USR-14016

Document No.: D-SLD-CMN-0015-01

Register IF Generator (RIGen) User manual

Renesas System Design Co., Ltd.

Design Integration Department

RENESAS CONFIDENTIAL

04/03/2015 Rev. 1.0

00000-A

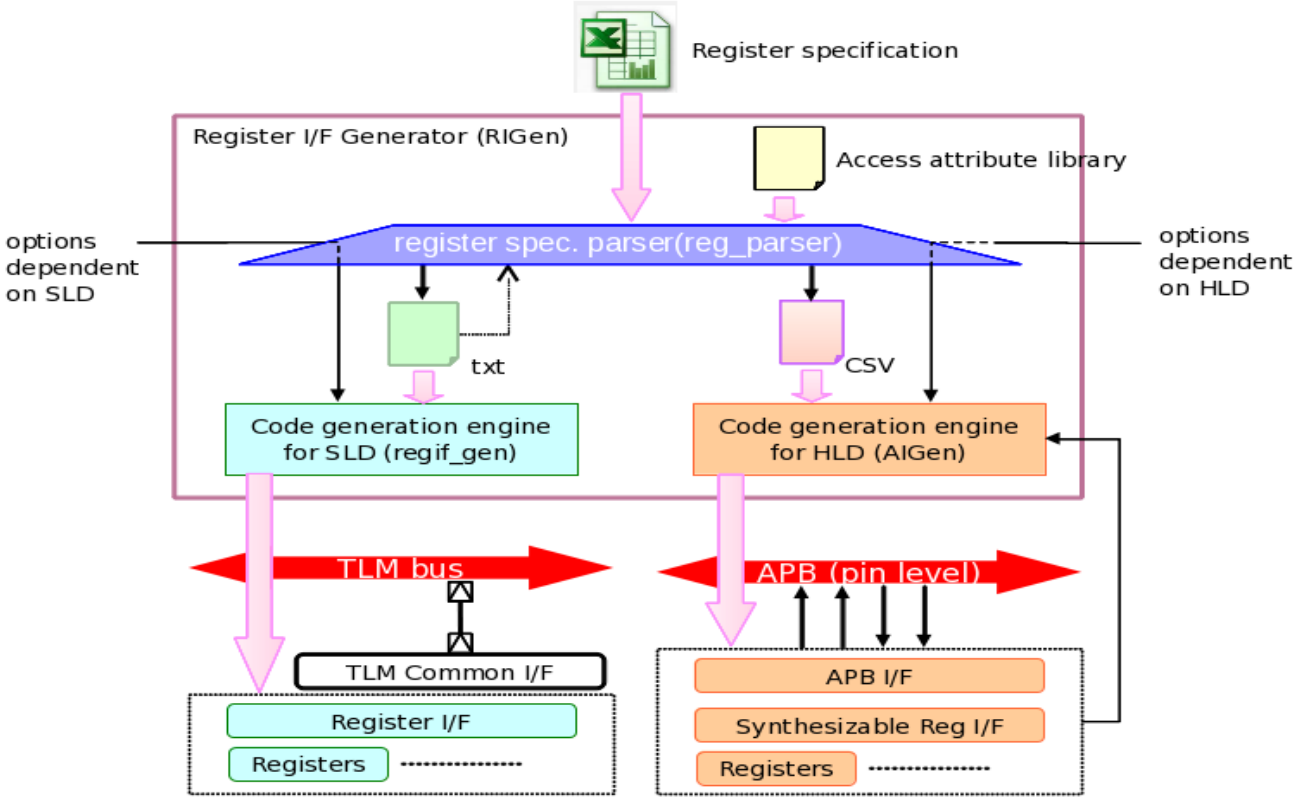
Outline

- Overview
  - Summary
  - Input/Output
  - Design Flow
- Usage
  - Setting & Syntax
  - Dependent options
- Features
  - Single register
  - Multiple registers
  - Multiple access size
  - Support to update from an intermediate file
  - Access Attribute Library File
- Related documents

Summary

RIGen, Register I/F Generator, consists of regif_gen, AIGen and reg_parser. This tool is used to unify input of System Level Design (SLD) and High Level Design (HLD) flows.

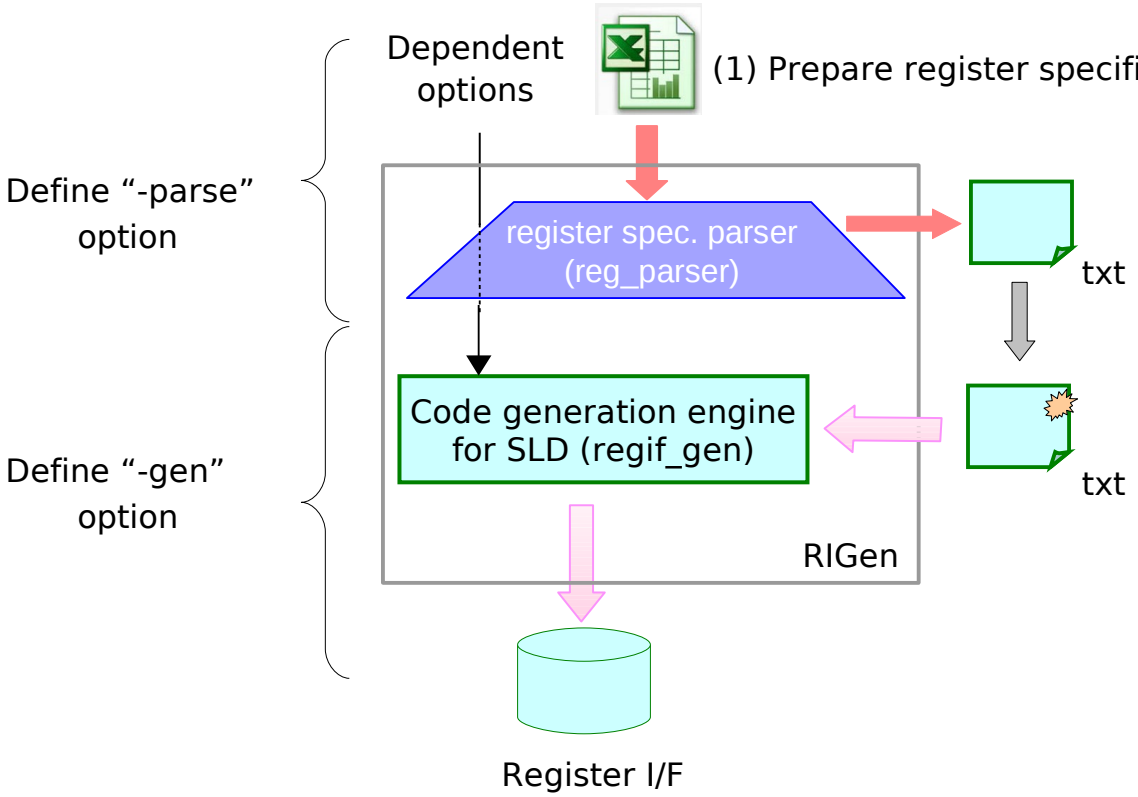
RIGen parses a register specification file, in Excel format, and generates register I/F source code used for transaction level or pin level, respectively.



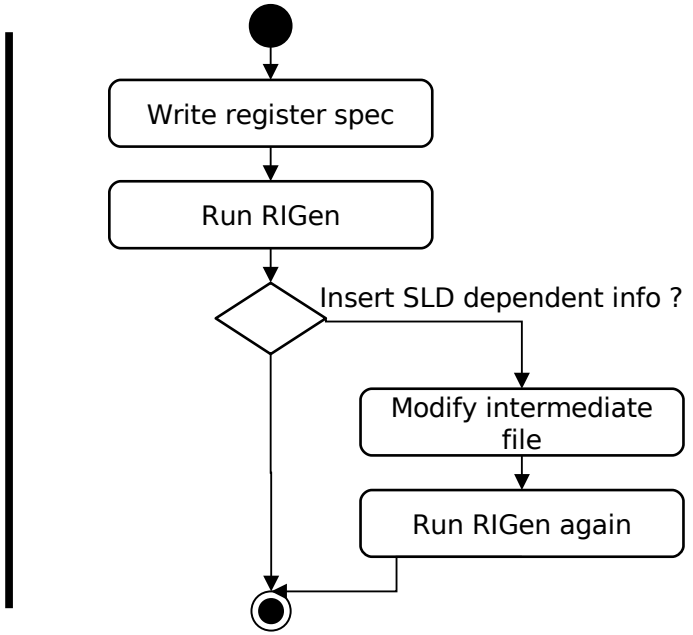
Input/Output

- Input
  - Register specification file, in Excel format, describes register information of at least one module/peripheral.
  - Options for RIGen tool. Refer to slide 9 for more detail.
  - Options for SLD/HLD generation tool: specific options will be handed-over to code generation engine of SLD/HLD. Refer to slide 10 & 11 for more detail.
  - Intermediate file, the file was generated by reg_parser script.
    This file is an unfinished file after the first RIGen's execution and need to modify.
    Intermediate file can be RIGen's input file to do as following:
      - To get updated information of both register spec. file and intermediate file.
      - To execute code generation engine.
  - Access attribute library file
- Output
  - Register I/F source code

SLD Design Flow

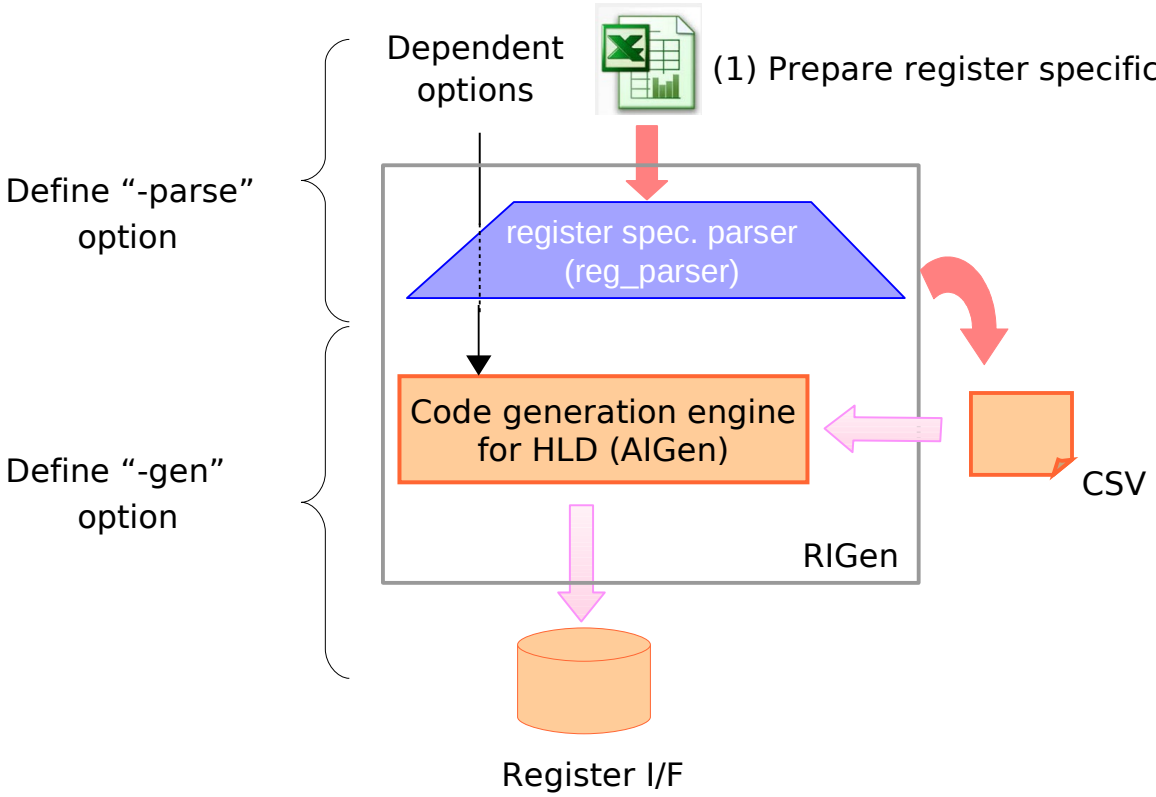


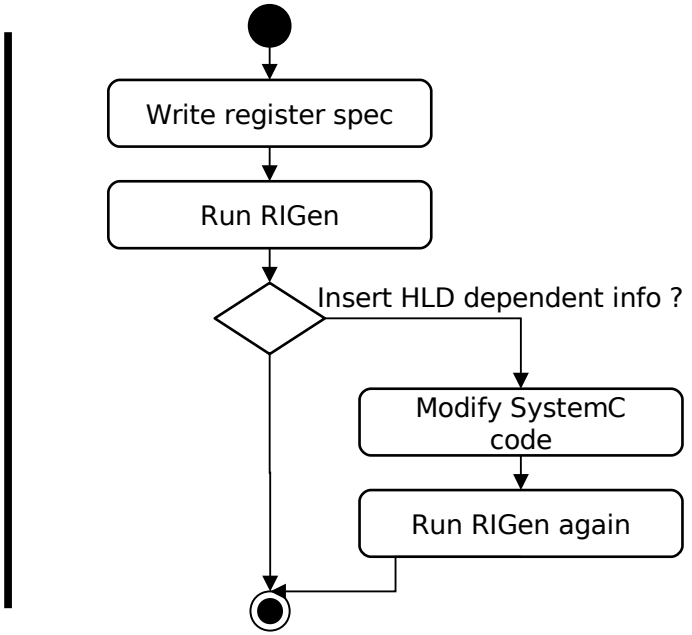
(3) Adjust intermediate (.txt) file



Intermediate file has been modified

HLD Design Flow





Outline

- Overview
  - Summary
  - Input/Output
  - Design Flow
- Usage
  - Setting & Syntax
  - Dependent options
- Features
  - Single register
  - Multiple registers
  - Multiple access size
  - Support to update from an intermediate file
  - Access Attribute Library File
- Related documents

Usage of Register IF Generator (1/2) - Setting

The following information is required before executing RIGen script:

(1) System requirement

Software/Module	Version
OS	Linux Red Hat 5.0
Python	3.1.2 or later
xlrd (*)	0.9.3

(*) It is a python library to extract data from Microsoft Excel (tm) spreadsheet files
Author : John Machin

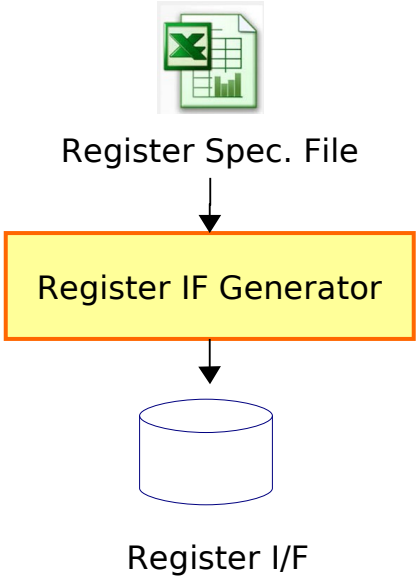
(2) Setting environment variable

No.	Variable Name	Description
1	SLD_PATH	SLD's generation tool directory
2	HLD_PATH	HLD's generation tool directory

Usage of Register IF Generator (2/E) - Syntax

python3 rigen [-parse|-gen] [-big] [-help] [-s sheet_name] <--sld [sld_option] | --hld [hld_option]> <register spec.>

e.g: python3 rigen -parse -big -s CRC --sld --onefile reg_spec.xls



No.	Option	Meaning	Mandatory
1	-parse | -gen	-parse: generate intermediate file only -gen: execute code generation engine only -Not defined: execute both (-parse and -gen)	No
2	-help	Show an instruction to define RIGen's option	No
3	-big	Specify register in big endian. If not defined, register is specified in little endian.	No
4	-s sheet_name	Specify target sheet name of register spec. If not defined, RIGen will get the first sheet of register spec.	No
5	--sld [sld_option] or --hld [hld_option]	--sld [sld_option]: execute RIGen in SLD flow --hld [hld_option]: execute RIGen in HLD flow Refer to slide 10 & 11 for sld/hld options.	Yes
6	register spec. (*)	Input file of RIGen	Yes

Note: (*) It is not necessary to define register spec. file when executing RIGen script from code generation engine.

Restriction: It is necessary to specify --sld/--hld after options 1-4

Dependent options of SLD flow

The following options are defined in RIGen and handed-over to code generation engine:

No.	Option	Meaning	Mandatory
1	--help	Show this help message and exit	No
2	--version	Show code generation engine version number and exit	No
3	-i, --hier (*)	Generate register I/F with hierarchical type	No
4	-a, --metadata	Generate metadata python file	No
5	-o, --onefile	Generate only header file (.h)	No
6	-k KEYWORD, Or --keyword=KEYWORD	Keyword for output file naming <keyword>_regif.*	No
7	-m MODULENAME1, MODULENAME2, ..., Or --module=MODULENAME1, MODULENAME2, ...	List of generated module name	No

Note: (*) Hierarchical-type only support when intermediate file has already created, RIGen is executed with "-gen" option.

Dependent options of HLD flow

The following options are defined in RIGen and handed-over to code generation engine:

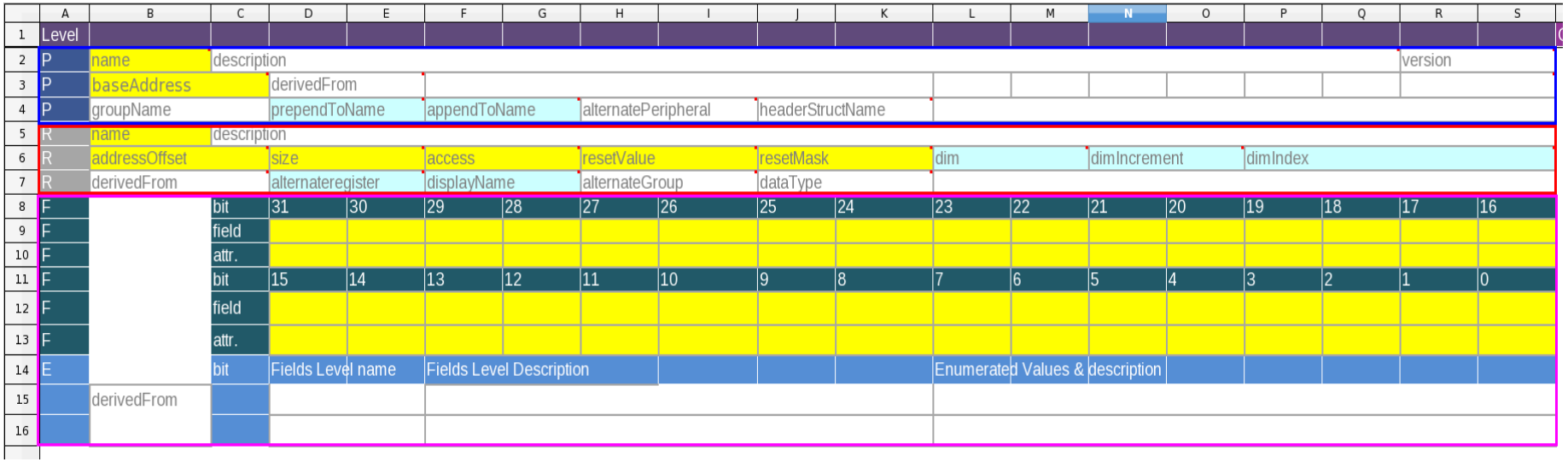
No.	Option	Meaning	Mandatory
1	-out <outdir>	Specify the directory path where code generation engine (aigen) generates output files. If not specified, aigen generates to current directory	No
2	-help	Show this help message and exit	No
3	-subdir	Generate file into each sub directory according to the category	No

Outline

- Overview
  - Summary
  - Input/Output
  - Design Flow
- Usage
  - Setting & Syntax
  - Dependent options
- Features
  - Single register
  - Multiple registers
  - Multiple access size
  - Support to update from an intermediate file
  - Access Attribute Library File
- Related documents

Register specification preparation (1/7) – Format

Register specification, in excel format, is an input file of register I/F generator.
The format of register spec. is shown as below:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	Level																		
2	P	name	description															version	
3	P	baseAddress		derivedFrom															
4	P	groupName		prependToName		appendToName		alternatePeripheral		headerStructName									
5	R	name	description																
6	R	addressOffset		size		access		resetValue		resetMask		dim		dimIncrement		dimIndex			
7	R	derivedFrom		alternateregister		displayName		alternateGroup		dataType									
8	F		bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
9	F		field																
10	F		attr.																
11	F		bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
12	F		field																
13	F		attr.																
14	E		bit	Fields Level name		Fields Level Description						Enumerated Values &		description					
15		derivedFrom																	
16																			

- Field
  - “P” is used to specify module/peripheral description
  - “R” is used to specify register description
  - “F” & “E” is used to specify bit description

Note: the above definitions in ▭ are mandatory for RIGen script, and ▭ are optional.
The other items will not be used in RIGen.

Restriction: RIGen supports register spec. file in Excel 97/2003 format only.

Register specification preparation (2/7) - Example

An example of register spec. is shown as below:

Module	Symbol	BaseAddress
Serial Communication Interface	SCIM	0x7FFFC000

Register name	Symbol	Offset Address	Register Length	Access Size (bit)	Access Attribute	ResetValue	ResetMask
SCI Configuration	SMR	0x04	8	8	RW	0xF6	0xFF

Register	Bit	Bit symbol	Bit name	Description	R|W
SMR	[7]	CM	Communications Mode	0 Asynchronous mode 1 Clock synchronous mode	RW
	[6:3]	-	Reserved	These bits are read as 0. The write value must be 0	RW
	[2]	MP	Multi-Processor Mode (Valid only in asynchronous mode)	0 Multi-processor communications function is disabled 1 Multi-processor communications function is enabled	RW
	[1:0]	CKS	Clock Select	00 PCLK clock 01 PCLK/4 clock 10 PCLK/16 clock 11 PCLK/64 clock	RW

Register specification preparation (3/7) - Example

The definition in register spec. file is shown as below:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	Level																		
2	P	SCIM	Serial Communication Interface module															v0.40	
3	P	0x7FFFC000		derivedFrom															
4	P	groupName		prependToName		appendToName		alternatePeripheral		headerStructName									
5	R	SMR	SCI Configuration																
6	R	0x04				8	rw	0xF6		0xFF		dim		dimIncrement		dimIndex			
7	R	derivedFrom		alternateregister		displayName		alternateGroup		dataType									
8	F		bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
9	F		field	x								CM	-				MP	CKS	
10	F		attr.	x								rw	rw0				rw	rw	
11	E		bit	Fields Level name		Fields Level Description						Enumerated Values & description							
12	E	derivedFrom	7	CM		Communications Mode						0 Asynchronous mode 1 Clock synchronous mode							
13	E	derivedFrom	6	Reserved		These bits are read as 0. The write value must be 0.						-							
14			3																
15	E	derivedFrom	0	CKS		Clock Select						00 PCLK clock 01 PCLK/4 clock 10 PCLK/16 clock 11 PCLK/64 clock							
16	E		1																
17	E	derivedFrom	2	MP		Multi-Processor Mode (Valid only in asynchronous mode)						0 Multi-processor communications function is disabled 1 Multi-processor communications function is enabled							

Note: the definition at “Field Level Description” and “Enumerated Valued & description” will not be used to generate an intermediate file.

Register specification preparation (4/7) - Multiple Registers

Register I/F Generator supports to define multiple registers:

Module	Symbol	BaseAddress
Serial Communication Interface	SCIM	0x7FFFC000

Register name	Symbol	Offset Address	Register Length	Access Size (bit)	Access Attribute	ResetValue	ResetMask
SCI Configuration	SMR0	0x04	8	8	RW	0xF6	0xFF
	SMR1	0x14	8	8	RW	0xF6	0xFF
	SMR2	0x24	8	8	RW	0xF6	0xFF

(*) Bit description of each register is same as bit information in slide 14.

Register specification preparation (5/7) – Multiple Registers

Register spec. file define multiple registers as below:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	Level																		
2	P	SCIM	Serial Communication Interface module															v0.40	
3	P	0x7FFFC000		derivedFrom															
4	P	groupName		prependToName		appendToName		alternatePeripheral		headerStructName									
5	R	SMR	SCI Configuration																
6	R	0x04			8	rw		0xF6		0xFF		3		0x10		0-2			
7	R	derivedFrom		alternateregister		displayName		alternateGroup		dataType									
8	F		bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
9	F		field	x								CM	-				MP	CKS	
10	F		attr.	x								rw	rw0				rw	rw	
11	E		bit	Fields Level name		Fields Level Description						Enumerated Values & description							
12	E	derivedFrom	7	CM		Communications Mode						0 Asynchronous mode 1 Clock synchronous mode							
13	E	derivedFrom	6	Reserved		These bits are read as 0. The write value must be 0.						-							
14			3																
15	E	derivedFrom	0	CKS		Clock Select						00 PCLK clock 01 PCLK/4 clock 10 PCLK/16 clock 11 PCLK/64 clock							
16	E		1																
17	E	derivedFrom	2	MP		Multi-Processor Mode (Valid only in asynchronous mode)						0 Multi-processor communications function is disabled 1 Multi-processor communications function is enabled							

To define for multiple registers; “dim”, “dimIncrement” & “dimIndex” will be used as following notice:

- dim: used to define amount of registers.
- dimIncrement: used to define the address increment between two neighboring registers.
- dimIndex: used to append into register symbol.

Note: it is necessary to define “dimIndex” explicitly, in case that it's defined by letter.

e.g. dimIndex = A,B,C ...

Register specification preparation (6/7)-Multiple Access Size

The description of register spec. is shown as below:

Module	Symbol	BaseAddress
CRC Calculator	CRC	0x40074000

Register name	Symbol	Offset Address	Register Length	Access Size (bit)	Access Attribute	ResetValue	ResetMask
CRC Data Output	CRCDOR	0x08	32	16|32	RW	0x00000000	0xFFFFFFFF

Register	Bit	Bit symbol	Bit name	Description	R|W
CRCDOR	[31:0]	CRCDOR	Calculation output data	-	RW
	[15:0]	CRCDOR_HA	Calculation output data (word access)	-	RW

Note: Bit CRCDOR_HA[15:0] is overlap bit with CRCDOR[31:0]. Refer to slide 22 for more detail

Register specification preparation (7/E)-Multiple Access Size

The definition of register spec. is shown as below:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	Level																		
2	P	CRC	CRC Calculator															version	
3	P	0x40074000		derivedFrom															
4	P	groupName		prependToName		appendToName		alternatePeripheral		headerStructName									
5	R	CRCDOR	CRC Data Output Register																
6	R	0x08		32		rw		0x00000000		0xFFFFFFFF		dim		dimIncrement		dimIndex			
7	R	derivedFrom		alternateregister		displayName		alternateGroup		dataType									
8	F		bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
9	F		field	CRCDOR															
10	F		attr.	rw															
11	F		bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
12	F		field																
13	F		attr.																
14	E		bit	Fields Level name		Fields Level Description						Enumerated Values & description							
15	E		31	CRCDOR		Calculation output Data (Case of CRC-32, CRC-32C)													
16	E		0																
17	R	CRCDOR_HA	CRC Data Output Register (word access)																
18	R	0x08		16		rw		0x0000		0xFFFF		dim		dimIncrement		dimIndex			
19	R	derivedFrom		CRCDOR		displayName		alternateGroup		dataType									
20	F		bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
21	F		field	CRCDOR_HA															
22	F		attr.	rw															
23	E		bit	Fields Level name		Fields Level Description						Enumerated Values & description							
24	E		15	CRCDOR_HA		Calculation output Data (Case of CRC-16 or CRC-CCITT)													
25	E		0																

Define “alternateregister” cell (CRCDOR) is required in definition of CRCDOR_HA

Restriction: It is necessary to specify CRCDOR_HA after CRCDOR definition.

Intermediate File – SLD Flow

For an example in slide 14, the intermediate file is shown as below:

```
%MODULE   SCIM
                    #name    offset_size
    %%REG_INSTANCE   reg_def   3

%REG_CHANNEL     reg_def
    %%TITLE  name   reg_name  size  length  offset      init        access  support  callback
    %%REG    SMR    SMR       8     8       0x00000004  0x000000F6  R|W     TRUE     -

%REG_NAME  SMR
    %%TITLE  name   upper  lower  init        access    support  callback  value
    %%BIT    CM     7      7      0x0001      R|W       TRUE     -         -
    %%BIT    MP     2      2      0x0001      R|W       TRUE     -         -
    %%BIT    CKS    1      0      0x0002      R|W       TRUE     -         -
```

Note:

It is necessary to modify the intermediate file at “access”, “support”, “callback” and “value” columns before executing code generation engine (regif_gen).
(“value” column may be omitted in case that there is no bit defined writable value list).

Register I/F generator also supports to get update information from both register spec. file and intermediate file, in case that the intermediate file name is specified in [sld_option].
Syntax: python3 rigen -parse [...] <--sld [sld_options]> <reg_spec.>
([sld_options] includes <intermediate_file_name.txt>)

Intermediate File - HLD Flow

For an example in slide 14, the intermediate file is shown as below:

IP name SCIM
Mask address
Asynchronous transfer

1 Register List

No	Register Name	Symbol Name	Address	Register Size	Access Size	Access Type	Initial Value	Index	Address Step
1	Register 1	SMR	0x0004	8	8	R/W	0x00F6		

2 Register Settings

2.1 SMR

No	Bit name	Symbol Name	Upper Bit	Lower Bit	Access Type	Initial Value	Index	Placement	Access from
1	CM	CM	7	7	R/W	0x1			
2	MP	MP	2	2	R/W	0x1			
3	CKS	CKS	1	0	R/W	0x2			

Support Overlap Bit (1/2)

As an example in slide 18, the intermediate file is shown as below:

```
%MODULE   CRC
                    #name    offset_size
    %%REG_INSTANCE   reg_def   4

%REG_CHANNEL     reg_def
    %%TITLE  name    reg_name  size   length  offset      init        access  support  callback
    %%REG    CRCDOR  CRCDOR    16|32  32      0x00000008  0x00000000  R|W     TRUE     -

%REG_NAME  CRCDOR
    %%TITLE  name       upper  lower  init     access    support  callback  overlap
    %%BIT    CRCDOR     31     0      0x0000   R|W       TRUE     -         FALSE
    %%BIT    CRCDOR_HA  15     0      0x0000   R|W       TRUE     -         TRUE
```

Note:

"overlap" column is defined to support overlap bit.

In above example, the range of CRCDOR_HA bit overlaps with that of CRCDOR bit

The "overlap" is TRUE means that overlap bit is an expectation

Support Overlap Bit (2/E)

As an example in slide 18, the standard output shows below information:

```
<INFO>  Implement in callback function for each register
        if access in other position which is out of following:
    Register  Address  Access_Size(bytes)
    -------------------------
    CRCDOR    0x0008     4
              0x0008     2
```

Explanation:

Register CRCDOR can be accessed with size 2 or 4 bytes at 0x08.

In order to access 2-byte at 0x0A, implementation callback function is required.

Access Attribute Library File (1/2)

“access_attr.lib” is a library file of RIGen. It's used to convert access attribute from SC32 format to SLD or HLD format, respectively.

```
#access ,SLD-equivalent,HLD-equivalent
W       ,W              ,W
w--zc   ,W0             ,W0
w--zs   ,W0:1           ,W0:1
w--zt   ,W0:T           ,W0:T
w--oc   ,W1:0           ,W1:0
w--os   ,W1             ,W1
w--ot   ,W1:T           ,W1:T
w--se   ,W:1            ,W:1

w--cl   ,W:0            ,W:0

…
```

Limitation: Refer to slide 26 & 27 for limitation of HLD access attribute

Access Attribute Library File (2/E)

Following information is rules of defining SC32 format access attribute:

(1) Register definition

Attribute (Register Field)	
r	read-only
w	write-only
w0	write “0”-only
w1	write “1” -only
rw	read-write
rw0	read-write”0”
rw1	read-write”1”

(2) Bit definition

ReadAction	
r*-c-	clear
r*-s-	set
r*-m-	modify
r*-e-	modifyExternal

WriteAction	
w--oc	oneToClear
w--os	oneToSet
w--ot	oneToToggle
w--zc	zeroToClear
w--zs	zeroToSet
w--zt	zeroToToggle
w--cl	clear
w--se	set

Example:

SC32 format	Meaning	Equivalent in SLD
rw1-c-oc	Separate each action as below: (1) r-c- : clear bit after read (2) w1--oc: clear bit if write 1 into bit	R:0|W1:0

Limitation (1/2)

Following HLD access attribute have not supported yet

```
#access,HLD-equivalent
w--zc,W0
w--zs,W0:1
w--zt,W0:T
w--oc,W1:0
w--ot,W1:T
w--cl,W:0
w0,W0
w0--zc,W0
w0--zs,W0:1
w0--zt,W0:T
w0--se,W0:1
w0--cl,W0
w1--oc,W1:0
w1--ot,W1:T
w1--cl,W1:0
rw--zc,R/W0
rw--zs,R/W0:1
rw--zt,R/W0:T
rw--oc,R/W1:0
rw--ot,R/W1:T
rw--cl,R/W:0
rw-c-,R:0/W
```

```
#access,HLD-equivalent
rw-c-zc,R:0/W0
rw-c-zs,R:0/W0:1
rw-c-zt,R:0/W0:T
rw-c-oc,R:0/W1:0
rw-c-os,R:0/W1
rw-c-ot,R:0/W1:T
rw-c-se,R:0/W:1
rw-c-cl,R:0/W:0
rw-s-zc,R:1/W0
rw-s-zs,R:1/W0:1
rw-s-zt,R:1/W0:T
rw-s-oc,R:1/W1:0
rw-s-ot,R:1/W1:T
rw-s-cl,R:1/W:0
rw-e-zc,R/W0
rw-e-zs,R/W0:1
rw-e-zt,R/W0:T
rw-e-oc,R/W1:0
rw-e-ot,R/W1:T
rw-e-cl,R/W:0
rw0,R/W0
rw0--zc,R/W0
```

```
#access,HLD-equivalent
rw0--zs,R/W0:1
rw0--zt,R/W0:T
rw0--se,R/W0:1
rw0--cl,R/W0
rw0-c-,R:0/W0
rw0-c-zc,R:0/W0
rw0-c-zs,R:0/W0:1
rw0-c-zt,R:0/W0:T
rw0-c-se,R:0/W0:1
rw0-c-cl,R:0/W0
rw0-s-,R:1/W0
rw0-s-zc,R:1/W0
rw0-s-zs,R:1/W0:1
rw0-s-zt,R:1/W0:T
rw0-s-se,R:1/W0:1
rw0-s-cl,R:1/W0
rw0-e-,R/W0
```

Limitation (2/E)

Following HLD access attribute have not supported yet.

```
#access,HLD-equivalent
rw0-e-zc,R/W0
rw0-e-zs,R/W0:1
rw0-e-zt,R/W0:T
rw0-e-se,R/W0:1
rw0-e-cl,R/W0
rw1--oc,R/W1:0
rw1--ot,R/W1:T
rw1--cl,R/W:0
rw1-c-,R:0/W1
rw1-c-oc,R:0/W1:0
rw1-c-os,R:0/W1
rw1-c-ot,R:0/W1:T
rw1-c-se,R:0/W1
rw1-c-cl,R:0/W1:0
rw1-s-oc,R:1/W1:0
rw1-s-ot,R:1/W1:T
rw1-s-cl,R:1/W1:0
rw1-e-oc,R/W1:0
rw1-e-ot,R/W1:T
rw1-e-cl,R/W:0
```

Outline

- Overview
  - Summary
  - Input/Output
  - Design Flow
- Usage
  - Setting & Syntax
  - Dependent options
- Features
  - Single register
  - Multiple registers
  - Multiple access size
  - Support to update from an intermediate file
  - Access Attribute Library File
- Related documents

Related documents

- User Manual of regif_gen
  - USR-SLD-14011_RegisterIF.pdf (ver1.0)
- User Manual of AIGen
  - USR-HLD-14005_APBIF_generator_user_manual.pdf (ver1.0)

Revision History

Rev.No	Contents	Agreed by Customer	Approval by RVC	Checked	Created
1.0	•New creation	A.Imoto 2015/04/06	Son Tran 2015/04/03	Duc Duong Hiep Nguyen 2015/04/03	Thanh Phan 2015/04/03

Renesas System Design Co., Ltd.