

SEMI E122-0217 (Reapproved 0623) SPECIFICATION FOR TESTER EQUIPMENT SPECIFIC EQUIPMENT MODEL (TSEM)

This Standard was technically approved by the Information & Control Global Technical Committee. This edition was approved for publication by the global Audits and Reviews Subcommittee on December 21, 2022. Available at www.semiviews.org and www.semi.org in June 2023; originally published March 2003; previously published February 2017.

NOTICE: The section symbol (§) is used to reference a numbered section, paragraph, or header and all subordinate paragraphs, headers, and embedded material (i.e., EXCEPTIONs, lists) therein. The paragraph symbol (§) is used to reference a particular numbered paragraph and embedded material therein. When duplicated (i.e., §§ and §§) the symbols are used to reference more than one section or paragraph, respectively.

1 Purpose

1.1 This Standard establishes a Specific Equipment Model for testing equipment (TSEM). The TSEM consists of equipment characteristics and behaviors that apply to this class of equipment. These characteristics and behaviors are required to be implemented. The intent of this Standard is to facilitate the integration of testing equipment into an automated semiconductor factory. This Standard accomplishes this by defining an operational model for testing equipment as viewed by a factory automation controller. This definition provides a standard host interface and equipment operational behavior.

2 Scope

2.1 This Standard defines the view of the equipment through the host communications link but does not define the internal operation of the equipment. It includes a specific processing state model as the basis for the behavior of all equipment of this class.

2.2 This Standard expands testing equipment requirements and capabilities in the areas of the processing state model, collection events, alarm documentation, remote commands, variable items, and process program management.

NOTICE: SEMI Standards and Safety Guidelines do not purport to address all safety issues associated with their use. It is the responsibility of the users of the Documents to establish appropriate safety and health practices, and determine the applicability of regulatory or other limitations prior to use.

3 Limitations

3.1 Communications

3.1.1 It is required that any TSEM-compliant equipment follow the Communications State Model in SEMI E30. In addition, TSEM-compliant equipment shall support the High-Speed Messaging Service (SEMI E37) Communication Standard sending messages over TCP/IP to maximize the amount of data available for monitoring from this class of equipment. This Specification deals only with the behavior of the tester in communicating with the host application.

3.2 Virtual Tester

3.2.1 The concept of the Virtual Tester is described in § 7. The requirements described in this Standard must be fully implemented on each virtual tester, independent of any other virtual tester. For the purposes of this Standard, it can be assumed that the term ‘tester’ refers to the virtual tester as defined in § 7. Any references to the physical tester will be explicitly noted.

4 Referenced Standards and Documents

4.1 SEMI Standards and Safety Guidelines

SEMI E30 — Generic Model for Communications and Control of Manufacturing Equipment (GEM)

SEMI E37 — High-Speed SECS Message Services (HSMS) Generic Services

NOTICE: Unless otherwise indicated, all documents cited shall be the latest published versions.

5 Terminology

5.1 Definitions

5.1.1 *calibration fixture* — any electromechanical fixture required to perform system calibration. The calibration fixture may consist of multiple components with different part and serial numbers.

5.1.2 *class* — classes represent the most coarse view of the test results. At a minimum, there should be two classes defined for each process program: one class representing good units and another class representing failed units.

5.1.3 *datalog* — collection of results of individual test measurements gathered during the execution of the test program.

5.1.4 *diagnostic fixture* — any electromechanical fixture required to perform system diagnostics. The diagnostic fixture may consist of multiple components with different parts and serial numbers.

5.1.5 *execution area* — the area from which a current copy of the process program instructions is executed.

5.1.6 *hard-bin* — hard-bins represent the typical view of the test results. Within a process program, each hard-bin is associated with a single class. Generally, multiple hard-bins are associated with a particular class.

5.1.7 *multi-site testing* — testing of multiple units with one execution of the test program. Each unit has its own test results.

5.1.8 *soft-bin* — soft-bins represent the most detailed view of the test results. Within a process program, each soft-bin is associated with a single hard-bin. Generally, multiple soft-bins are associated with a particular hard-bin.

5.1.9 *station controller (SC)* — the station controller consists of software that coordinates the actions of the test system and the unit handling equipment (wafer prober, package handler, etc.). It may reside on the test system computer or some other computer. One station controller may be in charge of one or more virtual testers.

5.1.10 *system calibration* — test system process required to bring the test system into compliance with the test system manufacturer's system specifications.

5.1.11 *test-board* — the electromechanical interface necessary to enable temporary electrical contact between the unit to be tested and the tester resource. The test-board may consist of multiple components.

5.1.12 *test-head* — a resource of the tester. The test-head is the electromechanical interface between the unit and the tester.

5.1.13 *test-site* — a location on a test-board where one unit at a time is positioned for testing.

5.1.14 *tester executive* — the tester software which controls test program execution.

5.1.15 *testing equipment* — an equipment class generally consisting of integrated mechanisms and controls for performing electrical tests of packaged devices and/or wafer die (units) during the manufacturing process.

5.1.16 *unit* — the functional integrated circuit (or chip) that is to be electrically tested.

5.1.17 *virtual tester* — the virtual tester is a logical concept which describes the dynamic allocation of a portion of the physical hardware available in a test system to a test program that uses those resources to test a unit or set of units.

6 State Model

6.1 The purpose is to define the equipment-specific processing state model and other state models necessary to portray the expected operational states of the equipment to enable host tracking and control in place of a local operator.

6.2 State Model Requirements

6.2.1 The processing state model in this Standard is required for implementing a TSEM-compliant tester. A state model consists of a processing state model diagram, processing state definitions, and a processing state transition table. A state model represents the host's view of the virtual tester, but not necessarily the actual tester operations. All TSEM state model transitions shall be mapped sequentially into the actual equipment events that satisfy the requirements of those transitions. In certain implementations, the tester may enter a state and have already satisfied all of the conditions required by the TSEM state model for transition to another state. In this situation, the tester makes the required transition without any additional actions.

6.2.2 Some equipment may need to include additional states. However, any additional states must not change the TSEM-defined state transitions. All expected transitions between TSEM states must occur.

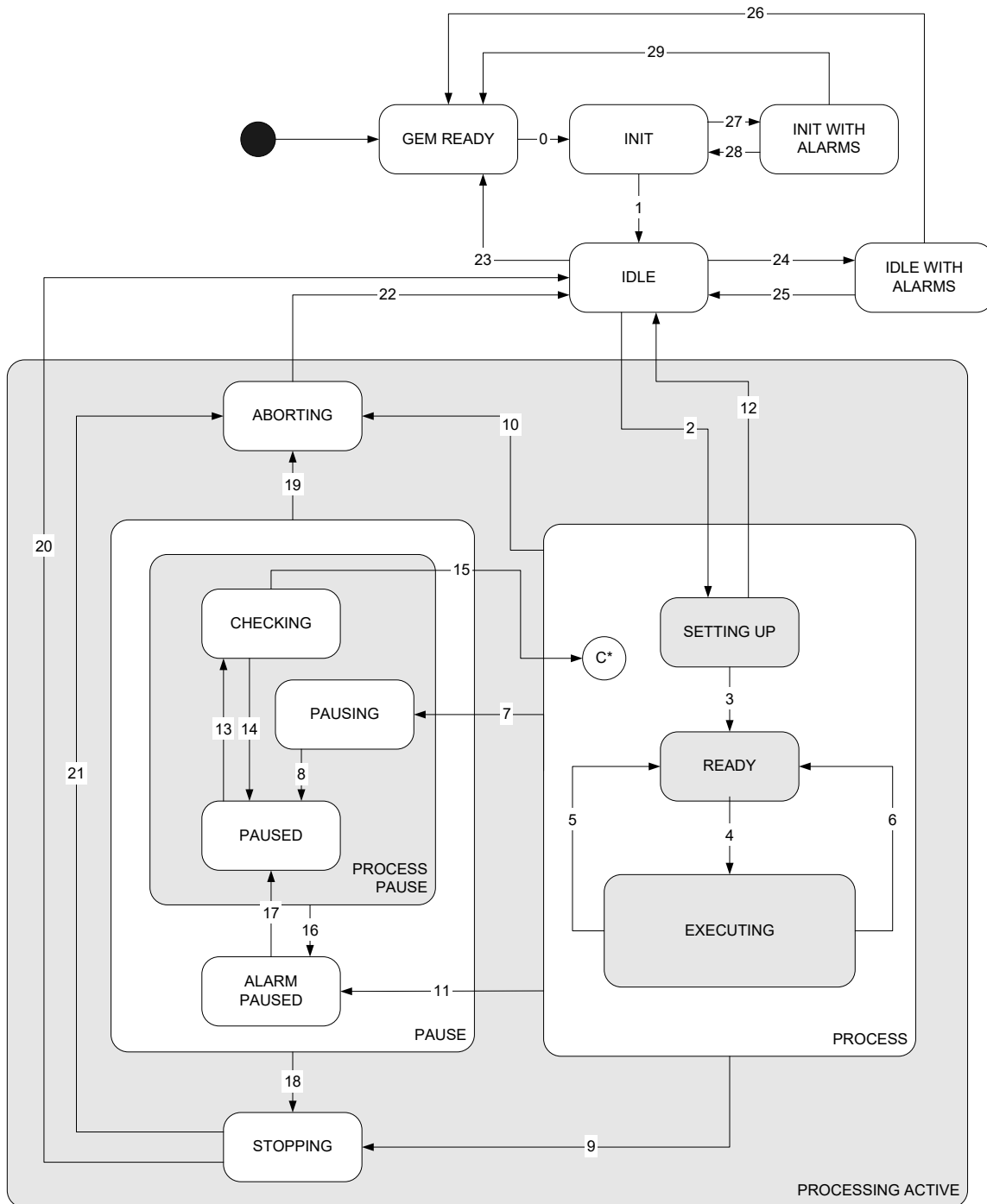


Figure 1
TSEM Processing State Model

6.3 Description of Tester Processing States

6.3.1 *ABORTING (PROCESSING ACTIVE Substate)* — The tester has received an ABORT command. All activity is immediately suspended. The tester is taking appropriate action to bring itself and material to a ‘safe’ state where possible.

6.3.2 *ALARM PAUSED (PAUSE Substate)* — An alarm has occurred in the PROCESS or PROCESS PAUSE states and the tester is waiting for the alarm to be cleared.

6.3.3 *CHECKING (PROCESS PAUSE Substate)* — The tester verifies that updates made to the process program are valid. Process program updates result in the definition of a new process program. This is a similar procedure to that which is done in SETTING UP. If there are no process program updates, this is a pass through state. At the successful completion of verification, a transition, based on the process model condition table, is made to the process state.

6.3.4 *EXECUTING (PROCESS Substate)* — The tester is processing material.

6.3.5 *IDLE* — The tester executive is loaded and the tester is awaiting a command. IDLE is free of ALARMS and error conditions. If this state is entered with valid alarms (e.g., as a result of an abnormal exit from ABORTING, STOPPING, or SETTING UP) then this state is a pass through to IDLE WITH ALARMS.

6.3.6 *IDLE WITH ALARMS* — An alarm has occurred in the IDLE state and the tester is waiting for all alarms to be cleared. An operator may be required to clear alarm conditions. The system shall remain in this state until all alarm conditions are cleared, including those that require operator intervention.

6.3.7 *INIT* — Tester initialization is occurring. The tester executive is loading.

6.3.8 *INIT WITH ALARMS* — An alarm occurred during the initialization process.

6.3.9 *PAUSE (PROCESSING ACTIVE Substate)* — This state is the parent of those substates in which the process program is paused.

6.3.10 *PAUSED (PROCESS PAUSE Substate)* — The PROCESS state has been suspended and the tester is waiting for a command (RESUME, STOP, or ABORT). In this state, the operator may modify conditions of the current Process Program selection.

6.3.11 *PAUSING (PROCESS PAUSE Substate)* — The current state will be suspended at the completion of the current unit(s), if any, and the tester will be brought to a ‘safe’ state.

6.3.12 *PROCESS (PROCESSING ACTIVE Substate)* — This state is the parent of those substates that refer to the preparation and execution of a process program.

6.3.13 *PROCESS PAUSE (PAUSE Substate)* — This state is the parent state of the substates that refer to the pausing of tester operation. The tester is free of alarm conditions in the PAUSE state.

6.3.14 *PROCESSING ACTIVE* — This state is the parent state of the substates that know the identity of the process program.

6.3.15 *READY (PROCESS Substate)* — The process program is loaded. The tester is ready to begin processing and is awaiting a START command.

6.3.16 *SETTING UP (PROCESS Substate)* — The selected process program is loading. The tester is satisfying conditions so that processing can begin. This includes the initialization and validation of the process program and process program-specific calibration. This may be accomplished independently by the tester or may require interaction with the operator and/or host.

6.3.17 *STOPPING (PROCESSING ACTIVE Substate)* — The tester has been instructed to stop processing and shall do so at the next opportunity. All necessary cleanup is completed within this state with regard to material, data, control system, etc.

6.3.18 *GEM READY* — The GEM session is running and ready to accept a START EXEC command from the host.

6.4 TSEM Processing State Transitions Table

Table 1 Processing State Transitions Table

#	Current State	Trigger	New State	Actions	Comments
0	GEM READY	The GEM session has received a START EXEC command.	INIT	If first START EXEC on this physical tester, start the tester executive.	The first tester executive to start initializes all hardware.
1	INIT	All tester initialization is complete with no alarms or error conditions.	IDLE	None.	The tester executive is running.
2	IDLE	A process program is selected.	SETTING UP	Tester-dependent.	Commit has been made to setup and a virtual tester has been defined.
3	SETTING UP	All setup activity has completed and the tester is ready to receive a START command.	READY	None.	The selected Process Program is available for execution, all required hardware for this virtual tester is allocated, and the test program is loaded.
4	READY	The tester receives a START command from a valid source.	EXECUTING	Begins testing the unit(s) at the test-site(s).	The definition of a valid source is application dependent.
5	EXECUTING	The processing of the current unit(s) has completed normally.	READY	The tester processes end of test data.	'Normal' completion of the test program execution.
6	EXECUTING	The processing of the current unit(s) has completed abnormally.	READY		'Abnormal' completion of the test program execution, etc.
7	PROCESS	The tester has received a PAUSE command from a valid source.	PAUSING	The current state is suspended at the completion of the current unit(s).	The definition of a valid source is application dependent.
8	PAUSING	The tester has completed processing the current unit(s).	PAUSED	The tester is waiting for a command (RESUME, STOP, or ABORT).	None.
9	PROCESS	The tester has received a STOP command from a valid source.	STOPPING	The tester executes the actions required to perform a STOP.	The definition of a valid source is application dependent.
10	PROCESS	The tester has received an ABORT command from a valid source.	ABORTING	The tester executes the actions required to perform an ABORT.	The definition of a valid source is application dependent.
11	PROCESS	An alarm occurs.	ALARM PAUSED	PROCESS activity is suspended and the tester is waiting for all alarms to be cleared.	
12	SETTING UP	An equipment initiated transition indicating that the process program failed to load.	IDLE	The hardware resources are released. If the process program load failure caused an alarm condition, this transition is followed immediately by a transition 24 to IDLE WITH ALARMS.	No process program is loaded.
13	PAUSED	The tester receives a RESUME command.	CHECKING	If the RESUME command contains variable process program parameters, validation of the process program parameters begins.	If there are no parameters on the RESUME command, the CHECKING state is a pass through.

#	Current State	Trigger	New State	Actions	Comments
14	CHECKING	Error detected in a new parameter setting.	PAUSED	The tester waits for the parameter correction by operator or host.	The definition of a valid source is application dependent.
15	CHECKING	Parameter checking completes successfully.	State based on conditional table.	The tester resumes PROCESS execution.	This is a conditional re-entry to the PROCESS state (refer to Table 2). If there are no parameters, the CHECKING state is a pass through.
16	PROCESS PAUSE	An alarm is set.	ALARM PAUSED	The tester waits for all alarms to be cleared.	None.
17	ALARM PAUSED	All alarms are cleared.	PAUSED	An operator has verified that all alarms and abort conditions have been cleared. The tester is waiting for a valid command.	None.
18	PAUSE	The tester has received a STOP command from a valid source.	STOPPING	The tester executes the actions required to perform a STOP.	The definition of a valid source is application dependent.
19	PAUSE	The tester has received an ABORT command from a valid source.	ABORTING	The tester executes the actions required to perform an ABORT.	The definition of a valid source is application dependent.
20	STOPPING	The tester completes the actions required to perform a STOP.	IDLE	The hardware resources for this virtual tester are released.	
21	STOPPING	The tester has received an ABORT command from a valid source.	ABORTING	The tester executes the actions required to perform an ABORT.	The definition of a valid source is application dependent.
22	ABORTING	The tester has completed the action required to perform an ABORT.	IDLE	The hardware resources for this virtual tester are released. Unsafe conditions have been resolved where possible. An operator has verified that all alarms and abort conditions have been cleared.	
23	IDLE	The tester receives an STOP EXEC command from a valid source.	GEM READY	The tester executive is unloaded from all testers.	The definition of a valid source is application dependent.
24	IDLE	An alarm is set.	IDLE with ALARMS	The tester waits for all alarms to be cleared.	If the ALARM already exists upon entering the IDLE state, the tester transitions immediately from IDLE to IDLE with ALARMS.
25	IDLE WITH ALARMS	All alarms on the tester have been cleared.	IDLE	An operator has verified that all alarms and abort conditions have been cleared.	The IDLE state is free of alarms.
26	IDLE WITH ALARMS	The tester executive has been stopped by a STOP EXEC command from a valid source.	GEM READY	An operator has verified that all alarms and abort conditions have been cleared. The tester executive is unloaded from all testers.	The definition of a valid source is application dependent.
27	INIT	An error occurred during the loading of the tester exec.	INIT WITH ALARMS		None.

#	Current State	Trigger	New State	Actions	Comments
28	INIT WITH ALARMS	Alarms are cleared.	INIT	An operator has verified that all alarms and abort conditions have been cleared.	None.
29	INIT WITH ALARMS	The tester executive has been stopped by a STOP EXEC command from a valid source.	GEM READY	An operator has verified that all alarms and abort conditions have been cleared. The tester executive is unloaded from all virtual testers.	The definition of a valid source is application dependent.

6.5 Process Model Conditions Table

Table 2 Process Resume Conditions

Condition	Next State
Checking determines that process program conditions were changed.	SETTING UP
Previous state EXECUTING, and no process program conditions were changed.	READY
Previous state READY, and no process program conditions were changed.	READY
Previous state was SETTING UP.	SETTING UP

7 Virtual Testers

7.1 General Definitions

7.1.1 The virtual tester is defined as a single load instance of a test program (regardless of how many test-sites, test system resources, or test-heads are associated with that instance) that is capable of operating as an independent tester in accordance with the state model shown in this Standard.

7.1.2 Loading another instance of the same test program, which uses separate or shared test system hardware resources, establishes another virtual tester. The multiple copies of that test program are independent of each other. In this context, program independence describes the condition that one virtual tester program does not expect or rely on the existence of another virtual tester program in order to run.

7.1.3 For a single test system with one test-head that must be dedicated to a single test program, the physical tester and virtual tester are the same and a single GEM communication interface is needed. In a test system with two test-heads, where each test-head can execute a unique process program autonomously, two virtual testers may be operating at the same time with two GEM communication sessions. If a test-head may be divided into several segments, each capable of autonomous execution of a unique program, there will be one virtual tester and one matching GEM communication session for each of those segments.

7.1.4 The number of virtual testers operating at any time depends on how the test system is configured (hardware and software) and may range from one to the maximum capability of the particular test system. The number of GEM communication sessions is equal to the number of virtual testers supported by the tester.

7.2 GEM Communication Sessions

7.2.1 From the GEM communication host connection, only one program and virtual tester exists and the host GEM communication connection has no knowledge or view of other virtual testers. Independence does not preclude the tester executive or test hardware from sharing resources between virtual testers, such as in the case of power supplies or shared software libraries.

7.2.2 For each virtual tester, the test system must provide a separate GEM client connection. The test system controller has responsibility to manage the multiple client connections.

7.2.3 For each virtual tester the station controller establishes a unique host connection to a single tester GEM client. The station controller has responsibility to manage the multiple host connections.

7.2.4 The test system will have a set maximum number of allowable unique GEM host/client connections. The maximum number of connections (n) and the port IDs for these GEM communication connections are constants known to the station controller and available from the test system controller after that controller is booted.

7.2.5 The maximum number of GEM communication sessions allowed by the test system is also the maximum number of virtual testers allowed by the test system, regardless of the number and allocation of tester resources.

7.2.6 Each unique virtual tester host/client connection is an independent GEM communication session. The following hierarchy relationship can be established.

7.2.7 GEM Communication session (1 ... n) accepts program load instances (1 ... n) which uses tester hardware resources (x ... y) which establishes Virtual Tester (1 ... n). This relationship does not imply any order of assignment. The station controller can select any available GEM communication port in order to load a program and establish a virtual tester.

7.2.8 Multiple station controllers can address a single physical test system, with each station controller establishing its own virtual testers, until the physical limit for the test system is met.

7.3 *GEM Communication Synchronicity to Virtual and Physical Tester*

7.3.1 The GEM communication client interface must represent the actual real time state of the related virtual tester at all times.

7.3.2 Events from outside the GEM communication interface, such as local operator actions, can cause state transitions to occur.

7.3.3 Synchronization of the virtual tester to physical tester events (from any source) is required.

7.3.4 The methodologies and effectiveness of this virtual tester to physical tester synchronization is outside the scope of the TSEM and is the responsibility of the test vendor.

7.4 *GEM Communication Sessions and Alarms*

7.4.1 Any alarm condition that impacts more than one virtual tester must be communicated to all impacted virtual testers.

7.5 *State Model Implications*

7.5.1 When the test system workstation is powered up, and the workstation operating system (O/S) successfully loads, (n) GEM communication port IDs will be established and made available for the station controller. Each GEM communication port represents a potential virtual tester. Each GEM communication port will have a fully independent process state model, which initializes into the GEM READY state.

7.5.2 The station controller can connect to and establish a GEM communication session with any of the available GEM communication ports. Once the station controller establishes a GEM communication session, the remote command START EXEC will load the tester executive and, if successful, cause the process state model for that GEM communication session or virtual tester to transition to the IDLE state. This transition is wholly independent of any other GEM communication sessions. The other GEM sessions remain in their original states.

EXAMPLE: The host issues the first START EXEC to GEM communication port #1. The virtual tester transitions to INIT and starts the Tester Executive which in turn initializes all hardware. If successful, the virtual tester then transitions to IDLE. GEM communication ports 2 through (n) remain in a "logical" GEM READY state even though the physical test system has started the Tester Executive.

7.5.3 If a previous START EXEC remote command to a different GEM communication session on the same tester workstation has already started the tester executive, and the tester executive can support multiple virtual testers, then the new GEM communication session can transition to the IDLE state without starting a new tester executive.

7.5.4 The station controller can issue a STOP EXEC remote command to an active virtual tester. If another virtual tester is running in that test system, the virtual tester that received the remote command will transition to the GEM

READY state, but leave the tester executive running. If no other virtual testers are currently running on that test system, the test system can shut down the tester executive.

7.6 Physical Resources and Virtual Testers

7.6.1 For the case of a multi-site virtual tester, each test-site should be uniquely identified without having to identify the test-head associated with that test-site.

7.6.2 Once a GEM communication session or virtual tester is in the IDLE state, the station controller can issue the PP-SELECT remote command. Until a PP-SELECT has been issued to a virtual tester, and the SETTING UP state is entered, the GEM communication session exists only as a communications path to a potential virtual tester.

7.6.3 In the SETTING UP state, the test system will determine whether the physical hardware resources required for that test program are available. If the resources are available, they are reserved for this virtual tester, the test program loads, SETTING UP process state is successful, and the virtual tester transitions to READY. If the physical hardware resources required for that test program are not available, the SETTING UP process state fails, the virtual tester has an ALARM, and it returns to the IDLE state. Before entering the IDLE state, all resources that have been reserved during SETTING UP must be freed.

7.6.4 Depending on the architecture of the test system, the hardware resources may or may not be tied to a specific virtual tester.

EXAMPLE: A test system with three test-heads might require the use of virtual tester 1 to address test-head 1, virtual tester 2 to address test-head 2, and virtual tester 3 to address test-head 3. If a process-program requiring the resources of two test-heads is loaded on virtual tester 1, it might assign both test-head 1 and test-head 2 to that virtual tester. An attempt to load a program on virtual tester 2 would then fail, but a program might still be loaded to virtual tester 3.

7.6.5 If all test-sites on a test-head are disabled, the test-head itself may be considered to be disabled. If a test-head is disabled the test-head may be undocked from the unit handling equipment and the test-board may be removed from the test-head. Commands to enable test-sites on a test-head that is not connected to the test-board or is not docked should fail. Testing should be able to proceed on test-sites on other test-heads.

7.6.5.1 If a virtual tester includes test-sites on more than one test-head, disabling a test-head should not prevent testing from continuing on the test-sites on other test-heads.

8 Collection Event List

8.1 Requirements

8.1.1 All defined TSEM process state model state transitions are required collection events. In addition to defined process state model state transition collection events, the following collection events are also required.

Table 3 Additional Required Collection Events

<i>Collection Event Name</i>	<i>Collection Event Description</i>
TestBoardChange	A test-board is attached, removed, or changed.
DockStatusChange	The test-head is docked or undocked from the materials handler.
EnableSite	A previously disabled test-site is enabled.
DisableSite	A previously enabled test-site is disabled.

9 Variable Items

9.1 The purpose of this section is to define the list of variable items required by the TSEM. Values of these variables will be available to the host through collection event reports and host status queries.

9.2 Requirements

9.2.1 Variable items required by TSEM are categorized as follows:

- *Common Variables (CVs)* — Variables common to all testers.