Transparency Masters
for
An Introduction to Simulation Using GPSS/H

Chapter 10:
Extending the Set of Test-Mode Capabilities
Chapter 10
Expanding the Set of Test-Mode Capabilities

- The Concept of Block-Based Interrupt Conditions
- The Test-Mode break and unbreak Commands
- How Block Breakpoints Can Lead to Stuttering
- The Concept of Trapping on Specific Transactions
- The trap xact id Test-Mode Command and its counterpart, untrap xact id
- A Summary of display Modifiers
- The Qualified Form of the display Command
- AT-Points, AT-Lists, and the at and end Test-Mode Commands
Block Breakpoints

- A Block can be "sensitized" by putting a breakpoint on it.

- Then, whenever during a Scan Phase the moving Xact finds itself poised to try to execute such a Block, an interrupt occurs.

- The `break` command is used to set one or more Block breakpoints:

```
break Location-list
```

where `Location-list` is a list of one or more Locations occupied by the Blocks on which breakpoints are to be set.

(specified via Block Labels and/or Location numbers)

- Examples of `break` commands:

```
break backup
break backup 12 highway1
```

- `break`-command execution only sets breakpoints; it does not cause Xact movement to be started or resumed.

(the `run` or `step` commands can be used for this purpose)

- The minimal abbreviation for `break` is `b`
More About Block Breakpoints

- The interrupt message issued when a Block-breakpoint occurs is of the "poised at" type we've seen before:

  XACT 57 POISED AT BLOCK 32 (HIGHWAY1). RELATIVE CLOCK: 759.3725

- The `unbreak` command is used to turn off Block breakpoints:

  unbreak switch 35

- All Block breakpoints can be turned off by using an asterisk (*) as the wild card in an `unbreak` command:

  unbreak *

- The minimal abbreviation for `unbreak` is `unb`

- All breakpoints (and trap conditions) that are in effect are listed in response to issuing this `display` command:

  display breakpoints

- The minimal abbreviation for the `display breakpoints` command is:

  d bre
A Potential Stuttering Effect with Block Breakpoints

- An interrupt occurs whenever an Xact simply tries to move into a Block on which a breakpoint is set.

- In general, then, it may be best to avoid setting breakpoints on Blocks that can deny entry (such as the SEIZE; or such as ENTER, wait-until TEST, wait-until GATE, or PREEMPT Blocks, not yet discussed).

- Assume for example that a modeler wants to be given control whenever an Xact captures the SPRAYER Facility in the Figure 10.1 model:

```
SIMULATE
*
  GENERATE 50,20
  BLOCK2 QUEUE WAITAREA
  BLOCK3 SEIZE SPRAYER
    DEPART WAITAREA
    ADVANCE 48,15
  BLOCK6 RELEASE SPRAYER
    TERMINATE 1
*
  START 100
END
```

- Suppose a breakpoint is set on the SEIZE; then, whenever the RELEASE is executed, each Xact currently blocked at the SEIZE will cause an interrupt to occur, resulting in unwanted multiple interrupts (a stuttering effect).

  (it would be better to set a breakpoint on either the DEPART or ADVANCE Block; then only a single interrupt will occur when an Xact captures the SPRAYER Facility)
Transaction Traps

- An Xact can be "sensitized" by putting a trap on it

- Then, whenever during a Scan Phase GPSS/H picks up a scan-active Xact on which a trap has been set, GPSS/H immediately interrupts the simulation, issues a "poised at" message, and gives control to the Test-Mode user

- The `trap` command is used to set traps on one or more Xacts by Xact id number, e.g.:

```
trap xact 25
trap xact 25 xact 39
```

- As always, issuing a `trap` command only causes one or more traps to be set; it does not cause Xact movement to be started or resumed
  (the `run` or `step` commands can be used for this purpose)

- Note these features of the `trap xact` command:
  - Traps can be set on any number of existing Xacts
  - If an attempt is made to set a trap on an Xact that has been destroyed, GPSS/H issues a message to that effect
  - GPSS/H will "queue up" at most one request to set a trap on an Xact that has not yet been created
  - After a trap has been set on an Xact, it remains set until the Xact is destroyed or until it is turned off by execution of an `untrap xact` command, e.g.,

```
untrap xact 25
```

  - The `display traps` command displays a list of all trap conditions (`scan`, `next`, `system`, `clock`, and `xacts`)
  - The `display breakpoints` command displays a list of all breakpoints and of all trap conditions
The Uses of Block Breakpoints and Transaction Traps

- In general, Block breakpoints and Xact traps can be used to help accomplish four types of things:

  ✓ to help newcomers to discrete-event simulation and/or GPSS/H come to a more careful understanding of the underlying principles by enriching the set of tools available for observing the behavior of models during Test-Mode simulations

  ✓ to help debug certain types of buggy models

  ✓ to help search for conditions under which model behavior is to be monitored closely

  ✓ to help verify models

- Education
  Interacting with models in Test Mode stimulates the thought process and provides insights into and understanding of the GPSS/H data structures (chains) and algorithms (Scan Phase and Clock Update Phase)

  As a result, the modeler can build models more readily and with more confidence that the models correctly capture the logic of the system being modeled

  And if a modeler has questions about aspects of Block behavior, s/he can build test-case models to put a Block through its paces, then monitor model behavior to come to a better understanding of how the Block operates
The Uses of Block Breakpoints and Transaction Traps ... continued

- Debugging
  Suppose an Xact gets caught with its hand in the cookie jar, trying to RELEASE a Facility that it hasn't SEIZE-d; Then GPSS/H issues Error Message 415:

| ERROR NUMBER 415 - RELEASing Transaction did not SEIZE the Facility |
| XACT  591 FROM  461 TO  462 ABS CLOCK: 954.5831 REL CLOCK: 954.5831 |

The user can then re-simulate in Test Mode, set a trap on Xact 591, and follow its movement up to the point that it is poised to execute the RELEASE Block in Location 462. Does the Xact have any business being at that Block? If so, how did it manage to get there without executing the associated SEIZE Block in the process? If not, how did the Xact get to that Block in the first place? The answers to questions like this usually come into focus for the Test-Mode user well before the error itself re-occurs.

The preceding scenario is only suggestive of the types of error conditions that can be investigated by setting traps on Xacts; richer scenarios result in connection with GPSS/H features not yet discussed.
Searching for Special Model Conditions
Suppose a given sequence of Blocks in a model will be executed only under rare circumstances. A breakpoint can be set on the first Block in this sequence. An interrupt then occurs when the rare circumstances come about. The user can then use the `display` command to inspect the state of the model to determine if the rare circumstances should indeed be coming about at this time. The user can also set a trap on the Xact moving down the rare-circumstance path to determine if its experiences correspond to the state of affairs in effect at the time.

Again, the preceding scenario is only suggestive of the aspects of model behavior that can be investigated by setting Block breakpoints.
The Uses of Block Breakpoints and Transaction Traps ... continued

- **Model Verification**
  Verifying a model involves carrying out steps to see if the model has been programmed correctly ("did we build the model right?"; see Figure 1.2)

  For some types of model misbehavior, Error Messages occur, for example:

  ✓ RELEASing Xact did not SEIZE the Facility
  ✓ TERMINATing Xact controls one or more Facilities
  ✓ Xact attempting to decrement Current Contents of a Storage to a value less than zero
  ✓ Xact attempting to decrement Current Contents of a Queue to a value less than zero
  ✓ Non-embryonic Xact attempting to enter a GENERATE Block

  Some types of model misbehavior can occur, however, without resulting in Error Messages, for example:

  ✓ TRANSFER-Block exits have been mistakenly reversed
  ✓ The label on a Block to which Xacts transfer has been put on the wrong Block

  Many other problems similar in spirit to these can come up in connection with GPSS/H Blocks and capabilities yet to be discussed (e.g., TEST Blocks; GATE Blocks; SELECT Blocks; Boolean expressions)
The Uses of Block Breakpoints and Transaction Traps ... continued

- **Model Verification** (continued)
  One interesting way to investigate potential model misbehavior of the no-Error-Message variety is to set traps on one or more arbitrarily chosen Xacts (that is, for arbitrary choices of Xact id number), then monitor these Xacts as they go through their life cycle. Does each of these Xacts make the right moves at the right times under the model conditions that are in effect at the time the moves are being made?

  The process of trying to logically "defend" or explain or understand an Xact's moves under a range of model circumstances forces the modeler to think clearly about the model and can bring model flaws into focus as a result.
The Steps in a Simulation Study
(Figure 1.2)

1. Formulate problem and plan the study
2. Collect data and define a model
3. Valid? (Validation: Are we building the right model?)
   - No
4. Construct a computer program and verify
5. Make pilot runs
6. Valid? (Verification: Are we building the model right?)
   - No
7. Design experiments
8. Make production runs
9. Analyze output data
10. Document and implement results
Selected display-command Modifiers

- Selected modifiers used in display commands to display various types of information, e.g.

- \texttt{display fac}

are given in the following table:

<table>
<thead>
<tr>
<th>Modifier</th>
<th>Information Displayed</th>
</tr>
</thead>
<tbody>
<tr>
<td>blo</td>
<td>Current and Total Block Counts</td>
</tr>
<tr>
<td>breakpoints</td>
<td>Block breakpoints and a list of trap conditions</td>
</tr>
<tr>
<td>cec</td>
<td>Current Events Chain</td>
</tr>
<tr>
<td>clocks</td>
<td>Absolute and Relative clocks</td>
</tr>
<tr>
<td>fac</td>
<td>the Facility Report</td>
</tr>
<tr>
<td>fec</td>
<td>Future Events Chain</td>
</tr>
<tr>
<td>output</td>
<td>the Postsimulation Report</td>
</tr>
<tr>
<td>que</td>
<td>the Queue Report</td>
</tr>
<tr>
<td>traps</td>
<td>a list of trap conditions</td>
</tr>
<tr>
<td>xact=xact_id</td>
<td>the properties of the Xact whose id is xact_id</td>
</tr>
<tr>
<td></td>
<td>(e.g., \texttt{display xact=25})</td>
</tr>
</tbody>
</table>
The Qualified display Command

- The following `display` commands will respectively display the Facility and Queue Reports for all Facilities and Queues in a model, and Block counts for all Blocks in a model:

```plaintext
display fac
display que
display blo
```

- In an alternative form, the `display` command can be qualified to specify particular entity members for which information of the indicated type is to be displayed.

- For example, the following qualified `display` commands respectively display the Facility Report for the ABERTH and TUGBOAT Facilities, the Queue Report for the MOVING Queue, and Block Counts for the Blocks labeled BYPASS and BLOCK5:

```plaintext
display fac(abarth, tugboat)
display que(moving)
display blo(bypass, block5)
```
AT-Points

- An AT-point is a Block breakpoint with which a list of Test-Mode commands has been associated.

- The Test-Mode command-list is called an AT-list.

- When a Block breakpoint occurs, GPSS/H automatically executes the AT-list (if any) associated with that Block.

- The Test-Mode at command is used to simultaneously set one or more Block breakpoints and build the AT-List associated with the Block(s).

- Here is the pattern for at-command use:

  Figure 10.2

  ```
  : at Location_list
  @ first command
  @ second command
  ...
  @ last command
  @ end
  :
  ```

- Here is an example for building an at-list:

  Figure 10.3

  ```
  Ready!
  : at block6
  @ display blo(block2)
  @ run
  @ end
  : run
  ```
If an AT-list concludes with a run command (as in the preceding example), then GPSS/H of course automatically resumes Xact movement whenever the AT-list is executed.

If the modeler issues the command

```plaintext
display breakpoints
```

each Location occupied by a Block that has an AT-list associated with it is designated by suffixing it with an at-sign (@), e.g.,

```plaintext
block2@
```

The `display atlist` command can be used to display the commands making up an AT-list:

```plaintext
display atlist block2
```

The `unbreak` command is used to turn off an AT-Point and at the same time destroy the associated AT-list, e.g.:

```plaintext
unbreak block2
```