

# *The V8+ Technical Specification*



**Sun Microsystems Computer Company**

A Sun Microsystems, Inc. Business  
2550 Garcia Avenue  
Mountain View, CA 94043 USA  
415 960-1300 fax 415 969-9131

Part No.: 802-7447-10  
Revision 50, August 1996

Copyright 1996 Sun Microsystems, Inc. 2550 Garcia Avenue, Mountain View, California 94043-1100 U.S.A.

All rights reserved. This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any.

Portions of this product may be derived from the UNIX<sup>®</sup> system and from the Berkeley 4.3 BSD system, licensed from the University of California. UNIX is a registered trademark in the United States and in other countries and is exclusively licensed by X/Open Company Ltd. Third-party software, including font technology in this product, is protected by copyright and licensed from Sun's suppliers.

RESTRICTED RIGHTS LEGEND: Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 and FAR 52.227-19.

Sun, Sun Microsystems, the Sun logo, SunDocs, SunExpress, OpenWindows, and Solaris are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and in other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the United States and in other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK<sup>®</sup> and Sun<sup>™</sup> Graphical User Interfaces were developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox Corporation in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a nonexclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

X Window System is a trademark of X Consortium, Inc.

THIS PUBLICATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

---

Copyright 1996 Sun Microsystems, Inc., 2550 Garcia Avenue, Mountain View, Californie 94043-1100 U.S.A.

Tous droits réservés. Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie et la décompilation. Aucune partie de ce produit ou de sa documentation associée ne peut être reproduite sous aucune forme, par quelque moyen que ce soit, sans l'autorisation préalable et écrite de Sun et de ses bailleurs de licence, s'il y en a.

Des parties de ce produit pourront être dérivées du système UNIX<sup>®</sup> et du système Berkeley 4.3 BSD licencié par l'Université de Californie. UNIX est une marque enregistrée aux Etats-Unis et dans d'autres pays, et licenciée exclusivement par X/Open Company Ltd. Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fournisseurs de Sun.

Sun, Sun Microsystems, le logo Sun, SunDocs, SunExpress, OpenWindows, et Solaris sont des marques déposées ou enregistrées de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays. Toutes les marques SPARC, utilisées sous licence, sont des marques déposées ou enregistrées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc.

Les utilisateurs d'interfaces graphiques OPEN LOOK<sup>®</sup> et Sun<sup>™</sup> ont été développés de Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox Corporation pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique, cette licence couvrant aussi les licenciés de Sun qui mettent en place les utilisateurs d'interfaces graphiques OPEN LOOK et qui en outre se conforment aux licences écrites de Sun.

Le système X Window est un produit du X Consortium, Inc.

CETTE PUBLICATION EST FOURNIE "EN L'ETAT" SANS GARANTIE D'AUCUNE SORTE, NI EXPRESSE NI IMPLICITE, Y COMPRIS, ET SANS QUE CETTE LISTE NE SOIT LIMITATIVE, DES GARANTIES CONCERNANT LA VALEUR MARCHANDE, L'APTITUDE DES PRODUITS A REpondre A UNE UTILISATION PARTICULIERE OU LE FAIT QU'ILS NE SOIENT PAS CONTREFAISANTS DE PRODUITS DE TIERS.



# Contents

---

<b>1. V8+ Introduction</b> .....	<b>1-1</b>
<b>2. Differences</b> .....	<b>2-1</b>
2.1 Low-Level System Information .....	2-1
2.1.1 Permitted Instructions .....	2-1
2.1.2 Processor-Specific Opcodes in V8+ Objects.....	2-3
2.1.3 C “long long” .....	2-3
2.1.4 Function Register Volatility .....	2-3
2.1.5 Function Registers with Unassigned Roles .....	2-4
2.1.6 State Register Initialization .....	2-4
2.1.7 Auxiliary Vector .....	2-4
2.2 Object Files .....	2-4
2.2.1 ELF Header .....	2-4
2.2.2 Compile-time Setting .....	2-6
2.2.3 Static Linking.....	2-6
2.2.4 Run-time Checking.....	2-7

---

2.2.5 Relocation Types .....	2-7
2.3 Changes to <code>core(4)</code> .....	2-7
2.4 Changes to <code>ucontext(5)</code> .....	2-8
2.5 Changes to <code>proc(4)</code> .....	2-8
<b>A. UltraSPARC-I-Specific Changes .....</b>	<b>A-1</b>

## *Tables*

---

Table 2-1	V9 Instructions Allowed in V8+ .....	2-2
Table 2-2	Processor-Specific ELF Header Field Settings for a V8+ Object	2-5
Table 2-3	Examples of Supported Flags .....	2-6
Table 2-4	<code>e_machine</code> and <code>e_flags</code> Settings Based on Compilation Flags	2-6



## *Preface*

---

*The V8+ Technical Specification* is intended to give ISVs and other SPARC™ developers information about SPARC V8+'s restrictions and features.

We expect usage of V8+ objects to fall into one of three categories:

1. Platform-specific libraries, such as graphics libraries, and DDX drivers that need to use V8+ facilities in order to meet performance goals. This includes libraries that interpose on the standard multiply, divide, and storage copy and fill routines, so that vanilla V8 applications can transparently take advantage of performance-enhancing V8+ features.
2. End-user-written applications, where the end user compiles for a specific platform, wants every last bit of performance possible, and portability of binaries across different SPARC-based systems, are not a consideration.
3. Certain third-party applications where the ISV has seriously considered the costs and benefits of maintaining two binaries (V8 and V8+), and has decided that the extra performance outweighs the maintenance cost. Note that a V8+ application is ineligible for SPARCware branding.

### *Before You Read This Book*

It is assumed that you are familiar with the contents of these books:

- *SPARC Architecture Manual*, Version 9, SPARC International, SAV09R1429309, ISDN 0-13-099227-5.
- *System V Application Binary Interface*, ISBN 0-13-877598-2

- *System V Application Binary Interface SPARC Processor Supplement*, ISBN 0-13-877630-X
- *Linker and Libraries Guide* (part number 802-6319-10)

## How This Book Is Organized

The chapter titles and subject matter are as follows

**Chapter 1, “V8+ Introduction”**, is an overview of the V8+ specification.

**Chapter 2, “Differences”**, is comprised of definitions and discussions of the differences between SPARC V8+ and SPARC V8; also, the SPARC V9 and UltraSPARC™ features, which comprise the V8+ specification.

**Appendix A, “UltraSPARC-I-Specific Changes”**, is a discussion of these changes made to V8 for V8+.

## Typographic Conventions

The following table describes the uses of the type faces used in this book.

Typeface or Symbol	Meaning	Example
AaBbCc123	The names of commands, files, and directories; on-screen computer output	Edit your <code>.login</code> file. Use <code>ls -a</code> to list all files. machine_name% You have mail.
<b>AaBbCc123</b>	What you type, contrasted with on-screen computer output	<div style="border: 1px solid black; padding: 5px; width: fit-content;">           machine_name% <b>su</b>            Password:         </div>
<i>AaBbCc123</i>	Command-line placeholder: replace with a real name or value	To delete a file, type <code>rm filename</code> .
<b><i>AaBbCc123</i></b>	Book titles, new words or terms, or words to be emphasized	Read Chapter 6 in the <i>User's Guide</i> . These are called <i>class</i> options. You <i>must</i> be root to do this.

---

## Shell Prompts

The following table shows the default system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

Shell	Prompt
C shell	machine_name%
C shell superuser	machine_name#
Bourne shell and Korn shell	\$
Bourne shell and Korn shell superuser	#

## Related Documents

The following documents contain topics that relate to the information in *The V8+ Technical Specification*.

- *System V Application Binary Interface*, ISBN 0-13-877598-2
- *System V Application Binary Interface SPARC Processor Supplement*, ISBN 0-13-877630-X  
Referred to as 'SysV SPARC ABI'.
- *SPARC Architecture Manual*, Version 8, SPARC International, SAV080SI9106  
Referred to as SPARC V8.
- *SPARC Architecture Manual*, Version 9, SPARC International, SAV09R1419309.  
Referred to as SPARC V9.
- *UltraSPARC-1 User's Manual*, Revision 1.4, January 1996, STP1030-UG  
Referred to as UltraSPARC-I.
- *Solaris 2.x Reference Manual*
- *Linker and Libraries Guide* (part number 802-6319-10).

---

## *Sun<sup>TM</sup> Welcomes Your Comments*

Please use the *Reader Comment Card* that accompanies this document. We are interested in improving our documentation and welcome your comments and suggestions.

If a card is not available, you can email or fax your comments to us. Please include the part number of your document in the subject line of your email or fax message.

- Email: `smcc-docs@sun.com`
- Fax: SMCC Document Feedback  
1-415-786-6443

---

## *Ordering Sun Documents*

SunDocs™ is a distribution program for Sun Microsystems™ technical documentation. Easy, convenient ordering and quick delivery is available from SunExpress™. You can find a full listing of available documentation on the World Wide Web: <http://www.sun.com/sunexpress/>

---

<b>Country</b>	<b>Telephone</b>	<b>Fax</b>
United States	1-800-873-7869	1-800-944-0661
United Kingdom	0-800-89-88-88	0-800-89-88-87
France	05-90-61-57	05-90-61-58
Belgium	02-720-09-09	02-725-88-50
Luxembourg	32-2-720-09-09	32-2-725-88-50
Germany	01-30-81-61-91	01-30-81-61-92
The Netherlands	06-022-34-45	06-022-34-46
Sweden	020-79-57-26	020-79-57-27
Switzerland	155-19-26	155-19-27
Japan	0120-33-9096	0120-33-9097

---



# V8+ Introduction

---



V8+ is the name for the usage of SPARC Version 9 and UltraSPARC features in a SPARC Version 8 environment.

---

**Note** – V8+ is not being promoted as a new ABI, particularly as it is not a system-independent set of interfaces.

---

The V8+ environment offers SPARC V9 and UltraSPARC features and performance enhancements. A vanilla V8 ABI user-level binary benefits from most of these features and performance enhancements if dynamically linked. This is due to a new mechanism called Auxiliary Filters which interposes platform-specific library routines.

For more information about Auxiliary Filters, refer to *Linker and Libraries Guide*.

Due to the nature of these UltraSPARC platform-specific features, management of extra SPARC V9 and UltraSPARC-specific register state is required. This management of the extra register set requires interface changes in a few areas. These are discussed in Chapter 2, “Differences”.

---

**Note** – Unless otherwise specified, interfaces established by this document have a classification of Unstable.

---

A V8+ object is an ELF-executable or shared object that violates the SPARC V8 ABI. It does this by issuing instructions specific to SPARC V9 or UltraSPARC that are not available in SPARC V8, and/or by using registers specific to SPARC V9 or UltraSPARC that are not available in SPARC V8.

---

**Note** – General usage of V8+ objects is strongly discouraged.

---

We expect usage of V8+ objects to fall into one of three categories:

1. Platform-specific libraries, such as graphics libraries, and DDX drivers, that need to use V8+ facilities in order to meet performance goals. This includes libraries that interpose on the standard multiply, divide, and storage copy and fill routines, so that vanilla V8 applications can transparently take advantage of performance-enhancing V8+ features.
2. End-user-written applications, where the end user compiles for a specific platform, wants every last bit of performance possible, and portability of binaries across different SPARC-based systems, are not a consideration.
3. Certain third-party applications where the ISV has seriously considered the costs and benefits of maintaining two binaries (V8 and V8+), and has decided that the extra performance outweighs the maintenance cost. Note that a V8+ application is ineligible for SPARCware branding.

---

This chapter includes descriptions of the V8+ specification features and their differences from the V8 specification.

### 2.1 Low-Level System Information

This section should be read in conjunction with SysV SPARC ABI Chapter 3.

#### 2.1.1 Permitted Instructions

You may use all non-privileged V8 instructions and all non-privileged V9 instructions, except as listed below. The exceptions to these rules are based on the fact that the upper 32 bits of the *in* and *local* registers are not preserved in user stack frames. Thus, their contents are unpredictable, even after execution of an instruction intended to set them. A violation of these restrictions produces unpredictable results. The kernel preserves the upper 32 bits of the *out* and *global* registers on a trap or interrupt.

---

**Note** – *Out* registers are volatile across function calls. Even if the function being called is known not to use them, it could perform a `SAVE`, causing the *outs* to become *ins*, and putting their upper 32 bits at risk.

---

---

**Note** – Other restrictions limit the *global* registers which may be used; see below.

---

The section numbers in this table refer to *SPARC Architecture Manual, Version 9*.

Table 2-1 V9 Instructions Allowed in V8+

Instructions	Section Number	Restriction
BPr	A.3	rs1 must be a <i>global</i> (%g) or <i>out</i> (%o) register.
FMOVr	A.33	
MOVr	A.35	
BPcc	A.7	Use of %xcc is unpredictable unless the operation setting the condition codes uses <i>global</i> or <i>out</i> registers as sources.
FMOVcc	A.32	
MOVcc	A.34	
TCC	A.60	
CASXA	A.9	rs2 and rd must be <i>global</i> or <i>out</i> registers.
LDX	A.27	rd must be a <i>global</i> or <i>out</i> register.
LDXA	A.28	
STX	A.53	
STXA	A.54	
MULX	A.36	rs1, rs2 (if used), and rd must be <i>global</i> or <i>out</i> registers.
SDIVX		
UDIVX		
POPC	A.40	rs2 (if used) must be a <i>global</i> or <i>out</i> register.
RDTICK	A.43	rd must be a <i>global</i> or <i>out</i> register.
RDASR	A.43	rd must be a <i>global</i> or <i>out</i> register if the ASR is wider than 32 bits.
SLLX	A.48	With shift counts less than 32: rs1 and rd must be <i>global</i> or <i>out</i> registers.
SRLX		
SRAX		
SLLX	A.48	With shift counts greater than or equal to 32: rd must be a <i>global</i> or <i>out</i> register.

Table 2-1 V9 Instructions Allowed in V8+ (Continued)

Instructions	Section Number	Restriction
SRLX	A.48	With shift counts greater than or equal to 32: <i>rs1</i> must be a <i>global</i> or <i>out</i> register.
SRAX		
WRASR	A.62	<i>rs1</i> and <i>rs2</i> (if used) must be <i>global</i> or <i>out</i> registers if the ASR is wider than 32 bits.

### 2.1.2 Processor-Specific Opcodes in V8+ Objects

A V8+ object may contain processor-specific opcodes. It is strongly recommended that such opcodes not be compiler-generated, but used in assembler routines and in-line templates. Processor-specific opcodes should only be used by processor-specific shared libraries. They should not appear in application executables and shared objects that are intended to be portable to other systems.

### 2.1.3 C “long long”

Compare with SysV SPARC ABI, Figure 3-1, p.3-2.

A C long long is an 8-byte quantity, aligned on an 8-byte boundary. It usually occupies the low-order halves of a register pair: LDD and STD (or two LDs and STs) are usually used to move a long long; not LDX/STX. If you use LDX/STX, you must use a *global* or *out* register.

### 2.1.4 Function Register Volatility

Compare with SysV SPARC ABI, Figure 3-15, p.3-11.

- `%g5`, *global 5*, is volatile across function calls and may be used by an application.
- `%d32` (`%q32`) through `%d62` (`%q60`) and `%ccr` are volatile across function calls.
- `%asi` by default holds `ASI_PRIMARY_NOFAULT`. If modified, it must be restored to the default value before calling another function or returning.
- `%fprs` is reserved for use by the system.
- `%tick` is reserved.

### 2.1.5 Function Registers with Unassigned Roles

Compare with SysV SPARC ABI, p.3-16.

The following have no specified role in the calling sequence:

- “%f32” (%d32) through “%f63” (%d62)
- %g5; no longer reserved for system software
- %ccr register

The %asi register will contain ASI\_PRIMARY\_NOFAULT on function entry and must contain that value on function call or return. %tick is reserved.

### 2.1.6 State Register Initialization

Register	Initial State
CCR(icc,xcc)	Unspecified
ASI	ASI_PRIMARY_NOFAULT

Compare with SysV SPARC ABI, p.3-31.

### 2.1.7 Auxiliary Vector

Compare with SysV SPARC ABI, p.3-5.

V8+ uses the AT\_FLAGS entry in the auxiliary vector to communicate platform support between the kernel and the dynamic linker. See Section 2.2.4, “Runtime Checking,” on page 2-7, for more information.

## 2.2 Object Files

This section should be read in conjunction with SysV SPARC ABI, Chapter 4.

### 2.2.1 ELF Header

Compare with SysV SPARC ABI, p.4-1.

A binary that uses SPARC V9 and UltraSPARC features, referred to as a V8+ binary, must contain an indication of this in its ELF header since it will not execute in a vanilla V8 environment.

Processor identification resides in the `e_machine` field of the ELF header. A V8+ binary is identified by an `e_machine` field value of `EM_SPARC32PLUS`, which has the value 18. Note that the `e_ident[EI_CLASS]` field of the ELF header for a V8+ binary is `ELFCLASS32`.

SPARC V9 allows implementations to provide implementation-specific extensions. Utilization of such extensions, as well as utilization of generic V9 features, is indicated in the `e_flags` field. These `e_flags` bit definitions have been added to `<sys/elf_sparc.h>`. `EF_SPARC_32PLUS_MASK` bits are reserved for indicating V8+ typeness.

```
EF_SPARC_32PLUS_MASK 0xffff00 /* bits indicating V8+ type */
EF_SPARC_32PLUS      0x000100 /* generic V8+ features */
EF_SPARC_SUN_US1     0x000200 /* Sun UltraSPARC1 extensions */
EF_SPARC_HAL_R1      0x000400 /* HAL R1 extensions */
```

Based on the preceding, the settings for these processor-specific fields of the ELF header for a V8+ binary are as follows.

*Table 2-2* Processor-Specific ELF Header Field Settings for a V8+ Object

Field	Setting
<code>e_ident[EI_CLASS]</code>	<code>ELFCLASS32</code>
<code>e_ident[EI_DATA]</code>	<code>ELFDATA32MSB</code>
<code>e_machine</code>	<code>EM_SPARC32PLUS</code>
<code>e_flags</code>	<code>EF_SPARC_32PLUS</code> OR'd with zero or more of the other bits within <code>EF_SPARC_32PLUS_MASK</code>

### 2.2.2 Compile-time Setting

This is an example of the various flags a compilation system might support to allow for building V8 and V8+ binaries. The example names of the flags are only for this description and may not correspond to any compiler, existing or planned.

Table 2-3 Examples of Supported Flags

Flag	Purpose
default mode	To build a V8 binary
-v8plus flag <sup>1</sup>	Specifies building a generic V8+ binary
-v8plus_sun_us1 flag <sup>2</sup>	Specifies building a V8+ binary with Sun UltraSPARC 1 extensions
-v8plus_hal_r1 flag	Specifies building a V8+ binary with HAL R1 extensions

1. This corresponds to the -xarch=v8plus flag of the Sun C 4.0 compiler.
2. This corresponds to the -xarch=v8plusa flag of the Sun C 4.0 compiler.

This table describes the setting of the e\_machine and e\_flags fields of the resulting binary based on the compilation flags.

Table 2-4 e\_machine and e\_flags Settings Based on Compilation Flags

	e_machine	e_flags
default	EM_SPARC	0
-v8plus <sup>1</sup>	EM_SPARC32PLUS	EF_SPARC_32PLUS
-v8plus_sun_us1 <sup>2</sup>	EM_SPARC32PLUS	EF_SPARC_32PLUS   EF_SPARC_SUN_US1
-v8plus_hal_r1	EM_SPARC32PLUS	EF_SPARC_32PLUS   EF_SPARC_HAL_R1

1. This corresponds to the -xarch=v8plus flag of the Sun C 4.0 compiler.
2. This corresponds to the -xarch=v8plusa flag of the Sun C 4.0 compiler.

### 2.2.3 Static Linking

The link editor, ld(1), links any combination of V8 and V8+ objects:

- If all linked objects contain e\_machine fields of EM\_SPARC, the resulting linked object's e\_machine field will be EM\_SPARC.
- If any of the linked objects contain an e\_machine field of EM\_SPARC32PLUS, then the resulting linked object's e\_machine field will be EM\_SPARC32PLUS.

- If `e_machine` is `EM_SPARC32PLUS` then the `e_flags` `EF_SPARC_32PLUS` bit must also be set.
- Any other `e_machine` field value is an error.

When linking objects, `ld(1)` ORs all the `e_flags` fields of the linked objects to produce the resulting linked object's `e_flags` value.

### 2.2.4 Run-time Checking

When executing a binary, the kernel and the runtime linker perform validity checking of the processor-specific ELF header fields. The kernel provides the runtime linker with information on this platform's V8+ binary support by way of the `AT_FLAGS` auxiliary vector. The kernel will construct the `AT_FLAGS` information based on the capabilities of the platform's `cpu` module(s), using the same bits as in `e_flags`.

The kernel checks the processor-specific fields of the ELF header. If `e_machine` is `EM_SPARC32PLUS` then `EF_SPARC_32PLUS` must be set in `e_flags`. If `e_machine` is `EM_SPARC32PLUS` then the only acceptable set `e_flags` bits within `EF_SPARC_32PLUS_MASK` are those allowed based on this platform's current `cpu` module support.

The runtime linker performs checking similar to the kernel, using the `AT_FLAGS` information to validate any dynamically linked objects.

### 2.2.5 Relocation Types

Compiler enhancements to take advantage of SPARC V9 and UltraSPARC via supporting additional instructions requires the usage of new relocation types. These new relocation types are generated by the assembler and supported by the linkers. The relocation types are described in *Linker and Libraries Guide*.

## 2.3 Changes to `core(4)`

A V8+ core file has the same structure as a V8 core file, with the addition of new `NOTE` types 4 and 5, described in the Solaris 2.5 (or later) `core(4)` man page.

## 2.4 Changes to `ucontext(5)`

V8+ adds a new mechanism for associating extra register state with the `ucontext` structure. This is accomplished by using entries in the `uc_mcontext` filler array. The following structure is put at the beginning of the `uc_mcontext` filler array, beginning at what used to be `uc_mcontext.filler[0]` (and is now `uc_mcontext.xrs`).

```
typedef struct {
    unsigned int xrs_id;
    caddr_t xrs_ptr;
} xrs_t;
```

If the `xrs_id` field is equal to the string “xrs”, in other words 0x78727300, then `xrs_ptr` contains a valid pointer to a `prxregset_t` (see `/usr/include/sys/procfs.h`).

## 2.5 Changes to `proc(4)`

An executing V8+ object has additional register state consisting of additional floating-point registers, all 64 bits of the global and out registers, and the address space identifier registers. This state may be examined with `proc(4)` commands `PIOCGXREGSIZE`, `PIOCGXREG`, `PIOCSXREG`, described in the Solaris 2.5 (or later) `proc(4)` man page.

## *UltraSPARC-I-Specific Changes*

---



This appendix discusses UltraSPARC-I-specific changes made to V8 for V8+.

UltraSPARC defines user-accessible registers GSR, PCR, and PIC. GSR is used by certain of the UltraSPARC Graphics Instructions. PCR and PIC are performance-monitoring registers. They are all accessed by way of RDASR and WRASR instructions.

`%gsr` is volatile across function calls, plays no role in the calling sequence, and may be used by a called function without saving its value before changing it and without restoring its value before returning. Its initial state is unspecified.

`%pcr` and `%pic` are reserved.

The `pr_filler` array inside the `prxregset_t` structure is used to hold platform-dependent extra register state. For UltraSPARC-based platforms, `pr_filler[0]` contains the graphics state register.

`sysinfo(2)` can be invoked with the command `SI_PLATFORM` to identify the current platform, so that a program can determine how to interpret the contents of the `pr_filler` array.



# Index

---

## A

ASI, 2-4  
%asi, 2-3, 2-4  
ASI\_PRIMARY\_NOFAULT, 2-3, 2-4  
ASR, 2-2  
AT\_FLAGS, 2-7  
AT\_FLAGS auxiliary vector, 2-7

## B

BPcc, 2-2  
BPr, 2-2

## C

C long long, 2-3  
CASXA, 2-2  
CCR, 2-4  
%ccr, 2-3, 2-4  
core(4), 2-7

## D

%d32, 2-3, 2-4  
%d62, 2-3, 2-4  
DDX, vii, 1-2

## E

e\_ident, 2-5  
e\_ident[EI\_CLASS], 2-5  
e\_machine, 2-7  
EF\_SPARC\_32PLUS\_MASK, 2-5, 2-7  
EF\_SPARC\_HAL\_R1, 2-6  
EF\_SPARC\_SUN\_US1, 2-6  
[EI\_DATA], 2-5  
ELF header fields  
    processor-specific  
        validity checking of, 2-7  
ELFCLASS32, 2-5  
EM\_SPARC, 2-6  
EM\_SPARC32PLUS, 2-5, 2-6  
extra register state  
    platform-dependent, A-1

## F

%f32, 2-4  
%f63, 2-4  
FMOVcc, 2-2  
FMOVr, 2-2  
%fprs, 2-3  
function register volatility, 2-3

---

function registers with unassigned  
roles, 2-4

## G

`%g`, 2-2  
`%g5`, 2-3, 2-4  
GSR, A-1  
`%gsr`, A-1

## L

LD, 2-3  
`ld(1)`, 2-6, 2-7  
LDD, 2-3  
LDX, 2-2, 2-3  
LDXA, 2-2  
link editor, 2-6

## M

`MOVcc`, 2-2  
`MOVr`, 2-2  
MULX, 2-2

## N

NOTE type, 2-7

## O

`%o`, 2-2  
object files, 2-4

## P

PCR, A-1  
`%pcr`, A-1  
PIC, A-1  
`%pic`, A-1  
PIOGXREG, 2-8  
PIOGXREGSIZE, 2-8  
PIOCSXREG, 2-8  
POPC, 2-2

processor identification, 2-5  
processor-specific  
  opcodes, 2-3  
`prx_filler` array, A-1  
PSR (Platform-Specific Routines), 1-1

## Q

`%q32`, 2-3  
`%q60`, 2-3

## R

`rd`, 2-2  
RDASR, 2-2, A-1  
RDTICK, 2-2  
`rs1`, 2-2, 2-3  
`rs2`, 2-2, 2-3

## S

SDIVX, 2-2  
SI\_PLATFORM, A-1  
SLLX, 2-2  
SRAX, 2-2, 2-3  
SRIX, 2-3  
SRLX, 2-2  
ST, 2-3  
STD, 2-3  
STX, 2-2, 2-3  
STXA, 2-2  
`<sys/elf_SPARC.h>`, 2-5  
`sysinfo(2)`, A-1

## T

TCC, 2-2  
`%tick`, 2-3, 2-4

## U

`uc_mcontext` filler array, 2-8  
`uc_mcontext.filler[0]`, 2-8

---

ucontext(5), 2-8  
user-accessible registers, A-1

## V

V8 instructions  
  non-privileged, 2-1  
-v8plus, 2-6  
-v8plus flag, 2-6  
-v8plus\_hal\_r1, 2-6  
-v8plus\_hal\_r1 flag, 2-6  
-v8plus\_sun\_us1, 2-6  
V9 instructions  
  non-privileged, 2-1

## W

WRASR, 2-3, A-1

## X

%xcc, 2-2  
xrs\_id field, 2-8  
xrs\_ptr, 2-8

