# REAL PRODUCTS REFERENCE MANUAL

**Manual Number: MN740 Rev A**

30 May 1986

Valid Logic Systems, Incorporated
2820 Orchard Parkway
San Jose, CA 95134
(408)945-9400 Telex 371 9004

# PREFACE

This manual describes Valid's Realchip™, Realmodel™, and Realfast™ products. The manual assumes that the reader is familiar with the SCALD application tools, and specifically, the SCALD language and ValidSIM, Valid's logic simulator as described in the *SCALD Language Reference Manual* and the *ValidSIM Reference Manual.*

## MANUAL ORGANIZATION

The manual is organized in sections that describe the following topics:

- **General Information.** Presents an overview of hardware modeling and simulation acceleration; defines system specifications.

- **Installation.** Describes how to install and configure the hardware and software of Realchip, Realfast, and Realmodel.

- **Simulator Operation.** Describes how Realchip, Realfast, and Realmodel interact with the ValidSIM logic simulator.

- **Modeling.** Describes how to define device models for hardware modeling by Realchip or Realmodel.

- **Principles of Operation.** Describes basic principles of hardware modeling and simulation acceleration.

- **Networked Realchip.** Describes the additional considerations for hardware modeling in a networked environment.

- **Appendices.** Describes how to install user-created device models on the individual personality modules available from Valid.

## RELATED PUBLICATIONS

The following publications provide additional information for the effective use of Realchip, Realfast, and Realmodel:

- *Tutorial I, Designing with Your SCALDsystem.* Describes basic use of the Graphics Editor, Compiler, and Packager and defines directory structure.

- *Tutorial II, Advanced Topics.* Describes basic use of the simulator and timing verifier and introduces the concepts of structured and hierarchical design.

- *ValidGED Reference Manual.* Describes Valid's graphics editor for schematic capture.

- *ValidCOMPILER Reference Manual.* Describes Valid's schematic compiler that processes designs for the simulator, timing verifier, and packager analysis programs.

- *SCALD Language Reference Manual.* Describes the signal naming conventions and syntax, properties, and macro facility intrinsic to SCALDsystem.

- *ValidSIM Reference Manual.* Describes the operation of Valid's logic simulator.

- *Packager Reference Manual.* Describes the operation of Valid's packager.

- *Library Reference Manual.* Describes how libraries are created and maintained on the SCALDsystem.

# TABLE OF CONTENTS

**Section 1**
**General Information**
VLSI Device Models ..... 1-1
Hardware Modeling ..... 1-1
Device Control Module ..... 1-2
Master Control Module ..... 1-3
Simulation Acceleration ..... 1-3
Network-Sharable Resources ..... 1-4
Hardware Modeling Specifications ..... 1-4
Simulation Accelerator Specifications ..... 1-6
Environmental ..... 1-6
Personality Modules ..... 1-7

**Section 2**
**Installation**
Device Control Module Configuration ..... 2-2
Device Control Module Address ..... 2-2
Device Control Module Terminators ..... 2-4
Realmodel Setup/Hold Time Strapping ..... 2-5
Installing Personality Modules ..... 2-5
Realchip Personality Module Installation ..... 2-7
Realmodel Personality Module Installation .... 2-7
Device Control Module Installation ..... 2-8
Realmodel DCM Installation ..... 2-8
Realchip DCM Installation ..... 2-8
Connecting Realmodel External Power Supplies .. 2-9
External Voltage Input Connectors ..... 2-10
AC Power Connection ..... 2-11
Software Installation ..... 2-12
Testing the Reference Element ..... 2-15
Running the Demo Program ..... 2-15

**Section 3**
**Simulator Operation**
Realchip and Realmodel System Requirements.... 3-1
Realchip and Realmodel Libraries ..... 3-1
Using the Graphics Editor ..... 3-2

The Master.lib File ..... 3-2
Compiling a Design ..... 3-3
Compiling for Logic ..... 3-3
Compiling for SIM ..... 3-4
Simulating a Design ..... 3-4
Simulation Acceleration ..... 3-4
Packaging a Design ..... 3-5
Simulator Limitations ..... 3-5
Simulator Operation ..... 3-6
Realchip/Realmodel Error Messages ..... 3-6
Simulation Acceleration Error Messages ..... 3-12

**Section 4**
**Modeling**
Realchip Directory Structure ..... 4-2
Creating a Reference Element Directory ..... 4-3
Skeleton Files ..... 4-3
SCALD Directories ..... 4-4
Creating the .BODY Drawing ..... 4-4
Body Drawing Conventions ..... 4-5
Creating the .PRIM Drawing ..... 4-6
The Library File ..... 4-7
Library Drawing Method ..... 4-7
Text File Method ..... 4-10
Device Types ..... 4-13
Static Devices ..... 4-13
Dynamic Devices ..... 4-16
Static Forever Devices ..... 4-17
Selection Criteria ..... 4-18
Device Definition File ..... 4-19
Number of Pins Directive ..... 4-23
Pin Block ..... 4-24
Jig ID Directive ..... 4-30
Jig Type Directive ..... 4-31
The Clock Block ..... 4-31
The Reset Sequence Block ..... 4-32
The Delay Table Block ..... 4-37
Running the MAKEALLMSPRIM Script ..... 4-38
Verifying the Model ..... 4-38
Creating the Test Circuit Drawing ..... 4-40
Creating the Test Programs ..... 4-41
Compiling for Simulation ..... 4-41
Simulating the Test Circuit ..... 4-42

Interpreting the Results ..... 4-42
Modeling Considerations ..... 4-42
Pattern Presentation ..... 4-43
Sampling ..... 4-45
The SAMPLE=SPECIAL Directive ..... 4-48
Sampling Rules ..... 4-49
Device Definition File Examples ..... 4-50
68010L8 Device Definition File ..... 4-51
2901C Device Definition File ..... 4-53

**Section 5**
**Principles of Operation**
(currently unavailable)

**Section 6**
**Networked Realchip**
Networked Realchip Server ..... 6-1
Simulator Software ..... 6-2
Platforms Supported ..... 6-3
Networked Errors ..... 6-4
Bringing Down the System ..... 6-5
Bringing the System Up ..... 6-5
Restarting the Networked Realchip Server ..... 6-6

**Appendix A**
**Realmodel 128-Pin Adapter**
Personality Module Orientation ..... A-2
Socket Pin Numbering ..... A-3
PGA Pin Numbering ..... A-3
Vector Pin Pad Array ..... A-4
DIP Work Area ..... A-4
Discrete Component Work Area ..... A-5
Positioning the Reference Element ..... A-5
Unconnected PGA Connections ..... A-19
Power and Ground Wiring ..... A-23
Clock and Feedback Wiring ..... A-23
No-Connect Pins ..... A-24
Socket Interface Pin Assignments ..... A-24
External Voltages ..... A-27
Filter Capacitors ..... A-27
Jig ID Strapping ..... A-27

**Appendix B**
**Realchip 128-Pin Micro-Simulation Adapter**


Introduction ..... B-1
Personality Module Board Orientation ..... B-1
Socket Numbering ..... B-2
PGA Pin Numbering ..... B-2
Reference Element Support ..... B-3
Positioning the Reference Element ..... B-3
Jig ID Strapping ..... B-4
Clock and Feedback Wiring ..... B-4
No-Connect Pins ..... B-5
Filter Capacitors ..... B-5
Netlist ..... B-5
Unconnected PGA Connections ..... B-10


**Appendix C**
**64-Pin DIP Personality Module**


Personality Module Orientation ..... C-1
Mounting the Reference Element ..... C-2
Clock and Feedback Wiring ..... C-2
No-Connect Pins ..... C-3
Jig ID ..... C-3
Filter Capacitors ..... C-3
Clearance ..... C-4
Signal to DIP Signal Mapping ..... C-4


**Appendix D**
**68-Pin LCC Personality Module**


Personality Module Orientation ..... D-1
Inserting the Reference Element ..... D-2
Power Wiring ..... D-2
Clock and Feedback Wiring ..... D-4
No-Connect Pins ..... D-4
Jig ID ..... D-4
Alternate Signal Wiring ..... D-5
Filter Capacitors ..... D-6
Socket to LCC Pin Mapping ..... D-6

**Appendix E**
**64-Pin PGA Personality Module**

Personality Module Orientation ..... E-2
PGA Pin Numbering ..... E-2
DIP Work Area ..... E-3
Positioning the Reference Element ..... E-3
Unconnected PGA Connections ..... E-9
Power and Ground Wiring ..... E-11
Clock and Feedback Wiring ..... E-11
No-Connect Pins ..... E-12
Socket Interface Pin Assignments ..... E-12
Jig ID Strapping ..... E-15
Filter Capacitors ..... E-15

**Index**

# SECTION 1
# GENERAL INFORMATION

Realchip™ is a hardware modeling subsystem specifically designed for quick and efficient simulation of logic designs that include devices that cannot be effectively modeled in software. Realfast™ is a simulation acceleration subsystem that provides substantially increased simulation speeds. Realmodel™ combines the simulation acceleration capabilities of Realfast and the hardware modeling capabilities of Realchip into a single, high-performance system.

## 1.1 VLSI DEVICE MODELS

A large library of VLSI device models or "reference elements" is available from Valid for hardware modeling. These reference elements include common VLSI devices such as microprocessors, peripherals, and UARTS and include all of the required modeling software for immediate incorporation into design simulation. Each Valid-supplied reference element includes the physical device (mounted on a personality module that plugs into a device control module in the Realchip/Realmodel chassis), a device definition file that describes the reference element timing, a graphics symbol drawing for schematic entry, and a test circuit and test program for verifying the model.

Blank personality modules for DIP packages of up to 64 pins, pin grid arrays (PGAs) of up to 128 pins, and a 68-pin leadless chip carrier (LCC) package are available along with design aids to make your own hardware models for custom and semi-custom devices.

## 1.2 HARDWARE MODELING

The hardware modeling engines for both Realchip and Realmodel contain two types of operating modules, a Master Control Module and a Device Control Module or

"DCM." The Master Control Module is installed in the host, and the DCM is installed in the Realchip or Realmodel chassis. The Realchip Master Control Module contains 4K vectors of pattern memory, and the Realmodel Master Control Module contains 256K vectors of pattern memory; both modules include pattern memory management functions (i.e., repeat count and data compression logic). Both the Realchip and Realmodel DCMs contain drivers, receivers, and three-state sensing circuitry; the Realchip DCM supports up to 128 active signals from two 80-pin sockets, and the Realmodel DCM supports up to 256 active signals from four 80-pin sockets. Reference elements are mounted on personality modules that plug directly into one or more sockets on the DCM.

Each Realchip Master Control Module can control up to eight DCMs, and each Realmodel Master Control Module can control up to six DCMs.

Up to four Realchip Master Control Modules can be present in a Realchip System for a maximum capacity of 4096 active signals, and up to three Realmodel Master Control modules can be present in a Realmodel system for a maximum capacity of 4608 active signals.

## DEVICE CONTROL MODULE

A single Realchip or Realmodel DCM can interface any type of device. Reference elements can be modeled as static, dynamic, or static-forever devices, and DIP, LCC, and PGA package configurations are supported by the individual personality module types.

Generally, wire jumpers are not required to connect reference element inputs and outputs to DCM socket interface pins; a "device definition file" defines all input/output signal mapping in software.

Both Realchip and Realmodel support reference elements that require multiple clocks; Realmodel additionally supports multiple power supplies.

## MASTER CONTROL MODULE

The Realchip Master Control Module includes 4K vectors of pattern memory, and the Realmodel Master Control Module includes 256K vectors of pattern memory. The full 4K or 256K of pattern memory is available to each device or instance of a device being modeled. Hardware clock generation circuitry (which eliminates the overhead of including clock patterns in pattern memory) and feedback circuitry for synchronizing the application of data patterns on non-resettable devices and modules are contained on the Master Control Module.

Pattern repeat and data compression techniques optimize pattern memory utilization to allow simulations of up to 32 thousand clock edges for reference elements modeled as dynamic devices on Realchip and up to two million clock edges for reference elements modeled as dynamic devices on Realmodel. This long simulation capability makes software-hardware integration of microprocessor-based designs possible before the hardware is available. Hardware integration with entire application programs for dedicated controllers, instruments, and signal processors as well as entire operation system sequences can be simulated.

## 1.3 SIMULATION ACCELERATION

The Realfast and Realmodel simulation accelerator increases the simulation speed of system-level designs to up to 500,000 events/second. This speed is up to 500 times the speed of workstation simulation and allows the user to take advantage of the interactive capabilities of the Simulator when simulating large designs with many test vectors.

The simulation accelerator is based on two micro-coded processors, an "Event Engine" and an "Evaluation Engine." The Evaluation Engine evaluates the logic primitives that make up the design based on changes to the primitive's inputs. The Event Engine then schedules changes in outputs for future evaluation based on the results of the primitive's evaluation and propagation delay. Each engine has its own dedicated 64-bit wide local memory to allow parallel, concurrent operation on large

amounts of information with each cycle. Both engines operate at four million instructions per second and utilize a pipelined architecture. Execution of complex operations in a single instruction cycle, coupled with concurrent operation, pipelined processing, and high bandwidth memory allows the simulation accelerator to attain simulation speeds up to 500,000 events per second. Local memory (for storing the primitives) is expandable to 64 megabytes for a maximum capacity of up to one million primitives.

## 1.4 NETWORK-SHARABLE RESOURCES

A networked version of hardware modeling is available and allows simulations to be concurrently initiated from any workstation on the network. Design files may reside anywhere on the network and may be accessed remotely.

## 1.5 HARDWARE MODELING SPECIFICATIONS

### PATTERN MEMORY

| | Realmodel | Realchip |
|---|---|---|
| Pattern Memory Depth | 256K patterns | 4K patterns |
| Simulation Length (clock edges) | 2000K (typ) | 32K (typ) |
| | 8000K (max) | 128K (max) |

## DEVICE CONTROL MODULE

| | Realmodel | Realchip |
|---|---|---|
| Number of DCMs per Master Control Module | 6 | 8 |
| Number of DCMs per system | 18 | 32 |
| Number of active signals (channels) | 256 | 128 |

## CLOCKING AND SYNCHRONIZATION

- Hardware clock (dynamic devices) eliminates clock patterns.
- User-selectable clock speeds from 1 million to 16 million clock edges/sec (Realmodel) or xxx to 5 million edges/sec (Realchip).
- Operation can by synchronized to reference element state through feedback to accommodate non-resettable devices.

## CHANNEL DRIVER/RECEIVERS

- TTL-, MOS-, CMOS-compatible logic levels.
- Sink current: 24 mA
- Source current: 2.6 mA
- Loading capacitance: 100pF
- Setup and hold time user-settable (Realmodel)

## 1.6 SIMULATION ACCELERATOR SPECIFICATIONS

- Speed: 500,000 simulation events/sec maximum; 200,000-300,000 events/sec typical.
- Capacity: 32K standard primitives; upgradable to 1 million standard primitives (typically 100,000 gate equivalents, expandable to 3.2 million gate equivalents).
- Simulator states: 20.

### EVALUATION AND EVENT ENGINES

- Microcoded bit slice architecture.
- 250 ns instruction cycle time.
- Pipelined operation.
- Dedicated 64-bit wide local memory (1Mbyte, expandable to 32 Mbytes).
- Triple-ported memory accessible to multibus, event engine, and evaluation engine.
- Parallel operation.

## 1.7 ENVIRONMENTAL

| Dimensions | Realchip | Realmodel |
|---|---|---|
| Height | 9 in (23 cm) | 14 in (36 cm) |
| Width | 19 in (48 cm) | 19 in (48 cm) |
| Depth | 30 in (76 cm) | 30 in (76 cm) |

| Power Requirements | Realchip | Realmodel |
|---|---|---|
| Voltage | 110/115 Vac, 60Hz<br>220/240 Vac, 50Hz | 110/115 Vac, 60Hz<br>220/240 Vac, 50Hz |
| Current | xxA (110/115)<br>xxA (110/115) | 15A (110/115)<br>9A (220/240) |

## 1.8 PERSONALITY MODULES

- 64-pin DIP - accepts DIP devices of up to 64 pins.
- 68-pin LCC - accepts 68-pin LCC devices (64 active signals).
- 64-pin PGA - accepts PGA devices from 40 pins (8x8x2) to 64 pins (10x10x2).
- 128-pin PGA - accepts PGA devices from 64 pins (10x10x2) to 132 pins (14x14x3).

# SECTION 2
# INSTALLATION

This section describes the installation of Valid-supplied Realmodel and Realchip devices or "reference elements." Installation of user-defined reference elements follows a similar procedure with the additional requirements of mounting the reference element to be modeled on a blank Realchip or Realmodel personality module and creating the software model for the device. For these operations, refer to the appropriate appendix for the personality module being used and Section 4, *Model Development.*

All Valid-supplied Realchip and Realmodel reference elements include the following:

- BODY and PRIM drawings
- Library file
- Personality Module
- Device Definition File

The basic installation steps to be followed for installing a Realchip or Realmodel reference element are:

1. Configuring Device Control Module (DCM)
2. Installing Personality Module on DCM
3. Installing DCM in system
4. Loading software modeling information
5. Verifying installation

## 2.1 DEVICE CONTROL MODULE CONFIGURATION

The Realchip Device Control Module (DCM) is a 7- by 12-inch Multibus form-factor printed circuit board that plugs in the backplane of the Realchip enclosure. The Realmodel DCM is a 14.5- by 12-inch printed circuit board that plugs into the Realmodel enclosure. Up to eight Realchip DCMs can be used with each Realchip Master board installed in the system, and up to six Realmodel DCMs can be used with each Realmodel Master board.

**CAUTION**

Before removing or installing a Realchip or Realmodel DCM from the backplane, make sure that the front panel power switch has been turned off. Installing or removing a board while power is applied can permanently damage circuitry on the board or the Realchip or Realmodel reference element itself.

### DEVICE CONTROL MODULE ADDRESS

Each DCM installed in the Realchip or Realmodel chassis must have a unique board address number. On the Realmodel DCM, the board address is set with two "suitcase" jumpers; the jumper at matrix WK20 and the jumper at matrix WK46 (see figure 2-1). Note that with the Realmodel DCM, both jumpers (WK20 and WK46) must be in the same position. On the Realchip DCM, the board address is set with a single suitcase jumper at the jumper matrix at D-33 (see figure 2-2).

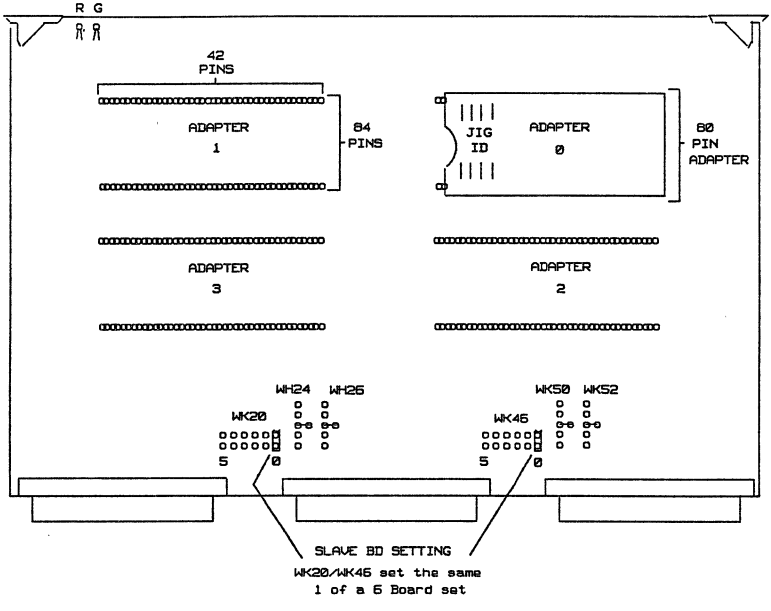


**Figure 2-1. Realmodel DCM Jumper Locations**

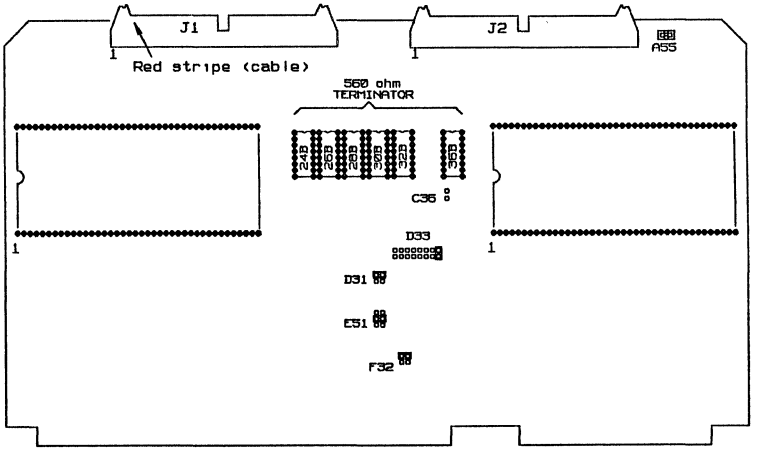


**Figure 2-2. Realchip DCM Jumper Locations**

## REALCHIP DEVICE CONTROL MODULE TERMINATORS

Each Realchip DCM is shipped from the factory with a set of five termination resistor packs installed in the sockets at board locations 24B through 36B and with the clock signal terminator enabled with the jumper strap at board location A55 (see figure 2-2). When multiple DCMs are installed, these termination resistor packs and the clock signal terminator strap must be removed from all but the last (leftmost) DCM in the chassis. Termination resistors and the clock signal terminator for Realmodel are located on the chassis backplane.

**NOTE**

The DCM with the termination resistors always must be present in the chassis when other DCMs are used.

## REALMODEL SETUP/HOLD TIME STRAPPING

The Realmodel DCM includes jumper selectable setup/hold time values for input signals (patterns) applied to the personality modules. The jumper matrices at WH24/WH26 and WH50/WH52 (see figure 2-1) select the setup time in 10-nanosecond increments from 10 to 50 nanoseconds. The default setting is 30 nanoseconds (jumpers installed in position 3) and usually is not changed. When a personality module requires setup/hold time modification, refer to the following table and change the position of all four jumpers. Note that increasing the setup time decreases the hold time (the combined signal setup/hold time is 125 nanoseconds).

**Table 2-1. Setup/Hold Time Selection**

| Jumper Position | Setup Time | Hold Time |
|---|---|---|
| 1 | 10ns | 115ns |
| 2 | 20ns | 105ns |
| 3 | 30ns | 95ns |
| 4 | 40ns | 85ns |
| 5 | 50ns | 75ns |

## 2.2 INSTALLING PERSONALITY MODULES

Valid provides 64-pin DIP and PGA, and a 68-pin LCC (Leadless Chip Carrier) single-position personality modules that can be used interchangeably between Realchip and Realmodel, a 128-pin dual-position personality module for both Realchip and Realmodel. These personality modules mount directly on the DCM; the number of personality modules that can be mounted on a single DCM board is outlined in the following table.

**Table 2-2. Personality Module Capacity**

| Personality Module | Realchip | Realmodel | Positions |
|---|---|---|---|
| 64-Pin | 2 | 4 | single |
| 128-Pin | 1 | 2 | dual |

**NOTE**

The 64-pin DIP and PGA, and the 68-pin LCC personality modules are interchangeable between Realchip and Realmodel systems; 128-pin PGA personality modules are unique to either Realchip or Realmodel.

Before installing a personality module on the DCM, make sure that:

1. The jig ID number strapped on the personality module is unique among all elements in the system. All personality modules (with reference elements) supplied by Valid have unique jig ID strapping; user-created personality modules must be strapped with a unique jig ID number (see appropriate appendix).

2. The termination resistors and clock signal terminator strap are removed from the Realchip DCM unless the board is the last DCM in the system (see section 2-1).

3. A conductive top surface of a reference element is adequately insulated.

The personality module is inserted on the component side of the DCM in the rows of pin connectors. Depending on the size (single or dual) and number of the personality modules, one, two, or, with the Realmodel device control module, all four sets of pin connectors may be used.

Regardless of size, all personality modules are installed with pin 1 oriented to the left when viewed from the component side of the DCM with the bus interface connectors facing down.

## REALCHIP PERSONALITY MODULE INSTALLATION

The Realchip DCM has two sets of 80-pin connectors in two rows of 40 pins. The two rows of pins on the right half of the DCM are referred to as socket 0, and the two rows of pins on the left half of the DCM are referred to as socket 1 (see figure 2-2). The 64-pin DIP and PGA, and the 68-pin LCC personality modules can be mounted in either socket (or one personality module can be mounted in each socket) or a single 128-pin Realchip personality module can be mounted in both sockets.

**NOTE**

To ensure proper pin contact, the 128-pin personality module, whenever possible, should be installed in a DCM that does not have its termination resistor packs installed (i.e., the personality module should not be installed in the last DCM).

## REALMODEL PERSONALITY MODULE INSTALLATION

The Realmodel DCM has four sets of 84-pin connectors in two rows of 42 pins. The two row set of pins in the upper right quadrant of the DCM is referred to as socket 0, the two row set of pins in the upper left quadrant is referred to as socket 1, the two row set of pins in the lower right quadrant is referred to as socket 2, and the two row set of pins in the lower left quadrant is referred to as socket 3 (see figure 2-1). The 64-pin DIP and PGA, and the 68-pin LCC personality modules can be mounted in any socket (up to four personality modules can be mounted) or a 128-pin Realchip personality module can be mounted in either sockets 0 and 1 or 2 and 3 (two 128-pin personality

modules can be mounted on a single Realmodel DCM). Note that when installing a 64-pin DIP or PGA personality module or a 68-pin LCC personality module (which has 64 interface pins), the interface pins are aligned with the 80 socket pins to the right; the two pins on the left of each row supply alternate power supply voltages to compatible Realmodel personality modules.

## 2.3 DEVICE CONTROL MODULE INSTALLATION

Once the DCM has been configured and the personality module or modules have been installed, the DCM is inserted into the backplane of the Realchip or Realmodel chassis.

### REALMODEL DCM INSTALLATION

The Realmodel DCM can be inserted into any of the first six card slots on the right side of the chassis (slots J1 through J6). The component side of the DCM faces the chassis' right side, and no additional cabling is required.

### REALCHIP DCM INSTALLATION

The Realchip DCM can be inserted into any of the card slots in the Realchip chassis. The DCM that contains the termination resistors must be installed in the last (i.e., highest numbered) slot used. The component side of the DCM faces the chassis' right side. After all of the DCMs are installed, the two interface ribbon cables must be installed in the two connectors on the top of DCM (connectors J1 and J2) as shown in the following figure.

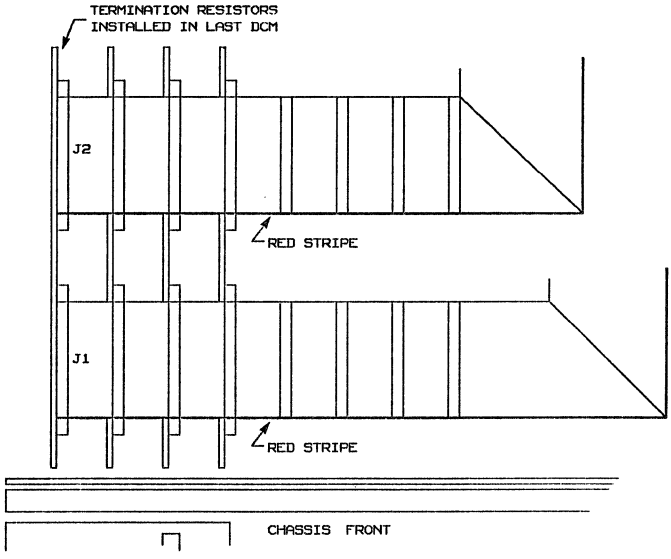


**Figure 2-3. Realchip DCM Cabling**

## 2.4 CONNECTING REALMODEL EXTERNAL POWER SUPPLIES

The Realmodel system is capable of providing three dc voltage levels in addition to the normal +5 volt level to a reference element on a personality module. These three voltage levels are provided from external power supplies (not supplied) that are cabled to three external voltage input connectors on the rear panel of the Realmodel chassis. The three external voltage input connectors are labeled VA, VB, and VC; the voltage level applied to a connector is determined by the power supply connected. An external voltage applied to one of these connectors is routed directly to the designated pin on each of the four personality module interface connector locations on the device control module as shown in figure 2-4. Note that the maximum current rating for each interface pin at each socket location is 1 ampere.

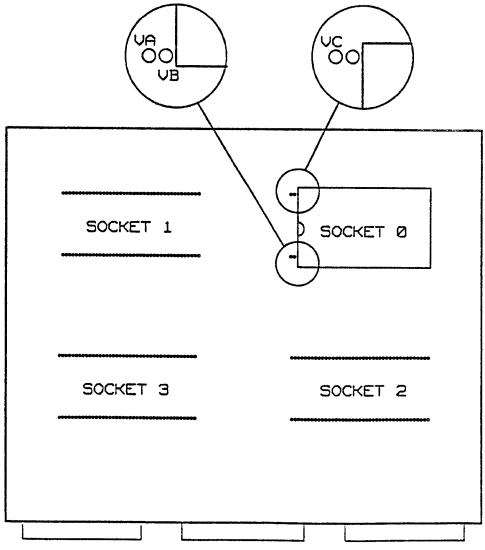


**Figure 2-4. External Voltage Pins on DCM**

## EXTERNAL VOLTAGE INPUT CONNECTORS

Each external voltage input connector has six pins (two pins for V+, two pins for V-, a "sense+" pin, and a "sense-" pin. The individual pin assignments for each input connector are outlined in the following table.

| Pin Number | Function |
|---|---|
| 1 | Sense '-' |
| 2 | V- |
| 3 | V+ |
| 4 | Sense '+' |
| 5 | V- |
| 6 | V+ |

The interface cable from the power supply to the rear panel connector must be fabricated; the required mating connector is a Molex 19-09-1069 with Molex 02-09-1104 pins. Each of the rear panel connectors is fused for a maximum of 15 amperes.

## AC POWER CONNECTION

Both an ac power outlet and a "switch closure" connector are included on the Realmodel rear panel to control power switching of the external supply from the front panel POWER switch. The ac outlet can be used when the external power supply draws no more than 1 ampere of ac current. When the external power supply draws more than 1 ampere, the "switch closure" connector must be used to control primary power to the external supply as shown in figure 2-5.

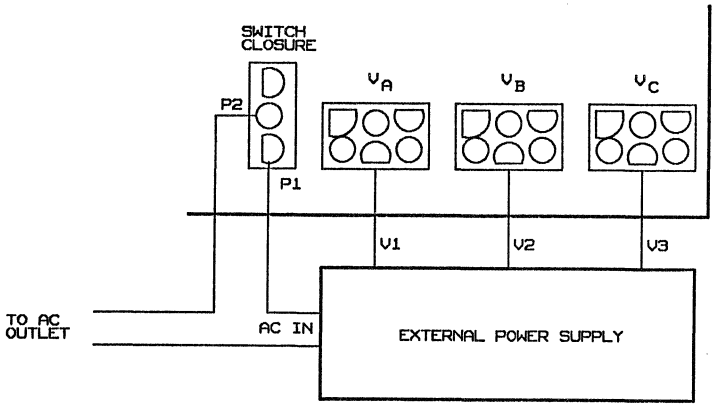


**Figure 2-5. Switch Closure Connector Wiring**

When the "switch closure" connector is used, the ac input cabling to the external power supply must be fabricated. The required mating connector is an AMP 350766-1 with AMP 350547 pins. The connector wiring is outlined in the following table.

| Pin | Function |
|---|---|
| 1 | ac input to power supply |
| 2 | ac input from outlet |
| 3 | not used |

## 2.5 SOFTWARE INSTALLATION

With the initial installation of a Realchip or Realmodel system, the installation of the individual personality modules purchased with the system is performed by the Valid field service engineer. Installation of user-defined personality modules and Valid-defined personality modules purchased after the initial installation is performed by the user.

**NOTE**

To install Realchip or Realmodel personality module software, you must have 'lib' or 'superuser' privilege.

The Realchip or Realmodel software consists of the following:

- A BODY drawing for use by the Graphic Editor
- A PRIM drawing for use by the Simulator and Packager
- A Library file that contains physical information for Packaging the reference element.
- A drawing of the personality module that shows the position of the reference element and the locations of cuts, jumpers, and filter capacitors.

- A device definition file that maps the reference element's signal names to the personality module's socket pins.

- A test circuit and script file for simulating the test circuit.

- A compiled expansion file (*element_name*.exp) and synonyms file (*element_name*.syn) for simulating the demo circuit.

This software is shipped with each personality module on a "tar" tape. To load this tape:

1. Load the tape into the tape drive and place the drive on line.

2. Log in as "lib" or "superuser."

3. Change to directory */u0/lib/realchip*):

   % **cd /u0/lib/realchip**

4. Extract the files from the tape:

   % **tar xv**

The files shown in figure 2-6 will be copied to the new directory. Note that the reference element name (*element_name*) is replaced by the abbreviation "xxx" in the figure.

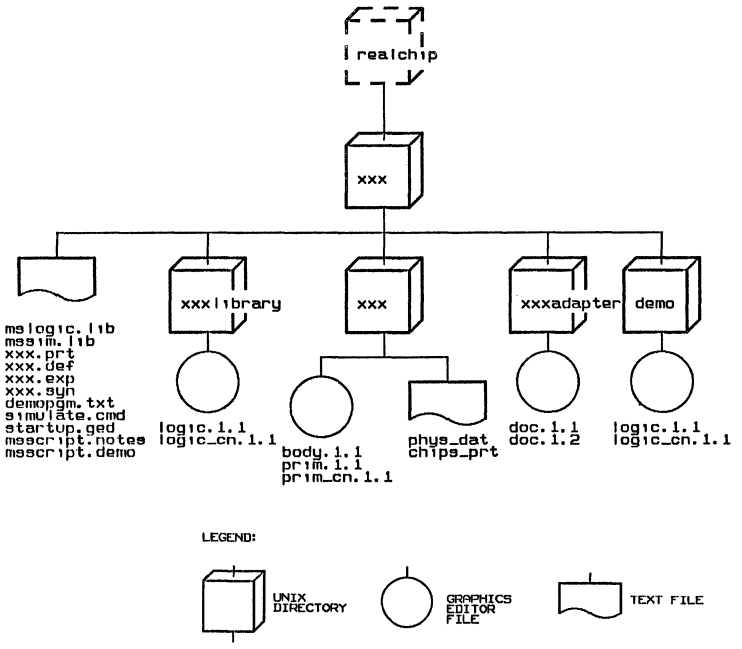


**Figure 2-6. Reference Element Files**

Once the files have been copied to the new directory:

1. Run the "makeallmsprim" shell script to append the device definition file for the new reference element (*element_name*.def) to the combined device definition file (*/u0/lib/realchip/allmsprim.def*):

   % **sh makeallmsprim**

2. Edit the */u0/lib/master.lib* file and add the lines

'*element_name*' '**/u0/lib/realchip/***element_name***/mslogic.lib**';
'*element_name*_**sim**' '**/u0/lib/realchip/***element_name***/mssim.lib**';

## 2.6 TESTING THE REFERENCE ELEMENT

After copying the required files from the tape and updating the *allmsprim.dev* and *master.lib* files, the reference element should be tested using the included demo circuit. This circuit includes a demo (test) circuit (in *demo.wrk*), a Simulator command file (*demo.lst*) and a script file for running the demo (*msscript.demo*); a description of the simulation test is in *msscript.notes*, and the assembly language program for the demo circuit is in *demopgm.txt*. If required to test the reference element, memory code is in the *mlfile.dat* or in the files *mleve.dat* (16-bit even memory) and *mlodd.dat* (16-bit odd memory).

Before running the demo program:

1. Change to the new directory:

   % **cd** *element_name*

2. Edit the **startup.ged** file and add the following lines

   LIBRARY '*element_name*';
   USE demo.wrk;

   where *element_name* is the library name of the new element.

## RUNNING THE DEMO PROGRAM

The expansion and synonym files required to simulate the demo circuit (*element_name*.exp and *element_name*.syn) are included in the directory for the reference element (during normal operation, these files are generated by the Compiler). Before running the demo program, edit the *simulate.cmd* file to include the directive:

ROOT_DRAWING 'demo';

To run the demo program:

1. Change to the directory of the reference element to be tested:

   % **cd /u0/lib/realchip/***element_name*

2. Read the description of the simulation test in the *msscript.notes* file:

   % **more msscript.notes**

3. Enter the Graphics Editor from a full-screen window:

   % **ged**

4. Select the *demo.wrk* directory

   **USE DEMO.WRK**

5. Edit the demo file:

   **EDIT demo**

6. After the demo circuit has been read in, enter

   **SIMULATE**

   to invoke the Simulator in split-screen mode.

7. Run the demo script:

   **script msscript.demo**

8. The Simulator will execute a subset of instructions and then stop (PAUSE statements are included at key points in the simulation). Observe the simulated waveforms and enter

   **resume**

   to continue the simulation.

9. After the simulation is complete, enter

   **EXIT**

   to exit the simulator and return to the Graphics Editor.

# SECTION 3
# SIMULATOR OPERATION

This section defines the operation of the Logic Simulator with Realchip, Realfast, and Realmodel. In addition to the information in this section, the reader is also referred to the "ValidSIM" *Logic Simulator Reference Manual.*

## 3.1 REALCHIP AND REALMODEL SYSTEM REQUIREMENTS

Before Realchip or Realmodel can be used, the device or "reference element" being modeled must be defined by body, primitive, and library drawings, a part file (for packaging), and by a device definition file (for simulation) and, as defined in section 2, the personality module must be mounted on a device control module and the device control module then installed in the Realchip or Realmodel chassis.

## 3.2 REALCHIP AND REALMODEL LIBRARIES

Like component libraries, Realchip and Realmodel reference elements are defined in libraries. Unlike the component libraries, each individual Realchip or Realmodel reference element is defined in its own library directory. With reference elements supplied initially with Realchip or Realmodel, each library is installed under */u0/lib/realchip* when the system is installed; reference elements purchased after initial installation can be user-installed as described in section 2 while user-created reference elements must be installed as described in section 4.

## 3.3 USING THE GRAPHICS EDITOR

Before a Realchip or Realmodel reference element can be added to a logic drawing, the Realchip/Realmodel library must be referenced either within the Graphics Editor session or in the user's local *startup.ged* file.

To reference the Realchip/Realmodel library from within the Graphic Editor, enter the Graphics Editor (type **ged**) and, after the Graphics Editor is loaded, type:

**USE /u0/lib/realchip/***element_name***/mslogic.lib**

where *element_name* is the name of the Realchip/Realmodel library.

To reference the Realchip/Realmodel library in the local *startup.ged* file, edit the file and add the line:

use /u0/lib/realchip/*element_name*/mslogic.lib

Once the Realchip/Realmodel library is referenced, the reference element is added to the schematic with the Graphics Editor's ADD command:

**ADD** *element_name*

Like the component libraries, multiple instances of the reference element can be specified, and the reference element can be wired, moved, rotated, and copied with the corresponding Graphics Editor commands.

## 3.4 THE MASTER.LIB FILE

A Realchip/Realmodel library can be "aliased" in the *master.lib* file to eliminate the library path name entered in the *startup.ged* and *compiler.cmd* files. Note that to edit the *master.lib* file (*/u0/lib/master.lib*), you must be logged in as "lib" or "superuser." Edit the file and add the following lines:

'*element_name*' '/u0/lib/realchip/*element_name*/mslogic.lib';
'*element_name_sim*' '/u0/lib/realchip/*element_name*/mssim.lib';

Note that two entries are required for each Realchip/Realmodel reference element; a name (*element_name*) and a name with a _sim extension (the library name without the extension is entered in the *startup.ged* file and in the *compiler.cmd* file when compiling for LOGIC; the library name with the _sim extension is entered in the *compiler.cmd* file when compiling for SIM).

## 3.5 COMPILING A DESIGN

Schematics that contain Realchip or Realmodel reference elements are compiled for LOGIC (for Packaging the design) and for SIM (for simulating the design). For addition information on the Compiler, see the "ValidCOMPILER" *Compiler Reference Manual.*

### COMPILING FOR LOGIC

To compile a Realchip or Realmodel design for LOGIC, edit the *compiler.cmd* file and add the line:

DIRECTORY '/u0/lib/realchip/*element_name*/mslogic.lib';

Note that if the Realchip/Realmodel library has been referenced in the *master.lib* file, the LIBRARY directive can be used in place of the DIRECTORY directive as follows:

LIBRARY *element_name*;

### COMPILING FOR SIM

To compile a Realchip or Realmodel design for SIM, edit the *compiler.cmd* file and add the line:

DIRECTORY '/u0/lib/realchip/*element_name*/mssim.lib';

Note that if the Realchip/Realmodel library has been referenced in the *master.lib* file, the LIBRARY directive can be used in place of the DIRECTORY directive as follows: line:

LIBRARY *element_name*_sim;

Compiling a design for simulation depends on which version of the Logic Simulator is being used (Realchip requires Simulator Release 7.6 or later, Realmodel requires ValidSIM Release 2.0 or later). With the 7.6 Release, the design must be Compiled for SIM (to generate the required *cmpexp.dat* and *cmpsyn.dat* files) before the circuit can be simulated. With Releases 1.0 and later of ValidSIM, the Compiler automatically creates the required files when the Simulator is invoked. For any version of the Simulator, edit the *simulate.cmd* file to include the directive:

ROOT_DRAWING '*design_name*';

## 3.6 SIMULATING A DESIGN

After a design has been successfully compiled for SIM, the designed can be simulated with the Logic Simulator. Before the Simulator can be run, the Simulator directives file (*simulate.cmd*) must be edited to include a REALCHIP_LIBRARY directive. This directive specifies the name of the Realchip or Realmodel library file that contains the concatenated set of the device definition files for simulating all of the Realchip/Realmodel reference elements. By default, this file is named "allmsprim.def" and is generated by running the "makeallmsprim" script (see section 2). The format of this directive is:

REALCHIP_LIBRARY '/u0/lib/realchip/allmsprim.def';

Note that this directive must be included if any Realchip or Realmodel reference elements are to be simulated and that the name of the file must be enclosed in quotes.

## 3.7 SIMULATION ACCELERATION

To simulate a design using Realfast or the simulation acceleration capability of Realmodel, the "USE_REALFAST ON;" directive must be included in the *simulate.cmd* file. Note that both Realfast and Realmodel use this same directive and that if this directive is omitted (or set to OFF), the simulation acceleration is disabled.

## 3.8 PACKAGING A DESIGN

After a design that includes a Realchip or Realmodel reference element has been successfully compiled (for LOGIC), the design can be packaged by including a "LIBRARY_FILE" directive in the Packager's directive file (*packager.cmd*) for each Realchip or Realmodel reference element used in the design. The form of this directive is

LIBRARY_FILE '/u0/lib/realchip/*element_name*/*element_name*.prt';

where *element_name* is the name of the Realchip or Realmodel reference element. A design using Realchip/Realmodel reference elements can be back annotated like any other design. For more information on packaging, refer to the "ValidPACKAGER" *Packager Reference Manual.*

## SIMULATOR LIMITATIONS

When simulating a design that contains a Realchip or Realmodel device, the following limitations are imposed:

- The duration of simulation is limited for reference elements that are modeled as dynamic or static devices.
- Only reference elements that can be reset to a known initial state can be simulated.
- Dynamic reference elements that sample inputs only on one edge must be capable of operating at a clock frequency of 3Mz or less (Realchip) or 16MHz or less (Realmodel). A dynamic reference element that samples inputs on both edges must be capable of operating at a clock frequency of 2MHz or less (Realchip) or 8MHz or less (Realmodel).

- Simulation of designs that use non-positive setup or hold times produces anomalous results.

- A single transition on a single input pin of a reference element must not produce more than one transition on any given output pin.

- Realchip reference elements must operate from a single +5 volt source (Realmodel supports multiple source voltages; see section 2).

In addition to the above limitations, only a single user can access the Realchip and Realmodel system at any one time (Networked Realchip/Realmodel hardware modeling supports concurrent users; see section 6).

## 3.9 SIMULATOR OPERATION

All of the Simulator's functions are fully supported with both Realchip or Realmodel hardware modeling. When using Realfast or the simulation acceleration capabilities of Realmodel, the following simulation restrictions are imposed:

1. Breakpoints are not supported.

2. Patching is not supported.

## 3.10 REALCHIP/REALMODEL ERROR MESSAGES

The Simulator's *simlst.dat* file contains a list of the commands entered during the simulation and any error messages returned. The following error messages are specific to Realchip and Realmodel; for a complete list of the Simulator error messages, see the "ValidSIM *Logic Simulator Reference Manual.* Realchip is supported by Simulator release 7.6 and ValidSIM releases 1.0 and later; Realmodel is supported by ValidSIM releases 2.0 and later. Note that error messages with an asterisk (*) indicate error messages associated with release 2.1 and later of the Logic Simulator, and error messages with two asterisks (**) are unique to releases 2.0 and earlier.

A majority of the error messages are associated with the device definition file; for information on device definition file format, see section 4.

**NOTE**

If an error is reported in the device definition file, correct the device definition file for the individual reference element and then rerun the "makeallmsprim" script to update the *allmsprim.def* file. Never edit the *allmsprim.def* file directly as running the script will update the file with the erroneous device definition file for the reference element.

*ERROR #28 Special sampling not supported in RM
Generated when a "SAMPLE=SPECIAL" directive is included in a device definition file for a device being modeled by Realmodel (SAMPLE=SPECIAL is only supported by Realchip).

*ERROR #29 Expected SPECIAL
Generated when the SAMPLE=SPECIAL directive in the device definition file for a device that requires the elimination of "precharging" is incorrectly specified.

*ERROR #47 RM hardware failure
Generated when an unexpected hardware failure is detected by Realmodel; an received was received, but the hardware did not stop. This error should be reported to Valid.

*ERROR #56 Realmodel parity error
Generated when a hardware (parity) error is detected in either the Realmodel Interface Board, Device Control Module, or the Master Control Module. This error should be reported to Valid.

*ERROR #59 Realmodel hardware never stopped
Generated when the stop bit in the Realmodel hardware status register is not set by the time that the Simulator is ready to replay the environment sequence. This error indicates either a hardware or software failure and should be reported to Valid.

*ERROR #133 Expected INPUT_SPEC or OUTPUT_SPEC

Generated when an INPUT_SPEC or OUTPUT_SPEC symbol is expected in the device definition file and some other data is found.

*ERROR #134 Expected END_PIN

Generated when an END_PIN statement is missing from the Pin Block of the device definition file.

ERROR #138 Cannot open Realchip Library File

Generated when the Simulator is unable to open the Realchip library file. Check the pathname for the REALCHIP_LIBRARY directive in the Simulator's directives file and the file's permissions.

ERROR #139 Expected INPUT_SPEC, OUTPUT_SPEC, IO_SPEC

Generated when the Simulator is expecting an INPUT_SPEC, OUTPUT_SPEC, or IO_SPEC symbol in the device definition file and finds some other data.

ERROR #140 Illegal JIG_ID value

Generated when the Simulator finds an invalid JIG_ID for the Realchip primitive used in the simulation. The JIG_ID number specified in the device definition file must be the same as the JIG_ID strapped on the personality module. Edit the device definition file for the reference element to correct the JIG_ID entry.

ERROR #141 Expected DYNAMIC, STATIC, or STATIC_FOREVER

Generated when the Simulator finds something other than a DYNAMIC, STATIC, or STATIC_FOREVER jig_type entry in the device definition file for the Realchip primitive.

*ERROR #142 Expected RISE, FALL, or BOTH

Generated when a value other than RISE, FALL, or BOTH is specified for the CLOCK_EDGE directive in the device definition file.

*ERROR #143 Illegal clock_period value(s)
Generated when a clock period of 0 or a negative clock period is specified by the CLOCK_PERIOD directive in the device definition file.

ERROR #150 Expected END_RESET_SEQ
Generated when the "END_RESET_SEQ;" terminating line is incorrectly entered (or omitted) at the end of the reset sequence block in the device definition file for the reference element.

ERROR #151 RESET_SEQ pin not found
Generated when a pin identified in the reset sequence block of the device definition file is incorrectly specified or not referenced in the pin block section.

*ERROR #152 Expected 0, 1, or Z
Generated when a value other than 0, 1, or Z is specified for a reset or pause sequence in the device definition file.

ERROR #153 CLOCK_PIN pin not found
Generated when the pin specified by the CLOCK_PIN directive is not defined in the pin block section of the device definition file.

**ERROR #154 Realchip adapter not found
The Realchip/Realmodel personality module containing the named reference element could not be located. Check that the reference element is installed and properly seated on the device control module. Also check that the JIG_ID in the device definition file matches the jig ID strapped on the personality module.

ERROR #157 Expected END_DELAY_TABLE
Generated when the Simulator could not find an END_DELAY_TABLE entry in the device definition file.

ERROR #158 DELAY_TABLE output pin not found
Generated when a pin identified in the delay table block of the device definition file is incorrectly specified or not referenced in the pin block section.

ERROR #168 odd # of reset_seq for clk_both device
Generated when the reset sequence in the device definition file for a dynamic device with a CLOCK_EDGE=BOTH directive contains an odd number of entries in the sequence.

*ERROR #169 Adapter found with duplicate jig ID
Generated when two personality modules have the same jig ID strapping.

ERROR #170 Pattern RAM overflow. Invalid results
Generated when the Realchip/Realmodel simulation pattern RAM overflows.

ERROR #171 Unsupported clock period range
Generated when the Simulator finds a CLOCK_PERIOD in the device definition file that is outside of the range supported by Realchip/Realmodel.

ERROR #172 Missing pin number specification
Generated when the Simulator is expecting a pin number specification in the pin block of the device definition file and finds some other data.

ERROR #173 DELAY_TABLE input pin not found
Generated when an input pin identified in the delay table block of the device definition file is not referenced in the pin block section.

ERROR #174 PAUSE sequence already given
Generated when the Simulator finds more than one pause sequence in the device definition file.

ERROR #175 Pin number is already used
Generated when the Simulator finds a pin number in the device definition file that is already specified for another pin.

**ERROR #176 **** Realchip in use or not present
Generated when an attempt is made to use Realchip or Realmodel on a system where it is already in use (or non-existent).

*ERROR #176 Cannot find available modeling system
Generated when an attempt is made to use Realchip or Realmodel on a system where it is already in use (or non-existent).

ERROR #178 Feedback may be disconnected
Generated when the feedback connection on a personality module for a reference element that requires feedback is incorrectly wired or disconnected.

**ERROR #180 Static_forever device is not unique
Generated when there is more than one instance of the same static forever device in a design.

ERROR #220 UCP/Realchip delay >= 4096
Generated when the Simulator finds a Realchip/Realmodel (or user-coded primitive) with a delay of 4096 nanoseconds or greater.

**ERROR #231 Expected SPECIAL
Generated when the SAMPLE=SPECIAL directive in the device definition file for a device that requires the elimination of "precharging" is incorrectly specified.

*ERROR #250 Illegal delay of zero in delay table
Generated when a delay of zero for a static or static forever device is specified in the device definition file.

## SIMULATION ACCELERATION ERROR MESSAGES

The following error messages are specific to simulation acceleration (i.e., Realfast or the simulation acceleration capability of Realmodel.

ERROR #201 Output already has Realfast data
Generated when the Simulator finds that the data structure for an output already contains Realfast data.

ERROR #202 Not enough data structure memory
Generated when the Simulator finds that the design is too large to fit in available Realfast memory. Specifically, all of the data structure memory was used up. The design must be run on a Realfast with a larger memory capacity.

ERROR #203 Input already has Realfast data
Generated when the Simulator finds that the data structure for an input already contains Realfast data.

ERROR #204 Primitive not yet implemented
Generated when the Simulator finds that the design contains a Simulator primitive that has not been implemented in Realfast. Either change the design to exclude the primitive or simulate without using Realfast.

ERROR #205 Not enough microcode memory
Generated when the Simulator finds that the design is too large to fit in available Realfast memory. Specifically, all of the evaluation memory was used up. The design must be run on a Realfast with a larger memory capacity.

ERROR #206 SET_MICRO_FIELD has invalid parameters
Generated when an illegal value is found is found in an instruction field of the Realfast microcode. This error should be reported to Valid.

ERROR #207 Output has width > 1
Generated when the Simulator finds an internal error that indicates a Realfast data structure with a width greater than 1.

ERROR #208 Undriven input has illegal default value
Generated when the Simulator finds an internal error that indicates a Realfast data structure with an improper default value.

ERROR #210 Primitive already has Realfast data
Generated when the Simulator finds that the data structure for a primitive already contains Realfast data.

ERROR #211 Monitor code too large
Generated when the Simulator finds that the microcode monitor was larger than expected. This can only happen when the Simulator and /u0/scald/simulator/monitor.int are out of sync; check the installation.

ERROR #212 Monitor returned an error code
Generated when the Simulator finds that the monitor gave a failure indication. This error usually indicates a hardware problem, but also can indicate an internal consistency failure.

ERROR #213 Did not get access to Realfast hardware
Generated when the Simulator is unable to access the Realfast hardware. This error usually indicates that another user currently is using Realfast although this error also can indicate that the Realfast hardware is either incorrectly connected or not connected or that the system is powered down.

ERROR #214 Realfast interrupt but hardware busy
Generated when the Simulator services a Realfast interrupt and finds that Realfast is still running. This error indicates a probable hardware failure; call your field service representative.

ERROR #217 Ran out of event blocks
Generated when the Simulator finds that the Realfast data structure memory was exhausted during simulation; run on a Realfast with more memory.

ERROR #219 Not enough value memory
Generated when the Simulator finds that a design is too large to fit in available Realfast memory. Specifically, all of evaluation memory was used. Run the design on a Realfast with more memory.

ERROR #221 Realfast memory parity error
Generated when the Simulator finds a parity error in Realfast memory during simulation. This error indicates a hardware failure; call your field service representative.

ERROR #222 Feature not yet implemented for Realfast
Generated when the Simulator finds that the user tried to invoke a Simulator feature not supported by Realfast (e.g., patching and breakpoints).

ERROR #223 Cannot open Realfast monitor file
Generated when the Simulator cannot access the /u0/scald/simulator/monitor.int file. Check that this file is present and that it has the proper permission; also check /usr/bin/simassign to ensure that a "RFMON=/u0/scald/simulator/monitor.int" entry is present.

ERROR #224 Cannot open Realfast ALU file
Generated when the Simulator cannot access the /u0/scald/simulator/alumem.int file. Check that this file is present and that it has the proper permission; also check /usr/bin/simassign to ensure that a "RFALU=/u0/scald/simulator/alumem.int" entry is present.

# SECTION 4
# MODELING

This section describes the procedures to be followed to create a software model for Realchip or Realmodel. Installation of the reference element on the corresponding personality module is described in Appendices A through E.

The following steps must be performed to create a Realchip or Realmodel model:

1. Make a Realchip or Realmodel sub-directory.

2. Create the .BODY and .PRIM drawings.

3. Create a Library drawing and .prt file.

4. Characterize the reference element in a Device Definition file.

5. Install the reference element on the personality module.

In addition to the above steps, the following steps are recommended:

1. Create a personality module drawing that documents the mounting of the reference element, any cuts and jumpers, and the installation of filter capacitors and any additional devices.

2. Create a test circuit schematic for verifying operation of the reference element.

3. Create a Simulator script file to exercise the test circuit and simulator stimulus file.

## 4.1 REALCHIP DIRECTORY STRUCTURE

The Realchip (or Realmodel) directory structure for a reference element will include the files and subdirectories outlined in the following table. Note that the file or directory names with an asterisk support the recommended test circuit and adapter documentation and are not mandatory.

**Table 4-1. Realchip Directory Structure**

| Name | Description |
|---|---|
| *element_name* | A directory that contains the .BODY and .PRIM drawing files for the reference element. |
| *element_name*library | A directory that contains the library drawing for the reference element. |
| *element_name*adapter* | A directory that contains the adapter drawing. |
| demo* | A directory that contains the demo (test) circuit schematic. |
| msslogic.lib | A SCALD directory for compiling for logic. |
| mssim.lib | A SCALD directory for compiling for sim. |
| *element_name*.prt | A text file that contains the physical information for use by the Packager. |
| *element_name*.def | The device definition file for the reference element. |

## 4.2 CREATING A REFERENCE ELEMENT DIRECTORY

A Realchip or Realmodel UNIX sub-directory for a user-created personality module must be created under the */u0/lib/realchip* directory. Note that both Realchip and Realmodel use this directory.

**NOTE**

To create a Realchip or Realmodel model, you must be logged in as "lib" or "superuser."

To create the UNIX sub-directory, change to */u0/lib/realchip* and:

% **mkdir** *element_name*

where *element_name* is the name of the reference element to be modeled. When the model is completed, this directory will contain the files and directories outlined in table 4-1.

### SKELETON FILES

In order to create a complete reference element directory, two of the files in the 8086mx directory supplied with your Realchip or Realmodel system are required and can be copied from the 8086mx directory to your new directory. These files are a startup.ged file and a simulate.cmd file. To copy these files, change to your new directory (**cd** *element_name*) and:

% **cp /u0/lib/realchip/8086mx/startup.ged startup.ged**
% **cp /u0/lib/realchip/8086mx/simulate.cmd simulate.cmd**

Make sure that your startup.ged file includes a reference to the standard library (i.e., library standard).

## SCALD DIRECTORIES

In order to compile a Realchip/Realmodel component for logic (for use by the Packager) and for sim (for use by the Simulator), the following SCALD directories (files) are required in the Realchip directory for the device being modeled:

mslogic.lib
mssim.lib

The *mslogic.lib* directory is created when the .BODY drawing is written; the *mssim.lib* directory must be created by the user from the *mslogic.lib* directory with a text editor before the model can be simulated as described in section 4-9.

## 4.3 CREATING THE .BODY DRAWING

To create the .BODY drawing, change to your new directory (**cd** *element_name*) and enter the Graphics Editor (type **ged**). From the Graphics Editor, create the SCALD directory "mslogic.lib"

**USE MSLOGIC.LIB**

This is the SCALD directory where the .BODY and .PRIM drawings will be located. Edit the .BODY drawing for the reference element:

**EDIT** *element_name*.**BODY**

Note that the grid is automatically set to 0.05 2 (i.e., grid spacing is 0.1 inches). Always use this grid setting when making bodies.

## BODY DRAWING CONVENTIONS

The following standards should be followed when making a .BODY drawing for a reference element; refer to the *Library Reference Manual*:

1. Move the body name (note) away from the origin (the small 'x' at coordinates **0,0**) with the SPLIT command.

2. Make a symmetrical outline of the body around the origin ('x') using the WIRE command; make the width wide enough to include pin names within the outline (typically 1.2 inches or 12 grid lines), and make the length long enough to include all of the pins on 0.1-inch centers (i.e., grid lines).

3. In general, place input pins on the left and output pins on the right. Control pins can be placed across the bottom; bidirectional pins can be placed on either side although usually are placed on the right. Connect all pins to the body (outline) with either a 0.1-inch stub (made with the WIRE) command, or a bubble (a 0.1-inch diameter circle with its center between vertical grid lines). Place a tie point (with the DOT command) at the end of each wire stub or on the outside circumference of each bubble.

4. Use the SIGNAME command to attach a signal name to each pin. Position the signal names outside the perimeter of the body and use SHOW ATTACH to verify name/pin attachments. Note that when attaching signal names, be sure that the signal assertion level matches the pin (i.e., that low-asserted signals are attached to bubbled pins).

5. Use the NOTE command to label the pin names as a guide to the user; position the names within the outline as close to the edge as possible.

6. Move the body (element) name to the top of the outline.

7. Attach the "NEEDS_NO_SIZE = TRUE" property 0.05 inches from the origin point and make the property invisible.

Figure 4-1 shows a typical .BODY drawing. After completing the drawing, write the drawing out to create the SCALD directory for the device.

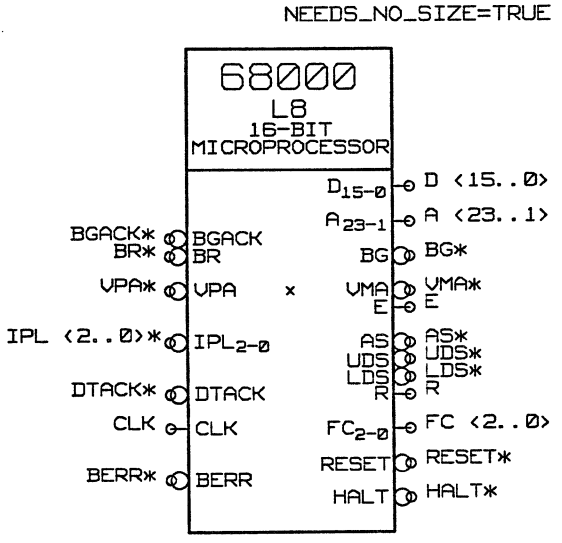


**Figure 4-1. Typical .BODY Drawing**

## 4.4 CREATING THE .PRIM DRAWING

The .PRIM drawing defines the body drawing as a primitive for simulation. Create the .PRIM drawing while still in the Graphics Editor by:

**EDIT** *element_name*.**PRIM**

Add a DRAWING body from the standard library

**ADD DRAWING**

and attach a TITLE and an ABBREV property to the DRAWING body with the PROPERTY command:

TITLE=*element_name*
ABBREV=*element_abbreviation*

Write out the .PRIM drawing.

## 4.5 THE LIBRARY FILE

The Library File (*element_name*.prt) is used to convey physical information to the Packager. This file can be created directly from a Library Drawing or the file can be created from a "skeleton" library drawing and a text file.

### LIBRARY DRAWING METHOD

The following procedure outlines how to create a library file directly from a library drawing:

1. Enter the Graphics Editor and open the directory *demo.wrk*:

   **USE DEMO.WRK**

2. Edit the library drawing for the reference element:

   **EDIT** *element_name* **LIBRARY**

3. Add a B-size page border from the standard library:

   **ADD B SIZE PAGE**

4. Add the body:

   **ADD** *element_name*

5. Use the PROPERTY command to attach a PIN_NUMBER property to each pin (e.g., PIN_NUMBER=1).

6. Use the PROPERTY command to attach an INPUT_LOAD or OUTPUT_LOAD property to each input and output pin or, if the pin is bidirectional, attach an INPUT_LOAD, OUTPUT_LOAD, and a "BIDIRECTIONAL=TRUE" property to the pin (for load values, refer to the manufacturer's specifications). Also, if an output pin can be tied to another output pin (i.e., open collector, open emitter, or three-state, attach an OUTPUT_TYPE property to the pin
(e.g., "OUTPUT_TYPE=(TS,TS)").

7. Use the PROPERTY command to attach a POWER_PINS property to the reference element body
(e.g., POWER_PINS=(VCC:14,49; GND:16,53)).

8. Use the PROPERTY command to attach a FAMILY property to the reference element body (e.g., FAMILY=REALCHIP or REALMODEL).

9. Add a DRAWING body to the drawing and attach a TITLE and ABBREV property to the DRAWING body.

10. Verify the property attachments (SHOW ATTACH).

11. Write out the library drawing and exit the Graphics Editor.

Figure 4-2 shows a typical library drawing.

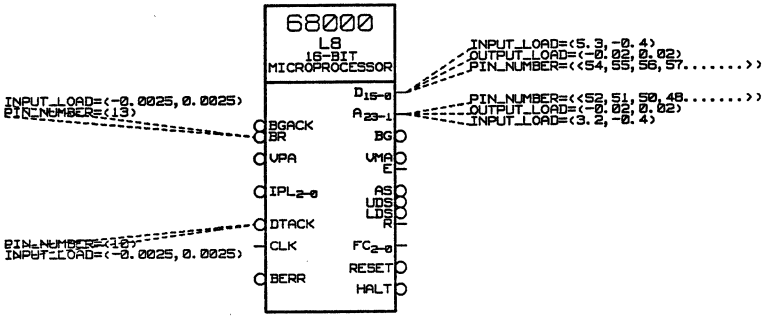


**Figure 4-2. Typical Library Drawing**

After the library drawing has been completed, edit the compiler directives file:

% **vi compiler.cmd**

Add or change the following lines to read:

root_drawing '*element_name* library';
compile logic;
directory 'demo.wrk', 'mslogic.lib';
library standard;
output list, chips;

Compile the library drawing to create the *chips.dat* file:

% **compile**

When the drawing has been compiled, check for Compiler errors (examine the *cmplst.dat* file). Correct any errors and recompile the drawing until an error-free run is attained. After the drawing has been successfully compiled, move the chips.dat file to *element_name*.prt:

% **mv chips.dat** *element_name*.**prt**

## TEXT FILE METHOD

The text file method is similar to the library drawing method except that the pin and body properties are entered into a text file rather than directly on the drawing. This method may prove faster especially with reference elements with a large number of pins. Once the property information is entered into the text file, the *chips.dat* file is automatically updated with the information from the file by running the "addphysinfo" script. To create a library file using this method:

1. Enter the Graphics Editor and open the directory *demo.wrk*:

   **USE DEMO.WRK**

2. Edit the library drawing for the reference element:

   **EDIT** *element_name* **LIBRARY**

3. Add a B-size page border from the standard library:

   **ADD B SIZE PAGE**

4. Add the body:

   **ADD** *element_name*

5. Add a DRAWING body to the drawing and attach a TITLE and ABBREV property to the DRAWING body.

6. Write out the drawing.

7. After the library drawing has been completed, edit the compiler directives file:

   % **vi compiler.cmd**

   Add or change the following lines to read:

   root_drawing '*element_name* library';
   compile logic;
   directory 'demo.wrk', 'mslogic.lib';
   library standard;
   output list, chips;

8. Compile the library drawing to create the *chips.dat* file:

   % **compile**

9. When the drawing has been compiled, check for Compiler errors (examine the *cmplst.dat* file). Correct any errors and recompile the drawing until an error-free run is attained.

10. Create the *phys_dat* file in the *element_name* sub-directory:

    % **cd** *element_name*
    % **vi phys_dat**

11. Enter the part (element) name, family, and power/ground pin assignments and enter the pin name, pin number (in parentheses), type (INPUT, OUTPUT, OC or TS), input/output load, and note any bidirectional pins (e.g., BIDIR). The example on the following page shows a typical phys_dat file:

```
PART element_name
  FAMILY REALCHIP
  POWER PINS (VCC:14,49;GND:16,53)
PIN
  BGACK*  (12)  INPUT   (-0.0025,0.0025)
  BR*     (13)  INPUT   (-0.0025,0.0025)
  CLK     (15)  INPUT   (-0.0025,0.0025)
            .
            .
  D<15>   (54)  TS      (-0.02,0.02)  (5.3,-0.4)  BIDIR
  D<14>   (55)  TS      (-0.02,0.02)  (5.3,-0.4)  BIDIR
  D<13>   (56)  TS      (-0.02,0.02)  (5.3,-0.4)  BIDIR
            .
            .
  E       (20)  OUTPUT  (3.2,-0.4)
            .
            .
  FC<2>   (26)  TS      (-0.02,0.02)  (3.2,-0.4)
  FC<1>   (27)  TS      (-0.02,0.02)  (3.2,-0.4)
  FC<0>   (28)  TS      (-0.02,0.02)  (3.2,-0.4)
END
```

12. Change to the top directory (*/u0/lib/realchip/element_name)*:

    % **cd ..**

13. Execute the "addphysinfo" script:

    % **addphysinfo chips.dat**

    This script automatically updates the *chips.dat* file from the information in the *phys_dat* file. The addphysinfo script creates a number of files that begin with the characters "lib." The *liblst.dat* file contains a summary of execution; after an error-free run of the script, these files can be deleted.

14. After the *chips.dat* file has been updated, rename this file *element_name*.prt:

    % **mv chips.dat** *element_name*.**prt**

## 4.6 DEVICE TYPES

Both Realchip and Realmodel divide reference elements into three modeling classes or types; "dynamic," "static," and "static forever." How a reference element is modeled is determined by the device specifications and is declared in the device definition file. The following table shows a functional comparison for reference elements modeled as dynamic, static, and static forever devices.

**Table 4-2. Device Types**

| Function | Dynamic | Static | Static Forever |
|---|---|---|---|
| Pattern Memory Usage | Device Refreshing | Device Refreshing | Reset Sequence Only |
| Hardware Clock | Yes (increases length of Simulation) | No | No |
| Multiple Device Instances | Yes | Yes | No |
| Data Compression | Yes | Yes (strobed devices) | Yes (strobed devices) |
| Simulation Length: | | | |
| Realchip | 32K clock edges (typ) | 16K strobe edges (approx) | unlimited |
| Realmodel | 2M clock edges (typ) | 1M strobe edges (approx) | unlimited |
| Application | MPU's Peripherals | Semicustom Bipolar parts-multiple instances | "Semi-static" long simulation of bipolar parts |

## STATIC DEVICES

Static is the most general device type. For reference elements modeled as static devices, Realchip and Realmodel maintain a sequence of patterns that is used to repeatedly

recreate the environment of the model instance from the beginning of simulation to the current simulation time. This sequence is called the "environment sequence." Each time a simulated input or input/output of the model changes state, a reset sequence is appended at the beginning of the environment sequence, a sampling sequence is appended at the end, and the entire resulting sequence is presented to the model. The reset sequence causes the reference element to be initialized to a known internal state. The sampling sequence causes the hardware to sample the reference element's output and I/O pins. After the sampling sequence has been presented, the Simulator reads the sampling registers and calculates a signal value for each pin.

For reference elements modeled as static devices, a transition on an input or I/O pin does not necessarily result in adding a pattern to the environment sequence. If the transition is on a pin that cannot cause a permanent state change within the device, the last pattern in the environment sequence is simply altered to be consistent with the new state of the pin, the pattern sequence is presented, and the outputs are sampled. Details of the order of transitions of such pins are not preserved in the environment sequence.

Transitions on some pins may cause permanent state changes within the static device. This type of pin is called a "strobe pin," and is defined as "STROBE_PIN" in the device definition file. By definition, a non-strobe pin is any pin that can undergo the following test at any time without changing the internal state of the device, and a strobe pin is any pin which cannot:

- Establish any constant value on any other pin.
- Change the state of the pin under test any number of times with any timing.
- Return the pin to its original state.

An example of a strobe pin is the CLK input of an AM2901; an example of a non-strobe pin is the 'A' input of an AM2901.

When a strobe pin transitions, a new pattern is always added to the environment sequence; this new pattern causes only the strobe pin to transition (relative to the previous pattern) to guarantee ample setup and hold time of any other pins relative to the transitioning strobe pin (125 ns setup and hold time for both Realchip and Realmodel, with setup time increasing with the clock period).

Declaring pins as strobe pins when they really are non-strobe pins does not cause a model to produce incorrect results, but may cause much longer environment sequences and much lower rates of presenting transitions to strobe pins.

Note that the environment sequence preserves the order of transitions of strobe pins relative to each other and guarantees that the non-strobe pins have ample setup and hold time relative to the strobe pins. The exact sequence of non-strobe pin transitions is not preserved, however, and the time intervals between transitions of any pins (strobe pins or non-strobe pins) may be stretched.

Modeling a reference element as a static device allows multiple instances of the reference element to be used in the design. Since a separate pattern is required for each strobe pin transition (the hardware clock is not used), the maximum length of simulation is shorter than for a reference element modeled as a dynamic device; data compression is still used to optimize pattern memory. A primary advantage to modeling a reference element as a static device is that pin-to-pin (combinational) timing delays can be specified (in the delay block of the device definition file). Usually, reference elements that do not require refreshing (e.g., gate arrays) are modeled as static forever devices.

## DYNAMIC DEVICES

Reference elements that must be refreshed (microprocessors and NMOS devices) are modeled as dynamic devices. A dynamic device is a special case of a static device in which there is only one strobe pin (defined as the CLOCK_PIN in the device definition file) and all delays are referenced to this pin. The CLOCK_PIN is connected by a wire on the personality module. For this type of device, all transitions of pins other than the CLOCK_PIN are accumulated in a single pattern and are not presented to the device until a transition occurs on the CLOCK_PIN (for a CLOCK_BOTH type device, either transition causes the pattern to be presented; for a CLOCK_RISE or CLOCK_FALL type device, the transition must be a rising or falling edge, respectively). The accumulated pattern, which is timed by the hardware, is then presented to the device along with the proper transition on the CLOCK_PIN.

Reference elements modeled as dynamic devices do not require a pattern for the CLOCK_PIN transition and, accordingly, can have a longer simulation length. Also, adjacent patterns in the environment sequence are likely to be identical for dynamic devices and allow the repetition count to be used instead of storing separate patterns. Conversely, reference elements modeled as static devices never have identical adjacent patterns in the environment sequence and always require two new patterns for each strobe pin transition. Reference elements modeled as dynamic devices use pattern memory much more efficiently than static devices.

As a side effect of better pattern memory utilization, the maximum pattern presentation rates are higher for a reference element modeled as a dynamic device than if modeled as a static device. Although any reference element modeled as a dynamic device can theoretically be modeled as a static device with a single strobe pin, the resulting pattern presentation rate would be twice as slow. This consideration can be critical for reference elements that require high-speed clocks. However, many reference elements have low enough minimum clock-rate specifications and will continue

to function properly when modeled as a static device.

Some dynamic devices require multi-phase clocks. If the clock frequency requirement is low enough, such a device can be modeled as static with multiple STROBE pins. Otherwise, it may be possible to model it as dynamic device by using logic on the personality module to generate the necessary multi-phase clocks from the single incoming hardware clock.

## STATIC FOREVER DEVICES

A static forever device is a special case of a static device in which pattern memory is used only to present an initial reset sequence. With reference elements modeled as a static forever device, an environment sequence is not used; each time a transition occurs on an input or I/O pin, the latest pattern is presented to the device, the sampling sequence is presented, and the latest pattern is again presented and remains until the Simulator is ready with another transition. Once begun, the presentation of patterns continues at the same rate for static forever models as for static models (i.e., at intervals which are half the declared CLOCK_PERIOD) and, since an environment sequence is not recreated for each transition, there is no limit on the length of simulation.

Accordingly, reference elements modeled as static forever retain their internal state within the reference element itself while the Simulator computes between simulated transitions. Since the state is retained in the actual physical device rather than in the environment sequence, only one instance of a static forever device can be used in a design.

## SELECTION CRITERIA

A reference element can be modeled as a dynamic device if:

1. it has only one clock input
2. all inputs are sampled on a clock edge
3. all output transitions are triggered by a clock edge (the device clock is hard-wired to an external clock driver on the personality module).

If a reference element does not fit these criteria, it must be modeled as a static device even if the device includes internal dynamic memory. Dynamic device models are similar to static models in which the clock pin is the only strobe pin and where all output delays are referenced to the clock. Because of these restrictions, the Realchip/Realmodel hardware can simulate a device at a higher physical clock speed and for a longer time interval if the device is modeled as a dynamic device rather than as a static device. Dynamic models also allow separate float delays to be specified.

Static models have no such clock restrictions, but do not allow the optimizations used with dynamic device simulation. A truly dynamic device can be modeled using a "static" device model (defining the clock input as a STROBE_PIN); each clock transition would, however, require a separate pattern in pattern memory. The effective rate of pattern presentation to the device consequently would be degraded (possibly below the minimum-frequency specification of the dynamic device), and the effective capacity of pattern memory would be decreased. Similarly, a truly static device with no internal dynamic memory can be modeled more efficiently as a dynamic device if it has only one STROBE_PIN and all output transitions are effected by a transition on the strobe pin. Static models allow for multiple delays according to which input causes the transition, but do not allow float delays.

Static forever models, like static models, do not have the clock restrictions of dynamic models and, since pattern memory is used only for the reset sequence, offer unlimited simulation length. Devices that require no internal refresh can be modeled as static forever. Static forever models are limited to only one instance (occurrence) with a design.

## 4.7 DEVICE DEFINITION FILE

A device definition file is required for each reference element. Additionally, different speed versions for the same reference element may require separate device definition files if the timings differ in any significant way. Also, some devices operate in multiple "modes" (e.g., the Intel 8086 can operate in either MIN or MAX mode); if the modes of operation differ significantly, a separate device definition file must be created for each mode.

The device definition file is created with a text editor; the name of this file is the reference element name with a .def extension (e.g., *element_name*.def). The file must be located in the top directory of the reference element being modeled.

Before the device definition file can be created, the reference element must be mounted on the personality module to determine the pin mapping between the reference element and the personality module's interface pins. Appendices A through E describe the individual personality modules available and the pin mappings.

Figures 4-3 and 4-4 are examples of device definition files for dynamic and static devices.

```
primitive 'DYNAMIC_DEVICE';
number_dev_pins=128;

  pin
    input_spec =
      'CLK'       pin=12: nc_pin,
      '-RESET'    pin=6,
      'IN'<1>     pin=7,
      'IN'<2>     pin=74,
      'IN'<3>     pin=73,
      'IN'<4>     pin=72;

    io_spec =
      'IO'<1>     pin=11: delay=60 : float_delay=80,
      'IO'<2>     pin=10: delay=60 : float_delay=80,
      'IO'<3>     pin=9  : delay=60 : float_delay=80,
      'IO'<4>     pin=8  : delay=60 : float_delay=80;

    output_spec =
      'OUT'<1>  pin=68: delay=45,
      'OUT'<2>  pin=69: delay=30,
      'OUT'<3>  pin=70: delay=30 : float_delay=80 : ts_pin,
      'OUT'<4>  pin=71: delay=50 : output_type=(oc,and);

    {VCC is pin 75; GND is pin 13}
  end_pin;

  jig_id = 79;
  jig_type = dynamic;
  clock_pin = 'CLK';
  clock_edge = both;
  clock_period = 200-500;

  reset_seq
    '-RESET'      0,0,0,0,0,0,0,0,0,0;
  end_reset_seq;

end_primitive;
```

**Figure 4-3. Definition for a Dynamic Device**

```
primitive 'STATIC_DEVICE';

pin
   input_spec =
      'IN'<1>    pin=74,
      'IN'<2>    pin=73,
      'IN'<3>    pin=72,
      'IN'<4>    pin=71,
      'IN'<5>    pin=70,
      'IN'<6>    pin=69,
      'STB'      pin=68 : strobe_pin;
   io_spec =
      'IO'<1>    pin=6,
      'IO'<2>    pin=7,
      'IO'<3>    pin=8,
      'IO'<4>    pin=9;
   output_spec =
      'OUT'<1>  pin=12 : output_type=(oc,and),
      'OUT'<2>  pin=11,
      'OUT'<3>  pin=10;
   {VCC is pin 75; GND is pin 13}
end_pin;

jig_id = 3;
jig_type = static;

delay_table
   'IO'<1>, 'IO'<2>, 'IO'<3>, 'IO'<4>:
      'IN'<1> = 35,
      'IN'<2>, 'IN'<3> = 45,
      'IN'<4> = 50,
      'IN'<5>, 'IN'<6> = 40,
      'STB' = 30;

   'OUT'<1>:
      'IN'<1> = 45,
      'IN'<2>, 'IN'<3>, 'IN'<4> = 50,
      'IN'<5>, 'IN'<6> = 75,
      'STB' = 40;
```

**Figure 4-4. Definition for a Static Device**

```
  'OUT'<2>, 'OUT'<3>:
    'IN'<1> = 70,
    'IN'<2>, 'IN'<3>, 'IN'<4>, 'IN'<5>, 'IN'<6> = 60,
    'STB' = 50,
    'IO'<1>, 'IO'<2>, 'IO'<3>, 'IO'<4> = 45;
end_delay_table;

end_primitive;
```

**Figure 4-4. Definition for a Static Device (Con't)**

Each device definition file begins with the word "PRIMITIVE" followed by the "logical device" name. The logical device name must be identical to the reference element name of the corresponding body drawing used in the schematics. The name is enclosed in single quotes and is terminated by a semicolon. Single quotes embedded within a name are indicated by including two single quotes in succession. All names must be unique within the Library File (thus different speed versions of the same device may require body drawings of a different name and separate device definition files). The name of a device should, by convention, be the part number (e.g., '8086' or '68000'). When there is more than one version of a reference element, add a version suffix or prefix to the name (e.g., '8086X2' or '68000L8'). Note that all letters in a device name must be upper case and, since file names are constructed from this string, special characters should be avoided.

A device definition file is terminated with the word "END_PRIMITIVE" followed by a semicolon. Each device definition file, depending on the size and type of device (i.e., dynamic, static, or static forever), will include the following directives and directive blocks:

- **Number of Pins** -- The number of pins directive is required for all device types with 128 pins.

- **Pin Block** -- The pin block is common to all device types and specifies the mapping of reference element pins to personality module interface pins. This block is subdivided into input, output, and I/O sections

and also specifies the individual properties associated with each pin.

- **Jig ID** -- The jig ID directive is common to all device types and specifies the jig ID number strapped on the personality module.

- **Jig Type** -- The jig type directive is common to all device types and specifies how the reference element is to be modeled (dynamic, static, or static forever).

- **Clock Block** -- The clock block is common only to dynamic devices and specifies the clock edge, clock period, and clock pin (clock period is also required for static and static forever models).

- **Reset Sequence Block** -- The reset sequence block is common to all device types and defines the series of input patterns applied to the reference element to bring it to a known initial state.

- **Delay Table Block** -- The delay table block is common only to static and static forever devices and specifies the delays associated with each output pin based on input transitions.

The following subsections detail the above sections within the device definition file.

## NUMBER OF PINS DIRECTIVE

The number of pins directive is required when the number of personality module interface pins for a reference element is greater than 64 (if this directive is omitted, 64 pins are assumed). The directive begins with the word "NUMBER_DEV_PINS" followed by an equal sign, the number of pins specified as an integer in multiples of 64, and a semicolon (e.g., NUMBER_DEV_PINS=128;).

## PIN BLOCK

The pin block defines the association between the pins of the reference element and the interface pins of the personality module. This block also specifies the various properties associated with each device pin. The pin block begins with the word "PIN" and ends with the word "END_PIN" followed by a semicolon. The pin block must precede any "RESET_SEQUENCE" (dynamic devices) or "DELAY_TABLE" (static and static forever devices) entries.

The pin block can contain any of the following sections:

- The INPUT_SPEC section contains definitions for pins that are exclusively inputs.
- The OUTPUT_SPEC section contains definitions for pins that are exclusively outputs.
- The IO_SPEC section contains definitions for bidirectional pins.

Each section begins with one of the three keywords listed above followed by an equal sign and a list of individual pin specifications. Each pin specification is separated by a comma; the last specification in a section is terminated by a semicolon. Specification blocks may occur in any order.

Each pin specification consists of a pin name and its corresponding pin number on the personality module, and a list of properties for the pin. The pin name and pin number are required for every device pin; a pin property may or may not be necessary depending on the nature of the pin. The text portion of the pin name, which is always the first field in a pin specification, is enclosed in single quotes. If the pin has a bit number, the bit number is enclosed in angle brackets and follows the quoted text portion of the pin name. The full pin name (with bit number) must be identical to the name of the corresponding pin of the body drawing with the exception that bit ranges are not permitted (only single-bit subscripts are allowed) and low-asserted pins are noted with a dash prefix (inside the quotes) rather than a '*' suffix. The pin name must

conform to the SCALD signal-name syntax rules as described in *SCALD Language Reference Manual.* The following are examples of pin names:

'-RESET'
'DATA IN'<12>
'ENABLE'

The pin name is followed by its corresponding personality module pin number and, if required, one or more pin properties. The pin number and the pin properties are separated by colons. Each pin entry is terminated by a comma; the last pin entry in a section is terminated by a semicolon.

The pin number begins with the word "PIN" followed by, in order, an equals sign and the pin number of the personality module interface pin that connects to the corresponding reference element pin (refer to the appropriate appendix for the personality module being installed).

**Pin Properties**

The following pin properties are used within a pin block:

- DELAY. The DELAY property (e.g., "DELAY=10") defines the delay (in nanoseconds) between the clock edge that causes a transition on the pin and the occurrence of the transition in the simulated design. Worst-case timing models use maximum times for DELAY. If DELAY is not specified, the FLOAT_DELAY value is used; if FLOAT_DELAY also is not specified, '0' is used. The DELAY property applies only to output and I/O pins of dynamic devices. Delays for static and static forever devices are defined in the delay table block (see below).

- FLOAT_DELAY. The FLOAT_DELAY property (e.g., "FLOAT_DELAY=10") defines the delay (in nanoseconds) between the clock edge that causes the pin to change to the high-impedance ('Z') state and the occurrence of the change in the simulated

design. The FLOAT_DELAY value is used only for transitions to the high-impedance state. Worst-case timing models use minimum times for FLOAT_DELAY and maximum times for DELAY. If FLOAT_DELAY is not specified, the DELAY value is used for this type of transition (if DELAY also is not specified, '0' is used). The FLOAT_DELAY property applies only to output and I/O pins of dynamic devices.

- TS_PIN. The TS_PIN property identifies an output pin that can assume the high-impedance state and enables the Realchip/Realmodel hardware to sense high-impedance when sampling the value of the pin. This property applies only to output pins; by default, all I/O pins are assumed to be tri-state and are sampled accordingly. Note that this property can not be used with output pins that have a high-state drive current of less that 0.400 milliamps. (If the output drive current is less than 0.400 milliamps, the system may report that the pin is in its high-impedance mode when it is actually a high-state output.)

- NC_PIN. The NC_PIN property instructs the Logic Simulator to ignore the identified pin in the determination of identical stimulus patterns and is used for the clock input to a dynamic device that uses the personality module's 'CK' input.

- STROBE_PIN. The STROBE_PIN property indicates an input or I/O pin on which transitions can cause a change of state within the device. Examples of such pins include clock lines, RAM write enables, and latch enables. The STROBE_PIN property applies only to static and static forever devices; for dynamic devices, the single clock input is effectively treated as the only "strobe" pin.

  Accurate identification of strobe pins is critical to the proper operation of static and static forever device models. Any input that causes a change of state within a device is a strobe pin. Accordingly, any clock pin, write enable, asynchronous reset signal, or

any signal that independently sets or resets a status bit within the device is considered a strobe pin. For example, the Intel 8274 has 14 strobe pins.

- 'CLK', '-TxCa', '-RxCa', '-TxCb', and '-RxCb' are all clock lines.
- '-WR' causes an 8274 command or data register write.
- '-INTA' starts an 8274 interrupt sequence.
- '-RESET' causes the 8274 to reset.
- '-CTSa', 'CDa', '-CTSb', 'CDb', '-SYNDETa', and '-RTSb' all can set bits in the 8274 status registers.

Note that all of these pins are strobes in the most general sense. When the device is operating in a particular mode, all of these pins may not function as strobes. Again, it may be advantageous to customize device models to fit the particular application.

A pin that causes a state change only if asserted during a clock edge or other signal transition is not a strobe pin (the other pin is the strobe). When several pins must be asserted to cause a state change, the strobe signal is the last to be asserted or the first to be deasserted (generally one and the same). Similarly, a write enable signal can be the strobe for some devices and a chip select signal for others.

- OUTPUT_TYPE. The OUTPUT_TYPE property specifies general output-type properties to the Logic Simulator (see "Properties Affecting Simulation" in *Logic Simulator Reference Manual* and the *Library Reference Manual*). The only relevant property value is "OUTPUT_TYPE=(OC,AND)"; including this property in the list of properties of an output or I/O pin causes any '1' or 'Z' sensed by the Realchip/Realmodel hardware to be converted to a

"soft-1" by the Logic Simulator (i.e., as if the driver was a pulled-up, open-collector TTL output). The '0' state is unaffected by the Logic Simulator. Any I/O pin that has a low-state input leakage current in excess of 0.250 mA must be labeled as an "(OC,AND)" pin (otherwise when the state of the pin is sensed by the hardware, a "hard 1" may be reported when the pin is actually in input mode). Similarly, any I/O pin that has a high-state drive current of less than 0.400 milliamps must be assigned the "(OC,AND)" property to prevent the system from reporting that the pin is in its high-impedance mode when it is actually a high-state output.

**Bubbled Pins**

The Simulator always inverts bubbled inputs or outputs before delivering them to (or after reading them from) the Realchip or Realmodel driver software. Furthermore, the '-' preceding a pin name in the device definition file causes Realchip or Realmodel driver software to re-invert the value of the corresponding input before it is delivered to the reference element, or to re-invert the value of the corresponding output after it is sampled at the reference element.

It is critically important that these inversions cancel for each pin. Therefore the model builder must follow these rules:

1. Every "bubbled" pin on the body drawing for a Realchip or Realmodel model must have a low-asserted pin name, and every "non-bubbled" pin must have a high-asserted pin name.

2. In the device definition file, names for low-asserted pins must be preceded by '-', and names for high-asserted pins must not be preceded by '-'.

## Choosing Delay Values

When choosing values for delay times, the model builder must use values that are allowed by the device specifications. Ideally, the delay value chosen should represent "worst case" (i.e., maximum delay if the transition is to a stable state and minimum delay if the transition is to the high-impedance state). Using worst-case delays tends to reveal most design errors. However, some pins may have output delays that depend either on input values or the state of the device. In these cases, the model builder should use the maximum (or minimum) delay that applies to all possible situations.

The models provided by Valid are, as a rule, as general as possible. This generalization occasionally requires that approximations be made so that the delays are within the allowable limits for all states and modes of the device. Within this constraint, Valid models reflect worst-case timing as defined above. The user should make any changes in the models that will more accurately reflect device behavior for the modes of interest.

Occasionally, all of the output transitions on a dynamic device are explicitly referenced to the device clock in the manufacturer's data sheet. If not, the model builder must DEDUCE a valid (i.e., within specification) delay for each output relative to a clock edge. A delay of 0 should not be used since it can cause some reference elements to go into an infinite loop; since minimum delays are rarely specified, a delay of 1ns should be used in place of a "zero delay."

The timing specification for a static device output can have several delays, each one relative to a particular input or set of inputs. If a delay for every input-output signal pair were specified for a large device (e.g., an AMD 2960), the model would be unmanageable. The model builder usually needs to make some approximations (e.g., grouping nearly-identical delays together) in order to model nearly worst-case timing while keeping the model simple.

The Delay Table block is used only for static and static forever models (i.e., models that do not use the hardware clock) and gives the delay for each output from each input.

For static or static forever devices, Realchip and Realmodel always present input or I/O transitions one at a time. Whenever a transition is detected on a device output in response to a particular input transition, Realchip and Realmodel infer the delay of the output transition by examining the delay table and finding the delay appropriate to that particular input/output pair. If the delay table does not include the appropriate delay, then the transition is scheduled with 0 delay.

The direction of input or output transitions is not considered when examining the delay table. Every transition of a given output from a given input is always scheduled with the same delay, whether the output transitions to 0, 1, or Z, from 0, 1, or Z.

For dynamic models, outputs are assumed to change only on clock edges, therefore all delays are relative to clock edges. The output delays for dynamic models are specified in the Pin block of the device definition file; a delay table is not required.

The models at the end of this section are intended as examples. The model builder is encouraged to reference these and any other models in the Valid Realchip Library when creating models.

## JIG ID DIRECTIVE

The JIG_ID directive entered in the device definition file must be the same as the ID number hard-wired on the personality module. The Logic Simulator uses this number to link a particular reference element to its corresponding device definition file. Multiple device definition files (e.g., different speed versions of the same device) can have the same jig ID number, and consequently are associated with the same reference element. The JIG_ID directive begins with the word "JIG_ID" followed, in order, by an equals sign, an integer between 1 and 126 for Realchip or between

1 and 126 or between 128 and 254 for Realmodel, and a semicolon terminator (e.g., JIG_ID =123;). Note that jig ID numbers 127 and 255 are reserved and cannot be used.

**NOTE**

Realchip and Realmodel devices supplied from Valid are assigned jig ID numbers in ascending order. To avoid possible duplication of jig ID numbers, user-created models should be assigned in descending order from 126 (Realchip) or 254 (Realmodel).

## JIG TYPE DIRECTIVE

The JIG_TYPE directive specifies the type of device (dynamic, static, or static forever). The Logic Simulator treats each device type differently. Also, the contents of the device definition file vary according to the jig type. The JIG_TYPE syntax is the same as the JIG_ID syntax except that the keyword is "JIG_TYPE" and the type argument is either "STATIC," "STATIC_FOREVER," or "DYNAMIC" (e.g., JIG_TYPE = STATIC_FOREVER;).

## THE CLOCK BLOCK

The clock block includes three directives that specify clock input properties for reference elements modeled as dynamic devices; the CLOCK_PERIOD directive is also required with devices modeled as static and static forever.

- CLOCK_PIN -- identifies the clock input signal.
- CLOCK_PERIOD -- specifies the permissible range of periods for a full clock cycle for the physical reference element.
- CLOCK_EDGE -- specifies which edge or edges of the clock signal are to be used as the timing reference.

The syntax for all three directives is identical to JIG_ID. A legal clock pin is the name of an input signal (enclosed in single quotes) as defined in a pin specification. A legal clock period is a pair of integers separated by a dash (e.g., 200-500) that represent the range of the period in nanoseconds. A legal clock edge is either "RISE," "FALL," or "BOTH" and references the rising edge only, the falling edge only, or both edges of the clock. An example of these clock input directives is:

```
CLOCK_PIN = 'CLK';
CLOCK_PERIOD = 200-500;
CLOCK_EDGE = BOTH;
```

Both the Realchip and Realmodel hardware support a fixed set of clock periods with a minimum clock period of 480 ns for a "BOTH" clock or 320 ns for either a "RISE" or "FALL" clock. The Logic Simulator selects the smallest supported clock period consistent with the CLOCK_PERIOD property. If the specified clock period range does not overlap a supported clock period, the Logic Simulator issues an error message.

When a CLOCK_PERIOD is specified for a static device, the smallest supported clock period within the specified range is used as the pattern-delivery clock (one pattern delivered each clock period). When a CLOCK_PERIOD is not specified, patterns are delivered to static devices at a rate of approximately 1 million per second.

## THE RESET SEQUENCE BLOCK

In order for successive samplings of device outputs and I/Os to form a coherent history of device operation, static and dynamic devices must be forced into a known initial state before presenting the environment sequence. For this reason, the model builder specifies a reset sequence in the device definition file. The reset sequence can be any sequence of patterns; it is always presented immediately before the environment sequence for static and dynamic models, and is presented once at the beginning of simulation for static forever models.

A reset-sequence block begins with the word "RESET_SEQ" and contains a reset sequence for each relevant pin in the reset cycle. Each pin reset sequence is terminated by a semicolon; the reset-sequence block is terminated by "END_RESET_SEQ" followed by a semicolon. Each pin reset sequence begins with a pin name in single quotes (the pin name specified must have already been defined in a pin block). The pin name is followed by a list of signal values ('0', '1', or 'Z') to be applied to the pin on sequential clock edges as defined by the CLOCK_EDGE directive (if CLOCK_EDGE=BOTH, the first pattern of the reset sequence is presented on a rising clock edge). All of the pin reset sequences within a block must be the same length. A '0' is continuously applied to all pins not specified in the block. The signal values in the list are separated by commas and the sequence for each pin is terminated by a semicolon. The following example illustrates a reset sequence.

```
RESET_SEQ
        'PIN'<1>   1,1,1,1,1,1,1,1,1,1;
        'PIN'<2>   1,1,1,1,0,0,0,0,0,0;
        'PIN'<3>   0,0,0,0,Z,Z,Z,Z,Z,Z;
                        .
                        .
END_RESET_SEQ;
```

When the design under test contains a functional reset circuit, simulation of the design itself automatically builds up an environment sequence to initialize the reference element (the sequence of sampled outputs during the early period when reset is not complete may be anomalous). In this case, the reset sequence specified in the device definition file could be considered superfluous. However, the reset sequence in the device definition file helps prevent anomalous results in cases where an external reset circuit is not included or does not operate properly.

For dynamic models with CLOCK_EDGE=BOTH, Realchip and Realmodel always present the first pattern of the reset sequence on (i.e., just prior to) a rising edge of the device clock, and present subsequent patterns on alternating edges.

Some devices have internal states that cannot be initialized by a fixed sequence of patterns. For example, the 'E' output of the M68010 has fixed period, but indeterminate phase after the M68010 is fully reset. If a fixed reset sequence were used to initialize the M68010, the sampled E output would be anomalous (not even periodic).

To allow complete initialization of such devices, Realchip and Realmodel support limited feedback. To use feedback, a fixed "feedback output" of the device is hardwired to a dedicated pin of the personality module, and a flag is specified for each pattern in the reset sequence. During presentation of the reset sequence, an appropriately flagged pattern causes the hardware to pause while continually presenting transitions on the hardware clock, but presenting the same "PAUSE" pattern to the reference element until an appropriate level or transition is sensed on the feedback output. At that point, presentation of the remaining patterns in the reset sequence commences. Feedback cannot be used for static or static forever models since these models do not have a hardware clock to advance the internal state of the reference element.

To use feedback, a "PAUSE" sequence is included in the reset sequence block. The PAUSE sequence follows the pin reset sequence syntax with the exception that the single quotes are omitted from around the word "PAUSE." The PAUSE sequence directs the control hardware to continue to present the current input pattern to the reference element until the feedback signal changes state.

The flags in the PAUSE sequence can be '0', '1', or 'Z'. For Realchip, a 'Z' flag in the PAUSE sequence specifies that no pausing should occur, and a '0' or '1' flag specifies that the system wait for either a 0 or a 1 level, respectively, on the feedback output (i.e., a rising transition can be detected by first pausing for a 0, then pausing for a 1).

For Realmodel, the 'Z' flag has the same meaning (i.e., "don't pause"). However, '0' also means "don't pause," and '1' means "wait for a rising transition."

After sensing the appropriate level or transition on the feedback output, Realchip and Realmodel continue presenting the PAUSE pattern to the reference element while internal pipelines are flushed. For Realchip, the flagged pattern is presented once more after the clock edge that causes the transition of the feedback output to the desired level (or after the first presentation of the PAUSE pattern, in case the level already existed). For Realmodel, the flagged pattern is presented five more times after the clock edge that causes the rising transition on the feedback output.

The following Realchip reset sequence block example illustrates the feedback sequence.

```
RESET_SEQ
        'PIN'<1>   1,1,1,1,1,1,1,1,1,1;
        'PIN'<2>   1,1,1,1,0,0,0,0,0,0;
        'PIN'<3>   0,0,0,0,Z,Z,Z,Z,Z,Z;
        PAUSE      Z,Z,Z,0,1,Z,Z,Z,Z,Z;
                   .
                   .
                   .
END_RESET_SEQ;
```

In the example, the PAUSE sequence causes:

1. The first four patterns to be presented to the reference element.

2. The fourth pattern to be maintained until the feedback signal goes to a '0' before presenting the fifth pattern.

3. The fifth pattern to be maintained until the feedback signal goes to a '1' before presenting the final five patterns.

Because Realchip and Realmodel continue presenting the PAUSE pattern for different lengths of time before continuing the reset sequence, the actual waveforms produced by Realchip and Realmodel may differ slightly (Realchip waveforms may be offset slightly toward greater time relative to Realmodel waveforms). This difference can often

be eliminated by adding dummy patterns to the end of the reset sequence.

Note that looking for a single event in Realchip may cause confusing results because the appropriate level may already exist when the first PAUSE pattern begins to be presented. The correct workaround for Realchip is to look for pairs of events representing a transition. On the other hand, looking for two or more feedback events in a single PAUSE sequence may cause confusing results if the second event occurs while flushing the first event out of the pipeline. This second problem is of particular importance for Realmodel, which has a long pipeline. Since Realmodel feedback events are transitions rather than levels, there is never any reason to look for two feedback events with Realmodel.

Model builders at Valid have observed that some devices require the reset signal be asserted longer than the time specified by the device manufacturer to complete the full reset cycle (incomplete resetting of a device can cause anomalous results). It is prudent to add some margin to the published reset time. As a general rule, dynamic devices that require feedback should be reset twice by the patterns in the reset sequence: the first reset starts the device (the PAUSE sequence waits for the device to reach a known state), and the second reset restarts the device (causing the device to end at a known state). Simulated time 0 begins at the end of the full reset sequence. The function of the second reset in the sequence is simply to ensure that simulated time 0 finds the device just finishing reset; if the second reset were not present, simulated time 0 would find the device having already operated for some number of cycles after the end of reset.

Because of pipeline delays within the hardware, the pattern following a "paused" pattern is not presented to the reference element until several cycles after the feedback signal has attained its designated state. During these cycles, the paused pattern continues to be presented to the reference element.

The feedback signal must be a two-state output capable of driving at least one TTL load.

## THE DELAY TABLE BLOCK

The delay table block defines the delays associated with each output in a static or static forever reference element and allows for different output delays based on the input transition responsible for the output transition. A delay table begins with the word "DELAY_TABLE" and ends with the word "END_DELAY_TABLE" followed by a semicolon.

Each delay table entry consists of a list of output (or I/O) signals followed by one or more lists of input or I/O signals. Each signal name in the table must have been defined previously in the pin block and must be enclosed in single quotes. The output signal list elements are separated by commas and terminated by a colon. The input signal list elements also are separated by commas. Each input list is terminated with the corresponding delay (an equals sign and integer value) followed by a comma; the last input list is terminated by a semicolon. The delay value represents the input-to-output delay in nanoseconds.

The delay table in the following example specifies the delays to the 'Y' outputs from various inputs.

```
delay_table

  'Y'<0>, 'Y'<1>, 'Y'<2>, 'Y'<3>:
    'A'<0>, 'A'<1>, 'A'<2>, 'A'<3> = 35,
    'B'<0>, 'B'<1>, 'B'<2>, 'B'<3> = 40,
    'D'<0>, 'D'<1>, 'D'<2>, 'D'<3> = 30,
    'CN' = 22,
    'I'<0>, 'I'<1>, 'I'<2>, 'I'<3>, 'CP' = 35,
    'I'<4>, 'I'<5>, 'I'<6>, 'I'<7>, 'I'<8> = 25,
    '-OE' = 23;

end_delay_table;
```

The Logic Simulator uses the delay values in this block to derive all timing information required for the simulation.

When an output transition occurs on a Realchip/Realmodel reference element output, the Logic Simulator schedules the event to occur at the time of the input transition plus the delay specified for that output pin. Note that the actual delays that occur within the reference element have no affect on simulation timing.

**NOTE**

A missing delay table entry for an output (or I/O) pin causes the output (or I/O) not to be scheduled by the Simulator even when a transition has occurred.

## 4.8 RUNNING THE MAKEALLMSPRIM SCRIPT

After the device definition file is created, it must be added to the allmsprim.def file in */u0/lib/realchip*. To run this script:

1. Change to the realchip directory:

   % **cd /u0/lib/realchip**

2. Run the script:

   % **makeallmsprim**

3. The makeallmsprim script will announce that it is creating the allmsprim.def file and will list each of the elements included in this file.

Note that the script must be run from */u0/lib/realchip* and that it must be run whenever a new device is added and whenever a device definition file is changed.

## 4.9 VERIFYING THE MODEL

It is easy to make errors in the body drawing, the library and device definition files, and even in the construction of the personality module. An error in any one of these areas can cause the model to produce erroneous results. After building a model, it is recommended to verify the model by

developing a test circuit using the model and then running a simulation session.

Each Realchip or Realmodel system is shipped with an Intel 8086mx personality module and complete software for verifying system operation. The software is located in the */u0/lib/realchip/8086mx* directory and includes the directories and files outlined in table 4-3.

**Table 4-3. 8086mx Directory Contents**

| Name | Description |
|---|---|
| 8086mx | The directory containing the 8086mx .BODY and .PRIM drawings and the chips_prt and phys_dat files. |
| 8086mx.def | The device definition file for the 8086mx. |
| 8086mx.exp | The compiler expansion file. |
| 8086mx.prt | The part file for the 8086mx. |
| 8086mx.syn | The compiler synonyms file. |
| 8086mxadapter | The directory containing the component- and circuit-side drawings of the 8086mx personality module. |
| 8086mxlibrary | The directory containing the 8086mx library drawing. |
| 8288 | The sub-directory that describes the Intel 8288 bus controller chip used in the test circuit. |
| demo | The directory containing the simulation test (demo) circuit. |
| demo.wrk | The SCALD directory used for developing the 8086mx test circuit. |

**Table 4-3. 8086mx Directory Contents (Con't)**

| Name | Description |
|---|---|
| demopgm.txt | The test (demo) program. |
| mleve.dat | The memload file for loading the even data byte. |
| mlodd.dat | The memload file for loading the odd data byte. |
| mslogic.lib | The SCALD directory for compiling for logic. |
| msscript.demo | The simulator script file. |
| msscript.notes | The text file describing the test program. |
| simcmd.dat | The simulator command file. |
| simlog.dat | The Simulator log file. |
| simlst.dat | The Simulator listing file. |
| simulate.cmd | The Simulator directives file. |
| startup.ged | The Graphics Editor command file. |

## CREATING THE TEST CIRCUIT DRAWING

The test circuit includes the reference element being modeled and support circuitry that, when simulated, provides the user with sufficient data to verify proper operation of the model. The test circuit is created in the demo.wrk directory and for consistency, should be named "demo."

## CREATING THE TEST PROGRAMS

The test program (*demopgm.txt*) and simulator script (*msscript.demo*) as well as any memload files must be developed to simulate the test circuit.

## COMPILING FOR SIMULATION

Before the test circuit can be simulated, the test circuit must be compiled for simulation. To compile the test circuit:

1. Copy the *mslogic.lib* file to *mssim.lib*:

   % **cp mslogic.lib mssim.lib**

2. Edit the *mssim.lib* file and change the file type line to read:

   FILE_TYPE=SIM_DIR;

3. Edit the *compiler.cmd* file and add or change the following lines to read:

   root_drawing 'demo';
   compile sim;
   directory 'demo.wrk', 'mssim.lib';
   library standard, *required libraries*;
   output list, expand, synonym;

   Note that the *mssim.lib* file must be referenced in the *compiler.cmd* file for all subsequent designs that simulate the reference element. To reference this file, include the line

   directory '/u0/lib/realchip/*element_name*/mssim.lib';

   in the compiler's directives file.

4. Compile the test circuit:

   % **compile**

## SIMULATING THE TEST CIRCUIT

Before the test circuit can be simulated, the *simulate.cmd* must be edited to add or change the following lines:

```
compiler_output 'element_name.exp';
synonym_file 'element_name.syn';
realchip_library '/u0/lib/realchip/allmsprim.def';
output list, command_log
session_log on;
```

After editing the *simulate.cmd* file, simulate the test circuit by typing:

```
% simulate
```

## INTERPRETING THE RESULTS

The Simulator's listing file (*simlst.dat*) includes a number of error messages specific to Realchip/Realmodel. A majority of these messages are associated with the device definition file. If an error is made in this file, edit the file to correct the error and rerun the "makeallmsprim" script to update the *allmsprim.def* file before simulating the circuit.

The Realchip/Realmodel software drivers ordinarily provide only '0', '1', and 'Z' values of output and I/O pins for use by the Logic Simulator. However, if an output or I/O pin has anomalous behavior during the sampling sequence (e.g., if it changes state unexpectedly), the drivers report 'U' (unknown). Any 'U' values observed on output or I/O pins of a reference element during normal operation indicate a probable problem with the model.

## 4.10 MODELING CONSIDERATIONS

The following sections outline basic principles of pattern presentation and sampling that must be considered when modeling a reference element for Realchip or Realmodel.

## PATTERN PRESENTATION

Realchip and Realmodel both consist of a fixed-length pattern memory that is capable of presenting patterns to reference elements at a high rate. Realchip pattern memory can hold up to 4K 64-pin patterns. Realmodel pattern memory can hold up to 256K 64-pin patterns. These pattern-memory capacities are effectively increased by the use of encoding and a 4-bit repetition count in each pattern.

Each pin of a reference element can be driven to 0, 1, or Z (LS125 output levels) by Realchip or Realmodel pin electronics. This drive capability requires two bits per pin in each pattern. The maximum pattern presentation rate for Realmodel is one per 125 ns, and for Realchip is one per 375 ns; the CLOCK_PERIOD directive in the device definition file file can be used to reduce the pattern presentation rate.

Realchip allows up to 16 reference elements to be connected to a single pattern memory, and Realmodel allows up to 64 reference elements. However, only one pattern sequence can be presented at a time, and it can be presented only to a single reference element. A design that contains multiple VLSI devices is simulated by stimulating and then sampling each device separately whenever an input or I/O pin of the device changes in the simulated design. The control software automatically handles switching between reference elements.

While patterns are being presented to one reference element, a fixed pattern can be held on each of the other reference element in the system. Accordingly, the internal state of static reference elements can be maintained while presenting patterns to other reference elements.

A single large reference element can span the physical mounting sockets of several smaller reference elements. In Realchip, two 64-pin sockets on the same device control module can be ganged to serve a single 128-pin reference element. In Realmodel, up to 24 64-pin sockets on six device control modules can be ganged.

When a new pattern is presented to a reference element, the different pins of the device may attain their new states at different times. In fact, Realchip presents the two halves of a 64-pin pattern at different times (i.e., 32-pins each, separated by at least 125 ns). For multi-socket reference elements, the skew between individual pins within a single pattern can be even greater.

Because of the undefined skews associated with presenting patterns to pins of a reference element, Realchip and Realmodel software never combine transitions of multiple inputs or I/Os within a single pattern unless the order of the transitions is irrelevant to the correct operation of the device. For example, in presenting patterns to an AM2901, a transition on the "CLK" pin is never presented in the same pattern as a transition on the 'A' pin.

Even while preserving the order of transitions presented to the reference element, Realchip and Realmodel will arbitrarily stretch the intervals between these transitions. In fact, transitions that occur simultaneously in the simulated design (e.g., on multiple clock pins of a model) may be presented to the reference element at different times, even separated by multiple pattern periods.

The general effect of this stretching of intervals between transitions is to increase setup and hold times that have been provided by the designer in the design being simulated. If the designer has provided negative or zero setup and hold times, the result of stretching intervals between transitions will almost certainly be in violation of the specifications for the device and will produce anomalous results from Realchip and Realmodel. Thus designers using Realchip and Realmodel are not permitted to take advantage of non-positive setup or hold times.

A hardware clock is provided for dynamic models. Realchip guarantees that the entire pattern associated with a clock edge is set up at least 125 ns before the hardware clock edge and that it is held for at least 125 ns after the clock edge. Realmodel guarantees 100 ns and 25 ns, respectively. As the pattern presentation period increases, the setup time increases correspondingly for both Realchip and Realmodel, but the hold time does not increase.

Similarly, for models that do not use the hardware clock (i.e., static and static forever models), the separation between individual transitions presented to pins of the device is guaranteed to be at least 125 ns for both Realchip and Realmodel.

For dynamic models, patterns are presented one setup time before both the rising and falling edges, just the rising edge, or just the falling edge of the hardware clock, depending on the value of the CLOCK_EDGE directive in the device definition file (BOTH, RISE, or FALL, respectively). In any case, the Realchip or Realmodel system attempts to select a hardware clock period that lies within the range specified in the CLOCK_PERIOD directive. If the system cannot find an appropriate clock period in the set of available clock periods, an error message is produced.

For models that are not dynamic, the interval between presentation of patterns is set to be half the clock period specified in the device definition file, if such a pattern presentation period is available, otherwise an error message is produced.

## SAMPLING

Realchip and Realmodel use essentially the same method to sample device pins. Sampling occurs whenever the Realchip or Realmodel software needs to determine the state of device pins after presenting some input transition to the device. For static models, sampling occurs after presentation of the entire saved sequence of previous input patterns. For dynamic models, sampling occurs after all of the previous input patterns have been presented AND after presentation of the last clock edge following the last pattern. For static forever models, individual new input patterns are presented without replaying any previous patterns; in this case, sampling occurs after presentation of each new input pattern.

Sampling is accomplished using the circuit shown in Figure 4-5. One instance of this circuit is connected to each device pin. First, consider only three-state output pins (i.e., pins listed in the OUTPUT_SPEC section of the device

definition file that are flagged TS_PIN). The following sequence of patterns is applied to the circuit shown in Figure 4-5 simultaneously for each such three-state output pin.

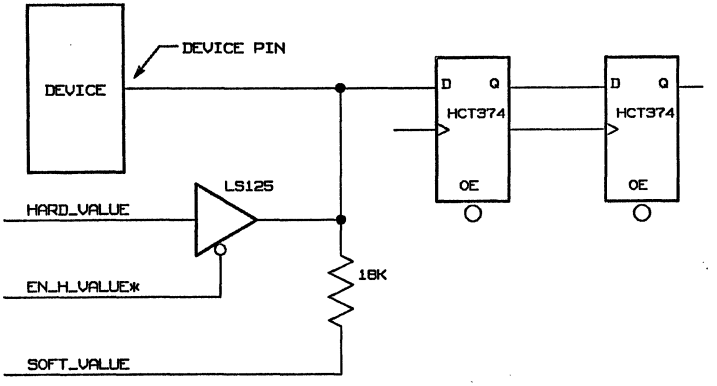


**Figure 4-5. Sampling Circuit**

| Step | Soft_Value | EN_H_Value* | Hard_Value |
|---|---|---|---|
| 1 | +5 V | 0 | 1 |
| 2 | +5 V | 1 | 1 |
| 3 | +5 V | 1 | 1 |
| 4 | -5 V | 0 | 0 |
| 5 | -5 V | 1 | 0 |
| 6 | -5 V | 1 | 0 |

For dynamic models, these patterns are presented at the same timing (i.e., same clock rate and phase) as the environment sequence; in fact they are simply concatenated onto the environment sequence. For static and static forever models, the sampling patterns are presented at intervals of half the CLOCK_PERIOD specified in the device definition file and follow the last normal input pattern by the same amount.

At steps 3 and 6, the device output pin is sampled by a 1-bit-wide-by-2-bit-deep shift register with HCT inputs. The sampling occurs at the normal time for the edge of the hardware clock associated with the pattern (i.e., the previous pattern in the sampling sequence has been setup for at least 125 ns in Realchip, and at least 100 ns in Realmodel).

In summary, the sampling sequence for a single three-state output pin hard drives to 1, soft drives to 1, samples (sample #1) while continuing soft driving to 1, hard drives to 0, soft drives to 0, and samples (sample #2) while continuing soft driving to 0.

After the sampling sequence is fully presented, all pins of dynamic and static devices are driven to soft 1. For static forever devices, the last pattern of the environment sequence is presented again and remains until the the Simulator is ready to use the model again.

Pure outputs (i.e., pins in the OUTPUT_SPEC section of the device definition file that are not flagged TS_PIN) have the same sequence of patterns applied (at the same time), but with EN_H_VALUE* always set to 1. Thus pure outputs are not overdriven during sampling.

Each pure input pin (i.e., pins in the INPUT_SPEC section of the device definition file) has a different sequence applied (at the same time), in which HARD_VALUE is the same value as was presented to the pin in the last pattern of the environment sequence. Thus valid constant levels are presented to pure inputs during sampling.

I/O pins (i.e., pins in the IO_SPEC section of the device definition file) are sampled the same way as three-state outputs (and at the same time). Note that I/O pins are therefore normally hard driven during sampling.

These two 1-bit samples are read by the software and used to compute a value for each sampled OUTPUT or I/O pin as follows:

| SAMPLE 1 | SAMPLE 2 | OUTPUT STATE |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | U |
| 1 | 0 | Z |
| 1 | 1 | 1 |

'U' is anomalous and should never occur if the device definition file and the hardware are both correct. 'U' can indicate problems in the device definition file (see below).

Any pin flagged "(OC,AND)" in the device definition file will have the value 1 or Z converted to a "soft 1." Otherwise, the "(OC,AND)" flag has no effect in Realchip and Realmodel.

The reset sequence and environment sequence are presented to the device (for dynamic and static devices only; static forever devices simply have the two sampling sequences presented in succession, and then the final input pattern is presented again and left indefinitely).

## THE SAMPLE=SPECIAL DIRECTIVE

For some devices modeled with Realchip, precharging I/O pins that are in the output state or even precharging three-state output pins using an output from the LS125 can cause other device pins to be incorrectly sampled. Including the directive "SAMPLE=SPECIAL" in the device definition file (in the same area as the JIG_ID and other global directives) causes precharging to be eliminated for the sampling of all pins. In this case, steps 1 and 4 of the sampling sequence in the previous section have EN_H_VALUE* set to 1, and each of these steps continues for 1.0 microsecond to allow the attached 18K resistor to pull the device pin up or down to the appropriate value.

The SAMPLE=SPECIAL directive is only supported by Realchip and should only be used with reference elements modeled as static or static forever (reference elements modeled as dynamic may lose their internal state by the time sampling is complete). This directive can always be included with static and static forever models without causing any adverse side effects.

## SAMPLING RULES

The following rules related to the sampling method have been useful in custom model building:

1. For static forever devices, an I/O pin should never also be a strobe pin. Driving the I/O pin during sampling may cause the device to change state, and the change of state will affect all later operation of the device.

2. Verify that any three-state output or I/O pin (i.e., any pin hard driven during sampling) is adequately buffered within the device so that temporarily forcing the pin to an artificial level will not affect the state of any other pin or the state of the device, and will not damage the device.

3. Any I/O pin that has a low-state input leakage current in excess of 0.250 mA should be flagged "(OC,AND)," otherwise a "hard 1" may be returned when the pin is actually in the input state. Similarly, any pin that has a high-state drive current of less than 0.400 mA should be flagged "(OC,AND)" so that Z will not be reported when the pin is actually a high-state output.

4. Realchip and Realmodel are not able to directly observe pulses on device pins. After presentation of a single input transition, all device pins are sampled only once. If a device has an internal oscillator or pulse generator that causes outputs to pulse in response to single input transitions, these outputs cannot be correctly sampled by Realchip and Realmodel.

## 4.11 DEVICE DEFINITION FILE EXAMPLES

The following examples show device definition blocks for a dynamic device (Motorola 68010L8 microprocessor) and a static device (AMD 2901C 4-bit bipolar microprocessor slice).

## 68010L8 DEVICE DEFINITION FILE

```
primitive '68010L8';
{Motorola 68010 8MHz Virtual Memory Microprocessor}
{Data from 68010 spec dated December 1982}
  pin
    input_spec =
        '-DTACK'  pin=15,
        '-BR'     pin=19,
        '-BGACK'  pin=18,
        '-IPL'<2> pin=30,
        '-IPL'<1> pin=31,
        '-IPL'<0> pin=32,
        '-BERR'   pin=29,
        '-VPA'    pin=27,
        'CLK'     pin=21: nc_pin;

    io_spec =
        'D'<15>  pin=64: delay=70  : float_delay=0,
        'D'<14>  pin=66: delay=70  : float_delay=0,
        'D'<13>  pin=67: delay=70  : float_delay=0,
        'D'<12>  pin=68: delay=70  : float_delay=0,
        'D'<11>  pin=69: delay=70  : float_delay=0,
        'D'<10>  pin=70: delay=70  : float_delay=0,
        'D'<9>   pin=71: delay=70  : float_delay=0,
        'D'<8>   pin=72: delay=70  : float_delay=0,
        'D'<7>   pin=73: delay=70  : float_delay=0,
        'D'<6>   pin=74: delay=70  : float_delay=0,
        'D'<5>   pin=75: delay=70  : float_delay=0,
        'D'<4>   pin=6 : delay=70  : float_delay=0,
        'D'<3>   pin=7 : delay=70  : float_delay=0,
        'D'<2>   pin=8 : delay=70  : float_delay=0,
        'D'<1>   pin=9 : delay=70  : float_delay=0,
        'D'<0>   pin=10: delay=70  : float_delay=0,

        '-RESET'  pin=24: delay=0    : output_type=(oc,and),
  '-HALT'           pin=23       : delay=0: output_type=(oc,and);
        {values for RESET and HALT are arbitrary - no data available}

    output_spec =
        'A'<23>  pin=62: delay=70  : float_delay=0 : ts_pin,
        'A'<22>  pin=61: delay=70  : float_delay=0 : ts_pin,
        'A'<21>  pin=60: delay=70  : float_delay=0 : ts_pin,
```

```
        'A'<20>  pin=58: delay=70  : float_delay=0 : ts_pin,
        'A'<19>  pin=57: delay=70  : float_delay=0 : ts_pin,
        'A'<18>  pin=56: delay=70  : float_delay=0 : ts_pin,
        'A'<17>  pin=55: delay=70  : float_delay=0 : ts_pin,
        'A'<16>  pin=54: delay=70  : float_delay=0 : ts_pin,
        'A'<15>  pin=52: delay=70  : float_delay=0 : ts_pin,
        'A'<14>  pin=51: delay=70  : float_delay=0 : ts_pin,
        'A'<13>  pin=50: delay=70  : float_delay=0 : ts_pin,
        'A'<12>  pin=49: delay=70  : float_delay=0 : ts_pin,
        'A'<11>  pin=48: delay=70  : float_delay=0 : ts_pin,
        'A'<10>  pin=47: delay=70  : float_delay=0 : ts_pin,
        'A'<9>   pin=46: delay=70  : float_delay=0 : ts_pin,
        'A'<8>   pin=45: delay=70  : float_delay=0 : ts_pin,
        'A'<7>   pin=44: delay=70  : float_delay=0 : ts_pin,
        'A'<6>   pin=43: delay=70  : float_delay=0 : ts_pin,
        'A'<5>   pin=42: delay=70  : float_delay=0 : ts_pin,
        'A'<4>   pin=39: delay=70  : float_delay=0 : ts_pin,
        'A'<3>   pin=38: delay=70  : float_delay=0 : ts_pin,
        'A'<2>   pin=37: delay=70  : float_delay=0 : ts_pin,
        'A'<1>   pin=36: delay=70  : float_delay=0 : ts_pin,

        '-AS'    pin=11: delay=60  : float_delay=0 : ts_pin,

        'R'      pin=14: delay=70  : float_delay=0 : ts_pin,

        '-UDS'   pin=12        : delay=60: float_delay=0 : ts_pin,
        '-LDS'   pin=13: delay=60  : float_delay=0 : ts_pin,

        '-BG'    pin=17: delay=70,
        'E'      pin=26: delay=70,
        '-VMA'   pin=25: delay=70  : float_delay=0 : ts_pin,

        'FC'<2>  pin=33: delay=70  : float_delay=0: ts_pin,
        'FC'<1>  pin=34: delay=70  : float_delay=0: ts_pin,
        'FC'<0>  pin=35: delay=70  : float_delay=0: ts_pin;

        {Vcc is pins 20,59; Gnd is pins 22,63}

    end_pin;

    jig_id = 15;
    jig_type = dynamic;
    clock_edge = both;
```

```
    clock_period = 125-700;
    clock_pin = 'CLK';

    reset_seq
        '-RESET'   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
        '-HALT'    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
        PAUSE      0,1,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z;

    end_reset_seq;
  end_primitive;
```

## 2901C DEVICE DEFINITION FILE

```
primitive '2901C';
{AMD 2901C four-bit bipolar microprocessor slice}
{Data from AMD 2900 Family 1983 Databook}
 {Combinational Propagation Delays, Output Enable/Disable
   Times Tables (Page 5-12)}
   pin
      input_spec =
        'A'<0>   pin=9,
        'A'<1>   pin=8,
        'A'<2>   pin=7,
        'A'<3>   pin=6,

        'B'<0>   pin=23,
        'B'<1>   pin=24,
        'B'<2>   pin=25,
        'B'<3>   pin=26,

        'D'<0>   pin=59,
        'D'<1>   pin=58,
        'D'<2>   pin=57,
        'D'<3>   pin=56,

        'I'<0>   pin=18,
        'I'<1>   pin=19,
        'I'<2>   pin=20,
        'I'<3>   pin=60,
        'I'<4>   pin=62,
        'I'<5>   pin=61,
        'I'<6>   pin=10,
        'I'<7>   pin=12,
        'I'<8>   pin=11,
```

```
    'CN'        pin=63,

    '-OE'       pin=75,
    'CP'        pin=21 : strobe_pin;

  io_spec =
    'Q_LSB'     pin=55 : output_type=(oc,and),
    'Q_MSB'     pin=22 : output_type=(oc,and),
    'RAM_LSB'          pin=14 : output_type=(oc,and),
    'RAM_MSB'          pin=13 : output_type=(oc,and);

  output_spec =
    'Y'<0>      pin=71 : ts_pin,
    'Y'<1>      pin=72 : ts_pin,
    'Y'<2>      pin=73 : ts_pin,
    'Y'<3>      pin=74 : ts_pin,
    'CN+4'      pin=68,
    'OVR'       pin=69,
    'F=0'       pin=17 : output_type=(oc,and),
    'F_MSB'     pin=66,
    '-G'        pin=67,
    '-P'        pin=70;

{Vcc is pin 15; Gnd is pin 64}

end_pin;

jig_id = 25;
jig_type = static;

delay_table

  'Y'<0>, 'Y'<1>, 'Y'<2>, 'Y'<3>:
    'A'<0>, 'A'<1>, 'A'<2>, 'A'<3> = 40,
    'B'<0>, 'B'<1>, 'B'<2>, 'B'<3> = 40,
    'D'<0>, 'D'<1>, 'D'<2>, 'D'<3> = 30,
    'CN' = 22,
    'I'<0>, 'I'<1>, 'I'<2>, 'I'<3> = 35,
    'I'<4>, 'I'<5>, 'CP' = 35,
    'I'<6>, 'I'<7>, 'I'<8> = 25,
    '-OE' = 23;
```

```
'F_MSB':
 'A'<0>, 'A'<1>, 'A'<2>, 'A'<3> = 40,
 'B'<0>, 'B'<1>, 'B'<2>, 'B'<3> = 40,
 'D'<0>, 'D'<1>, 'D'<2>, 'D'<3> = 30,
 'CN' = 22,
 'I'<0>, 'I'<1>, 'I'<2>, 'I'<3> = 35
 'I'<4>, 'I'<5>, 'CP' = 35;

'CN+4':
 'A'<0>, 'A'<1>, 'A'<2>, 'A'<3> = 40,
 'B'<0>, 'B'<1>, 'B'<2>, 'B'<3> = 40,
 'D'<0>, 'D'<1>, 'D'<2>, 'D'<3> = 30,
 'CN' = 20,
 'I'<0>, 'I'<1>, 'I'<2>, 'I'<3> = 35,
 'I'<4>, 'I'<5>, 'CP' = 35;

'-G', '-P':
 'A'<0>, 'A'<1>, 'A'<2>, 'A'<3> = 37,
 'B'<0>, 'B'<1>, 'B'<2>, 'B'<3> = 37,
 'D'<0>, 'D'<1>, 'D'<2>, 'D'<3> = 30,
 'I'<0>, 'I'<1>, 'I'<2> = 37,
 'I'<3>, 'I'<4>, 'I'<5>, 'CP' = 35;

'F=0':
 'A'<0>, 'A'<1>, 'A'<2>, 'A'<3> = 40,
 'B'<0>, 'B'<1>, 'B'<2>, 'B'<3> = 40,
 'D'<0>, 'D'<1>, 'D'<2>, 'D'<3> = 38,
 'CN' = 25,
 'I'<0>, 'I'<1>, 'I'<2> = 37,
 'I'<3>, 'I'<4>, 'I'<5> = 38,
 'CP' = 35;

'OVR':
 'A'<0>, 'A'<1>, 'A'<2>, 'A'<3> = 40,
 'B'<0>, 'B'<1>, 'B'<2>, 'B'<3> = 40,
 'D'<0>, 'D'<1>, 'D'<2>, 'D'<3> = 30,
 'CN' = 22,
 'I'<0>, 'I'<1>, 'I'<2>, 'I'<3> =35,
 'I'<4>, 'I'<5>, 'CP' = 35;
```

```
    'RAM_LSB', 'RAM_MSB':
     'A'<0>, 'A'<1>, 'A'<2>, 'A'<3> = 40,
     'B'<0>, 'B'<1>, 'B'<2>, 'B'<3> = 40,
     'D'<0>, 'D'<1>, 'D'<2>, 'D'<3> = 30,
     'CN' = 25,
     'I'<0>, 'I'<1>, 'I'<2>, 'I'<3> = 35,
     'I'<4>, 'I'<5>, 'CP' = 35,
     'I'<6>, 'I'<7>, 'I'<8> = 26;

    'Q_LSB', 'Q_MSB':
     'I'<6>, 'I'<7>, 'I'<8> = 26,
     'CP' = 28;

  end_delay_table;

end_primitive;
```

# SECTION 6
# NETWORKED REALCHIP

This section introduces the networked version of Realchip and outlines the system considerations.

Networked Realchip makes it possible to utilize both the Realchip Modeling System and Realmodel from a number of different platforms. The Networked Realchip software includes the Networked Realchip Server, and the "hosted" and "networked" modes of operation in ValidSIM, Valid's logic simulator.

## 6.1 NETWORKED REALCHIP SERVER

The Networked Realchip Server manages the hardware resources available in Realchip/Realmodel plugged into the system where it is running; it also communicates with remote logic simulators using a set of application-level protocols. Any Simulator can query the server to obtain information about a particular device, create an instance of a device, reset an instance of a device, add an input pattern to be played to a device, sample an instance and get the results, etc.

When the Networked Realchip Server is initialized, it locates all Realchip/Realmodel device models on the system and "sleeps" awaiting a request from a remote simulator. When a request is received, the server awakes and services the request; requests are queued by the operating system so that all requests are serviced in the order in which they are received.

The Networked Realchip Server runs on two platforms - the Networked Realchip chassis (Valid hardware) and on the VAX. On the Networked Realchip chassis, the server runs on top of the TCP/IP protocol to allow multiple simulators on multiple remote hosts to talk to the server. On the VAX, the server uses inter-process communication

facilities available under VMS; this implementation only allows simulators running on the same machine to access the server. Accordingly, when ValidSIM is run on a MicroVAX, it can talk to the server on the Networked Realchip chassis, but the VAX server can only service requests from local simulators on the VAX.

## 6.2 SIMULATOR SOFTWARE

The ValidSIM simulator supports two modes of operation: "hosted" and "networked." In the hosted mode, the simulator interfaces to the Realchip/Realmodel hardware directly; It looks to see if any Realchip/Realmodel is available on the system where it is running and prevents other users from using it concurrently (equivalent to non-networked Realchip).

In the networked mode, the simulator interfaces to a Networked Realchip Server, as described above. Input patterns are sent to the modeling hardware using the network connection, and the results are read back into the simulator. The networked mode is enabled by specifying the directive, REMOTE_HOST, in the simulator's directives file (*simulate.cmd*). The syntax for this directive is as follows:

REMOTE_HOST *host name* [,*host name*,...];

When the networked mode is enabled, ValidSIM first looks for the requested device locally. If the device is not found, the ValidSIM queries each host specified by the REMOTE_HOST directive to see what devices are available on the remote hosts. When a device is available on more than one host, ValidSIM utilizes an algorithm to select the device that is the least likely to cause contention on that host.

The REALCHIP_LIBRARY directive automatically uses the hosted mode if the REMOTE_HOST directive is not specified.

## 6.3 PLATFORMS SUPPORTED

ValidSIM supports Networked Realchip on four platforms: the S-32, IBM PC/AT, VAX, and MicroVAX. The implementations vary somewhat between each of these platforms and the differences are described below.

- On the S-32 and IBM PC/AT, ValidSIM uses TCP/IP protocols to communicate with the Networked Realchip Server program(s) running on the host(s) specified by the REMOTE_HOST directive. Both the hosted and networked modes of operation are supported on the S-32, while only the networked mode is supported on the IBM PC/AT.

- The MicroVAX also supports both the hosted and networked modes. In the hosted mode, the Realchip hardware is connected to the system through the Q-bus. TCP/IP protocols are used in the networked mode. Special network software must be installed on the system in order to use TCP/IP. As on the S-32, the hosted mode takes precedence over the networked mode (i.e., the simulator does not search remotely when the requirements are satisfied locally).

- In addition to the hosted and networked modes, the simulator supports the Inter-Process Communication (IPC) mode on the VAX. In IPC mode, ValidSIM uses VMS facilities (Global Section and Lock Management) to communicate with the Realchip Server. Note that both ValidSIM and the Realchip Server must be running on the same machine for the IPC to take place. For the hosted mode, the Realchip hardware is connected to the VAX system through the UNIBUS. If the TCP/IP software package is installed on the system, ValidSIM is also able to use the networked mode.

The order of precedence of searching for reference elements is as follows:

1. HOSTED
2. IPC
3. NETWORKED

ValidSIM first checks if local Realchip hardware (i.e., VAX Realchip) is available. If VAX Realchip (VRC) is not found or if the requested reference elements are not found in the local VRC, ValidSIM trys to attach itself to the Realchip Server using IPC. Finally, if additional reference elements are requested, ValidSIM goes off the network and starts querying the hosts specified by the REMOTE_HOST directive.

## 6.4 NETWORKED ERRORS

The following are some of the types of network errors reported by ValidSIM:

- the host specified in the REMOTE_HOST directive is down.
- the Networked Realchip Server for the host specified by the REMOTE_HOST directive is down.
- the Realchip service is not defined in /etc/services.
- the host specified by the REMOTE_HOST directive is unknown.

## 6.5 BRINGING DOWN THE SYSTEM

As with Realchip and Realmodel, the system must be powered down before a Device Control Module (DCM) can be removed or installed. Since the DCM and Realchip or Realmodel Master board reside in the same enclosure, the system must be brought down in an orderly manner before turning the power off.

Before powering down the system, make sure that no one is logged on the system (use **who** command to list the logged-on users). If there are other users on the system, have them logout. Login as superuser and type:

```
% /etc/shutdown – r +1
```

Wait until the '>' prompt comes up and turn off the power switch.

## 6.6 BRINGING THE SYSTEM UP

Turn on the power switch in the front panel and wait a couple of minutes. The system is set to be in the autoboot mode from the factory and will automatically start itself when power is applied. As part of the autoboot sequence, the system displays messages on the console and prompts for the system date. Type in the date according to the syntax specified on the screen. The system next asks for confirmation of the date. Type 'y' (or RETURN) or 'n'' accordingly. After the date is entered and confirmed, the system next asks if the file system is to be checked. If the system was brought down gracefully, this check should not be necessary (i.e., enter 'n'); if the system is being brought up following a crash, the file system check should be run. It is a good idea to run the file system check periodically (e.g., once a week), even if the system has been functioning correctly. For a more detailed explanation of this procedure please refer to the *SCALDsystem Manager's Reference Manual.*

## 6.7 RESTARTING THE NETWORKED REALCHIP SERVER

The Networked Realchip server (*netrc*) is started automatically when the system is brought up. If it becomes necessary restart the server (i.e., if the network "hangs"), login as superuser and type

% **ps -alx**

and look for the line "/etc/netrc." Find the PID (process ID number) of this entry and then (as the superuser) type:

% **kill -9** *<letc/netrc PID number>*

While still logged-in as superuser, restart the server by typing:

% **/etc/netrc**

It is also possible to start up the Networked Realchip server under the control of a console and get its initialization message. To do this, first kill the server as previously described and, while logged in as superuser, type:

% **/etc/netrc -d**

The system should immediately respond with:

> Valid Logic Systems, Inc. Networked Realchip Server 1.1 26Sep85
>
> No errors encountered during initialization.
> Ready to accept connections.

Any other messages are indicative of error conditions and should be reported to Valid.

# APPENDIX A
# REALMODEL 128-PIN PERSONALITY MODULE

The Realmodel 128-Pin Personality Module is a six layer printed circuit board designed to interface the device to be modeled (i.e., the reference element) and its supporting circuitry with Valid's Realmodel Device Control Module (DCM). Note: the Realmodel 128-Pin Personality Module is NOT interchangeable with the Realchip 128-Pin Personality Module.

The Realmodel 128-Pin Personality Module models reference elements in PGA packages with 100 mil center-to-center pin spacing in the following configurations:

| | |
|---|---|
| 10x10x2 | |
| 11x11x2 | 11x11x3 |
| 12x12x2 | 12x12x3 |
| 13x13x2 | 13x13x3 |
| 14x14x2 | 14x14x3 |

For the above devices, signals are automatically mapped; larger devices (up to 16x16) and devices with all pin positions occupied can be accommodated with additional user wiring as long as the number of active pins does not exceed 128.

The personality module is 3.0 by 12.3 inches with four rows of 42 pins on the bottom for interface with the Realmodel Device Control Module. Figure A-1 shows the blank Realmodel 128-Pin Personality Module.

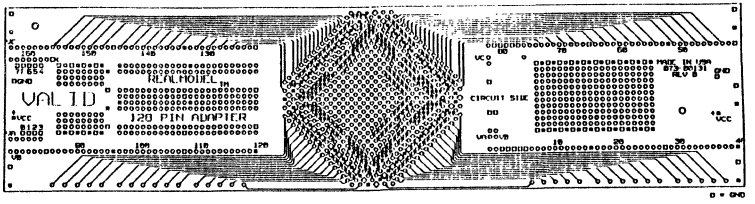


**Figure A-1. Realmodel 128-Pin Personality Module**

## A.1 PERSONALITY MODULE ORIENTATION

Most of the traces have been routed to the personality module's two outer layers to allow for easy customizing. The module contains:

- A central Pin Grid Array (PGA) to hold the reference element being modeled.
- A set of two horizontal, parallel rows of pins on each side of the PGA area that plug into the Realmodel DCM.
- A vector pin pad array to the right of the PGA area for use in breadboarding.
- A DIP (Dual In-line Package) work area to the left of the PGA area for integrating required support ICs.
- A set of pads to the left of the DIP area to mount required discrete devices.

For identification purposes, the two rows of pins to the right of the PGA area are called socket 0, and the two rows of pins to the left of the PGA area are called socket 1. Left and right are referenced with the component side up with the Valid logo to the left (pins down).

## SOCKET PIN NUMBERING

The pins of socket 0 are numbered from 1 to 80 as noted by the silkscreen on the component side of the personality module. The pins of socket 1 are numbered from 81 to 160 and are likewise noted on the silkscreen.

## PGA PIN NUMBERING

The rows of pins of the PGA are named with either a letter or a number so that each pin position can be identified by using a unique letter/number pair. The lettered rows are labeled f, e, d, c, b, a, A, B, C, and so on up to and including V. Note that the letters I and O are NOT used. The numbered rows are named @6, @5, @4, @3, @2, @1, 1, 2, 3, and so on up to and including 20. These rows and their names are indicated in silkscreen on the board's component side. As an example, the square pad bordered by a small silkscreened diamond is named A1. Rows d, T, @4, and 18 are made up of alternating ground and Vcc pads; the square pads are ground, and the round pads are Vcc (except pads G@4, T7, H18, and d8, which are signal pads). Rows a, Q, @1, and 15 are made up of unconnected signal pads. These rows are provided to simplify customizing.

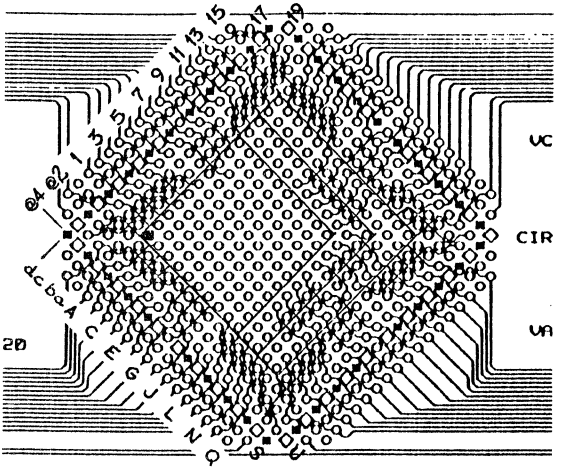


**Figure A-2. PGA Pin Numbering**

## VECTOR PIN PAD ARRAY

The vector pin pad array to the right of the PGA is a 23x12 rectangular array of unconnected pads intended as a breadboard area for circuitry (i.e., DIPs, other PGAs, pull-up resistor packs, etc.) in support of the reference element being modeled. The pads are spaced at 100 mils center-to-center. At the top and bottom of the array is a row of 23 alternating ground and Vcc pads for power tapping. As with the PGA, square pads are ground while the round pads are Vcc.

## DIP WORK AREA

The DIP work area to the left of the PGA is designed to accept up to four 24-pin standard DIP ICs (legs spaced 100 mil center-to-center with a span across the chip of 300 mil). To simplify customizing, each pad is connected to an unused pad, and four ground and two Vcc pads are located near where pins 12 and 24 fall when the maximum configuration of four 24-pin devices is used.

### DISCRETE COMPONENT WORK AREA

The discrete component work area consists of two parallel horizontal rows of eight pads spaced at 100 mil center to center with a span of 500 mil center to center. Each of the sixteen pads in these two rows is connected to two pads to simplify wiring. Outside each outer row of wiring pads is a row of eight alternating ground and Vcc pads for power tapping. This work area is intended to support discrete components required in the supporting circuitry of the reference element being modeled. The discrete component work area is located to the left of the DIP work area between the two pin rows of socket 1.

## A.2 POSITIONING THE REFERENCE ELEMENT

Pin A1 of the device to be modeled must be inserted in pad A1 of the central PGA area on the component side of the board so that the device falls completely within one of the silkscreened outlines or "footprints." Note that devices should always be socket mounted (either a low insertion force socket or zero insertion force socket).

The following tables define the active signal mapping between the socket interface pins for the supported reference element sizes. The socket pin number in the table is the pin number entered in the device definition file. Note that for each size reference element, there are pins that are not routed to socket interface pins (i.e., the NC or "no connect" entries in the tables.

## Table A-1. Socket to 10x10x2 PGA Pin Mapping

| PGA Pin | Socket Pin | PGA Pin | Socket Pin |
|---|---|---|---|
| A1 | 87 | A2 | 123 |
| A3 | 125 | A4 | 127 |
| A5 | 129 | A6 | 131 |
| A7 | NC | A8 | 140 |
| A9 | 142 | A10 | 149 |
| B1 | 86 | B2 | 89 |
| B3 | 122 | B4 | 126 |
| B5 | 128 | B6 | 134 |
| B7 | 136 | B8 | 138 |
| B9 | 144 | B10 | 147 |
| C1 | 88 | C2 | 93 |
| C9 | 139 | C10 | 141 |
| D1 | 90 | D2 | 95 |
| D9 | 46 | D10 | 48 |
| E1 | 92 | E2 | 94 |
| E9 | 52 | E10 | 50 |
| F1 | 99 | F2 | 97 |
| F9 | 55 | F10 | 51 |
| G1 | 101 | G2 | 103 |
| G9 | 57 | G10 | 54 |
| H1 | NC | H2 | 105 |
| H9 | 59 | H10 | 56 |
| J1 | 110 | J2 | 107 |
| J3 | 109 | J4 | 15 |
| J5 | 18 | J6 | 20 |
| J7 | 22 | J8 | 24 |
| J9 | 26 | J10 | 62 |
| K1 | 112 | K2 | 113 |
| K3 | 111 | K4 | 13 |
| K5 | 17 | K6 | 19 |
| K7 | 25 | K8 | 27 |
| K9 | 30 | K10 | 31 |

## Table A-2. Socket to 11x11x2 PGA Pin Mapping

| PGA Pin | Socket Pin | PGA Pin | Socket Pin | PGA Pin | Socket Pin |
|---|---|---|---|---|---|
| A1 | 87 | A2 | 123 | A3 | 125 |
| A4 | 127 | A5 | 129 | A6 | 131 |
| A7 | NC | A8 | 140 | A9 | 142 |
| A10 | 149 | A11 | 151 | | |
| B1 | 86 | B2 | 89 | B3 | 122 |
| B4 | 126 | B5 | 128 | B6 | 134 |
| B7 | 136 | B8 | 138 | B9 | 144 |
| B10 | 147 | B11 | 146 | | |
| C1 | 88 | C2 | 93 | C10 | 141 |
| C11 | 143 | | | | |
| D1 | 90 | D2 | 95 | D10 | 48 |
| D11 | 45 | | | | |
| E1 | 92 | E2 | 94 | E10 | 50 |
| E11 | 47 | | | | |
| F1 | 99 | F2 | 97 | F10 | 51 |
| F11 | 49 | | | | |
| G1 | 101 | G2 | 103 | G10 | 54 |
| G11 | NC | | | | |
| H1 | NC | H2 | 105 | H10 | 56 |
| H11 | 58 | | | | |
| J1 | 110 | J2 | 107 | J10 | 62 |
| J11 | 60 | | | | |
| K1 | 112 | K2 | 113 | K3 | 111 |
| K4 | 13 | K5 | 17 | K6 | 19 |
| K7 | 25 | K8 | 27 | K9 | 30 |
| K10 | 31 | K11 | 67 | | |
| L1 | 114 | L2 | 115 | L3 | 117 |
| L4 | 12 | L5 | 14 | L6 | 21 |
| L7 | 23 | L8 | NC | L9 | 32 |
| L10 | 34 | L11 | 36 | | |

**Table A-3. Socket to 11x11x3 PGA Pin Mapping**

| PGA Pin | Socket Pin | PGA Pin | Socket Pin | PGA Pin | Socket Pin |
|---|---|---|---|---|---|
| A1 | 87 | A2 | 123 | A3 | 125 |
| A4 | 127 | A5 | 129 | A6 | 131 |
| A7 | NC | A8 | 140 | A9 | 142 |
| A10 | 149 | A11 | 151 | | |
| B1 | 86 | B2 | 89 | B3 | 122 |
| B4 | 126 | B5 | 128 | B6 | 134 |
| B7 | 136 | B8 | 138 | B9 | 144 |
| B10 | 147 | B11 | 146 | | |
| C1 | 88 | C2 | 93 | C3 | 91 |
| C4 | 124 | C5 | 130 | C6 | 132 |
| C7 | 135 | C8 | 137 | C9 | 139 |
| C10 | 141 | C11 | 143 | | |
| D1 | 90 | D2 | 95 | D3 | 98 |
| D9 | 46 | D10 | 48 | D11 | 45 |
| E1 | 92 | E2 | 94 | E3 | 100 |
| E9 | 52 | E10 | 50 | E11 | 47 |
| F1 | 99 | F2 | 97 | F3 | 102 |
| F9 | 55 | F10 | 51 | F11 | 49 |
| G1 | 101 | G2 | 103 | G3 | 104 |
| G9 | 57 | G10 | 54 | G11 | NC |
| H1 | NC | H2 | 105 | H3 | 106 |
| H9 | 59 | H10 | 56 | H11 | 58 |
| J1 | 110 | J2 | 107 | J3 | 109 |
| J4 | 15 | J5 | 18 | J6 | 20 |
| J7 | 22 | J8 | 24 | J9 | 26 |
| J10 | 62 | J11 | 60 | | |
| K1 | 112 | K2 | 113 | K3 | 111 |
| K4 | 13 | K5 | 17 | K6 | 19 |
| K7 | 25 | K8 | 27 | K9 | 30 |
| K10 | 31 | K11 | 67 | | |
| L1 | 114 | L2 | 115 | L3 | 117 |
| L4 | 12 | L5 | 14 | L6 | 21 |
| L7 | 23 | L8 | NC | L9 | 32 |
| L10 | 34 | L11 | 36 | | |

**Table A-4. Socket to 12x12x2 PGA Pin Mapping**

| PGA Pin | Socket Pin | PGA Pin | Socket Pin | PGA Pin | Socket Pin |
|---|---|---|---|---|---|
| A1 | 87 | A2 | 123 | A3 | 125 |
| A4 | 127 | A5 | 129 | A6 | 131 |
| A7 | NC | A8 | 140 | A9 | 142 |
| A10 | 149 | A11 | 151 | A12 | 153 |
| B1 | 86 | B2 | 89 | B3 | 122 |
| B4 | 126 | B5 | 128 | B6 | 134 |
| B7 | 136 | B8 | 138 | B9 | 144 |
| B10 | 147 | B11 | 146 | B12 | 148 |
| C1 | 88 | C2 | 93 | C11 | 143 |
| C12 | 150 | | | | |
| D1 | 90 | D2 | 95 | D11 | 45 |
| D12 | 46 | | | | |
| E1 | 92 | E2 | 94 | E11 | 47 |
| E12 | 52 | | | | |
| F1 | 99 | F2 | 97 | F11 | 49 |
| F12 | 55 | | | | |
| G1 | 101 | G2 | 103 | G11 | NC |
| G12 | 57 | | | | |
| H1 | NC | H2 | 105 | H11 | 58 |
| H12 | 59 | | | | |
| J1 | 110 | J2 | 107 | J11 | 60 |
| J12 | 61 | | | | |
| K1 | 112 | K2 | 113 | K11 | 67 |
| K12 | 63 | | | | |
| L1 | 114 | L2 | 115 | L3 | 117 |
| L4 | 12 | L5 | 14 | L6 | 21 |
| L7 | 23 | L8 | NC | L9 | 32 |
| L10 | 34 | L11 | 36 | L12 | 66 |
| M1 | 116 | M2 | 119 | M3 | 9 |
| M4 | 15 | M5 | 18 | M6 | 20 |
| M7 | 22 | M8 | 24 | M9 | 26 |
| M10 | 29 | M11 | 35 | M12 | 72 |

## Table A-5. Socket to 12x12x3 PGA Pin Mapping

| PGA Pin | Socket Pin | PGA Pin | Socket Pin | PGA Pin | Socket Pin |
|---|---|---|---|---|---|
| A1 | 87 | A2 | 123 | A3 | 125 |
| A4 | 127 | A5 | 129 | A6 | 131 |
| A7 | NC | A8 | 140 | A9 | 142 |
| A10 | 149 | A11 | 151 | A12 | 153 |
| B1 | 86 | B2 | 89 | B3 | 122 |
| B4 | 126 | B5 | 128 | B6 | 134 |
| B7 | 136 | B8 | 138 | B9 | 144 |
| B10 | 147 | B11 | 146 | B12 | 148 |
| C1 | 88 | C2 | 93 | C3 | 91 |
| C4 | 124 | C5 | 130 | C6 | 132 |
| C7 | 135 | C8 | 137 | C9 | 139 |
| C10 | 141 | C11 | 143 | C12 | 150 |
| D1 | 90 | D2 | 95 | D3 | 98 |
| D10 | 48 | D11 | 45 | D12 | 46 |
| E1 | 92 | E2 | 94 | E3 | 100 |
| E10 | 50 | E11 | 47 | E12 | 52 |
| F1 | 99 | F2 | 97 | F3 | 102 |
| F10 | 51 | F11 | 49 | F12 | 55 |
| G1 | 101 | G2 | 103 | G3 | 104 |
| G10 | 54 | G11 | NC | G12 | 57 |
| H1 | NC | H2 | 105 | H3 | 106 |
| H10 | 56 | H11 | 58 | H12 | 59 |
| J1 | 110 | J2 | 107 | J3 | 109 |
| J10 | 62 | J11 | 60 | J12 | 61 |
| K1 | 112 | K2 | 113 | K3 | 111 |
| K4 | 13 | K5 | 17 | K6 | 19 |
| K7 | 25 | K8 | 27 | K9 | 30 |
| K10 | 31 | K11 | 67 | K12 | 63 |
| L1 | 114 | L2 | 115 | L3 | 117 |
| L4 | 12 | L5 | 14 | L6 | 21 |
| L7 | 23 | L8 | NC | L9 | 32 |
| L10 | 34 | L11 | 36 | L12 | 66 |
| M1 | 116 | M2 | 119 | M3 | 9 |
| M4 | 15 | M5 | 18 | M6 | 20 |
| M7 | 22 | M8 | 24 | M9 | 26 |
| M10 | 29 | M11 | 35 | M12 | 72 |

## Table A-6. Socket to 13x13x2 PGA Pin Mapping

| PGA Pin | Socket Pin | PGA Pin | Socket Pin | PGA Pin | Socket Pin |
|---|---|---|---|---|---|
| A1 | 87 | A2 | 123 | A3 | 125 |
| A4 | 127 | A5 | 129 | A6 | 131 |
| A7 | NC | A8 | 140 | A9 | 142 |
| A10 | 149 | A11 | 151 | A12 | 153 |
| A13 | 155 | | | | |
| B1 | 86 | B2 | 89 | B3 | 122 |
| B4 | 126 | B5 | 128 | B6 | 134 |
| B7 | 136 | B8 | 138 | B9 | 144 |
| B10 | 147 | B11 | 146 | B12 | 148 |
| B13 | 152 | | | | |
| C1 | 88 | C2 | 93 | C12 | 150 |
| C13 | 44 | | | | |
| D1 | 90 | D2 | 95 | D12 | 46 |
| D13 | 48 | | | | |
| E1 | 92 | E2 | 94 | E12 | 52 |
| E13 | 50 | | | | |
| F1 | 99 | F2 | 97 | F12 | 55 |
| F13 | 51 | | | | |
| G1 | 101 | G2 | 103 | G12 | 57 |
| G13 | 54 | | | | |
| H1 | NC | H2 | 105 | H12 | 59 |
| H13 | 56 | | | | |

**Table A-6. Socket to 13x13x2 PGA Pin Mapping** (Con't)

| PGA Pin | Socket Pin | PGA Pin | Socket Pin | PGA Pin | Socket Pin |
|---|---|---|---|---|---|
| J1 | 110 | J2 | 107 | J12 | 61 |
| J13 | 62 | | | | |
| K1 | 112 | K2 | 113 | K12 | 63 |
| K13 | 64 | | | | |
| L1 | 114 | L2 | 115 | L12 | 66 |
| L13 | 68 | | | | |
| M1 | 116 | M2 | 119 | M3 | 9 |
| M4 | 15 | M5 | 18 | M6 | 20 |
| M7 | 22 | M8 | 24 | M9 | 26 |
| M10 | 29 | M11 | 35 | M12 | 72 |
| M13 | 70 | | | | |
| N1 | 118 | N2 | 7 | N3 | 11 |
| N4 | 13 | N5 | 17 | N6 | 19 |
| N7 | 25 | N8 | 27 | N9 | 30 |
| N10 | 31 | N11 | 33 | N12 | 37 |
| N13 | 74 | | | | |

## Table A-7. Socket to 13x13x3 PGA Pin Mapping

| PGA Pin | Socket Pin | PGA Pin | Socket Pin | PGA Pin | Socket Pin |
|---|---|---|---|---|---|
| A1 | 87 | A2 | 123 | A3 | 125 |
| A4 | 127 | A5 | 129 | A6 | 131 |
| A7 | NC | A8 | 140 | A9 | 142 |
| A10 | 149 | A11 | 151 | A12 | 153 |
| A13 | 155 | | | | |
| B1 | 86 | B2 | 89 | B3 | 122 |
| B4 | 126 | B5 | 128 | B6 | 134 |
| B7 | 136 | B8 | 138 | B9 | 144 |
| B10 | 147 | B11 | 146 | B12 | 148 |
| B13 | 152 | | | | |
| C1 | 88 | C2 | 93 | C3 | 91 |
| C4 | 124 | C5 | 130 | C6 | 132 |
| C7 | 135 | C8 | 137 | C9 | 139 |
| C10 | 141 | C11 | 143 | C12 | 150 |
| C13 | 44 | | | | |
| D1 | 90 | D2 | 95 | D3 | 98 |
| D11 | 45 | D12 | 46 | D13 | 48 |
| E1 | 92 | E2 | 94 | E3 | 100 |
| E11 | 47 | E12 | 52 | E13 | 50 |
| F1 | 99 | F2 | 97 | F3 | 102 |
| F11 | 49 | F12 | 55 | F13 | 51 |
| G1 | 101 | G2 | 103 | G3 | 104 |
| G11 | NC | G12 | 57 | G13 | 54 |
| H1 | NC | H2 | 105 | H3 | 106 |
| H11 | 58 | H12 | 59 | H13 | 56 |

**Table A-7. Socket to 13x13x3 PGA Pin Mapping** (Con't)

| PGA Pin | Socket Pin | PGA Pin | Socket Pin | PGA Pin | Socket Pin |
|---|---|---|---|---|---|
| J1 | 110 | J2 | 107 | J3 | 109 |
| J11 | 60 | J12 | 61 | J13 | 62 |
| K1 | 112 | K2 | 113 | K3 | 111 |
| K11 | 67 | K12 | 63 | K13 | 64 |
| L1 | 114 | L2 | 115 | L3 | 117 |
| L4 | 12 | L5 | 14 | L6 | 21 |
| L7 | 23 | L8 | NC | L9 | 32 |
| L10 | 34 | L11 | 36 | L12 | 66 |
| L13 | 68 | | | | |
| M1 | 116 | M2 | 119 | M3 | 9 |
| M4 | 15 | M5 | 18 | M6 | 20 |
| M7 | 22 | M8 | 24 | M9 | 26 |
| M10 | 29 | M11 | 35 | M12 | 72 |
| M13 | 70 | | | | |
| N1 | 118 | N2 | 7 | N3 | 11 |
| N4 | 13 | N5 | 17 | N6 | 19 |
| N7 | 25 | N8 | 27 | N9 | 30 |
| N10 | 31 | N11 | 33 | N12 | 37 |
| N13 | 74 | | | | |

## Table A-8. Socket to 14x14x2 PGA Pin Mapping

| PGA Pin | Socket Pin | PGA Pin | Socket Pin | PGA Pin | Socket Pin |
|---|---|---|---|---|---|
| A1 | 87 | A2 | 123 | A3 | 125 |
| A4 | 127 | A5 | 129 | A6 | 131 |
| A7 | NC | A8 | 140 | A9 | 142 |
| A10 | 149 | A11 | 151 | A12 | 153 |
| A13 | 155 | A14 | 154 | | |
| B1 | 86 | B2 | 89 | B3 | 122 |
| B4 | 126 | B5 | 128 | B6 | 134 |
| B7 | 136 | B8 | 138 | B9 | 144 |
| B10 | 147 | B11 | 146 | B12 | 148 |
| B13 | 152 | B14 | 42 | | |
| C1 | 88 | C2 | 93 | C13 | 44 |
| C14 | 43 | | | | |
| D1 | 90 | D2 | 95 | D13 | 48 |
| D14 | 45 | | | | |
| E1 | 92 | E2 | 94 | E13 | 50 |
| E14 | 47 | | | | |
| F1 | 99 | F2 | 97 | F13 | 51 |
| F14 | 49 | | | | |
| G1 | 101 | G2 | 103 | G13 | 54 |
| G14 | NC | | | | |
| H1 | NC | H2 | 105 | H13 | 56 |
| H14 | 58 | | | | |

**Table A-8. Socket to 14x14x2 PGA Pin Mapping** (Con't)

| PGA Pin | Socket Pin | PGA Pin | Socket Pin | PGA Pin | Socket Pin |
|---|---|---|---|---|---|
| J1 | 110 | J2 | 107 | J13 | 62 |
| J14 | 60 | | | | |
| K1 | 112 | K2 | 113 | K13 | 64 |
| K14 | 67 | | | | |
| L1 | 114 | L2 | 115 | L13 | 68 |
| L14 | 69 | | | | |
| M1 | 116 | M2 | 119 | M13 | 70 |
| M14 | 71 | | | | |
| N1 | 118 | N2 | 7 | N3 | 11 |
| N4 | 13 | N5 | 17 | N6 | 19 |
| N7 | 25 | N8 | 27 | N9 | 30 |
| N10 | 31 | N11 | 33 | N12 | 37 |
| N13 | 74 | N14 | 73 | | |
| P1 | 6 | P2 | 8 | P3 | 10 |
| P4 | 12 | P5 | 14 | P6 | 21 |
| P7 | 23 | P8 | NC | P9 | 32 |
| P10 | 34 | P11 | 36 | P12 | 38 |
| P13 | 39 | P14 | 75 | | |

**Table A-9. Socket to 14x14x3 PGA Pin Mapping**

| PGA Pin | Socket Pin | PGA Pin | Socket Pin | PGA Pin | Socket Pin |
|---|---|---|---|---|---|
| A1 | 87 | A2 | 123 | A3 | 125 |
| A4 | 127 | A5 | 129 | A6 | 131 |
| A7 | NC | A8 | 140 | A9 | 142 |
| A10 | 149 | A11 | 151 | A12 | 153 |
| A13 | 155 | A14 | 154 | | |
| B1 | 86 | B2 | 89 | B3 | 122 |
| B4 | 126 | B5 | 128 | B6 | 134 |
| B7 | 136 | B8 | 138 | B9 | 144 |
| B10 | 147 | B11 | 146 | B12 | 148 |
| B13 | 152 | B14 | 42 | | |
| C1 | 88 | C2 | 93 | C3 | 91 |
| C4 | 124 | C5 | 130 | C6 | 132 |
| C7 | 135 | C8 | 137 | C9 | 139 |
| C10 | 141 | C11 | 143 | C12 | 150 |
| C13 | 44 | C14 | 43 | | |
| D1 | 90 | D2 | 95 | D3 | 98 |
| D12 | 46 | D13 | 48 | D14 | 45 |
| E1 | 92 | E2 | 94 | E3 | 100 |
| E12 | 52 | E13 | 50 | E14 | 47 |
| F1 | 99 | F2 | 97 | F3 | 102 |
| F12 | 55 | F13 | 51 | F14 | 49 |
| G1 | 101 | G2 | 103 | G3 | 104 |
| G12 | 57 | G13 | 54 | G14 | NC |
| H1 | NC | H2 | 105 | H3 | 106 |
| H12 | 59 | H13 | 56 | H14 | 58 |

**Table A-9. Socket to 14x14x3 PGA Pin Mapping** (Con't)

| PGA Pin | Socket Pin | PGA Pin | Socket Pin | PGA Pin | Socket Pin |
|---|---|---|---|---|---|
| J1 | 110 | J2 | 107 | J3 | 109 |
| J12 | 61 | J13 | 62 | J14 | 60 |
| K1 | 112 | K2 | 113 | K3 | 111 |
| K12 | 63 | K13 | 64 | K14 | 67 |
| L1 | 114 | L2 | 115 | L3 | 117 |
| L12 | 66 | L13 | 68 | L14 | 69 |
| M1 | 116 | M2 | 119 | M3 | 9 |
| M4 | 15 | M5 | 18 | M6 | 20 |
| M7 | 22 | M8 | 24 | M9 | 26 |
| M10 | 29 | M11 | 35 | M12 | 72 |
| M13 | 70 | M14 | 71 | | |
| N1 | 118 | N2 | 7 | N3 | 11 |
| N4 | 13 | N5 | 17 | N6 | 19 |
| N7 | 25 | N8 | 27 | N9 | 30 |
| N10 | 31 | N11 | 33 | N12 | 37 |
| N13 | 74 | N14 | 73 | | |
| P1 | 6 | P2 | 8 | P3 | 10 |
| P4 | 12 | P5 | 14 | P6 | 21 |
| P7 | 23 | P8 | NC | P9 | 32 |
| P10 | 34 | P11 | 36 | P12 | 38 |
| P13 | 39 | P14 | 75 | | |

## A.3 UNCONNECTED PGA CONNECTIONS

The following pins of the PGA are interconnected, but have no interface to the personality module socket interface pins (these pins are designated 'NC' in tables A-1 through A-9). To use the table, locate the desired pad in the footprint area in the first column and read across to find the pad to wire to a socket interface pin pad described in the next section.

**Table A-10. Unconnected PGA Pins**

| Footprint Pad | Interconnect Pad |
|---|---|
| A7 | d8 |
| G11 | G14, H18 |
| G14 | H18 |
| H1 | G@4 |
| L8 | P8, T7 |
| P8 | T7 |

As an example, if pin G11 on a PGA other than 14x14x2 or 14x14x3 is defined as an active signal, a jumper wire would be installed between either pad G14 or pad H18 and one of the unassigned socket interface pins defined in table A-11.

The following table lists the unassigned socket interface pins for each of the supported PGA pin configurations. The number in parentheses following the pin number is the first PGA pad connected to the socket interface pin. Note that this pad can be jumpered to one of the PGA connections outside the footprint area (see table A-10) to route a NC (no connect) pad to a socket interface pin.

**Table A-11. Unassigned Socket Interface Pins**

| PGA Size | Unused Pin Number | | |
|---|---|---|---|
| 10x10x2 | 6(U@2) | 7(U1) | 8(V1) |
| | 9(U2) | 10(V2) | 11(U3) |
| | 12(V3) | 14(V4) | 21(V7) |
| | 23(V8) | 29(U11) | 32(V12) |
| | 33(U13) | 34(V13) | 35(U14) |
| | 36(V14) | 37(U16) | 38(V16) |
| | 39(U17) | 42(b19) | 43(A20) |
| | 44(A19) | 45(B20) | 47(C20) |
| | 49(D20) | 58(H20) | 60(J20) |
| | 61(J19) | 63(K19) | 64(L20) |
| | 66(L19) | 67(M20) | 68(M19) |
| | 69(N20) | 70(N19) | 71(P20) |
| | 72(P19) | 73(R20) | 74(R19) |
| | 75(S19) | 91(A@5) | 98(D@5) |
| | 100(E@5) | 102(F@5) | 104(G@5) |
| | 106(H@5) | 114(N@6) | 115(M@5) |
| | 116(P@6) | 117(N@5) | 118(R@5) |
| | 119(P@5) | 124(e1) | 130(e4) |
| | 132(e5) | 135(e6) | 137(e7) |
| | 143(e10) | 146(e11) | 148(e12) |
| | 150(e13) | 151(f12) | 152(e14) |
| | 153(f13) | 154(e16) | 155(f14) |
| 11x11x2 | 6(U@2) | 7(U1) | 8(V1) |
| | 9(U2) | 10(V2) | 11(U3) |
| | 29(U11) | 33(U13) | 35(U14) |
| | 37(U16) | 38(V16) | 39(U17) |
| | 42(b19) | 43(A20) | 44(A19) |
| | 47(C20) | 61(J19) | 63(K19) |
| | 64(L20) | 66(L19) | 68(M19) |
| | 69(N20) | 70(N19) | 71(P20) |
| | 72(P19) | 73(R20) | 74(R19) |
| | 75(S19) | 91(A@5) | 98(D@5) |
| | 100(E@5) | 102(F@5) | 104(G@5) |
| | 106(H@5) | 116(P@6) | 118(R@5) |
| | 119(P@5) | 124(e1) | 130(e4) |
| | 132(e5) | 135(e6) | 137(e7) |
| | 148(e12) | 150(e13) | 152(e14) |
| | 153(f13) | 154(e16) | 155(f14) |

**Table A-11. Unassigned Socket Interface Pins** (Con't)

| PGA Size | Unused Pin Number | | |
|---|---|---|---|
| 11x11x3 | 6(U@2) | 7(U1) | 8(V1) |
| | 9(U2) | 10(V2) | 11(U3) |
| | 29(U11) | 33(U13) | 35(U14) |
| | 37(U16) | 38(V16) | 39(U17) |
| | 42(b19) | 43(A20) | 44(A19) |
| | 47(C20) | 61(J19) | 63(K19) |
| | 64(L20) | 66(L19) | 68(M19) |
| | 69(N20) | 70(N19) | 71(P20) |
| | 72(P19) | 73(R20) | 74(R19) |
| | 75(S19) | 116(P@6) | 118(R@5) |
| | 119(P@5) | 148(e12) | 150(e13) |
| | 152(e14) | 153(f13) | 154(e16) |
| | 155(f14) | | |
| 12x12x2 | 6(U@2) | 7(U1) | 8(V1) |
| | 10(V2) | 11(U3) | 13(U4) |
| | 17(V5) | 19(V6) | 25(V9 |
| | 27(V10) | 30(V11) | 33(U13) |
| | 37(U16) | 38(V16) | 39(U17) |
| | 42(b19) | 43(A20) | 44(A19) |
| | 48(C19) | 50(D19) | 51(E20) |
| | 54(F20) | 56(G20 | 62(K20) |
| | 64(L20) | 68(M19) | 69(N20) |
| | 70(N19) | 71(P20) | 73(R20) |
| | 74(R19) | 75(S19) | 91(A@5) |
| | 98(D@5 | 100(E@5) | 102(F@5) |
| | 104(G@5) | 106(H@5) | 109(J@5) |
| | 111(K@5) | 118(R@5) | 124(e1) |
| | 130(e4) | 132(e5) | 135(e6) |
| | 137(e7) | 139(e8) | 141(e9) |
| | 152(e14) | 154(e16) | 155(f14) |
| 12x12x3 | 6(U@2) | 7(U1) | 8(V1) |
| | 10(V2) | 11(U3) | 33(U13) |
| | 37(U16) | 38(V16) | 39(U17) |
| | 42(b19) | 43(A20) | 44(A19) |
| | 64(L20) | 68(M19) | 69(N20) |
| | 70(N19) | 71(P20) | 73(R20) |
| | 74(R19) | 75(S19) | 118(R@5) |
| | 152(e14) | 154(e16) | 155(f14) |

**Table A-11. Unassigned Socket Interface Pins** (Con't)

| PGA Size | Unused Pin Number | | |
|---|---|---|---|
| 13x13x2 | 6(U@2) | 8(V1) | 10(V2) |
| | 12(V3) | 14(V4) | 21(V7) |
| | 23(V8) | 32(V12) | 34(V13) |
| | 36(V14) | 38(V16) | 39(U17) |
| | 42(b19) | 43(A20) | 45(B20) |
| | 47(C20) | 49(D20) | 58(H20) |
| | 60(J20) | 67(M20) | 69(N20) |
| | 71(P20) | 73(R20) | 75(S19) |
| | 91(A@5) | 98(D@5 | 100(E@5) |
| | 102(F@5) | 104(G@5) | 106(H@5) |
| | 109(J@5) | 111(K@5) | 117(N@5) |
| | 124(e1) | 130(e4) | 132(e5) |
| | 135(e6) | 137(e7) | 139(e8) |
| | 141(e9) | 143(e10) | 152(e14) |
| | 154(e16) | | |
| 13x13x3 | 6(U@2) | 8(V1) | 10(V2) |
| | 38(V16) | 39(U17) | 42(b19) |
| | 43(A20) | 69(N20) | 71(P20) |
| | 73(R20) | 75(S19) | 152(e14) |
| | 154(e16) | | |
| 14x14x2 | 9(U2) | 15(U5) | 18(U6) |
| | 20(U7) | 22(U8) | 24(U9) |
| | 26(U10) | 29(U11) | 35(U14) |
| | 46(B19) | 52(E19) | 55(F19) |
| | 57(G19) | 59(H19) | 61(J19) |
| | 63(K19) | 66(L19) | 72(P19) |
| | 91(A@5) | 98(D@5 | 100(E@5) |
| | 102(F@5) | 104(G@5) | 106(H@5) |
| | 109(J@5) | 111(K@5) | 117(N@5) |
| | 124(e1) | 130(e4) | 132(e5) |
| | 135(e6) | 137(e7) | 139(e8) |
| | 141(e9) | 143(e10) | 150(e13) |
| 14x14x3 | none | | |

## A.4 POWER AND GROUND WIRING

No power (Vcc) and ground are routed to the footprint area and must be provided by jumper wire. Rows @ 4, 18, d, and T are made up of alternating ground and Vcc pads; the square pads are ground, and the round pads are Vcc. If the Vcc and ground pins on the reference element are routed to socket interface pins (see tables A-1 through A-9), the corresponding traces between the inboard and outboard pads in table A-12 must be cut and jumper wires installed between Vcc/ground and the inboard pads. If the Vcc and ground pins on the reference element are not routed to a socket interface pin, refer to table A-10 and connect jumpers between Vcc and the specified interconnect pad and between ground and the specified interface pad.

## A.5 CLOCK AND FEEDBACK WIRING

In order to use the hardware clock (dynamic devices), connect an insulated wire (on the component side) from the CK pad in the upper left corner of the personality module (adjacent to socket interface pin 156) to the clock pin of the reference element. If the CLOCK pin on the reference element is routed to a socket interface pin (see tables A-1 through A-9), the corresponding trace between the inboard and outboard pads in table A-12 must be cut and a jumper installed between the CK pad and the inboard pad. Note that if the trace is not cut, the CLOCK signal will be fed back to the device control module. If the CLOCK pin on the reference element is not routed to a socket interface pin, refer to table A-10 and connect a wire between the CK pad and the specified interconnect pad.

When a feedback signal is required, connect an insulated wire from pad F (adjacent to socket interface pin 160) to the pin of the reference element being used as the feedback signal. When feedback is not being used, this pad should be connected to the adjacent ground pad (the square pad labeled 'F').

## A.6 NO-CONNECT PINS

Unused or no-connect (NC) pins on a reference element should be isolated from the device control module to prevent possible overdriving or feedback. To isolate a no-connect pin, identify the corresponding socket interface pin (see tables A-1 through A-9) and then cut the trace between the outboard pad connected to the interface pin and the inboard pad into the array defined in table A-12.

## A.7 SOCKET INTERFACE PIN ASSIGNMENTS

The following table lists the PGA pads that are connected directly to the 128 active signal socket interface pins. In the table, the "outboard pad" is the pad connected directly to the socket interface pin, and the "inboard pad" is the pad into the PGA. Generally, cutting the trace between an inboard and outboard pad is only necessary to isolate power and ground, NC (no-connect) pins, and the clock and feedback signals.

## Table A-12. Socket Interface Pin Assignments

| Socket Pin | Outboard Pad | Inboard Pad | Socket Pin | Outboard Pad | Inboard Pad |
|---|---|---|---|---|---|
| 6 | U@ 2 | S@ 3 | 42 | b19 | b17 |
| 7 | U1 | R@ 2 | 43 | A20 | A17 |
| 8 | V1 | S@ 2 | 44 | A19 | A16 |
| 9 | U2 | R1 | 45 | B20 | B17 |
| 10 | V2 | S1 | 46 | B19 | B16 |
| 11 | U3 | R2 | 47 | C20 | C17 |
| 12 | V3 | S2 | 48 | C19 | C16 |
| 13 | U4 | R3 | 49 | D20 | D17 |
| 14 | V4 | S3 | 50 | D19 | D16 |
| 15 | U5 | R4 | 51 | E20 | E17 |
| 17 | V5 | S4 | 52 | E19 | E16 |
| 18 | U6 | R5 | 54 | F20 | F17 |
| 19 | V6 | S5 | 55 | F19 | F16 |
| 20 | U7 | R6 | 56 | G20 | G17 |
| 21 | V7 | S6 | 57 | G19 | G16 |
| 22 | U8 | R7 | 58 | H20 | H17 |
| 23 | V8 | S7 | 59 | H19 | H16 |
| 24 | U9 | R8 | 60 | J20 | J17 |
| 25 | V9 | S8 | 61 | J19 | J16 |
| 26 | U10 | R9 | 62 | K20 | K17 |
| 27 | V10 | S9 | 63 | K19 | K16 |
| 29 | U11 | R10 | 64 | L20 | L17 |
| 30 | V11 | S10 | 66 | L19 | L16 |
| 31 | U12 | R11 | 67 | M20 | M17 |
| 32 | V12 | S11 | 68 | M19 | M16 |
| 33 | U13 | R12 | 69 | N20 | N17 |
| 34 | V13 | S12 | 70 | N19 | N16 |
| 35 | U14 | R13 | 71 | P20 | P17 |
| 36 | V14 | S13 | 72 | P19 | P16 |
| 37 | U16 | R14 | 73 | R20 | R17 |
| 38 | V16 | S14 | 74 | R19 | R16 |
| 39 | U17 | S16 | 75 | S19 | S17 |

## Table A-12. Socket Interface Pin Assignments

| Socket Pin | Outboard Pad | Inboard Pad | Socket Pin | Outboard Pad | Inboard Pad |
|---|---|---|---|---|---|
| 86 | b@ 6 | b@ 3 | 122 | e@ 2 | b1 |
| 87 | c@ 5 | c@ 3 | 123 | e@ 3 | c@ 2 |
| 88 | A@ 6 | A@ 3 | 124 | e1 | b2 |
| 89 | b@ 5 | b@ 2 | 125 | f@ 2 | c1 |
| 90 | B@ 6 | B@ 3 | 126 | e2 | b3 |
| 91 | A@ 5 | A@ 2 | 127 | f1 | c2 |
| 92 | C@ 6 | C@ 3 | 128 | e3 | b4 |
| 93 | B@ 5 | B@ 2 | 129 | f2 | c3 |
| 94 | D@ 6 | D@ 3 | 130 | e4 | b5 |
| 95 | C@ 5 | C@ 2 | 131 | f3 | c4 |
| 97 | E@ 6 | E@ 3 | 132 | e5 | b6 |
| 98 | D@ 5 | D@ 2 | 134 | f4 | c5 |
| 99 | F@ 6 | F@ 3 | 135 | e6 | b7 |
| 100 | E@ 5 | E@ 2 | 136 | f5 | c6 |
| 101 | G@ 6 | G@ 3 | 137 | e7 | b8 |
| 102 | F@ 5 | F@ 2 | 138 | f6 | c7 |
| 103 | H@ 6 | H@ 3 | 139 | e8 | b9 |
| 104 | G@ 5 | G@ 2 | 140 | f7 | c8 |
| 105 | J@ 6 | J@ 3 | 141 | e9 | b10 |
| 106 | H@ 5 | H@ 2 | 142 | f8 | c9 |
| 107 | K@ 6 | K@ 3 | 143 | e10 | b11 |
| 109 | J@ 5 | J@ 2 | 144 | f9 | c10 |
| 110 | L@ 6 | L@ 3 | 146 | e11 | b12 |
| 111 | K@ 5 | K@ 2 | 147 | f10 | c11 |
| 112 | M@ 6 | M@ 3 | 148 | e12 | b13 |
| 113 | L@ 5 | L@ 2 | 149 | f11 | c12 |
| 114 | N@ 6 | N@ 3 | 150 | e13 | b14 |
| 115 | M@ 5 | M@ 2 | 151 | f12 | c13 |
| 116 | P@ 6 | P@ 3 | 152 | e14 | b16 |
| 117 | N@ 5 | N@ 2 | 153 | f13 | c14 |
| 118 | R@ 5 | R@ 3 | 154 | e16 | c17 |
| 119 | P@ 5 | P@ 2 | 155 | f14 | c16 |

## A.8 EXTERNAL VOLTAGES

Three external voltages (VA, VB, and VC) can be supplied through the Realmodel back panel. Each of these voltages appears on the Realmodel 128-Pin Personality Module twice at the locations indicated by the silkscreen on the component side of the board (once between pins 1 and 80 in socket 0, and again between pins 81 and 160 in socket 1). Pads are provided for bypass capacitors on each of these voltages if required.

## A.9 FILTER CAPACITORS

The personality module requires filter capacitors between Vcc and ground. Referring to figure A-1, install two 22uF electrolytic capacitors between the VCC and GND pads noting proper polarity. Install two 0.01uF capacitors along the left edge and three 0.01uF capacitors along the right edge at the silkscreened locations. Additional 0.01uF capacitors should be installed in rows d, T, @ 4, and 18. Note that when any of the optional supply voltages are used (i.e., VA, VB, or VC), 0.01uF capacitors should be installed at the indicated positions adjacent to the VA, VB, and VC pads to the left of socket 1.

## A.10 JIG ID STRAPPING

Each personality module used in a Realmodel system must have a unique jig ID number between 1 and 126 or between 128 and 254 (note that jig ID numbers 0, 127, and 255 are reserved and cannot be assigned). This number is read by the Simulator (from the JIG_ID entry in the device definition file) to determine which reference elements are mounted. Valid-supplied personality modules are assigned jig ID numbers in ascending order beginning with 1; to reduce the possibility of jig ID number duplication, user-created personality modules should be assigned jig ID number in descending order beginning with either 126 (maximum Realchip jig ID number) or 255 (maximum Realmodel jig ID number).

On the 128-pin personality module, the jig ID pads are located on the left edge of the module and are labeled 0 through 7 (pad 0 is the low-order bit). These pads are routed to pins 81, 82, 83, 84, 157, 158, 159, and the pin to the left of pin 160 of socket 1. With pads 0 through 6, if the pad is connected to ground (the adjacent square pad), its logical value is 0, and if the pad is left unconnected, its logical value is 1. Pad 7 is inverted and is a logical 0 when left unconnected and is a logical 1 when connected. Note that the jig ID pins of socket 0 are permanently configured for a jig ID of 0.

To select a jig ID number, ground the pads of the appropriate bits by strapping the corresponding pads to their adjacent ground (square) pads. As an example of jig ID strapping, jig ID '251' would have ground straps installed in pads 7 and 4 (i.e., bits 0, 1, 2, 3, 5, 6, and 7 would be a logic '1'; binary value 11111011).

# APPENDIX B
# REALCHIP 128-PIN MICRO-SIMULATION PERSONALITY MODULE

## B.1 INTRODUCTION

The Realchip 128-Pin Micro-Simulation Personality Module is the first version of a 128-pin personality module for Realchip (this personality module has been replaced with a new 128-pin personality module). The personality module is designed to interface the device to be modeled (i.e., the reference element) with Valid's Realchip Device Control Module (DCM). Note: the Realchip 128-Pin Micro-Simulation Personality Module is NOT interchangeable with the Realmodel 128-Pin Personality Module.

## B.2 PERSONALITY MODULE BOARD ORIENTATION

Most of the traces have been routed on the module's two outer layers to allow for easy customizing. The module contains a central Pin Grid Array (PGA) to hold the device being modeled and a set of two horizontal, parallel rows of pins on each side of the PGA area that interface with the two sockets of the Realchip DCM.

For identification purposes, the two rows of pins to the right of the PGA area are called socket 0, and the two rows of pins to the left of the PGA area are called socket 1. Left and right are referenced with the component side up and the Valid logo to the left.

## B.3 SOCKET NUMBERING

The pins of socket 0 are numbered from 1 to 80 as indicated by the silkscreen on the component side of the board. The pins of socket 1 are numbered from 81 to 160 and are likewise labeled on the silkscreen.

## B.4 PGA PIN NUMBERING

The rows of pins of the PGA are named with either a letter or a number so that each pin position may be identified by using a unique letter/number pair. The lettered rows are labeled A through X, and the number rows are labeled 1 through 24 as shown in figure B-1. These letters and numbers are silkscreened on the board's component side.

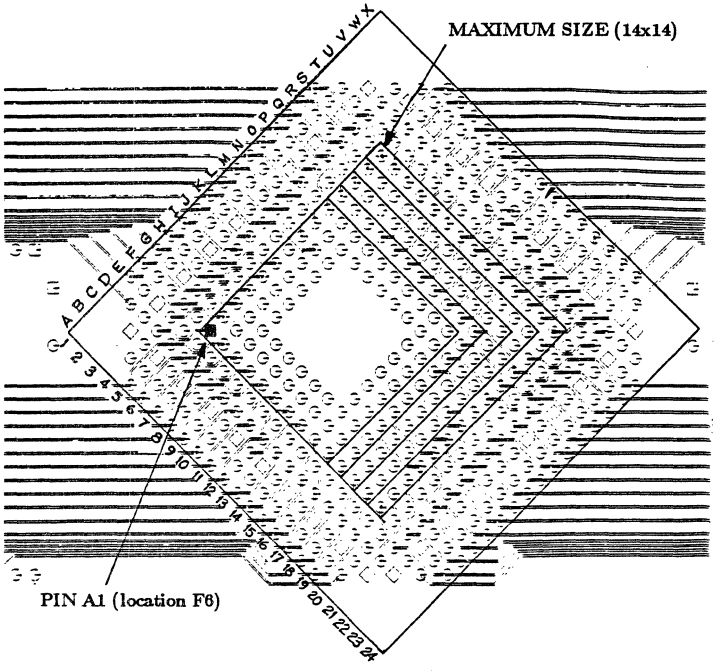


**Figure B-1. PGA Pin Labeling**

Rows C, V, 3, and 22 are made up of alternating ground and Vcc pads. The diamond shaped pads are ground, and the round pads are Vcc (except pads C12, M3, L22, and V13, which are signal pads).

## B.5 REFERENCE ELEMENT SUPPORT

The Realchip 128-Pin Personality Module models reference elements in PGA packages with 100 mil center-to-center pin spacing in the following configurations:

| | |
|---|---|
| 10x10x2 | |
| 11x11x2 | 11x11x3 |
| 12x12x2 | 12x12x3 |
| 13x13x2 | 13x13x3 |
| 14x14x2 | 14x14x3 |

For the above devices, signals are automatically mapped.

## B.6 POSITIONING THE REFERENCE ELEMENT

Pin 1 of the reference element must be inserted in pad F6 of the central PGA area on the component side of the board so that the reference element falls completely within one of the silkscreened outlines. Note that devices should always be socket mounted (either a low insertion force socket or zero insertion force socket).

Traces on the Realchip 128-Pin Personality Module are routed for default wiring as defined by the netlist in section B.11. Required changes can be affected by cutting traces on either surface layer of the board and by adding any necessary point-to-point jumper wires. NOTE: All cuts within the PGA area should be made outside of rows 6, 19, F, and S). Generally, cuts are only necessary for connecting power and ground and the clock signal to the appropriate device pins.

## B.7 JIG ID STRAPPING

Each personality module used in a Realchip system must have a unique jig ID number between 1 and 126. Note that jig IDs 0 and 127 are reserved and cannot be used. The jig ID number is read by the Simulator (from the JIG_ID entry in the device definition file) to determine which reference elements are mounted. Valid-supplied personality modules are assigned jig ID numbers in ascending order beginning with 1; to reduce the possibility of jig ID number duplication, user-created personality modules should be assigned jig ID numbers in descending order beginning with 126.

On the 128-pin personality module, the jig ID pads are located on left edge of the module and are labeled 0 through 6 (pad 0 is the low-order bit). These pads are routed to pins 81, 82, 83, 84, 157, 158, and 159 of socket 1. If an ID pad is connected to ground (the adjacent square pad), its logical value is 0. If an ID pad is left unconnected, its logical value is 1. Note that the jig ID pins of socket 0 are automatically grounded (JIG_ID=0) by the personality module.

To select a jig ID number, ground the pads of the appropriate bits by strapping the corresponding pads to their adjacent ground (square) pads. As an example of jig ID strapping, jig ID '101' would have ground straps installed in vias 1, 3, and 4 (i.e., bits 0, 2, 5, and 6 would be a logic '1'; binary value 1100101).

## B.8 CLOCK AND FEEDBACK WIRING

In order to use the hardware clock for dynamic devices, connect a wire (on the component side) from the pad labeled CK (connected to pin 156) on the left side of the personality module to the clock pin of the device to be modeled. Refer to section B.11 and cut the trace corresponding to the reference element's clock pin at the socket interface pin. Note that if this trace is not cut, the clock signal will be fed back to the device control module.

When feedback is required, connect a wire (on the component side) from pad 7 (connected to pin 160) on the left side of the personality module to the feedback pin of the device to be modeled. Note that when feedback is not being used, this pad should be connected to its adjacent ground (square) pad.

## B.9 NO-CONNECT PINS

Unused or no-connect (NC) pins on a reference element should be isolated from the device control module to prevent possible overdriving or feedback. To isolate a no-connect pin, identify the corresponding socket pin number in section B.11 for the no-connect pin and then cut the trace between the socket interface pin and the "outside" array pad (i.e., the last PGA pad number listed in the table).

## B.10 FILTER CAPACITORS

The personality module requires filter capacitors between Vcc and ground. Referring to figure B-2, install two 22 uF electrolytic capacitors between the VCC and GND pads noting proper polarity. Install two 0.01 uF capacitors along the left edge and three 0.01 uF capacitors along the right edge. Additional 0.01 uF capacitors should be installed in rows C, V, 3, and 22 between Vcc (round pad) and ground (square pad). Make sure that pads C12, M3, L22, and V13 (the signal pads in the Vcc/ground rows) are not used.

## B.11 NETLIST

The following table defines the active signal mapping between the socket pins and the reference element installed in the PGA area of the Realchip 128-Pin Micro-Simulation Personality Module. The socket pin number in the table is the pin number entered in the pin block of the device definition file.

## Socket to PGA Pin Mapping

| Socket Pin Number | PGA Pad Number | Signal Count |
|---|---|---|
| 6 | D21, E23, F19 | 1 |
| 7 | E20, F23, G18 | 2 |
| 8 | E21, F24, G19 | 3 |
| 9 | F20, G23, H17 | 4 |
| 10 | F21, G24, H19 | 5 |
| 11 | G20, H18, H23 | 6 |
| 12 | G21, H24, I16, I19 | 7 |
| 13 | H20, I15, I18, I23 | 8 |
| 14 | H21, I24, J16, J19 | 9 |
| 15 | I14, I17, I20, J23 | 10 |
| 17 | I21, J15, J18, J24 | 11 |
| 18 | J14, J17, J20, K23 | 12 |
| 19 | J21, K15, K18, K24 | 13 |
| 20 | K14, K17, K20, L23 | 14 |
| 21 | K16, K19, K21, L24 | 15 |
| 22 | L14, L17, L20, M23 | 16 |
| 23 | L16, L19, L21, M24 | 17 |
| 24 | M14, M17, M20, N23 | 18 |
| 25 | L15, L18, M21, N24 | 19 |
| 26 | N14, N17, N20, O23 | 20 |
| 27 | M15, M18, N21, O24 | 21 |
| 29 | O17, O20, P23 | 22 |
| 30 | N15, N18, O21, P24 | 23 |
| 31 | O15, O18, P20, Q23 | 24 |
| 32 | N16, N19, P21, Q24 | 25 |
| 33 | P18, Q20, R23 | 26 |
| 34 | O16, O19, Q21, R24 | 27 |
| 35 | P17, R20, S23 | 28 |
| 36 | P16, P19, R21, S24 | 29 |
| 37 | Q18, S20, T23 | 30 |
| 38 | Q19, S21, T24 | 31 |
| 39 | R19, T21, U23 | 32 |

## Socket to PGA Pin Mapping

| Socket Pin Number | PGA Pad Number | Signal Count |
|---|---|---|
| 42 | S7, U5, W5 | 33 |
| 43 | S8, U6, X6 | 34 |
| 44 | R8, T6, W6 | 35 |
| 45 | P9, S9, U7, X7 | 36 |
| 46 | N9, Q9, T7, W7 | 37 |
| 47 | P10, S10, U8, X8 | 38 |
| 48 | O9, R9, T8, W8 | 39 |
| 49 | P11, S11, U9, X9 | 40 |
| 50 | O10, R10, T9, W9 | 41 |
| 51 | O11, R11, U10, X10 | 42 |
| 52 | N10, Q10, T10, W10 | 43 |
| 54 | O12, R12, U11, X11 | 44 |
| 55 | N11, Q11, T11, W11 | 45 |
| 56 | O13, R13, U12, X12 | 46 |
| 57 | N12, Q12, T12, W12 | 47 |
| 58 | P13, S13, U13, X13 | 48 |
| 59 | N13, Q13, T13, W13 | 49 |
| 60 | P14, S14, U14, X14 | 50 |
| 61 | Q14, T14, W14 | 51 |
| 62 | O14, R14, U15, X15 | 52 |
| 63 | Q15, T15, W15 | 53 |
| 64 | R15, U16, X16 | 54 |
| 66 | Q16, T16, W16 | 55 |
| 67 | P15, S15, U17, X17 | 56 |
| 68 | R16, T17, W17 | 57 |
| 69 | S16, U18, X18 | 58 |
| 70 | R17, T18, W18 | 59 |
| 71 | S17, U19, X19 | 60 |
| 72 | Q17, T19, W19 | 61 |
| 73 | S18, U20, X20 | 62 |
| 74 | R18, T20, W20 | 63 |
| 75 | S19, U21, W21 | 64 |

## Socket to PGA Pin Mapping

| Socket Pin Number | PGA Pad Number | Signal Count |
|---|---|---|
| 86 | A5, D5, F7 | 65 |
| 87 | B4, D4, F6 | 66 |
| 88 | A6, D6, F8 | 67 |
| 89 | B5, E5, G7 | 68 |
| 90 | A7, D7, F9 | 69 |
| 91 | B6, E6, H8 | 70 |
| 92 | A8, D8, F10 | 71 |
| 93 | B7, E7, G8 | 72 |
| 94 | A9, D9, G10 | 73 |
| 95 | B8, E8, G9 | 74 |
| 97 | A10, D10, G11 | 75 |
| 98 | B9, E9, H9 | 76 |
| 99 | A11, D11, F11 | 77 |
| 100 | B10, E10, H10 | 78 |
| 101 | A12, D12, F12 | 79 |
| 102 | B11, E11, H11 | 80 |
| 103 | A13, D13, G12 | 81 |
| 104 | B12, E12, H12 | 82 |
| 105 | A14, D14, G13 | 83 |
| 106 | B13, E13, H13 | 84 |
| 107 | A15, D15, G14 | 85 |
| 109 | B14, E14, H14 | 86 |
| 110 | A16, D16, F14 | 87 |
| 111 | B15, E15, H15 | 88 |
| 112 | A17, D17, F15 | 89 |
| 113 | B16, E16, G15 | 90 |
| 114 | A18, D18, F16 | 91 |
| 115 | B17, E17, G16 | 92 |
| 116 | A19, D19, F17 | 93 |
| 117 | B18, E18, H16 | 94 |
| 118 | B20, D20, F18 | 95 |
| 119 | B19, E19, G17 | 96 |

## Socket to PGA Pin Mapping

| Socket Pin Number | PGA Pad Number | Signal Count |
|---|---|---|
| 122 | E2, F5, H7 | 97 |
| 123 | D2, E4, G6 | 98 |
| 124 | F2, G5, I8 | 99 |
| 125 | E1, F4, H6 | 100 |
| 126 | G2, H5, I7 | 101 |
| 127 | F1, G4, I6 | 102 |
| 128 | H2, I5, J7 | 103 |
| 129 | G1, H4, J6 | 104 |
| 130 | I2, J5, J8 | 105 |
| 131 | H1, I4, K6 | 106 |
| 132 | J2, K5, K8 | 107 |
| 134 | I1, J4, K7 | 108 |
| 135 | K2, L5, L8 | 109 |
| 136 | J1, K4, L7 | 110 |
| 137 | L2, M5, M8 | 111 |
| 138 | K1, L4, M7 | 112 |
| 139 | M2, N5, N8 | 113 |
| 140 | L1, M4, M6 | 114 |
| 141 | N2, O5, O8 | 115 |
| 142 | M1, N4, N6 | 116 |
| 143 | O2, P5, P8 | 117 |
| 144 | N1, N7, O4 | 118 |
| 146 | P2, P7, Q5 | 119 |
| 147 | O1, O7, P4 | 120 |
| 148 | Q2, Q7, R5 | 121 |
| 149 | O6, P1, Q4 | 122 |
| 150 | Q8, R2, S5 | 123 |
| 151 | P6, Q1, R4 | 124 |
| 152 | R7, S2, T5 | 125 |
| 153 | Q6, R1, S4 | 126 |
| 154 | S6, T2, U4 | 127 |
| 155 | R6, S1, T4 | 128 |

## B.12 UNCONNECTED PGA CONNECTIONS

The following pins of the PGA are interconnected, but have no interface to the personality module interface pins.

C12-F13
L22-M16-M19
P12-S12-V13
L6-M3

# APPENDIX C
# 64-PIN DIP PERSONALITY MODULE

The 64-pin DIP personality module can be used interchangeably with both Realchip and Realmodel. The blank personality module is designed for mounting DIP integrated circuits of up to 64 pins. The personality module is 4.1 by 1.9 inches with two rows of 40 pins on the bottom for interface with the Realchip or Realmodel Device Control Module. Figure C-1 shows a blank 64-pin DIP personality module.

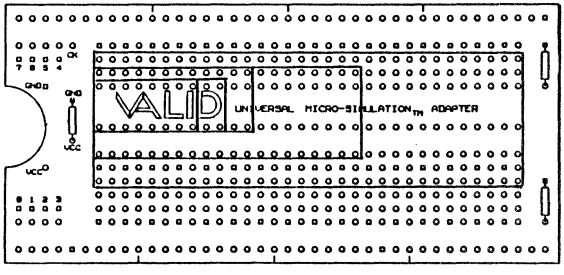


**Figure C-1. 64-Pin DIP Personality Module**

## C.1 PERSONALITY MODULE ORIENTATION

Referring to the figure, most of the personality module is made up of vertically-oriented columns of connected pads or "vias" arranged in horizontal rows. Most of these vias are connected to the interface pins and allow any common-width DIP device to be mounted using the appropriate rows. There are three rows of power and ground pads (the rows with alternating square and round pads). At the notched end of the adapter are eight pairs of vias numbered

0 through 7. Vias 0 through 6 are used to strap the jig ID code for the adapter, and via 7 is used as a feedback path to allow the resetting of certain devices. The single via labeled CK is the clock input from the Device Control Module required by dynamic devices.

## C.2 MOUNTING THE REFERENCE ELEMENT

The reference element is mounted on the personality module in the appropriate rows. Regardless of the number of pins on the reference element, pin 1 of the reference element always must be installed in the via connected to pin 6 of the adapter. Power (Vcc) and ground connections to the reference element are accomplished by strapping (with short bus wires) the via of the corresponding reference element pin to an adjacent power or ground pad (Vcc is the round pad, and ground is the square pad) and then cutting the trace at the personality module interface pin.

**CAUTION**

> After a pin is strapped to Vcc or ground, be sure to cut the corresponding trace to the personality module interface pin. Failure to prevent Vcc or ground from being fed back to the Device Control Module can cause permanent circuit damage.

## C.3 CLOCK AND FEEDBACK WIRING

When a dynamic device (e.g., an Intel 8086) is mounted on the personality module, connect the reference element's clock input to the CK pad using a short, insulated 30 AWG wire across the top (component side) of the adapter. Static and static forever devices (e.g., an AMD 2960) do not use the CK via connection.

The "feedback output" for a reference element that requires feedback must be connected to pad 7 of the personality module. To complete the feedback circuit, identify the pin of the reference element to be used for feedback and connect this pin to the outside pad of via 7 (i.e., the circular pad) using a short, insulated 30 AWG wire on the

component side of the personality module. Do NOT cut the trace between the circular pad and the personality module interface pin. If the reference element does not require feedback, pad 7 should be grounded (i.e., install a bus wire between pad 7 and an adjacent ground pad).

## C.4 NO-CONNECT PINS

Unused or no-connect (NC) pins on a reference element should be isolated from the device control module to prevent possible overdriving or feedback. To isolate a no-connect pin, cut the trace (via) between the socket interface pin and the DIP pin near the socket interface pin.

## C.5 JIG ID

Each personality module used in a Realchip or Realmodel system must have a unique jig ID number. Note that since only seven jig ID bits are available on the 64-Pin DIP Personality Module, the jig ID is limited to 126 (jig IDs 0 and 127 are reserved). The jig ID number is read by the Simulator (from the JIG_ID entry in the device definition file) to determine which reference elements are mounted. Valid-supplied personality modules are assigned jig ID numbers in ascending order beginning with 1; to reduce the possibility of jig ID number duplication, user-created personality modules should be assigned jig ID numbers in descending order beginning with 126.

The individual jig ID pads are located on the left edge of the personality module; pad 0 is the low-order bit, and pad 6 is the high-order bit. A bit is selected or a logic '1' when it is not connected to its adjacent ground (square) pad. To select a jig ID number, ground the appropriate bits by strapping the corresponding round jig ID pads to the adjacent ground pads. As an example of jig ID strapping, jig ID '117' would have ground straps installed in pads 1 and 3 (i.e., bits 0, 2, 4, 5, and 6 would be a logic '1'; binary value 1110101).

## C.6 FILTER CAPACITORS

The adapter requires filter capacitors between Vcc and ground. Referring to figure C-1, install three 0.1 uF, axial-lead ceramic capacitors on the top (component side) of the adapter and install a 22 uF tantalum capacitor on the bottom (circuit side) of the personality module.

## C.7 CLEARANCE

A reference element mounted on a personality module cannot extend more than 0.150 inches above the personality module surface. Reference element beyond this height limit can be used if the adjacent card slot in the Realchip/Realmodel chassis is left empty.

**CAUTION**

> If the top surface of the reference element is conductive, the conductive area must be insulated with mylar tape to avoid accidentally shorting the top of the reference element to the pins on an adjacent device control module. Failure to adequately insulate a reference element's conductive surface can cause permanent damage to both the reference element and device control module.

After the reference element, straps, and capacitors are installed, make sure that no pins or wires protrude more than 0.08 inches beyond the bottom of the personality module.

## C.8 SOCKET TO DIP SIGNAL MAPPING

The following tables define the active signal mapping between the socket interface pins and the installed reference element for most of the popular DIP pin packages. The socket pin number in the first column is the pin number entered in the pin block of the device definition file.

## 16-Pin DIP Package

| Socket Pin | DIP Pin | Socket Pin | DIP Pin |
|---|---|---|---|
| 6 | 1 | 68 | 9 |
| 7 | 2 | 69 | 10 |
| 8 | 3 | 70 | 11 |
| 9 | 4 | 71 | 12 |
| 10 | 5 | 72 | 13 |
| 11 | 6 | 73 | 14 |
| 12 | 7 | 74 | 15 |
| 13 | 8 | 75 | 16 |

## 18-Pin DIP Package

| Socket Pin | DIP Pin | Socket Pin | DIP Pin |
|---|---|---|---|
| 6 | 1 | 67 | 10 |
| 7 | 2 | 68 | 11 |
| 8 | 3 | 69 | 12 |
| 9 | 4 | 70 | 13 |
| 10 | 5 | 71 | 14 |
| 11 | 6 | 72 | 15 |
| 12 | 7 | 73 | 16 |
| 13 | 8 | 74 | 17 |
| 14 | 9 | 75 | 18 |

## 20-Pin DIP Package

| Socket Pin | DIP Pin | Socket Pin | DIP Pin |
|---|---|---|---|
| 6 | 1 | 66 | 11 |
| 7 | 2 | 67 | 12 |
| 8 | 3 | 68 | 13 |
| 9 | 4 | 69 | 14 |
| 10 | 5 | 70 | 15 |
| 11 | 6 | 71 | 16 |
| 12 | 7 | 72 | 17 |
| 13 | 8 | 73 | 18 |
| 14 | 9 | 74 | 19 |
| 15 | 10 | 75 | 20 |

## 24-Pin DIP Package

| Socket Pin | DIP Pin | Socket Pin | DIP Pin |
|---|---|---|---|
| 6 | 1 | 63 | 13 |
| 7 | 2 | 64 | 14 |
| 8 | 3 | 66 | 15 |
| 9 | 4 | 67 | 16 |
| 10 | 5 | 68 | 17 |
| 11 | 6 | 69 | 18 |
| 12 | 7 | 70 | 19 |
| 13 | 8 | 71 | 20 |
| 14 | 9 | 72 | 21 |
| 15 | 10 | 73 | 22 |
| 17 | 11 | 74 | 23 |
| 18 | 12 | 75 | 24 |

## 40-Pin DIP Package

| Socket Pin | DIP Pin | Socket Pin | DIP Pin |
|---|---|---|---|
| 6 | 1 | 56 | 21 |
| 7 | 2 | 57 | 22 |
| 8 | 3 | 58 | 23 |
| 9 | 4 | 59 | 24 |
| 10 | 5 | 60 | 25 |
| 11 | 6 | 61 | 26 |
| 12 | 7 | 62 | 27 |
| 13 | 8 | 63 | 28 |
| 14 | 9 | 64 | 29 |
| 15 | 10 | 66 | 30 |
| 17 | 11 | 67 | 31 |
| 18 | 12 | 68 | 32 |
| 19 | 13 | 69 | 33 |
| 20 | 14 | 70 | 34 |
| 21 | 15 | 71 | 35 |
| 22 | 16 | 72 | 36 |
| 23 | 17 | 73 | 37 |
| 24 | 18 | 74 | 38 |
| 25 | 19 | 74 | 39 |
| 26 | 20 | 75 | 40 |

## 64-Pin DIP Package

| Socket Pin | DIP Pin | Socket Pin | DIP Pin |
|---|---|---|---|
| 6 | 1 | 42 | 33 |
| 7 | 2 | 43 | 34 |
| 8 | 3 | 44 | 35 |
| 9 | 4 | 45 | 36 |
| 10 | 5 | 46 | 37 |
| 11 | 6 | 47 | 38 |
| 12 | 7 | 48 | 39 |
| 13 | 8 | 49 | 40 |
| 14 | 9 | 50 | 41 |
| 15 | 10 | 51 | 42 |
| 17 | 11 | 52 | 43 |
| 18 | 12 | 54 | 44 |
| 19 | 13 | 55 | 45 |
| 20 | 14 | 56 | 46 |
| 21 | 15 | 57 | 47 |
| 22 | 16 | 58 | 48 |
| 23 | 17 | 59 | 49 |
| 24 | 18 | 60 | 50 |
| 25 | 19 | 61 | 51 |
| 26 | 20 | 62 | 52 |
| 27 | 21 | 63 | 53 |
| 29 | 22 | 64 | 54 |
| 30 | 23 | 66 | 55 |
| 31 | 24 | 67 | 56 |
| 32 | 25 | 68 | 57 |
| 33 | 26 | 69 | 58 |
| 34 | 27 | 70 | 59 |
| 35 | 28 | 71 | 60 |
| 36 | 29 | 72 | 61 |
| 37 | 30 | 73 | 62 |
| 38 | 31 | 74 | 63 |
| 39 | 32 | 75 | 64 |

# APPENDIX D
# 68-PIN LCC PERSONALITY MODULE

The 68-pin LCC (leadless chip carrier) personality module can be used interchangeably with both Realchip and Realmodel. The blank personality module is designed for mounting 68-pin LCC integrated circuits. The personality module is 4.1 by 1.9 inches with two rows of 40 pins on the bottom for interface with the Realchip or Realmodel Device Control Module. Figure D-1 shows a blank 68-pin LCC personality module.

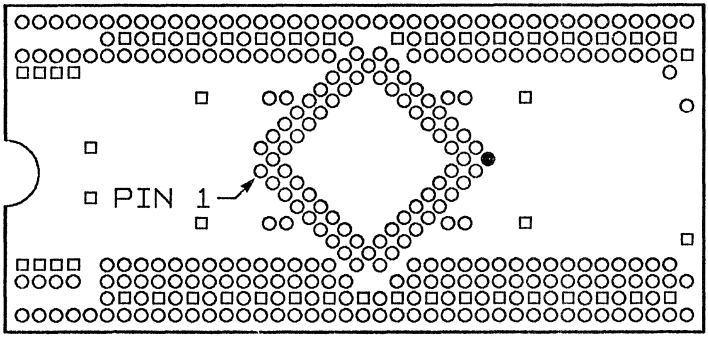


**Figure D-1. 68-Pin LCC Personality Module**

## D.1 PERSONALITY MODULE ORIENTATION

Referring to the figure, the personality module includes a centrally mounted 68-pin leaded chip carrier socket and five horizontal rows of pads adjacent to the socket interface pins (two rows at the top and three rows at the bottom.) The row of pads adjacent to the interface pins at the top and bottom is made up of alternating Vcc and ground pads (the square pads are ground). The remaining three rows are connected to the corresponding socket interface pins and

are used for modifying the reference element to socket interface pin mapping. At the notched end of the personality module are eight pairs of pads numbered 0 through 7. Pads 0 through 6 are used to strap the jig ID code for the personality module (see section D.9), and pad 7 is used as a feedback path to allow the resetting of certain dynamic devices. The single pad labeled CK is the hardware clock input from the Device Control Module required by dynamic devices.

## D.2 INSERTING THE REFERENCE ELEMENT

The leaded chip carrier socket is keyed to orient pin 1 of the reference element to the left. To insert the reference element, open the strap and remove the top mounting plate. Insert the reference element into the carrier socket with pin 1 (usually notched) aligned with the key. Replace the top mounting plate and push the strap back over the mounting plate.

## D.3 POWER WIRING

Power and ground to the reference element must be connected from the corresponding Vcc and ground pads to the appropriate reference element power and ground pins by a jumper wire. If power/ground does not appear on pins 9, 26, 43, or 60 of the reference element:

1. Refer to section D.9 to determine which socket interface pins correspond to the reference element's power and ground pins.

2. Install a short jumper wire between the nearest power/ground pad in the adjacent power/ground row and the corresponding signal pad in the next row (i.e., the pad connected to the reference element on the circuit side of the module).

3. On the circuit side of the board, cut the trace(s) to the socket interface pin(s).

**CAUTION**

Failure to cut the corresponding trace to the personality module interface pin can cause permanent circuit damage to the DCM.

When the reference element's power and ground pins appear on pins 9, 26, 43, or 60, it is not necessary to cut any of the traces since these pins are routed only to the four pads around the perimeter of the LCC socket (just under the edge) and are not connected to any of the socket interface pins. As shown in Figure D-2, the adjacent round pads are Vcc, and the nearby square pads are ground. To connect power or ground, install an insulated jumper between the "pin" pad and Vcc or ground. Note that the wire must be installed on the component side of the board and that care must be taken when soldering the wire to avoid shorting the pad to the LCC socket legs.

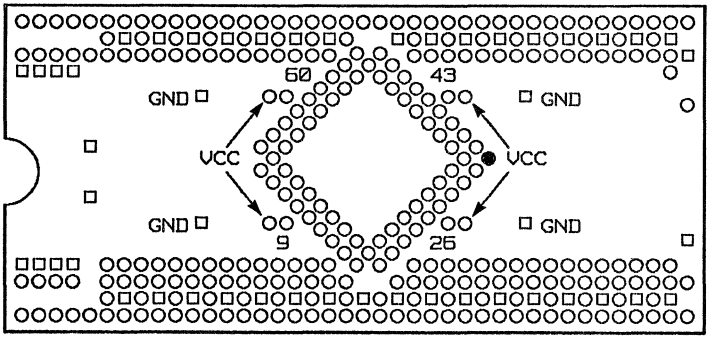


**Figure D-2. Power and Ground Wiring**

## D.4 CLOCK AND FEEDBACK WIRING

In order to use the hardware clock (dynamic devices), connect an insulated wire (on the component side) from the 'CK' pad in the upper left corner of the personality module to the clock pin of the reference element. is mounted on the personality module, connect the reference element's clock input to the CK via using a short, insulated 30 AWG wire across the top (component side) of the adapter. Static and static forever devices (e.g., an AMD 2960) do not use the CK via connection.

The "feedback output" for a reference element that requires feedback must be connected to via 7 of the personality module. To complete the feedback circuit, identify the pin of the reference element to be used for feedback and connect this pin to the outside pad of via 7 (i.e., the circular pad) using a short, insulated 30 AWG wire on the component side of the personality module. Do NOT cut the trace between the circular pad and the personality module interface pin. If the reference element does not require feedback, via 7 should be grounded (i.e., install a bus wire between the round and square pads of via 7).

## D.5 NO-CONNECT PINS

Unused or no-connect (NC) pins on a reference element should be isolated from the device control module to prevent possible overdriving or feedback. To isolate a no-connect pin, cut the trace at the socket interface pin.

## D.6 JIG ID

Each personality module used in a Realchip or Realmodel system must have a unique jig ID number. Note that since only seven jig ID bits are available on the 68-Pin LCC Personality Module, the jig ID is limited to 126 (jig IDs 0 and 127 are reserved). The jig ID number is read by the Simulator (from the JIG_ID entry in the device definition file) to determine which reference elements are mounted. Valid-supplied personality modules are assigned jig ID numbers in ascending order beginning with 1; to reduce the possibility of jig ID number duplication, user-created

personality modules should be assigned jig ID numbers in descending order beginning with 126.

The individual jig ID pads are located on the left edge of the personality module; pad 0 is the low-order bit, and pad 6 is the high-order bit. A bit is selected or a logic '1' when it is not connected to its adjacent ground (square) pad. To select a jig ID number, ground the appropriate bits by strapping the corresponding round jig ID pads to the adjacent ground pads. As an example of jig ID strapping, jig ID '117' would have ground straps installed in pads 1 and 3 (i.e., bits 0, 2, 4, 5, and 6 would be a logic '1'; binary value 1110101).

## D.7 ALTERNATE SIGNAL WIRING

Since the 68-Pin LCC Personality Module only allows 64 active pins to be defined, when a signal from the reference element appears on pins 9, 26, 43, or 60, a wire must be installed from the corresponding "pin" pad (see section D.3) to an unused socket interface pin. To perform this modification:

1. Identify a NC (no connect) pin on the reference element.

2. Refer to section D.9 to determine the corresponding socket interface pin assignment.

3. On the circuit side of the personality module, cut the trace to the socket interface pin.

4. Install an insulated wire (on the component side) between the "pin" pad to a pad on the cut trace. Note that care must be taken when soldering the wire to avoid shorting the pad to the LCC socket legs.

## D.8 FILTER CAPACITORS

The personality module requires filter capacitors between Vcc and ground. Referring to figure D-1, install a 22uF electrolytic capacitor (noting proper polarity) on the circuit side of the personality module between the VCC and GND pads nearest the "notch." Install a 0.01uF capacitor on the component side between the VCC and GND pads near the "notch," and install two 0.01uF capacitors along the right edge of the personality module. Additional 0.01uF capacitors should be installed around the perimeter of the LCC socket (see figure D-2).

## D.9 SOCKET TO LCC PIN MAPPING

The following table defines the active signal mapping between the socket interface pins and the reference element. The socket pin number in the first column is the pin number entered in the pin block of the device definition file. Note that since there are only 64 active socket interface pins, pins 9, 26, 43, and 60 of the reference element are not brought out to a socket interface pin. If these pins are required (i.e., if the pins correspond to an active signal), see section D.7.

## Socket to LCC Pin Mapping

| Socket Pin Number | LCC Pin Number | Signal Count |
|---|---|---|
| 6 | 1 | 1 |
| 7 | 2 | 2 |
| 8 | 3 | 3 |
| 9 | 4 | 4 |
| 10 | 5 | 5 |
| 11 | 6 | 6 |
| 12 | 7 | 7 |
| 13 | 8 | 8 |
| 14 | 10 | 9 |
| 15 | 11 | 10 |
| 17 | 12 | 11 |
| 18 | 13 | 12 |
| 19 | 14 | 13 |
| 20 | 15 | 14 |
| 21 | 16 | 15 |
| 22 | 17 | 16 |
| 23 | 18 | 17 |
| 24 | 19 | 18 |
| 25 | 20 | 19 |
| 26 | 21 | 20 |
| 27 | 22 | 21 |
| 28 | 23 | 22 |
| 29 | 24 | 23 |
| 31 | 25 | 24 |
| 32 | 27 | 25 |
| 33 | 28 | 26 |
| 34 | 29 | 27 |
| 35 | 30 | 28 |
| 36 | 31 | 29 |
| 37 | 32 | 30 |
| 38 | 33 | 31 |
| 39 | 34 | 32 |

## Socket to LCC Pin Mapping

| Socket Pin Number | LCC Pin Number | Signal Count |
|---|---|---|
| 42 | 35 | 33 |
| 43 | 36 | 34 |
| 44 | 37 | 35 |
| 45 | 38 | 36 |
| 46 | 39 | 37 |
| 47 | 40 | 38 |
| 48 | 41 | 39 |
| 49 | 42 | 40 |
| 50 | 44 | 41 |
| 51 | 45 | 42 |
| 52 | 46 | 43 |
| 54 | 47 | 44 |
| 55 | 48 | 45 |
| 56 | 49 | 46 |
| 57 | 50 | 47 |
| 58 | 51 | 48 |
| 59 | 52 | 49 |
| 60 | 53 | 50 |
| 61 | 54 | 51 |
| 62 | 55 | 52 |
| 63 | 56 | 53 |
| 64 | 57 | 54 |
| 66 | 58 | 55 |
| 67 | 59 | 56 |
| 68 | 61 | 57 |
| 69 | 62 | 58 |
| 70 | 63 | 59 |
| 71 | 64 | 60 |
| 72 | 65 | 61 |
| 73 | 66 | 62 |
| 74 | 67 | 63 |
| 75 | 68 | 64 |

# APPENDIX E
# 64-PIN PGA PERSONALITY MODULE

The 64-pin PGA Personality Module can be used interchangeably with both Realchip and Realmodel. The blank personality module is designed for mounting a PGA (Pin Grid Array) reference element in any of the following pin configurations:

| | |
|---|---|
| 8x8x2 | 8x8x3 |
| 9x9x2 | 9x9x3 |
| 10x10x2 | |

The personality module is 4.1 by 1.9 inches with two rows of 40 pins on the bottom for interface with the Realchip or Realmodel Device Control Module. Figure E-1 shows the blank 64-Pin PGA Personality Module.

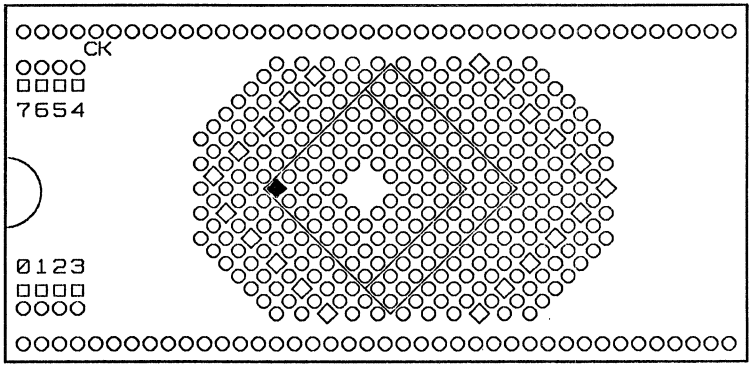


**Figure E-1. 64-Pin PGA Personality Module**

## E.1 PERSONALITY MODULE ORIENTATION

The 64-pin PGA Personality Module contains a central Pin Grid Array (PGA) and a DIP (Dual In-line Package) work area for incorporating required support ICs on either side of the PGA.

### PGA PIN NUMBERING

The rows of pads of the PGA area are named with either a letter or a number so that each pin position can be identified by a unique letter/number pair. Referring to figure E-2, the lettered rows are labeled 'e' through 'a' and 'A' through 'R' (omitting 'I' and 'O'). The numbered rows are labeled '@ 5' through '@ 1' and '1' through '16'. This labeling scheme causes the largest (10x10x2) PGA footprint to be defined by lettered rows 'A' through 'K' and numbered rows '1' through '10' (i.e., pin A1 of the PGA reference element is aligned with coordinates A1).

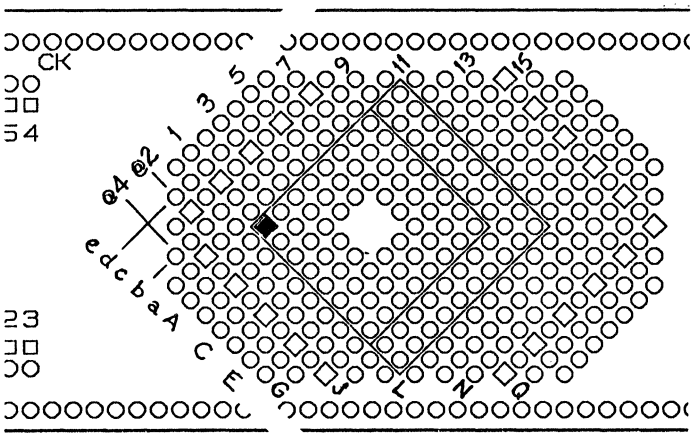


**Figure E-2. PGA Pin Numbering**

### DIP WORK AREA

The DIP work areas on either side of the PGA are each designed to accept up to a 20-pin standard DIP integrated circuit (legs spaced 100 mil center-to-center with a width of 300 mils). At opposite ends of each pin row are a Vcc pad (round) and a ground pad (square).

## E.2 POSITIONING THE REFERENCE ELEMENT

Pin A1 of the device to be modeled must be inserted in pad A1 of the central PGA area on the component side of the board so that the device falls completely within the silkscreened outline or "footprint.". Note that the reference element should always be socket mounted.

The following tables define the active signal mapping between the socket interface pins for the supported reference element sizes. The socket pin number in the table is the pin number entered in the pin block of the device definition file. Note that for each size reference element, there are pins that are not routed to socket interface pins (i.e., the NC or "no connect" entries in the tables).

**Table E-1. Socket to 8x8x2 PGA Pin Mapping**

| PGA Pin | Socket Pin | PGA Pin | Socket Pin |
|---|---|---|---|
| A1 | 75 | A2 | 74 |
| A3 | 72 | A4 | 70 |
| A5 | 68 | A6 | 63 |
| A7 | 61 | A8 | 59 |
| B1 | 6 | B2 | 73 |
| B3 | 71 | B4 | 69 |
| B5 | 67 | B6 | 66 |
| B7 | 64 | B8 | 62 |
| C1 | 7 | C2 | 8 |
| C7 | NC | C8 | NC |
| D1 | 9 | D2 | 10 |
| D7 | 55 | D8 | NC |
| E1 | 11 | E2 | 12 |
| E7 | 49 | E8 | NC |
| F1 | 13 | F2 | 14 |
| F7 | 47 | F8 | NC |
| G1 | 18 | G2 | 15 |
| G3 | NC | G4 | 25 |
| G5 | 27 | G6 | 32 |
| G7 | 34 | G8 | NC |
| H1 | 20 | H2 | 17 |
| H3 | NC | H4 | NC |
| H5 | NC | H6 | NC |
| H7 | NC | H8 | NC |

**Table E-2. Socket to 8x8x3 PGA Pin Mapping**

| PGA Pin | Socket Pin | PGA Pin | Socket Pin | PGA Pin | Socket Pin |
|---|---|---|---|---|---|
| A1 | 75 | A2 | 74 | A3 | 72 |
| A4 | 70 | A5 | 68 | A6 | 63 |
| A7 | 61 | A8 | 59 | | |
| B1 | 6 | B2 | 73 | B3 | 71 |
| B4 | 69 | B5 | 67 | B6 | 66 |
| B7 | 64 | B8 | 62 | | |
| C1 | 7 | C2 | 8 | C3 | NC |
| C4 | NC | C5 | NC | C6 | NC |
| C7 | NC | C8 | NC | | |
| D1 | 9 | D2 | 10 | D3 | NC |
| D6 | 51 | D7 | 55 | D8 | NC |
| E1 | 11 | E2 | 12 | E3 | NC |
| E6 | 50 | E7 | 49 | E8 | NC |
| F1 | 13 | F2 | 14 | F3 | NC |
| F4 | 29 | F5 | 30 | F6 | 31 |
| F7 | 47 | F8 | NC | | |
| G1 | 18 | G2 | 15 | G3 | NC |
| G4 | 25 | G5 | 27 | G6 | 32 |
| G7 | 34 | G8 | NC | | |
| H1 | 20 | H2 | 17 | H3 | NC |
| H4 | NC | H5 | NC | H6 | NC |
| H7 | NC | H8 | NC | | |

**Table E-3. Socket to 9x9x2 PGA Pin Mapping**

| PGA Pin | Socket Pin | PGA Pin | Socket Pin |
|---|---|---|---|
| A1 | 75 | A2 | 74 |
| A3 | 72 | A4 | 70 |
| A5 | 68 | A6 | 63 |
| A7 | 61 | A8 | 59 |
| A9 | 60 | | |
| B1 | 6 | B2 | 73 |
| B3 | 71 | B4 | 69 |
| B5 | 67 | B6 | 66 |
| B7 | 64 | B8 | 62 |
| B9 | 52 | | |
| C1 | 7 | C2 | 8 |
| C8 | NC | C9 | 54 |
| D1 | 9 | D2 | 10 |
| D8 | NC | D9 | 51 |
| E1 | 11 | E2 | 12 |
| E8 | NC | E9 | 50 |
| F1 | 13 | F2 | 14 |
| F8 | NC | F9 | 48 |
| G1 | 18 | G2 | 15 |
| G8 | NC | G9 | 46 |
| H1 | 20 | H2 | 17 |
| H3 | NC | H4 | NC |
| H5 | NC | H6 | NC |
| H7 | NC | H8 | NC |
| H9 | 44 | | |
| J1 | 22 | J2 | 19 |
| J3 | 26 | J4 | 29 |
| J5 | 30 | J6 | 31 |
| J7 | 33 | J8 | 35 |
| J9 | 37 | | |

**Table E-4. Socket to 9x9x3 PGA Pin Mapping**

| PGA Pin | Socket Pin | PGA Pin | Socket Pin | PGA Pin | Socket Pin |
|---|---|---|---|---|---|
| A1 | 75 | A2 | 74 | A3 | 72 |
| A4 | 70 | A5 | 68 | A6 | 63 |
| A7 | 61 | A8 | 59 | A9 | 60 |
| B1 | 6 | B2 | 73 | B3 | 71 |
| B4 | 69 | B5 | 67 | B6 | 66 |
| B7 | 64 | B8 | 62 | B9 | 52 |
| C1 | 7 | C2 | 8 | C3 | NC |
| C4 | NC | C5 | NC | C6 | NC |
| C7 | NC | C8 | NC | C9 | 54 |
| D1 | 9 | D2 | 10 | D3 | NC |
| D7 | 55 | D8 | NC | D9 | 51 |
| E1 | 11 | E2 | 12 | E3 | NC |
| E7 | 49 | E8 | NC | E9 | 50 |
| F1 | 13 | F2 | 14 | F3 | NC |
| F7 | 47 | F8 | NC | F9 | 48 |
| G1 | 18 | G2 | 15 | G3 | NC |
| G7 | 34 | G8 | NC | G9 | 46 |
| H1 | 20 | H2 | 17 | H3 | NC |
| H4 | NC | H5 | NC | H6 | NC |
| H7 | NC | H8 | NC | H9 | 44 |
| J1 | 22 | J2 | 19 | J3 | 26 |
| J4 | 29 | J5 | 30 | J6 | 31 |
| J7 | 33 | J8 | 35 | J9 | 37 |

**Table E-5. Socket to 10x10x2 PGA Pin Mapping**

| PGA Pin | Socket Pin | PGA Pin | Socket Pin |
|---|---|---|---|
| A1 | 75 | A2 | 74 |
| A3 | 72 | A4 | 70 |
| A5 | 68 | A6 | 63 |
| A7 | 61 | A8 | 59 |
| A9 | 60 | A10 | 57 |
| B1 | 6 | B2 | 73 |
| B3 | 71 | B4 | 69 |
| B5 | 67 | B6 | 66 |
| B7 | 64 | B8 | 62 |
| B9 | 52 | B10 | 58 |
| C1 | 7 | C2 | 8 |
| C9 | 54 | C10 | 56 |
| D1 | 9 | D2 | 10 |
| D9 | 51 | D10 | 55 |
| E1 | 11 | E2 | 12 |
| E9 | 50 | E10 | 49 |
| F1 | 13 | F2 | 14 |
| F9 | 48 | F10 | 47 |
| G1 | 18 | G2 | 15 |
| G9 | 46 | G10 | 45 |
| H1 | 20 | H2 | 17 |
| H9 | 44 | H10 | 43 |
| J1 | 22 | J2 | 19 |
| J3 | 26 | J4 | 29 |
| J5 | 30 | J6 | 31 |
| J7 | 33 | J8 | 35 |
| J9 | 37 | J10 | 42 |
| K1 | 21 | K2 | 24 |
| K3 | 23 | K4 | 25 |
| K5 | 27 | K6 | 32 |
| K7 | 34 | K8 | 36 |
| K9 | 38 | K10 | 39 |

## E.3 UNCONNECTED PGA CONNECTIONS

The following pins of the PGA are interconnected, but have no interface to the personality module socket interface pins (these pins are designated 'NC' in tables E-1 through E-5). To use the table, locate the desired pad in the footprint area in the first column to find the pad to wire to the pad corresponding to the desired socket interface pin described in the next section.

**Table E-6. Unconnected PGA Pins**

| Footprint Pad | Interconnect Pad |
|---|---|
| C3 | a@ 1 |
| C4 | a3 |
| C5 | a7 |
| C6 | b9 |
| C7 | a10 - A11 |
| C8 | B11 - B12 |
| D3 | d@ 2 |
| D8 | C13 |
| E3 | B@ 2 |
| E8 | E11 - E12 |
| F3 | F@ 2 |
| F8 | J11 - J12 |
| G3 | J@ 2 |
| G8 | L11 - M12 |
| H3 | K@ 1 - L1 |
| H4 | L2 - M2 |
| H5 | L3 - N3 |
| H6 | L6 - N6 |
| H7 | L10 - N10 |
| H8 | N13 |

As an example, if pin D8 on an 8x8x2 PGA is defined as an active signal, a jumper wire would be installed between pad C13 and one of the unassigned socket interface pins defined in table E-7. pad

The following table lists the unassigned socket interface pins for each of the supported PGA pin configurations. The number in parentheses following the pin number is the PGA pad that is connected to the socket interface pin. Note that this pad can be jumpered to one of the PGA connections outside of the footprint area (see table E-6) to route a NC (no connect) pad to a socket interface pin.

**Table E-7. Unassigned Socket Interface Pins**

| PGA Size | Unused Pin Number | | | |
|---|---|---|---|---|
| 8x8x2 | 19(F@5) | 21(G@4) | 22(F@4) | 23(Q6) |
| | 24(Q5) | 26(R6) | 29(R7) | 30(R8) |
| | 31(R9) | 33(R10) | 35(R11) | 36(Q11) |
| | 37(R12) | 38(Q12) | 39(Q13) | 42(N15) |
| | 43(M15) | 44(M16) | 45(L15) | 46(L16) |
| | 48(K16) | 50(J16) | 51(H16) | 52(F16) |
| | 54(G16) | 56(G15) | 57(E15) | 58(F15) |
| | 60(d7) | | | |
| 8x8x3 | 19(F@5) | 21(G@4) | 22(F@4) | 23(Q6) |
| | 24(Q5) | 26(R6) | 33(R10) | 35(R11) |
| | 36(Q11) | 37(R12) | 38(Q12) | 39(Q13) |
| | 42(N15) | 43(M15) | 44(M16) | 45(L15) |
| | 46(L16) | 48(K16) | 52(F16) | 54(G16) |
| | 56(G15) | 57(E15) | 58(F15) | 60(d7) |
| 9x9x2 | 21(G@4) | 23(Q6) | 24(Q5) | 25(Q7) |
| | 27(Q8) | 32(Q9) | 34(Q10) | 36(Q11) |
| | 38(Q12) | 39(Q13) | 42(N15) | 43(M15) |
| | 45(L15) | 47(K15) | 49(J15) | 50(J16) |
| | 51(H16) | 55(H15) | 56(G15) | 57(E15) |
| | 58(F15) | | | |
| 9x9x3 | 21(G@4) | 23(Q6) | 24(Q5) | 25(Q7) |
| | 27(Q8) | 32(Q9) | 36(Q11) | 38(Q12) |
| | 39(Q13) | 42(N15) | 43(M15) | 45(L15) |
| | 47(K15) | 49(J15) | 56(G15) | 57(E15) |
| | 58(F15) | | | |
| 10x10x2 | none | | | |

## E.4 POWER AND GROUND WIRING

No power (Vcc) and ground are routed to the footprint area and must be provided by jumper wire. Rows @ 3, 14, c and P are made up of alternating ground and Vcc pads; the square pads are ground, and the round pads are Vcc. If the Vcc and ground pins on the reference element are routed to socket interface pins (see tables E-1 through E-5), the corresponding traces between the inboard and outboard pads in table E-8 must be cut and jumper wires installed between Vcc/ground and the inboard pads. If the Vcc and ground pins on the reference element are not routed to a socket interface pin, refer to table E-6 and connect jumpers between Vcc and the specified interconnect pad and between ground and the specified interface pad.

## E.5 CLOCK AND FEEDBACK WIRING

In order to use the hardware clock (dynamic devices), connect an insulated wire (on the component side) from socket interface pin 76 (the CK pin) to the clock pin of the reference element. If the CLOCK pin on the reference element is routed to a socket interface pin (see tables E1 through E-5), the corresponding trace between the inboard and outboard pads in table E-8 must be cut and a jumper installed between the CK pin and the inboard pad. Note that if the trace is not cut, the clock signal will be fed back to the device control module. If the CLOCK pin on the reference element is not routed to a socket interface pin, refer to table E-6 and connect a wire between the CK pin and the specified interconnect pad.

When a feedback signal is required, connect an insulated wire from the round pad adjacent to socket interface pin 80 (the pad next to pad '7') to the pin of the reference element being used as the feedback signal. When feedback is not being used, this pad should be connected to the adjacent square pad (pad '7').

## E.6 NO-CONNECT PINS

Unused or no-connect (NC) pins on a reference element should be isolated from the device control module to prevent possible overdriving or feedback. To isolate a no-connect pin, identify the corresponding socket interface pin (see tables E-1 through E-9) and then cut the trace between the outboard pad connected to the interface pin and the inboard pad into the array defined in table E-8.

## E.7 SOCKET INTERFACE PIN ASSIGNMENTS

The following table lists the PGA pads that are connected directly to the 64 active signal socket interface pins. Note that when cuts are required to isolate any of the socket interface pins, all cuts are made in rows @ 5, e, 16, or R on the component side and in rows @ 4, d, 15, or Q on the circuit side. Generally, cuts are only necessary for connecting power and ground, NC (no-connect) pins, and the clock and feedback signals.

**Table E-8. Socket Interface Pin Assignments**

| Socket Pin | Outboard Pad | Inboard Pad |
|---|---|---|
| 6 | b@ 4 | A@ 1 |
| 7 | a@ 4 | B@ 1 |
| 8 | a@ 5 | a@ 2 |
| 9 | A@ 4 | C@ 1 |
| 10 | A@ 5 | A@ 2 |
| 11 | B@ 4 | D@ 1 |
| 12 | B@ 5 | C@ 2 |
| 13 | C@ 4 | E@ 1 |
| 14 | C@ 5 | D@ 2 |
| 15 | D@ 5 | Ea@ |
| 17 | E@ 5 | G@ 2 |
| 18 | D@ 4 | F@ 1 |
| 19 | F@ 5 | H@ 2 |
| 20 | E@ 4 | G@ 1 |
| 21 | G@ 4 | J@ 1 |
| 22 | F@ 4 | H@ 1 |
| 23 | Q6 | M4 |
| 24 | Q5 | M3 |
| 25 | Q7 | M5 |
| 26 | R6 | N4 |
| 27 | Q8 | M6 |
| 29 | R7 | N5 |
| 30 | R8 | N7 |
| 31 | R9 | N8 |
| 32 | Q9 | M7 |
| 33 | R10 | N9 |
| 34 | Q10 | M8 |
| 35 | R11 | N11 |
| 36 | Q11 | M9 |
| 37 | R12 | N12 |
| 38 | Q12 | M10 |
| 39 | Q13 | M11 |

**Table E-8. Socket Interface Pin Assignments** (Con't)

| Socket Pin | Outboard Pad | Inboard Pad |
|---|---|---|
| 42 | N15 | M13 |
| 43 | M15 | L13 |
| 44 | M16 | L12 |
| 45 | L15 | K13 |
| 46 | L16 | K12 |
| 47 | K15 | J13 |
| 48 | K16 | H12 |
| 49 | J15 | H13 |
| 50 | J16 | G12 |
| 51 | H16 | F12 |
| 52 | F16 | C12 |
| 54 | G16 | D12 |
| 55 | H15 | G13 |
| 56 | G15 | F13 |
| 57 | E15 | D13 |
| 58 | F15 | E13 |
| 59 | d6 | b7 |
| 60 | d7 | b8 |
| 61 | d5 | b6 |
| 62 | e6 | a9 |
| 63 | d4 | b5 |
| 64 | e5 | a8 |
| 66 | e4 | a6 |
| 67 | e3 | a5 |
| 68 | d3 | b4 |
| 69 | e2 | a4 |
| 70 | d2 | b3 |
| 71 | e1 | a2 |
| 72 | d1 | b2 |
| 73 | e@ 1 | a1 |
| 74 | d@ 1 | b1 |
| 75 | d@ 2 | b@ 1 |

## E.8 JIG ID STRAPPING

Each personality module used in a Realchip or Realmodel system must have a unique jig ID number. Note that since only seven jig ID bits are available on the 64-Pin PGA Personality Module, the jig ID is limited to 126 (jig IDs 0 and 127 are reserved). The jig ID number is read by the Simulator (from the JIG_ID entry in the device definition file) to determine which reference elements are mounted. Valid-supplied personality modules are assigned jig ID numbers in ascending order beginning with 1; to reduce the possibility of jig ID number duplication, user-created personality modules should be assigned jig ID numbers in descending order beginning with 126.

The individual jig ID pads are located on the left edge of the personality module; pad 0 is the low-order bit, and pad 6 is the high-order bit. A bit is selected or a logic '1' when it is not connected to its adjacent ground (square) pad. To select a jig ID number, ground the appropriate bits by strapping the corresponding round jig ID pads to the adjacent ground pads. As an example of jig ID strapping, jig ID '117' would have ground straps installed in pads 1 and 3 (i.e., bits 0, 2, 4, 5, and 6 would be a logic '1'; binary value 1110101).

## E.9 FILTER CAPACITORS

The personality module requires filter capacitors between Vcc and ground. Referring to figure E-1, install a 22uF electrolytic capacitor (noting proper polarity) between pads GND and '+' on the right side of the personality module. Install a 0.01uF capacitor on each side of the electrolytic capacitor and install a 0.01uF near the "notch" on the left side of the personality module. Additional 0.01uF capacitors should be installed around the perimeter of the PGA between the round and square pads in rows @ 3, 14, c, and P.

# INDEX

128-pin Realchip personality module, B-1
128-pin Realmodel personality module, A-1
64-pin DIP personality module, C-1
64-pin PGA personality module, E-1
68-pin LCC personality module, D-1
8086mx directory structure, 4-39
8086mx model, 4-39

ABBREV property, 4-7, 4-8, 4-10
addphysinfo errors, 4-12
addphysinfo script, 4-10, 4-12
allmsprim.def file, 2-14, 3-4, 3-7, 4-38, 4-42

BIDIRECTIONAL property, 4-8
.BODY drawing, 4-4
body, DRAWING, 4-7, 4-8, 4-10
body properties, NEEDS_NO_SIZE, 4-6
body shape, 4-5
bubbled pins, 4-28

chips.dat file, 4-9, 4-11, 4-12
clock pin
    transitions, 4-16
    wiring, 4-16
clock specifications, 1-5
CLOCK_EDGE directive, 4-31, 4-33
CLOCK_PERIOD directive, 4-31
CLOCK_PIN directive, 4-31
cmpexp.dat file, 2-15, 3-4
cmplst.dat file, 4-9, 4-11
cmpsyn.dat file, 3-4
compilation, 3-3
Compiler errors, 4-9, 4-11
Compiler listing file, 4-9
compiler.cmd file, 3-3, 4-9, 4-11, 4-41

compiling
for chips, 4-9, 4-11
for LOGIC, 3-3
for SIM, 3-3, 4-41
creating .prt file (drawing method), 4-7
creating .prt file (text file method), 4-10

DELAY pin property, 4-25
delay table example, 4-37
delay values, 4-29
demo circuit, simulation, 2-15, 3-4
demo circuit, 2-15
device control module, 1-2, 1-5
addressing, 2-2
cabling, 2-8
clock termination, 2-4
configuration, 2-2
installation, 2-8
jumpers, 2-2
number supported, 2-2
termination resistors, 2-4, 2-8
device definition file
bubbled pins, 4-28
clock block, 4-31
CLOCK_EDGE directive, 4-16, 4-31, 4-33, 4-45
CLOCK_PERIOD directive, 4-31, 4-45
CLOCK_PIN directive, 4-31
CLOCK_PIN, 4-16
contents, 4-22
delay table block, 4-37
directives, 4-22
errors, 3-7, 4-42, 4-48
examples, 4-19, 4-50
extension, 4-19
format, 4-22
JIG_ID directive, 4-30
JIG_TYPE directive, 4-31
low-asserted pins, 4-24
NUMBER_DEV_PINS directive, 4-23

device definition file (con't)
pin block
INPUT_SPEC section, 4-24
IO_SPEC section, 4-24
order, 4-24
OUTPUT_SPEC section, 4-24
pin properties, 4-25
sections, 4-24
pin section, format, 4-24
reset sequence, 4-32
STROBE_PIN property, 4-14
device types
dynamic, 4-13, 4-16
selection criteria, 4-18
static forever, 4-13, 4-17
static, 4-13
directives
CLOCK_EDGE, 4-31, 4-33
CLOCK_PERIOD, 4-31
CLOCK_PIN, 4-31
device definition file, 4-22
DIRECTORY, 3-3
JIG_ID, 4-30
JIG_TYPE, 4-31
LIBRARY, 3-3
LIBRARY_FILE, 3-5
NUMBER_DEV_PINS, 4-22, 4-23
REALCHIP_LIBRARY, 3-4, 6-2
REMOTE_HOST, 6-2
SAMPLE=SPECIAL, 4-48
USE_REALFAST, 3-4
directory
creating, 4-3
mslogic.lib, 4-4
mssim.lib, 4-4
structure, 2-13, 4-2
DIRECTORY directive, 3-3
directory structure, 8086mx, 4-39
drawing, .BODY, 4-4
drawing, .PRIM, 4-6
drawing conventions, 4-5
DRAWING body, 4-7, 4-8, 4-10

driver/receiver specifications, 1-5
dynamic devices
    considerations, 4-16
    pattern presentation, 4-16
    requirements, 4-18
    with multi-phase clocks, 4-17

environment sequence, 4-13
environmental specifications, 1-6
error messages, 3-7
errors
    addphysinfo, 4-12
    Compiler, 4-11
    device definition file, 3-7
    modeling, 4-38
    network, 6-4
    Realchip, 3-6
    Realmodel, 3-6
    simulation acceleration, 3-12
evaluation engine, 1-3
event engine, 1-3
expansion file, *see* cmpexp.dat
external power supplies
    ac power connection, 2-11
    connector pin assignments, 2-10
    input connectors, 2-9
    mating connector, 2-10
    socket pin assignments, 2-9
    switch closure connector 2-12

FAMILY property, 4-8
feedback
    device limitations, 4-34
    signal requirements, 4-37
    wiring, 4-34
file permissions, 2-12, 3-2

files
allmsprim.def, 2-14, 3-4, 3-7, 4-38, 4-42
chips.dat, 4-9, 4-11
cmpexp.dat, 3-4
cmplst.dat, 4-9, 4-11
cmpsyn.dat, 3-4
compiler.cmd, 3-3, 4-9, 4-11, 4-41
device definition, 4-19
liblst.dat, 4-12
master.lib, 3-2
mslogic.lib, 4-41
mssim.lib, 4-41
packager.cmd, 3-5
phys_dat, 4-11
.prt, 4-7, 4-10
simlst.dat, 3-6, 4-42
simulate.cmd, 2-15, 3-4, 4-3, 4-42
startup.ged, 3-2, 4-3
synonym, 2-15
FLOAT_DELAY pin property, 4-25

Graphics Editor
ADD command, 3-2
USE command, 3-2
grid settings, 4-4

hardare modeling specifications, 1-4
hardware clock, 4-44
hardware modeling, 1-1

INPUT_LOAD property, 4-8

jig ID number assignment, 4-30
jig ID, 4-30
JIG_ID directive, 4-30
JIG_TYPE directive, 4-31

liblst.dat file, 4-12
LIBRARY directive, 3-3
library file, *see* .prt file
LIBRARY_FILE directive, 3-5
loading software, 2-13

makeallmsprim script, 2-14, 3-4, 3-7, 4-38, 4-42
    messages, 4-38
    when to run, 4-38
master control module, 1-3
master.lib file, 3-2
model definition, 2-1
model types, *see* device types
model verification, 4-38
modeling considerations, 4-42
modeling errors, 4-38, 4-42
moving the chip.dat file, 4-9, 4-12
mslogic.lib directory, 4-4
mslogic.lib file, 4-41
mssim.lib directory, 4-4
mssim.lib file, 4-41

NC_PIN pin property, 4-26
NEEDS_NO_SIZE property, 4-6
network errors, 6-4
Network Server, restarting, 6-6
Networked Realchip
    bringing down, 6-5
    bringing up, 6-5
    platforms, 6-1, 6-3
    server, 6-1
    simulator modes, 6-2
NUMBER_DEV_PINS directive, 4-23

OUTPUT_LOAD property, 4-8
OUTPUT_TYPE pin property, 4-27
OUTPUT_TYPE property, 4-8

packager.cmd file, 3-5
packaging, 3-5
pattern memory, 4-43
    capacity, 1-3, 1-4
    utilization, 1-3
pattern presentation
    clock timing, 4-44
    CLOCK_EDGE directive affect, 4-45
    dynamic devices, 4-45
    rate, 4-43

pattern presentation (con't)
setup/hold timing, 2-5
skew, 4-44
static devices, 4-45
stretching, 4-44
with multiple elements, 4-43
pause sequence
example, 4-35
pattern definition, 4-34
presentation, 4-35
permissions, *see* file permissions
personality module
128-pin Realchip, B-1
128-pin Realmodel, A-1
64-pin DIP, C-1
64-pin PGA, E-1
68-pin LCC, D-1
capacity, 2-6
installation, 2-5, 2-7
interchangeability, 2-6
jig ID, 4-30
orientation, 2-6
size, 2-5
socket numbering, 2-7
supported, 1-7
phys_dat file, 4-11
adding properties, 4-11
creating, 4-11
example, 4-12
pin properties, 4-11
in device definition file
DELAY, 4-25
FLOAT_DELAY, 4-25
NC_PIN, 4-26
OUTPUT_TYPE, 4-27
STROBE_PIN, 4-26
TS_PIN, 4-26
pins
bubbled, 4-5
denoting, 4-5
labeling, 4-5
naming, 4-5

PIN_NUMBER property, 4-8
POWER_PINS property, 4-8
.PRIM drawing, 4-6
properties, pin, 4-11
property
    ABBREV, 4-7, 4-8, 4-10
    BIDIRECTIONAL, 4-8
    FAMILY, 4-8
    INPUT_LOAD, 4-8
    NEEDS_NO_SIZE, 4-6
    OUTPUT_LOAD, 4-8
    OUTPUT_TYPE, 4-8
    PIN_NUMBER, 4-8
    POWER_PINS, 4-8
    TITLE, 4-7, 4-8, 4-10
.prt file, 4-7, 4-10

Realchip errors, 3-6
Realchip libraries, 3-1
REALCHIP_LIBRARY directive, 3-4, 6-2
Realfast errors, *see* simulation acceleration errors
Realmodel errors, 3-6
REMOTE_HOST directive, 6-2
reset sequence, 4-13, 4-32
    CLOCK_EDGE directive, 4-33
    considerations, 4-36
    example, 4-33
    format, 4-33
    pattern presentation, 4-33
    pause sequence, 4-34
    with reset circuitry, 4-33

SAMPLE=SPECIAL directive, 4-48
sampling, 4-45
    circuit, 4-46
    I/O pins, 4-47
    input pins, 4-47
    output pins, 4-47
    rules, 4-49
    sequence, 4-13

setup/hold time strapping, 2-5
simlst.dat file, 3-6, 4-42
simulate.cmd file, 2-15, 3-4, 4-3, 4-42
simulating, 3-4
simulation acceleration, 1-3, 3-4
    errors, 3-12, 4-42
    limitations, 3-6
    specifications, 1-6
simulator limitations, 3-5
software contents, 2-12
software installation, 2-12
srobe pin, setup and hold time, 4-15
startup.ged file, 3-2, 4-3
static forever devices, 4-17
strobe pin
    declarations, 4-15
    definition, 4-14
    example, 4-15
    identification, 4-26
strobe pins, 4-14, 4-26
STROBE_PIN pin property, 4-26
synonym file, 2-15, *see also* cmpsyn.dat

test drawing, 4-40
test programs, 4-41
test script, 4-41
TITLE property, 4-7, 4-8, 4-10
TS_PIN pin property, 4-26

USE command, 3-2
USE_REALFAST directive, 3-4

verifying installation, 2-15
verifying models, 4-38