

Accelerator Manual

Data Alignment

Addendum

Abstract

This document is for application programmers familiar with HP NonStop servers and the HP NonStop Kernel operating system. It documents the data alignment requirements of all TNS compilers and the Accelerator. It also describes misalignment diagnostic tools added in G06.17.

Product Version

NonStop Kernel G06.17
Subsystem Control Facility G06.17

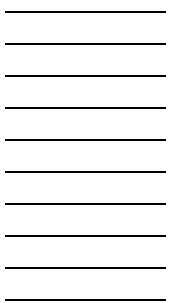
Supported Release Version Updates (RVUs)

This document supports G06.17 and all subsequent RVUs unless otherwise indicated in a new edition.

Part Number	Published
524963-001	August 2002

Document History

Part Number	Product Version	Published
524963-001	NonStop Kernel G06.17 Subsystem Control Facility G06.17	August 2002



Accelerator Manual Data Alignment Addendum

Glossary	Index	Tables
-----------------	--------------	---------------

- [What's New in This Guide](#) iii
- [Guide Information](#) iii
- [New and Changed Information](#) iii
- [About This Guide](#) v
- [Audience](#) v
- [Related Reading](#) v
- [Your Comments Invited](#) vi
- [Abbreviations](#) vi

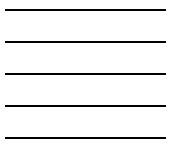
- [1. Introduction](#)
- [2. Misalignment Tracing Facility](#)
- [3. Misalignment Handling](#)
- [4. Misalignments in TNS and Accelerated Modes](#)

[Glossary](#)

[Index](#)

Tables

- [Table 3-1. TNS Misalignment Handling Methods](#) 3-1



What's New in This Guide

Guide Information

Abstract

This document is for application programmers familiar with HP NonStop servers and the HP NonStop Kernel operating system. It documents the data alignment requirements of all TNS compilers and the Accelerator. It also describes misalignment diagnostic tools added in G06.17.

Product Version

NonStop Kernel G06.17
Subsystem Control Facility G06.17

Supported Release Version Updates (RVUs)

This document supports G06.17 and all subsequent RVUs unless otherwise indicated in a new edition.

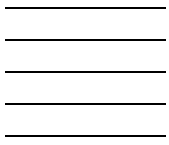
Part Number	Published
524963-001	August 2002

Document History

Part Number	Product Version	Published
524963-001	NonStop Kernel G06.17 Subsystem Control Facility G06.17	August 2002

New and Changed Information

This is a new document.



About This Guide

- [Audience](#) on page v
- [Related Reading](#) on page v
- [Your Comments Invited](#) on page vi
- [Abbreviations](#) on page vi

Audience

This guide is intended for application programmers who are familiar with the following HP NonStop systems products:

- NonStop servers
- NonStop Kernel operating system
- Accelerator

Related Reading

Manual

Accelerator Manual

C/C++ Programmer's Guide Data Alignment Addendum

COBOL85 Manual Data Alignment Addendum

EMS Manual

Guardian Programmer's Guide

Open System Services System Calls Reference Manual

pTAL Reference Manual Data Alignment Addendum

SCF Reference Manual for the Kernel Subsystem

TAL Programmer's Guide Data Alignment Addendum

Description

Explains how to improve the performance of TNS programs running on NonStop Series/RISC systems.

Documents the data alignment requirements of C and C++ for HP NonStop systems.

Documents the data alignment requirements of COBOL85 for HP NonStop systems.

Describes the Event Management Service (EMS). The misalignment tracing facility generates EMS events.

Explains how to use the Guardian programmatic interface of the NSK operating system.

Describes all OSS system calls for the NSK operating system.

Documents the data alignment requirements of Portable TAL (pTAL).

Describes the Subsystem Control Facility (SCF), whose user interface you can use to control tracing.

Documents the data alignment requirements of Transaction Application Language (TAL).

Your Comments Invited

After using this guide, please take a moment to send us your comments. You can do this by:

- Completing a Contact NonStop Publications form online at <http://nonstop.compaq.com/view.asp?FOID=20>.
- Faxing or mailing the form called the Reader Comment Card, which is included as a separate file in Total Information Manager (TIM) collections and located at the back of printed manuals. Our fax number and mailing address are included on the form.
- Sending an e-mail message to the address included on the form. We'll immediately acknowledge receipt of your message and send you a detailed response as soon as possible. Be sure to include your name, company name, address, and phone number in your message. If your comments are specific to a particular manual, also include the part number and title of the manual.

Many of the improvements you see in manuals are a result of suggestions from our customers. Please take this opportunity to help us improve future manuals.

Abbreviations

CISC. complex instruction-set computing

EMS. Event Management Service

RISC. reduced instruction-set computing

SCF. Subsystem Control Facility

RVU. release version update

TAL. Transaction Application Language

1 Introduction

On TNS systems, a word is 16 bits. The TNS instruction set includes data access instructions that use 32-bit byte addresses that must be even-byte aligned (that is, aligned 0 modulo 2) for correct operation. This requirement remains even after the Accelerator translates TNS code into RISC code. In TNS mode and accelerated mode, addresses that are odd-byte aligned (that is, aligned 1 modulo 2) are called [misaligned](#). TNS processors consistently “round down” misaligned addresses (that is, they ignore the low-order bit).

TNS/R processors handle the misaligned addresses of TNS programs inconsistently, rounding down some but not others and behaving differently in TNS mode and accelerated mode. These problems do not occur in programs that follow all of the programming rules for TNS languages, but compilers cannot catch misaligned addresses that are computed at run time.

The behavior of TNS programs with misaligned addresses on TNS/R processors is almost impossible to predict. If you are not sure that your program has only [aligned](#) addresses, you can use the tracing facility to detect whether programs are using misaligned pointers, and if so, where. You should then change the programs to avoid misalignment.

In addition to the Accelerator, the data misalignment issue might affect the following HP compilers. If you use these compilers, see the appropriate data alignment addenda:

Compiler	T Number	Addendum
TNS C	T9255	<i>C/C++ Programmer's Guide Data Alignment Addendum</i>
TNS C++	T9541	
TNS <code>c89</code>	T8629	
TNS COBOL85	T9257	<i>COBOL85 Manual Data Alignment Addendum</i>
TNS/R pTAL	T9248	<i>pTAL Reference Manual Data Alignment Addendum</i>
TNS TAL	T9250	<i>TAL Programmer's Guide Data Alignment Addendum</i>

2 Misalignment Tracing Facility

The misalignment tracing facility is enabled or disabled on a system-wide basis (that is, for all processors in the node). By default, it is enabled (set to ON). It can be disabled (set to OFF) only by the persons who configure the system, by means of the Subsystem Control Facility (SCF) attribute MISALIGNLOG. Instructions are in the *SCF Reference Manual for the Kernel Subsystem*.

Note. HP recommends that the MISALIGNLOG attribute be left ON (its default setting) so that any process that is subject to rounding of misaligned addresses will generate log entries, facilitating diagnosis and repair of the code. Only if the volume of misalignment events becomes burdensome should this attribute be turned OFF.

When a misaligned address causes an exception that RVUs prior to G06.17 would have rounded down, the tracing facility traces the exception.

Note. The tracing facility does not count and trace every misaligned address, only those that cause round-down exceptions. Other accesses that use misaligned addresses without rounding them down do not cause exceptions and are not counted or traced. Also, only a periodic sample of the counted exceptions are traced by means of their own EMS event messages.

While a process runs, the tracing facility:

- Counts the number of misaligned-address exceptions that the process causes (the exception count)
- Records the program address and code-file name of the instruction that causes the first misaligned-address exception

Because a process can run for a long time, the tracing facility samples the process (that is, checks its exception data) periodically (approximately once an hour). If the process recorded an exception since the previous sample, the tracing facility records an entry in the EMS log. If the process ends, and any exception has occurred since the last sample, the operating system produces a final Event Management Service (EMS) event.

The EMS event includes:

- The process's exception count
- Details about one misaligned-address exception, including the program address, data address, and relevant code-file names

Sampling is short and infrequent enough to avoid flooding the EMS log, even for a continuous process with many misaligned-address exceptions. One sample logs a maximum of 100 events, and at most one event is logged for any process.

If misaligned-address exceptions occur in different periods of a process, the operating system produces multiple EMS events for the same process, and these EMS events might have different program addresses.

For more information about EMS events or the EMS log, see the *EMS Manual*.

3

Misalignment Handling

Misalignment handling is determined by the following SCF attributes, which are set system-wide (that is, for all processors in the node) by the persons who configure the system:

- MISALIGNLOG
- TNSMISALIGN
- NATIVEATOMICMISALIGN (applies only to programs running in [TNS/R native mode](#), and therefore, does not apply to the Accelerator)

MISALIGNLOG enables or disables the tracing facility (see [Section 2, Misalignment Tracing Facility](#)).

TNSMISALIGN applies to all programs running in TNS mode or accelerated mode, but not to programs running in TNS/R native mode.

[Table 3-1](#) lists and describes the possible settings for TNSMISALIGN. Each setting represents a misalignment handling method. For more information about TNSMISALIGN, see the *SCF Reference Manual for the Kernel Subsystem*.

Table 3-1. TNS Misalignment Handling Methods

Method	Description
ROUND (default)*	After rounding down a misaligned address, the system proceeds to access the address, as in G06.16 and earlier RVUs.
FAIL	Instead of rounding down a misaligned address and proceeding to access the target, the operating system considers the instruction to have failed. For a Guardian process, this failure generates an Instruction Failure trap (trap #1). By default, this trap causes the process to go into the debugger, but the program can specify other behavior (for example, process termination or calling a specified trap-handling procedure). For information about trap handling, see the <i>Guardian Programmer's Guide</i> . For a OSS process, this failure generates a SIGILL signal (signal #4). By default, this signal causes process termination, but the program can specify other behavior (for example, entering the debugger or calling a specified signal-handler procedure). The signal cannot be ignored. For information about signal handling, see the explanation of the <code>sigaction()</code> function in the <i>Open System Services System Calls Reference Manual</i> .
NOROUND	The system uses the operand's given odd address (not rounded down) to complete the operation. If the operation is an atomic operation, atomicity is no longer guaranteed.

* Use this method on production systems to avoid changing the semantics of old TNS programs. FAIL could cause possibly fatal Instruction Failure traps in faulty TNS programs. NOROUND might change the semantics of some faulty programs.

The method that you choose does not apply to every misaligned address, only to those that would have been rounded down in RVUs prior to G06.17.

Note. ROUND and NOROUND misalignment handling are both intended as temporary solutions, not as a substitute for changing your program to ensure that it has only aligned addresses.

Programs that depend on NOROUND misalignment handling cannot be safely migrated to all present and future NonStop Kernel platforms or to systems configured with ROUND or FAIL misalignment handling.

Programs that depend on ROUND misalignment handling cannot be safely migrated “as is” to future NonStop Kernel platforms or to systems configured with NOROUND or FAIL misalignment handling.

4

Misalignments in TNS and Accelerated Modes

TNS compilers catch most violations of programming rules for their languages; however, some coding errors, particularly misuses of pointers and addresses, can produce erroneous TNS object code without compile-time or run-time warnings. When erroneous TNS object code executes, its misbehavior depends on its execution environment (TNS mode or accelerated mode, or execution on TNS CISC systems). These execution environments are compatible for legal TNS object code, but they sometimes handle erroneous TNS object code differently.

A common cause of erroneous TNS object code is a source program that uses odd-byte addresses in contexts where the compiler expects even-byte (word-aligned) operands. When the compiler applies a TNS word instruction to an operand with an odd-byte address, run-time results are unpredictable. The instruction might do any, or any combination, of the following:

- “Round down” the [misaligned](#) address (as the TNS CISC processors always did)
- Use the misaligned address without rounding it down
- Cause the instruction to fail

The TNS instruction’s misalignment behavior depends on the following:

- Operand size (16-bit, 32-bit, or 64-bit)
- Execution mode

In TNS mode or accelerated mode, the action also depends on the following:

- TNS opcodes such as QCMP
 - Although 64-bit integers with odd-byte addresses usually are not rounded down, they can be—QCMP is one such example.
- Contextual optimizations
- How the instruction's result is used by later instructions
- The SCF attribute TNSMISALIGN (see [Section 3, Misalignment Handling](#))

The varying behaviors and symptoms that misalignment errors cause can be confusing, for the following reasons:

- A source program can produce erroneous TNS object code that seems to execute correctly until you change the source program in an apparently innocent way.
- The program might behave differently in TNS mode than it does in accelerated mode. (Neither mode is better at avoiding alignment problems.)
- The debugger might display a different value for a variable than the program sees, because the debugger never “rounds down” addresses.
- The program might “round down” addresses for storage operations but not for subsequent retrieval operations of the same misaligned operand (or the reverse).

To avoid these confusing behaviors:

- Use pointer conversions as recommended for the source code language so that TNS word instructions are never applied to odd-byte addresses.
- Set the system’s SCF attribute TNSMISALIGN to either FAIL or NOROUND (see [Table 3-1](#) on page 3-1). FAIL can be unsafe on production systems, but is recommended on test systems.

Glossary

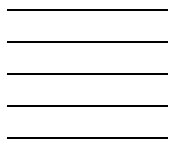
aligned. In TNS/R native mode, a data item is aligned if its address is a multiple of its size; for example, a 4-byte data item is aligned if its byte address is a multiple of 4.

misaligned. In TNS mode and accelerated mode, an erroneous address that is odd-byte aligned; in TNS/R native mode, an inefficient address that is not [aligned](#).

signal. A software interrupt that provides a way of handling certain events, such as detection of a hardware (or software) fault, a timer expiration, a lack of system resources, a process sending a signal to itself, or (for OSS processes) a change in the execution state of another process or another process sending a signal. A signal is an often an indication of a run-time event that requires immediate attention; many such events preclude continuing the interrupted instruction stream. Signals are generated for TNS/R native Guardian processes and all OSS processes. (TNS Guardian processes receive traps instead.) The signal mechanism is much richer for OSS than for Guardian processes. A SIGILL signal indicates that an instruction could not be executed because the instruction or its data were invalid. Compare to [trap](#).

TNS/R native mode. The operational environment in which unaccelerated native-compiled RISC instructions execute.

trap. A software interrupt that provides a way of handling certain events, such as detection of a hardware (or software) fault, a timer expiration, or a lack of system resources. A trap is often an indication of a run-time event that requires immediate attention; most such events preclude continuing the interrupted instruction stream. Traps are generated for TNS Guardian processes. (TNS/R native Guardian processes and all OSS processes receive signals instead.) An Instruction Failure trap indicates that an instruction could not be executed because the instruction or its data were invalid. Compare to [signal](#).



Index

A

Address misalignment
causes of [1-1](#), [4-1](#)
handling [3-1/3-2](#)
symptoms of [4-2](#)
tracing facility for [2-1](#)

D

Data misalignment
See Address misalignment
Debugger [4-2](#)
Default misalignment handling
method [3-1/3-2](#)

E

EMS [2-1](#)
Event Management Service [2-1](#)
Execution environment [4-1](#)
Execution mode [4-1](#)

F

FAIL misalignment handling
method [3-1/3-2](#)

I

Instruction Failure trap (trap #1) [3-1](#)

M

MISALIGNLOG attribute (SCF) [2-1](#), [3-1](#)
Misalignment
See Address misalignment
Misalignment tracing facility [2-1](#)

N

NATIVEATOMICMISALIGN attribute
(SCF) [3-1](#)
NOROUND misalignment handling
method [3-1/3-2](#)

P

Production systems [3-1](#)

R

ROUND (default) misalignment handling
method [3-1/3-2](#)
Rounding down addresses [1-1](#), [4-1](#), [4-2](#)

S

SCF user interface
misalignment handling and [3-1](#)
misalignment tracing facility and [2-1](#)
SIGILL signal (signal #4) [3-1](#)
Subsystem Control Facility
See SCF user interface

T

TNS opcodes [4-1](#)
TNSMISALIGN attribute (SCF) [3-1](#), [4-1](#),
[4-2](#)
Tracing facility [2-1](#)
Trap #1 [3-1](#)

