efficēon

Software Configuration Guide

Transmeta™ Efficeon™ TM8x00
Hardware Revision 1.x
Code Morphing™ Software Version 6.x

Transmeta PROPRIETARY Information
Provided Under Nondisclosure Agreement

Preliminary Information—SUBJECT TO CHANGE

September 10, 2003

TRANSMETA Corporation

Property of:

Transmeta Corporation
3990 Freedom Circle
Santa Clara, CA 95054
USA
(408) 919-3000
http://www.transmeta.com

Directory of Contents

Directory of Tables

Directory of Figures

Chapter 1

Code Morphing Software Overview

The following chapters provide information necessary to configure Code Morphing™ software (CMS), the software portion of the Efficeon™ TM8x00 processor.

Reference Documents

The following documents should be used in conjunction with this guide:

- *Efficeon™ Processor Data Book*
- *Efficeon™ Processor BIOS Programmer's Guide*
- *Efficeon™ Processor Bringup and Configuration Tools*
- *Efficeon™ Processor System Design Guide (Preliminary)*
- JEDEC Standard No. 21-C (11/25/97)

Changes from 7/10/03 Revision

- Updated entire document with Efficeon markings.
- Updated Chapter 2, *Code Morphing Software Image*.
- Added Chapter 3, *Memory Configuration*.
- Updated Chapter 4, *OEM Configuration Table*. Added documentation for the several fields.
- Added Chapter 5, *Code Morphing Software: Initial State*.
- Added Appendix A, *POST Codes*.

Chapter 2

Code Morphing Software Image

This chapter describes the Code Morphing™ software image and how to create it. This chapter also provides an overview of the process of bringing up an Efficeon processor in a development laboratory or pre-production environment, and configuring it correctly for production.

The nature of an Efficeon processor is different from other products of its class in two significant ways:

- Parts of the processor itself are implemented in software. This enables many features not feasible in hardware-only processors. For example, upgrades can be devised that involve a customer simply executing a small, downloadable utility to flash a “processor upgrade” into ROM, with no swapping motherboards or soldering required.

- The processor converts application code, in particular x86 code, into its own machine code on the fly. This enables the processor to emulate other popular processors.

These innovations require different configuration techniques from other types of processors. There are several steps required to bring up and configure a Transmeta Efficeon processor. In addition, there are policies for memory layout, future processor upgrades, and other features that must be decided.

Code Morphing software provides many innovative services such as dramatically improved power management and thermal management capabilities (LongRun™), a Virtual Northbridge (VNB) that eliminates the need for a hardware northbridge, and a translation caching mechanism that enables the processor to “learn” how users interact with their applications and increase actual working speed while running.

Since Code Morphing software loads and runs even before the BIOS, it must reside in a ROM where the processor can find it and boot from it. Generally, you create a Code Morphing software image with a Transmeta-supplied utility, then upload this image into the ROM using a ROM programmer or a Transmeta-supplied device (see the book *Efficeon™ Tools Guide*). The new image is used to boot the system on reset.

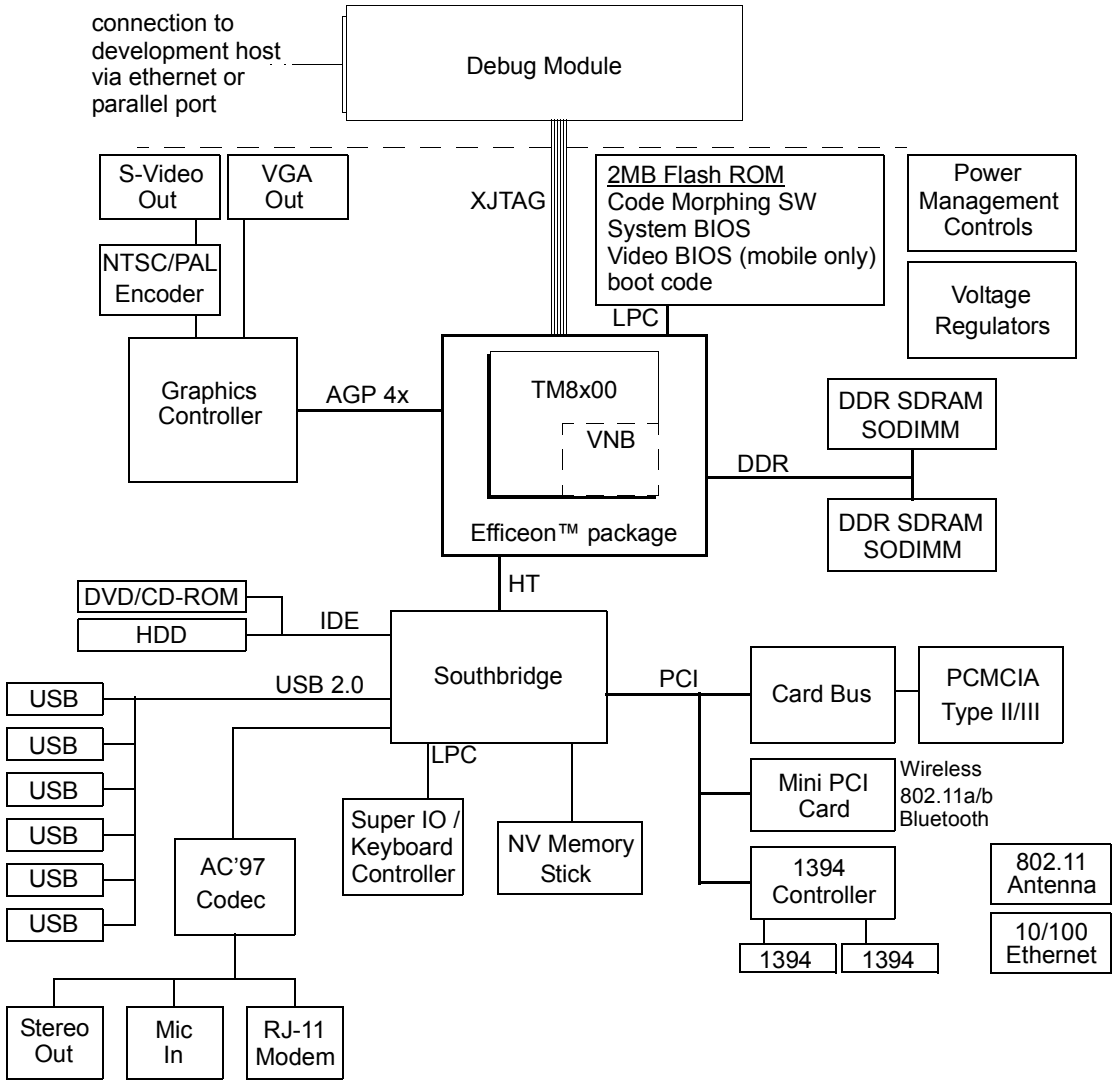
2.1 Hardware Environment

In general, an Efficeon chip resides in a framework where it communicates with a southbridge via an HT bus, a flash LPC ROM module via an LPC bus, a graphics module via an AGP bus, and a DDR RAM memory module. Northbridge functions such as power and memory management are provided on the Cruose package in a Virtual Northbridge (VNB)—no separate hardware northbridge is necessary. Code Morphing™

Software, which translates x86 instructions into VLIW instructions for the processor, resides with the BIOS in a serial ROM accessed by LPC.

The following diagram shows the Efficeon package laid out with other necessary components, including the Transmeta Debugging Module (TDM). (For more information on the TDM, see the *Efficeon Tools Guide*.) Note that this diagram is for illustrative purposes only; for a detailed discussion of the hardware environment, including true schematic diagrams, see the *Efficeon TM8x00 System Design Guide*.

Figure 1: Efficeon Package in a Bringup or Debugging Environment



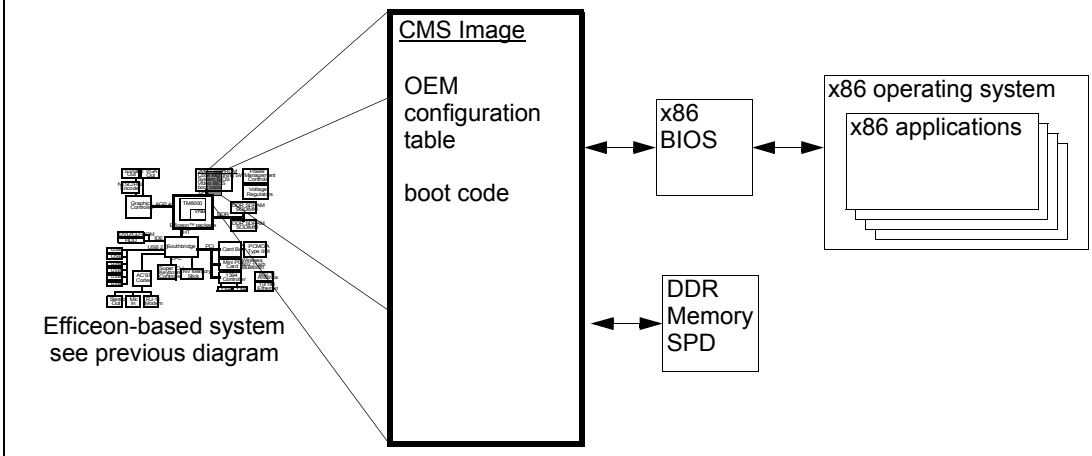
Virtual Northbridge (VNB)	System controller, implemented on the chip.
LPC ROM	Provides storage for boot code, BIOS, and Code Morphing™ software.
AGP Controller	AGP 4x graphics controller and SVGA adapter on desktop/server configurations. On mobile configurations, video resides in the LPC ROM.
DDR Memory Modules	The TM8x00 provides two double-data-rate (DDR) DIMM slots.
Southbridge	Peripheral bus controller, accessed via HyperTransport (HT) bus.
Debug Module	The TDM is a separate debugging module available from Transmeta. It connects to an Efficeon chip via an XJTAG connection.

2.2 Software Environment

Unlike most other microprocessors, the Efficeon core also contains a robust software environment that handles many tasks that are controlled by hardware on other platforms. The centerpiece of this design is Code Morphing™ software (CMS), which provides a translation service from x86 code to Efficeon-native VLIW code.

The following diagram shows the arrangement of the Code Morphing software image in the Efficeon package.

Figure 2: Code Morphing Software Image as a Software Component



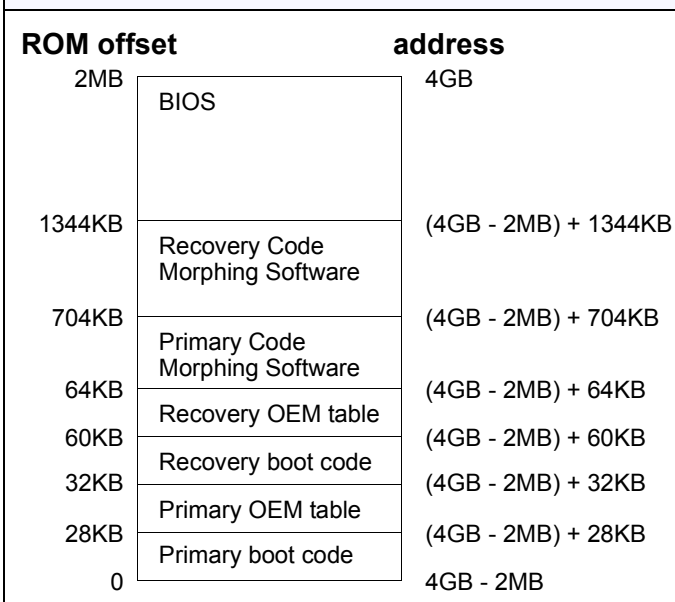
Code Morphing Software (CMS) image	CMS is a configurable, field-upgradable software component of the processor. The primary task of CMS is to translate x86 instructions into Efficeon-native VLIW instructions. The CMS image contains two copies of itself in order to recover in case one copy becomes corrupted during an interrupted upgrade.
OEM Configuration Table	A field-upgradable[1] configuration table which contains variables that can be modified to suit your system's needs. For more information, see Chapter 4, *OEM Configuration Table* on page 23.
boot code	The CMS image contains the CPU's field-upgradable boot code. It is the first software loaded when the system starts. It configures the memory ports and selects and decompresses one of the two copies of CMS for execution.
BIOS	A developed x86 BIOS, based on the needs of the system. Transmeta has developed a reference BIOS, or you can develop your own BIOS. For more information, see the *BIOS Programmer's Guide*.
DDR Memory SPD	Code Morphing software sizes and configures system memory using information supplied in the memory SPD (Serial Presence Detect) chips.

1. The OEM configuration table and boot code are field-upgradable, provided the LPC ROM in use has its bottom 64kb split into independently-erasable 4kb sectors. For more information, see TBD.

2.3 Configuration Overview

2.3.1 ROM Sections

A 2MB (16 Mbit) Code Morphing software ROM image consists of several logical components, described below.



1KB = 1024 bytes, 1MB = 1024KB, and 1GB = 1024MB

The non-BIOS components of CMS are readable through x86 space at these addresses if ROM windows in the northbridge are programmed appropriately.

- **Boot code**. These are the first native (VLIW) instructions executed upon reset. It includes some power-on self-test (POST) code and a decompressor, necessary because Code Morphing software is stored in compressed form. Also contains memory configuration code.
- **Code Morphing software**. This software translates x86 instructions from the operating system, applications, drivers, etc. into VLIW instructions for the processor, using configuration information found in the OEM configruation table.
- **OEM configuration table**. This is a data structure programmed by the OEM/ODM to control several aspects of the behavior of the CPU, the boot code, and Code Morphing software.
- **Recovery boot code/OEM configuration table/Code Morphing software**. Backup copies of each of the main components of the image.

2.3.2 Creating a CMS Image

A CMS image is created by first programming the OEM configuration table by editing the OEM configuration template, and then compiling an image with the Transmeta-supplied utility makerom. The image is uploaded to the LPC ROM using the utility flashtool, and the system is rebooted.

For information about makerom, flashtool, and other Transmeta utilities, see the *Efficeon Tools Guide*.

Chapter 3

Memory Configuration

There are several parameters in the OEM configuration table that affect memory configuration by the Code Morphing software boot code. This chapter outlines memory configuration parameters from a hands-on perspective.

Refer to Chapter 4, *OEM Configuration Table* on page 23 for more details on individual parameters.

3.1 DIMM Slots

The Efficeon memory controller supports up to 4 logical DIMMs. That is, although it supports 8 *ranks* of memory, the ranks are collected into one-ranked or symmetric two-ranked DIMMs. That is, there is no way to have 8 independent ranks. At most there are 4 DIMMs, each two-ranked, but the two ranks in each DIMM have to be equal, i.e., same kinds of chips and organizations on both ranks of a DIMM.

This is independent of whether physical (So)DIMMs or soldered-down memory are used. In other words, even if memory is soldered down, it must be described to the controller as a logical DIMM that may be one-ranked or two-ranked, but if it is two-ranked, then the two ranks have to be identical.

Thus the memory configuration code is organized around the concept of *DIMM slots*, which are the logical places where a DIMM is connected. Of course, for soldered-down memory there is no physical DIMM slot, but there is still the concept, and this is how the memory is described to the software and the controller.

Note that although the controller supports up to 4 logical DIMMs, it can not support 4 physical DIMMs because it does not have enough clock pairs for the general case. In specific configurations (registered DIMMs only use one clock pair) and specific boards, it may support 4. Nothing above two is supported at this time, but the controller and the software is forward-compatible in case a reference design without these constraints is created at some point.

3.2 Required Memory Configuration

Following is the short list of parameters that OEMs are expected to set, all in the OEM-Managed section of the configuration table:

- mem_probe_spd (also see *mem_probe_spd* on page 28)
- mem_sbus_spd_base_addr (also see *mem_smbus_spd_base_addr* on page 28)
- mem_slot_to_clocks[4] (also see *mem_slot_to_clocks[4]* on page 28)
- mem_spd[4] (also see *mem_spd[4]* on page 29)

mem_probe_spd

mem_probe_spd is a 4-bit bitmask that tells the boot code whether to read SPD data from the SPD ROM in a DIMM, or from the OEM configuration table. There is one bit per logical DIMM slot (4 theoretically). Bit 0 of the bitmask corresponds to slot 0, and so on.

A bit value of 0 means that the SPD data is to be found in the OEM config table, i.e. this 'DIMM slot' is a soldered-down DIMM and we are using the OEM config table to contain the SPD data rather than soldering down a separate SPD ROM.

A bit value of 1 means that the SPD data is to be found in the SPD ROM for this DIMM slot. In real terms, this “DIMM slot” is a true DIMM slot that can have an arbitrary DIMM inserted into it, or otherwise looks enough like a DIMM that it has a separate SPD ROM readable through the SMBUS.

Transmeta's current reference design boards have two DIMM slots (slots 0 and 1) and no soldered down memory. Thus we use the default value 3 (binary 0011), to make the boot code read SPD data from the SMBUS for slots 0 and 1, and from the OEM configuration table for slots 2 and 3 (but see mem_slot_to_clocks below).

mem_sbus_spd_base_addr

mem_smbus_spd_base_addr is the SMBUS address of the SPD ROM for slot 0. The other SPD ROMs are assumed to be at increasing SMBUS addresses from this one. Note that if slot 0 is soldered-down, but slot 1 really has a DIMM slot, this field still needs to contain the SMBUS address that slot 0 would have used, i.e. 1 less than the SMBUS address for slot 1.

For Transmeta's current reference design boards, the value is 0x50 (decimal 80) meaning that the SPD ROM for DIMM slot 0 is at SMBUS address 0x50, the SPD ROM for DIMM slot 1 is at address 0x51, etc.

mem_slot_to_clocks[4]

mem_slot_to_clocks[4] is an array of 4 8-bit bitmasks. This array tells the boot code what clock pairs to enable for each DIMM slot that is present.

If a slot has no clocks, then the boot code assumes that that slot is not present and does not try to configure it. Similarly, if the boot code decides that there is no memory in some DIMM slot, it does not turn its clocks on, to save a little power. On Transmeta's current reference design boards, the clocks are set to 0 for slots 2 and 3 in order to turn them off.

Each clock pair is represented by a bit in an 8-bit bitmask. Clock pair 0 corresponds to bit 0 and clock pair 7 corresponds to bit 7.

DIMMs generally require 3 clock pairs each. A board designer can use any 3 of the 8 available clock pairs and needs to inform the boot code of which pairs are used for which DIMM slot. On Transmeta's current reference design boards, clock pairs 0-2 are used for slot 0, clock pairs 3-5 are used for slot 1, and clock pairs 6 and 7 are unused.

Thus, mem_slot_to_clocks[4] defaults to { 0x07, 0x38, 0, 0 } for Transmeta's current reference design boards.

If a designer solders down some memory, clock pair assignments may differ, and there mayb be fewer than 3 clock pairs for the soldered-down memory. For example, on a hypothetical board with a soldered-down DIMM slot using clock pairs 0 and 1, and two DIMM slots using clock pairs 2-4 and 5-7 respectively, the board would have mem_slot_to_clocks[4] set to { 0x3, 0x1c, 0xe, 0 }.

mem_spd[4]

mem_spd is an array of 4 SPD data structures. An element of mem_spd is only used when the corresponding bit in mem_probe_spd is 0. This is where an OEM provides SPD data to the boot code for soldered-down memory.

Note that Efficeon only uses the first 64 bytes of the SPD data, so mem_spd only has room for those 64 bytes, which include the checksum for those 64 bytes. If an OEM only has true DIMM slots, these fields need not be programmed. This is the case for Transmeta's current reference design boards.

The recommendation for setting SPD data for soldered-down memory is to obtain a SoDIMM that has the same chips and configuration as the soldered-down memory, read the first 64 bytes of the SPD data out of the DIMM, and then program those bytes into the appropriate slot in the OEM config table. Otherwise, Transmeta maintains a set of canonical SPD data files that can be used, but it is best to use the manufacturer's data.

The default for this fields is all zeros since they are unused with Transmeta's current reference design boards.

3.3 Optional Memory Configuration

The default memory configuration parameters are designed to work for all OEMs. However, in case of problems, this section describes parameters that can be adjusted, with Transmeta's help, to tune the boot code to an OEM's board. *DO NOT change any of these parameters without help from Transmeta engineers.*

- cpu_feature (also see *cpu_feature* on page 27)
- mem_freq_min (also see *mem_freq_min* on page 30)
- mem_freq_max (also see *mem_freq_max* on page 30)
- sclkdly_to_mem_frequency[4] (also see *sclkdly_to_mem_frequency[4]* on page 32)
- group_to_min_loads[4] and group_to_max_loads[4] (also see *group_to_min_loads[4]* on page 32 and *group_to_max_loads[4]* on page 33)

cpu_feature

cpu_feature contains two bits that affect memory configuration that OEMs may want to use.

- Bit disable_ecc means that ECC should not be used even if all the DIMM slots support ECC. This is used in Transmeta's current reference design boards by default because of the performance problems associated with using ECC. The default for OEM boards is disabled (1).

- Bit SSTL2_termination means that the board has SSTL_2 termination for the signals (command, address, data, and strobes). This bit should not be necessary since the boot code should configure memory automatically either way, but it may be necessary in some boards of hardware revision 1.1. We recommend that customers not use SSTL_2 termination on rev. 1.1 hardware, so the default is off (0).

mem_freq_min

mem_freq_min specifies a frequency below which the boot code does *not* configure memory. The frequency is specified in multiples of 16.67 MHz. and limits the command frequency of the DDR bus (e.g. 133 MHz for DDR266 = PC2100, 200 MHz for DDR 400 = PC3200).

The boot code tries to configure memory at the highest rated speed (unless clamped by mem_freq_max or some characterization fields). If configuration fails at that speed, the boot code decreases the speed of the memory in 16.67 MHz or 33.33 MHz steps until it finds a working configuration. If at some point it tries to go below mem_freq_min, it fails.

The default is 5, encoding 83.3 MHz, which is the minimum speed according to the JEDEC specs for DDR200, DDR266, and DDR333. DDR400 may have a minimum speed of 100 MHz, which would be encoded as 6.

mem_freq_max

mem_freq_max specifies the maximum frequency to use for memory even if the installed memory supports higher frequencies. The frequency is specified in multiples of 16.67 MHz, and limits the command frequency of the DDR bus.

This allows an OEM, for example, to qualify a board with DDR266, and force the boot code to use 133 MHz frequency and not try higher speeds even if the memory supports it.

The default is 10, encoding 166.67 MHz (i.e. DDR333 = PC2700).

sclkdly_to_mem_frequency[4]

sclkdly_to_mem_frequency is an array of four values specifying a mapping between memory frequencies and the internal sclkdly parameter.

sclkdly is a parameter used to compensate for board roundtrip time. A board with long DDR traces would require larger sclkdly values than a board with short traces. The sclkdly value controls the phase difference between the internal NB (Northbridge) clock and the external DRAM clock that the DRAM chips see. The DRAM clock can lead the Northbridge clock by up to 3/8 of a cycle.

Loosely speaking, an sclkdly value of *N* means that the DRAM clock leads the NB clock by *N*/8 cycles. Thus with a setting of 0, there is no phase difference between both clocks, while with a setting of 3, there is a 3/8 cycle difference between them. The larger the phase difference, the more round trip delay that can be

accomodated. However, the phase difference can't be made too large because otherwise data may return 'too soon'. At higher frequencies, the same 1/8 fraction of a cycle becomes less absolute time, so even though the board round-trip time may remain a constant, the fraction needs to be increased as well.

The encoding of sclkdly_to_mem_frequencies is such that each element of the array encodes the highest memory command frequency value (in multiples of 16.67 MHz) that can use the value of the index of the element. The default, which works well with Transmeta's current reference design boards, is:

Field	Value	Description
sclkdly_to_mem_frequency[0]	7	116.67 MHz is max frequency for sclkdly 0
sclkdly_to_mem_frequency[1]	10	166.67 MHz is max frequency for sclkdly 1
sclkdly_to_mem_frequency[2]	255	oo is max frequency for sclkdly 2
sclkdly_to_mem_frequency[3]	255	oo means that sclkdly 3 is unused

For boards with less round-trip time (e.g. with soldered-down memory close to the CPU), it is best to use 0 for higher frequencies (e.g. DDR266 encoded as 8, or even DDR333 encoded as 10), and so on.

For boards with more round-trip time (e.g. 4 DIMM slots very far away), it would be best to use 0 for lower frequencies (e.g. DDR200 encoded as 6, or even DDR167 encoded as 5), and so on.

group_to_min_loads[4] and group_to_max_loads[4]

group_to_min_loads and group_to_max_loads are best described together. Each is an array of four values used to control the drive strength for the different signal groups in Efficeon processors.

Drive Strength Background

Drive strength must be reasonably well matched to the actual load, especially for boards lacking SSTL_2 termination. If the drive strength is too low, the signals rise and fall slowly and reduced frequency of operation may result. If the drive strength is too high, ringing and coupling can occur, and this also reduces the effective frequency of operation.

The Efficeon controller has programmable drive strength, where the configuration software (i.e. the boot code) can enable more or fewer drivers for each of four signal groups. The drive strength for each signal group is affected by the entries in these two arrays with the corresponding index.

- Group 0 is the clock signals (CLK, CLK#). The boot code assumes that there are always four effetive loads for this signal group, as the JEDEC SoDIMM specs require the addition of capacitors to make all clock pairs be similarly loaded.

- Group 1 is the per-rank commands signals (CS#, CKE). The boot code computes the loads per rank given the configuration of the DIMMs and assumes that all chips present the same load, so it determines drive strength given the rank with the largest number of loads (e.g. if one DIMM has 4 x16 parts and another has 8 x8 parts, it assumes 8 chip loads).

- Group 2 is the shared command and address signals (RAS#, CAS#, WE#, BA<1:0>, and address). These signals are typically the most loaded as they go to every DRAM chip in every DIMM and rank, unless registered DIMMs are in use. The boot code sums the total number of DRAM chips given the actual configuration, and considers that the logical chip load for this signal group.

- Group 3 is the strobe and data signals (DQS and DQ). Each of these signals is loaded once per rank of memory, but each chip load is typically twice as large as the loads on the other signals.

Specifying Drive Strength

In the controller, drive strength is specified, per signal group, as a value between 0 and 7, both inclusive. Each of these values enables some progressive number of drivers for that signal group. The actual numbers of drive transistors used are:

```
4,                // for drive strength = 0
5,
6,
7,
8,
10,
12,
18,               // for drive strength = 7
```

group_to_min_loads specifies, per signal group, the minimum logical chip loads that can be driven when the strength parameter is set to 0. Similarly, group_to_max_loads specifies, per signal group, the maximum logical chip loads that can be driven when the strength parameter is set to 7. The boot code linearly interpolates between these sets of values, given the number of logical chip loads in each signal group.

Note that when lightly loaded (few logical chip loads), board capacitance may matter more than logical chip loads, while when heavily loaded (large number of chip loads), the logical chip loads is the dominant effect. Board capacitance partly determines drive strength. As different board layouts, routing, and materials can affect drive strength, the parameters are tunable by OEMs, with Transmeta's help.

The defaults are all-zeros for min_loads, meaning that at their lowest setting we can drive no chip loads, and { 18, 18, 18, 9 } for max_loads, meaning that at the highest setting, we can drive 18 logical loads for most signal groups, and only 9 logical loads for the DQ/DQS signal group. To allow for increased drive strength at the low end, these arrays contain signed values, so negative values can be used causing the drive strength to become higher.

Here are some examples of how the min, max, and logical chip loads results in a setting of drive strength:

	For min=4, max-18:			For min = 0, max = 18:	
N loads	**N drivers**	**Drive strength**		**N drivers**	**Drive strength**
1	1 -> 4	0		5	1
2	2 -> 4	0		6	2
3	3 -> 4	0		7	3
4	4	0		8	4
5	5	1		8	4
6	6	2		9 -> 10	5
7	7	3		10	5
8	8	4		11 -> 12	6
9	9 -> 10	5		11 -> 12	6
10	10	5		12	6
11	11	6		-> 18	7
12	12	6			
13+	->18	7			

Note that for group 2 (shared command and address signals), the controller uses 'double command mode' when the number of logical chip loads exceeds the value in group_to_max_loads[2].

Chapter 4

OEM Configuration Table

The OEM configuration table provides an interface for manufacturers to edit various Code Morphing™ software settings, notably LongRun voltage/frequency points but also VRDA information and (optionally) memory settings to override SPD information. This table also provides boot code settings that were controlled by a mode bit ROM in previous Transmeta processors.

Manufacturers create a configuration table using a SKU provided by Transmeta as well as the manufacturer's decisions about various aspects of processor configuration. The table is encoded using a utility called ROM Compiler, and then uploaded with the BIOS to a ROM on the motherboard. Code Morphing software is the first software to run on the processor—it sizes memory, sets voltage and frequency points, and configures the processor prior to loading the BIOS and operating system.

OEM configuration table fields are categorized in this chapter as follows:

- *OEM-Managed Fields* on page 26
- *Transmeta SKU Fields* on page 34
- *Read-Only Fields* on page 24

Each section provides a summary of all fields described in that section. All sizes shown are in bytes.

Field Access

Each field in this configuration table is listed as one of the following:

RO	Read-only. Do not change these fields.
RW	Read-write. Change these fields to suit your application.
TO	Transmeta-recommended values only. Contact your Transmeta representative to obtain the proper settings for these fields.

Note
This configuration table is a work in progress, and many details are to be determined. This table described here is not to be used with any Transmeta processor without assistance from a Transmeta representative.

4.1 Read-Only Fields

The fields described in this section are read-only parameters that define the size and scope of Code Morphing software. They are administrative in nature and do not need to be changed by manufacturers.

Table 1: Header Fields

Field Name	Table Offset	Width (bytes)	Access	Default Value	Page
header	0x0000	16	RO	`"OEM Config Table"`	24
cpu_type	0x0010	1	RO	0x41	24
format_rev_major	0x0011	1	RO	0x04	24
table_size	0x0012	2	RO	0x0C7C	25
checksum	0x0014	4	RO	*checksum*	25
format_rev_minor	0x0018	4	RO	0x00000001	25
upgrade_compatibility_version	0x001C	4	RO	0x00000000	25
timestamp	0x0020	4	RO	0x00000000	25

header

Field Name	Table Offset	Width (bytes)	Access	Default Value
header	0x0000	16	RO	`"OEM Config Table"`

String of characters demarcating the beginning of the OEM configuration table.

cpu_type

Field Name	Table Offset	Width (bytes)	Access	Default Value
cpu_type	0x0010	1	RO	0x41

Processor type designation.

format_rev_major

Field Name	Table Offset	Width (bytes)	Access	Default Value
format_rev_major	0x0011	1	RO	0x04

Primary format revision.

table_size

Field Name	Table Offset	Width (bytes)	Access	Default Value
table_size	0x0012	2	RO	0x0C7C

Size of the defined part of the OEM configuration table.

checksum

Field Name	Table Offset	Width (bytes)	Access	Default Value
checksum	0x0014	4	RO	*checksum*

Negated checksum of the entire table (total checksum = 0). This field should not be changed by hand.

format_rev_minor

Field Name	Table Offset	Width (bytes)	Access	Default Value
format_rev_minor	0x0018	4	RO	0x00000001

Minor table revision number.

upgrade_compatibility_version

Field Name	Table Offset	Width (bytes)	Access	Default Value
upgrade_compatibility_version	0x001C	4	RO	0x00000000

This bitfield identifies the potential compatibility of the running Code Morphing software version to possible upgrades. Each bit corresponds to one compatibility issue, to be defined in a future revision.

timestamp

Field Name	Table Offset	Width (bytes)	Access	Default Value
timestamp	0x0020	4	RO	0x00000000

Timestamp generated by Transmeta.

4.2 OEM-Managed Fields

The fields in this section can be changed by manufacturers to suit specific applications. See the individual fields for descriptions of their operation.

Table 2: OEM-Managed Fields

Field Name	Table Offset	Width (bytes)	Access	Default Value	Pg
vr_100mV_ramp_time	0x0100	2	RW	0x0023	26
vr_voltage[32]	0x0104	2 (64 total)	RW	*See description*	27
cpu_feature	0x0144	4	RW	0x00000002	27
cms_memory_size	0x0148	1	RW	0x20	28
mem_probe_spd	0x0149	1	RW	0x03	28
mem_smbus_spd_base_addr	0x014a	1	RW	0x50	28
mem_slot_to_clocks[4]	0x014c	1 (4 total)	RW	*See description*	28
mem_spd[4]	0x0150	64 (256 total)	RW	All zeros	29
io_port_debug_led	0x0254	2	RW	0x0000	29
mem_freq_min	0x0256	1	RW	0x05	30
mem_freq_max	0x0257	1	RW	0x0A	30
rom_size_total	0x0258	2	RW	0x0020	30
rom_size_bios	0x025a	2	RW	0x000B	30
cms_main_start_block	0x025c	1	RW	0x01	30
cms_main_num_blocks	0x025d	1	RW	0x0A	31
cms_recovery_start_block	0x025e	1	RW	0x0B	31
cms_recovery_num_blocks	0x025f	1	RW	0x0A	31
upgrade_oem_id0	0x0260	4	TO	*Contact TMTA*	31
upgrade_oem_id1	0x0264	4	RW	0x00000000	31
upgrade_options	0x0268	4	RW	0x00000000	32
upgrade_virtual_rom_model	0x026c	4	RW	0x00000101	32
sclkdly_to_mem_frequency[4]	0x0270	1 (4 total)	RW	*See description*	32
group_to_min_loads[4]	0x0274	1 (4 total)	RW	*See description*	32
group_to_max_loads[4]	0x0278	1 (4 total)	RW	*See description*	33
longrun_frequencies[8]	0x027C	2 (16 total)	RW	*See description*	33

vr_100mV_ramp_time

Field Name	Table Offset	Width (bytes)	Access	Default Value
vr_100mV_ramp_time	0x0100	2	RW	0x0023

Time in microseconds required to ramp voltage up or down by 100mV.

vr_voltage[32]

Field Name	Table Offset	Width (bytes)	Access	Default Value
vr_voltage[32]	0x0104	2 (64 total)	RW	*See description*

An array of 32 fields which stores the voltage (in millivolts) associated with the corresponding VRDA value. The default values (in decimal) are as follows:

Field	Default Value
vr_voltage[0]	1750
vr_voltage[1]	1700
vr_voltage[2]	1650
vr_voltage[3]	1600
vr_voltage[4]	1550
vr_voltage[5]	1500
vr_voltage[6]	1450
vr_voltage[7]	1400
vr_voltage[8]	1350
vr_voltage[9]	1300
vr_voltage[10]	1024
vr_voltage[11]	1200
vr_voltage[12]	1150
vr_voltage[13]	1100
vr_voltage[14]	1050
vr_voltage[15]	1000

Field	Default Value
vr_voltage[16]	975
vr_voltage[17]	950
vr_voltage[18]	925
vr_voltage[19]	900
vr_voltage[20]	875
vr_voltage[21]	850
vr_voltage[22]	825
vr_voltage[23]	800
vr_voltage[24]	775
vr_voltage[25]	750
vr_voltage[26]	725
vr_voltage[27]	700
vr_voltage[28]	675
vr_voltage[29]	650
vr_voltage[30]	625
vr_voltage[31]	600

Also see *longrun_frequencies[8]* on page 33 and *longrun_manifold[8]* on page 39.

cpu_feature

Field Name	Table Offset	Width (bytes)	Access	Default Value
cpu_feature	0x0144	4	RW	0x00000002

CPU feature control bits.

Bits	Name	Function Description	Default
31:3		Reserved	0
2	SSTL2_termination	**SSTL2 Termination**—If set, SSTL_2 termination is in use for the DRAM bus.	0
1	disable_ecc	**Disable ECC**—If set, treat ECC memory as non-ECC memory.	1
0	psn_disable	**PSN Disable**—Permanently disable the processor serial number (PSN). Default is 0 (enable PSN). This feature is normally controlled via the Virtual Northbridge (VNB); for details see the *BIOS Programmer's Guide*.	0

cms_memory_size

Field Name	Table Offset	Width (bytes)	Access	Default Value
cms_memory_size	0x0148	1	RW	0x20

Size of memory (in MB) reserved for CMS usage.

mem_probe_spd

Field Name	Table Offset	Width (bytes)	Access	Default Value
mem_probe_spd	0x0149	1	RW	0x03

Any bank that has mem_slot_to_clocks (see below) clear is ignored. Otherwise, for any bank which has a set bit in this field, the memory configuration code attempts to probe for an SPD ROM.

When the appropriate bit is 1, if an SPD is found, it is used. Otherwise the slot is assumed to be empty. The only way to use SPD data from the OEM config table is to have the relevant mem_probe_spd field set to 0.

mem_smbus_spd_base_addr

Field Name	Table Offset	Width (bytes)	Access	Default Value
mem_smbus_spd_base_addr	0x014a	1	RW	0x50

Address of first SPD ROM on the SMBUS.

mem_slot_to_clocks[4]

Field Name	Table Offset	Width (bytes)	Access	Default Value
mem_slot_to_clocks[4]	0x014c	1 (4 total)	RW	*See description*

This table of 4 fields maps DDR DIMM slot numbers to the clocks needed for that DIMM.

Field	Default Value
mem_slot_to_clocks[0]	0x07
mem_slot_to_clocks[1]	0x38
mem_slot_to_clocks[2]	0x00
mem_slot_to_clocks[3]	0x00

mem_spd[4]

Field Name	Table Offset	Width (bytes)	Access	Default Value
mem_spd[4]	0x0150	64 (256 total)	RW	All zeros

An array of SPD data for each DDR DIMM (soldered down or override). One mem_spd field exists for each DDR memory bank. SDR memory is not supported. Default values are zero for all sub-fields.

The structure for each field is shown below. SPD fields contained in each mem_spd field correspond directly with those described in the SPD specification. See the specification for more information on individual fields. Note that only the first 64 bytes of SPD data are used.

Offset in Structure	Size (bytes)	SPD Field
0 0x0000	1	written_bytes
1 0x0001	1	total_bytes
2 0x0002	1	memory_type
3 0x0003	1	row_addrs
4 0x0004	1	col_addrs
5 0x0005	1	phys_banks
6 0x0006	2	dwidth
8 0x0008	1	vlvl
9 0x0009	1	tckmin
10 0x000a	1	tac
11 0x000b	1	dimmconf
12 0x000c	1	ref
13 0x000d	1	width
14 0x000e	1	eccwidth
15 0x000f	1	mindelay
16 0x0010	1	blength
17 0x0011	1	nbanks
18 0x0012	1	cl
19 0x0013	1	csl
20 0x0014	1	wl
21 0x0015	1	modflags
22 0x0016	1	genflags

Offset in Structure	Size (bytes)	SPD Field
23 0x0017	1	tckmin1
24 0x0018	1	tac1
25 0x0019	1	tckmin2
26 0x001a	1	tac2
27 0x001b	1	trp
28 0x001c	1	trrd
29 0x001d	1	trcd
30 0x001e	1	tras
31 0x001f	1	density
32 0x0020	1	tas
33 0x0021	1	tah
34 0x0022	1	tds
35 0x0023	1	tdh
36 0x0024	1	vcsdram[5]
41 0x0029	1	trc
42 0x002a	1	trfc
43 0x002b	1	tckmax
44 0x002c	1	tdqsqmax
45 0x002d	1	tqhs
46 0x002e	1	tpll
62 0x003e	1	rev
63 0x003f	1	checksum

io_port_debug_led

Field Name	Table Offset	Width (bytes)	Access	Default Value
io_port_debug_led	0x0254	2	RW	0x0000

I/O port address that forwards debug LED codes (0 to disable).

mem_freq_min

Field Name	Table Offset	Width (bytes)	Access	Default Value
mem_freq_min	0x0256	1	RW	5

Lower limit of memory frequency in 100/6 MHz increments.

mem_freq_max

Field Name	Table Offset	Width (bytes)	Access	Default Value
mem_freq_max	0x0257	1	RW	10

Upper limit of memory frequency in 100/6 MHz increments.

rom_size_total

Field Name	Table Offset	Width (bytes)	Access	Default Value
rom_size_total	0x0258	2	RW	0x0020

Total size of ROM for partitioning (in units of 64K).

rom_size_bios

Field Name	Table Offset	Width (bytes)	Access	Default Value
rom_size_bios	0x025a	2	RW	0x000B

x86 BIOS size for partitioning (in units of 64K).

cms_main_start_block

Field Name	Table Offset	Width (bytes)	Access	Default Value
cms_main_start_block	0x025c	1	RW	0x01

This field encodes how Code Morphing software is stored in the ROM. It refers to the first contiguous 64K block of Code Morphing software.

cms_main_num_blocks

Field Name	Table Offset	Width (bytes)	Access	Default Value
cms_main_num_blocks	0x025d	1	RW	0x0A

This field encodes how Code Morphing software is stored in the ROM. It refers to the total number of contiguous 64K blocks of Code Morphing software.

cms_recovery_start_block

Field Name	Table Offset	Width (bytes)	Access	Default Value
cms_recovery_start_block	0x025e	1	RW	0x0B

This field encodes how recovery Code Morphing software is stored in the ROM. It refers to the first contiguous 64K block of recovery Code Morphing software.

cms_recovery_num_blocks

Field Name	Table Offset	Width (bytes)	Access	Default Value
cms_recovery_num_blocks	0x025f	1	RW	0x0A

This encodes how recovery Code Morphing software is stored in the ROM. It refers to the total number of contiguous 64K blocks of recovery Code Morphing software.

upgrade_oem_id0

Field Name	Table Offset	Width (bytes)	Access	Default Value
upgrade_oem_id0	0x0260	4	TO	*Contact TMTA*

A description of this field is reserved for a future edition of this document. Note that this field is NOT USABLE as of this writing—OEMs must use a value provided by Transmeta.

upgrade_oem_id1

Field Name	Table Offset	Width (bytes)	Access	Default Value
upgrade_oem_id1	0x0264	4	RW	0x00000000

A description of this field is reserved for a future edition of this document

upgrade_options

Field Name	Table Offset	Width (bytes)	Access	Default Value
upgrade_options	0x0268	4	RW	0x00000000

A description of this field is reserved for a future edition of this document

upgrade_virtual_rom_model

Field Name	Table Offset	Width (bytes)	Access	Default Value
upgrade_virtual_rom_model	0x026c	4	RW	0x00000101

Virtual ROM model description. A description of this field is reserved for a future edition of this document.

sclkdly_to_mem_frequency[4]

Field Name	Table Offset	Width (bytes)	Access	Default Value
sclkdly_to_mem_frequency[4]	0x0270	1 (4 total)	RW	*See description*

Table mapping sclkdly values to maximum memory frequencies (in increments of 16.67 MHz).

Field	Default Value
sclkdly_to_mem_frequency[0]	7
sclkdly_to_mem_frequency[1]	10
sclkdly_to_mem_frequency[2]	255
sclkdly_to_mem_frequency[3]	255

group_to_min_loads[4]

Field Name	Table Offset	Width (bytes)	Access	Default Value
group_to_min_loads[4]	0x0274	1 (4 total)	RW	*See description*

Table specifying how many chip loads in each signal group can be driven with drive strength set to 0 (min). It depends strongly on board capacitance.

Field	Default Value
group_to_min_loads[0]	0
group_to_min_loads[1]	0

Field	Default Value
group_to_min_loads[2]	0
group_to_min_loads[3]	-1

group_to_max_loads[4]

Field Name	Table Offset	Width (bytes)	Access	Default Value
group_to_max_loads[4]	0x0278	1 (4 total)	RW	*See description*

Table specifying how many chip loads in each signal group can be driven with drive strength set to 7 (max). It depends weakly on board capacitance.

Field	Default Value
group_to_max_loads[0]	18
group_to_max_loads[1]	18
group_to_max_loads[2]	18
group_to_max_loads[3]	9

longrun_frequencies[8]

Field Name	Table Offset	Width (bytes)	Access	Default Value
longrun_frequencies[8]	0x027C	2 (16 total)	RW	*See description*

Table specifying LongRun™ frequencies. Each entry is a frequency in MHz, or 0 for an 'empty' entry. The non-empty entries must be contiguous at the start of the array. There is space for 8 entries in the array. Empty entries are ignored. Frequencies must be in ascending order.

To set a LongRun point, place a frequency entry in this table. Corresponding voltages are computed by Code Morphing software. Pre-set defaults are shown below.

Also see *vr_voltage[32]* on page 27, *longrun_manifold[8]* on page 39, and *longrun_min_frequency* on page 40. To set this field, contact your Transmeta representative.

Field	Default Value
longrun_frequencies[0]	400
longrun_frequencies[1]	533
longrun_frequencies[2]	667
longrun_frequencies[3]	800

Field	Default Value
longrun_frequencies[4]	0
longrun_frequencies[5]	0
longrun_frequencies[6]	0
longrun_frequencies[7]	0

4.3 Transmeta SKU Fields

The fields summarized in the following table comprise a Transmeta SKU, which is a single tested package configuration that controls several aspects of Code Morphing software. Contact your Transmeta representative to obtain the proper settings for these fields.

Table 3: Transmeta SKU Fields

Field Name	Table Offset	Width (bytes)	Access	Default Value	Page
sku_flags	0x0800	4	TO	0x00000000	35
bpctrl	0x0804	4	TO	0x00000019	35
ifctrl	0x0808	4	TO	0x00000002	35
mode_da_low_v_range	0x080C	4	TO	0x04420F23	35
mode_da_high_v_range	0x0810	4	TO	0x04420F23	35
mode_da_low_v_range_max_v	0x0814	2	TO	1100	36
mode_da_high_v_range_min_v	0x0816	2	TO	900	36
sc_laconfig	0x0818	2	TO	0x0aa3	36
sc_ltconfig	0x081A	2	TO	0x00a3	36
mode_tlb	0x081C	1	TO	0x02	36
mode_wq	0x081D	1	TO	0x02	36
pm_tag_mode	0x081E	1	TO	0x05	37
pm_data_mode	0x081F	1	TO	0x05	37
mc_pll_mode	0x0820	2	TO	0x6400	37
mc_core_clkdiv	0x0822	2	TO	0x0090	37
mc_misc_mode	0x0824	2	TO	0x6985	37
mc_elroy_clkdiv	0x0826	2	TO	0x0604	37
longrun_pll_relock	0x0828	1	TO	0x14	38
core_voltage	0x0829	1	TO	0x0E	38
rd_dqsdly_temp_adjust	0x082A	1	TO	0x52	38
wr_dqdly_temp_adjust	0x082B	1	TO	0x00	38
longrun_manifold[8]	0x082C	6 (48 total)	TO	*See description*	39
longrun_min_frequency	0x085C	2	TO	0x012c	40
cspec_lim	0x085E	1	TO	0x1b	40
vc_full	0x085F	1	TO	0x1e	40
vc_priority	0x0860	1	TO	0x15	40
vc_threshold	0x0861	1	TO	0x0e	40
dc_enable	0x0862	1	TO	0x03	41
it_ena	0x0863	1	TO	0x07	41
pcctl	0x0864	2	TO	0x03ef	41
sc_cs	0x0866	2	TO	0x02ff	41
ap_esr_io_dly_ctl	0x0868	4	TO	0x1c30300	41
volt_max	0x086C	2	TO	1500	42
volt_min	0x086E	2	TO	750	42
volt_cons_mult	0x0870	1	TO	0	42

sku_flags

Field Name	Table Offset	Width (bytes)	Access	Default Value
sku_flags	0x0800	4	TO	0x00000000

Flags to set SKU-specific features, as described below. Please set this field to a value designated by Transmeta.

Bits	Name	Description	Default
31:3		Reserved	0
2	enable_AGP_fast_write	Enable AGP fast writes [1]	0
1	enable_AGP_4x	Enable AGP 4x [1]	0
0	enable_elroy_pll_divby	Allow Northbridge to work at odd multiples of 16.67 MHz	0

1. Note that BIOS can still disable this feature.

bpctrl

Field Name	Table Offset	Width (bytes)	Access	Default Value
bpctrl	0x0804	4	TO	0x00000019

Transmeta defined field. Please set this field to a value designated by Transmeta.

ifctrl

Field Name	Table Offset	Width (bytes)	Access	Default Value
ifctrl	0x0808	4	TO	0x00000002

Transmeta defined field. Please set this field to a value designated by Transmeta.

mode_da_low_v_range

Field Name	Table Offset	Width (bytes)	Access	Default Value
mode_da_low_v_range	0x080C	4	TO	0x04420F23

Transmeta defined field. Please set this field to a value designated by Transmeta.

mode_da_high_v_range

Field Name	Table Offset	Width (bytes)	Access	Default Value
mode_da_high_v_range	0x0810	4	TO	0x04420F23

Transmeta defined field. Please set this field to a value designated by Transmeta.

mode_da_low_v_range_max_v

Field Name	Table Offset	Width (bytes)	Access	Default Value
mode_da_low_v_range_max_v	0x0814	2	TO	1100

Transmeta defined field. Please set this field to a value designated by Transmeta.

mode_da_high_v_range_min_v

Field Name	Table Offset	Width (bytes)	Access	Default Value
mode_da_high_v_range_min_v	0x0816	2	TO	900

Transmeta defined field. Please set this field to a value designated by Transmeta.

sc_laconfig

Field Name	Table Offset	Width (bytes)	Access	Default Value
sc_laconfig	0x0818	2	TO	0x0aa3

Transmeta defined field. Please set this field to a value designated by Transmeta.

sc_ltconfig

Field Name	Table Offset	Width (bytes)	Access	Default Value
sc_ltconfig	0x081A	2	TO	0x00a3

Transmeta defined field. Please set this field to a value designated by Transmeta.

mode_tlb

Field Name	Table Offset	Width (bytes)	Access	Default Value
mode_tlb	0x081C	1	TO	0x02

Transmeta defined field. Please set this field to a value designated by Transmeta.

mode_wq

Field Name	Table Offset	Width (bytes)	Access	Default Value
mode_wq	0x081D	1	TO	0x02

Transmeta defined field. Please set this field to a value designated by Transmeta.

pm_tag_mode

Field Name	Table Offset	Width (bytes)	Access	Default Value
pm_tag_mode	0x081E	1	TO	0x05

Transmeta defined field. Please set this field to a value designated by Transmeta.

pm_data_mode

Field Name	Table Offset	Width (bytes)	Access	Default Value
pm_data_mode	0x081F	1	TO	0x05

Transmeta defined field. Please set this field to a value designated by Transmeta.

mc_pll_mode

Field Name	Table Offset	Width (bytes)	Access	Default Value
mc_pll_mode	0x0820	2	TO	0x6400

Transmeta defined field. Please set this field to a value designated by Transmeta.

mc_core_clkdiv

Field Name	Table Offset	Width (bytes)	Access	Default Value
mc_core_clkdiv	0x0822	2	TO	0x0090

Transmeta defined field. Please set this field to a value designated by Transmeta.

mc_misc_mode

Field Name	Table Offset	Width (bytes)	Access	Default Value
mc_misc_mode	0x0824	2	TO	0x6985

Transmeta defined field. Please set this field to a value designated by Transmeta.

mc_elroy_clkdiv

Field Name	Table Offset	Width (bytes)	Access	Default Value
mc_elroy_clkdiv	0x0826	2	TO	0x0604

Transmeta defined field. Please set this field to a value designated by Transmeta.

longrun_pll_relock

Field Name	Table Offset	Width (bytes)	Access	Default Value
longrun_pll_relock	0x0828	1	TO	0x14

Value (in tenths of a microsecond) required for PLL relock during LongRun frequency change. Please set this field to a value designated by Transmeta.

core_voltage

Field Name	Table Offset	Width (bytes)	Access	Default Value
core_voltage	0x0829	1	TO	0x0E

Force core voltage used by boot code after reset, shown as VRDA code rather than a hardcoded mV value (see *vr_voltage[32]* on page 27). Must be no higher than 50 mV higher than the lowest LongRun voltage, and must support 533 MHz core operation and 167 MHz Northbridge operation. Please set this field to a value designated by Transmeta.

rd_dqsdly_temp_adjust

Field Name	Table Offset	Width (bytes)	Access	Default Value
rd_dqsdly_temp_adjust	0x082A	1	TO	0x52

Number of taps lost/gained per 50°C increment at Vmin. Two nibbles, the high nibble is taps lost at the high end, the low nibble is taps gained at the low end. Please set this field to a value designated by Transmeta.

wr_dqdly_temp_adjust

Field Name	Table Offset	Width (bytes)	Access	Default Value
wr_dqdly_temp_adjust	0x082B	1	TO	0x00

Number of taps lost/gained per 50°C increment at Vmin. Two nibbles, the high nibble is taps lost at the high end, the low nibble is taps gained at the low end. Please set this field to a value designated by Transmeta.

longrun_manifold[8]

Field Name	Table Offset	Width (bytes)	Access	Default Value
longrun_manifold[8]	0x082C	6 (48 total)	TO	*See description*

This array of 8 fields contains a SKU which describes all supported LongRun voltage and frequency points. Not all 8 points are required; zeroes can be filled for unused points. The structure for each field is as follows:

Offset in Structure	Width (bytes)	Value	Units	Function Description
0x0000	2	frequency	MHz	Frequency for this LongRun point
0x0002	2	voltage	mV	Voltage for this LongRun point
0x0004	2	temp	°C	Allowable temperature for this point

It is important that non-empty entries have zero as their voltage (meaning that Code Morphing software computes the voltage), or all of them have non-zero values. All non-empty entries must be contiguous at the start of the array. There is space for 8 entries in the array.

There must be at least 3 non-empty entries in the longrun manifold for any system implementing LongRun Advanced Thermal Management:

- A top MHz entry at 80 (or 70) degrees C.
- Two different entries at 100 degrees C, one with high MHz (but presumably lower than the 80 degree entry), and one with low MHz, at or above 533 MHz.

For systems not implementing LongRun Advanced Thermal Management, at least two non-empty entries are necessary, namely the two entries at 100 degrees C described above. For more information about LongRun Advanced Thermal Management, see the documentation for the LR_ATM register in the Transmeta Virtual Northbridge (VNB) in the *Efficeon TM8x00 BIOS Programmer's Guide*.

NOTE: To fix the frequency of operation, use the *longrun_frequencies[8]* table on page 33, not the manifold.

Default values for this field are as follows.

Field (frequency)	Default
longrun_point[0].frequency	400
longrun_point[1].frequency	533
longrun_point[2].frequency	667
longrun_point[3].frequency	800
longrun_point[4].frequency	0
longrun_point[5].frequency	0
longrun_point[6].frequency	0
longrun_point[7].frequency	0

Field (voltage)	Default
longrun_point[0].voltage	1000
longrun_point[1].voltage	1100
longrun_point[2].voltage	1200
longrun_point[3].voltage	1250
longrun_point[4].voltage	0
longrun_point[5].voltage	0
longrun_point[6].voltage	0
longrun_point[7].voltage	0

Field (temp)	Default
longrun_point[0].temp	100
longrun_point[1].temp	100
longrun_point[2].temp	100
longrun_point[3].temp	80
longrun_point[4].temp	0
longrun_point[5].temp	0
longrun_point[6].temp	0
longrun_point[7].temp	0

Also see *vr_voltage[32]* on page 27 and *longrun_frequencies[8]* on page 33.

Note that LongRun will not set a frequency lower than the value shown in *longrun_min_frequency*, below, nor will it set voltages out of the range shown in *volt_max* on page 42, *volt_min* on page 42, and *volt_cons_mult* on page 42.

Please set this field to a value designated by Transmeta.

longrun_min_frequency

Field Name	Table Offset	Width (bytes)	Access	Default Value
longrun_min_frequency	0x085C	2	TO	300

Minimum LongRun frequency, in MHz. Also see *volt_max* on page 42, *volt_min* on page 42, and *volt_cons_mult* on page 42, as well as *longrun_manifold[8]* above.

Please set this field to a value designated by Transmeta.

cspec_lim

Field Name	Table Offset	Width (bytes)	Access	Default Value
cspec_lim	0x085E	1	TO	0x1b

Cached speculation limit modebits register. Please set this field to a value designated by Transmeta.

vc_full

Field Name	Table Offset	Width (bytes)	Access	Default Value
vc_full	0x085F	1	TO	0x1e

Victim Cache Full indicator threshold modebits register. Please set this field to a value designated by Transmeta.

vc_priority

Field Name	Table Offset	Width (bytes)	Access	Default Value
vc_priority	0x0860	1	TO	0x15

Victim Cache drain Priority threshold modebits register. Please set this field to a value designated by Transmeta.

vc_threshold

Field Name	Table Offset	Width (bytes)	Access	Default Value
vc_threshold	0x0861	1	TO	0x0e

Victim Cache drain Theshold modebits register. Please set this field to a value designated by Transmeta.

dc_enable

Field Name	Table Offset	Width (bytes)	Access	Default Value
dc_enable	0x0862	1	TO	0x03

Data Cache Enable modebits register. Please set this field to a value designated by Transmeta.

it_ena

Field Name	Table Offset	Width (bytes)	Access	Default Value
it_ena	0x0863	1	TO	0x07

I/O Transfer Enable modebits register. Please set this field to a value designated by Transmeta.

pcctl

Field Name	Table Offset	Width (bytes)	Access	Default Value
pcctl	0x0864	2	TO	0x03ef

PC-unit control modebits register. Please set this field to a value designated by Transmeta.

sc_cs

Field Name	Table Offset	Width (bytes)	Access	Default Value
sc_cs	0x0866	2	TO	0x02ff

Secondary Cache Control modebits register. Please set this field to a value designated by Transmeta.

ap_esr_io_dly_ctl

Field Name	Table Offset	Width (bytes)	Access	Default Value
ap_esr_io_dly_ctl	0x0868	4	TO	0x1c30300

AGP I/O buffer circuitry delay values. Please set this field to a value designated by Transmeta.

volt_max

Field Name	Table Offset	Width (bytes)	Access	Default Value
volt_max	0x086C	2	TO	1500

Maximum LongRun voltage, in mV. Also see *longrun_min_frequency* on page 40, *volt_min* on page 42, and *volt_cons_mult* on page 42, as well as *longrun_manifold[8]* on page 39. Please set this field to a value designated by Transmeta.

volt_min

Field Name	Table Offset	Width (bytes)	Access	Default Value
volt_min	0x086E	2	TO	750

Minimum LongRun voltage, in mV. Also see *longrun_min_frequency* on page 40, *volt_min* on page 42, and *volt_cons_mult* on page 42, as well as *longrun_manifold[8]* on page 39. Please set this field to a value designated by Transmeta.

volt_cons_mult

Field Name	Table Offset	Width (bytes)	Access	Default Value
volt_cons_mult	0x0870	1	TO	0

A scaling factor in mV, part of the computation to calculate voltage for a given LongRun point. Please set this field to a value designated by Transmeta.

Chapter 5

Code Morphing Software: Initial State

Table 4: OEM Configuration Table Initial State

Field Type	Field Name	Offset	Default Value	Pg
readonly	header	0x0000	"OEM Config Table"	18
readonly	cpu_type	0x0010	0x41	18
readonly	format_rev_major	0x0011	0x04	18
readonly	table_size	0x0012	0x0C7C	19
readonly	checksum	0x0014	0x0827b9c4	19
readonly	format_rev_minor	0x0018	0x00000001	19
readonly	upgrade_compatibility_version	0x001C	0x00000000	19
readonly	timestamp	0x0020	0x00000000	19
OEM	vr_100mV_ramp_time	0x0100	0x0023	20
OEM	vr_voltage[32]	0x0104	*See description*	21
OEM	cpu_feature	0x0144	0x00000002	21
OEM	cms_memory_size	0x0148	0x20	22
OEM	mem_probe_spd	0x0149	0x03	22
OEM	mem_smbus_spd_base_addr	0x014a	0x50	22
OEM	mem_slot_to_clocks[4]	0x014c	*See description*	22
OEM	mem_spd[4]	0x0150	All zeros	23
OEM	io_port_debug_led	0x0254	0x0000	23
OEM	mem_freq_min	0x0256	0x05	24
OEM	mem_freq_max	0x0257	0x0A	24
OEM	rom_size_total	0x0258	0x0020	24
OEM	rom_size_bios	0x025a	0x000B	24
OEM	cms_main_start_block	0x025c	0x01	24
OEM	cms_main_num_blocks	0x025d	0x0A	25
OEM	cms_recovery_start_block	0x025e	0x0B	25
OEM	cms_recovery_num_blocks	0x025f	0x0A	25
OEM	upgrade_oem_id0	0x0260	0x00000000	25
OEM	upgrade_oem_id1	0x0264	0x00000000	25

Table 4: OEM Configuration Table Initial State (Continued)

Field Type	Field Name	Offset	Default Value	Pg
OEM	upgrade_options	0x0268	0x00000000	26
OEM	upgrade_virtual_rom_model	0x026c	0x00000101	26
OEM	sclkdly_to_mem_frequency[4]	0x0270	*See description*	26
OEM	group_to_min_loads[4]	0x0274	*See description*	27
OEM	group_to_max_loads[4]	0x0278	*See description*	27
OEM	longrun_frequencies[8]	0x027C	*See description*	27
SKU	sku_flags	0x0800	0x00000000	29
SKU	bpctrl	0x0804	0x00000019	29
SKU	ifctrl	0x0808	0x00000002	29
SKU	mode_da_low_v_range	0x080C		29
SKU	mode_da_high_v_range	0x0810		29
SKU	mode_da_low_v_range_max_v	0x0814	1100	30
SKU	mode_da_high_v_range_min_v	0x0816	900	30
SKU	sc_laconfig	0x0818	0x0aa3	30
SKU	sc_ltconfig	0x081A	0x00a3	30
SKU	mode_tlb	0x081C	0x02	30
SKU	mode_wq	0x081D	0x02	30
SKU	pm_tag_mode	0x081E	0x05	31
SKU	pm_data_mode	0x081F	0x05	31
SKU	mc_pll_mode	0x0820	0x6400	31
SKU	mc_core_clkdiv	0x0822	0x0090	31
SKU	mc_misc_mode	0x0824	0x6985	31
SKU	mc_elroy_clkdiv	0x0826	0x0604	31
SKU	longrun_pll_relock	0x0828	0x14	32
SKU	core_voltage	0x0829	0x0E	32
SKU	rd_dqsdly_temp_adjust	0x082A	0x52	32
SKU	wr_dqdly_temp_adjust	0x082B	0x00	32
SKU	longrun_manifold[8]	0x082C	*See description*	32
SKU	longrun_min_frequency	0x085C	0x012c	33
SKU	cspec_lim	0x085E	0x1b	33
SKU	vc_full	0x085F	0x1e	33
SKU	vc_priority	0x0860	0x15	33
SKU	vc_threshold	0x0861	0x0e	34
SKU	dc_enable	0x0862	0x03	34
SKU	it_ena	0x0863	0x07	34
SKU	pcctl	0x0864	0x03ef	34
SKU	sc_cs	0x0866	0x02ff	34
SKU	ap_esr_io_dly_ctl	0x0868	0x1c30300	34
SKU	volt_max	0x086C	1500	35
SKU	volt_min	0x086E	750	35
SKU	volt_cons_mult	0x0870	0	35

Chapter 6

Code Morphing Software: Examples

TBD

Chapter 7

Code Morphing Software: Checklist

TBD

Appendix A

POST Codes

This section describes the Code Morphing software POST and error codes for Transmeta Efficeon™ TM8x00 processors.

A **POST code** is a “progress marker”, and if displayed indicates the executed code stream has successfully moved past that point. Code Morphing software boot code typically issues a POST code prior to some new functional section of code being executed, and if the next POST code never appears, the last POST code issued indicates the last known working section of code executed before a problem arose.

An **error code** is a status descriptor giving some information about a problem encountered. Code Morphing software typically issues an error code after doing something and finding a problem with it.

POST and error codes are two bytes in length. *xx* in the second byte position indicates a variable value, that for most codes is 00. The table below lists the Efficeon TM8x00 Code Morphing software (CMS) POST and error codes.

Table 5: Efficeon TM8000 POST Codes in Receipt Order

Code	Description
04 00	Hello world, posts within 10 instructions of boot
58 20	Checksums match, caches initialized, late boot code decompressed and running—before memory configuration
9x xx	Error computing chip capacity, where xxx = calculated size
a0 xx	Error in configuring and sizing memory (xx = nonzero)
68 02	Finished reading SPD data
69 xx	Memory configured, xx = result (00 = success)
6a xx	Memory verified, xx = memory verification return val (00 = success)
68 03	Finished configuring memory (Note, sometimes this code doesn't show up because it is emitted in quick succession with 5830, too fast for the debugger to recognize)
58 30	Memory successfully configured
58 65	(if xboot present) before xboot init
58 80	Before CMS decompression
58 90	Main CMS successfully decompressed
58 91	Main CMS decompression failed, decompressed recovery CMS
AC ED	CMS decompressed, instructions begin
AC 04	Begin CMS nucleus initialization for regular cold boot

Table 5: Efficeon TM8000 POST Codes in Receipt Order (Continued)

Code	Description
AC 06	Begin CMS nucleus initialization for resume path
AC 14	CMS component initialization complete for cold boot
AC 16	CMS component initialization complete for resume path
AC 18	CMS x86 install state complete
AC 1C	CMS about to execute first x86 instruction

Appendix B

Recommended Reading

The following documents should be used in conjunction with this guide. Due to the preliminary nature of this document, not all reference documents may be available.

- *Efficeon™ Data Book*
- *Efficeon™ Code Morphing Software Guide*
- *Efficeon™ BIOS Programmer's Guide*
- *Efficeon™ Functional and Specification Errata*
- *Efficeon™ System Design Guide*
- *Efficeon TM8x00 Code Morphing Software Release Notes*

Glossary

A

ABI—Application Binary Interface. The detailed low-level conventions that applications use to communicate with an operating system or a language/environment library. An ABI is the low-level realization of an API. It consists of the register conventions, parameter passing and return conventions, invocation of the O/S conventions (e.g. through an exception, etc.), etc.

ACPI—Advanced Configuration and Power Interface. An operating-system-centric power management scheme based on a description of a set of hardware interfaces that the operating system can use to both change the active and suspend power states of the CPU and to determine the relative advantages and implement an appropriate policy. BIOS sets up the ACPI tables, and the OS uses the tables to enter and exit states and decide when it is profitable to do it.

AGP—Advanced Graphics Port. An interface added to modern northbridge chips used by graphics accelerators. It is similar to the PCI bus, but wider and running at a faster frequency. It is logically a PCI bus on its own right, but running at a higher frequency.

API—Application Programming Interface. The high-level (often C-language) interface that applications use to communicate with an operating system or a language/environment library. By extension, the high-level interface presented by any subsystem (e.g. a telephony API).

APIC—Advanced Programmable Interrupt Controller. An extension to the x86 architecture which allows finer granularity of control over interrupts and more interrupt sources without sharing an IRQ. The APIC architecture is composed of two components, the local APIC in each CPU, and the I/O APICs in the southbridge and elsewhere. The APICs communicate either by a dedicated protocol (dedicated bus in the P6 class of Intel CPUs) to arbitrate interrupt delivery.

atom—An individually-encoded operation in a VLIW instruction. A VLIW instruction, also called a molecule, consists of a collection of operations called atoms. Atoms are similar to the instructions in RISC architectures.

B

BIOS—Basic Input Output System. A collection of software utilities that virtualizes some of the platform for the benefit of the operating system. Typically it gains control after reset (power up), and after configuring some platform-specific hardware, loads the operating system into memory and transfer control to it. It remains resident as a set of subroutines to virtualize the keyboard, the mouse, the screen, the disk, etc. Modern operating systems bypass the BIOS for most services, and the BIOS has become almost exclusively a boot loader.

bridge—A device connected to two or more buses that forwards requests or transactions from one to another. The northbridge, among oth er things, is a bridge logically connecting the processor local (or front-side) bus to the PCI (and AGP) bus. Some southbridges, among other things, are bridges connecting the PCI bus to an ISA bus.

C

CMS—Code-Morphing™ Software. A dynamic translator and interpreter system that emulates some well-established architecture (e.g. x86) on top of Transmeta's unique VLIW hardware.

CPU—Central Processing Unit. The processor in a computer system. It executes the SW instructions which advance state and may in turn access peripheral devices. Generally excludes memory and I/O devices, unless embedded in the CPU.

CPX—Cycles Per X-tick. A measure of x86 machine efficiency/performance. At the same frequency and running the same application or benchmark, a lower CPX generally indicates better performance.

D

DDR—Double Data Rate. A bus protocol technique in which data is transferred on both rising and falling edges of the bus clock, effectively achieving twice the data throughput. It is the counterpart to SDR on which data is transferred only on rising or falling edges of the clock. Examples of DDR interfaces are the TM8000 MI (memory interface) unit, and the HyperTransport link from the TM8000 Virtual Northbridge to the southbridge.

DIMM—Dual In-line Memory Module. A memory (DRAM) module with a particular organization.

DMA—Direct Memory Access. Traditionally, the ability of devices on a bus to access main memory without the processor's intervention—a form of doing I/O where the CPU is not directly involved in the transfers, unlike programmed I/O (PIO). In modern computer systems the DMA facility is implemented in the memory controller, called the northbridge in PC terminology. In Transmeta products, the processor is involved to maintain T-cache coherence.

DRAM—Dynamic Random Access Memory. RAM implemented by capacitor circuits (single transistor cells, typically). Very dense, but requires frequent refreshing since the capacitors slowly discharge (leak), eventually losing their data contents. In addition, reads typically require rewriting because driving the output lines also discharges the capacitors. Refresh is sometimes implemented internally by parallel read/write.

E

ECC—Error Correcting Code. A system in which data words (esp. memory) are extended with a few extra bits such that some number of bit errors can be detected and some (smaller) number of bit errors can be corrected. Typically done on 64-bit words with 1 bit error correction and 2 bit error detection because a simple scheme requires only 8 additional bits, thereby taking no more storage than byte-parity and enabling the use of the same memory DIMMs.

EIP—Extended Instruction Pointer. The architectural 32-bit register in an x86 (a.k.a. IA-32) processor that holds the address of the currently executing instruction. The lower 16 bits are called the IP or `instruction pointer'.

EISA—Extended Industry Standard Architecture. A 32-bit peripheral (I/O) bus defined by several companies in the PC industry as a counterpart to IBM's proprietary MCA bus. Never dominant, it was superseded by the PCI bus.

F

FPR—Floating-Point Register. A register in the floating-point unit. It can contain floating-point data, integer data, or packed media data. It cannot be used to address memory directly.

FPU—Floating-Point Unit. A part of a modern CPU that contains floating-point registers and implements floating-point and media operations.

front-side bus—A bus connecting a CPU (or CPU core) to the memory controller/northbridge. Also called the processor local bus. It is typically CPU-core specific, and may be only conceptual when the northbridge is physically in the same chip as the CPU core, as in Transmeta products.

G

GART—Graphics Address Relocation Table. A facility in modern AGP-enhanced northbridge chips used to give SW and the graphics accelerator the illusion that they are dealing with contiguous physical memory which is in fact sparsely allocated out of a page pool by the operating system without regard to contiguity.

GR—General Register Same as GPR.

GPR—General Purpose Register. An integer register that can be used to contain integer data or memory addresses. As opposed to dedicated or special-purpose registers such as floating-point registers or special registers that control the memory subsystem, etc.

H

HT(1)—HyperTransport. See LDT.

HT(2)—HyperTransport controller. The unit in the TM8000 Virtual Northbridge that implements the HyperTransport host bridge.

HW—Hardware

HyperTransport—A new name for LDT.

I

ICE—In-Circuit Emulator. A hardware device that allows very low-level debugging of a target system when controlled by software from a debugging host. It typically allows single stepping of system instructions and low level operations before and while the operating system and/or BIOS are running.

I/O, IO—Input Output

IOIO—I/O-port I/O. On the x86 architecture, I/O performed by using IO ports, i.e. in and out instructions to ports. These ports are not memory-mapped. They are an altogether different address space.

IP(1)—Internet Protocol. A low-level network protocol that covers routing and delivery over large area networks. UDP and TCP are protocols built on top of IP, which is often built on top of Ethernet and other such protocols.

IP(2)—Intellectual Property. Knowledge protected by trade secret, copyright, or patents. Sometimes used to describe some synthesizable logic (e.g. Verilog) that provides some modular functionality and that companies license to other companies for the use in their chips, e.g. an IDE controller.

IRQ—Interrupt request line. Each interrupt request line results in a distinct interrupt vector to the CPU. Interrupt request lines can be uniquely assigned to devices or shared between devices. When unique, the interrupt handler can know exactly which device needs service. When shared, the interrupt handler must poll the devices that share the line to figure out which need service.

ISA(1)—Instruction Set Architecture. The software-visible portion of a CPU's organization. It typically consists of the resources (e.g. registers), operations (e.g. instructions), and encodings of both that software needs to use to make use of the computer. On traditional CPU families, what is common between the members of the family and allows SW to run across the family.

ISA(2)—Industry Standard Architecture. The 16-data-bit, 24-address-bit memory and peripheral (I/O) bus for the IBM PC-AT—the first with an Intel 286 processor. Until recently, all PCs had one. The trend these days, especially in laptops, but even on desktops, is to abandon it after moving all legacy devices to the southbridge.

L

L1—Level-1. First level of the cache hierarchy. On the TM8000 there is an L1 data cache and an L1 instruction cache.

L2—Level-2. Second level of the cache hierarchy. On the TM8000 there is a combined data and instruction L2 cache.

LDT—Lightning Data Transport. A peripheral interconnect fabric designed by AMD to replace the PCI bus and possibly others (AGP). Logically it is very similar to PCI. Physically it is not a bus at all, but instead a point-to-point connection. LDT devices can be daisy chained and additional chains can be split from the

primary chain. It is the TM8000's primary external interface. During 2001, LDT has been renamed 'HyperTransport'.

LongRun—A Transmeta technology that allows the x86 processor to change operating clock frequency and supply voltage on the fly to consume as little power as possible while satisfying the computational needs of the running OS and applications. It is implemented by a combination of HW (SW-controlled frequency and voltage) and SW (mechanisms and policies for frequency and voltage ramping).

LongRun Advanced Thermal Management—A Transmeta technology

l-value—Left value. A compiler term describing a location written by an assignment statement. Opposed to r-value which is a value that can be stored into such a location. Thus an assignment consists of an l-value and an r-value.

M

MCA(1)—Machine Check Architecture. An x86 extension that allows diagnostic software to inspect the state of some internal processor registers when an MCE is raised. It is processor-specific.

MCA(2)—Micro-Channel Architecture. A 32-bit peripheral (I/O) bus defined by IBM to supersede the ISA bus in the PS2 PC architecture. It was proprietary, and hence other companies came up with a different standard, EISA, that ultimately proved more popular. The fiasco over MCA marked the end of IBM's dominance of the PC system architecture.

MCE—Machine Check Exception. An x86 exception raised by an x86 processor when it encounters some irrecoverable internal condition. It indicates a failure in the processor, not the running program at the time the fault is raised.

MI—Memory Interface. The unit in the Virtual Northbridge that controls and communicates with the DRAM DIMMs. It supports only DDR DIMMs.

MMIO—Memory-mapped I/O. Input/output to devices by using accesses to the regular "memory" address space. Device control registers and buffers are mapped into the "memory" address space and respond to ordinary reads and writes on the bus by effecting the appropriate action.

MMX—Multi Media eXtensions. SIMD integer extensions to the x86 architecture.

molecule —A VLIW instruction encoding multiple operations that logically occur simultaneously. Each individual operation is called an atom.

MTRR—Memory Type Range Register. An x86 architectural extension by which the memory attributes of an access (cacheability) can be described at the physical level -- the address space can be divided into multiple regions with different cacheability properties for reads and writes. It was introduced by Intel to handle several problems in earlier systems where the cacheability of an access was known to the northbridge (PAB bits) but not the processor. It is in effect a processor-local "copy" of the PAB bits and memory top register.

N

NaN—Not A Number. An IEEE 754 format floating-point "value" that does not have a numeric value. They are computed as the masked response to invalid operations (e.g. divide 0 by 0), and also sometimes used for uninitialized memory. They come in two flavors. Signalling NaNs (SNaN) raise the invalid numeric exception unconditionally. Quiet NaNs (QNaN) only raise it in certain circumstances. Typical masked response is to produce quiet NaNs that will propagate through the computation contaminating the result (only comparisons can eliminate NaNs -- arithmetic produces NaN outputs when given any NaN inputs).

Natural Alignment—A power-of-two-sized memory operand is naturally aligned if the address of the operand is an integer multiple of the size of the operand. For example a 1-byte-sized operand is always naturally aligned. An 8-byte-sized operand is naturally aligned only at addresses that are multiples of 8, etc.

NC—Non-cached or Non-cacheable. A memory (load/store) operation that does not go through the cache subsystem. It may be an operation to an IO port (IOIO), memory mapped IO (MMIO), or a memory-bound

operation that CMS has decided must follow the NC path in order to guarantee cache coherence in the system as a whole or for performance reasons.

NMI—Non-Maskable Interrupt. An interrupt that cannot be blocked by software. The x86 NMI is only blocked by the execution of the SMI handler. Transmeta HW allows blocking x86 NMI, but CMS makes it appear that it is not blocked except in this circumstance.

normal memory—An x86 address space region where reads and writes (or x86 instruction fetches) can be optimized and reordered because it corresponds to cacheable DRAM on the memory controller in the northbridge.

Northbridge—A part of a PC's chip set that implements the memory controller, the DMA engine, and the PCI bus master. Modern ones also implement AGP. It is connected to the CPU via a non-standard local (or front-side) bus, and is connected to the southbridge via the PCI bus and a few additional signals.

O

OOL(1)—Out-Of-Line subroutine. A subroutine in CMS that implements the functionality of some complex x86 instruction. Rather than inlining the code in translations and the interpreter, these out of line handlers are used instead. The OOL doesn't have to implement the complete instruction -- part of it (e.g. loading from the stack) can be done in line.

OOL(2)—Object Oriented Language. Sometimes used abbreviation for a high-level computer language whose principal data items are organized as objects with lifetimes and properties independent of the programs that manipulate them. Often used for languages with message-based invocation semantics and modularity and encapsulation organized around data instead of code.

OS—Operating System. The main control program of a computer system. It typically multiplexes the computer's resources (including execution time) among competing application programs and provides several services to them such as uniform I/O APIs that abstract the details of the individual devices in use.

P

PAB—Programmable Attribute Bits. The set of bits in a traditional northbridge that control the access to the legacy region (640K - 1M), especially in regards to ROM shadowing. The PAB bits allow reads and writes to independently be routed to memory or the PCI bus.

PAE—Physical Address Extension. An extension to the x86 architecture by which physical addresses can be 36-bits wide, instead of the usual 32-bits wide. Each PTE/PDE becomes 8 bytes long instead of the usual 4 bytes long, the page tables are 3 levels deep instead of the usual 2 levels, and the large pages (as in PSE) are 2MB instead of 4MB.

PAT—Physical Attribute Table. An x86 architectural extension by which the memory attributes of an access (cacheability) can be described at a virtual level instead of physical level. Each virtual to physical mapping has a memory type index in the relevant PTE that indexes the PAT to produce a memory type that is then combined with the type from the MTRRs to produce the actual memory type for the access.

PC(1)—Personal Computer. A computer intended to be used by a single person, i.e. traditionally running a single-user operating system. More commonly, an x86-compatible computer, i.e. what used to be called an IBM-compatible computer, irrelevant of whether the operating system running on it makes it multi-user or not.

PC(2)—Program Counter. The conceptual register in a processor that holds the address of executing instructions. In pipelined processors there may not be such a register -- each pipeline stage may have its own copy. PCs are usually not visible as registers -- they are reified on interrupts/exceptions, and reflected on resume from exception/interrupt return. In the x86 architecture, the PC is called EIP.

PCI—Peripheral Component Interconnect. The peripheral (I/O) bus of a modern PC. It is a 32-bit bus with multiple address spaces: The I/O port (IOIO) address space, the ordinary memory-mapped (MMIO) address space, and the configuration address space. It was first introduced in the PC architecture by Intel with the Pentium processor, to rationalize and standardize the multiple peripheral buses then extant (VESA local bus, EISA, MCA, ISA). Although memory-like devices can be added to the bus, this is unusual, and memory is

typically elsewhere on a system. The system's northbridge typically connects to both memory and the PCI bus, but the memory is not on the PCI bus, although accessible from it by using DMA.

PDE—Page Directory Entry. A data structure in the page tables that aggregates a collection of virtual to physical mappings. It does not describe any virtual to physical mapping itself, but allows the page table walker to find the individual mappings. In the usual two-level x86 page table structure, the PDEs are the level between the root and the PTEs.

PGE—Page Global Extension. An extension to the x86 architecture by which specially-marked 'global' virtual to physical mappings survive TLB invalidations and can only be invalidated explicitly.

PIC—Programmable Interrupt Controller. Traditionally, a device on a PC's motherboard that multiplexed interrupts to the CPU. These days the PIC is implemented as part of modern southbridges, and not as separate components.

PIO—Programmed I/O (input/output). A way of doing I/O where the CPU actively executes instructions to perform the actual transfer of data, as opposed to DMA, where the CPU is passive: the transfer of data is done by an asynchronous device without any CPU instructions needed except to set up the transfer.

PSE—Page Size Extensions. An extension to the x86 architecture by which virtual pages can be either 4MB (large pages) or the usual 4KB. Virtual to physical mappings for large pages take the place of PDEs in the page tables.

PSE-36—Page Size Extensions - 36 bits. An extension to the x86 architecture that combines features of PSE and PAE. The PTEs and PDEs remain as in the normal 2-level page tables, but the 4MB pages can be mapped to a full 36-bit address space, instead of the usual 4KB address space.

PTE—Page Table Entry. A data structure in the page tables that describes the virtual to physical mapping for a single virtual page. In the usual multi-level x86 page table structure, the PTEs are the level farthest from the root.

Q

QNaN—Quiet NaN. See NaN.

R

RAM—Random Access Memory. The main memory of a computer system, addressable in no particular order.

ROM—Read-Only Memory. A memory-like device that can only be read and not written. Most modern ROMs are not really ROMs at all, but Flash ROMs or EEPROMs. EEPROMs are electrically-erasable programmable read-only memories, i.e. memory-like devices that are easy to read and difficult -- but possible -- to write. Flash ROMs are similar to EEPROMs that cannot be erased at the individual byte or word level but can be erased in units of moderate to large sectors. These EEPROMs and Flash ROMs are treated largely like true ROMs, except when their contents are upgraded, which is a very infrequent operation. Programming them (i.e. writing to them), is usually called "flashing", perhaps because early EPROMS (erasable programmable read-only memories) could only be erased by exposing them to ultra violet light.

r-value—Right value. See l-value.

S

SDR—Single Data Rate. A bus protocol technique in which data is transferred once per clock cycle, typically on rising edges of the clock. It is the counterpart to DDR in which data is trasferred twice per cycle.

serialization —The property that prevents two operations from being either reordered or carried out simultaneously. There are many forms of serialization in the x86 architecture, including but not restricted to code serialization and data serialization.

serializing instruction—An instruction that has some serialization property with respect to other instructions. x86 serializing instructions can be strong or weak.

SIMD—Single Instruction Multiple Data. A form of obtaining parallelism that consists on having single instructions operate on multiple data items simultaneously. The operations are usually uniform. The Intel variants of this, called MMX, SSE, and SSE2 divide large (64-bit or 128-bit) registers into multiple same-sized fields, and the operations operate on all the fields independently.

SMC—Self-Modifying Code. Code that either modifies its own instructions (the proper usage), or has its instructions modified by other pieces of code (the general usage). Particular common patterns are Bit-blit generators, which frequently generate code on the fly in a common buffer, dynamic compilers (such as CMS itself), and runtime patching of immediates.

SMI—System Management Interrupt. A special interrupt in an x86 CPU that causes the CPU to enter SMM.

SMM—System Management Mode. An extension to the x86 architecture that allows BIOSs to control peripheral devices without OS awareness. Often used to implement power-management functions such as powering off and restarting disks without the OS being aware. SMM has access to regions of memory (SMRAM) that the regular OS cannot address. SMM is the state of the processor when servicing an SMI.

SMP—Simultaneous Multi-Processor. A multi-CPU computer system (a multicomputer) where the CPUs logically share memory so that ordinary accesses from one CPU are visible from other CPUs without additional software support. The rules for serialization and ordering of memory references and visibility among the multiple CPUs can vary a lot. The memory may be either uniformly accessible or non-uniformly accessible, where logically there is no difference, but there is a large performance difference depending on the locality of the memory being accessed.

SMRAM—System Management RAM. An alternate address space, or extension of the regular x86 physical address space, used in SMM.

SNaN—Signalling NaN. See NaN.

Southbridge—A part of a PC's chip set that implements several sundry functions: bridge between the PCI and ISA buses, power-control of the platform, IDE bus controllers, serial, parallel, mouse, and keyboard ports, USB controllers, audio devices, ROM control, etc. Connected to the Northbridge via a PCI or LDT interface and to both the northbridge and the CPU through additional signals.

SRAM—Static Random Access Memory. RAM implemented out of storage cells that do not decay if unattended. Much less dense than DRAM, but typically much faster and not requiring refresh. Common modern designs are made out of 6 or 4 transistor cells (6T or 4T).

SSE—Streaming SIMD Extensions. SIMD floating-point extensions to the x86 architecture. There are two levels: SSE proper which introduced SIMD 32-bit (single precision) floating-point operations, and SSE2 which introduced SIMD 64-bit (double precision) floating-point operations. The extensions also include integer and MMX operations on the new XMM registers.

SSM—State Save Map. Data structure in SMRAM where an x86 CPU saves its state on entry to SMM and from where it restores it on exit from SMM.

sticky bit—A bit in a register is said to be sticky when it records a condition in a monotonic way. That is, when the condition arises, it changes state, and does not change back even if the condition disappears. It is only changed back by explicit software intervention.

STPCLK—Stop Clock. A (logical) signal in a PC system originating in the southbridge and used to stop the CPU at an appropriate point. Such stop can be a prelude to power down, or a temporary stop due to power management events. STPCLKs can be expected, when the CPU initiates a power management transition or shutdown, or unexpected, when the southbridge initiates clock throttling often caused by overheating.

STPGRNT—Stop Grant. A (logical) signal in a PC system originating in a CPU and used to inform the southbridge that a preceding STPCLK has been honored. Typically the STPGRNT must follow quickly after the STPCLK because southbridges are intolerant of many intervening transactions.

strapping options—Hardware interface pins whose logic value is sampled at reset time by hardware in order to choose an option among initial configuration possibilities. Examples of such strapping options are processor number for SMP systems, device ID selection addresses for identical devices on a bus, etc.

strongly-serializing instruction—A serializing instruction that constitutes an absolute barrier to previous and following instructions or memory operations. All previous instructions are guaranteed to have completed by the time that the strongly serializing instruction executes, and all following instructions are guaranteed not to have started until after the strongly serializing instruction completes. Strongly serializing instructions cause write combiners to be drained.

SW—Software

T

TBD—To Be Determined. Something not yet fully understood or specified.

thread—A control-flow and register state within an address space. Several threads can execute within a single process. Each thread typically has its own logical register set and stack but shares the address space and O/S resources with other threads in the same process. Sometimes threads also have thread-local data. Win32 threads and POSIX threads are the most common thread APIs.

TLB—Translation Look-aside Buffer. A hardware (and sometimes software) structure used to accelerate memory references in a paged environment. It is a cache of mappings from virtual addresses to physical addresses and some attributes of the mappings used to avoid walking the page tables on every access. These virtual to physical mappings are called translations, but are not to be confused with the instruction translations that CMS produces.

trip count—The number of iterations that a loop executes when entered from outside. Integer loops often have very low trip counts, in the vicinity of 3. Floating-point and media loops often have very high trip counts (hundreds of thousands or millions). Many loop optimizations are only worthwhile if the typical trip count is moderately high.

U

ULP—Unit of Least Precision. The least significant bit position in a floating-point number. Often used to describe the accuracy of the implementation of some floating-point function (e.g. transcendentals). If the accuracy is within 1/2 ULP, the answer is exact. If within 1 ULP, then a result is off by at most the quantity that a bit in the least significant position of the mantissa would represent, and so on.

UMA—Unified Memory Architecture. A system setup where all graphics memory (frame buffer and off frame storage for display lists, textures, etc.) lives in system memory -- directly accessible by the CPU, rather than in dedicated memory that the CPU cannot directly access. In other contexts, the acronym stands for 'uniform memory architecture', as opposed to 'non uniform memory architecture'.

USB—Universal Serial Bus. A bus in modern PCs that is distinguished by using a serial protocol over a small number of wires. Several devices that used to be connected through dedicated interfaces are now connected to the computer through this bus.

V

VICE—Virtual In-Circuit Emulator. A software component of Code Morphing software, coupled with a software program running on a debugging host that together emulate a true x86 ICE.

Visible interrupts—Interrupts that are not invisible interrupts.

VLIW—Very Long Instruction Word. A style of ISA characterized for large (in bits) instructions that group several independent operations that are executed in parallel and atomically.

VNB—Virtual North Bridge. A software component of CMS that logically emulates the registers and operations of a physical northbridge chip, mapping them to whatever hardware the Transmeta product implements directly.

VSB—Virtual South Bridge. A software component of CMS that logically emulates the registers and operations of a physical southbridge chip, mapping them to whatever hardware the Transmeta product implements directly.

W

weakly-serializing instruction —A serializing instruction that has some reordering properties with respect to some other instructions but not all. Thus some instructions are serialized with respect to a weakly serializing instruction, but not all instructions are.

X

x86—Of or compatible with 80x86 processors manufactured by Intel. Common x86 platforms include Intel's Pentium processors, AMD Athlon processors, and Transmeta processors.

X-bus —A derivative of the 8088's original bus. The 8088 was a variant of the 8086 (first processor in the x86 architecture line from Intel) with only an 8-bit-wide data bus (20-bit addresses). It is still often found hanging off the southbridge in a PC system to connect to the flash ROM containing the BIOS, and a few very simple devices such as the keyboard controller.

Index

T

U

V

W