Arm® Cortex®-A55 Core Advanced SIMD and Floating-point Support

Revision: r2p0

Technical Reference Manual

Arm® Cortex®-A55 Core Advanced SIMD and Floating-point Support

Technical Reference Manual

Release Information

Document History

Issue	Date	Confidentiality	Change
0000-00	30 September 2016	Confidential	First release for r0p0
0001-00	16 December 2016	Confidential	First release for r0p1
0100-00	23 June 2017	Non-Confidential	First release for r1p0
0100-01	15 December 2017	Non-Confidential	Second release for r1p0
0200-00	30 November 2018	Non-Confidential	First release for r2p0

Product Status

The information in this document is Final, that is for a developed product.

Web Address

http://www.arm.com

Contents

Arm® Cortex®-A55 Core Advanced SIMD and Floating-point Support Technical Reference Manual

Preface

This preface introduces the *Arm® Cortex®-A55 Core Advanced SIMD and Floating-point Support Technical Reference Manual*.

It contains the following:
- *About this book* on page 7.
- *Feedback* on page 9.

About this book

This book is for the Cortex-A55 core Advanced SIMD and floating-point support.

Product revision status

The r*m*p*n* identifier indicates the revision status of the product described in this book, for example, r1p2, where:

r*m*	Identifies the major revision of the product, for example, r1.
p*n*	Identifies the minor revision or modification status of the product, for example, p2.

Intended audience

This manual is for system designers, system integrators, and programmers who are designing or programming a *System-on-Chip* (SoC) that uses the Cortex®-A55 core with the optional Advanced SIMD and floating-point support.

Using this book

This book is organized into the following chapters:

Chapter 1 Functional description
This chapter introduces the optional Advanced SIMD and floating-point support.

Chapter 2 AArch64 Register Descriptions
This chapter describes the AArch64 registers for the Cortex-A55 core Advanced SIMD and floating-point support.

Chapter 3 AArch32 Register Descriptions
This chapter describes the AArch32 registers for the Cortex-A55 core Advanced SIMD and floating-point support.

Appendix A Revisions
This appendix describes the technical changes between released issues of this book.

Glossary

The Arm® Glossary is a list of terms used in Arm documentation, together with definitions for those terms. The Arm Glossary does not contain terms that are industry standard unless the Arm meaning differs from the generally accepted meaning.

See the *Arm® Glossary* for more information.

Typographic conventions

italic
Introduces special terminology, denotes cross-references, and citations.

bold
Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.

`monospace`
Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.

`monospace` (underlined: mono)
Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.

`monospace italic`
Denotes arguments to monospace text where the argument is to be replaced by a specific value.

`monospace bold`

Denotes language keywords when used outside example code.

`<and>`

Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example:

```
MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2>
```

SMALL CAPITALS

Used in body text for a few terms that have specific technical meanings, that are defined in the *Arm® Glossary*. For example, IMPLEMENTATION DEFINED, IMPLEMENTATION SPECIFIC, UNKNOWN, and UNPREDICTABLE.

Additional reading

This book contains information that is specific to this product. See the following documents for other relevant information.

Arm publications

- *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* (DDI 0487)
- *Arm® Cortex®-A55 Core Technical Reference Manual* (100442)

The following confidential documents are only available to licensees:

- *Arm® Cortex®-A55 Core Integration Manual* (100445)
- *Arm® Cortex®-A55 Core Configuration and Sign-off Guide* (100443)
- *Arm® Cortex®-A55 Core Cryptographic Extension Technical Reference Manual* (100444)

Other publications

- ANSI/IEEE Std 754-2008, IEEE Standard for Floating-Point Arithmetic.

Feedback

Feedback on this product

If you have any comments or suggestions about this product, contact your supplier and give:

- The product name.
- The product revision or version.
- An explanation with as much information as you can provide. Include symptoms and diagnostic procedures if appropriate.

Feedback on content

If you have comments on content then send an e-mail to *errata@arm.com*. Give:

- The title *Arm Cortex-A55 Core Advanced SIMD and Floating-point Support Technical Reference Manual*.
- The number 100446_0200_00_en.
- If applicable, the page number(s) to which your comments refer.
- A concise explanation of your comments.

Arm also welcomes general suggestions for additions and improvements.

Note

Arm tests the PDF only in Adobe Acrobat and Acrobat Reader, and cannot guarantee the quality of the represented document when used with any other PDF reader.

Chapter 1
Functional description

This chapter introduces the optional Advanced SIMD and floating-point support.

It contains the following section:

1.1 About the Advanced SIMD and floating-point support

The Cortex-A55 core supports the Advanced SIMD and scalar floating-point instructions in the A64 instruction set and the Advanced SIMD and floating-point instructions in the A32 and T32 instruction sets.

The Cortex-A55 floating-point implementation:

- Does not generate floating-point exceptions.
- Implements all scalar operations in hardware with support for all combinations of:
  - Rounding modes.
  - Flush-to-zero.
  - Default *Not a Number* (NaN) modes.

The Armv8-A architecture does not define a separate version number for its Advanced SIMD and floating-point support in the AArch64 execution state because the instructions are always implicitly present.

Chapter 2
AArch64 Register Descriptions

This chapter describes the AArch64 registers for the Cortex-A55 core Advanced SIMD and floating-point support.

It contains the following sections:

2.1 Accessing the AArch64 feature identification registers

Software can identify the Advanced SIMD and floating-point features using the feature identification registers in the AArch64 execution state.

You can access the feature identification registers in the AArch64 execution state using the `MRS` instruction, for example:

```
        MRS <Xt>, ID_AA64PFR0_EL1 ; Read ID_AA64PFR0_EL1 into Xt
        MRS <Xt>, MVFR0_EL1       ; Read MVFR0_EL1 into Xt
        MRS <Xt>, MVFR1_EL1       ; Read MVFR1_EL1 into Xt
        MRS <Xt>, MVFR2_EL1       ; Read MVFR2_EL1 into Xt
```

Table 2-1 AArch64 Advanced SIMD and scalar floating-point feature identification registers

AArch64 name	Description
ID_AA64PFR0_EL1	Gives additional information about implemented core features in AArch64. See the *Arm® Cortex®-A55 Core Technical Reference Manual*.
MVFR0_EL1	See *2.5 MVFR0_EL1, Media and VFP Feature Register 0, EL1* on page 2-19.
MVFR1_EL1	See *2.6 MVFR1_EL1, Media and VFP Feature Register 1, EL1* on page 2-21.
MVFR2_EL1	See *2.7 MVFR2_EL1, Media and VFP Feature Register 2, EL1* on page 2-23.

2.2 AArch64 register summary

The core has several Advanced SIMD and floating-point system registers in the AArch64 execution state. Each register has a specific purpose, specific usage constraints, configurations, and attributes.

The following table gives a summary of the Cortex-A55 core Advanced SIMD and floating-point system registers in the AArch64 execution state.

Table 2-2 AArch64 Advanced SIMD and floating-point system registers

Name	Type	Reset	Description
FPCR	RW	`0x00000000`	See *2.3 FPCR, Floating-point Control Register* on page 2-15.
FPSR	RW	`0x00000000`	See *2.4 FPSR, Floating-point Status Register* on page 2-17.
MVFR0_EL1	RO	`0x10110222`	See *2.5 MVFR0_EL1, Media and VFP Feature Register 0, EL1* on page 2-19.
MVFR1_EL1	RO	`0x13211111`	See *2.6 MVFR1_EL1, Media and VFP Feature Register 1, EL1* on page 2-21.
MVFR2_EL1	RO	`0x00000043`	See *2.7 MVFR2_EL1, Media and VFP Feature Register 2, EL1* on page 2-23.
FPEXC32_EL2	RW	`0x00000700`	See *2.8 FPEXC32_EL2, Floating-point Exception Control Register, EL2* on page 2-25.

2.3 FPCR, Floating-point Control Register

The FPCR controls floating-point behavior.

Bit field descriptions

FPCR is a 32-bit register.

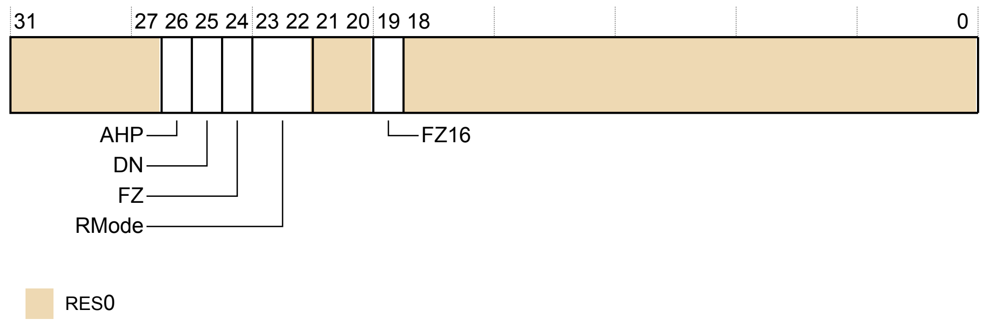

Figure 2-1 FPCR bit assignments

RES0, [31:27]

RES0	Reserved.

AHP, [26]

Alternative half-precision control bit. The possible values are:

`0`	IEEE half-precision format selected. This is the reset value.
`1`	Alternative half-precision format selected.

DN, [25]

Default NaN mode control bit. The possible values are:

`0`	NaN operands propagate through to the output of a floating-point operation. This is the reset value.
`1`	Any operation involving one or more NaNs returns the Default NaN.

FZ, [24]

Flush-to-zero mode control bit. The possible values are:

`0`	Flush-to-zero mode disabled. Behavior of the floating-point system is fully compliant with the IEEE 754 standard. This is the reset value.
`1`	Flush-to-zero mode enabled.

RMode, [23:22]

Rounding Mode control field. The encoding of this field is:

`0b00`	*Round to Nearest* (RN) mode. This is the reset value.
`0b01`	*Round towards Plus Infinity* (RP) mode.
`0b10`	*Round towards Minus Infinity* (RM) mode.
`0b11`	*Round towards Zero* (RZ) mode.

RES0, [21:20]

RES0	Reserved.

FZ16, [19]

Flush-to-zero mode control bit on half-precision data-processing instructions. The possible values are:

`0`	Flush-to-zero mode disabled. Behavior of the floating-point system is fully compliant with the IEEE 754 standard. This is the default value.
`1`	Flush-to-zero mode enabled.

RES0, [18:0]

RES0	Reserved.

Configurations

The named fields in this register map to the equivalent fields in the AArch32 FPSCR. See *3.4 FPSCR, Floating-Point Status and Control Register* on page 3-32.

Usage constraints

Accessing the FPCR

To access the FPCR:

```
MRS <Xt>, FPCR ; Read FPCR into Xt
MSR FPCR, <Xt> ; Write Xt to FPCR
```

Register access is encoded as follows:

Table 2-3 FPCR access encoding

op0	op1	CRn	CRm	op2
11	011	0100	0100	000

Accessibility

This register is accessible as follows:

EL0	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
RW	RW	RW	RW	RW	RW

2.4 FPSR, Floating-point Status Register

The FPSR provides floating-point system status information.

Bit field descriptions

FPSR is a 32-bit register.

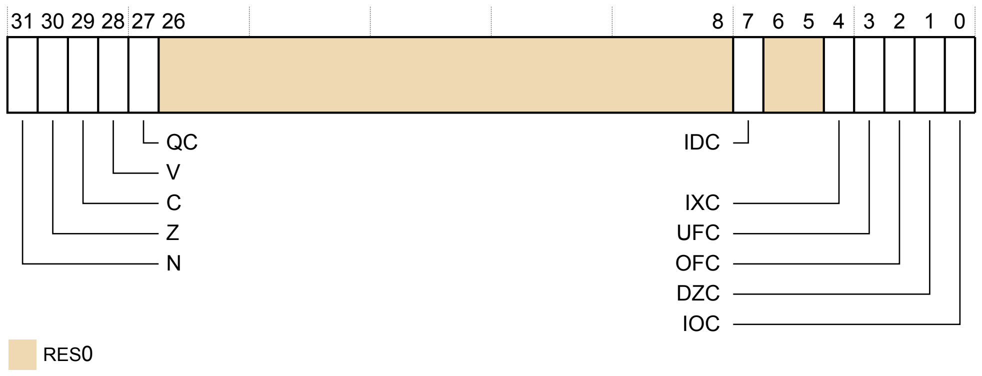

Figure 2-2 FPSR bit assignments

N, [31]

Negative condition flag for AArch32 floating-point comparison operations. AArch64 floating-point comparisons set the PSTATE.N flag instead.

Z, [30]

Zero condition flag for AArch32 floating-point comparison operations. AArch64 floating-point comparisons set the PSTATE.Z flag instead.

C, [29]

Carry condition flag for AArch32 floating-point comparison operations. AArch64 floating-point comparisons set the PSTATE.C flag instead

V, [28]

Overflow condition flag for AArch32 floating-point comparison operations. AArch64 floating-point comparisons set the PSTATE.V flag instead.

QC, [27]

Cumulative saturation bit. This bit is set to 1 to indicate that an Advanced SIMD integer operation has saturated since a 0 was last written to this bit.

RES0, [26:8]

Reserved, RES0.

IDC, [7]

Input Denormal cumulative exception bit. This bit is set to 1 to indicate that the Input Denormal exception has occurred since 0 was last written to this bit.

RES0, [6:5]

Reserved, RES0.

IXC, [4]

Inexact cumulative exception bit. This bit is set to 1 to indicate that the Inexact exception has occurred since 0 was last written to this bit.

UFC, [3]

Underflow cumulative exception bit. This bit is set to 1 to indicate that the Underflow exception has occurred since 0 was last written to this bit.

OFC, [2]

Overflow cumulative exception bit. This bit is set to 1 to indicate that the Overflow exception has occurred since 0 was last written to this bit.

DZC, [1]

Division by Zero cumulative exception bit. This bit is set to 1 to indicate that the Division by Zero exception has occurred since 0 was last written to this bit.

IOC, [0]

Invalid Operation cumulative exception bit. This bit is set to 1 to indicate that the Invalid Operation exception has occurred since 0 was last written to this bit.

Configurations

The named fields in this register map to the equivalent fields in the AArch32 FPSCR. See *3.4 FPSCR, Floating-Point Status and Control Register* on page 3-32.

Usage constraints

Accessing the FPSR

To access the FPSR:

```
MRS <Xt>, FPSR; Read FPSR into Xt
MSR FPSR, <Xt>; Write Xt to FPSR
```

Register access is encoded as follows:

Table 2-4 FPSR access encoding

op0	op1	CRn	CRm	op2
11	011	0100	0100	001

Accessibility

This register is accessible as follows:

EL0	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
RW	RW	RW	RW	RW	RW

2.5 MVFR0_EL1, Media and VFP Feature Register 0, EL1

The MVFR0_EL1 describes the features provided by the AArch64 Advanced SIMD and floating-point implementation.

Bit field descriptions

MVFR0_EL1 is a 32-bit register.

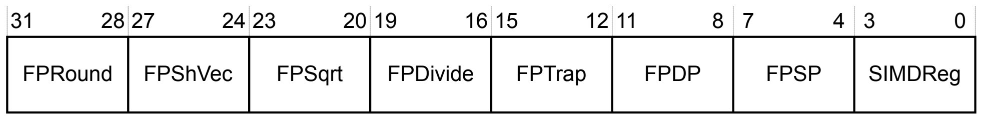

Figure 2-3 MVFR0_EL1 bit assignments

FPRound, [31:28]

Indicates the rounding modes supported by the floating-point hardware:

`0x1`	All rounding modes supported.

FPShVec, [27:24]

Indicates the hardware support for floating-point short vectors:

`0x0`	Not supported.

FPSqrt, [23:20]

Indicates the hardware support for floating-point square root operations:

`0x1`	Supported.

FPDivide, [19:16]

Indicates the hardware support for floating-point divide operations:

`0x1`	Supported.

FPTrap, [15:12]

Indicates whether the floating-point hardware implementation supports exception trapping:

`0x0`	Not supported.

FPDP, [11:8]

Indicates the hardware support for floating-point double-precision operations:

`0x2`	Supported, VFPv3 or greater.

See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.

FPSP, [7:4]

Indicates the hardware support for floating-point single-precision operations:

`0x2`	Supported, VFPv3 or greater.

See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.

SIMDReg, [3:0]

Indicates support for the Advanced SIMD register bank:

`0x2` Supported, 32 x 64-bit registers supported.

See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.

Configurations

MVFR0_EL1 is architecturally mapped to AArch32 register MVFR0. See *3.5 MVFR0, Media and VFP Feature Register 0* on page 3-35.

Usage constraints

Accessing the MVFR0_EL1

To access the MVFR0_EL1:

```
MRS <Xt>, MVFR0_EL1 ; Read MVFR0_EL1 into Xt
```

Register access is encoded as follows:

Table 2-5 MVFR0_EL1 access encoding

op0	op1	CRn	CRm	op2
11	000	0000	0011	000

Accessibility

This register is accessible as follows:

EL0	EL1(NS)	EL1(S)	EL2	EL3 (SCR.NS = 1)	EL3(SCR.NS = 0)
-	RO	RO	RO	RO	RO

2.6 MVFR1_EL1, Media and VFP Feature Register 1, EL1

The MVFR1_EL1 describes the features provided by the AArch64 Advanced SIMD and floating-point implementation.

Bit field descriptions

MVFR1_EL1 is a 32-bit register.

31 28	27 24	23 20	19 16	15 12	11 8	7 4	3 0
SIMDFMAC	FPHP	SIMDHP	SIMDSP	SIMDInt	SIMDLS	FPDNaN	FPFtZ

Figure 2-4 MVFR1_EL1 bit assignments

SIMDFMAC, [31:28]

Indicates whether the Advanced SIMD and floating-point unit supports fused multiply accumulate operations:

`1`	Implemented.

FPHP, [27:24]

Indicates whether the Advanced SIMD and floating-point unit supports half-precision floating-point conversion instructions:

`3`	Floating-point half precision conversion and data processing instructions implemented.

SIMDHP, [23:20]

Indicates whether the Advanced SIMD and floating-point unit supports half-precision floating-point conversion operations:

`2`	Advanced SIMD half precision conversion and data processing instructions implemented.

SIMDSP, [19:16]

Indicates whether the Advanced SIMD and floating-point unit supports single-precision floating-point operations:

`1`	Implemented.

SIMDInt, [15:12]

Indicates whether the Advanced SIMD and floating-point unit supports integer operations:

`1`	Implemented.

SIMDLS, [11:8]

Indicates whether the Advanced SIMD and floating-point unit supports load/store instructions:

`1`	Implemented.

FPDNaN, [7:4]

Indicates whether the floating-point hardware implementation supports only the Default NaN mode:

`1`	Hardware supports propagation of NaN values.

FPFtZ, [3:0]

Indicates whether the floating-point hardware implementation supports only the Flush-to-zero mode of operation:

`1`	Hardware supports full denormalized number arithmetic.

Configurations

MVFR1_EL1 is architecturally mapped to AArch32 register MVFR1. See *3.6 MVFR1, Media and VFP Feature Register 1* on page 3-37.

Usage constraints

Accessing the MVFR1_EL1

To access the MVFR1_EL1:

```
MRS <Xt>, MVFR1_EL1 ; Read MVFR1_EL1 into Xt
```

Register access is encoded as follows:

Table 2-6 MVFR1_EL1 access encoding

op0	op1	CRn	CRm	op2
11	000	0000	0011	001

Accessibility

This register is accessible as follows:

EL0	EL1(NS)	EL1(S)	EL2	EL3 (SCR.NS = 1)	EL3(SCR.NS = 0)
-	RO	RO	RO	RO	RO

2.7 MVFR2_EL1, Media and VFP Feature Register 2, EL1

The MVFR2_EL1 describes the features provided by the AArch64 Advanced SIMD and floating-point implementation.

Bit field descriptions

MVFR2_EL1 is a 32-bit register.

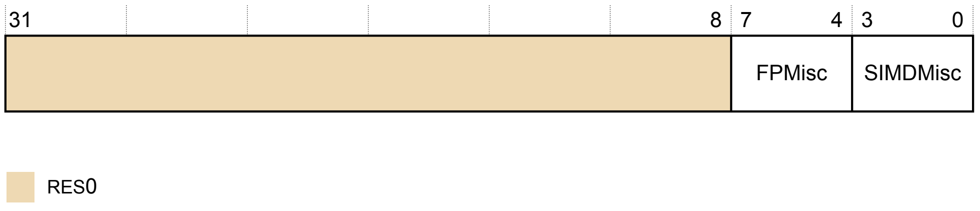

Figure 2-5 MVFR2_EL1 bit assignments

[31:8]

RES0 Reserved.

FPMisc, [7:4]

Indicates support for miscellaneous floating-point features.

`0x4` Supports:
- Floating-point selection.
- Floating-point Conversion to Integer with Directed Rounding modes.
- Floating-point Round to Integral Floating-point.
- Floating-point MaxNum and MinNum.

SIMDMisc, [3:0]

Indicates support for miscellaneous Advanced SIMD features.

`0x3` Supports:
- Floating-point Conversion to Integer with Directed Rounding modes.
- Floating-point Round to Integral Floating-point.
- Floating-point MaxNum and MinNum.

Configurations

MVFR2_EL1 is architecturally mapped to AArch32 register MVFR2. See *3.7 MVFR2, Media and VFP Feature Register 2* on page 3-39.

Usage constraints

Accessing the MVFR2_EL1

To access the MVFR2_EL1:

```
MRS <Xt>, MVFR2_EL1 ; Read MVFR2_EL1 into Xt
```

Register access is encoded as follows:

Table 2-7 MVFR2_EL1 access encoding

op0	op1	CRn	CRm	op2
11	000	0000	0011	010

Accessibility

This register is accessible as follows:

EL0	EL1(NS)	EL1(S)	EL2	EL3 (SCR.NS = 1)	EL3(SCR.NS = 0)
-	RO	RO	RO	RO	RO

2.8 FPEXC32_EL2, Floating-point Exception Control Register, EL2

The FPEXC32_EL2 provides access to the AArch32 register FPEXC from AArch64 state only. Its value has no effect on execution in AArch64 state.

Bit field descriptions

FPEXC32_EL2 is a 32-bit register.

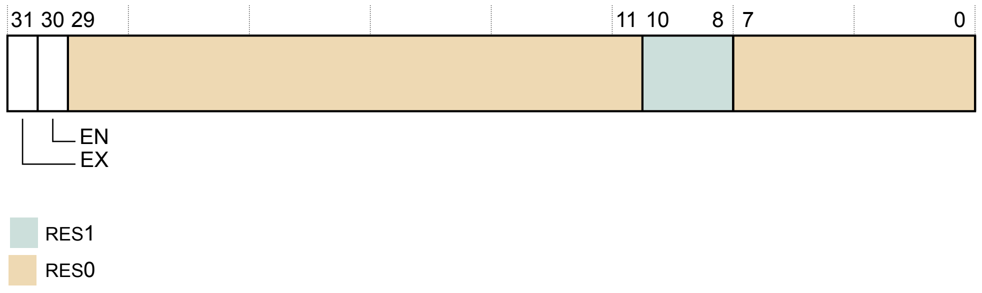

Figure 2-6 FPEXC32_EL2 bit assignments

EX, [31]

Exception bit.

RES0	The Cortex-A55 core implementation does not generate asynchronous floating-point exceptions.

EN, [30]

Enable bit. A global enable for the Advanced SIMD and floating-point support:

`0`	The Advanced SIMD and floating-point support is disabled. This is the reset value.
`1`	The Advanced SIMD and floating-point support is enabled and operates normally.

This bit applies only to AArch32 execution, and only when EL1 is not AArch64.

[29:11]

RES0 Reserved.

[10:8]

RES1 Reserved.

[7:0]

RES0 Reserved.

Configurations

FPEXC32_EL2 is architecturally mapped to AArch32 register FPEXC. See *3.8 FPEXC, Floating-Point Exception Control register* on page 3-41.

Usage constraints

Accessing the FPEXC32_EL2

To access the FPEXC32_EL2:

```
MRS <Xt>, FPEXC32_EL2 ; Read FPEXC32_EL2 into Xt
              MSR FPEXC32_EL2, <Xt> ; Write Xt to FPEXC32_EL2
```

Register access is encoded as follows:

Table 2-8 FPEXC32_EL2 access encoding

op0	op1	CRn	CRm	op2
11	100	0101	0011	000

Accessibility

This register is accessible as follows:

EL0	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	RW	RW	RW

Chapter 3
AArch32 Register Descriptions

This chapter describes the AArch32 registers for the Cortex-A55 core Advanced SIMD and floating-point support.

It contains the following sections:

3.1 Accessing the AArch32 feature identification registers

Software can identify the Advanced SIMD and floating-point features using the feature identification registers in the AArch32 execution state.

You can access the feature identification registers in the AArch32 execution state using the `VMRS` instruction, for example:

```
VMRS <Rt>, FPSID ; Read FPSID into Rt
VMRS <Rt>, MVFR0 ; Read MVFR0 into Rt
VMRS <Rt>, MVFR1 ; Read MFFR1 into Rt
VMRS <Rt>, MVFR2 ; Read MVFR2 into Rt
```

Table 3-1 AArch32 Advanced SIMD and scalar floating-point feature identification registers

AArch32 name	Description
FPSID	See *3.3 FPSID, Floating-Point System ID Register* on page 3-30.
MVFR0	See *3.5 MVFR0, Media and VFP Feature Register 0* on page 3-35.
MVFR1	See *3.6 MVFR1, Media and VFP Feature Register 1* on page 3-37.
MVFR2	See *3.7 MVFR2, Media and VFP Feature Register 2* on page 3-39.

3.2 AArch32 register summary

The core has several Advanced SIMD and floating-point system registers in the AArch32 execution state. Each register has a specific purpose, usage constraints, configurations, and attributes.

The following table gives a summary of the Cortex-A55 core Advanced SIMD and floating-point system registers in the AArch32 execution state.

Table 3-2 AArch32 Advanced SIMD and floating-point system registers

Name	Type	Reset	Description
FPSID	RO	`0x41034052`	See *3.3 FPSID, Floating-Point System ID Register* on page 3-30.
FPSCR	RW	`0x00000000`	See *3.4 FPSCR, Floating-Point Status and Control Register* on page 3-32.
MVFR0	RO	`0x10110222`	See *3.5 MVFR0, Media and VFP Feature Register 0* on page 3-35.
MVFR1	RO	`0x13211111`	See *3.6 MVFR1, Media and VFP Feature Register 1* on page 3-37.
MVFR2	RO	`0x00000043`	See *3.7 MVFR2, Media and VFP Feature Register 2* on page 3-39.
FPEXC	RW	`0x00000700`	See *3.8 FPEXC, Floating-Point Exception Control register* on page 3-41.

Note

The *Floating-Point Instruction Registers*, FPINST and FPINST2 are not implemented, and any attempt to access them is UNPREDICTABLE.

See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for information on permitted accesses to the Advanced SIMD and floating-point system registers.

3.3 FPSID, Floating-Point System ID Register

The FPSID provides top-level information about the floating-point implementation.

Bit field descriptions

FPSID is a 32-bit register.

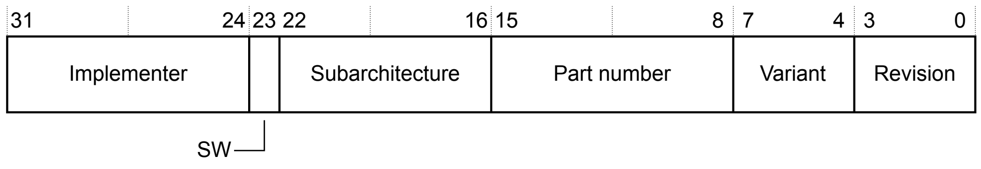

Figure 3-1 FPSID bit assignments

Implementer, [31:24]

Indicates the implementer:

`0x41`	Arm Limited

SW, [23]

Software bit. This bit indicates that a system provides only software emulation of the floating-point instructions:

`0`	The system includes hardware support for floating-point operations.

Subarchitecture, [22:16]

Subarchitecture version number:

`0x03`	VFPv3 architecture, or later, with no subarchitecture. The entire floating-point implementation is in hardware, and requires no software support code. The MVFR0, MVFR1, and MVFR2 registers indicate the VFP architecture version.

Part number, [15:8]

Indicates the part number for the floating-point implementation:

`0x40`	v8-A profile.

Variant, [7:4]

Indicates the variant number:

`5`	Cortex-A55 core.

Revision, [3:0]

Indicates the revision number for the floating-point implementation:

`3`	r2p0.

Configurations

Access to this register depends on the values of CPACR.{cp10,cp11}, NSACR.{cp10,cp11}, and HCPTR.{TCP10,TCP11}. For details of which field values permit access at specific Exception levels, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

This register largely duplicates information that is held in the MIDR. Arm deprecates use of it.

Usage constraints

Accessing the FPSID

To access the FPSID:

```
VMRS <Rt>, FPSID ; Read FPSID into Rt
```

Register access is encoded as follows:

Table 3-3 FPSID access encoding

spec_reg
0000

Accessibility

This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	Config	RO	Config	Config	RO

3.4 FPSCR, Floating-Point Status and Control Register

The FPSCR provides floating-point system status information and control.

Bit field descriptions

FPSCR is a 32-bit register.

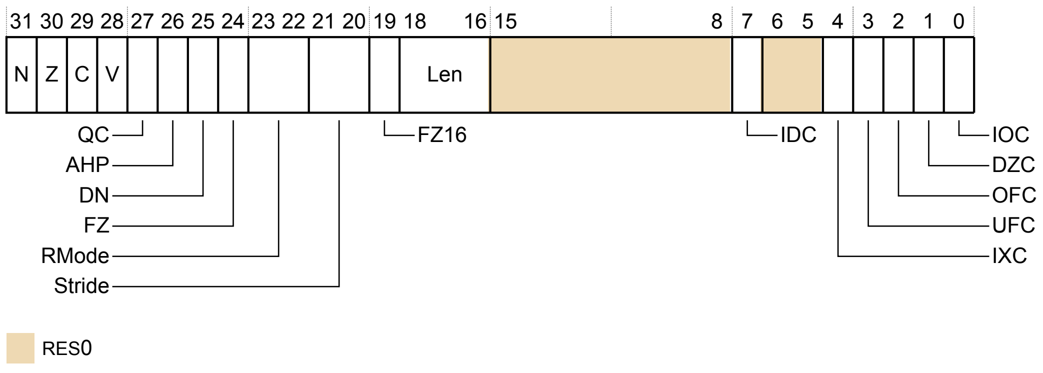

Figure 3-2 FPSCR bit assignments

N, [31]

Floating-point Negative condition code flag.

Set to 1 if a floating-point comparison operation produces a less than result.

Z, [30]

Floating-point Zero condition code flag.

Set to 1 if a floating-point comparison operation produces an equal result.

C, [29]

Floating-point Carry condition code flag.

Set to 1 if a floating-point comparison operation produces an equal, greater than, or unordered result.

V, [28]

Floating-point Overflow condition code flag.

Set to 1 if a floating-point comparison operation produces an unordered result.

QC, [27]

Cumulative saturation bit.

This bit is set to 1 to indicate that an Advanced SIMD integer operation has saturated after 0 was last written to this bit.

AHP, [26]

Alternative Half-Precision control bit:

`0`	IEEE half-precision format selected. This is the reset value.
`1`	Alternative half-precision format selected.

DN, [25]

Default NaN mode control bit:

`0`	NaN operands propagate through to the output of a floating-point operation. This is the reset value.
`1`	Any operation involving one or more NaNs returns the Default NaN.

The value of this bit only controls floating-point arithmetic. AArch32 Advanced SIMD arithmetic always uses the Default NaN setting, regardless of the value of the DN bit.

FZ, [24]

Flush-to-zero mode control bit:

`0`	Flush-to-zero mode disabled. Behavior of the floating-point system is fully compliant with the IEEE 754 standard. This is the reset value.
`1`	Flush-to-zero mode enabled.

The value of this bit only controls floating-point arithmetic. AArch32 Advanced SIMD arithmetic always uses the Flush-to-zero setting, regardless of the value of the FZ bit.

RMode, [23:22]

Rounding Mode control field:

`0b00`	*Round to Nearest* (RN) mode. This is the reset value.
`0b01`	*Round towards Plus Infinity* (RP) mode.
`0b10`	*Round towards Minus Infinity* (RM) mode.
`0b11`	*Round towards Zero* (RZ) mode.

The specified rounding mode is used by almost all floating-point instructions. AArch32 Advanced SIMD arithmetic always uses the Round to Nearest setting, regardless of the value of the RMode bits.

Stride, [21:20]

RES0	Reserved.

FZ16, [19]

Flush-to-zero mode control bit on half-precision data-processing instructions:

`0`	Flush-to-zero mode disabled. Behavior of the floating-point system is fully compliant with the IEEE 754 standard.
`1`	Flush-to-zero mode enabled.

Len, [18:16]

RES0	Reserved.

RES0, [15:8]

RES0	Reserved.

IDC, [7]

Input Denormal cumulative exception bit. This bit is set to 1 to indicate that the Input Denormal exception has occurred since 0 was last written to this bit.

RES0, [6:5]

RES0	Reserved.

IXC, [4]

Inexact cumulative exception bit. This bit is set to 1 to indicate that the Inexact exception has occurred since 0 was last written to this bit.

UFC, [3]

Underflow cumulative exception bit. This bit is set to 1 to indicate that the Underflow exception has occurred since 0 was last written to this bit.

OFC, [2]

Overflow cumulative exception bit. This bit is set to 1 to indicate that the Overflow exception has occurred since 0 was last written to this bit.

DZC, [1]

Division by Zero cumulative exception bit. This bit is set to 1 to indicate that the Division by Zero exception has occurred since 0 was last written to this bit.

IOC, [0]

Invalid Operation cumulative exception bit. This bit is set to 1 to indicate that the Invalid Operation exception has occurred since 0 was last written to this bit.

Configurations

There is one copy of this register that is used in both Secure and Non-secure states.

The named fields in this register map to the equivalent fields in the AArch64 FPCR and FPSR. See *2.3 FPCR, Floating-point Control Register* on page 2-15 and *2.4 FPSR, Floating-point Status Register* on page 2-17

.

Usage constraints

Accessing the FPSCR

To access the FPSCR:

```
VMRS <Rt>, FPSCR ; Read FPSCR into Rt
VMSR FPSCR, <Rt> ; Write Rt to FPSCR
```

Register access is encoded as follows:

Table 3-4 FPSCR access encoding

spec_reg
0001

Note

The Cortex-A55 core implementation does not support the deprecated VFP short vector feature. Attempts to execute the associated VFP data-processing instructions result in an UNDEFINED Instruction exception.

Accessibility

This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
Config	RW	Config	RW	Config	Config	RW

Access to this register depends on the values of CPACR.{cp10,cp11}, NSACR.{cp10,cp11}, HCPTR.{TCP10,TCP11} and FPEXC.EN. For details of which values of these fields allow access at which Exception levels, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

3.5 MVFR0, Media and VFP Feature Register 0

The MVFR0 describes the features provided by the AArch32 Advanced SIMD and floating-point implementation.

Bit field descriptions

MVFR0 is a 32-bit register.

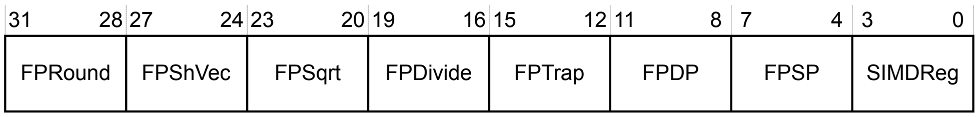

Figure 3-3 MVFR0 bit assignments

FPRound, [31:28]

Indicates the rounding modes supported by the floating-point hardware:

`0x1` All rounding modes supported.

FPShVec, [27:24]

Indicates the hardware support for floating-point short vectors:

`0x0` Not supported.

FPSqrt, [23:20]

Indicates the hardware support for floating-point square root operations:

`0x1` Supported.

FPDivide, [19:16]

Indicates the hardware support for floating-point divide operations:

`0x1` Supported.

FPTrap, [15:12]

Indicates whether the floating-point hardware implementation supports exception trapping:

`0x0` Not supported.

FPDP, [11:8]

Indicates the hardware support for floating-point double-precision operations:

`0x2` Supported, VFPv3, or greater.

See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.

FPSP, [7:4]

Indicates the hardware support for floating-point single-precision operations:

`0x2` Supported, VFPv3, or greater.

See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.

SIMDReg, [3:0]

Indicates support for the Advanced SIMD register bank:

`0x2`	Supported, 32 x 64-bit registers supported.

See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.

Configurations

MVFR0 is architecturally mapped to AArch64 register MVFR0_EL1. See *2.5 MVFR0_EL1, Media and VFP Feature Register 0, EL1* on page 2-19.

There is one copy of this register that is used in both Secure and Non-secure states.

Usage constraints

Accessing the MVFR0

To access the MVFR0:

```
VMRS <Rt>, MVFR0 ; Read MVFR0 into Rt
```

Register access is encoded as follows:

Table 3-5 MVFR0 access encoding

spec_reg
0111

Accessibility

This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	Config	RO	Config	Config	RO

Access to this register depends on the values of CPACR.{cp10,cp11}, NSACR.{cp10,cp11}, HCPTR.{TCP10,TCP11}, and FPEXC.EN. For details of which values of these fields allow access at which Exception levels, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

MVFR0 must be interpreted with MVFR1 and MVFR2. See *3.6 MVFR1, Media and VFP Feature Register 1* on page 3-37 and *3.7 MVFR2, Media and VFP Feature Register 2* on page 3-39.

3.6 MVFR1, Media and VFP Feature Register 1

The MVFR1 describes the features provided by the AArch32 Advanced SIMD and floating-point implementation.

Bit field descriptions

MVFR1 is a 32-bit register.

31 28	27 24	23 20	19 16	15 12	11 8	7 4	3 0
SIMDFMAC	FPHP	SIMDHP	SIMDSP	SIMDInt	SIMDLS	FPDNaN	FPFtZ

Figure 3-4 MVFR1 bit assignments

SIMDFMAC, [31:28]

Indicates whether the Advanced SIMD and floating-point unit supports fused multiply accumulate operations:

`0x1`	Implemented.

FPHP, [27:24]

Indicates whether the Advanced SIMD and floating-point unit supports half-precision floating-point conversion instructions:

`0x3`	Floating-point half precision conversion and data processing instructions implemented.

SIMDHP, [23:20]

Indicates whether the Advanced SIMD and floating-point unit supports half-precision floating-point conversion operations:

`0x2`	Advanced SIMD precision conversion and data processing instructions implemented.

SIMDSP, [19:16]

Indicates whether the Advanced SIMD and floating-point unit supports single-precision floating-point operations:

`0x1`	Implemented.

SIMDInt, [15:12]

Indicates whether the Advanced SIMD and floating-point unit supports integer operations:

`0x1`	Implemented.

SIMDLS, [11:8]

Indicates whether the Advanced SIMD and floating-point unit supports load/store instructions:

`0x1`	Implemented.

FPDNaN, [7:4]

Indicates whether the floating-point hardware implementation supports only the Default NaN mode:

`0x1`	Hardware supports propagation of NaN values.

FPFtZ, [3:0]

Indicates whether the floating-point hardware implementation supports only the Flush-to-zero mode of operation:

`0x1`	Hardware supports full denormalized number arithmetic.

Configurations

MVFR1 is architecturally mapped to AArch64 register MVFR1_EL1. See *2.6 MVFR1_EL1, Media and VFP Feature Register 1, EL1* on page 2-21.

There is one copy of this register that is used in both Secure and Non-secure states.

Usage constraints

Accessing the MVFR1

To access the MVFR1:

```
VMRS <Rt>, MVFR1 ; Read MVFR1 into Rt
```

Register access is encoded as follows:

Table 3-6 MVFR1 access encoding

spec_reg
0110

Accessibility

This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	Config	RO	Config	Config	RO

Access to this register depends on the values of CPACR.{cp10,cp11}, NSACR.{cp10,cp11}, HCPTR.{TCP10,TCP11}, and FPEXC.EN. For details of which values of these fields allow access at which Exception levels, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

MVFR1 must be interpreted with MVFR0 and MVFR2. See *3.5 MVFR0, Media and VFP Feature Register 0* on page 3-35 and *3.7 MVFR2, Media and VFP Feature Register 2* on page 3-39.

3.7 MVFR2, Media and VFP Feature Register 2

The MVFR2 describes the features provided by the AArch32 Advanced SIMD and floating-point implementation.

Bit field descriptions

MVFR2 is a 32-bit register.

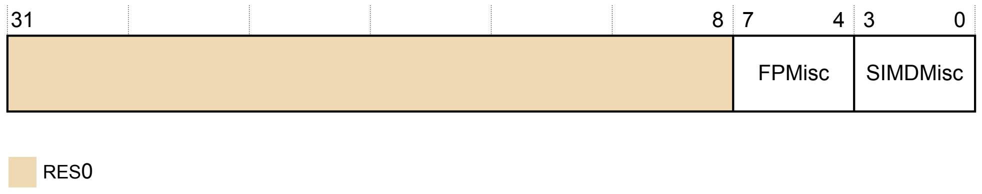

Figure 3-5 MVFR2 bit assignments

[31:8]

RES0 Reserved.

FPMisc, [7:4]

Indicates support for miscellaneous VFP features.

`0x4` Supports:

- Floating-point selection.
- Floating-point Conversion to Integer with Directed Rounding modes.
- Floating-point Round to Integral Floating-point.
- Floating-point MaxNum and MinNum.

SIMDMisc, [3:0]

Indicates support for miscellaneous Advanced SIMD features.

`0x3` Supports:

- Floating-point Conversion to Integer with Directed Rounding modes.
- Floating-point Round to Integral Floating-point.
- Floating-point MaxNum and MinNum.

Configurations

MVFR2 is architecturally mapped to AArch64 register MVFR2_EL1. See *2.7 MVFR2_EL1, Media and VFP Feature Register 2, EL1* on page 2-23.

There is one copy of this register that is used in both Secure and Non-secure states.

Usage constraints

Accessing the MVFR2

To access the MVFR2:

```
VMRS <Rt>, MVFR2 ; Read MVFR2 into Rt
```

Register access is encoded as follows:

Table 3-7 MVFR2 access encoding

spec_reg
0101

Accessibility

This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	Config	RO	Config	Config	RO

Access to this register depends on the values of CPACR.{cp10,cp11}, NSACR.{cp10,cp11}, HCPTR.{TCP10,TCP11}, and FPEXC.EN. For details of which values of these fields allow access at which Exception levels, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

MVFR2 must be interpreted with MVFR0 and MVFR1. See *3.5 MVFR0, Media and VFP Feature Register 0* on page 3-35 and *3.6 MVFR1, Media and VFP Feature Register 1* on page 3-37.

3.8 FPEXC, Floating-Point Exception Control register

The FPEXC provides a global enable for the Advanced SIMD and floating-point support, and indicates how the state of this support is recorded.

Bit field descriptions

FPEXC is a 32-bit register.

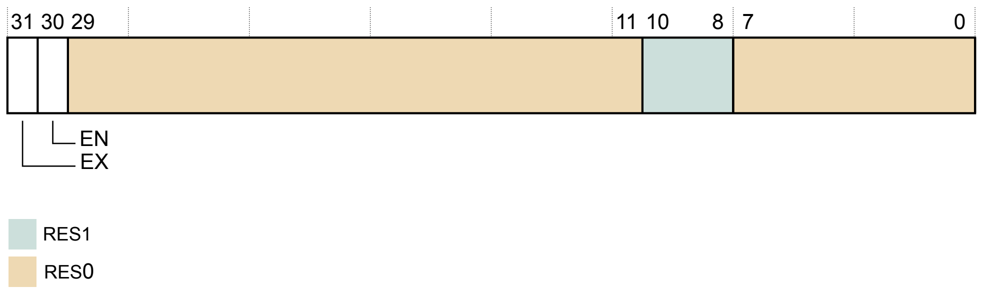

Figure 3-6 FPEXC bit assignments

EX, [31]

Exception bit. The Cortex-A55 core implementation does not generate asynchronous floating-point exceptions, therefore this bit is RES0.

EN, [30]

Global enable for the Advanced SIMD and floating-point support:

`0` The Advanced SIMD and floating-point support is disabled. This is the reset value.

`1` The Advanced SIMD and floating-point support is enabled and operates normally.

It applies only to AArch32 executions, and only when EL1 is not AArch64.

[29:11]

RES0 Reserved.

[10:8]

RES1 Reserved.

[7:0]

RES0 Reserved.

Configurations

FPEXC is architecturally mapped to AArch64 register FPEXC32_EL2. See *2.8 FPEXC32_EL2, Floating-point Exception Control Register, EL2* on page 2-25.

There is one copy of this register that is used in both Secure and Non-secure states.

Usage constraints

Accessing the FPEXC

To access the FPEXC register:

```
VMRS <Rt>, FPEXC ; Read FPEXC into Rt
```

Register access is encoded as follows:

Table 3-8 FPEXC access encoding

spec_reg
1000

Accessibility

This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	Config	RW	Config	Config	RW

Access to this register depends on the values of CPACR.{cp10,cp11}, NSACR.{cp10,cp11}, and HCPTR.{TCP10,TCP11}. For details of which values of these fields allow access at which Exception levels, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Appendix A
Revisions

This appendix describes the technical changes between released issues of this book.

It contains the following section:

A.1 Revisions

This appendix describes the technical changes between released issues of this book.

Table A-1 Issue 0000-00

Change	Location	Affects
First release	-	-

Table A-2 Differences between issue 0000-00 and issue 0001-00

Change	Location	Affects
Changed revision to r0p1	*3.3 FPSID, Floating-Point System ID Register* on page 3-30	r0p1

Table A-3 Differences between issue 0001-00 and issue 0100-00

Change	Location	Affects
Changed revision to r1p0	*3.3 FPSID, Floating-Point System ID Register* on page 3-30	r1p0
Updated product name	-	r1p0
Global terminology change from 'processor' to 'core' for the product.	-	r1p0

Table A-4 Differences between issue 0100-01 and issue 0100-00

Change	Location	Affects
Updated company name to Arm	-	r1p0

Table A-5 Differences between issue 0200-00 and issue 0100-01

Change	Location	Affects
Changed revision to r2p0	*3.3 FPSID, Floating-Point System ID Register* on page 3-30	r2p0