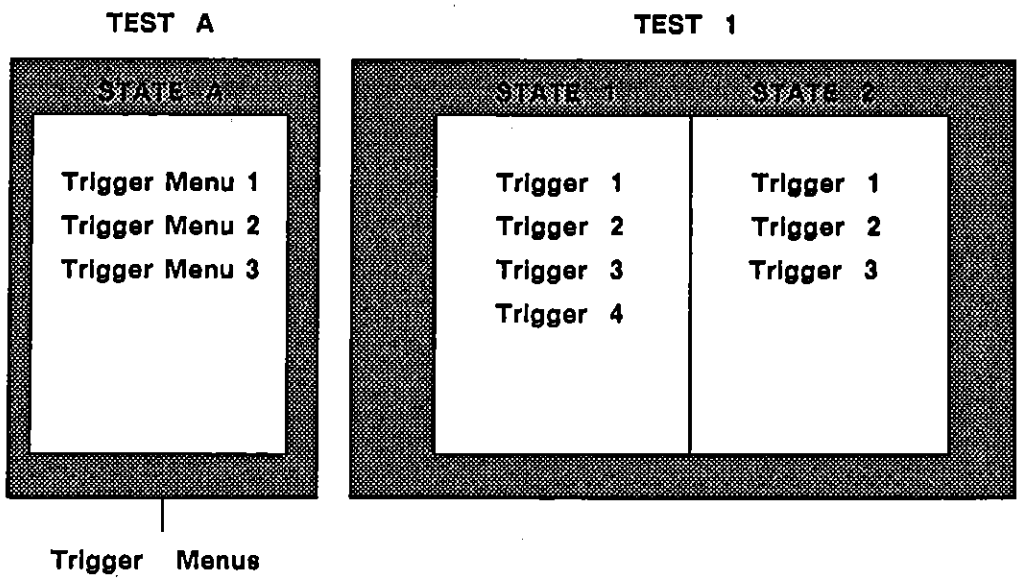


Figure 23-2 Distinct sets of states can be created so that the INTERVIEW can perform several different tests at the same time.

23.3 Layers and the OSI Model

Finally, groups of simultaneous tests can be "layered." In this way, separate tests or groups of tests can be run at a maximum of seven levels simultaneously (Figure 23-3).

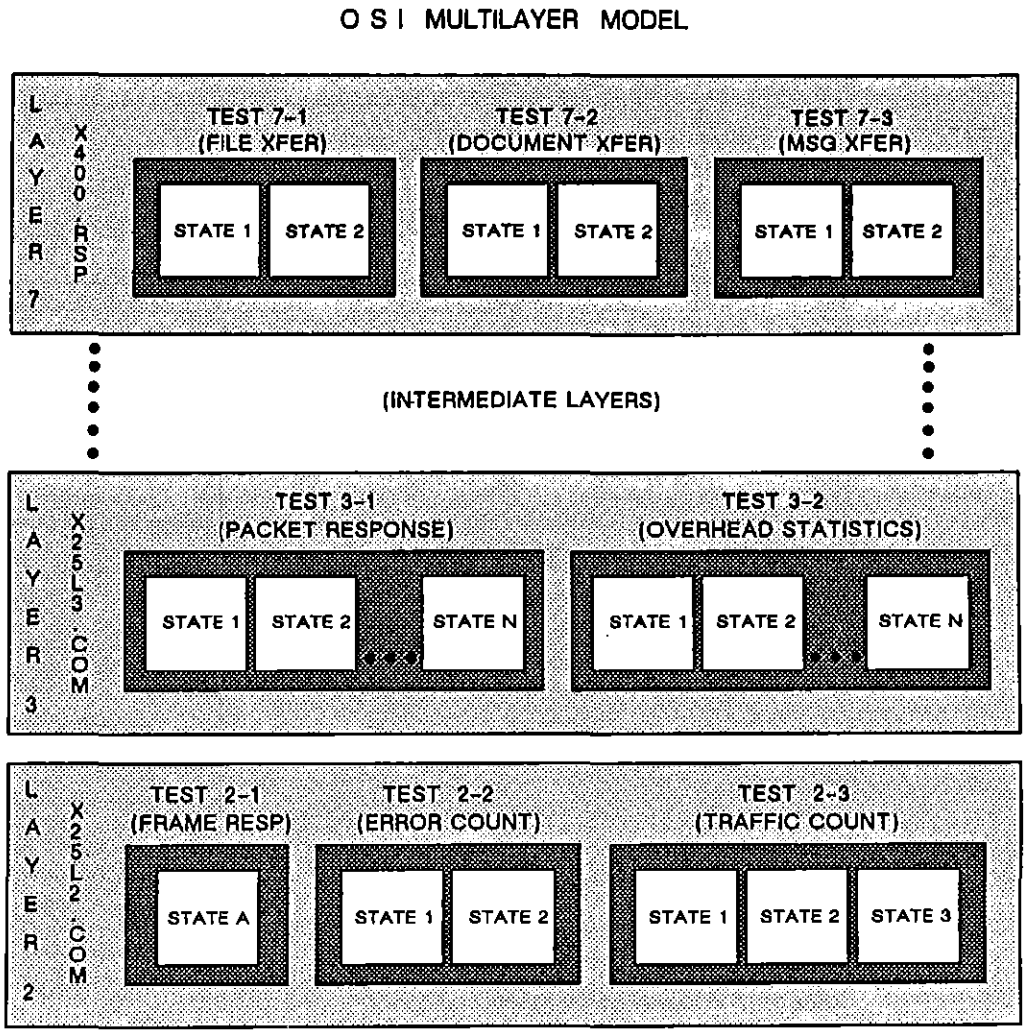


Figure 23-3 Separate tests or groups of tests can be run at a maximum of seven layers simultaneously. This capability parallels the OSI seven-layer model.

This layered structure is specifically designed to handle protocols which conform to the CCITT Open Systems Interconnection (OSI) model. The OSI model is fully described in CCITT Recommendation X.200.

This is a seven-layer model (see Figure 23-4) in which each layer performs a different data communication function. Conceptually, each layer is independent. One layer can be modified without other layers being affected, as long as the modified layer respects prescribed communication with the layer immediately above and the layer immediately below it.

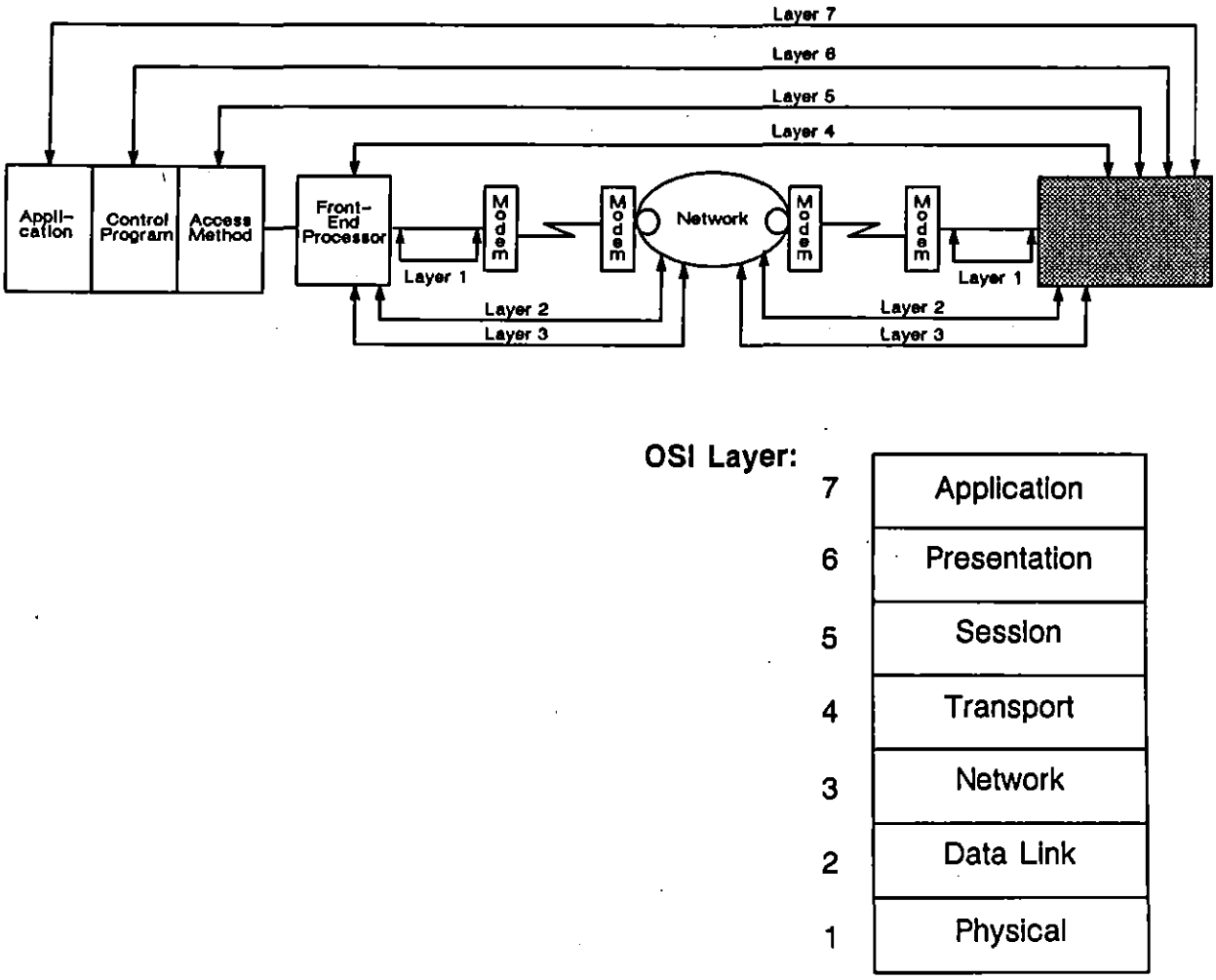


Figure 23-4 The seven OSI layers.

Suppose that the physical link between two nodes in a network were changed from copper wire to optic fiber. In an OSI configuration, only the physical layer (and possibly certain aspects of the data link layer) would be modified. The remainder of the communication process would stay the same.

The separation of programs into discrete layers generally reduces the complexity of test conditions and actions. This simplifies programming for the user. The structure allows you to verify your system—and to debug your own tests—layer-by-layer. For

example, it is not necessary at Layer 3 of a protocol to anticipate variations in line-level or frame-level events. Searches for strings and protocol elements focus only on the portion of a frame which pertains to Layer 3. The validity of the frame which contains the string has already been checked.

23.4 Personality Packages

The layered structure of the OSI model allows you to use different protocols at different layers—again, provided that the rules of OSI interlayer communication are observed.

The INTERVIEW provides layer-specific protocol packages, called personality packages, which you can load from the Layer Setup screen. While certain layer protocols are more commonly used together (SDLC at Layer 2 and SNA at upper layers, for instance), it is possible to mix and match them. You could, with the correct Personality Package, load and run X.25 protocol at Layer 2 and SNA protocol at higher layers.

Personality packages are not in themselves protocol emulations; rather, they are high-level interfaces to routines in the given protocol. A package at Layer 2 X.25, for example, allows the user to *design his own application* by simple softkey-entry of a routine such as

```
CONDITIONS: RCV DISC
ACTIONS: SEND SABM
```

or

```
CONDITIONS: T1_EXPIRED
ACTIONS: RESEND
```

Personality packages are selected and loaded from the Layer Setup screen (shown in Figure 23-5; see Section 8 for a description of this screen). The contents of each personality package are described in a section dedicated to the package (refer to the Table of Contents, Sections 35 and following).

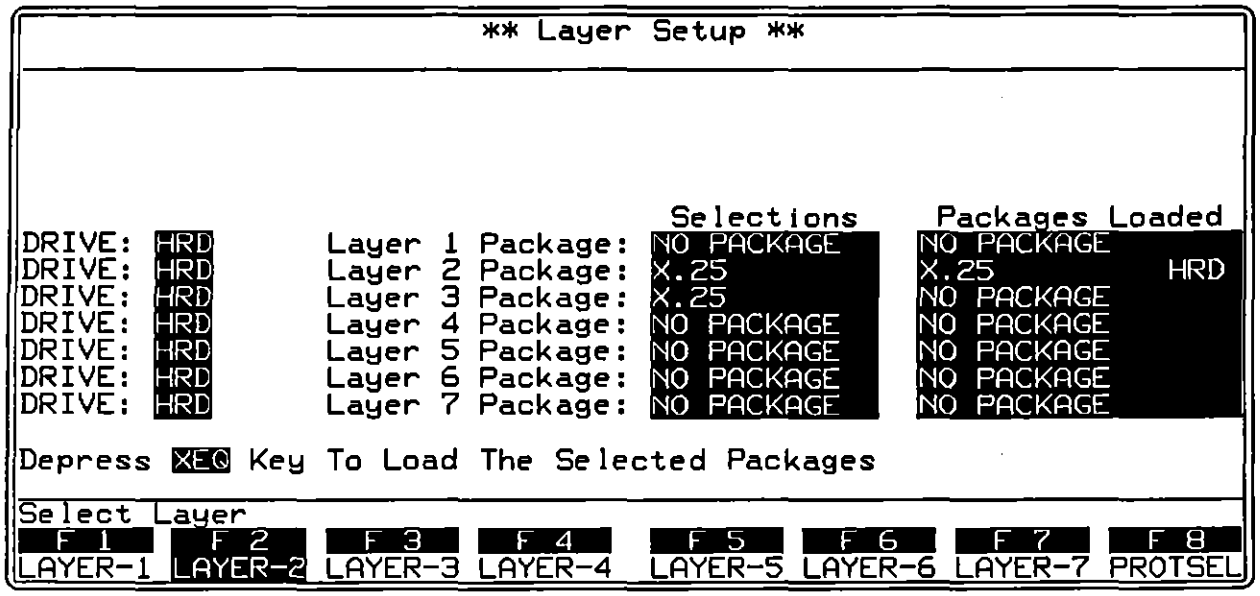


Figure 23-5 Personality packages, which provide the protocol elements in INTERVIEW programming, are loaded from the Layer Setup screen.

23.5 Primitives

The OSI Layers use limited-range messages called *primitives* to communicate with each other. Primitives are defined by the OSI model and are not linked to any one protocol. No matter what personality package is loaded, these generic primitives are available at each layer. This gives you the freedom to create or modify a protocol. Primitives available on the Protocol Spreadsheet are discussed in Section 33.

23.6 Constants

To represent a frequently used test value, you may define a constant once in your program and reference the constant elsewhere in the program as needed. Replacing the test value with a new value then becomes easy, since you need only change the constant definition one time.

Constants, which may be used to represent any textual string, can be defined at several levels in the spreadsheet program. The function key labeled CONSTS: is only present when it is legal to define constants.

Depending on where they are defined, constants vary in scope. You have the option of creating constants which can be used globally, throughout a layer, or throughout a test. Refer to Section 28 for a full description of constants.

24 Trigger Conditions

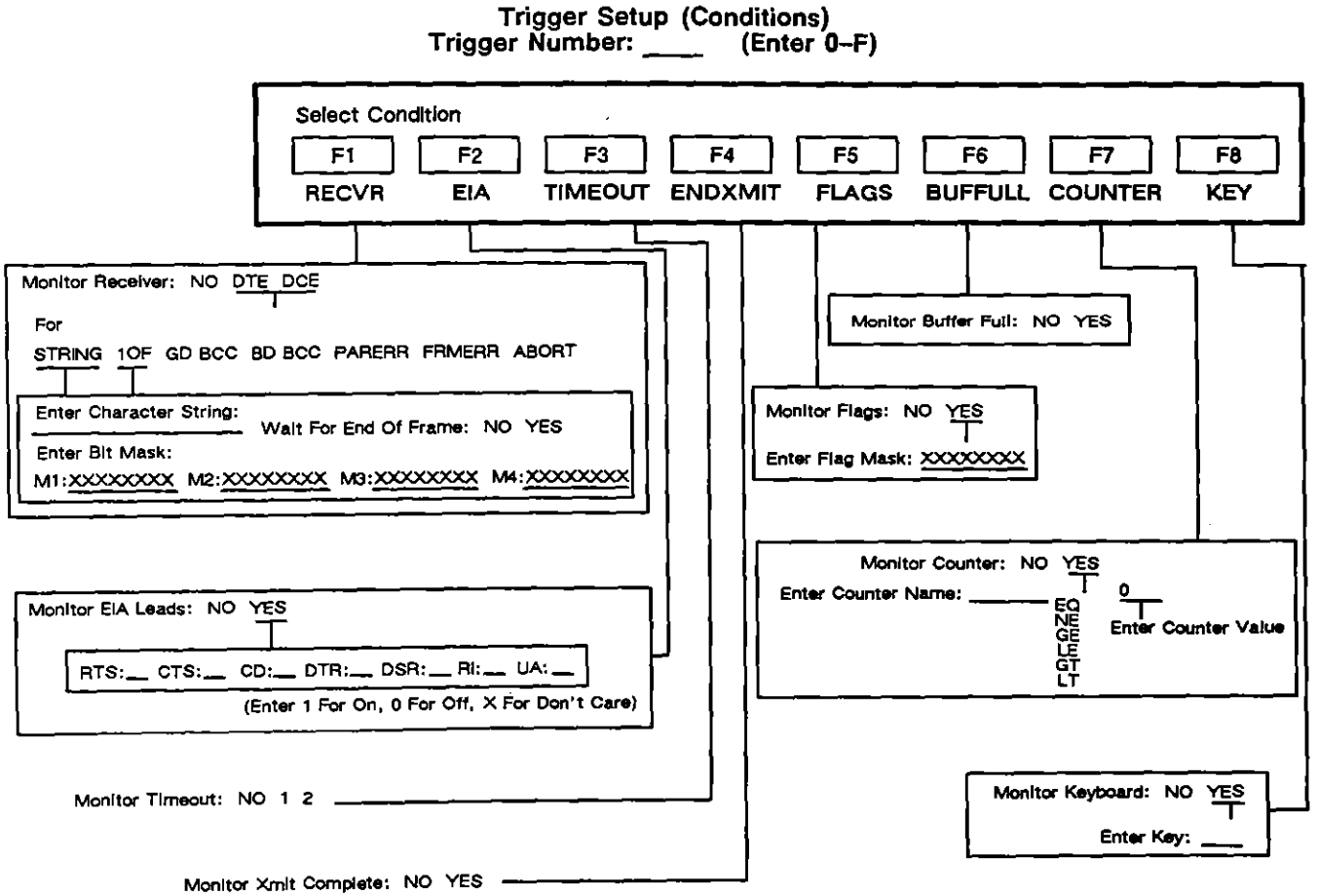


Figure 24-1 Conditions on Trigger Setup menu.

24 Trigger Conditions

Triggers can be thought of as "IF, THEN" statements, represented on the screen as "Conditions" (IF...) and "Actions" (THEN...). This section pertains to Trigger Conditions available on the preconfigured Trigger Setup screens, of which there are 16 in the INTERVIEW 7000 Series. All possible conditions available on the Trigger Setup screen are shown in Figure 24-1.

Triggers are numbered 0 through F. To access a particular trigger screen, press the TRIGS function key on the Main Program Menu. This calls up the Trigger Summary screen. Enter the Trigger Number desired (it will appear highlighted at the top of the screen) to see that trigger screen.

Each trigger screen is divided in half, with Conditions at the top of the screen and Actions at the bottom of the screen. A default Trigger Setup screen is shown in Figure 24-2.

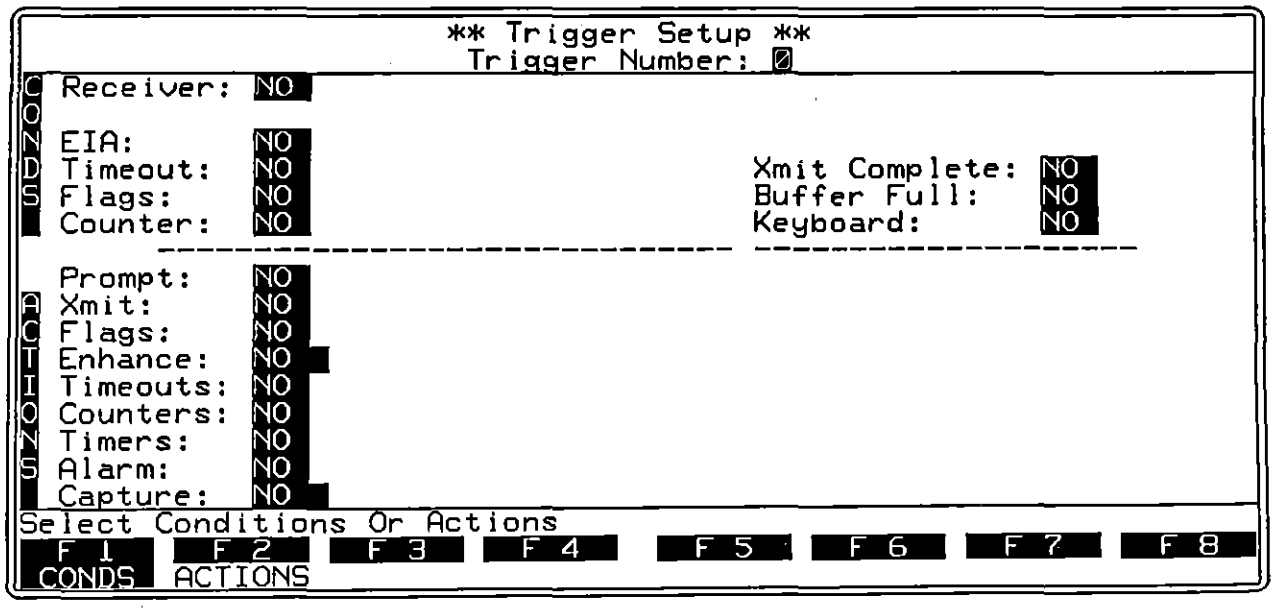


Figure 24-2 Default trigger menu.

24.1 Active Triggers

Only active triggers are tested. Trigger Setup screens are always active. (This is not true of triggers on the Protocol Spreadsheet, where triggers are configured in alternately active states as a matter of program design.)

24.2 Combining Conditions on the Same Trigger Setup Screen

A trigger is true and can take action only at the instant that *all* trigger conditions are met.

(A) Static vs. Instantaneous Conditions

Internal flag, Counter, EIA lead, and Buffer Full conditions differ from other conditions on the Trigger Setup screen. When it is used in a trigger *by itself*, each of these conditions, like other independent conditions, initiates its trigger actions only at the instant that it transitions to true. In addition, these four conditions can retain a *status* of true for a long period of time.

The *static* value of these four conditions is tested for true or false when they are combined in the same trigger with another condition.

All other trigger conditions are true only at the instant that they happen. We will refer to them as "instantaneous" or "transitional" conditions.

> NOTE: It is important to remember that even a "static" condition is "transitional" when it is used *alone* in a trigger. An EIA condition, for example, used by itself in a trigger cannot come true without a transition.
>
> An exception to this rule is when the test enters Run mode. At that moment, static conditions—flags, counters, EIA leads, and buffer full—used alone on a Trigger Setup menu (*not* on the Protocol Spreadsheet) are tested once for a status of true. Then they revert to being true only upon transitions.

(B) Rules for Combining Conditions

These "static" conditions can be combined with other trigger conditions on the same Trigger Setup screen to form compound "IF" statements. Here are some rules to remember in combining trigger conditions:

1. When "static" conditions appear on the same trigger menu with an "instantaneous" condition, the trigger is keyed to the instantaneous condition. All static conditions must be true when the instantaneous condition transitions to true. On that transition, trigger actions are taken.

   Suppose, for example, that a trigger is looking for a Bad BCC and a counter value = 20. The counter value must first increment to 20, then the Bad

BCC must be detected. As soon as the Bad BCC is detected, the trigger becomes true and takes action.

2. When static conditions are combined, both (or all) are transitional. When one of them transitions true, the other(s) becomes a static condition and is checked for a status of true. The user does not have to try to anticipate which of two (or more) conditions will transition first.

NOTE: On the Protocol Spreadsheet, static conditions are prioritized in the order that the user lists them: only the first is transitional. The Protocol Spreadsheet therefore requires you to define which static condition will be the controlling, transitional condition. See Section 30.2.

24.3 Receiver

This condition monitors the data lead specified (DCE or DTE) for designated data. When **Receiver:** DCE or DTE is selected, several options become available: String, 1of, Good BCC, Bad BCC, Parity Error, Frame Error, and Abort.

(A) DTE or DCE

In using the Receive condition, you must specify which side of the line you wish to monitor. Select DTE to denote the TD lead. Select DCE to denote the RD lead.

(B) String

This selection allows you to enter a string of up to 16 characters in the field provided. The entire, exact sequence of characters entered must be received for the condition to be true.

(C) "One of"

When 1 OF is selected, the trigger looks for any one of the characters entered in the next field. Up to 16 individual characters can be entered.

(D) Good or Bad BCC

GD BCC (Good Block Check Calculation) and BD BCC (Bad Block Check Calculation) cannot be used as conditions unless **Rcv Blk Chk** is on in the unit (selectable on the Line Setup screen; see Section 5). Select GD BCC or BD BCC when you want the trigger to take action on receipt of the Block Check Calculation (referred to as FCS, or Frame Check Sequence in Bit-Oriented Protocols).

(E) Parity Error

PARERR looks for a parity error in relation to the **Parity** selection made on the Line Setup screen.

(F) Framing Error

FRMERR applies to start-stop formats (ASYNC and ISOC) and locates framing errors, based on the stop bits anticipated. Both **Format** and **Stop Bits** are selected on the Line Setup screen.

(G) Abort

This selection applies to all Bit-Oriented Protocols. When ABORT is selected, the INTERVIEW triggers off of the seven consecutive 1 bits which constitute an Abort. BOP should be used as **Format** on the Line Setup Menu when ABORT is selected.

> NOTE: The trigger condition will not respond to idle-time aborts, unless **Display Abort:** YES has been selected on the Line Setup screen.

(H) Character Entry Field

This field appears only if STRING or 1 OF is selected. It is the data-entry line for a sequential character string, if STRING has been selected; or a non-sequential character list, if 1 OF has been selected. Up to 16 characters may be entered, in either case.

1. *String entry*. The 16 characters allowed in the string may include any of the following in any order or number:

   All upper and lower-case ASCII characters available on the keyboard.

   All control character mnemonics on the keyboard.

   Two-digit hexadecimal entries. These are entered by first turning the HEX key on, then using alphanumeric keys 0 through 9 and A through F. Two alphanumeric key strokes are required for each hex character. A hex character is represented on the screen as a pair of small characters, the first ascending and the second descending. Compare hex characters to regular alphanumeric characters in Figure 24-3.

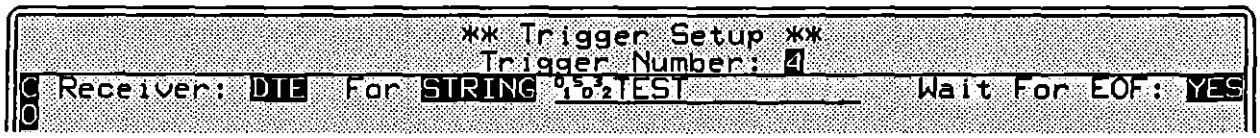


Figure 24-3 Both alphanumeric and hexadecimal characters can be entered as part of a condition search string.

Characters entered in hexadecimal are not translated, and parity is not calculated for them; therefore, you must include the parity bit, whether good or bad, in your entry.

2. *Flags.* You must press [FLAG] to enter the T_E Flag byte used in Bit-Oriented Protocols. The INTERVIEW's logic will not read a hexadecimal entry made with [HEX] as a flag.

3. *Sync.* Press [SHIFT]-[FLAG] to enter the sync symbol. The character [5] is displayed on the Trigger screen.

4. *Not equal (≠) entry.* When a character key is preceded by [NOT (NOUN)], all characters not equal to that character will satisfy that position in the string. These characters are represented in the data entry field with a horizontal bar through them.

5. *Don't care.* [DON'T CARE] permits *any* character received in that position to satisfy the condition.

6. *Bit masks.* Four bit masks can be positioned anywhere in the data-entry string. To enter a bit mask, use [BIT MASK] at the desired location in the string. Each time you press [BIT MASK], a new mask field appears below the string-entry field. To move the cursor to the next position in the string-entry field, press [DONE]. The mask fields are numbered M1 through M4, to denote the order in which they appear in the string (see Figure 24-4).

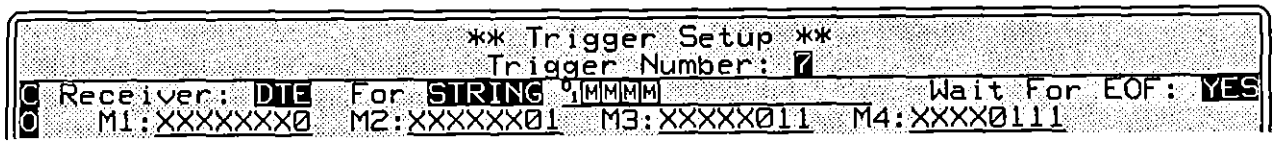


Figure 24-4 Four bit masks can be used as part of the search string.

If you have used Bit Masks 1 and 2, for example, at positions 5 and 8 of the string (as in Figure 24-5), you may decide to change the character at position 1 to a bit mask. Return the cursor to position 1 of the string and

press [M]. The character at position 1 will be overwritten with [M], the two prior bit masks will be renumbered to 2 and 3, and their menu location shifted to make room for the new Bit Mask 1. The cursor will be on the left-most bit of the new mask. (Compare Figure 24-5 and Figure 24-6.)

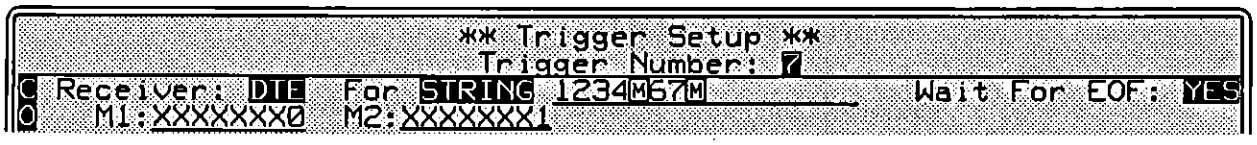


Figure 24-5 Bit masks M1 and M2 are entered at positions 5 and 8 of this string.

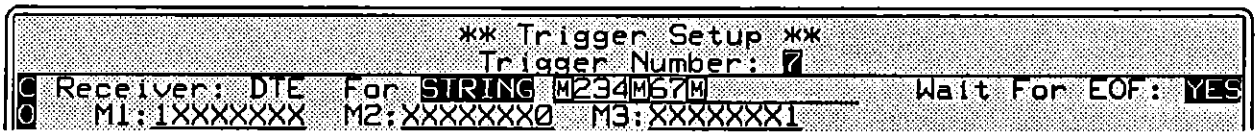


Figure 24-6 A third bit mask has been entered at position 1, and the old M1 and M2 have shifted automatically to M2 and M3.

NOTE: When a Bit-Oriented Protocol is being tested, the INTERVIEW ignores inserted zero bits. You can specify the search string on the Trigger Setup screen without considering zero bit insertion.

7. *Double parens.* A global constant declared on the Protocol Spreadsheet can be entered as part of a Receiver String. Enter the name of the constant exactly as it appears on the spreadsheet and enclose it in double parentheses, as follows: ((drop)). The double parentheses are special characters created by pressing [CTRL]-[9] for ((, and [CTRL]-[0] for)). When constants are used, the receive string cannot be longer than 32 characters *after all constants are expanded.*

(l) "1 OF" Character Entry

Up to 16 characters may be entered. The same types of characters valid in **String** entry are valid **1 OF** characters. The trigger will take action upon receipt of the first character that matches any one of the characters anywhere in the list. If [DON'T CARE] is used, only this character should appear in the **1 OF** entry field (since any character is a match for Don't Care). If *one or more* characters in the field are entered as [NOT EQUAL], only those characters not in the field will satisfy the condition. Thus, pqz in the **1 OF** field means the same thing as pqz: all characters other than p, q, and z will satisfy the condition.

(J) Wait For End of Frame

This is a subfield which appears when STRING or 1 OF is selected. The default selection is NO. When Wait For EOF: YES is selected, the trigger first tests for the data specified on the trigger. Then it evaluates the block check at the end of the frame. The trigger will not take action when a Bad Block Check or an Abort is detected on a received frame.

24.4 EIA

Select YES in the EIA field if you want a trigger to monitor status of up to seven RS-232/V.24 leads (see Figure 24-7).

> NOTE: For line data, EIA lead-status is not detected if control leads are not buffered in the Front-End Buffer. See Section 9. For recorded data, EIA lead-status is not detected if control leads were not buffered in the FEB at the time of recording.

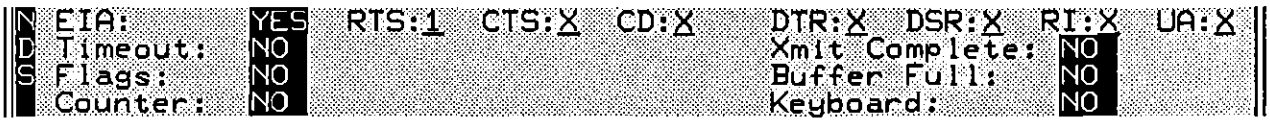


Figure 24-7 Each trigger can monitor the status of seven EIA leads.

Enter a 1 in the box under a lead to indicate ON; a 0 for OFF. No entry (X) is read as Don't Care. Entry fields are provided for six leads: RTS (Pin 4), CTS (Pin 5), DSR (Pin 6), DTR (Pin 20), CD (Pin 8), and RI (Pin 22). You may monitor a seventh RS-232/V.24 lead by strapping the desired lead to UA on the Test Interface Module (see Section 12 for instructions).

In using the EIA condition, you should keep the following points in mind.

- If only EIA conditions are selected, the trigger will wait for all EIA conditions to be satisfied. It will become true on the last transition necessary to satisfy these conditions.
- The EIA condition is a static condition. The rules for combining static conditions with other conditions are explained in Section 24.2.

24.5 Timeout

Two timeout timers can be monitored from the Trigger Setup screen.

The decrementing timeout timer is set for a specific time as part of a trigger action. The trigger which sets a monitored timeout may be any of the Trigger Setup screens or any trigger on the Protocol Spreadsheet.

The default **Timeout** selection is NO. Select 1 to monitor Timeout timer 1; 2 to monitor Timeout timer 2. The condition is satisfied at the instant of the timeout.

24.6 Transmission Complete

When **Xmit Complete:** YES is selected, the INTERVIEW tests for the end of its own transmissions.

> NOTE: The INTERVIEW transmits only when operating in Emulate DTE or DCE mode (selectable on the Line Setup screen; see Section 5).

The **Xmit Complete** condition is frequently used with an internal flag or counter condition to control when or how many times it will be tested.

24.7 Internal Flag Bits

There are eight internal flag bits reserved for the INTERVIEW's Trigger Setup screens. The purpose of the flag bits is to provide a simple way to interconnect several triggers in order to make one trigger dependent on another or to set up triggers in sequence. Each bit is a simple switch that one trigger may set as an action and all triggers can test later.

Internal flags may all be monitored and set by any individual trigger. (The same flag bit can be set and sensed by a single trigger.) Since each flag bit can be set and monitored separately, it can also be shared among the Trigger Setup screens.

The internal flag condition is a static condition and can be used in combination with other trigger conditions as explained in Section 24.2.

To test internal flags, select **Flags:** YES. In the flag mask which then appears, enter a 1 to test for a flag bit turned ON, a 0 to test for a flag bit turned OFF. Enter X in the appropriate position of the mask (or press [SKIP]) if you do not wish to test a particular bit. See the example in Figure 24-8.

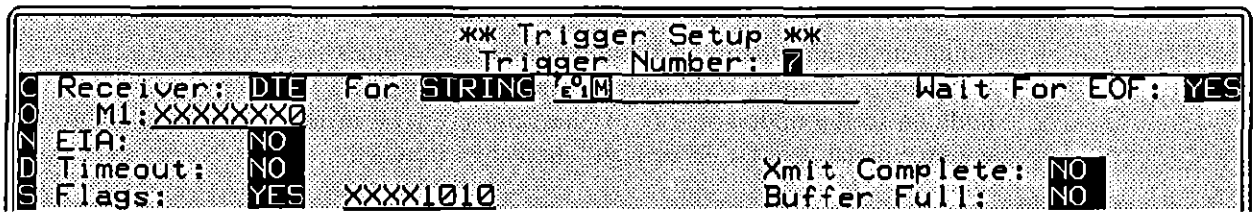


Figure 24-8 The internal flag condition is frequently used in combination with other trigger conditions.

NOTES:

All flags are set to 0 as the INTERVIEW enters Run mode.

The eight flag bits on the Trigger Setup screens are the low-order bits of a flag mask that can be accessed on the Protocol Spreadsheet by the name **trig_flag**. See Section 30.3(G).

24.8 Buffer Full

This condition checks the screen's 64 Kbyte character buffer and becomes true as soon as the buffer is full. The condition then remains true throughout the program. Buffer Full is a static condition which may be used in conjunction with other conditions. The rules for combining trigger conditions are outlined in Section 24.2.

24.9 Counter

Each Trigger Setup screen can monitor a counter with a range of 0 to 999,999. The counter which is monitored may be named and controlled either on a Trigger Setup screen or on the Protocol Spreadsheet.

The default selection is NO. When **Counter:** YES is selected, new menu fields appear (see Figure 24-9).

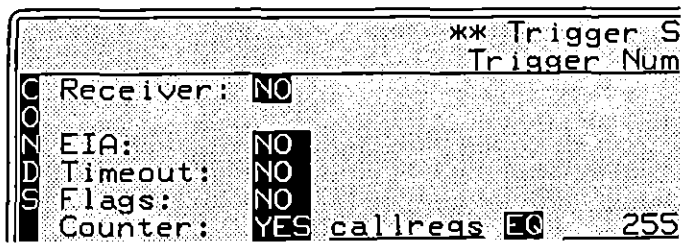


Figure 24-9 Any counter named on a trigger can be monitored as a Trigger Setup screen condition.

(A) Counter Name

Enter the counter name in the field provided. Names must start with a letter. Any of the 52 alpha characters (upper and lower) and the 10 numerals in addition to the underscore (_) character are legal in all other positions. A counter name may be up to eight characters in length.

(B) Relational Operator

Make the appropriate selection to specify when the **Counter** condition will be true. The counter may be tested for a value equal to (EQ), not equal to (NE), greater than or equal to (GE), less than or equal to (LE), strictly greater than (GT), or strictly less than (LT) the entered value on the trigger screen.

(C) Counter Value

Enter the counter value as a whole decimal number in the field provided.

24.10 Keyboard

Select **Keyboard:** YES to display a one character entry field. Then press the key which you wish to use as the condition. In Run mode when that key is pressed, the condition will be true and (if this is the only condition) will initiate a trigger action, such as a transmission. Any key *or key-combination* that produces a character listed in the ASCII chart in Appendix D1 is valid input in this field.

25 Trigger Actions

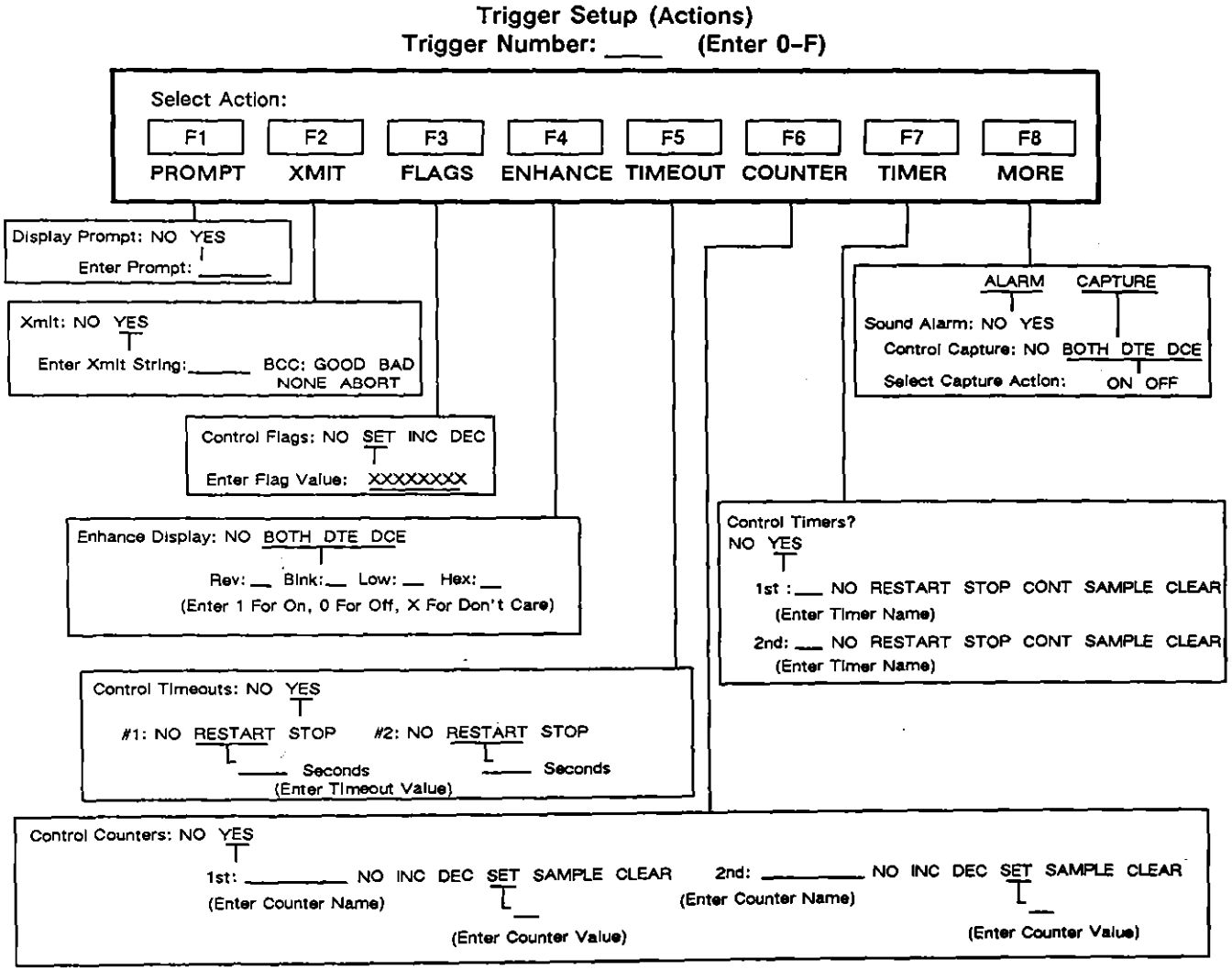


Figure 25-1 Actions on Trigger Setup menu.

25 Trigger Actions

Figure 25-1 shows all actions available on the Trigger Setup screen. Figure 25-2 shows a default trigger menu. The top half of the menu contains available trigger conditions, discussed in Section 24. A more complete set of trigger conditions and actions is available on the Protocol Spreadsheet (see Sections 30 and 31, as well as individual sections on the Protocol Packages).

All conditions selected on the top half of the Trigger Setup menu must be satisfied for the trigger to be true. Only then will the actions on the lower half of the menu be taken.

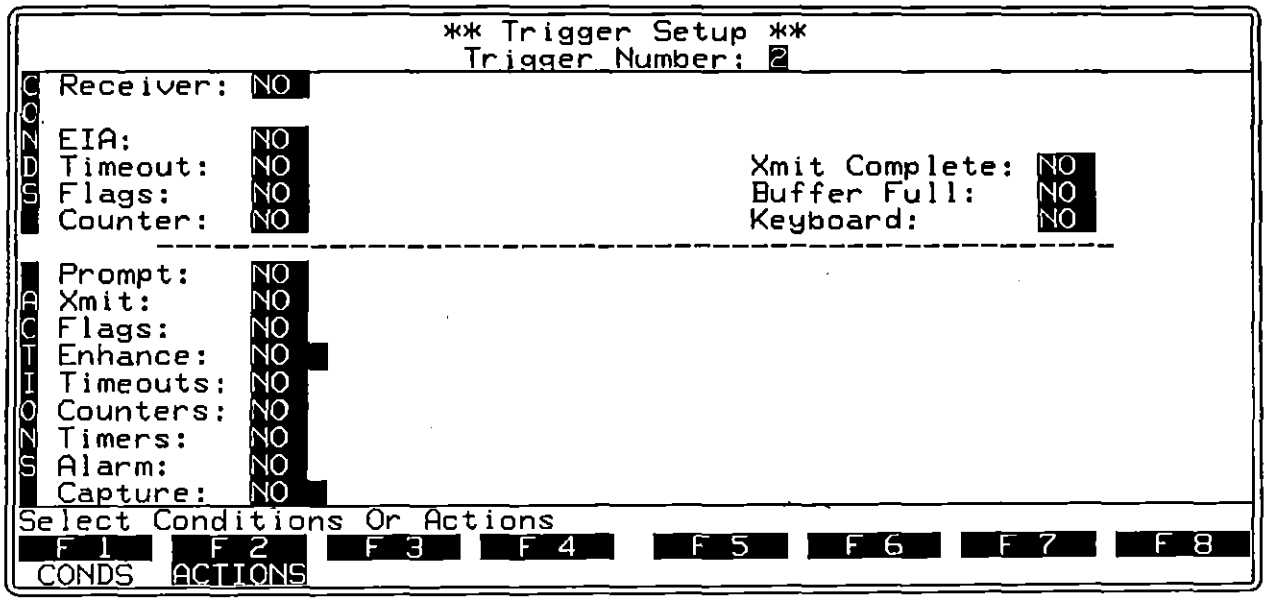


Figure 25-2 Default trigger menu.

25.1 Displaying a Prompt

Select **Prompt:** YES to display a data entry field (see Figure 25-3). You may enter a message of up to 47 characters here. Any ASCII characters are legal entries. When the trigger is true, the Prompt will be displayed on line 2 of the screen. (The prompt is NOT transmitted.) It will stay on the screen until it is replaced by another prompt, or until you clear it by changing the display mode or by pressing the RUN key.

NOTE: A new prompt does not reinitialize the prompt line on the screen. Instead it overwrites the old prompt to the extent of the new one. If the prompt "LINK-UP" is overwritten by the prompt "CALL," the result will be "CALL-UP."

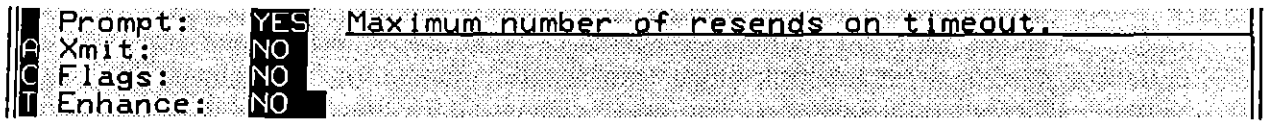


Figure 25-3 Messages entered in the *Prompt* field will appear on the second line of the screen in Run mode.

Prompts can be used to call your attention to certain occurrences, or to help you follow the course of a test. Prompts are also useful in program development because you can use a prompt to tell you when a trigger is true. Any alphanumeric or control character from the keyboard, including spaces, may be part of a prompt.

25.2 Transmitting

INTERVIEW transmissions—with the exception of BERT transmissions, described in Section 11—are always under trigger control, either from the Trigger Setup screen or from the Protocol Spreadsheet. (You may, however, control trigger operation manually by selecting **Keyboard** trigger conditions; see Section 24.)

Select YES to display a data entry field and a rotating window for **BCC** (see Figure 25-4). When the trigger is true, the INTERVIEW will transmit any message you enter here (up to 37 characters), ending with the block-check selection you make in the **BCC** field. ASCII characters, hexadecimal entries, and control characters are valid in this field.

Any global constant declared on the Protocol Spreadsheet may be referenced as part of the **Xmit** string on a Trigger Setup screen. Enter the name exactly as it was declared on the Protocol Spreadsheet, and enclose it in double parentheses—for example, ((drop)). The double parentheses are special characters created by pressing CTRL-9 for ((, and CTRL-0 for)) .

Since the standard fox message, containing the set of upper-case alpha characters and the ten numerals, is pre-defined, you may reference it as a constant on a Trigger Setup screen. It need not be declared on the Protocol Spreadsheet. Enter it as follows: ((FOX)) .

Figure 25-4 Transmitted messages may be terminated with good or bad BCC's or an Abort.

(A) BCC

You have the option of following each text block transmitted from the Trigger Setup screen with a block-check calculation. Block checks are calculated according to your selections on the BCC Setup screen (see Section 10).

On the INTERVIEW screen, the final byte of the calculation appears as a highlighted overlay (G, H, or ▪), as long as you have selected **Rcv Blk Chk:** ON on the Line Setup screen (see Section 5). This selection is only available for synchronous and asynchronous formats. When **Rcv Blk Chk** is OFF for these two formats, block-check characters appear as they are actually transmitted.

The block-check symbol will always be displayed for Bit-Oriented Protocols.

In the rotating **BCC** window, you may select GOOD, BAD, NONE, or ABORT. The default selection is GOOD.

1. *Good Block Check.* Select GOOD to terminate your text blocks with a correct block check. (Remember that not all transmissions are text blocks: a bisync poll will not receive a block check even if GOOD or BAD is selected.)

2. *Bad Block Check.* Select BAD to end your transmission with an erroneous block check. For Bit-Oriented Protocols, the bad BCC is CRC-16 instead of CCITT; for other formats, the bad BCC is an inverted good BCC.

3. *None.* When NONE is selected, no block check is sent at the end of the transmission. (For BOP transmissions, NONE has the same effect as ABORT.)

   You may cause messages to be sent in succession by different triggers, with no intervening block checks if you wish; however, at least one full character of idle (or $^{7}\varepsilon$ flag, in the case of Bit-Oriented Protocols) will be transmitted between blocks. When leads are switched (as indicated on the Interface Setup screen; see Section 12), the interface leads will be controlled between blocks.

4. *Abort*. This selection causes the message to which it is appended to abort before completion. When selected with Bit-Oriented Protocols, this action causes the INTERVIEW to transmit seven consecutive 1's at the end of the message. (For non-BOP transmissions, selecting ABORT has the same effect as selecting NONE.)

25.3 Internal Flags

Internal flags are bits that can be set on or off and sensed by triggers. Eight internal flag bits are shared among the Trigger Setup screens. Any combination of flag bits can be controlled by any trigger or combination of triggers.

> NOTE: The flag bits on the Trigger Setup can be controlled and monitored on the Protocol Spreadsheet, where they are referred to as trig_flag (see Section 30).

By default the Flags option is NO. Three other selections are available in the rotating window: SET, INC, and DEC.

(A) Set

When you select SET, a flag mask appears (see Figure 25-5). Use the arrow keys (→ and ←) to move the cursor to the bits you wish to set.

Enter a 1 or a 0 in any position you wish to set. Enter an X for "Don't Care." The trigger will not change the existing value of this bit.

Figure 25-5 A set of eight flag bits may be set on any of the trigger menus.

(B) Increment

The internal flags, consisting of Flags 0 through 7, can be thought of as a binary number. This action increases the value of the flags by one each time the trigger is true. (Other trigger actions may change the value of the flag bits in the intervening period.) Incrementing flags is one technique for controlling recursive routines.

As the flag bits increment past 255, they roll over to zero.

(C) Decrement

This action decreases the value of the flag byte by one each time that the trigger is true. In the event that the flag decrements below zero, the value of the byte wraps to 255.

> NOTE: The value of the flag bits is always reset to zero when you enter Run mode.

25.4 Enhancing the Display

Triggers can be used to enhance display data selectively. Data on either or both sides of the line can be enhanced. Enhanced data is also stored in the character buffer with the enhancements for later review.

(A) BOTH, DTE, or DCE

Select **Enhance:** BOTH, DTE, or DCE to enable enhancement options (see Figure 25-6). BOTH indicates that enhancements will be turned on or off on both TD and RD data at the same time.

DTE pinpoints TD data for enhancement; DCE specifies RD data for enhancement.

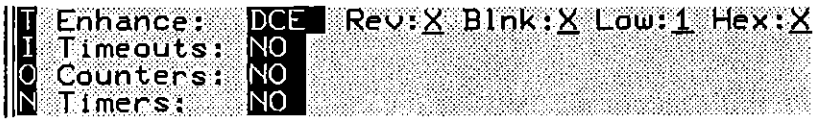


Figure 25-6 Data may be enhanced with respect to DTE, DCE, or both.

Four options, **Rev**, **Blink**, **Low**, and **Hex**, appear to the right. To turn on an enhancement, enter a 1 on the line immediately following it. To turn off an enhancement, enter a 0 on the same line. When an X follows the enhancement, the trigger takes no action.

1. *Reverse image.* Reverse-imaged (**Rev**) characters are presented as dark letters on a lighter background.

2. *Blink.* **Blink** causes data to blink on and off rapidly. This is the most conspicuous highlight for small portions of data.

3. *Low intensity.* **Low** has no effect on the plasma display. However, if you have installed a black and white monitor, it provides a low-intensity highlight for selected data.

4. *Hexadecimal.* When Hex is turned on, all data affected by the trigger is displayed in hexadecimal. Once data is stored in the buffer as hexadecimal, it remains in hexadecimal form.

(B) Color Enhancement

Color enhancement is controlled by the settings of three trigger enhancements: Reverse, Blink and Low. The three combined settings are mapped to color enhancements on the Miscellaneous Utilities screen as described in Section 17.

25.5 Controlling Timeouts

Each Trigger Setup screen can restart or stop either or both of the two timeout timers. These timers decrement from a value set on any of the triggers and, like flags and counters, serve as useful trigger conditions for internal program control.

When Timeout: YES is selected, identical new fields appear for Timeout #1 and Timeout #2. Both fields may be filled in on the same trigger.

(A) RESTART

Select RESTART to start or reset the timeout timer. The amount of time remaining on the timeout timer is entered in the data entry field provided (see Figure 25-7).

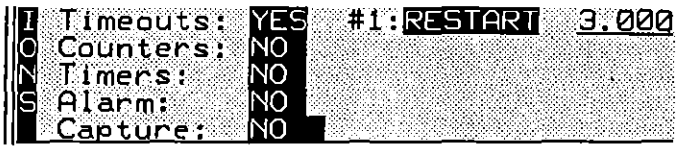


Figure 25-7 Timeout #1 activated to expire in three seconds.

1. *Entering timeout values.* The duration of the timeout is entered in seconds in the 5-character data-entry field provided. To enter a timeout value that is less than one second, use a leading zero before the decimal point, as follows: 0.25. The smallest valid timeout is 1 millisecond (0.001). The largest valid timeout is 65.535 seconds.

   Create a ten-minute timeout as follows: Start a timeout with a value of 60 seconds. When it expires, restart a similar timeout and increment a counter. When the counter equals ten, ten minutes will have elapsed.

(B) STOP

Select STOP to halt and clear the timeout timer, without causing the timeout to occur. If Timeout is selected as a trigger condition, the condition will not become true in this instance.

NOTE: Timeouts created on a Trigger Setup screen can be monitored and controlled from the Protocol Spreadsheet. These timeouts are entered as **trig_timeout_1** and **trig_timeout_2** when referred to on the Protocol Spreadsheet.

25.6 Counters

Each Trigger Setup screen can control two counters. These counters can be unique to the trigger (controlled only by it), or they may be shared with other triggers, which can monitor them and change their values. As long as the same counter name is used, the same counter is invoked.

NOTE: Counter names used on the Protocol Spreadsheet also refer to these counters, if the names match any counter name on the Trigger Setup screens. This means that program control can be shared between these screens and the spreadsheet.

NOTE: Trigger Setup screens monitor counter values from 0 to 999,999. However, Protocol Spreadsheet triggers can monitor counter values up to 4,294,967,295.

(A) Menu Fields

When **Counters:** YES is selected, two sets of new menu fields, labeled 1st and 2nd, appear (see Figure 25-8).

1. *Counter name.* Enter the counter name in the field provided. The name may be up to eight characters long and must start with a letter. Upper- and lower-case alpha characters, numerals, and underscore (_) are legal in the other positions.

   When the name field is empty, the trigger takes no action for that counter field.

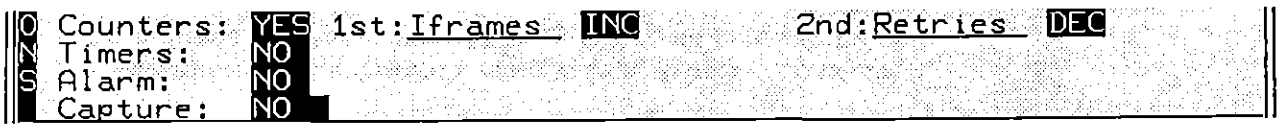


Figure 25-8 One counter is incremented, another decremented in this action.

2. *No.* The default selection is NO. It allows you to disregard one or both counters.

3. *Increment*. When INC is selected, each trigger occurrence adds 1 to the counter.

4. *Decrement*. When DEC is selected, each trigger occurrence subtracts 1 from the counter. When a counter decrements below zero, it wraps not to 9,999,999, but to the decimal equivalent of $2^{32} - 1$, the actual maximum value of a 32-bit counter. The seven least-significant decimal digits that appear on the Tabular Statistics screen are 4967295. The complete number is over 4 billion.

5. *Set*. Select SET in order to specify the value which the counter will take when the trigger becomes true. Then, enter the decimal value of the counter in the field provided. The field is six positions long, making it possible to set counters to a value from 0 to 999999. Any leading positions not specified in your entry will be set to zero. This action does not cause statistical samples to be taken, nor does it reset last value, minimum value, maximum value, or average value for the counter. (Compare to Sample and Clear.)

6. *Sample*. This action causes the counter to reset to zero and causes measurements to be taken for last value, minimum value, maximum value, and average value. Refer to Section 20 for an explanation of how statistics are gathered and tabulated.

7. *Clear*. This action resets the counter to zero and also resets minimum value, maximum value, and average value for the counter.

25.7 Timers

Two timers are shared among the Trigger Setup screens. While these timers are not available as trigger conditions, they can be run and sampled as trigger actions. When timers are invoked by triggers, their values can be tracked on the statistics screens (see Sections 20 and 21).

> NOTE: Timer names referred to on the Protocol Spreadsheet may also be used on Trigger Setup screens. Thus, timer control of programs is shared between these screens and the spreadsheet.

(A) Menu Fields

The default timer selection is NO. When YES is selected, two identical subfields appear, for Timer 1 and Timer 2 (see Figure 25-9).

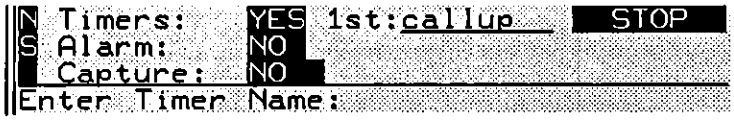


Figure 25-9 One or two timers may be controlled by the same trigger (second field not shown).

1. *No.* The default selection for each Timer is also NO. This allows trigger action to disregard both timers or to focus on one timer, if necessary.

2. *Restart.* When selected, RESTART causes the timer to reset to zero and begin incrementing. RESTART does not cause statistical measurements to be taken. (Compare to SAMPLE and CLEAR.)

3. *Stop.* The STOP action suspends the timer and allows it to retain its value. The timer may be started again at this value by a CONTINUE action on another trigger.

4. *Continue.* CONTINUE, when selected, causes the specified timer to increment, starting from the value at which it was stopped.

5. *Sample.* The SAMPLE action resets and stops the timer. Prior to resetting the timer, its value is read as a "last" value and passed along for other statistical measurements. Refer to Section 20 for an explanation of how statistics are gathered and tabulated.

6. *Clear.* The CLEAR action resets the current value, the last value, the minimum value, the maximum value, and the average value of the timer. Refer to Section 20 for an explanation of statistical measurements.

25.8 Alarm

The alarm is a short beep. The alarm is useful for calling your attention to the data being analyzed, especially when the situation of interest occurs infrequently. When you select Alarm: YES, it is sounded each time the trigger becomes true.

25.9 Capture of Data in the Screen Buffer

Capture of character-oriented data to the screen buffer can be stopped and restarted by triggers, using the Capture action (see Figure 25-10). When capture is turned off, data is neither presented to the screen nor stored in the buffer.

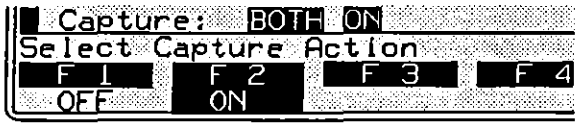


Figure 25-10 Data capture to the screen buffer can be controlled by triggers.

(A) NO, BOTH, DTE, or DCE.

The default **Capture** selection is NO. This represents no change; that is, the trigger does not influence character buffer capture. By default, data is continuously captured in the character buffer.

Select BOTH to control capture to the character buffer for TD and RD data at the same time. Select DTE to control only TD data; DCE to control only RD data.

1. *OFF, ON.* Select OFF to suppress data from the screen buffer. Select ON when another trigger has turned off capture and you wish to begin storing data in the buffer again.

26 The Trigger Summary Screen

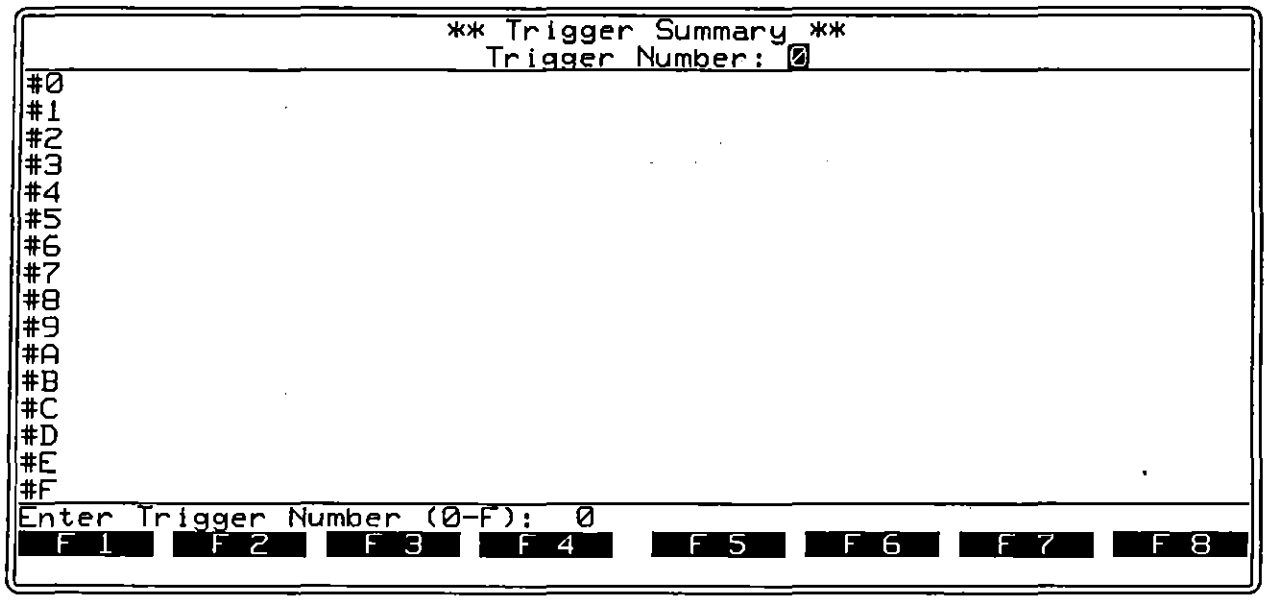


Figure 26-1 Default Trigger Summary screen.

26 The Trigger Summary Screen

The Trigger Summary screen is the access screen to all Trigger Setup screens. The default Trigger Summary screen is shown in Figure 26-1. Call up the Trigger Summary screen by pressing the function key marked TRIGS on the main Program Menu. With the summary displayed, access any trigger by typing the number of the desired screen (0 through F). To see a synopsis of configured Trigger Setups, you may return to the summary screen from any trigger menu by pressing [PAGE].

Entries you make on any of the 16 Trigger Setup screens appear on the summary in abbreviated form. Each setup screen is allotted a one-line summary. A summary of conditions appears on the left-hand side of the line; a summary of actions appears on the right-hand side of the line. The summary for Trigger Setup screen 0 (Figure 26-2) is shown in Figure 26-3.

Abbreviations for possible Trigger Setup conditions are listed in Table 26-1. Abbreviations for Trigger Setup actions are given in Table 26-2.

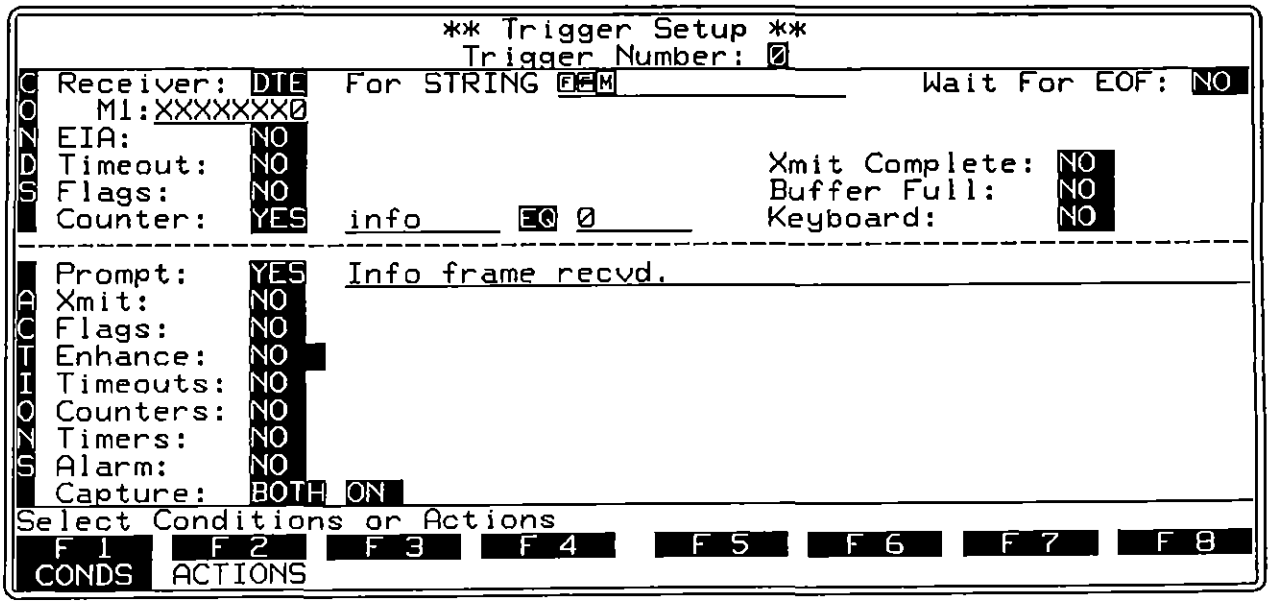


Figure 26-2 Entries on each Trigger Setup screen are indicated on the Trigger Summary.

NOTES:

Abbreviations displayed on the Trigger Summary screen are not necessarily keywords and should not be referred to when you are typing entries on the Protocol Spreadsheet.

When multiple conditions or actions are selected on a single Trigger Setup screen, the summary screen may not be able to show all selections; however, as many conditions and actions as possible will be displayed in the available space.

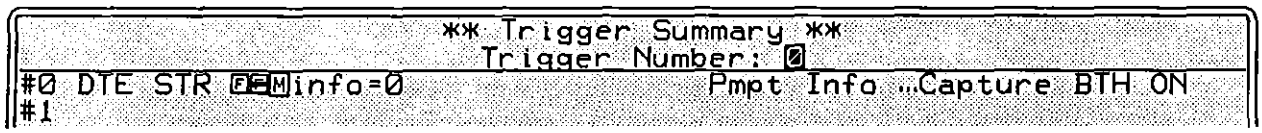


Figure 26-3 Summary of entries made on Trigger Setup screen 0, shown in previous figure.

Table 26-1
Abbreviations, Trigger Summary Conditions

Receiver (Word does not appear on summary.)

DTE, DCE
STR: String, 1OF: One of (Character string also appears for STR and 1OF.),
G: Good BCC, B: Bad BCC, PrErr: Parity Error, FrErr: Frame Error,
A: Abort, M: Bit Mask

EIA:
RTS, CTS, CD, DTR, DSR, RI, UA

TimeOut 1, 2

Xmit_Cmpl: Transmission Complete

Buffr_Ful: Buffer full

Flag: (Value only appears.)

Counter: (Name and value only appear.)

KeyBd: Keyboard (Key Indicated.)

Table 26-2
Abbreviations, Trigger Summary Actions

Pmpt: Prompt (Prompt string also appears.)

Xmit: Transmit (Xmit string also appears.)
[G]: Good BCC, [B]: Bad BCC, [A]: Abort
(Nothing appears if no BCC is selected.)

Flag: INC: Increment, DEC: Decrement (Value only appears if selection is SET.)

ENH: Enhance Display
BTH: Both DTE and DCE, DTE, DCE
REV=: Reverse, **BLN=:** Blink, **LOW=:** Low, **HEX=:** Hexadecimal

TO #1, TO #2: Timeout #1 or 2
RST: Restart, **STP:** Stop

Counter: (Only name and value appear.)
INC: Increment, **DEC:** Decrement, **=:** Set
SMP: Sample, **CLR:** Clear

TM: Timer (Name also appears.)
RST: Restart, **STP:** Stop, **CNT:** Continue, **SMP:** Sample, **CLR:** Clear

Alarm: Audible Alarm

Capture: Capture Memory
BTH: Both DTE and DCE, DTE, DCE
ON, OFF

27 Programming Blocks

1. These keys are at highest level:

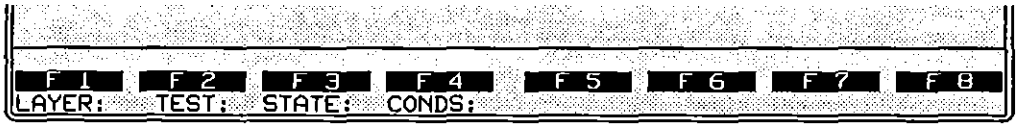
Press MORE to obtain:

2.

Press MORE to return to:

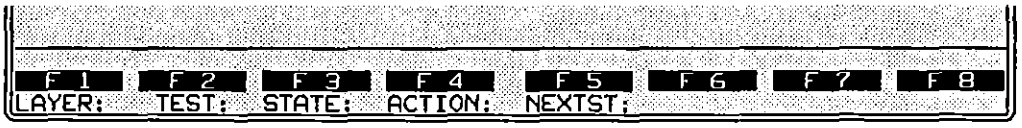
Press STATE: enter name and press RETURN or the spacebar to obtain:

3.



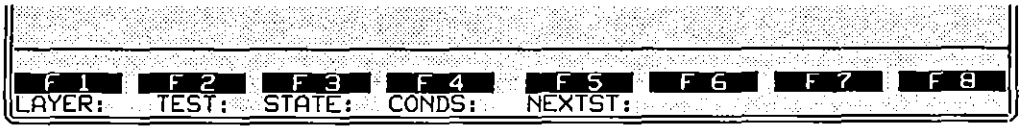
Press CONDS: (Conditions) to obtain variable set of Conditions.* Press F8 to access alternate racks of Conditions keys. Complete entries, then press DONE to obtain:

4.



Press ACTION: to obtain variable set of Actions.* Press F8 to access alternate racks of Actions keys. Complete entries, then press DONE to obtain:

5.



Press NEXTST:, enter name of next state, and press RETURN or the spacebar *to obtain rack 3.* (Pressing CONDS: obtains variable sets of conditions.)

*Conditions and Actions available depend on what protocols are loaded and what layer number you have specified. The following selections are always available:

GENERAL CONDITIONS		GENERAL ACTIONS	
ENTER_STATE	TIME	FLAG	LOAD_PROGRAM
TIMEOUT	FLAG	ACCUMULATE	PROMPT
KEYBOARD	ON_SIGNAL	SIGNAL	PRINT
BUFFER_FULL		COUNTER	TRACE
COUNTER		TIMER	ALARM
		TIMEOUT	RECORD

Figure 27-1 Function key hierarchy, Protocol Spreadsheet.

27 Programming Blocks

The Protocol Spreadsheet is a highly flexible programming approach which enhances trigger conditions and actions provided on the Trigger Setup menus, furnishes new general options, and incorporates protocol-specific conditions and actions on a layer-by-layer basis.

27.1 Before You Begin a Spreadsheet Program

Be certain prior to programming that you have loaded the Personality Packages for the protocols you will be testing. Automatic protocol options are part of each layer's Personality Package. These packages are loaded from the Layer Setup screen as described in Section 8.

Check the configuration of the various Test Setup screens before you test or save your program, since the behavior of the INTERVIEW during testing is influenced by setup selections.

27.2 Creating a Spreadsheet Program

Press [F3] to access the Protocol Spreadsheet from the Program Menu. Any program which you have loaded from the File Maintenance screen appears on the spreadsheet. If no program has been loaded or created, the Protocol Spreadsheet, since it is a free-form menu, will be blank except for a header line, function key labels, and tildes (-) down the left side of the screen. Tildes always mark the end of your program file.

(A) Two Sets of Function Keys: Programming and Editing

Two full sets of softkeys are active with the spreadsheet. One set of softkeys groups available programming options, including keywords (LAYER:, TEST:, CONDITIONS: etc.). The alternate set groups sophisticated editing functions.

These editing functions, which complement the editing keypad, are accessible from the spreadsheet at any time. Press [EDIT] to activate edit softkeys. Press [EDIT] again to return to program softkeys. For a discussion of editing options, refer to Section 29.

(B) Programming Functions

Use the programming softkeys to make program entries, from the highest level of the program (OBJECT), down to individual trigger conditions and actions and their subfields. Softkeys guide you as you create your program by listing

available options and providing correct syntax wherever possible. (Errors are indicated by strikeover of incorrect text as you make your program entries.) For each level of function keys, a cue near the bottom of the screen explains selectable options or prompts you for keyboard entry.

Program softkeys are immediately available when you enter the spreadsheet. The hierarchy of the program softkeys is shown in Figure 27-1. The conditions and actions listed, which are always available, are explained in Section 30. Other trigger conditions and actions are added when protocol packages are loaded. Because protocols are layer-specific, trigger options will vary from layer to layer. For each LAYER block within your program, different options are likely to appear when you enter the keyword CONDITIONS or ACTIONS. For more information on the specific trigger options enabled by a protocol, consult the section devoted to that protocol (see Table of Contents, Section 35 and following.)

You also have the freedom of typing in any program entry, if you prefer, as long as you enter the block identifiers and conditions and actions keywords as they would be posted on the screen by softkeys. Syntax errors still are automatically highlighted by a strike-through.

> NOTE: Softkey labels are not necessarily legal spellings on the spreadsheet. Pressing the function key usually posts an expanded keyword on the screen. Use these expanded keywords when typing entries.

1. *Successive racks of softkeys.* The rack of softkey options at the bottom of the spreadsheet screen (or the instructional prompt on the third line up from the bottom, or both the option rack and the prompt) will change automatically each time you *complete* a keyword entry. Keyword entries are complete when you make them via softkey *or* when you type the keyword followed by a space or a hard RETURN. (Pressing the softkey has the same effect as typing the keyword and then typing a space to complete the entry.)

   Programming movement is generally *down* the tree of softkey racks, as in this series of keywords:

   ```
   CONDITIONS: EIA CTS ON
   ```

   Each of the four keywords was selected from a rack of options, and each succeeding rack is a step farther down the "branch." The rack that follows ON, however—listing RTS, CTS, CD, and other EIA leads—is back up the tree, since "there is nowhere to go but up," and since a trigger with multiple EIA conditions (like the following) is valid.

   ```
   CONDITIONS: EIA CTS ON CD OFF
   ```

2. *Additional racks of valid softkeys.* There may be many more keywords that are valid to enter at a given point in the program than are showing on one rack of softkeys. Additional racks may be accessible via the [F8] softkey (MORE); and *higher* racks are generally available via the [DONE] key. In this series, [DONE] was pressed following the softkey for ON to access the softkey for COUNTER:

```
CONDITIONS: EIA CTS ON
  COUNTER xmit LT 6
```

   In the next series of keywords, [DONE] was pressed *twice* following the softkey for ON, to access the softkey for ACTIONS:

```
CONDITIONS: EIA CTS ON
ACTIONS: SEND "((FOX))" GOOD_BCC
```

   Note that [DONE] is not a valid keystroke following CTS above, since the condition syntax is not "done." Whenever it is not valid to move to a rack of softkeys higher up the tree, [DONE] produces an alarm tone.

   Note also that it is never necessary to press [DONE] if you are typing in your keywords directly from the keyboard. [DONE] merely changes the *rack* that is showing, not the entire set of keywords that is valid. A keyword does not have to be showing to be typed in legally.

3. *Insert mode versus overstrike mode.* Touch-typists in particular should be aware that the Protocol Spreadsheet has an insert mode as well as an overstrike mode. The insert mode is invoked by either of two keys, [INSERT CHAR] or [INSERT LINE]. When the mode is enabled, the word <Insert> appears at the top left of the screen. In insert mode, the programmer types in a block of data while succeeding text is pushed forward with every keystroke.

   Press [INSERT CHAR] (but not [INSERT LINE]) a second time to exit *insert* mode and return to *overstrike* mode.

The remainder of this section is devoted to the fundamentals of program structure and to programming components available on the Protocol Spreadsheet which are independent of trigger options.

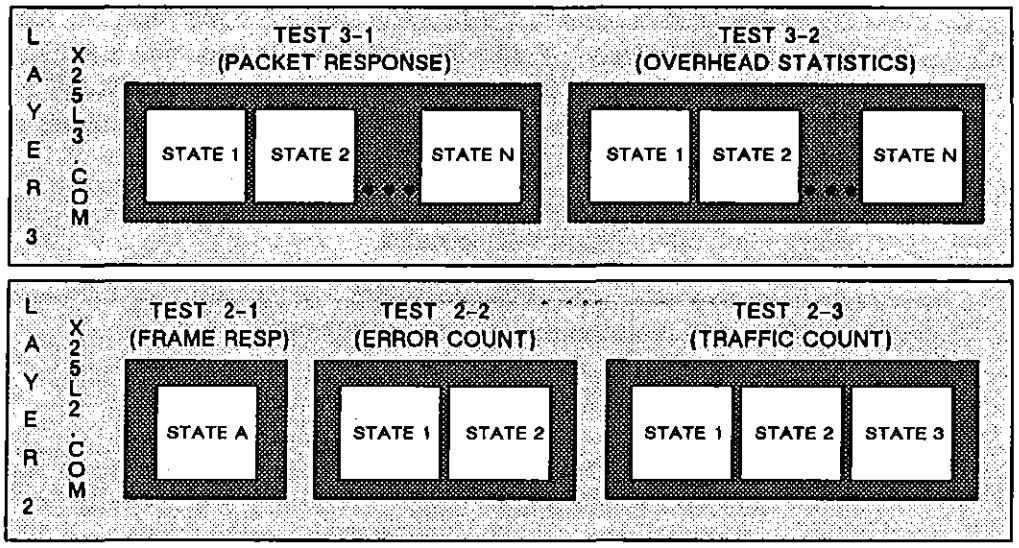


Figure 27-2 Discrete states inhabit separate tests at separate layers.

27.3 Program Structure

The components of the INTERVIEW's programming model, introduced in Section 23, are integrated into a spreadsheet program as discrete blocks according to specific structural rules. Compare the abstract program model in Figure 27-2 to the spreadsheet program outlined in Figure 27-3.

(A) Block Identifiers

The INTERVIEW's compiler must respect the distinction between one layer and the next and between one test and the next. Further, it must group triggers into designated states and track the transition from one active state to another. To indicate the boundaries of these various blocks, specific keywords are used. Each block normally begins with an identifier in upper-case letters, (optionally) followed by a colon. A block ends when a new block identifier is inserted in the program.

> NOTE: The identifier must not be enclosed in quotes (that is, must not be part of a text string) if it is intended as a block delimiter.

Available program blocks, from largest to smallest, are described in subsequent paragraphs. The valid block identifier for each is printed above its description.

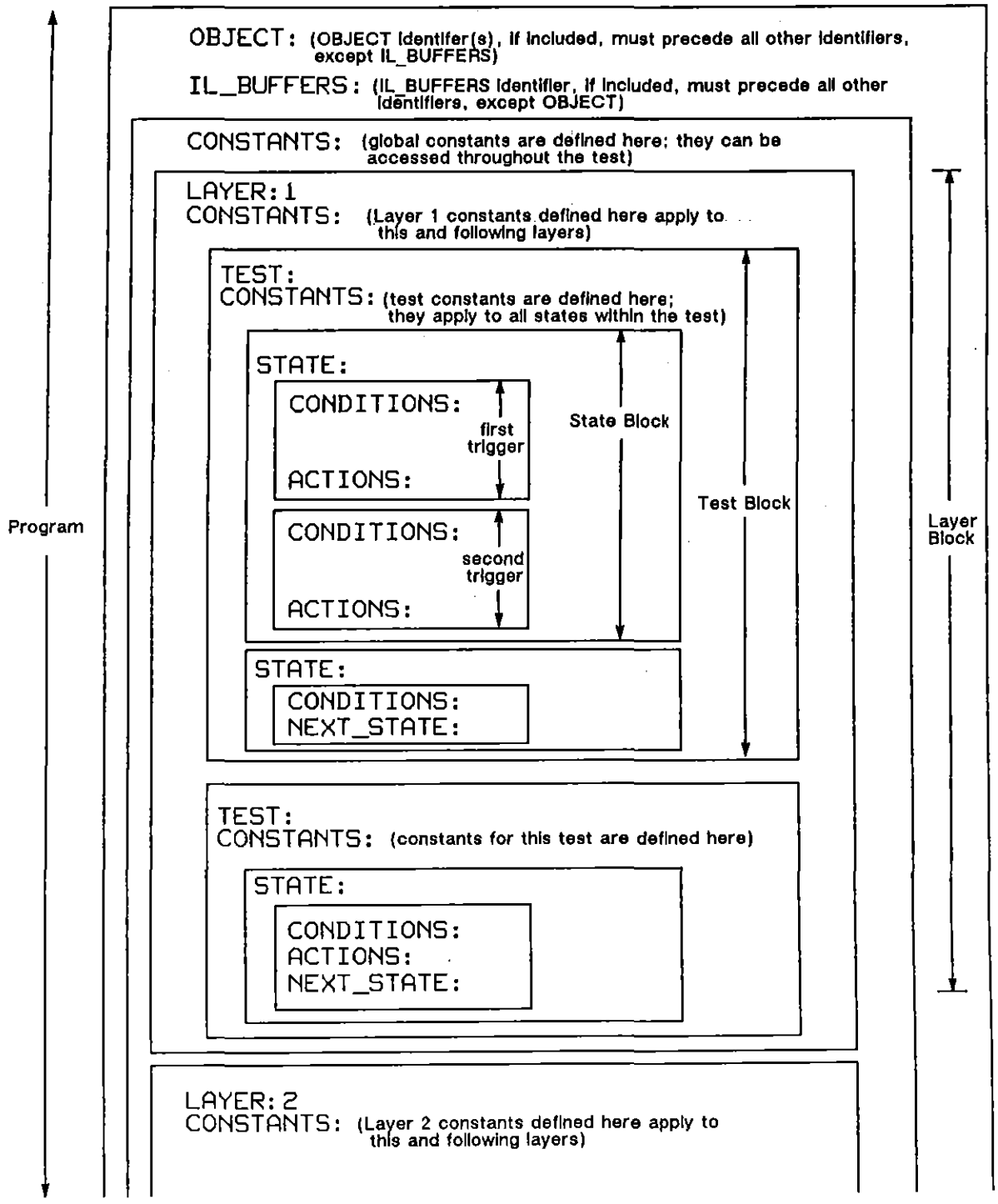


Figure 27-3 Program Structure. Component blocks begin with a keyword.

OBJECT:

1. *Referencing linkable-object files.* Use the OBJECT block-identifier to access the compiled code in a linkable-object file. See Section 27.4 below. The OBJECT identifier(s) must appear at the top of the Protocol Spreadsheet. IL_BUFFERS is the only identifier which may precede OBJECT. C regions or spreadsheet comments may also precede the OBJECT block identifier.

IL_BUFFERS

2. *Configuring the number/size of IL buffers.* Interlayer (IL) message buffers are used to pass data up the layers as it is received and down the layers as it is transmitted. Press the IL BUFS softkey to set the number and size of the IL buffers. The IL_BUFFERS identifier(s) must appear at the top of the Protocol Spreadsheet. OBJECT is the only identifier which may precede IL BUFS. C regions or spreadsheet comments may also precede the IL_BUFFERS block identifier.

CONSTANTS:

3. *Defining constants.* There are three legal locations for the definition of a constant: in the opening lines of a program, at the beginning of a layer, or at the beginning of a test. The relative placement of a constant's definition within a program determines its scope, or active range. For a complete discussion on constants, refer to Section 28.

LAYER:

4. *Layers.* The largest block, the layer, corresponds to the OSI model. There may be up to seven layers in any test.

TEST:

5. *Tests.* A layer may contain any number of simultaneous tests. Every test resides inside a layer.

STATE:

6. *States.* In turn, each test may contain any number of states. A state always resides inside a test. Only one state in each test is active at one time.

   Within each state, there may be a number of triggers. A trigger always resides inside a state. Each trigger is composed of a conditions portion and an actions portion.

CONDITIONS:

7. *Trigger conditions.* A single condition or a group of conditions is normally listed after the CONDITIONS identifier. Rules for grouping trigger conditions, as well as the meaning of each trigger condition, are explained in Sections 30 and 31.

ACTIONS:

8. *Trigger actions.* The ACTIONS identifier precedes the list of trigger actions. This list may be empty, or it may include one or several trigger actions. The various trigger actions are described in Sections 30 and 31.

NEXT_STATE:

9. *Next state.* The identifier NEXT_STATE, explained in the following paragraphs, can replace the ACTIONS: identifier in a trigger if there are no other actions; or it can follow the ACTIONS: identifier to indicate that branching to another state is one of several actions taken by the trigger.

(B) Run-time Transitions Between States

Run-time transitions between states are controlled by triggers. To indicate a run-time branch from one state to another, use the NEXT_STATE action, followed by (a) the name of the state you wish to go to, or (b) the NEXT token, indicating whatever state happens to follow sequentially in the spreadsheet program.

You may use a NEXT_STATE action once per trigger and as many times as needed in one state to allow for multiple branching possibilities.

When two triggers come true at the same time and both potentially result in branching to another state, the trigger which is checked last (the last trigger sequentially displayed on the spreadsheet) will cause branching to the state it names. (The first trigger will not cause branching.)

Look at the two triggers shown in the example which follows. The first searches for any SDLC Information frame. The second searches for an Info frame with a particular frame address. By definition, whenever the second trigger is true, the first trigger is also true. When an Info frame with the correct address is received, the second trigger causes the test to branch to the State **respfrm**. However, if these triggers are reversed as shown in the second example, the test *always* branches to the State **otherfrm**, regardless of the frame address.

correct order

```
STATE: frmadd
   CONDITIONS: DTE INFO
   NEXT_STATE: otherfrm
   CONDITIONS: DTE INFO ADR=C1
   NEXT_STATE: respfrm
```

wrong order

```
STATE: frmadd
   CONDITIONS: DTE INFO ADR=C1
   NEXT_STATE: respfrm
   CONDITIONS: DTE INFO
   NEXT_STATE: otherfrm
```

(C) Recommended Format

The format of a Protocol Spreadsheet is entirely flexible. The only rule is that block identifiers must (with rare exception) be included in the program to designate boundaries between programming blocks.

The following is a suggested program format. To create a visual distinction, the keywords which define program blocks are placed at the beginning of a line. Smaller blocks are indented to show that they reside within a larger block. An automatic indent feature, described in Section 29, is included as an editing function and is turned on by default.

```
LAYER: 1
  TEST: echo_msg
    STATE: message
      CONDITIONS: DTE STRING "hello"
      ACTIONS: PROMPT: "Spreadsheet trigger true."
      NEXT_STATE: echo

    STATE: echo
      CONDITIONS: DCE STRING "hello"
      ACTIONS: PROMPT " Echoed message received"
      NEXT_STATE: message
```

(D) Omitted Block Identifiers

It is recommended that, for ease of tracking a program, block identifiers be placed at the beginning of every block. However, in brief programs, certain block identifiers may be omitted.

It is, in fact, possible for a program to begin with a STATE identifier. The compiler then assumes that you have begun the first test inside the first layer of the program. To start another program block, you must use a STATE, TEST, or LAYER identifier.

> NOTE: Any constant declared in the opening lines of a test which omits the LAYER and/or the TEST keyword is still a global constant, as long as it precedes a STATE or CONDITIONS identifier.

27.4 Compiled Spreadsheet

Using the Compile command on the File Maintenance screen, you can compile and save the contents of the Protocol Spreadsheet in a linkable-object file. Later, this program can be combined with an active spreadsheet program. To do so, simply reference the file at the top of the Protocol Spreadsheet.

(A) The OBJECT Block-Identifier

Use the OBJECT block-identifier on the Protocol Spreadsheet to access the compiled spreadsheet code in a linkable-object file.

> **Note to C Programmers:** The OBJECT identifier may also be used to access definitions for user routines. Refer to Section 59.4(C).

1. *Placement.* The OBJECT block-identifier(s) must appear at the top of your spreadsheet program, ahead of any other identifier (except IL BUFS). Access the OBJECT: softkey by pressing MORE on the initial rack of softkeys. Notice that the MORE and OBJECT: softkey tokens are not available once any other programming block-identifier has been selected.

   > **NOTE:** Use OBJECT in your active spreadsheet program only. Do not incorporate it in a spreadsheet that will be compiled and saved as an LOBJ file. Although the code will compile, the referenced LOBJ file will not be read.

2. *Format.* The format for the OBJECT block-identifier is as follows:

   ```
   OBJECT: "filename.o"
   ```

   The identifier references only one linkable-object file, but you may include as many OBJECT identifiers as you wish.

   The relative or absolute pathname of the linkable-object file is enclosed in quotation marks.

3. *Search rules for linkable-object files.* As your spreadsheet program compiles, the INTERVIEW's filing system is searched for the linkable-object files referenced in OBJECT identifiers.

   - If the referenced LOBJ filename begins with *FD1/*, *FD2/*, or *HRD/*, the INTERVIEW interprets it as the absolute pathname and makes only that one search.
   - Pathnames beginning with a / indicate that the root directory on each drive should be the beginning point of the search. The drives are searched in the following order: current drive, FD1, FD2, and HRD.
   - Otherwise, the name may be a one-word filename or a relative pathname which includes the directories leading to the file. The highest directory in a relative pathname must reside in the current directory or in one of the */lib* subdirectories. The following directories—and only the following directories—are searched, in the order given:

1. current directory on the current drive (indicated on the File Maintenance screen)
2. */usr/lib* on the current drive
3. */sys/lib* on the current drive
4. *FD1/usr/lib*
5. *FD2/usr/lib*
6. *HRD/usr/lib*
7. *FD1/sys/lib*
8. *FD2/sys/lib*
9. *HRD/sys/lib*

If the pathname is not located in any of these directories, the program will not compile and an error message will be returned to the operator.

(B) Compiled LOBJ Code is Combined with Spreadsheet

During compilation, the compiled spreadsheet in the LOBJ file is combined with your active spreadsheet program. This means that the LOBJ code must be compatible with the current menu setups and spreadsheet program—as though the source code of the LOBJ file were actually present in the spreadsheet buffer.

(C) Counter and Flag Conditions

Special consideration is given to COUNTER and FLAG conditions during the Compile SPREADSHEET operation. The system identifies the condition as either transitional or status. (See Section 30.2.) If it is used both ways in the same spreadsheet file, it will always be identified as transitional.

Within a single spreadsheet program, you may reference more than one LOBJ file which uses the same COUNTER or FLAG. If one of the files uses the COUNTER (or FLAG) as a transitional condition, however, all other referenced files containing the same COUNTER (or FLAG) must also use it as a transitional condition at least once. This rule ensures that each action on the specified COUNTER (or FLAG) will consistently trigger the appropriate COUNTER (or FLAG) conditions.

(D) Advantages of Compiled Spreadsheet

Linkable-object files assist the programmer in efficiently using the INTERVIEW's memory and spreadsheet buffer.

- When commonly utilized conditions and actions are saved in linkable-object files, space in the spreadsheet buffer otherwise dedicated to this purpose can be used for additional programming.
- Since the code in LOBJ files has already been compiled, the INTERVIEW can support a larger program without a corresponding increase in compilation time.

- The spreadsheet code in a linkable-object file is transparent to the configuration of the unit. LOBJ files created on one unit can be used on a unit configured differently, as long as the code is compatible with the various menu parameters.

27.5 Configuring the Size/Number of IL Buffers

Interlayer (IL) message buffers are used to pass data up the layers as it is received and down the layers as it is transmitted. (See Section 23 on the layered-programming model and Sections 33 and 66 for more information on the uses of IL buffers.) The INTERVIEW allocates IL buffers, as needed, to pass data between layers. Then, the buffers are automatically erased and used again. In this way, the INTERVIEW maximizes its use of available memory space. Without these reusable buffers, data in Run mode would quickly eat up all of the memory in the unit. (See Section 66.3(A) for information on manipulating IL buffers.)

IL buffers contain the data itself or point to the memory location (outside the buffer) of the data. It follows, therefore, that the larger the IL buffer, the more data it can hold. By default there are 16 IL buffers that can be in use at a given time. The size of each buffer is 4,096 bytes.

When you are performing emulation with windowing, you can quickly use up these sixteen buffers. Once all buffers are in use, additional data is lost. To prevent this from happening, you may want to reconfigure the number and size of IL buffers.

Press the IL BUFS softkey to set the number and size of the IL buffers. Figure 27-4 shows the softkey selections. Select one of seven number/size combinations for the INTERVIEW's IL buffers. The default selection is 16/4K. This means that the INTERVIEW will have a maximum of 16 IL buffers in use at a given time, each one 4,096 bytes (4 Kbytes). This size, and all others, includes a 32-byte buffer header.

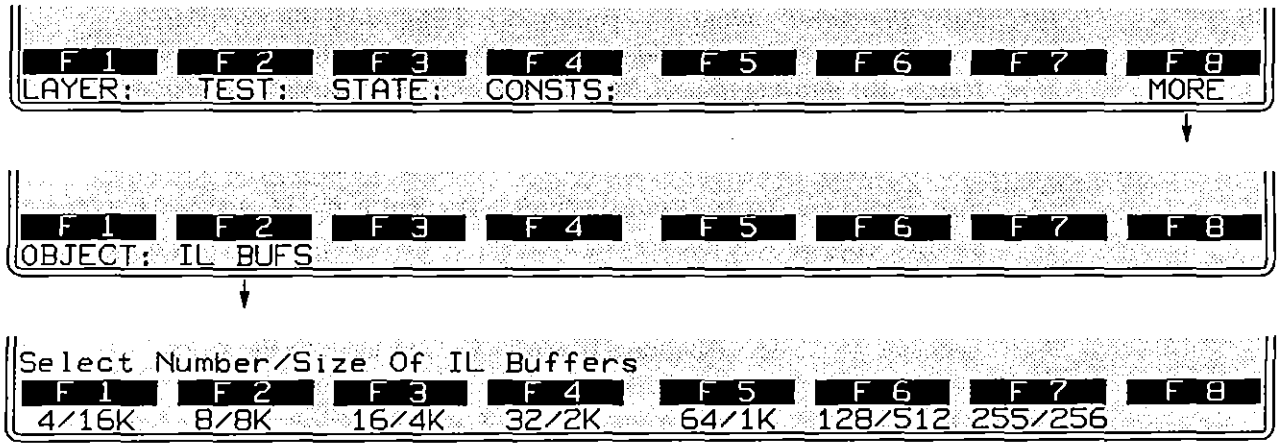


Figure 27-4 Softkey path to the seven number/size combinations for IL buffers.

Note to C Programmers: There are two preprocessor directives—*#pragma il_buffers* and *#pragma il_buffer_size*—which the C programmer may also use to configure the IL buffers. These directives provide additional flexibility. See Section 66.1(A).

Notice that each number/size combination utilizes 65,535 bytes (64 Kbytes) of RAM. This total represents the maximum amount of RAM that can be allocated for IL buffers from the Interlayer Buffers menu.

NOTE: Keep the following points about object-file compatibility in mind when setting the number/size of IL buffers:

- If the number of buffers is less than or equal to 16, the file will load and operate on a unit with equivalent hardware, yet a software release earlier than 8.00.
- If the number of buffers is greater than 16, the file cannot be loaded on a unit with a software release earlier than 8.00.
- An object file generated under a software release earlier than 8.00 will run on software revision 8.00, or higher, with 16 buffers of 4 Kbytes each (the default).

27.6 Comments in a Spreadsheet Program

You may write comments to yourself or to others who may view your spreadsheet program. Comments begin with /* and end with */, as in the examples below. Use comments generously throughout spreadsheet programs. Since comments are ignored by the compiler, they do not affect the compilation time of the program.

(A) Characteristics

1. *Valid characters.* When an opening /* is detected by the compiler, everything that follows is disregarded until a closing */ is encountered. This means that all hexadecimal, control, and ASCII characters (or character combinations) are valid in comments. The [F], [S], [X], and not-equal symbols are also legal entries.

   Two entries are not legal in comments. The first is the [M] symbol. It cannot be used because it is not a valid Protocol Spreadsheet entry. (Bit masks on the spreadsheet are delimited by « and »). An alarm will sound if you try to use the bit-mask symbol. The second invalid entry is the closing delimiter (*/). An embedded */ causes the comment to be ended prematurely. Since the remainder of the comment (and the programmer's *intended* closing */) is a syntax error, the program will not compile.

2. *Length.* For practical purposes, make comments as long as you wish. They may span several lines, or they may be empty.

3. *Location on spreadsheet.* Comments may be placed within any of the programming blocks: OBJECT, IL_BUFFERS, CONSTANTS, LAYER, TEST, STATE, CONDITIONS, ACTIONS, or NEXT_STATE. In CONDITIONS blocks, however, they must appear with at least one valid condition. The following CONDITIONS block containing only a comment will cause compilation to be aborted:

```
STATE: message
  CONDITIONS: /* KEYBOARD " "*/
  ACTIONS: SEND "((FOX))" GOOD_BCC
```

Since the compiler ignores anything inside the /* */ delimiters, it can find nothing in the CONDITIONS block. When you go to the Protocol Spreadsheet and search for error messages, the following message will be displayed: *"Empty Conditions Section."*

Comments may not be embedded within a keyword. This program also will not compile:

```
STATE: message
  CONDITIONS: KEY/* This comment will cause a syntax error*/BOARD " "
  ACTIONS: SEND "((FOX))" GOOD_BCC
```

(B) Using Comments

Comments are particularly useful in describing the purpose of a programming block. Let's return to the two programming examples in which branching to another state occurs based on DTE Info-frame addresses. The following comment makes the programmer's intentions clear.

```
STATE: frmadd

/* If a DTE INFO frame has an address of C1, go to state "respfrm." For all other
DTE INFO frames, go to state "otherfrm." */

  CONDITIONS: DTE INFO
  NEXT_STATE: otherfrm
  CONDITIONS: DTE INFO ADR= C1
  NEXT_STATE: respfrm
```

Comments can be useful debugging tools. Suppose the same comment appeared in the programming example with the order of the two triggers reversed.

```
STATE: frmadd

/* If a DTE INFO frame has an address of C1, go to state "respfrm." For all other
DTE INFO frames, go to state "otherfrm." */

  CONDITIONS: DTE INFO ADR= C1
  NEXT_STATE: respfrm
  CONDITIONS: DTE INFO
  NEXT_STATE: otherfrm
```

With the comment present, it is easier to identify the discrepancy between the programmer's expectations and the actual program.

28 Constants

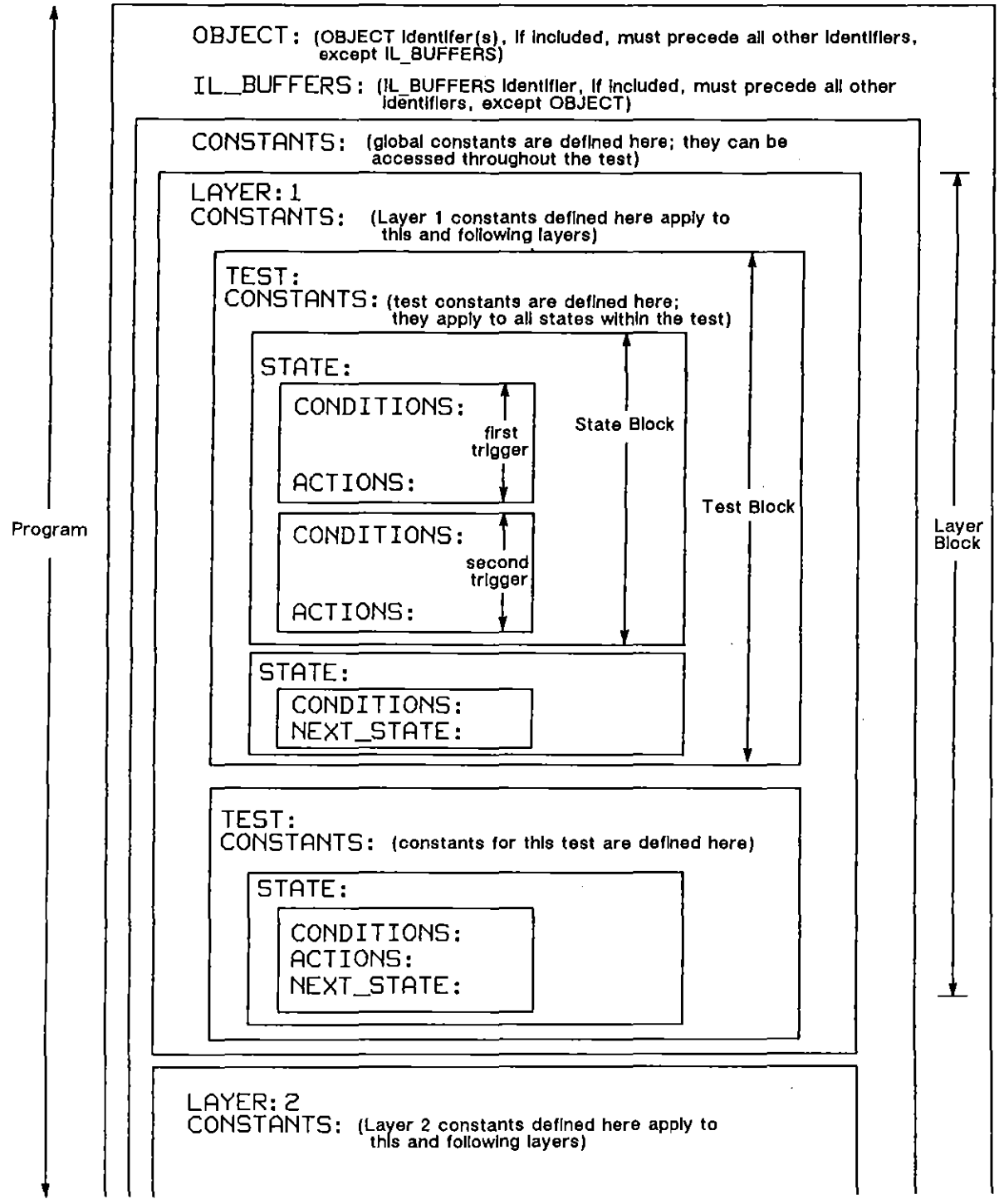


Figure 28-1 Constants may be defined in three different locations: before the first layer, after a layer identifier, or after a test identifier.

28 Constants

The Protocol Spreadsheet permits the use of constants as a means of simplifying the creation and modification of test programs. A constant is merely a symbolic reference to a predefined string of characters.

28.1 Definition of Constants

There are three legal locations for the definition of a constant: in the opening lines of a program, at the beginning of a layer, or at the beginning of a test (see Figure 28-1). Constants must always be defined at the beginning of the programming block in which they are referenced. A test-level constant may not be preceded by a lower-level block identifier (STATE, CONDITIONS, or ACTIONS). A constant definition or definition block must be followed by another keyword.

A constant definition begins with the identifier CONSTANTS. A colon (:) may follow. The constant name is then entered. Next comes the definition, a text string which the constant represents. The constant name and text string may be separated by an equal sign (=). The text string is enclosed in double quotes. Each constant definition comprises a single logical line. Logical lines wrap as needed to subsequent lines on the spreadsheet, but they do not contain hard returns. See Section 29.1(A).

More than one constant can be defined following a single CONSTANTS identifier. Following is the suggested format for a constant definition block. A constant definition block (whether it contains one or more definitions) must be followed by another block identifier.

```
CONSTANTS:

  cmd_adr    = "03"
  resp_adr   = "01"
  resend     = "1.5"
  retries    = "10"
```

To include quotation marks or backslashes in the definition string of a constant, precede each with a backslash escape-character (\). Here, for example, is the constant definition of a general poll:

```
CONSTANTS: drop_A= "AA\"\"␅"
```

The backslash will be deleted by the parser when the constant definition is scanned:

```
AA""␅
```

If the constant is contained in a search or transmit string, it will not be scanned for the escape character or for closing quotation marks. The characters shown above represent the expanded constant. Notice that the enclosing quotation marks of the definition string are not actually part of the constant. In our example, the following string will be searched for or transmitted:

AA" "↵

28.2 Constant Names

Constant names must begin with a letter or underscore character. They may include any of the following characters: underscore (_), upper or lowercase letters, and decimal numbers 0 through 9. Upper and lower case letters are distinguishable in constant names; for example, constants big, Big, and BIG will not be confused by the INTERVIEW's compiler.

28.3 Scope

The relative placement of a constant's definition within a program determines its scope, or active range.

(A) Global Constants

If you want to be able to reference a constant anywhere within a program, you must define it in the first lines of the program. Only an OBJECT or IL_BUFFERS block may precede global constants. No other block—whether a block identifier (LAYER, TEST, STATE, CONDITIONS, or ACTIONS) is entered or implied—may be placed before a global constant.

(B) Layer Constants

A layer constant must be defined before the first reference to it. The definition is placed in the lines following the LAYER identifier. (In a single-layer program, the LAYER identifier may be omitted.)

The definition of a layer constant must fall outside component blocks of the layer (outside of tests and states).

A layer constant can be referenced within any test, state, or trigger which that layer contains. It may also be referenced in any other layer which follows on the spreadsheet. The only exception to this is when the constant is superseded by a constant of the same name (see the section on precedence which follows).

(C) Test Constants

A test constant must be defined at the beginning of a test block and before the first reference to the constant. While the TEST identifier may be absent in a single-layer, single-test program, the scope of a constant can only be limited to a test if it follows a TEST identifier.

A test constant cannot be defined within a state, but it can be referenced by any trigger in any state which the test contains.

28.4 Referencing Constants

Whenever you refer to a constant in your spreadsheet program, the constant name must be enclosed in double parentheses—for example, ((Frmsize)). Use the key sequence [CTRL]-[9] and [CTRL]-[0] to create double parentheses. Shown here is the constant ADDRESS which replaces an SDLC frame address used throughout the test. When the frame address is modified, only the constant need be changed.

```
LAYER: 2
  TEST: polling
    CONSTANTS:
     ADDRESS = "C1"
    STATE: Init
      CONDITIONS: ENTER_STATE
      ACTIONS: SEND SNRM  ADR= ((ADDRESS)) P/F= 1
        TIMEOUT retransm RESTART 3.000
      CONDITIONS: RCV UA  ADR= ((ADDRESS)) P/F= 1
      ACTIONS: TIMEOUT retransm STOP
        RESET NR RESET NS
      NEXT_STATE: Info_xfr
      CONDITIONS: TIMEOUT retransm
      NEXT_STATE: Init
```

As long as syntax is observed, a constant may be used to replace a large block of text which would otherwise be repeated. Following is an example of a long, repetitive text block given as a constant definition and referenced within the program as ((LK_SETUP)). Notice that the constant definition is contained in a single logical line. The highlighted plus symbols, automatically generated by the spreadsheet editor, indicate the point at which the line wraps on the screen.

```
LAYER: 2
CONSTANTS:
LK_SETUP  = "ACTIONS: SEND  DISC  PROMPT \"Disconnect  link\"  CONDI+
TIONS:  RCV UA ACTIONS:  PROMPT \"Disconnected\"  CONDITIONS:  RCV +
DISC ACTIONS: SEND UA CONDITIONS: RCV SABM ACTIONS: SEND UA PRO+
MPT \"Link  restarted\""

  TEST: link_up
    STATE:  fr_setup
      CONDITIONS: ENTER_STATE
      ((LK_SETUP))
    NEXT_STATE: fr
    STATE: fr
      CONDITIONS: ENTER_STATE
      ACTIONS: SEND INFO NR= 01
      CONDITIONS: RCV FRMR
      ACTIONS: PROMPT "FRMR received-test OK."
```

NOTE: Global and layer constants declared on the Protocol Spreadsheet may be referenced on any of the Trigger Setup screens as part of a receive or transmit string.

28.5 Nested Constants

The definition of a constant may include a reference to another constant. This is called nesting. An example of nested constants is shown below. On the Protocol Spreadsheet, it is possible to nest constants eight levels deep.

```
CONSTANTS:
send_lcn    = "000"
rcv_lcn     = "001"
send_data   = "data ((send_lcn))"
send_pkt    = "send ((send_data))"
```

NOTE: It is illegal to define two constants circularly. If, for example, you define CONSTANTS: peat = ((repeat)) and CONSTANTS: repeat = ((peat)), you will receive an error message when you attempt to run the program.

28.6 Precedence

Programming practice usually restricts constants to a single definition. A given name should remain the same throughout the entire program.

In some special cases a constant name may have definitions that differ in separate parts of the test. It is not legal to define the same constant name twice at the same level within the same block; however, the same constant name can be defined differently inside of distinct blocks. You might, for example, define a global constant as maxlength = "8" at the beginning of a program. Nothing prevents you from defining a constant as maxlength = "128" *within* a layer or test included in the same program.

NOTE: Use the ability to give different definitions to the same constant name sparingly and with great caution.

The rule of thumb is this: When the same constant name is defined more than once, the value of the constant is controlled by the smallest block in which it resides. When that block ends, its value is controlled by the next larger block, and so on. So, a constant might have different values within a TEST, within a LAYER, and throughout the remainder of the program.

Consult the following example. Globally, the constant maxlength has a value of 8. This value holds until the constant takes on a new value in Layer 2, where it is defined as **maxlength = "128"**. Inside the Layer 2 test named **shortfrm**, maxlength is briefly given a value of 4. In Layer 3, the constant maxlength is not redefined, and its value returns to 8 (since this is the global definition of the constant).

```
CONSTANTS:
      maxlength = "8"
LAYER: 2
CONSTANTS:
      maxlength = "128"
   TEST: shortfrm
CONSTANTS:
         maxlength = "4"
         STATE: supfrm
            CONDITIONS:

.

.

LAYER: 3
      TEST: pktlen
         STATE: datapkts
            CONDITIONS:

.

.

.
```

28.7 Expansion

The spreadsheet editor checks constant definitions and references for several types of errors as you enter your program. In the interest of time, however, it will not expand a reference to a constant embedded in a text string. This means that nested constants are not checked for errors as you write your program.

The compiler expands these constants when you run the program, and any obvious errors will result in an operator message. Be advised, however, that it is possible for embedded constants, once expanded, to produce a valid, but unintentional, program variation.

29 Editor

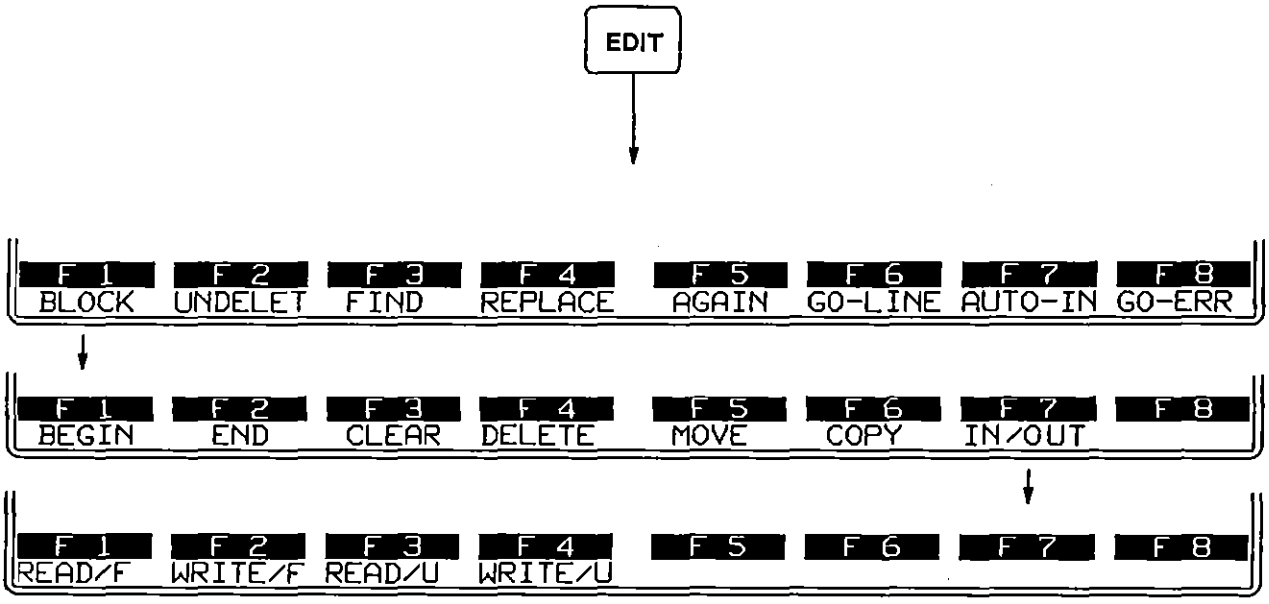


Figure 29-1 Press the *EDIT* key to access special editing functions on the Protocol Spreadsheet. Press *F1* to access additional editing functions.

29 Editor

As you create a spreadsheet program, you may use any of the keys on the editing keypad to modify your entries. Sophisticated editing options are added to these basic functions when you press [EDIT] (see Figure 29-1). [EDIT] is an alternate action key which returns you to program function keys if you press it a second time.

29.1 Basic Editing Functions

Use the editing keypad on the right of the keyboard to perform simple editing functions (see Figure 29-2).

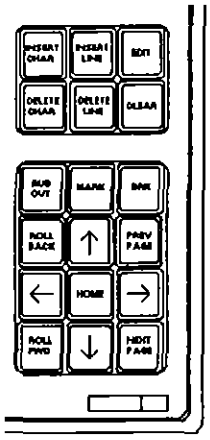


Figure 29-2 The editing keypad.

(A) Insert and Delete Keys

The top three rows of the keypad contain insert and delete functions. Available functions are insert a character, delete a character, rubout a character and insert, delete, or clear a line.

Insert Line and Delete Line functions apply to the logical line, not the physical line. A logical line has segments which end at the end of the screen but are not terminated with a [RETURN]. Instead, the logical line wraps to the next line or lines on the screen. You can distinguish a logical (wrapped) line by the highlighted plus symbols (+) at the end of each segment on the screen. When you insert a line, it appears above the first segment of the wrapped line. When you delete a logical line, all of its segments are deleted.

1. [INSERT CHAR] is an alternate action key. Press it once to enter Insert mode (the label <Insert> appears at the top left of the screen). Then type a character. The character is entered at the cursor position. All text moves right. Continue to insert characters as needed.

   Press [INSERT CHAR] again to leave Insert mode. Any character you type subsequently will overwrite an existing character at the cursor location.

2. [INSERT LINE] inserts a blank line above the logical line where the cursor is located. It also puts you into <Insert> character mode. Use [INSERT CHAR] as described in the previous paragraph to *exit* <Insert> mode.

3. [DELETE CHAR] deletes the character under the cursor. The next character to the right moves under the cursor, and remaining text shifts left.

4. [DELETE LINE] removes the logical line that the cursor is on.

5. [CLEAR] erases the remainder of the logical line from the cursor position and to the right, leaving the line empty. [SHIFT]-[CLEAR] erases the entire logical line which the cursor is on, but not the space the line occupies.

6. [BACK SPACE] deletes the character just to the left of the cursor and moves the cursor left one space. Use [BACK SPACE] to correct an error in the most recent keystrokes.

(B) Cursor and Movement Keys

1. [↑] and [↓] move up or down the screen one line at a time. [SHIFT]-[↑] moves the cursor to the first line of the file. [SHIFT]-[↓] moves the cursor to the last line of the file.

2. [←] and [→] move the cursor to the right or left one space at a time. [SHIFT]-[→] moves the cursor forward to the beginning of the next field. [SHIFT]-[←] moves the cursor back to the beginning of the previous field. [CTRL]-[→] moves the cursor forward to the end of the current line. [CTRL]-[←] moves the cursor back to the beginning of the current line.

3. [HOME] moves the cursor to the top left-hand corner of the current screen.

4. [ROLL BACK] leaves the cursor where it is and moves text down one line at a time.

5. [ROLL FWD] moves text up one line at a time, without changing the cursor location.

6. [PREV PAGE] recalls the previous screen of text and locates the cursor at the same relative position on the screen.

7. [NEXT PAGE] moves the cursor to the same relative position on the next screen of text.

(C) Other Keys on the Pad

1. [MARK] provides a means for "saving a place" in a program file. With the cursor at a desired location, press [MARK], then any number from 0 through 9. This marks the column and row in memory (no mark actually appears). At any time, you may locate one of ten possible marks in the file. Press [SHIFT]-[MARK], then the desired number to move forward or back through the file to the desired location.

2. [MAX] is not currently implemented.

29.2 Editing Function Keys

The editing keys shown in Figure 29-1 appear when you press [EDIT] with the Protocol Spreadsheet displayed. Press [EDIT] again to move back to the program function keys. When you press [F1] for BLOCK, a subset of editing options appear. Press [DONE] to move from this subset back to the top level of editing functions.

(A) Block Functions

With the regular spreadsheet programming selections displayed, press [EDIT], then BLOCK ([F1]) to display the block commands. Six editing commands (keys [F3] through [F8]) operate on blocks of text. When you are using a Clear, Delete, Move, Copy or Write command, you must mark the beginning and end of a block prior to executing the command.

1. *Begin and End.* Use BEGIN to mark the first character of a block at the cursor location. Move the cursor one position to the right of the last character you want to include in the block. Then press END. The block, once defined, is highlighted. Whenever a block is highlighted, you may clear, delete, move, or copy the block or write it to another file.

   NOTE: The block may be defined in the reverse direction. The cursor must be located one position to the right of the first character of the block and located over the last character of the block.

2. *Clear.* Press [F3] for CLEAR to "unmark" a block. The highlighting disappears to indicate that there is no longer an active editing block.

3. *Delete.* Press [F4] for DELETE to remove the marked block. Text below the block fills in the deleted area.

NOTE: You may recover a deleted block using the Undelete command on the alternate set of editing function keys. Repeated use of the Undelete command will recover up to ten deleted blocks. The text is recovered in the reverse order in which it was deleted—i.e., last deleted, first recovered.

4. *Move.* To move text, define a block and locate the cursor at the position where you want the text block to start. Then press MOVE. The text is removed from its original location and is inserted at the cursor location. The moved text remains highlighted as a block.

   To retain the original line breaks in the text, insert a blank line at the position where the new text will be located. Otherwise, inserted text will be placed at the beginning of the line marked by the cursor.

5. *Copy.* To copy text, mark a block, move the cursor to the desired location, and press COPY. A duplicate of the text block appears, highlighted. Since the block is already marked, you may copy it repeatedly without remarking it.

6. *In/Out.* To access the four Read/Write options, press the function key marked IN/OUT. A new rack of function keys appears (see Figure 29-1). These functions are explained in Section 29.2(B).

(B) Read and Write

The READ and WRITE commands are block commands but are exceptions in that they allow you to move text into and out of your program file. You can use a READ command as you would a load command to call in other Protocol Spreadsheet files. Likewise, you can save a copy of the Protocol Spreadsheet using a WRITE command.

The four command options on this rack of function keys are Read Formatted, Read Unformatted, Write Formatted, and Write Unformatted.

1. *Formatted Read and Write commands.* Read Formatted and Write Formatted are intended for use with spreadsheet files and any other files which contain non-printable (non-ASCII) characters:

   - Special characters such as bit masks, F, G, B, A, X, «, and »
   - Any control characters outside the limited subset listed in the following paragraphs for unformatted Read and Write commands
   - "Packed" hex characters; that is, hex characters as they appear on the screen (for example $^{7}_{E}$, $^{3}_{F}$, and $^{4}_{0}$).

The Write Formatted command saves these non-printable characters as expanded ASCII and uses pound signs (#) and backslashes (\) as prefixes to mark their location for later decoding. Thus, when a file is written, # becomes ##, \ becomes \\, while ³₀ becomes #30, ▣ becomes \7E, and so on.

The Read Formatted command decodes the expanded representations properly and displays them as they previously appeared on the Protocol Spreadsheet. If by mistake you use the Read Formatted command on a pure ASCII file which contains backslashes or pound signs, the INTERVIEW will attempt to decode the characters which immediately follow. For example, a preprocessor directive from an #include file such as

#define max 5

will be decoded as

°ᴇfine max 5—which obviously cannot be interpreted by the preprocessor.

2. *Unformatted Read and Write commands.* Read Unformatted and Write Unformatted are intended for use with #include files and other pure ASCII files. Any files that contain only ASCII and a limited subset of control characters may be successfully read in or written to disk with these commands. The set of control characters which are recognized and retained by these commands follows:

   - Tab (ᴴ)
   - Form Feed (ᶠᶠ)
   - Carriage Return (ᶜᴿ)
   - Bell (ᴮᴸ)
   - Line Feed (ᴸᶠ)

   Any other control characters are stripped from the file when one of these commands is used—as are packed hex characters (³ꜰ, ⁴₀, and so on) and special characters.

   NOTES:

   a. If you mistakenly use a Write Unformatted command on a file which contains non-printable characters, these characters will be stripped from the file without warning.

b. Since no messages inform you of whether file contents are formatted or unformatted when you perform a Read or Write, you should keep track of the file type for later reference. An easy way to do this is to append a suffix (such as *_u* for unformatted or *_f* for formatted) to the filename. *#include* files, which end with the *suffix .h*, require the Read Unformatted and Write Unformatted commands.

3. *How to execute a Read command.* To copy an existing file into the Protocol Spreadsheet, place the cursor at the location where you want the file to start. Press READ/F or READ/U, whichever is appropriate (see previous paragraphs), and type in the exact filename (full or relative pathname). Then press [XEQ] or [RETURN]. The entire file is highlighted and copied at the cursor location. Any original spreadsheet text beyond the cursor position is pushed to the end of the file which has been read in.

   NOTES:

   a. When giving the filename to be read, provide the location of the file by disk. If the destination disk is omitted, only that one named in the current directory on the File Maintenance screen will be searched. If the file is located on the current directory disk, it will be read; otherwise, an error message will appear at the top of the screen.

   b. You may read in an entire spreadsheet file without affecting the configuration of other menus in the INTERVIEW. A full program, containing the spreadsheet and the contents of all other menus, must be loaded from the File Maintenance screen. See Section 14.3(E).

4. *How to execute a Write command.* You may file a copy of all or part of your spreadsheet entries using one of the Write commands. First, mark the beginning and end of the block you wish to save to a file. Then press WRITE/F or WRITE/U, whichever is appropriate (see previous paragraphs), and give the full or relative pathname of the file when prompted. Press [XEQ] or [RETURN]. The file will appear in the directory listings on the File Maintenance screen. If you type in the name of a file which already exists, your spreadsheet text block will overwrite the entire file.

   NOTE: If you wish to save the configuration of other menus along with your spreadsheet program, use the Save command on the File Maintenance screen; see Section 14.3(F).

(C) Other Editing Commands

To return to the main set of edit keys from the bank of Block commands, press [DONE]. The remaining commands in the set are described in subsequent paragraphs.

1. *Undelete*. You can return the last deleted line or block to the screen. First, locate the cursor where you want the deleted text to appear, and press UNDELET. The deleted text will be inserted at the cursor location. Repeated use of the Undelete command will recover up to ten deleted blocks.

2. *Find*. Press FIND, and the prompt "Find:", along with the cursor, appear at the top of the screen. Type in the string you wish to locate, and press RETURN. The command performs a forward search to the end of the file. Press AGAIN to search for another occurrence of the same string. The message "Text not found" is posted at the top of the screen if the entered text does not occur between the last cursor location and the end of the file.

3. *Replace*. To replace a text string (with a maximum of 50 characters), press REPLACE. The prompt "Find:" appears at the top of the screen. Type in the string that you want to replace, and press XEQ or RETURN. The prompt "Replace with:" appears. Type in the new string, and press XEQ or RETURN. The command searches forward in the file from the cursor position and replaces the first occurrence of the string. To continue replacing the old string, press AGAIN, until the message "Text not found" is displayed at the top of the screen. The search for the text string stops at the end of the file.

   NOTES:

   a. If you want the entire file to be searched, make sure the cursor is positioned at the beginning by pressing SHIFT-↑.

   b. Case does make a difference. If the string "echo" is replaced, "Echo" will not be replaced.

4. *Again*. You may repeat Find and Replace commands by executing the command, then pressing AGAIN.

5. *Go-line*. To move from one line to any other line in the file, press GO-LINE. When prompted, enter the sequential number of the line you want, and press XEQ or RETURN.

6. *Auto-indent*. <Indent> will appear at the top right of the screen when Auto-indent is on. Auto-indent is an alternating function key. If the indent cue does not appear, press the function key once to turn on Auto-indent. Press the function key again to turn off indentation. Auto-indent is active both when editing keys and program function keys are active.

   NOTE: To move through the program one line at a time at the points of indentation, use the RETURN key instead of the ↑ and ↓ keys.

This feature is an aid in setting up spreadsheet programs. When you use a function key to enter a keyword, the keyword appears on a new line, and, if it is a component belonging to a larger block, it is indented. For example, if you press LAYER:, the keyword is not indented, but if you press TEST, the keyword TEST: appears on a new line, indented three spaces from the first letter of its "owner" (LAYER). When you press STATE, the keyword STATE: is indented another three spaces, to show that it is a component of the test.

NOTE: If you type in your spreadsheet entries, the last level of indentation is observed; however, other auto-indent features are not applied to manual entries.

7. *Go-error*. Most syntax errors made on the Protocol Spreadsheet are indicated by strike-through of the text where the error occurs. Press GO-ERR to move to the first editing error found moving forward (down) through the file. Press GO-ERR once more to move to the next editing error. The search for editing errors stops at the end of the file, and the message "No more errors" is displayed at the top of the screen.

   Errors which are detected by the C translator, preprocessor, or compiler are not indicated by the editor. When you press [RUN] and the test is compiled, the errors will be noted. If there are errors in the test, the INTERVIEW will revert to the Protocol Spreadsheet and display a diagnostic message about the first error rather than run the test. Press GO-ERR to search for additional errors until the "No more errors" messages is displayed.

   If you leave the Protocol Spreadsheet to go to another screen, but then want to review the list of the errors again, return to the Main Program menu. Press [F3], [EDIT], [F8] (spreadsheet screen, edit, GO-ERR). Repeat GO-ERR for the next one. When there are no more errors, a prompt to that effect will appear at the top of the screen.

   Error messages are listed in Appendix A.

30 Layer-Independent Conditions and Actions

Condition-and-action triggers are the basic programming elements on the INTERVIEW Protocol Spreadsheet. Triggers can be thought of as "If, Then" statements, organized on the spreadsheet under the headings CONDITIONS and ACTIONS. Each pairing of CONDITIONS and ACTIONS on the spreadsheet represents one trigger, similar to but also more comprehensive than one of the sixteen Trigger Setup screens (see Sections 24 and 25). Any number of triggers may be created in the spreadsheet program.

During a test, a trigger condition is active (potentially true) whenever the state it belongs to is active. An action is taken whenever the condition (or set of conditions) preceding it is true.

This section covers those conditions and actions that are *not* local to a particular protocol at a particular layer of programming. These are the conditions and actions that are made available as softkey selections in every state in the program without exception.

30.1 Naming Requirements

Flags, accumulators, signals, counters, timers, and timeouts are layer-independent trigger entities that are created by the user in any number and combination and called out by *keyword* (FLAG, ACCUMULATE, SIGNAL, COUNTER, TIMER, TIMEOUT) and by *name*. The names are assigned by the user and referenced in triggers throughout the program.

A name on the Protocol Spreadsheet must not exceed sixteen characters nor include any except the fifty-two alpha characters (upper and lower cases) and the ten numeric characters in addition to the underscore (_) character. The first character in each name must be an alpha character.

The practical size limit for the names of counters, timers, and accumulators is eight characters, since a longer name cannot be called out on the tabular and graphic statistics screens.

For the sake of program readability, we recommend that all user-assigned names be entered in lower case. In this way they will be distinguishable from keywords. The spreadsheet compiler does not insist on lower case for user-assigned names, however.

The spreadsheet compiler does treat upper- and lower-case names as distinct. A timer named **delay** will not be referenced by the name **DELAY** (or **Delay**), for example. Keywords are treated differently: typing timer has the same effect as typing Timer or TIMER or pressing the softkey that writes TIMER to the screen.

Names of different entities need not be kept distinct. The program will have no trouble keeping a SIGNAL named **ready** separate from a FLAG of the same name. (The user may have difficulty keeping them separate, however.)

30.2 Rules for Combining Conditions

Several layer-independent conditions are "transitional" (or "instantaneous") conditions, in that they are true only for the *instant* that they transition to true. These transitional conditions are enter-state, timeout, keyboard, time-of-day, and signal conditions. Triggers that combine two transitional conditions are illegal and will not compile, since there is no chance of two transitional events occurring simultaneously.

The other class of layer-independent conditions, comprised of buffer-full, counter, and flag conditions, may be thought of as transitional/status. When used alone in a trigger, these conditions are true only at the moment they transition to true.

For example, the condition COUNTER retries GE 5, used by itself preceding an Actions block, will be true once when the counter increments from 4 to 5, but not when the same counter increments to 6. For the condition ever to be true again, the counter must first transition to a value less than 5.

When used in combination with transitional conditions, these transitional/status conditions are checked for a current *status* of true at the moment the transitional condition transitions true. They may retain this status of true indefinitely.

Here is an example of a transitional/status condition (counter) used in combination with a transitional condition (timeout).

```
CONDITIONS: TIMEOUT response
   COUNTER retries GE 5
ACTIONS: ALARM
```

This set of conditions will be true every time the timeout occurs as long as the counter retains a *status* of greater than or equal to 5.

When a transitional/status condition is used in combination with one or more other transitional/status conditions, the first condition in the user-defined sequence of conditions will be transitional, while the others will be checked for truth or falsity only when the first condition transitions to true. Take, for example, a scenario where a counter increments five times and then a flag increments five times. On the fifth flag increment, the following set of conditions will be true:

```
CONDITIONS: FLAG true_last 101
   COUNTER true_first EQ 5
```

The conditions are satisfied because the flag is transitional while the counter is static: at the moment the flag transitions to binary 101 (decimal 5), the counter is checked for a status of 5. Both are true. But given the same scenario, this set of conditions is false:

```
CONDITIONS: COUNTER true_first EQ 5
   FLAG true_last 101
```

Here, the counter condition is transitional, the flag is static—simply because the counter condition is listed first. The flag condition is checked only at the moment the counter attains the count of 5. After that, the flag is not checked again.

The condition logic is streamlined in this manner in order to be economical of processor time, on the assumption that in a typical application the user knows which of two conditions will be satisfied first. If the user does not know whether the counter or the flag in the above example will increment to 5 first, nothing prevents him from entering two triggers, both having the same conditions but in a different sequence. Or he may enter the pair of conditions on a Trigger Setup menu, where combined transitional/status conditions generate enough code to cover all contingencies. See Section 24.2(B)2.

> NOTE: Additional rules may apply when the COUNTER or FLAG transitional/status condition is used in a spreadsheet program compiled and saved as a linkable-object file. See Section 27.4(B).

30.3 Layer-Independent Conditions

The eight softkeys that represent the full set of layer-independent conditions are shown in Figure 30-1.

(A) Enter State

This condition is true immediately as the current state is entered. Control of the action in effect reverts to the previous state. In the example below, ENTER_STATE is used as the condition for an alarm action in **second** state. The counter condition in first state effectively controls this alarm.

```
STATE: first
   CONDITIONS: COUNTER frm_err EQ 10
   NEXT STATE: second
STATE: second
   CONDITIONS: ENTER_STATE
   ACTIONS: ALARM
```

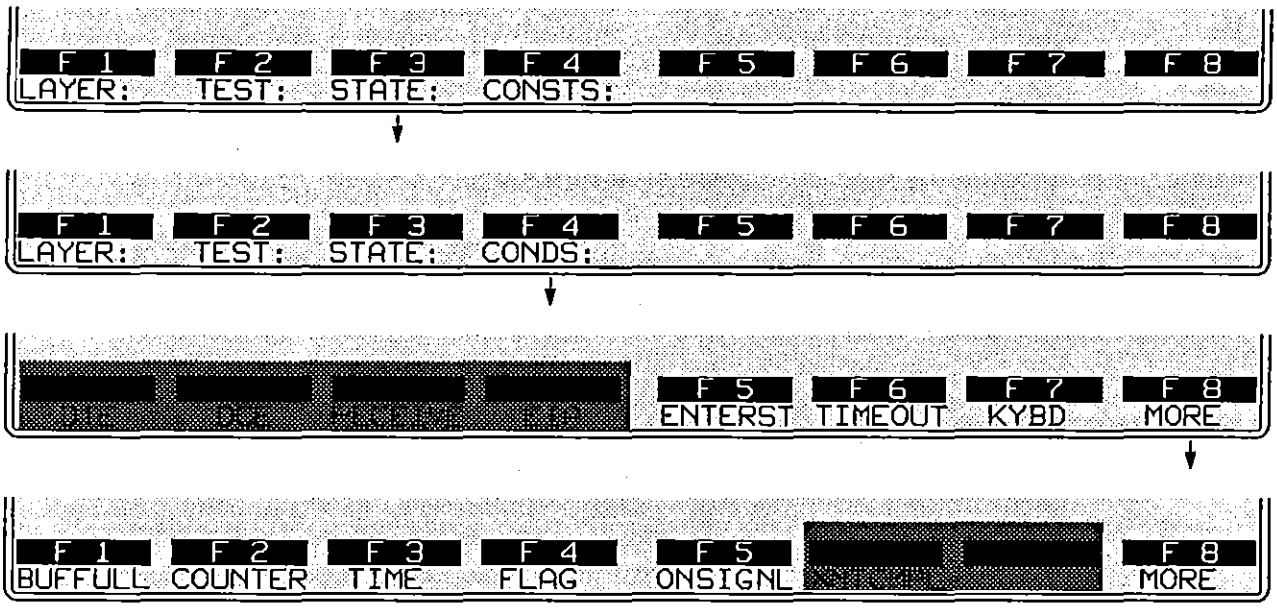


Figure 30-1 The eight layer-independent conditions are shown in the bottom two racks of softkeys.

(B) Timeout

Any number of decrementing timeout timers may be started as trigger actions and monitored by trigger conditions. The condition is true when the timeout timer expires.

Here is an example of a timeout condition:

```
TIMEOUT response
```

where **response** is the name of the timeout timer.

After pressing the TIMEOUT softkey or typing TIMEOUT followed by space, enter a name. The name can reference a timeout timer that was started either in a spreadsheet action or a trigger-menu action.

(C) Keyboard

Enter a list of characters produced by keystrokes. Any key or key-combination that produces a character on the ASCII table in Appendix D1 is valid input in this field. Lists in the spreadsheet program can extend to 128 characters.

In Run mode when any key on the list is pressed, the condition will be true and (if this is the only condition) will initiate a trigger action.

An example of a keyboard condition is the following:

```
CONDITIONS: KEYBOARD "1 "
```

Note the space following the 1 entry. Here the 1 key *or the space bar* will satisfy the trigger condition. Dual quotation marks are required for all lists and strings on the Protocol Spreadsheet.

(D) Buffer Full

This condition is true at the moment the 64-Kbyte character buffer is full. Use this condition to trigger a display-freeze (CAPTURE BOTH OFF) whenever the earliest data in the display buffer is the most important and you do not want it to be overwritten. Here is an example of a trigger that will retain the first full buffer of data:

```
CONDITIONS: BUFFER_FULL
ACTIONS: CAPTURE BOTH OFF
```

(E) Counter

Any counter named and operated as a trigger action may be monitored as a trigger condition. To create a counter condition, press the COUNTER softkey or type COUNTER followed by a space.

> NOTE: A counter named on a Trigger Menu screen also refers to a spreadsheet counter as long as the name matches. Timeouts and timers can also be shared between the Trigger Menu screens and the spreadsheet.

> NOTE ALSO: Trigger Setup screens monitor counter values from 0 to 999,999. However, Protocol Spreadsheet triggers can monitor counter values up to 4,294,967,295.

An example of a spreadsheet counter condition is the following:

```
CONDITIONS: COUNTER byte_no EQ 128
```

where **byte_no** is the name, EQ(ual) is the relational operator, and **128** is the decimal value.

1. *Enter counter name.* Name the counter to be monitored. See Section 30.1, Naming Requirements.

2. *Relational operator.* As soon as a counter name has been typed and followed by a space, a rack of softkeys appears with names of relational operators. See Figure 30-2.

Figure 30-2 A set of relational operators compares the counter value to a user-entered value.

Make the appropriate selection to specify when the counter condition will be true. The counter may be tested for a value equal to (EQ), not equal to (NE), greater than or equal to (GE), less than or equal to (LE), strictly greater than (GT), or strictly less than (LT) the value entered on the spreadsheet.

When a COUNTER condition is used alone, it is a *transitional* condition. This means that it is true only when it transitions to true. For example, a condition that said COUNTER **drops** NE 5 would be true when COUNTER **drops** transitioned from 5 to 6—that is, on the transition from *equal 5* to *not equal 5*; but the condition would not be true when 6 changed to 7.

In combination with another condition (that is, more than one condition *per* action or set of actions), a COUNTER condition normally is a status condition, not a transitional condition. As a status condition, COUNTER **drops** NE 5 is true any time the status of the counter is not 5. Refer to Section 30.2, Rules for Combining Conditions.

NOTE: Additional rules may apply when the COUNTER transitional/status condition is used in a spreadsheet program compiled and saved as a linkable-object file. See Section 27.4(B).

3. *Enter the counter value.* Enter the value as a whole decimal number. Each condition can monitor a 32-bit counter for decimal values ranging from 0 to 4,294,967,295.

NOTE: The Current value for a counter on the Tabular Statistics screen is maintained to seven decimal places, for a maximum counter display of 9,999,999. The 32-bit binary counter can attain much higher values than this, however—the decimal display on the statistics screen merely rolls over to zero and continues counting. Spreadsheet counter conditions can monitor for values up to the maximum of over four billion. If a trigger looks for a counter value higher than this maximum, it will never be satisfied.

(F) Time

The time of day once a day or once a month can satisfy a trigger condition. Here, for example, is a trigger condition that comes true at 3 P.M. each day:

```
CONDITIONS: TIME 1500
```

1. *Enter day of month or time of day.* Press the TIME softkey or type TIME followed by a space. The next entry will signify day of month if it is a two-digit entry. If it is four digits, it will signify the time of day in twenty-four hour format.

2. *Enter time of day.* If the entry following TIME is a two-digit, day-of-month entry, it must be followed by time of day in a four-digit, twenty-four hour format.

(G) Flag

Sixteen internal flag bits are reserved for every flag mask that is named in Protocol Spreadsheet conditions and actions.

> NOTE: The eight flag bits on the Trigger Setup screens are the low-order bits of a flag mask that can be accessed on the Protocol Spreadsheet by the name **trig_flag**.

A flag condition still is valid when fewer than sixteen flag bits are specified. The flag values that are specified are right-justified when the program is compiled, and leading X's (don't cares) are assumed.

The internal flag normally is a static condition when it is used in combination with other trigger conditions—that is, more than one condition per action or set of actions. Refer to Section 30.2, Rules for Combining Conditions. Since flag bits are completely under program control and can be used in combination with other conditions, they are useful chiefly to enable or disable entire triggers.

> NOTE: Additional rules may apply when the FLAG transitional/status condition is used in a spreadsheet program compiled and saved as a linkable-object file. See Section 27.4(B).

For example, a trigger action is taken if a flag bit is 1 and a % character is seen. Setting the flag to zero effectively disables this trigger.

An example of a flag condition is the following:

```
CONDITIONS: FLAG nak 1X
```

where nak is the name of the flag and XXXXXXXXXXXXXX1X is the flag bit mask.

1. *Enter the flag name.* After pressing the FLAG softkey or typing FLAG followed by space, enter a name not exceeding eight characters, beginning with an alpha character.

2. *Enter the flag condition bit mask.* A flag mask follows the flag name. The mask can include up to sixteen bits (with no spaces between them). Since the number of flag masks in your program is unlimited, you may want to restrict your masks to one or two bits. In effect you will be giving each bit or pair of bits a name.

   Legal bit-entries are 1, 0, or X (for "don't care"). Press [X] or [CLEAR] to enter an X. The condition will not test this bit.

(H) On Signal

Signals are communicated between tests and between layers. They are the simplest way to use an event in one test to start a state or an action in another test. Here is an example of an on-signal condition:

```
ON_SIGNAL testfail
```

After pushing the ONSIGNL softkey or typing ON_SIGNAL followed by space, enter the name of a signal you have created (or intend to create) in a trigger action.

30.4 Layer-Independent Actions

When a block of conditions has been entered, press [DONE] to access the ACTIONS softkey. The actions that are available in all states without exception are shown in Figure 30-3 as they appear in three successive racks of softkeys.

(A) Counter

The Protocol Spreadsheet screen can control any number of counters. The Tabular Statistics screen is an expanding display that can provide statistics for 100 counters, timers, and accumulators.

Here is an example of a counter action:

```
ACTIONS: COUNTER datapaks INC
```

1. *Enter counter name.* A counter can be unique to one trigger action or it may be shared with other actions and other triggers, which can monitor it and change its values. As long as the same counter name is used, the same counter is invoked.

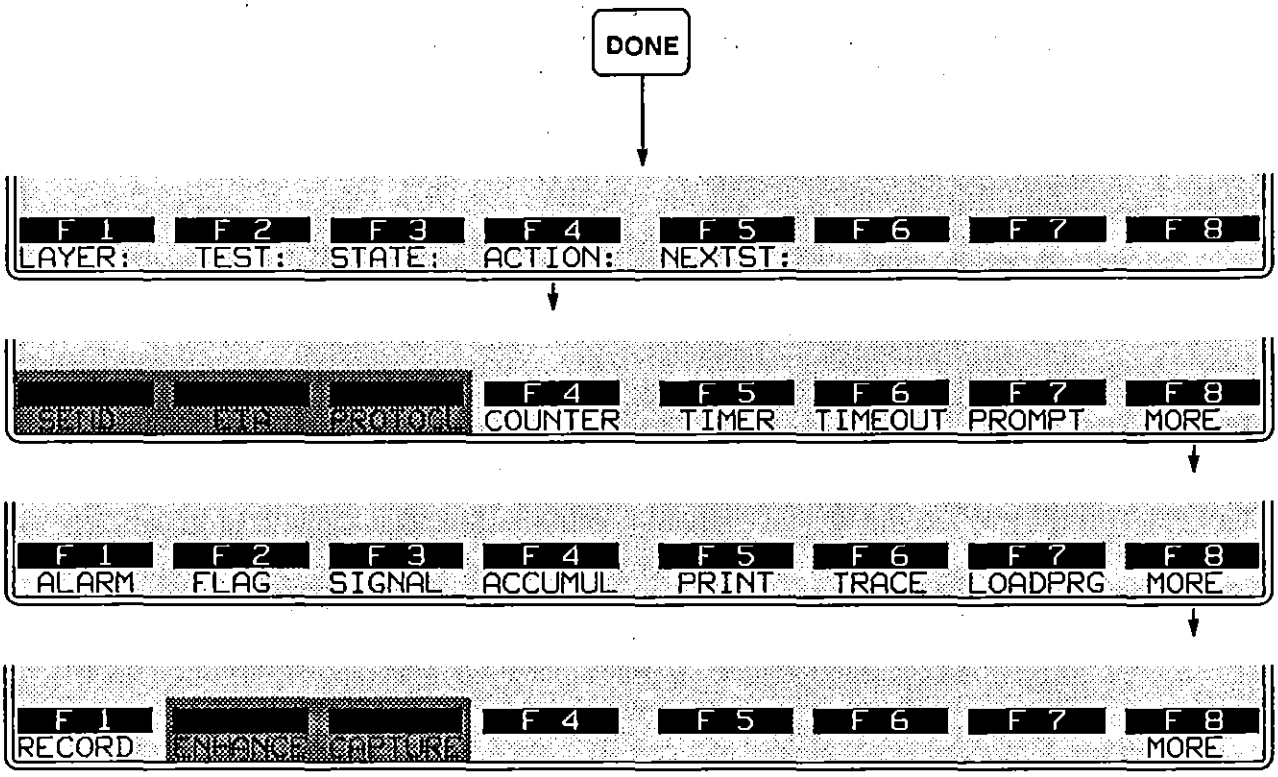


Figure 30-3 The twelve layer-independent actions are spread over the bottom three racks of softkeys in this figure.

NOTE: A counter named on a Trigger Menu screen also refers to a spreadsheet counter as long as the name matches. Timeouts and timers can also be shared between the Trigger Menu screens and the spreadsheet.

NOTE ALSO: Trigger Setup screens monitor counter values from 0 to 999,999. However, Protocol Spreadsheet triggers can monitor counter values up to 4,294,967,295.

After naming the counter, select among the actions shown in the rack of softkeys in Figure 30-4.

Figure 30-4 Counter actions.

2. *Increment.* Thirty-two bits are reserved for each counter. Therefore a counter will roll over after it attains a decimal value of 4,294,967,295.

Spreadsheet conditions can monitor a counter for any value from zero to the maximum. (Trigger Menu conditions can monitor up to a count of 999,999.) Note, however, that the counter value will only be displayed up to seven decimal places on the Tabular Statistics screen. The maximum displayed value therefore is 9,999,999.

3. *Decrement.* When this action is selected, each trigger occurrence subtracts 1 from the counter. A counter that decrements below zero wraps to 4,294,967,295. The last seven decimal places of this maximum value will be displayed in the **Current** column on the Tabular Statistics screen.

4. *Set.* Select SET in order to specify the value that the counter will take when the trigger comes true. Enter a decimal value for the counter. To reset a counter without taking a statistical sample, use the SET action and enter a value of zero.

5. *Sample.* This action stores the current value of the counter and then resets it to zero. The stored value is posted immediately to the statistics display in the **Last** column. This value is compared with previous **Last** values in order to compute **Minimum**, **Maximum**, and **Average** values for statistical display. Refer to Section 20 for a discussion of tabular statistics.

6. *Clear.* This action resets the counter to zero and also resets last, minimum, maximum, and average values for the counter.

(B) Timer

The Protocol Spreadsheet can control any number of timers. The Tabular Statistics screen is an expanding display that can provide statistics for 100 counters, timers, and accumulators.

While timers can be run and sampled as trigger actions, they are not available as trigger conditions. Timeouts, not timers, are the mechanism that allows you to trigger off of elapsed time.

Timer values may be based on an internal "wall" clock, or, if time ticks are enabled on the Front-End Buffer menu screen, on ticks that are stored along with the data. The "tick" mode of timing is the most accurate, especially when data is played back and you do not want playback conditions such as speed and idle-suppression to affect the timers.

Here is an example of a timer action:

```
ACTIONS: TIMER session SAMPLE
```

1. *Enter timer name.* After pressing the TIMER softkey or typing TIMER followed by a space, enter a name. Like counters and timeouts, a timer can be shared between the spreadsheet program and the Trigger Menu screens. If the same name is used, the same timer is invoked.

After naming a timer, select among the actions shown on softkeys in Figure 30-5.

Figure 30-5 Timer actions.

2. *Restart.* Use RESTART to start a timer. This causes the timer to reset to zero and begin incrementing. A restart does not affect any statistical values except **Current.**

3. *Stop.* A stop action suspends the timer at its present value. The timer may be started again at this value by a Continue action on another trigger.

4. *Continue.* This action restarts the timer beginning at the value that was frozen in the **Current** column when the timer was stopped. The Continue action has no effect on a timer that is incrementing already.

5. *Sample.* Sampling a timer resets it at zero and stops it. Prior to resetting, the current value is posted as a **Last** value and passed along for other statistical tabulation.

6. *Clear.* Clearing a timer resets and stops the timer and clears the last, minimum, maximum, and average values.

(C) Timeout

Any number of timeouts can be started and stopped in the spreadsheet program. Timeouts are timers that count down instead of up. Their values are not read on any statistical display; but when they time down to zero, they satisfy trigger conditions that monitor them by name. Timeout timers that are named on the Protocol Spreadsheet also may be monitored and controlled on the Trigger Menu screens.

Here is an example of a timeout action:

```
ACTIONS: TIMEOUT t2 RESTART 3
```

where **t2** is the name of the timeout and **3** is its duration in seconds.

1. *Enter timeout name.* After pressing the TIMEOUT softkey or typing TIMEOUT followed by space, enter the name of the timeout. As soon as a name has been entered and followed by a space, a rack of softkeys appears with the names of two timeout actions, RESTART and STOP.

2. *Restart.* Select RESTART to start the timer running down.

3. *Stop.* Select STOP to halt the timer and prevent the timeout.

4. *Enter timeout value.* The duration of the timeout is entered in seconds. The timeout value is a decimal field in which entries are valid to the millisecond (0.001). For values under 1 second, you must precede the decimal with a leading zero, as follows:

```
TIMEOUT delay RESTART 0.25
```

The maximum timeout entry in this field is 65.535 seconds.

You may expand the maximum timeout with a program such as the following, which produces an alarm every twenty minutes.

```
STATE: twenty_min_alarm
   CONDITIONS: ENTER_STATE
   ACTIONS: TIMEOUT sixtysec RESTART 60
   CONDITIONS: TIMEOUT sixtysec
   ACTIONS: COUNTER minutes INC
            TIMEOUT sixtysec RESTART 60
   CONDITIONS: COUNTER minutes EQ 20
   ACTIONS: COUNTER minutes SET 0
            ALARM
```

(D) Prompt

Prompts are user-entered ASCII messages that appear on the second status line at the top of the screen in Run mode as a result of a trigger coming true. They are messages to the operator from the program, alerting him to important protocol or program events. Prompts are written to the second status line of any current Run-mode display screen. Switching to Freeze mode or to another display screen clears the prompt from all screens except the Display Window.

> NOTE: The prompt line is not zeroed out with each new prompt, and prompts are overwritten only to the extent of the new prompt. For example, the prompt "POLL" does not completely overwrite the prompt "SELECT"—the result will be "POLLCT." It is a good practice to establish a uniform prompt length and space-fill shorter prompts to that length.
>
> Special C functions that position the cursor anywhere on the prompt line (or elsewhere in the display) and write messages to the cursor position are discussed in Section 64.

A prompt that has been triggered in Run mode is illustrated in Figure 30-6. Here is the same prompt as it appears on the Protocol Spreadsheet:

```
ACTIONS: PROMPT "Echoed message received"
```

Backslashes and double-quotation marks may be included in prompt messages if they are preceded by backslashes, in accordance with the rules for entering these characters in transmit strings. See Table 32-2. Example:

```
ACTIONS: PROMPT "\"hello\" string received"
```

1. *Enter prompt message.* After pressing the PROMPT softkey or typing PROMPT followed by a space, enter a message in quotation marks. The message should not exceed 64 characters, the width of the screen.

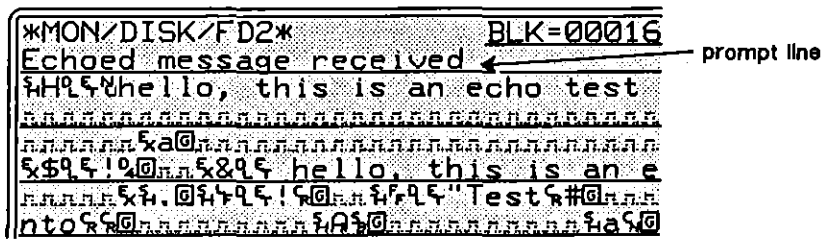


Figure 30-6 User-defined prompts are displayed at the top of the Run screen.

(E) Alarm

The alarm is a tone of less than a second duration. The alarm is sounded each time the trigger comes true. If the tone lasts longer than a second, the alarm has been triggered more than once.

The alarm action on the spreadsheet is simply the word ALARM.

(F) Flag

Internal flags are special programming bits. They can be set on or off by triggers and sensed by triggers. Flags come in masks of up to sixteen bits. Each flag mask is named and referenced by the spreadsheet program.

Any number of flag masks may be created. Flags are common to all tests and layers: if a flag name is used in tests in two different layers, it refers to the same sixteen programming bits.

A flag action still is valid when fewer than sixteen flag bits are specified. The flag values that are specified are right-justified when program is compiled, and leading X's (don't cares) are assumed.

> NOTE: The eight flag bits on the Trigger Setup screens are the low-order bits of a flag mask that can be accessed on the Protocol Spreadsheet by the name **trig_flag**.

Here is an example of a flag action:

```
ACTIONS: FLAG nak SET 0X
```

where nak is the name of the flag, SET is the action, and **0** is the only bit in the mask affected by the set action.

1. *Enter the flag name.* After pressing the softkey for FLAG followed by a space, a rack of softkeys appears with the names of flag actions. See Figure 30-7.

Figure 30-7 Flag actions.

2. *Increment.* The mask can be used as a sixteen-bit binary counter. The INC action increases the value of the mask by one each time the trigger is true.

   As the mask increments above 65,535, it wraps to zero.

   The INC action always toggles the least significant flag bit. If you monitor the flag for only one bit (for example, FLAG flagname 1), the INC action will toggle the condition true/false. This can be a useful tool when you want *every second occurrence* of an event to trigger an action.

3. *Decrement.* This action decreases the value of the flag byte by one each time the trigger is true. When the mask decrements below zero, it wraps to 65,535.

4. *Set.* This action rewrites the flag bits according to the flag-action bit mask that you enter following the SET keyword. The bit mask is comprised of up to sixteen 0's, 1's, and X's (no change).

   When you enter fewer than sixteen bits, you are leaving the leftmost bits in the mask unspecified. The action will not change the condition of unspecified bits.

(G) Signal

Use signals to convey instructions to other tests and other layers where conditions are monitoring these signals by name.

Other internal programming mechanisms, such as flags and counters, are common to all tests and layers and may perform a signaling function. Signals, however, are more efficient in that they are reusable: a signal that is sent and monitored can be sent and monitored again ten seconds later, but an action that sets a flag to 1 cannot be used again until another action has intervened to reset the flag to zero.

After pressing the SIGNAL softkey or typing SIGNAL followed by space, enter the name of the signal. Often the name will be descriptive of the event being signaled. An example of a signal action is the following:

```
ACTIONS: SIGNAL testfail
```

(H) Accumulate

The accumulate action reads a specified value for a counter or timer and presents this value to tabular and graphic statistics screens for statistical breakout. This action is distinct from the sampling action of a counter or timer in this important respect: sampling a counter or timer also resets it to zero. Accumulating a counter or timer has no effect on the ongoing counting or timing function. Examples of accumulators are given in Section 20, Tabular Statistics.

Values for more than one counter or timer may be brought into a single accumulator. For example, separate timers might measure response times for a group of multidropped DTEs. At the end of the test, a value for each timer could be brought, in separate trigger actions, into one accumulator.

Each accumulate action specifies one value only for a counter or timer. Thus the accumulator might provide meaningful statistical data based, for example, on maximum values only for a group of timers.

Here is an example of an accumulate action:

```
ACTIONS: ACCUMULATE alldrop COUNTER badbcc_a LAST
```

where **alldrop** is the name of the accumulator, **badbcc_a** is the name of a counter, and it is the last value of the counter that is being accumulated.

1. *Enter the accumulator name.* Both the accumulator and one counter or timer are referenced in the accumulate action. Counters and timers are referenced, not created, in accumulate actions.

   An accumulator is created by being named in an accumulate action. Like counters and timers, accumulators can be given display lines on either or both of the statistics screens.

2. *Clear.* This action clears the last, minimum, maximum, and average values of the accumulator. (Since accumulators neither count nor time, they never display a current value.)

3. *Counter.* This action accumulates a value for the counter named immediately following the keyword COUNTER. After the counter is named, one value for that counter is selected from the rack of softkeys in Figure 30-8.

4. *Timer.* This action accumulates a value for the timer named immediately following the keyword TIMER. After the timer is named, one value for that timer is selected from the rack of softkeys in Figure 30-8.

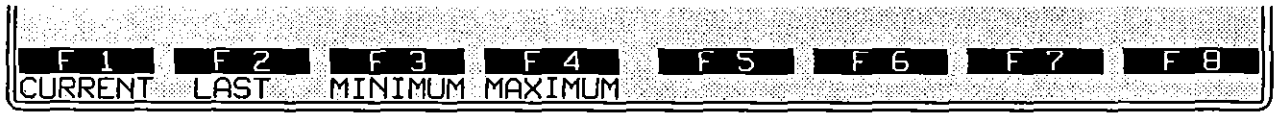


Figure 30-8 Counters and timers are accumulated with respect to one statistical value only.

(I) Print

Time-stamped printouts of single lines of data can be commanded by the spreadsheet program. The data can be a line of tabular statistics for an accumulator, counter, or timer; or a user-prompt that is sent to the printer after it has been written to the second line of the screen.

An example of a print action is the following:

ACTIONS: PRINT TIMER echotime MILLISECONDS

After pressing the softkey for PRINT or typing PRINT followed by a space, select an option for the type of data to be printed from the new rack of softkeys shown in Figure 30-9.

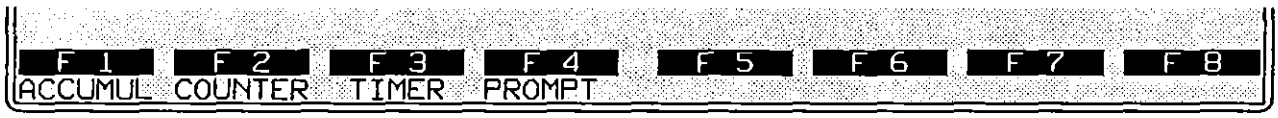


Figure 30-9 Four types of data may be printed out as a trigger action.

1. *Accumulator.* When this action is taken, the line of tabular statistics for the accumulator that you name will be printed. A line of tabular statistics includes last, minimum, maximum, and average values for an accumulator. Since accumulators neither count nor time, they never display a current value.

2. *Counter.* When this action is taken, the line of tabular statistics for the counter that you name will be printed. A line of tabular statistics includes current, last, minimum, maximum, and average values for a counter.

3. *Timer.* After the timer is named, a timer rate is selected from a new rack of softkeys as shown in Figure 30-10.

Figure 30-10 After a timer is named for printout display, a new softkey rack allows you to specify unit of time.

The selected rate will only display values to the smallest place value afforded by the tick rate selected on the FEB Setup screen. For example, if *milli*seconds is selected on the FEB screen, choosing *micro*seconds on the print-timer softkey selection will simply display three additional zeros as place holders—it will not calculate a more precise reading. Thus the most accurate selection for this example would be milliseconds, matching the FEB selection.

When a timer is controlled by a nondata event such as a keyboard condition, it will reference a "wall-time" clock whose smallest resolution is a millisecond.

When the PRINT TIMER action is taken, the line of tabular statistics for the timer that you name will be printed. A line of tabular statistics includes current, last, minimum, maximum, and average values for a timer. Figure 30-11 is an example of such a printout for the program given below.

```
STATE: message
   CONDITIONS: DTE STRING "hello"
   ACTIONS: PROMPT "String sent by DTE"
      TIMER echotime RESTART
   NEXT_STATE: echo
STATE: echo
   CONDITIONS: DCE STRING "hello"
   ACTIONS: PROMPT "Same string by DCE"
      TIMER echotime STOP
      TIMER echotime SAMPLE
      PRINT TIMER echotime MILLISECONDS
   NEXT_STATE: message
```

```
Time          Name       Current    Last Minimum Maximum    Average Unit

09/29 16:13 echotime           0     452     452     452     452.00 MSECS
09/29 16:13 echotime           0     341     341     452     396.50 MSECS
09/29 16:13 echotime           0     428     341     452     407.00 MSECS
```

Figure 30-11 In this printout, a PRINT TIMER action has been triggered three times.

4. *Prompt.* The PRINT PROMPT action is designed to be added to an action block that already contains a prompt. The example below inserts PRINT PROMPT actions into the program described in the previous section. The user does not have to key in a long prompt message twice, once for the printout and once for the screen. The printout for this program is shown in Figure 30-12.

```
STATE: message
   CONDITIONS: DTE STRING "hello"
   ACTIONS: PROMPT "String sent by DTE"
      PRINT PROMPT
      TIMER echotime RESTART
   NEXT_STATE: echo
STATE: echo
   CONDITIONS: DCE STRING "hello"
   ACTIONS: PROMPT "Same string by DCE"
      TIMER echotime STOP
      PRINT PROMPT
      TIMER echotime SAMPLE
      PRINT TIMER echotime MILLISECONDS
   NEXT_STATE: message
```

```
Time          Name          Current    Last Minimum Maximum    Average Unit

09/29 16:13 String sent by DTE
09/29 16:13 Same string by DCE
09/29 16:13 echotime              0     452     452     452     452.00 MSECS
09/29 16:13 String sent by DTE
09/29 16:13 Same string by DCE
09/29 16:13 echotime              0     341     341     452     396.50 MSECS
09/29 16:13 String sent by DTE
09/29 16:13 Same string by DCE
09/29 16:13 echotime              0     428     341     452     407.00 MSECS
```

Figure 30-12 Printout resulting from a combination of PRINT PROMPT and PRINT TIMER actions.

NOTE: If you want to print multiple prompts, place each PROMPT and PRINT PROMPT pair in its own conditions/actions block. (Otherwise, only one prompt will be printed since prompts overwrite each other.)

(J) Trace

Traces are user-entered ASCII data strings, identical to prompts in all ways except in their mode of display: traces are posted one to a line in the multiline Program Trace display (see Section 6.6), while prompts appear on the second status line in all data-display modes (including the Program Trace).

Numerous layers and numerous tests per layer can be active concurrently in the INTERVIEW. The Program Trace can be set up to track state-to-state movement only in a particular **Layer** and **Test** identified by the operator on the Display Setup menu. State names can be included in the Program Trace via the **Display States:** YES selection on the the Display Setup menu. See Figure 30-13.

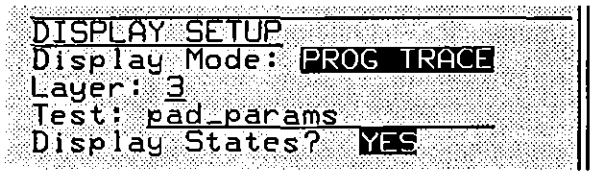


Figure 30-13 The user may select a particular layer and test for a Program Trace.

Traces are debugging tools. Inside a dead-end state they can inform you whether a particular condition that you are expecting is coming true. Prompts, by contrast, have a much fainter "trail": it is hard to be certain that a prompt was not activated and then overwritten by another prompt.

Traces also allow you to keep a record of selected protocol events—to design your own protocol analysis. Since they are written to consecutive lines rather than overwritten by other traces, they are highly useful when you are trying to track protocol events that occur in quick succession.

An example of a trace action is the following:

```
ACTIONS: TRACE " Network congestion"
```

1. *Enter trace message.* After pressing the TRACE softkey or typing TRACE followed by a space, enter a message in quotation marks.

(K) Load Program

A program (source code or object code) or setup that is stored in a file on hard disk or on a disk in either of the microfloppy drives can be loaded in by trigger action. This Load Program function is a means of chaining tests together.

Program files are a full set of configured menus, including the Layer Setup screen, Trigger Setup screens, and the Protocol Spreadsheet. Object files are the precompiled object-code versions of programs. Setup files are a set of configured menus which excludes trigger setups, the Layer Setup screen, and the spreadsheet. Remember that loading a program or setup file overwrites the program or setup file already in memory. Loading an object file overwrites only the object code of whatever program (if any) was compiled most recently. The new object file will not affect the data on any setup menu or programming screen.

EIA statuses can be maintained in between programs by a special menu selection on the Interface Control menu screen. (See Section 12.)

An example of a Load Program action is the following:

```
ACTIONS: LOAD_PROGRAM "FD1/usr/sna/sna_bind"
```

where *FD1* is microfloppy-diskette drive 1, the first slash (/) is the root directory, *usr* is the highest level of user-created files, *sna* is another directory, and *sna_bind* is the filename.

1. *Enter program name.* Enter the absolute pathname of your file. Put the name in quotation marks.

(L) Record

Use the RECORD action to activate or suspend line-data recording or disk-data playback. When the Line Setup menu is configured to monitor a disk, RECORD controls playback; otherwise it applies to recording. There are two selections under RECORD. Select ON to activate, or OFF to suspend, recording or playback.

During recording, the top status line of Run-mode screens will show incrementing block numbers and an "R" displayed in the record/playback field. During playback, a "P" is displayed. Whenever recording or playback has been suspended, an "S" is displayed.

For data playback, the status field will be blank if a disk is not present in the selected drive or when the end of the data-acquisition tracks are reached. This field will also be blank if you enter a starting block number on the Line Setup menu that a) precedes the block number at which data actually begins, or b) exceeds the block number at which data actually ends. Change your entry to zero.

For data recording, the status field will be blank when the end of RAM or the data-acquisition tracks is reached. It will also be blank if the **Capture Memory** field indicates that you will record to disk, but no disk is present in the selected drive or data-acquisition tracks are not available on the disk.

31 Layer 1 Conditions and Actions

There are seven protocol layers in the OSI (Open Systems Interconnect) model that is adopted in the INTERVIEW 7000 Series. Each layer reserves a distinctive set of trigger conditions and actions on the Protocol Spreadsheet.

As a rule, spreadsheet components for a given layer are loaded from disk via the Layer Setup screen. Layer 1, the Physical Layer, is an exception to this rule. Layer 1 conditions and actions are enabled on the Protocol Spreadsheet when the unit powers up.

Depending on the Test Interface Module (TIM) installed in the unit, the power-up also enables an Interface Control Menu screen, different for each module, that controls many Layer 1 parameters. For this reason, the set of Layer 1 conditions and actions is relatively small.

31.1 Layer 1 Conditions

To bring up the bank of softkey conditions for Layer 1, first press the CONDITIONS softkey. This key becomes available when the cursor enters a programming block at the state level.

The first four condition softkeys—DTE, DCE, RECEIVE and EIA—belong to Layer 1. These are followed by generic conditions discussed in the previous section. The set of Layer 1 conditions is shown in Figure 31-1. The softkey for a fifth Layer 1 condition—XMIT_COMPLETE—appears on the second rack of condition softkeys shown at the bottom of the figure.

EIA is a transitional/status condition and may be combined with other conditions. The other Layer 1 conditions are transitional only. Refer to Section 30.2 for a discussion of how conditions may be combined.

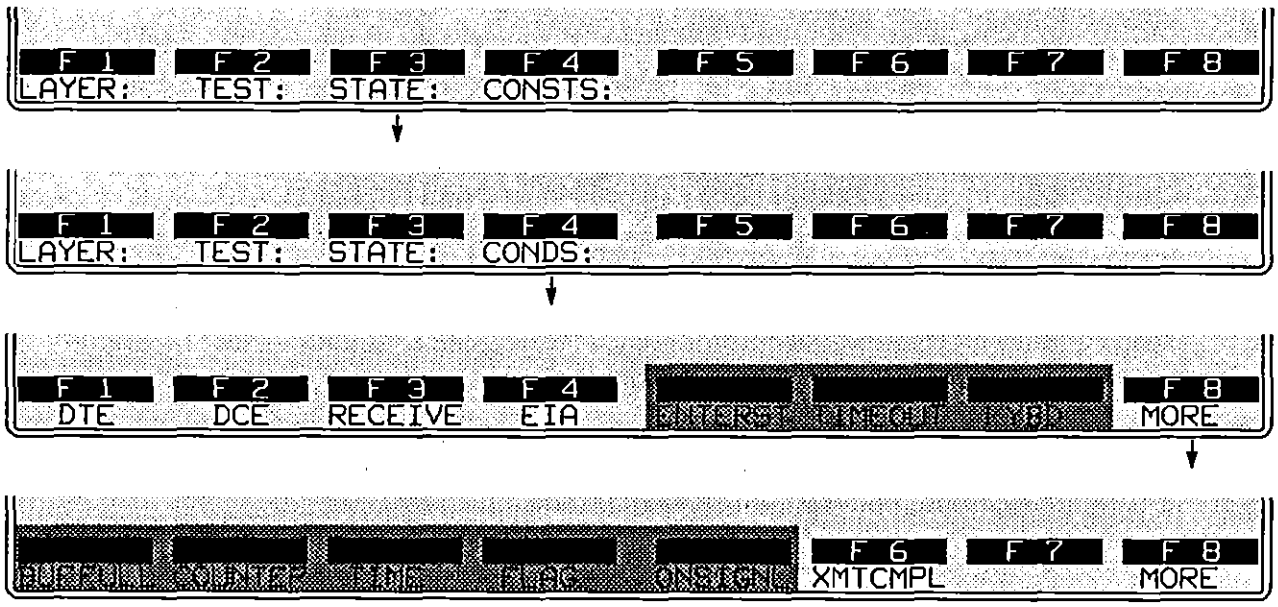


Figure 31-1 Layer 1 conditions.

(A) Data

The first three trigger conditions at Layer 1 can monitor one of the two data leads for a specific data event. This event can be any of several characters, a string of characters, a good BCC following the character or string, an error revealed by a block or parity check, and so on.

Data conditions at Layer 1 monitor the entire data stream. Conditions in other layers also check the data leads, of course, but conditions at Layer 2 and higher look for protocol events.

In searching the data stream byte by byte, Layer 1 data conditions behave similarly to **Receiver** conditions on the Trigger Setup screens. This is another way of saying that the sixteen trigger menus constitute a Layer 1 test. This test has a single state that is always current. Trigger menus with selections made on them are always active.

The three data conditions are DTE, DCE, and RECEIVE. When one of these conditions is selected, a new rack of softkeys appears. The new options are shown in Figure 31-2.

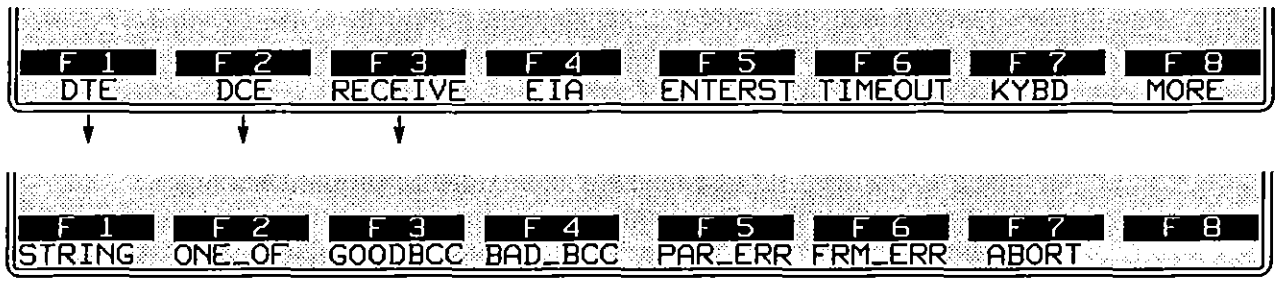


Figure 31-2 A spreadsheet trigger will monitor either data lead looking for these events.

(B) DTE

When DTE is selected, data on the TD lead will be monitored.

(C) DCE

This condition monitors the RD lead.

(D) Receive

This condition is intended for use in the emulate modes. It allows you to change the emulate mode of a program on the Line Setup screen without modifying the spreadsheet. When RECEIVE is selected, the INTERVIEW will always monitor the lead opposite its own transmit lead. With **Mode:** EMULATE DTE as the Line Setup selection, the trigger will monitor RD. In Emulate DCE mode, the trigger will monitor TD.

(E) String

When a trigger monitors a data lead for a string, it searches for the exact, entire sequence of characters entered in the condition. Strings have a size limit of 32 characters. If constants are entered in the string, the 32-character limit is applied after all constants have been expanded.

After pressing the STRING softkey or typing STRING followed by a space, begin the string. Strings are always enclosed in quotation marks on the spreadsheet.

Here is an example of a Layer 1 data condition:

```
CONDITIONS: RECEIVE STRING "␍%R" WAIT_EOF
```

where WAIT_EOF delays trigger-true until the block of data holding the string has ended with a good block check.

(F) One-Of Character

When ONE_OF is entered, the trigger looks for any one of the characters in the list that follows. A single character in the data is all that is necessary to match a list. The effect of a "not-equal" character in a one-of list is explained in Section 24.3(I).

After pressing the ONE_OF softkey or typing ONE_OF followed by a space, begin the list. Lists and strings are always enclosed in quotation marks.

(G) Good or Bad BCC

BCC is partly a Layer 1 function, in that the calculation normally is a "hardware" function that tests the physical medium. It also is a Layer 2 function, in that the frame-check calculation is transmitted as part of the Layer 2 protocol. BCC therefore appears as a set of spreadsheet functions both at Layer 1 and Layer 2.

GOOD_BCC (good block-check calculation) and BAD_BCC can only be used as conditions when **Rcv Blk Chk** is turned on. **Rcv Blk Chk** is a menu field on the Line Setup menu: see Section 5.

> NOTE: **Rcv Blk Chk** is *on* automatically when Format: BOP is the Line Setup selection.

Press the softkey for GOODBCC or BAD_BCC when when you want the trigger to take action on receipt of the BCC. The INTERVIEW does the block-check calculation that the user has defined on the BCC Parameters menu and compares it with the received block-check characters. See Table 10-1 and Table 10-2 for the block-check calculations done by the INTERVIEW.

(H) Parity Error

PARITY_ERROR looks for an error in relation to the **Parity** selection made on the Line Setup menu.

(I) Framing Error

FRAMING_ERROR applies to start-stop formats (ASYNC and ISOC) and detects framing errors in relation to the **Stop Bits** field on the Line Setup menu.

(J) Abort

When **Format:** BOP has been selected on the Line Setup menu, you can enter ABORT as a trigger condition. In τ_E-framed protocols, seven consecutive 1-bits in midframe constitute an abort.

(K) Enter Receive String

Enter strings and lists inside quotation marks. A list is a series of characters that can be matched by a single data character. (A string must be matched by a data string.) A one-of condition is an example of a list. All ASCII-keyboard,

control, and hexadecimal characters are legal in a receive string or list. Of the special-character keys, [NOT LOOP], [CONT SYNC], [FLAG], and [SHIFT]-[FLAG] are valid. [SHIFT]-[FLAG] displays the sync symbol [S] on the screen, and causes a search for the sync pattern.

[BIT MASK] is not valid. Bit masks are entered in receive strings by the keying sequence illustrated in Table 32-1.

Constants are also legal in any character position in a list or string. See Section 32, Strings, for an explanation of these string-search tools.

(L) Wait for End Of Frame

After the double-quotation mark is entered to close a string or list, the final Layer 1 condition appears under [F1] on the rack of softkeys. The condition is WAIT_EOF, or "wait for the end of the frame" before coming true and taking any actions. See Figure 31-3.

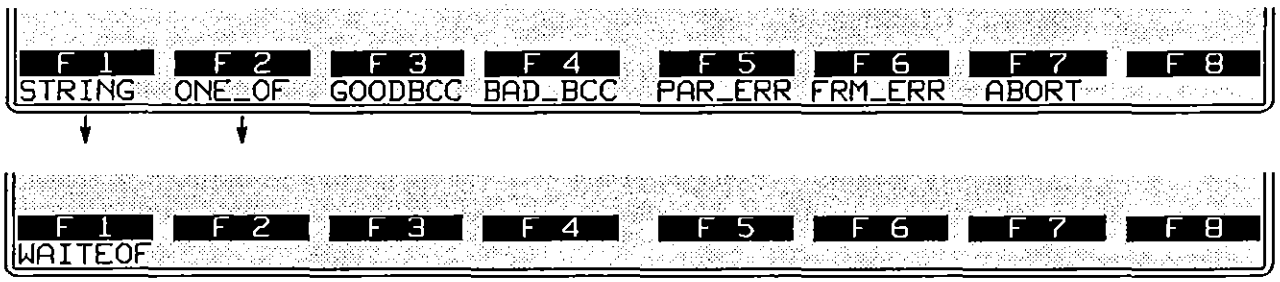


Figure 31-3 String and one-of conditions can be linked to a good BCC at the end of the frame ("EOF").

The WAIT_EOF condition does not occur above Layer 1, since data is not passed up to those layers until the frame is completed.

(M) EIA

Layer 1 conditions can monitor the status of six RS-232/V.24, V.35, or RS-449 control leads plus an additional seventh lead, the user-assigned (UA) input jack on the RS-232/V.24, V.35, or RS-449 test-interface module (TIM). Leads available for triggering are RTS, CTS, CD, DTR, DSR and RI.

The EIA condition is a transitional/status condition. This means that when it is used alone it is true only if it transitions to true; but used in a trigger in combination with other conditions, it retains its status of on or off without having to transition to either status. The rules for combining conditions are explained in Section 30.2.

After pressing the EIA softkey or typing EIA followed by a space, make your lead selection from the upper rack of softkeys in Figure 31-4. Then select a status of ON or OFF.

Figure 31-4 EIA leads monitored by the spreadsheet program.

For the standard RS-232/V.24 interface, ON implies that a lead is more positive than +3 volts with respect to signal ground. OFF implies only that a lead is not at or above the ON threshold, not necessarily that a minus threshold has been attained.

This is an example of an EIA condition:

CONDITIONS: EIA DTR OFF

(N) Xmit Complete

"SENDing" a transmission means queueing a transmission to send. The layer protocol (the RTS-CTS handshake, for example, at Layer 1) may delay the actual transmission. The XMIT_COMPLETE condition (selectable in the bottom rack of softkeys in Figure 31-1) will not come true until the transmission actually has been sent. Use this condition to start accurate response-time measurements.

31.2 Layer 1 Actions

When a block of Layer 1 conditions has been entered, press [DONE] to access the softkeys for ACTIONS. The set of seven Layer 1 actions is shown in the softkeys in Figure 31-5. The names of these actions are SEND, EIA, OUT_SYNC, IDLE_LINE, ENHANCE, and CAPTURE. The other, darkened softkeys in the figure are layer-independent actions present at every layer, discussed in the previous section of this manual.

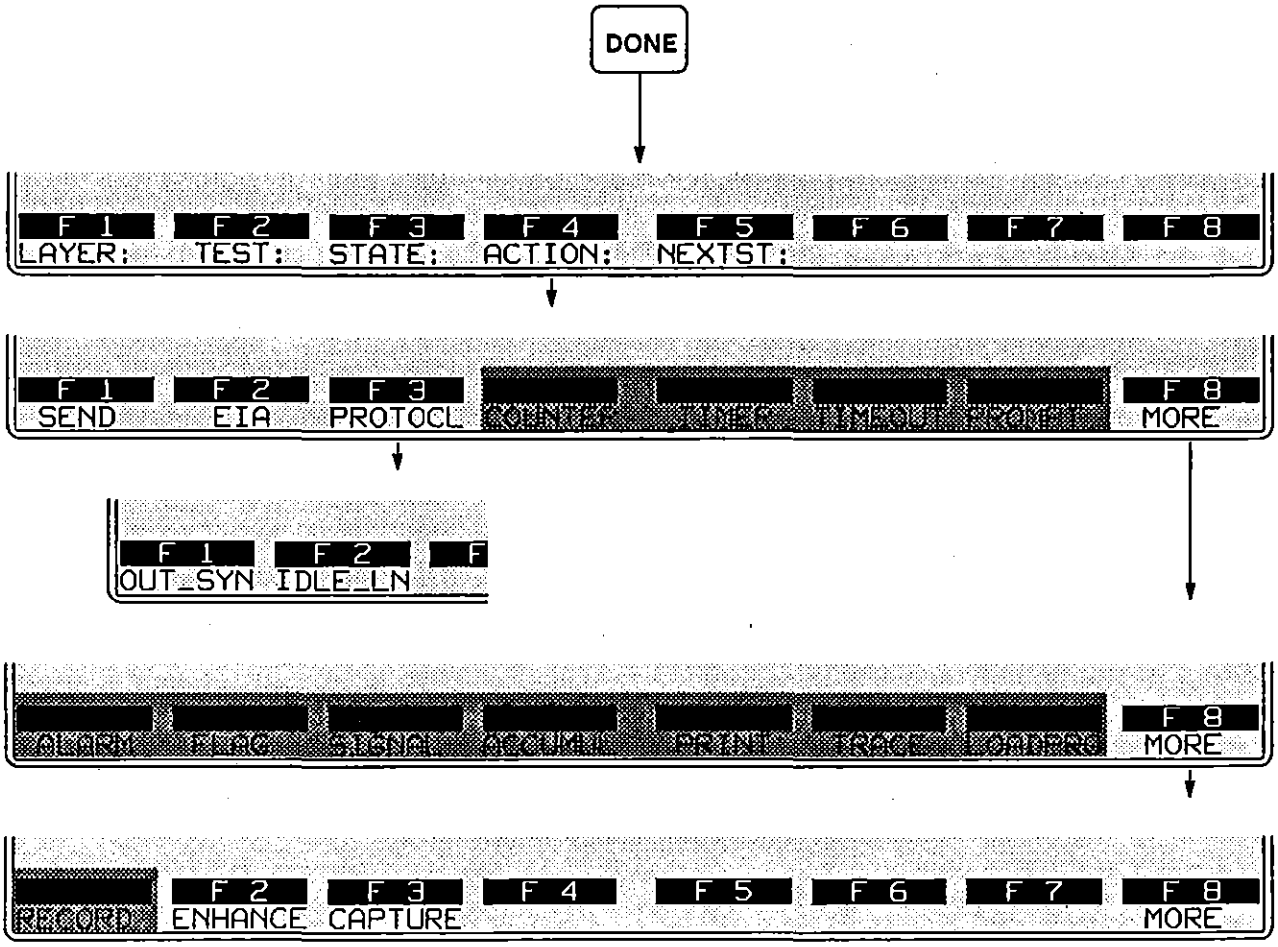


Figure 31-5 Layer 1 actions.

(A) Send

There is one SEND action—transmit a string. While transmissions occur at all layers, only Layer 1 allows the user to type in a complete transmission, character by character. At higher layers, the user types the names of protocol elements and the software converts these mnemonics to strings. The user enters character strings directly at higher layers only into specified user-data fields.

The spreadsheet compiler identifies strings by the quotation marks surrounding them. Send-strings have no size limit (for practical purposes). All ASCII-keyboard, control, and hexadecimal characters are legal in a send-string. Special keys (FLAG, CONT CODE, SET MASK) are not legal.

To insert a canned fox message into a transmit string, type FOX inside of double parens, as follows: ((FOX)). Remember that the double parens are special characters produced by the CTRL-9 and CTRL-8 combinations. Constants, counters, and flags can also be embedded in a string. See Section 32, Strings.

Press the SEND softkey or type SEND followed by space to begin the entry. The prompt **Enter Transmit String** appears as in Figure 31-6. Enter the string inside of quotation marks.

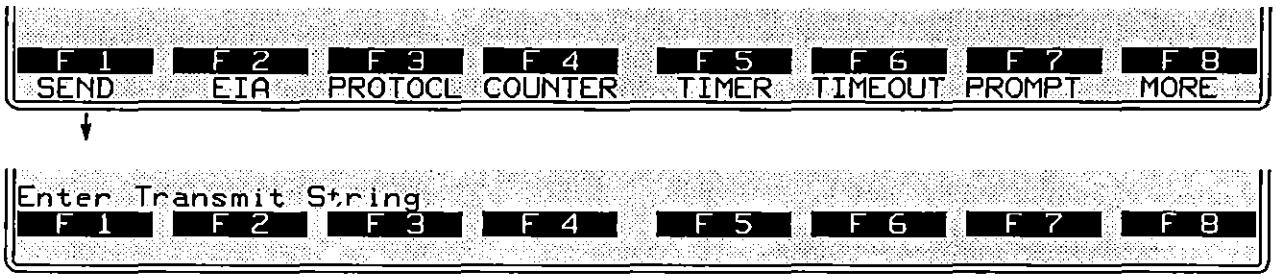


Figure 31-6 Transmit a string.

After quotation marks are typed in to close the transmit string, a set of softkeys appears for the error-checking value that will be appended to the transmit string. One of these must be selected; otherwise, the program will not compile and a **Premature End of File** error message is generated.

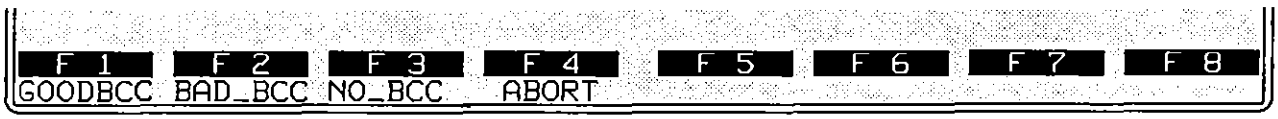


Figure 31-7 Select a block-check calculation to end the transmission.

1. *Good BCC*. This softkey entry allows you to append a good block-check sequence to your transmitted message. The INTERVIEW will make the proper calculation based on the parameters selected on the BCC Setup screen (see Section 10).

2. *Bad BCC.* Press the softkey labeled BAD_BCC to append an errorred block-check to your transmission. Assuming that Rcv Blk Chk: ON is the selection on the Line Setup menu, a BCC error will be indicated on the screen of the INTERVIEW by a ◘ symbol. See Figure 31-8.

   For BOP format, the bad BCC will be CRC-16 instead of CCITT. For other formats it will be an inverted good BCC.

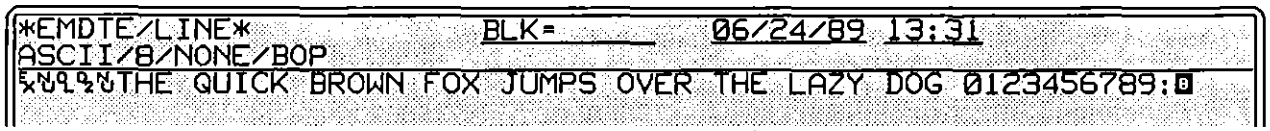


Figure 31-8 The INTERVIEW's TD monitor has detected a bad BCC transmitted by the unit's own TD driver.

3. *No BCC.* The NO_BCC softkey pertains to non-BOP formats only. Instead of appending a block-check calculation to a text message, the transmitter will revert directly to idle-line condition.

   Please note that receivers that are expecting BCC characters will treat the idle characters generated by the INTERVIEW as block-check characters. The INTERVIEW's own receivers (unless they go out of sync first) will display a bad-BCC symbol on the screen. (Refer to Figure 10-3.) The device under test probably will detect a BCC error and reject or ignore the message.

   The user may, of course, enter a good BCC "manually" as part of the text string that precedes the NO_BCC selection.

4. *Abort.* Abort is a BOP function only. Instead of appending a proper frame-check sequence (FCS), the transmitter will hold the lead at mark for eight bits (or longer if the transmitter is idling $^{F}_{F}$). Inside of a frame, seven 1-bits in a row are sufficient to signal an abort.

   An aborted message is shown in Figure 31-9.

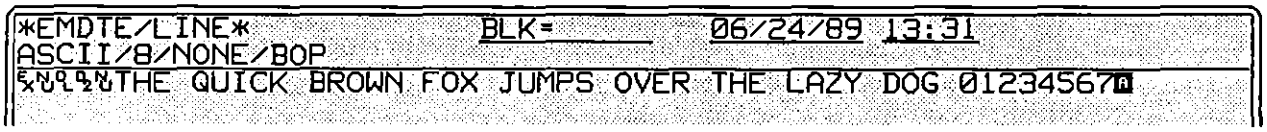


Figure 31-9 The INTERVIEW aborts a BOP frame by closing it with a byte of *FF* instead of *7E*.

(B) EIA

Press the softkey for EIA or type EIA followed by a space to bring five RS-232 leads and four auxiliary leads under spreadsheet control. The nine softkeys that represent EIA actions are illustrated in two separate racks of keys in Figure 31-10.

EIA actions are available only when the unit is in one of the emulate modes. A maximum three RS-232 leads are controllable at one time. When **Mode:** EMULATE DCE is the Line Setup parameter, you control CTS, CD, and DSR. You may enter RTS ON or DTR ON as a spreadsheet action; but the DTE, not the INTERVIEW, controls these leads, and the actions will not take effect. To turn RTS or DTR on, first you must emulate a DTE.

The AUX softkeys allow you to apply off/on voltage to any of the AUX output jacks (four on the RS-232 Test Interface Module, three on the V.35 and RS-449 TIMs) seated in the rear of the unit. (Refer to Figure 12-5 and Figure 47-3.) These AUX outputs are useful for turning on and off a signal that is not a softkey selection or not under the control of your emulation. Section 12.3 cites the example of an INTERVIEW in Emulate DTE mode that is using the AUX0 pin to control CTS from the "wrong" side of the interface.

> NOTE: The AUX actions on the spreadsheet have nothing to do with the 25-pin TTL AUXILIARY connector at the rear of the INTERVIEW.

After selecting a lead to control, select a status of OFF or ON. In the RS-232 specification for drivers, *on* is defined as +5 V to + 15 V while *off* means a range of -5 V to -15 V..

This is an example of an EIA action:

```
ACTIONS: EIA DTR ON
```

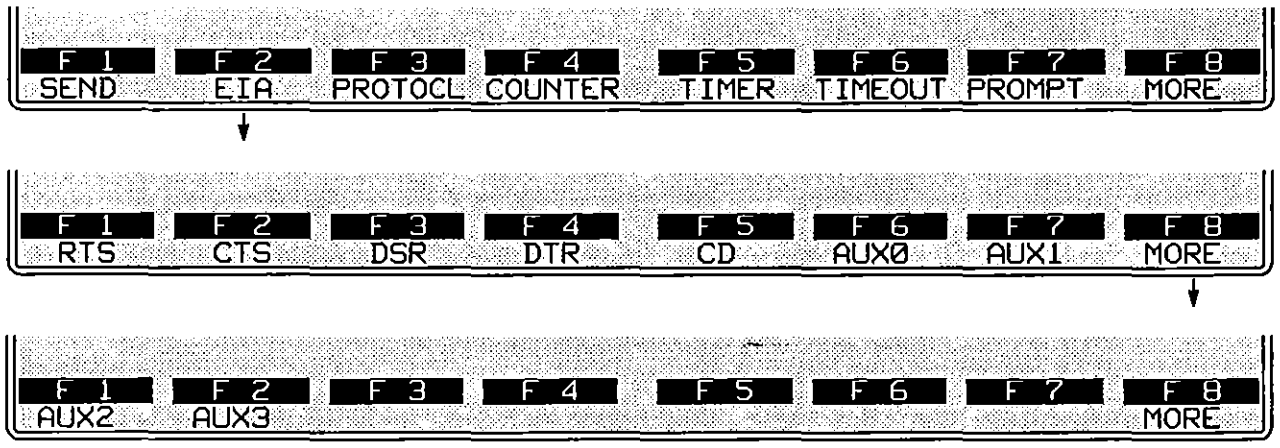


Figure 31-10 Five EIA leads and four AUX leads are under program control.

(C) Outsync

When the outsync action is taken, one or both receivers go out of synchronization from trigger true until the next synchronization pattern is received. All data that occurs in between outsync and resynchronization is considered "idle." If **Display Idle:** OFF is selected on the line setup, a receiver out of synchronization will prevent data from being presented to the screen and the character buffer as well as to the test program.

The outsync action also initiates the search for sync. Receivers that are already in sync do not look for sync. As soon as a receiver goes out of sync, the formatting logic begins to test for the one- or two-character sync pattern one bit at a time.

The outsync action may be useful when the information following a header group, for example, is of no interest. Simply go out of sync until the beginning of the next frame, when synchronization will restore the data display automatically. CAPTURE DTE (or DCE) OFF performs a similar function, except that "capture" must be turned on again by trigger when you want to resume the display.

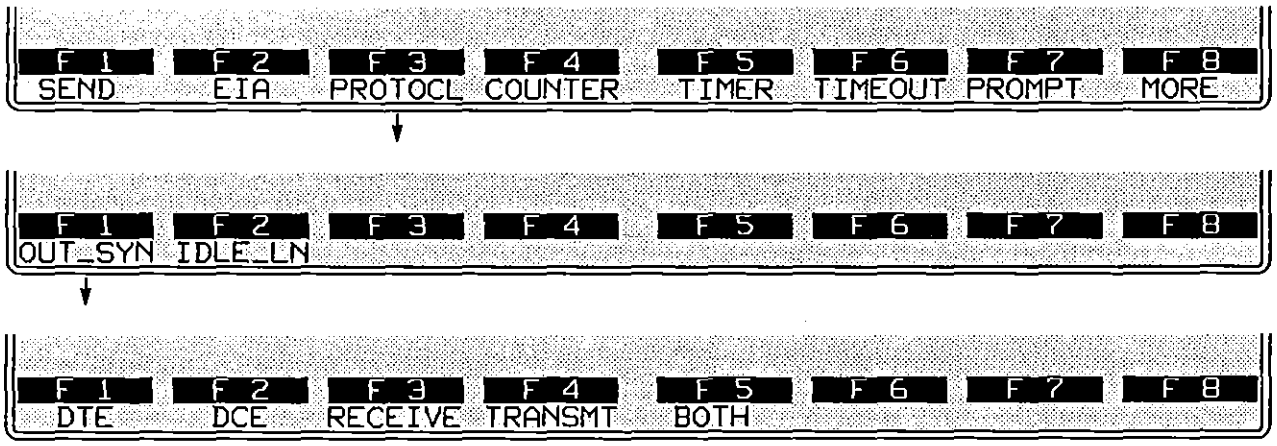


Figure 31-11 The spreadsheet program can force one or both data leads out of sync.

After you have pressed the OUT_SYN softkey or typed OUT_SYNC followed by a space, select one or both leads from the softkeys illustrated in Figure 31-11. RECEIVE and TRANSMT may refer to DTE or DCE, depending on your emulate mode at the moment. These selections allow you to change your emulation on the Line Setup menu without having to worry about changes to the spreadsheet program.

(D) Idle Line

IDLE_LINE allows you to use a trigger action to change the idle-line condition applied by the INTERVIEW. If you press the softkey for IDLE_LN (see Figure 31-11) or type IDLE_LINE followed by a space, the words **Enter Idle Character String** will appear on the prompt line in the softkey area at the bottom of the screen. Enter a single alphanumeric, control, or hexadecimal character in quotation marks. The red LED on the [HEX] key should be *on* for hexadecimal entry.

The idle-line action applies only when **Format:** SYNC has been selected on the Line Setup menu. This trigger action is useful for tests in protocols that employ different idle characters to signal changes in protocol state. An example is X.21 or X.21 BIS, which in various states will idle ^F, ^U, +, and so on.

Here is an example of an IDLE_LINE action:

```
ACTIONS: IDLE_LINE "+"
```

(E) Enhance

The spreadsheet program can be used to enhance display data selectively. Data on either or both sides of the line may be enhanced. Figure 31-12 shows typical reverse-image enhancements. Enhancements are stored in the character buffer for later review: see Section 7.3.

Enhancements that pertain to the plasma display are reverse-image, blink, and hex. In addition to these, a low-intensity enhancement can be applied to data that is transmitted to a black-and-white monitor connected at the RS-170 port (see Figure 1-6).

Blink, reverse and low enhancements activated by the trigger-menu or spreadsheet program can be mapped to colors on a color monitor attached at the INTERVIEW's RGB port (Figure 1-6). See Section 17.2 for an explanation of how blink, reverse, and low enhancements relate to character and background colors in the RGB output.

Enhancements are available at every protocol level, but only Layer 1 enhancements affect the raw-data display. Higher-level enhancements are applied to the protocol trace for a given layer.

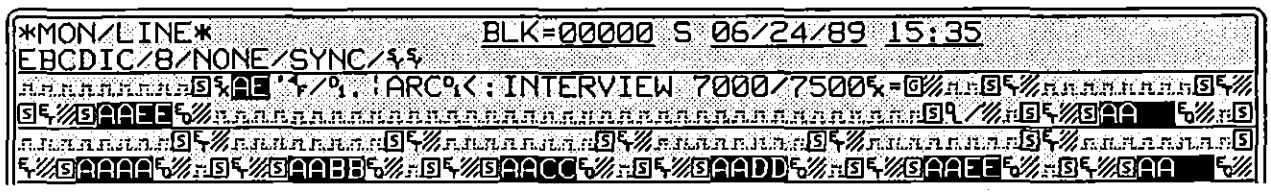


Figure 31-12 Enhancements may be used to highlight protocol fields.

After pressing the ENHANCE softkey or typing ENHANCE followed by a space, select one or both leads from the second level of softkeys in Figure 31-13.

Next, select the type of enhancement from the third tier of softkeys in Figure 31-13. Enhancements may be used in combination (such as reverse blink, or low-intensity reverse). Then at the final level, turn the enhancement ON or OFF.

1. *Reverse image*. Reverse-imaged characters are presented as dark letters on a lighter background.

2. *Low intensity*. This attribute does not affect data on the plasma display, which supports one display intensity only. Characters that are given this attribute will appear in low intensity on a CRT that is attached to the INTERVIEW through the RS-170 port.

Figure 31-13 Layer 1 enhancements must be turned off as well as on by trigger.

3. *Blink.* BLINK causes data to be highlighted by a high-intensity area that blinks on and off. This is the most conspicuous highlight for small portions of data.

4. *Hexadecimal.* When the HEX enhancement is turned on, all data affected by the trigger is displayed in hexadecimal. Once data is stored in the buffer as hexadecimal, it remains in this format even if the [HEX] key is toggled.

   Refer to Figure 6-17 for data in which hex translation has been turned on for protocol characters and off for user (ASCII) data.

(F) Capture

This action turns on and off the presentation of data to the screen—that is, it stops or "freezes" the display—and capture of data to the screen buffer (character RAM). Unlike the Manual Freeze mode initiated by the [FREEZE] key, however, the "capture off" action does not allow you to scroll through the buffer while the test continues.

This action allows you to use the spreadsheet program to find important data and then preserve it in the buffer when it would otherwise be overwritten and lost.

Here is a sample capture action:

```
ACTIONS: CAPTURE BOTH OFF
```

where OFF means freeze the display and BOTH means with respect to DTE and DCE.

After pressing the CAPTURE softkey or typing CAPTURE followed by a space, select DCE, DTE, or BOTH from the rack of softkeys shown in Figure 31-14. On a subsequent set of softkeys, select ON or OFF as the capture action.

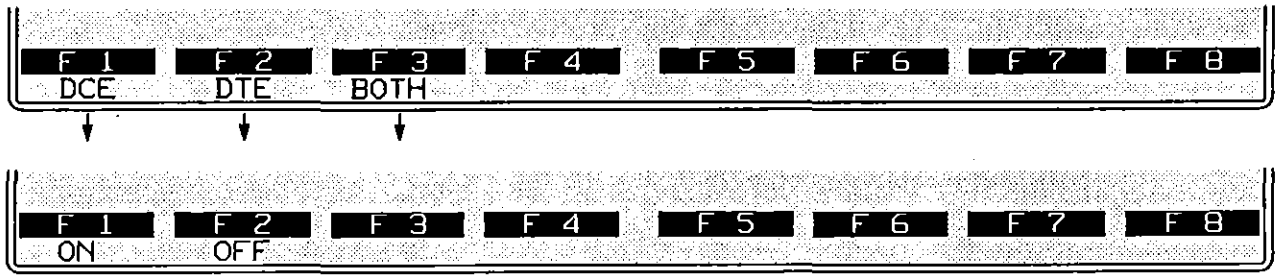


Figure 31-14 Screen display ("capture") can be turned on or off with respect to one data lead or both.

1. *DCE*. This option disables or enables the buffering and display of DCE (RD) data. Suppressing one data lead only does not serve the purpose of preserving data indefinitely in the buffer, since the other lead eventually will overwrite the buffer.

2. *DTE*. The TD lead by itself can likewise be suppressed or displayed.

3. *BOTH*. This option suppresses or displays all data.

4. *ON*. This action enables buffering and display of the selected data.

5. *OFF*. This action suspends buffering and display.

32 Strings

A string on the Protocol Spreadsheet is a sequence of text characters that the operator encloses in quotation marks and enters following certain keywords. Strings are valid in both conditions (at Layer 1) and actions (at any layer). Depending on its use in the program, the string may be searched for, transmitted, printed out, or written to the screen while the program is running.

"Lists" are a subset of strings with an important distinguishing feature: where a string is a sequence of characters, a list is a set of single characters. Examples of lists are one-of conditions at Level 1 of the spreadsheet, or keyboard conditions at any level.

Apart from Layer 1 receive conditions and transmit actions at all layers (discussed below), strings are valid also in KEYBOARD conditions, where a list of keys may be entered, any one of which will satisfy the condition; in IDLE_LINE actions, where a single-character "string" entry represents the new idle character; in LOAD_PROGRAM actions, where the string must match the absolute pathname of the file to be loaded; and in PROMPT and TRACE actions.

All ASCII-keyboard, control, and hexadecimal characters are legal both in receive and transmit strings.

Two ASCII characters are treated in a special way. If you wish to include a quotation mark within a string, you must precede it with a backslash character (\"). If you wish to include a backslash character in a string, you must precede it with a second backslash character (\\). A single backslash is never included in the string.

Control characters are entered into text strings by the action of the CTRL key together with the key that bears the control-character mnemonic at the top right corner. Note that CR ("carriage return") is the mnemonic at the top right corner of the M key. Press CTRL-M to enter ␍ into a text string. The RETURN key does not produce a character entry.

Table 32-1
Valid Entries in Receive Strings

Type entry	Example	Key sequence	Example in string or list (1of)	This data satisfies string condition	Data beginning (arbitrarily) w/ AB satisfies 1of condition
ASCII	2	[2]	"123"	123	AB2
"	\"	[\] [SHIFT]-[']	"1\"3"	1"3	AB"
\	\\	[\] [\]	"1\\3"	1\3	AB\
Control	^V	[CTRL]-[V]	"1^V3"	1^V3	AB^V
Hex	0B	[HEX] [0] [B]	"10B3"	10B3	AB0B
Not Equal	≠	[NOT EQUAL] [2]	"1≠3"	113	A
Bit Mask	((XXXX1111))	[CTRL]-[9] [X] [X] [X] [X] [1] [1] [1] [1] [CTRL]-[0]	"1((XXXX1111))3"	10F3	AB0F
Not equal to bit mask	≠XXXX1111))	[NOT EQUAL] [CTRL]-[9] [X] [X] [X] [X] [1] [1] [1] [1] [CTRL]-[0]	"1≠XXXX1111))3"	123	A
Don't Care	☒	[DON'T CARE]	"1☒3"	153	A
Flag	F	[FLAG]	"1F3"	1F3	ABF
Sync	S	[SHIFT]-[FLAG]	"1S3"	1S3	ABS
Constant	((A)) where A is defined in a CONSTANT field as A = "abcdefg"	[CTRL]-[9] [SHIFT]-[A] [CTRL]-[0]	1((A))3	1abcdefg3	ABabcdefg

32.1 Strings To Be Matched Against Line Data

String conditions are legal in STRING and ONE_OF conditions at Layer 1 only.

Receive strings (and DTE/DCE strings) have a size limit of 32 characters. Their size cannot be expanded through the use of constants. (Any constants will be expanded *before* the size limit is enforced during compilation of the program.)

(A) Special Characters

Of the special-character keys, [NOT COUNT], [DON'T CARE], [FLAG], and [SHIFT]-[FLAG] (for the [5] character) are valid. [BIT MASK] is not valid. Bit masks are entered in receive strings by the keying sequence illustrated in Table 32-1.

(B) Embedded Strings ("Constants")

The string represented by a constant may be embedded in a receive string or a list. A constant is a textual string that is represented by a symbolic name. This name is inserted into a string or list inside of double parens. Double parens are special non-ASCII characters produced on the keyboard by [CTRL]-[9] and [CTRL]-[0].

An example of a constant used in a spreadsheet condition is the following:

```
CONDITIONS:
  RECEIVE STRING "((ADDR_A))E0"
```

The data that satisfies this string will depend on the definition of the constant. Here is one possible definition:

```
CONSTANTS:
  ADDR_A = "AA7F7F"
```

The data that satisfies the condition will include the expanded constant along with the rest of the string: AA7F7FE0.

32.2 Strings To Be Transmitted

Only Layer 1 allows the user to type in a *complete* transmit string, character by character. In the following transmit string, the entire transmission including sync characters is inside of quotation marks:

```
SEND "5A5A0A7E0" NO_BCC
```

At higher layers, the user types the names of protocol units and values as "keywords" and the software converts these elements to strings. Immediately following the keyword entries, the user may add a string in quotation marks. Here is an example of a string following non-string entries in Layer 2 SDLC:

```
SEND FRMR ADR=C1 P/F=1 "1116600E0" GDBCC
```

All ASCII-keyboard, control, and hexadecimal characters are legal in a transmit string. None of the special-character keys ([NOT EQUAL], [ANY MASK], [DON'T CARE], [FLAG]) is valid.

(A) Constant

Constants may be transmitted. Simply place the name of the constant inside of double parens and insert the unit into the string. While the test is being compiled, the constant is replaced in the string by the text that is assigned to it.

The canned "fox" message is a built-in constant named FOX that is defined internally as follows: FOX = "THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG 0123456789." An example of the FOX constant as it appears in a transmit string is given in Table 32-2.

(B) Transmit Variables

Certain variables may be transmitted also. Any number of counters and flags may have their values transmitted at any point.

If a counter or flag is named inside of double parens in a transmit string, the current hexadecimal value of the *low-order byte* of that counter or flag is transmitted with the rest of the string. An example of a counter used in a transmit string is given in Table 32-2.

In order to be referenced in a transmit string, a counter or flag must first be created in a trigger-menu or spreadsheet condition or action. The counter or flag need not be named on a statistics screen.

Do not name a counter (or flag) in a transmit string if has the same name as another flag (or counter). It is unpredictable which one will be transmitted.

The low-order byte of a counter or flag is the default byte to be transmitted. The second byte will be transmitted instead if the name of the counter or flag is followed by **[1]** inside of the double parens. Here is an example of a Layer 2 transmission that includes both bytes of a flag named **seq** (as well as a fox message):

```
SEND INFO ADR=C1 NR=AUTO NS=AUTO "²c%0%2%1((seq[1]))((seq))%3%9%8 5F((FOX))¹₃"
```

Flags are two bytes long, counters are four. All four bytes of a 32-bit counter may be transmitted. Here is a transmit string that sends a complete counter named **fourbyte**:

```
SEND " counter = ((fourbyte[3]))((fourbyte[2]))((fourbyte[1]))((fourbyte))"
```

(C) Data Request

A transmit string that is created at one protocol layer may be passed down transparently to lower layers, one layer at a time. A user-entered message that is sent down at Layer 4, for example, is detected at Layer 3 as an N_DATA REQ primitive and may be handed down to Layer 2 as an "((N_DATA))" string.

The string is appended either to a SEND DATA action (or to a DL_DATA REQ primitive). See the example below. The SEND DATA action will append a packet header to the N-data automatically. The DL_DATA REQ primitive will not add a header to the N-data string; but the user may enter additional data inside of the quotation marks (*not* inside the double parens).

Layer 2, in turn, detects the data as a DL_DATA REQ primitive, and may hand it down to Layer 1 in the form of a "((DL_DATA))" string appended to a SEND INFO action (or to a PH_DATA REQ primitive).

```
LAYER: 4
     STATE: transport
        CONDITIONS: KEYBOARD " "
        ACTIONS: N_DATA REQ "((FOX))"
LAYER:3
     STATE: network
        CONDITIONS: N_DATA REQ
        ACTIONS: SEND DATA PATH= 0 "((N_DATA))"
LAYER:2
     STATE: datalink
        CONDITIONS: DL_CONNECT REQ
        ACTIONS: DL_CONNECT CONF
        CONDITIONS: DL_DATA REQ
        ACTIONS: SEND INFO "((DL_DATA))"
```

Data is sent up the layers also. The mechanism for passing data upward is the GIVE_DATA action included in the protocol personality package at each layer. Since the user will not normally wish to add protocol headers to upward-moving data, this data is not treated as a separable string inside of quotation marks. It is passed upward transparently in the GIVE_DATA action.

Table 32-2
Valid Entries in Transmit Strings

Type entry	Example	Key sequence	Example in transmit string	Data transmitted
ASCII	2	[2]	"123"	123
"	\"	[\] [SHIFT]-[']	"1\"3"	1"3
\	\\	[\] [\]	"1\\3"	1\3
Control	$^{\wedge}_{V}$	[CTRL]-[V]	"1$^{\wedge}_{V}$3"	1$^{\wedge}_{V}$3
Hex	$^{0}_{B}$	[HEX] [0] [B]	"1$^{0}_{B}$3"	1$^{0}_{B}$3
Constant	((A)) where A is defined in a CONSTANTS field as A = "abcdefg"	[CTRL]-[9] [SHIFT]-[A] [CTRL]-[0]	"1((A))3"	1abcdefg3
Fox	((FOX))	[CTRL]-[9] [CAPS LOCK] [F] [O] [X] [CTRL]-[0]	"1((FOX))3"	1THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG 01234567893
Counter or flag, (low-order byte)	((addr)) where addr is the name of a counter with a current decimal value of 14	[CTRL]-[9] [A] [D] [D] [R] [CTRL]-[0]	"1((addr))3"	1$^{0}_{E}$3
Counter or flag, second byte	((seq[1])) where seq is the name of a flag, the second (high-order) byte of which has a binary value of 01010100	[CTRL]-[9] [S] [E] [Q] [[] [1] []] [CTRL]-[0]	"1((seq[1]))3"	1$^{5}_{4}$3
Data in a data-request primitive	((DL_DATA)) at Layer 2, where Layer 3 string is fox message and Layer 3 header is $^{1}_{0}{}^{0}_{7}{}^{6}_{E}$	[CTRL]-[9] [D] [L] [SHIFT]-[-] [D] [A] [T] [A] [CTRL]-[0]	"1((DL_DATA))3"	1$^{1}_{0}{}^{0}_{7}{}^{6}_{E}$THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG 01234567893