Internal Specification

Development of SPIDCTL model for E2x, U2A and U2B

(v1.3)

Summary:

This document describes the Detail Specification of SPIDCTL model for U2B, U2A and E2x

Relative Document

Reference Manuals				
No.	**Title name**	**Document number**	**Description**	**Path**
1	SC-HEAP_E3 common requirement (v1.0)	-	The common requirement (**File:** Common_Requirement_RVC.pdf)	**SPO: RVC-DMS** 1. General Documents/ESW/MCU_SW/01_Automotive_Software_Tool/01_Automotive_Software_Tool_1/01_MCU_Modeling/INPUT
2	SC-HEAP_E3 Modeling guideline (Rev. 7.00)	IDF-14-010278-01	This document describes the Guideline for peripheral macro development which is connected to SC-HEAP_E3 simulator (**File**: SC-HEAP_E3_Modeling_Guideline.pdf)	
3	SC-HEAP_E3 BUS I/F outline	LLWEB-00010925 ZSG-F31-12-0029-01	The document describes the outline of bus I/F applied to SC-HEAP_E3 (**File**: scheap_e3_bus_if_outline_E.pdf)	
4	SC-HEAP_E3 PYTHON I/F function specification (v3.0)	LLWEB-00105192 MSS-SG-12-0062-03	The document describes how to use python interface (**File:** SC-HEAP_E3 Python IF_t.pdf)	
5	RH850/U2A-EVA Group User Manual: Hardware	r01uh0864ej0061-rh850u2a(Rev.0.61 Jun, 2019)	Hardware user's manual of SPID. (**File:** r01uh0864ej0061-rh850u2a.pdf)	**Server:** /shsv/MobAP2/prj_MCS/02_projects/2020/rel/2

Reference Manuals				
6	RH850/E2x-FCC2 User's manual: Hardware	r01uh0770ej0060-e2x-fcc2_E2UH_E2H(Rev.0.60 Feb 2019)	Hardware user's manual of SPID. (**File**: r01uh0770ej0060-e2x-fcc2_E2UH_E2H.pdf)	02001_prj_MCS_U2A_SPID/Input
7	RH850/E2x-FCC2 User's manual: Hardware	r01uh0771ej0060-e2x-fcc2_E2GUH_E2GH(Rev.0.60 Feb 2019)	Hardware user's manual of SPID. (**File**: r01uh0771ej0060-e2x-fcc2_E2GUH_E2GH.pdf)	
8	SPIDCTL model detail requirement	REQ-MCS-20001_SPIDCTL (*)	Detail requirement of SPIDCTL model (**File:** REQ-MCS-20001_SPIDCTL.xlsx)	**SVN:** http://172.29.139.78/mcu_modeling_01/release/docs/REQ/2020/20001_U2A_SPIDCTL
9	U2B/HWUM	r01uh0924ej0040-rh850u2b Rev: 040	U2B/HWUM (r01uh0924ej0040-rh850u2b.pdf)	**Server:** \\rvc-vnas-01\MobAP2\prj_MCS\06_From_REL\20210301_U2BxHWUM0.4\r01uh0924ej0040-rh850u2b.pdf

. ***Note**: (*) Refer to DEV-MCS-20001_SPIDCTL for version number.*

Contents

Index of Figures

Index of Tables

1.Model summary

(1) SPIDCTL is a SystemC model supporting U2B, U2A and E2x platforms, presents for System Protection Identifier module. This model is used for supervising SPIDs which can be used by specific BusMasters.

(2) In this design, the followings are supported in SPIDCTL model:

(2.1) 32-bit width of APB bus.

(2.2) One target socket for access registers inside. Refer chapter 3 for detail.

(2.3) Both loosely time mode (LT) and approximately time (AT) mode are supported.

(2.4) Little endian mode as the endian of APB bus interface.

(2.5) Parameters/Commands of Python IF to control operation of SPIDCTL model. Refer to chapter parameters/commands for more detail.

2.Supported features

Table 2.1: List of supported features in SPIDCTL model

Feature	Description		HWM Reference
	Hardware	Model	
Operating frequency (PCLK)	- Clock source CLK_HBUS	Unlimited frequency	
Asynchronous reset (Reset)	-	Asynchronous	
High flexibility	- SPID of each bus master is configurable by software with high flexibility *according* to user's system.	<-	[5] Chapter 44.5.3.1 Overview [9] Section 55.5.3.1 Overview
Assign SPID to bus master	- It is possible to assign one SPID to one bus master or assign one SPID to several bus masters.	<-	[5] Chapter 44.5.3.1 Overview [9] Section 55.5.3.1 Overview
Limit SPID by mask register	- It is possible to limit SPID that each bus master uses by SPIDMASK register. Moreover, this configuration can be locked.	<-	[5] Chapter 44.5.3.1 Overview [9] Section 55.5.3.1 Overview
Security purpose	- The lock function prevents unauthorized use of SPID for security purpose.	<-	[5] Chapter 44.5.3.1 Overview [9] Section 55.5.3.1 Overview

Note: "<-": same as hardware manual

3.Block diagram

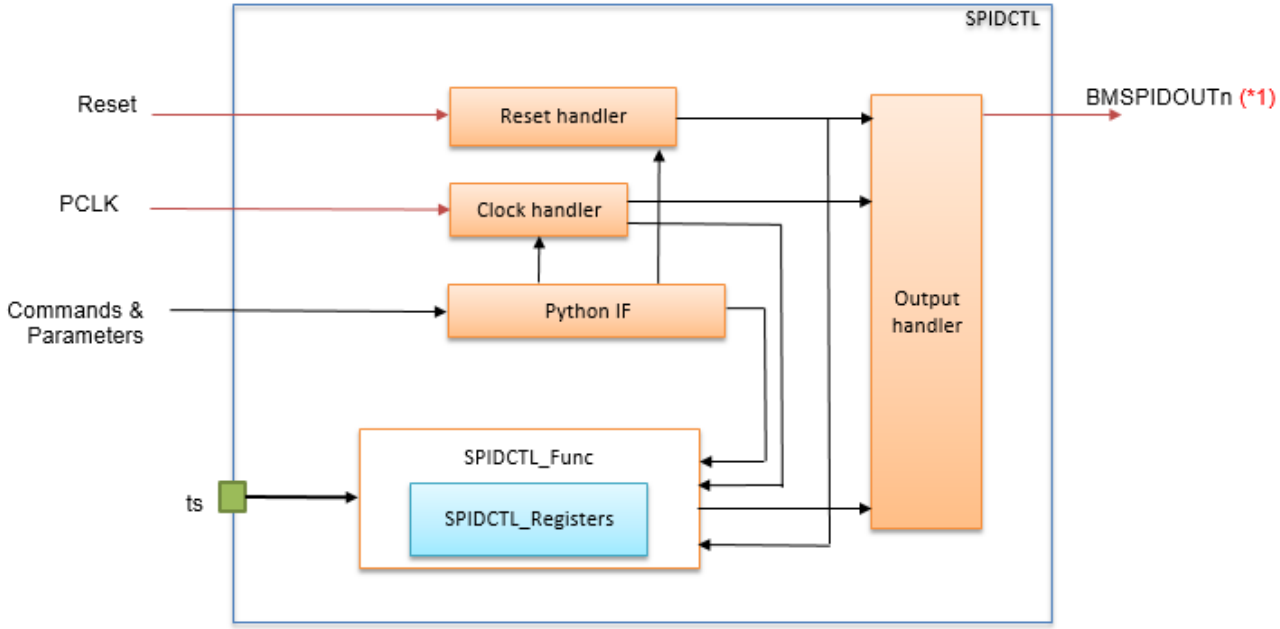

Figure 3-1: Block diagram of SPIDCTL model

Explanation:

(1) The SPIDCTL model has sub-blocks inside:

(1.1) Reset Handler: checks the reset port, the reset command from Python IF to reset the model.

(1.2) Clock Handler: checks the input clock, the clock command from Python IF to generate the period of operation clock.

(1.3) Python IF: with parameters/commands supports controlling SPIDCTL.

(1.4) SPIDCTL_Func: has registers inside. Its controls operation of SPIDCTL module.

(1.5) Output handler: control the output port.

(2) Target socket 'ts': APB interface to access model's registers.

Note: (*1) n =0~63.

4.List of registers

(1) Table 4.1 lists all registers in SPIDCTL model.

Table 4.1: List of registers in SPIDCTL model

Register name	Address Offset	Initial value	Size(Byte)	Write access size (bits)	Read access size (bits)	R/W	Bit position	Bit name		Operation	Support
								HWM	Model		
BMnSPID (BusMaster n SPID Register)**(*1)**	<SPIDCTL_base> + n * 0x4 (*1)	(*2)	4	8/16/32	8/16/32	R/W	[4:0]	SPID	<-	SPID for each bus master. These bits are set the SPID of each bus master.	Yes
BMnSPIDMSK (Bus Master n SPID Mask Register) **(*1)**	<SPIDCTL_base> + 0x100 + n * 0x4 (*1)	0xFFFFFFFF	4	8/16/32	8/16/32	R/W	[31:0]	SPIDMSK	<-	Set inhibition of SPID value which that each bus master can use. Each bit corresponds to a single SPID value 0: The bus master cannot use this SPID. Setting of this SPID is inhibited. 1: The bus master can use this SPID. Setting of this SPID is not inhibited.	Yes

BMnSPIDMSKLOCK (Bus Master n SPID Mask Lock Register)**(*1)**	<SPIDCTL_base> + 0x200 + n * 0x4 (*1)	0x0	4	8/16/32	8/16/32	RW	[0]	LOCK	<-	Set write mask to each BMSPIDMSKn. 0: Register BMSPIDMSKn can be re-written 1: Any further write to the register BMSPIDMSKn is ignored. This bit can be cleared by Power On Reset, System Reset 1, System Reset 2, Application Reset and DeepSTOP Reset.	Yes
SPIDKCPROT (SPID Key Code Protection Register)	<SPIDCTL_base> + 0x300	0x0	4	32	8/16/32	RW	[31:1]	KCPROT	<-	Enable or disable modification of the KCE bit. The value written is not retained. These bits are always read as 0.	Yes
							[0]	KCE	<-	Key Code Enable bit 0: Disable write access of protected registers. 1: Enable write access of protected registers.	Yes
GTMSPIDAS0 (GTM SPID Module Assignment Setting Register 0)	0xFFF50000 + 0x18	0x76543210	4	8/16/32	8/16/32	RW	[30:28]	CL7	<-	Select the SPID module assign to cluster 7. 0: SPID module(n=10) 1: SPID module(n=11) 2: SPID module(n=12) 3: SPID module(n=13) 4: SPID module(n=14) 5: SPID module(n=15) 6: SPID module(n=16) 7: SPID module(n=17)	No
							[26:24]	CL6	<-	Select the SPID module assign to cluster 6. 0: SPID module(n=10) 1: SPID module(n=11) 2: SPID module(n=12) 3: SPID module(n=13) 4: SPID module(n=14) 5: SPID module(n=15) 6: SPID module(n=16) 7: SPID module(n=17)	No

[22:20]	CL5	<-	Select the SPID module assign to cluster 5. 0: SPID module(n=10) 1: SPID module(n=11) 2: SPID module(n=12) 3: SPID module(n=13) 4: SPID module(n=14) 5: SPID module(n=15) 6: SPID module(n=16) 7: SPID module(n=17)	No
[18:16]	CL4	<-	Select the SPID module assign to cluster 4. 0: SPID module(n=10) 1: SPID module(n=11) 2: SPID module(n=12) 3: SPID module(n=13) 4: SPID module(n=14) 5: SPID module(n=15) 6: SPID module(n=16) 7: SPID module(n=17)	No
[14:12]	CL3	<-	Select the SPID module assign to cluster 3. 0: SPID module(n=10) 1: SPID module(n=11) 2: SPID module(n=12) 3: SPID module(n=13) 4: SPID module(n=14) 5: SPID module(n=15) 6: SPID module(n=16) 7: SPID module(n=17)	No
[10:8]	CL2	<-	Select the SPID module assign to cluster 2. 0: SPID module(n=10) 1: SPID module(n=11) 2: SPID module(n=12) 3: SPID module(n=13) 4: SPID module(n=14) 5: SPID module(n=15) 6: SPID module(n=16) 7: SPID module(n=17)	No
[6:4]	CL1	<-	Select the SPID module assign to cluster 1. 0: SPID module(n=10) 1: SPID module(n=11) 2: SPID module(n=12) 3: SPID module(n=13) 4: SPID module(n=14) 5: SPID module(n=15) 6: SPID module(n=16) 7: SPID module(n=17)	No

							[2:0]	CL0	<-	Select the SPID module assign to cluster 0. 0: SPID module(n=10) 1: SPID module(n=11) 2: SPID module(n=12) 3: SPID module(n=13) 4: SPID module(n=14) 5: SPID module(n=15) 6: SPID module(n=16) 7: SPID module(n=17)	No
GTMSPIDAS0 (GTM SPID Module Assignment Setting Register 0)	0xFFF50000 + 0x1C	0x00000010	4	8/16/32	8/16/32	RW	[6:4]	CL9	<-	Select the SPID module assign to cluster 9. 0: SPID module(n=10) 1: SPID module(n=11) 2: SPID module(n=12) 3: SPID module(n=13) 4: SPID module(n=14) 5: SPID module(n=15) 6: SPID module(n=16) 7: SPID module(n=17)	No
							[2:0]	CL8	<-	Select the SPID module assign to cluster 8. 0: SPID module(n=10) 1: SPID module(n=11) 2: SPID module(n=12) 3: SPID module(n=13) 4: SPID module(n=14) 5: SPID module(n=15) 6: SPID module(n=16) 7: SPID module(n=17)	No

Note: "<-": Same as hardware manual.

(*1): n = 0 to 63

(*2): Refer 10. Appendix: Initial value of SPID

Register base address:

Table 4.2: Base address of registers

Base Address Name	Base Address
SPIDCTL_base	FF0A 8000H

5.List of implemented ports

(1) Table 5.1 lists implemented ports in SPIDCTL model.

Table 5.1: List of implemented ports in SPIDCTL model

Signal name		I/O	Type	Initial	Active	Sync. clock	Description	Support
HWM	Model							
Clock/reset								
-	PCLK	In	sc_in<sc_dt::uint64>	-	-	-	Operation clock source	Yes
-	Reset	In	sc_in<bool>	HIGH	low level	-	Reset signal	Yes
APB I/F								
-	ts	In/Out	TlmTargetSocket<32>	-	-	-	Target socket of APB bus interface to access to registers of model	Yes
BMnSPIDOUT								
BMnSPIDOUT	BMSPIDOUT[64]	Out	sc_out<unsigned char>(*2)	(*1)	-	PCLK	SPID output value.	Yes

Note: (*1): Refer 10. Appendix: Initial value of SPID
(*2): Match with SPID input port of SPID Modifier model.

6.Direction for users

6.1.File structures

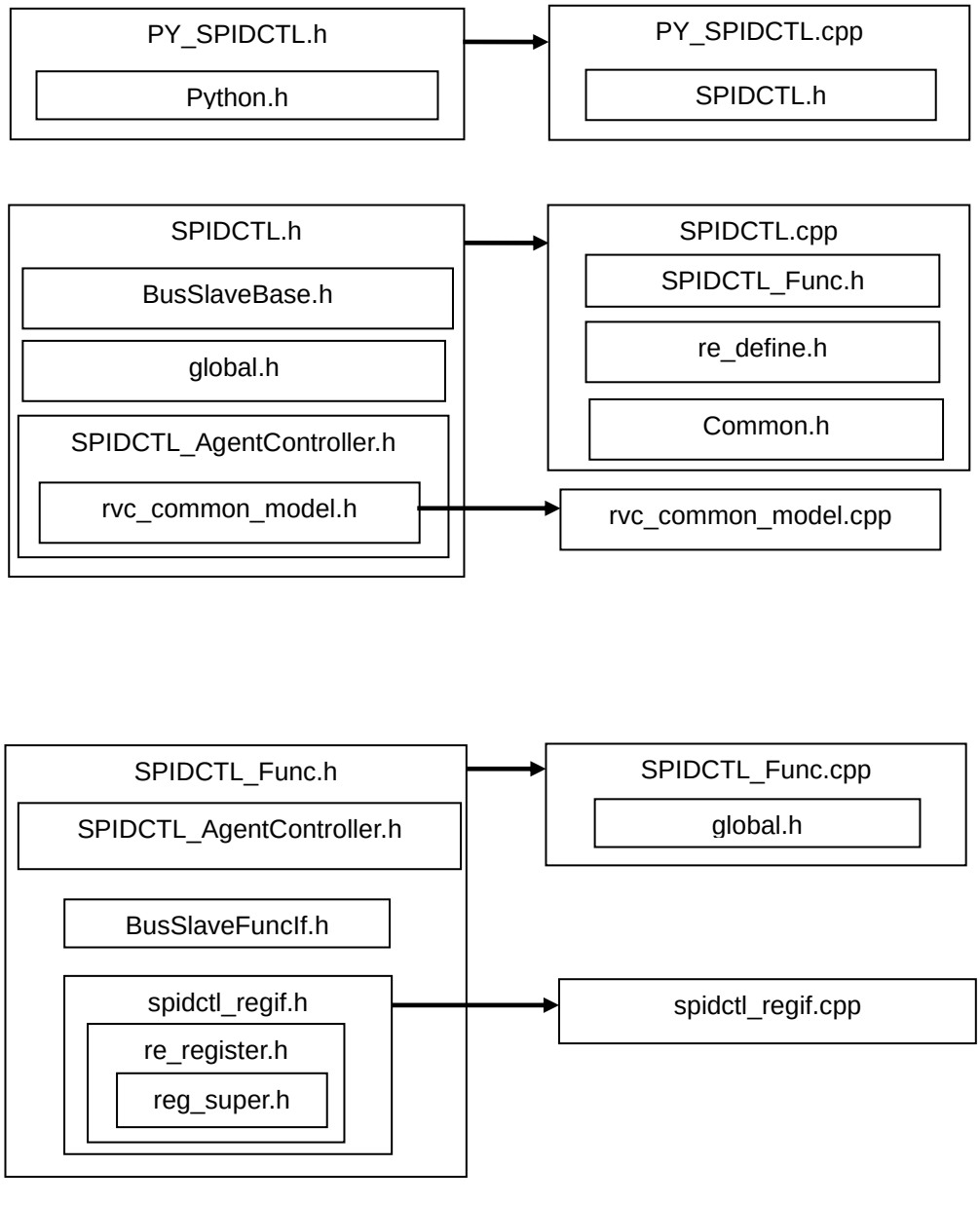

Figure 6-1: File structure of SPIDCTL model

Table 6.1: File description for SPIDCTL

No.	File name	Version	Developed/ Reused/ Generated	Description
1	PY_SPIDCTL.h	-	Developed	Header file of Python Interface of SPIDCTL model
2	PY_SPIDCTL.cpp		Developed	Implementation file of Python Interface of SPIDCTL model
3	SPIDCTL.h		Developed	Header file of SPIDCTL model
4	SPIDCTL.cpp		Developed	Implementation file of SPIDCTL model
5	SPIDCTL_Func.h		Developed	Header file of SPIDCTL function block
6	SPIDCTL_Func.cpp		Developed	Implementation file of SPIDCTL function block
7	SPIDCTL_AgentController.h		Developed	Header file includes virtual functions which are implemented in SPIDCTL model
8	spidctl_regif.h		Generated (*1)	Header file of SPIDCTL registers' interface
9	spidctl_regif.cpp		Generated (*1)	Implementation file of SPIDCTL registers' interface
10	Python.h	-	Reused	Header file of python library.
11	re_register.h	v2016_09_21	Reused	Header file of the re_register class.
12	re_register.cpp		Reused	Implement the attributes and the operations of common register class.
13	reg_super.h		Reused	General class for models to access to the memory array.
14	BusSlaveBase.h	-	Reused	Header file of BusSlaveBase class.
15	BusSlaveFuncIf.h	-	Reused	Header file of BusSlaveFuncIf class.
16	re_define.h	-	Reused	Define common define macro, enum and so on
17	rvc_common_model.cpp	-	Reused	Implementation file of common APIs
18	rvc_common_model.h	-	Reused	Header file of common APIs
19	global.h	-	Reused	Header file contains common macros
20	Common.h	-	Reused	Header file for token converting APIs

Note:

(*1) File with format *_regif.h/cpp in table above are generated from Register IF Generator tool v2020_03_20.

6.2.Input/Output file

(1) There is no input/output file.

6.3.How to connect Verification Environment

(1) The following steps should be done to connect SPIDCTL into environment.

(1.1) Declare an instance of the SPIDCTL class in environment.

```
spidctl = new SPIDCTL("spidctl", 0, 0, config_file);
```

Config file format:

[SPID_BUS_MASTER] = (x, y)

- x: Register position.
- y: Initial value.

Example: config file for U2B

[SPID_BUS_MASTER] = (10, 10)
[SPID_BUS_MASTER] = (11, 10)
[SPID_BUS_MASTER] = (12, 10)
[SPID_BUS_MASTER] = (13, 10)
[SPID_BUS_MASTER] = (14, 10)
[SPID_BUS_MASTER] = (15, 10)
[SPID_BUS_MASTER] = (16, 10)
[SPID_BUS_MASTER] = (17, 10)
[SPID_BUS_MASTER] = (18, 11)
[SPID_BUS_MASTER] = (19, 11)
[SPID_BUS_MASTER] = (20, 12)
[SPID_BUS_MASTER] = (21, 12)
[SPID_BUS_MASTER] = (22, 13)
[SPID_BUS_MASTER] = (23, 14)
[SPID_BUS_MASTER] = (24, 15)
[SPID_BUS_MASTER] = (27, 18)
[SPID_BUS_MASTER] = (28, 19)
[SPID_BUS_MASTER] = (32, 23)
[SPID_BUS_MASTER] = (33, 24)
[SPID_BUS_MASTER] = (34, 25)
[SPID_BUS_MASTER] = (35, 26)
[SPID_BUS_MASTER] = (36, 26)
[SPID_BUS_MASTER] = (37, 26)
[SPID_BUS_MASTER] = (43, 31)

Example: config file for E2x

[SPID_BUS_MASTER] = (19, 19)

[SPID_BUS_MASTER] = (21, 21)

[SPID_BUS_MASTER] = (22, 22)

[SPID_BUS_MASTER] = (23, 23)

[SPID_BUS_MASTER] = (24, 24)

[SPID_BUS_MASTER] = (26, 26)

(1.2) Connect the target socket of SPIDCTL instance to corresponding initiator socket.

(1.3) Connect reset/clock port of SPIDCTL to corresponding ports.

(1.4) Connect output ports to corresponding ports.

6.4.Commands and parameters

(1) Users set parameters/commands to the SPIDCTL via Python IF to control operation. Table 6.2 and Table 6.3 list supported parameters/commands in SPIDCTL model.

Table 6.2: List of supported parameters

Parameter	Type	Default	Description
SPIDCTL_MessageLevel	string	fatal|error	Select debug message level (“fatal”, “error”, “warning”, “info”). One or more than levels can be connected by vertical bar. Example “fatal|error”: SCHEAP.SPIDCTL_MessageLevel("RH850.spidctl", "fatal|error")
SPIDCTL_DumpRegisterRW	bool	false	Dump register access information when registers are accessed. + false: Not dump register access information + true: Dump register access information Example: SCHEAP.SPIDCTL_DumpRegisterRW("RH850.spidctl"," false")
SPIDCTL_EnableRegisterMessage	bool	true	Dump info/warning/error message of register IF + false: Not dump info/warning/error message of register IF + true: Dump info/warning/error message of register IF Example SPIDCTL_EnableRegisterMessage("RH850.spidctl", "true")

Table 6.3: List of supported commands

Command	Type	Argument	Description
SPIDCTL_DumpStatusInfo	void	-	Dump the status information of the SPIDCTL. Example: SCHEAP.SPIDCTL_DumpStatusInfo("RH850.spidctl")
SPIDCTL_AssertReset	void	reset_name, start-time,	Assert and de-assert reset signal + std::string <reset_name>: name of reset signal

		period	("Reset") + double <start-time>: the time until asserting reset signal from current time. The unit is "ns" + double <period>: the time from asserting reset signal to de-assert it. The unit is "ns" Example: SCHEAP.SPIDCTL_AssertReset("RH850.spidctl","Reset",10,10)
SPIDCTL_SetCLKFreq	void	clock_name, freq, unit	Set frequency value to these blocks + std::string <clock_name>: name of clock signal ("PCLK") + sc_dt::uint64 <freq>: clock frequency + std::string <unit>: frequency unit ("Hz", "KHz", "MHz" or "GHz") Example: SCHEAP.SPIDCTL_SetCLKFreq("RH850.spidctl","PCLK",1000000,"Hz")
SPIDCTL_GetCLKFreq	void	clock_name	Get frequency value of these blocks + std::string <clock_name>: name of clock signal ("PCLK") Example: SCHEAP.SPIDCTL_GetCLKFreq("RH850.spidctl","PCLK")
SPIDCTL_ForceRegister	void	reg_name, value	Force register with setting value + std::string <reg_name>: name of register + unsigned int <value>: value which is set to register Example: SCHEAP.SPIDCTL_ForceRegister("RH850.spidctl", "BMSPIDMSK00", 0xFFFFFFFF)
SPIDCTL_ReleaseRegister	void	reg_name	Release register from force value + std::string <reg_name>: name of register Example: SCHEAP.SPIDCTL_ReleaseRegister("RH850.spidctl ", "BMSPIDMSK00")
SPIDCTL_WriteRegister	void	reg_name, value	Write a value to register + std::string <reg_name>: name of register + unsigned int <value>: value which is set to register Example: SCHEAP.SPIDCTL_WriteRegister("RH850.spidctl","BMSPIDMSK00",0x00000001)
SPIDCTL_ReadRegister	void	reg_name	Read a value from register + std::string <reg_name>: name of register Example: SCHEAP.SPIDCTL_ReadRegister("RH850.spidctl","BMSPIDMSK00")
SPIDCTL_ListRegister	void	-	Dump register names of model Example: SCHEAP.SPIDCTL_ListRegister("RH850.spidctl")
SPIDCTL_Help	void	type	Dump the direction how to use python interface parameters and commands + std::string <type>: "parameters" or "commands" Example: SCHEAP.SPIDCTL_Help("RH850.spidctl","parameters")

Note:
How to disable message in register IF use SPIDCLT_EnableRegisterMessage parameter.
Step 1: Not define -DREGIF_SC_REPORT when compile environment.
Step 2: In test pattern python file, disable register message before setting message level.
The SPIDCTL_EnableRegisterMessage() is required to set before SPIDCTL_MessageLevel().
If not, register messages in register IF will be dumped.

Example:
SCHEAP.SPIDCTL_EnableRegisterMessage("RH850.spidctl", "false")
SCHEAP.SPIDCTL_MessageLevel("RH850.spidctl", "info|error|warning|fatal")

6.5.Message style

6.5.1.Register RW messages

Table 6.4: Dump Register RW message description

Condition	This message is output when registers are accessed and parameter SPIDCTL_DumpRegisterRW is set “true”.
Output	This message is printed to standard output (console).
Format: Info: <hier_instance_name>: [<time>ps] REG [<reg_name>] R Size = <size> Addr = <reg_address> Data = <reg_value> Info: <hier_instance_name>: [<time>ps] REG [<reg_name>] W Size = <size> Addr = <reg_address> Data = <reg_value> : <old_value> => <new_value> **Example**: Info: SPIDCTL: [2900000 ps] REG [BMSPID00] R Size= 4 Addr= 0xFF0A8000 Data= 0x1 Info: SPIDCTL: [3270000 ps] REG [BMSPID00] W Size= 4 Addr= 0xFF0A8000 Data= 0x1 : 0x00 => 0x01	
Tag name	**Description**
hier_instance_name	Hierarchy instance name of SPIDCTL model is being used.
time	Simulation time
reg_name	Name of accessed register.
size	Register size.
address	Register address.
value	Register value.
old_value	Register's value before writing.
new_value	Register's value after writing.

6.5.2.Help messages

Table 6.5: Dump parameter help message description

Condition	This message is dumped out when command SPIDCTL_Help is called with “parameters” argument.
Output	This message is printed to standard output (console). The help message is used for Python Interface.
--- parameters --- SPIDCTL_MessageLevel (“SPIDCTL instance”, “fatal|error|warning|info”) Set debug message level (Default: fatal|error). SPIDCTL_DumpRegisterRW (“SPIDCTL instance”, “true/false”) Enable/disable dumping access register (Default: false). SPIDCTL_EnableRegisterMessage (“SPIDCTL instance”, “true/false”) Enable/disable info/warning/error message of register IF (Default: true).	

Table 6.6: Dump command help message description

Condition	This message is dumped out when command SPIDCTL_Help is called with “commands” argument.
Output	This message is printed to standard output (console). The help message is used for Python Interface.
--- commands --- SPIDCTL_DumpStatusInfo (“SPIDCTL instance”) Dump the status information of the SPIDCTL model. SPIDCTL_AssertReset (“SPIDCTL instance”, “reset_name”, start_time, period) Assert and de-assert reset signal. SPIDCTL_SetCLKFreq (“SPIDCTL instance”, “clock_name”, freq, “unit”) Set clock frequency value of model. SPIDCTL_GetCLKFreq (“SPIDCTL instance”, “clock_name”) Get clock frequency value of model. SPIDCTL_ForceRegister (“SPIDCTL instance”, “reg_name”, value) Force a register with setting value. SPIDCTL_ReleaseRegister (“SPIDCTL instance”, “reg_name”) Release a register from force value. SPIDCTL_WriteRegister (“SPIDCTL instance”, “reg_name”, value) Write a value to a register. SPIDCTL_ReadRegister (“SPIDCTL instance”, “reg_name”) Read value from a register. SPIDCTL_ListRegister (“SPIDCTL instance”) Dump register names of model.	

6.5.3.Error and debugging messages

Table 6.7: Error and debugging message description

Condition	This message's kind is output when error occurs or some important events occur. It's depended on setting of parameter SPIDCTL_MessageLevel. Detailed conditions are described in the “Description” column of Table 6.8
Output	This message is printed to standard output (console).
Format: <severity>: <hier_instance_name>: [<time><unit>] <Message content> **Example**: Info: SPIDCTL: [2900000 ps] Cannot write during reset period	
Tag name	**Description**
Severity	Kind of message's severity
hier_instance_name	Hierarchy instance name of SPIDCTL model is being used.
Time	Simulation time
Unit	Simulation time's unit
Message content	Message content (message list is described in Table 6.8)

6.5.4.List of error and debugging messages

Table 6.8: Error and debugging message in SPIDCTL model

No.	Severity	Message	Description
1	error	Invalid access address 0x%08X with access size %d bytes	User access to model's register with wrong aligned address %08X: address %d: number of access bytes
2	error	Invalid access address 0x%08X	Users access the model with invalid address %08X: address
3	error	Cannot find the object of <model name> class	Users call PythonIF with wrong object of <model name> class
4	error	<command name> has too much arguments	Dump this message when number of input arguments is incorrect.

5	error	<command name> command needs an argument [true/false]	Dump this message when input argument is missed.
6	error	Reading access size to %s at address 0x%08X is wrong: %d byte(s).	Users read the value from registers with invalid size. %s: register name %8X: address %d: number of bytes
7	error	Writing access size to %s at address 0x%08X is wrong: %d byte(s).	Users write the value to registers with invalid size. %8X: address %d: number of bytes
8	warning	The <model name>_<command name> has not any arguments	Users call PythonIF of <model name>_<command name> with any argument. The argument should be not input.
9	warning	The arguments of <command name> are wrong	Users call PythonIF of <command name> with wrong arguments
10	warning	The name (%s) of <model name>_Help argument is wrong (commands or parameters).	Users call <model name>_Help command with invalid argument. It must be “commands” or “parameters”
11	warning	Register name <register_name> is invalid	Dump this message when register name is invalid.
12	warning	Should read all bit in a register	Dump this message when users read register with access size less than supported access size.
13	warning	%s forbids to write %X	Dump this message when a read-only bit is written value. %s: bit name %X: written value.
14	warning	Cannot access register when clock is 0	Register is written when clock PCLK is 0
15	warning	Cannot read register when clock is 0 or in reset state.	Register is read when clock PCLK is 0 or Reset port is asserted
16	warning	Cannot write 1 to reserved bit.	Users write 1 to reserved area in a register.
17	warning	Cannot write during reset period	Dump this message when users write to register during reset period of Reset.
18	warning	%s is blocked writing from Bus I/F.	Access write to register which it is locked by <model name>_ForceRegister.

19	warning	Clock name (%s) is invalid.	Dump this message when users call <model_name>_SetCLKFreq or <model_name>_GetCLKFreq with wrong clock name. %s: invalid clock name.
20	warning	Invalid argument: <command name> <argument name>	Users call <command name> with invalid argument
21	warning	The reset name (%s) is wrong. It should be Reset	Users call AssertReset with wrong reset name.
22	warning	The software reset of Reset is called in the reset operation of the model. So it is ignored.	The reset command is called in reset operation of model.
23	info	Initialize %s (%08x)	This message is dumped when initializing value of register during reset period. %s: register's name. %08X: register value.
24	info	%s frequency is zero	Dump this message when any clock is set frequency 0.
25	info	Reset period of %s is over.	Reset period of a reset name which is set by AssertReset is over.
26	info	The model is reset by AssertReset command of %s.	Users call AssertReset command for a reset name and reset operation is accepted after specified time at first argument of <model name>_AssertReset.
27	info	The model will be reset (%s) for %f ns after %f ns.	Users call AssertReset command for Reset with start reset time and reset period.
28	info	The reset port %s is asserted.	Users activate Reset port
29	info	The reset port %s is de-asserted.	Users deactivate Reset port
30	info	%s frequency is %0.0f %s	Users set frequency to PCLK clock.
31	info	<model name>_EnableRegisterMessage %s	This message is dumped when users call <model name>_EnableRegisterMessage without argument.
32	warning	Cannot write to %s register while key code is disabled.	This message is dumped when write SPID Mask register with protection mode is ON.
33	warning	Cannot write to BMSPIDMSK%02d when BMSPIDMSKLOCK%02d is locked	This message is dumped when write SPID Mask register when it is locked by corresponding SPID Mask Lock register.
34	warning	Invalid setting. The default initial value will be used.	This message is dumped when configuration file setting is invalid.

6.6.Define macro and template

(1) In this design, there is no macro.

(2) In this design, there is no template.

7.Flow diagrams

7.1.Sequence diagram of reset port handling

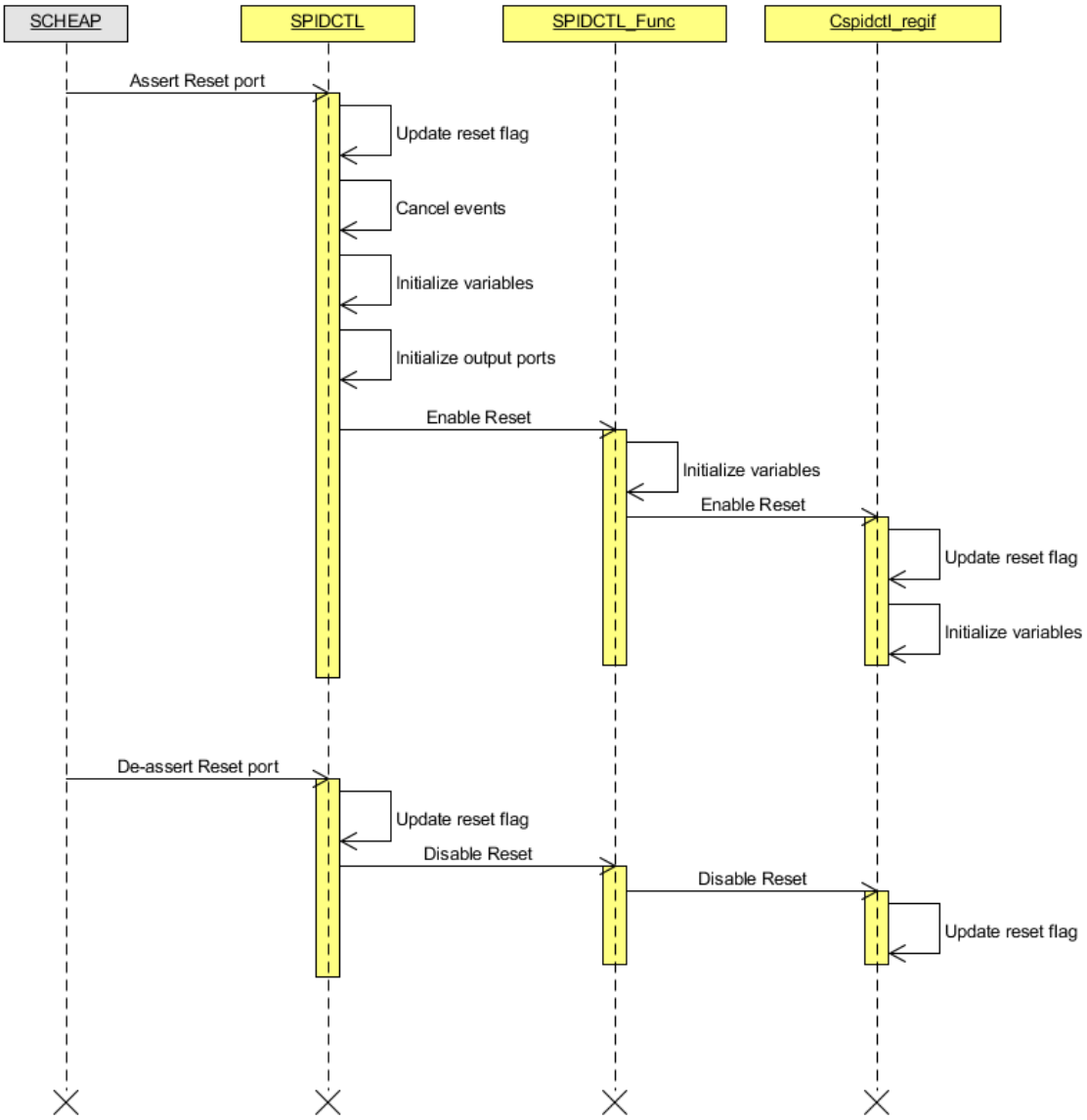

Figure 7-1: Sequence diagram of reset port handling

Explanation:

(1) When reset port is asserted: cancel events, initialize variables, initialize output ports, update reset flag value

(2) When reset port is de-asserted: update reset flag value

7.2.Sequence diagram of clock port handling

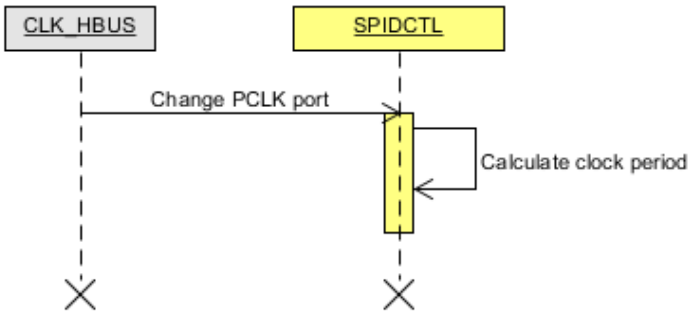

Figure 7-2: Sequence diagram of clock handling

Explanation:

(1) When the value of clock source port PCLK is changed, clock period is re-calculated.

7.3.Sequence diagram of Python IF handling

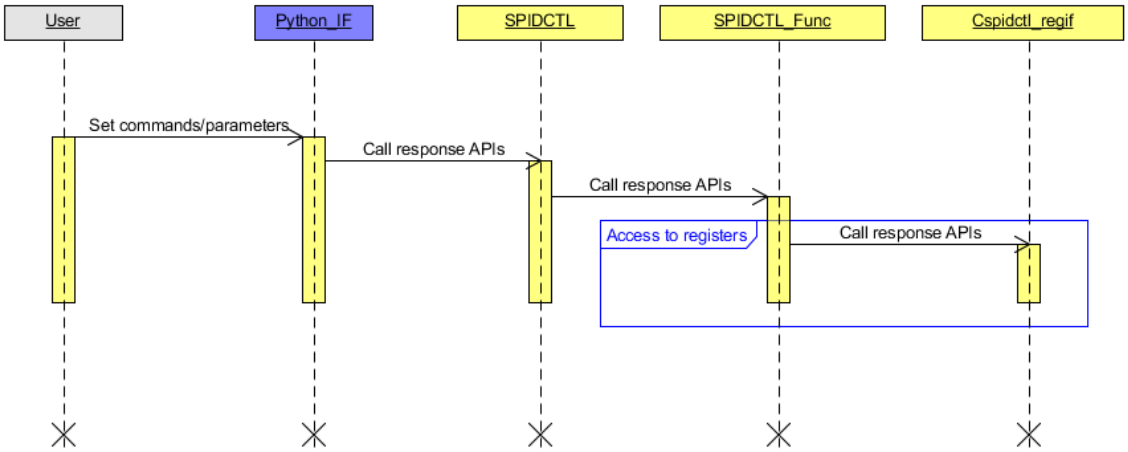

Figure 7-3: Sequence diagram of Python IF handling

Explanation:

(1) When user sets a Python IF command or parameter, the corresponding API is called.

7.4.Sequence diagram of register reading

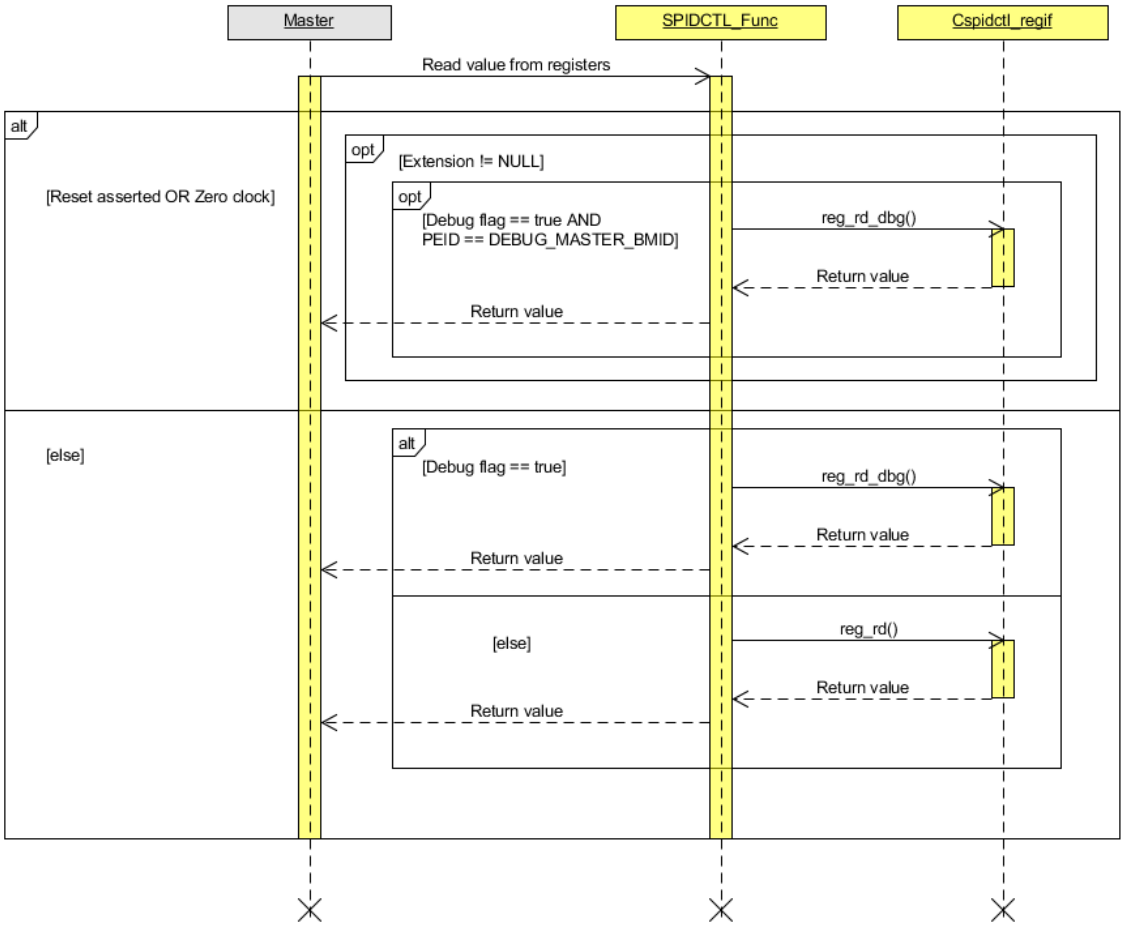

Figure 7-4: Sequence diagram of register reading

Explanation:

(1) If the current condition is reset port asserted or zero clock, and extension is not NULL, only software debugger (Debug flag == true and PEID == DEBUG_MASTER_BMID) can read the value of register by reg_rd_dbg() function (DEBUG_MASTER_BMID = 0xFF).

(2) Otherwise, the corresponding function is called based on debug flag.

7.5.Sequence diagram of register writing

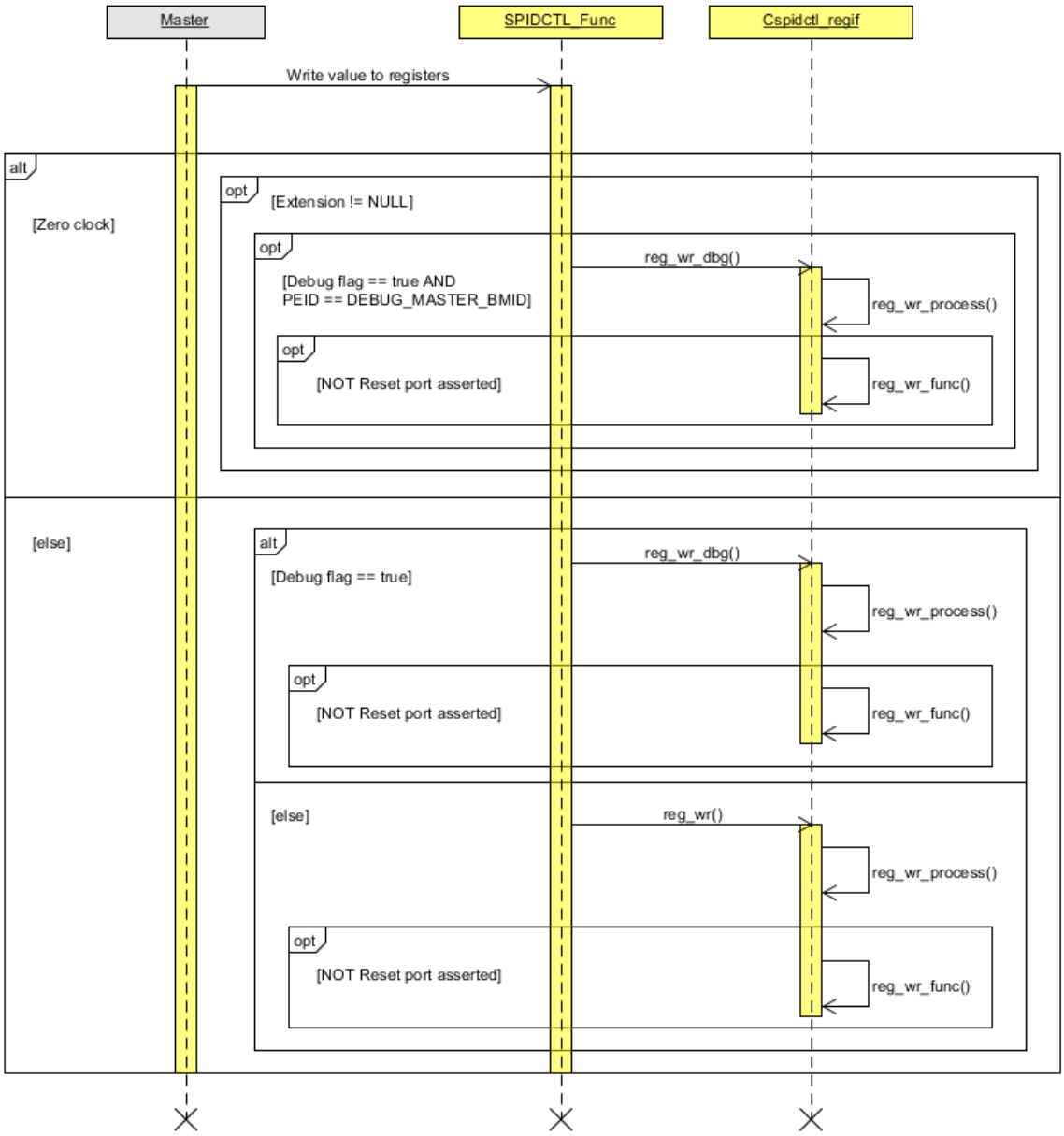

Figure 7-5: Sequence diagram of register writing

Explanation:

(1) If the current condition is zero clock, and extension is not NULL, only Software debugger (Debug flag == true and PEID == DEBUG_MASTER_BMID) can write to register (DEBUG_MASTER_BMID = 0xFF).

(2) Else, the corresponding function is called based on debug flag.

(3) If not reset port asserted, the value is written to register.

7.6.Flow of callback function of register SPID writing

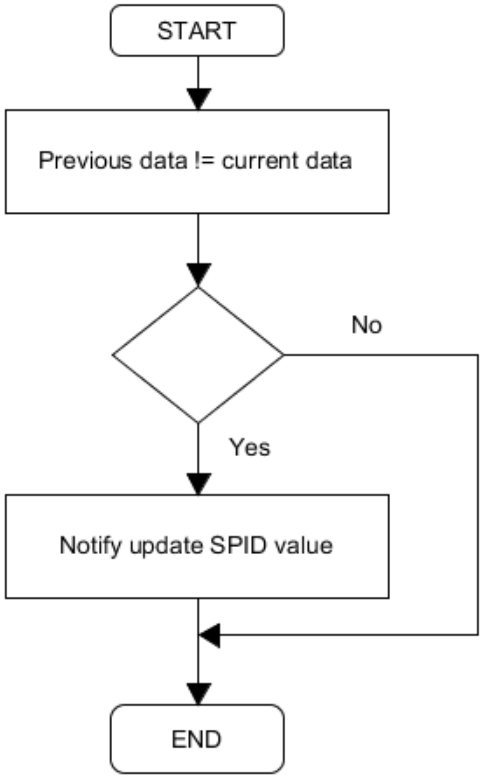

Figure 7-6: Flow of callback function of register SPID writing

Explanation:

(1) If previous data is different current data, notify process update SPID value to update model's output ports.

(2) Else, model does nothing.

7.7.Flow of callback function of register SPID Mask writing

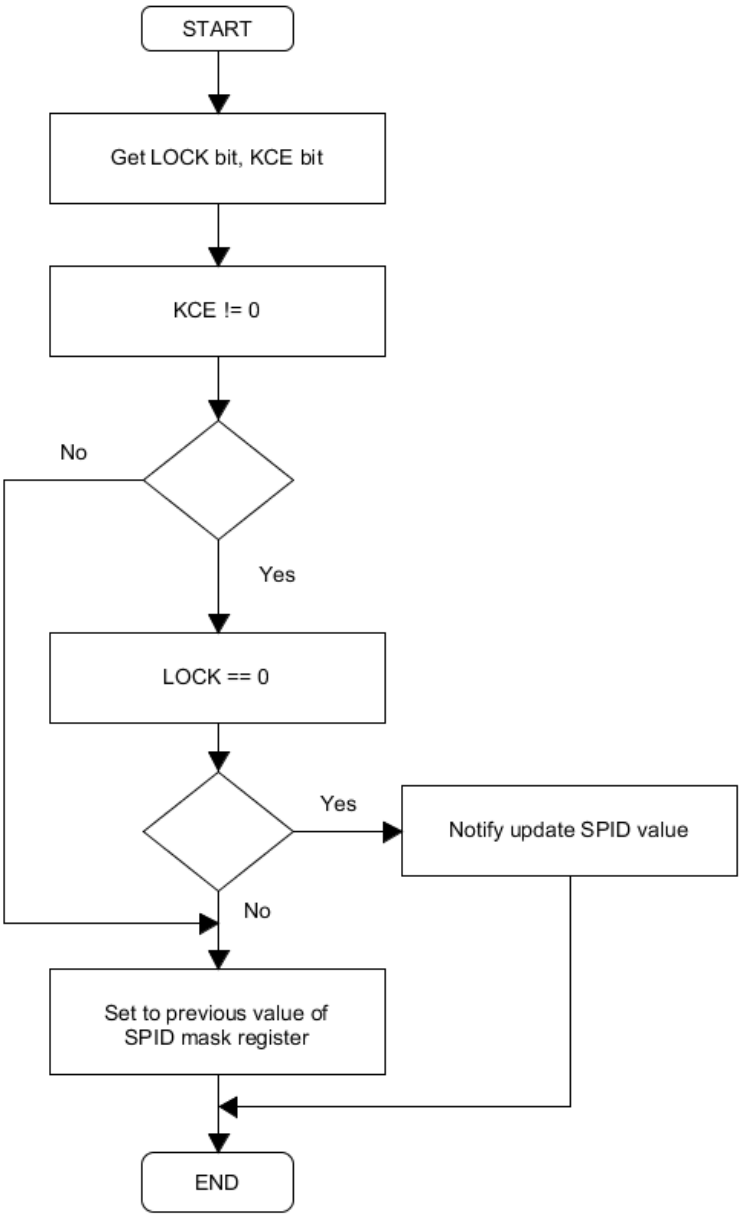

Figure 7-7: Flow of callback function of register SPID Mask writing

Explanation:

(1) If LOCK bit is zero and KCE bit is non-zero, notify process update SPID value to update model's output ports.

(2) Else, set to previous value of SPID mask register.

7.8.Flow of callback function of register SPID Mask Lock writing

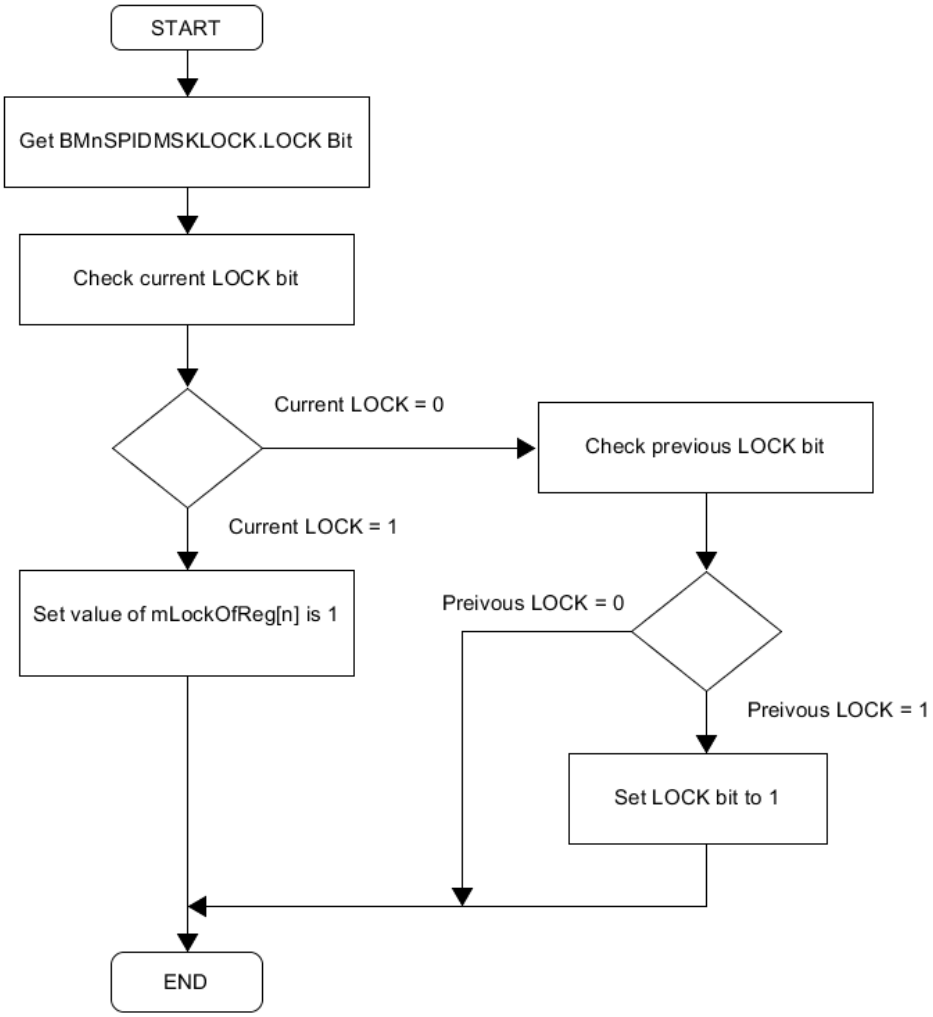

Figure 7-8: Flow of callback function of register SPID Mask Lock writing

Explanation:

(1) Figure above describes operations when model receives a writing access to LOCK bit.

(2) If current LOCK bit is written with value is 1, model sets value of mLockOfReg [n] is 1.

(3) Else, set LOCK bit to 1 in case previous LOCK bit is 1, else model does nothing.

(4) Only Reset port of model can reset LOCK bit and mLockOfReg.

7.9.Flow of callback function of register SPID Key code protection writing

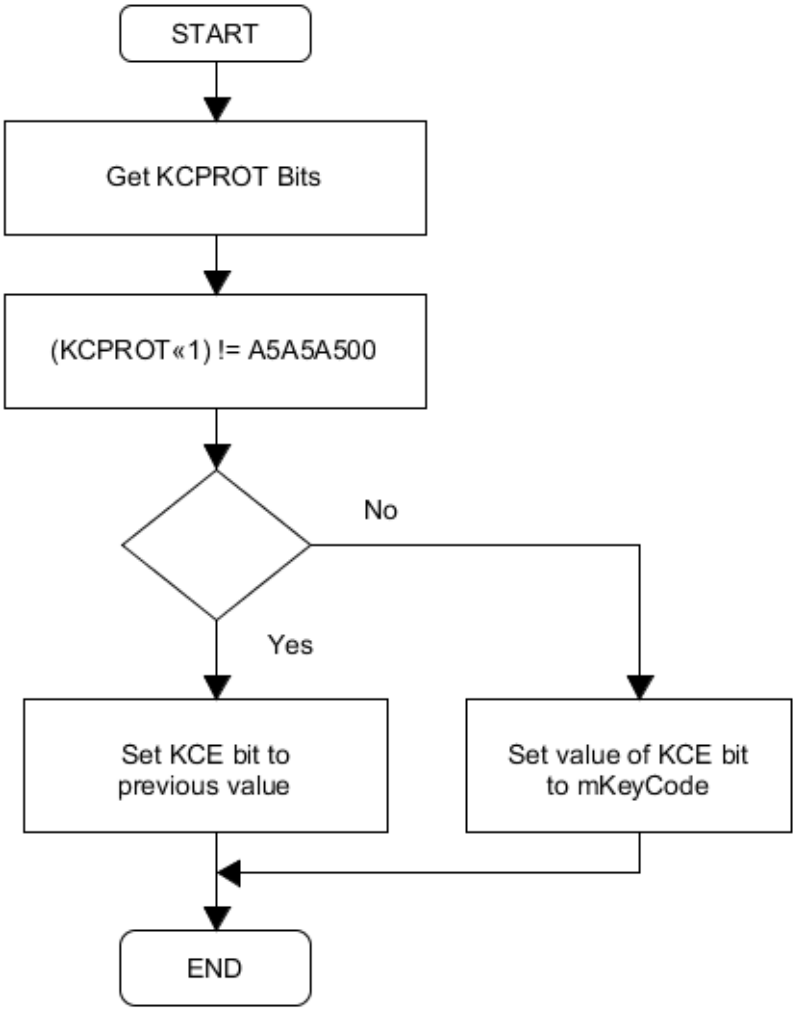

Figure 7-9: Flow of callback function of register SPID Key code protection writing

Explanation:

(1) Figure above describes operations when model receives a writing access to KCE bit.
(2) After write KCE bit, callback function will set KCE bit to previous value in case user write invalid KCPROT value.
(3) Else, model sets value of KCE bit to mKeyCode variable.

7.10.Flow process update SPID value.

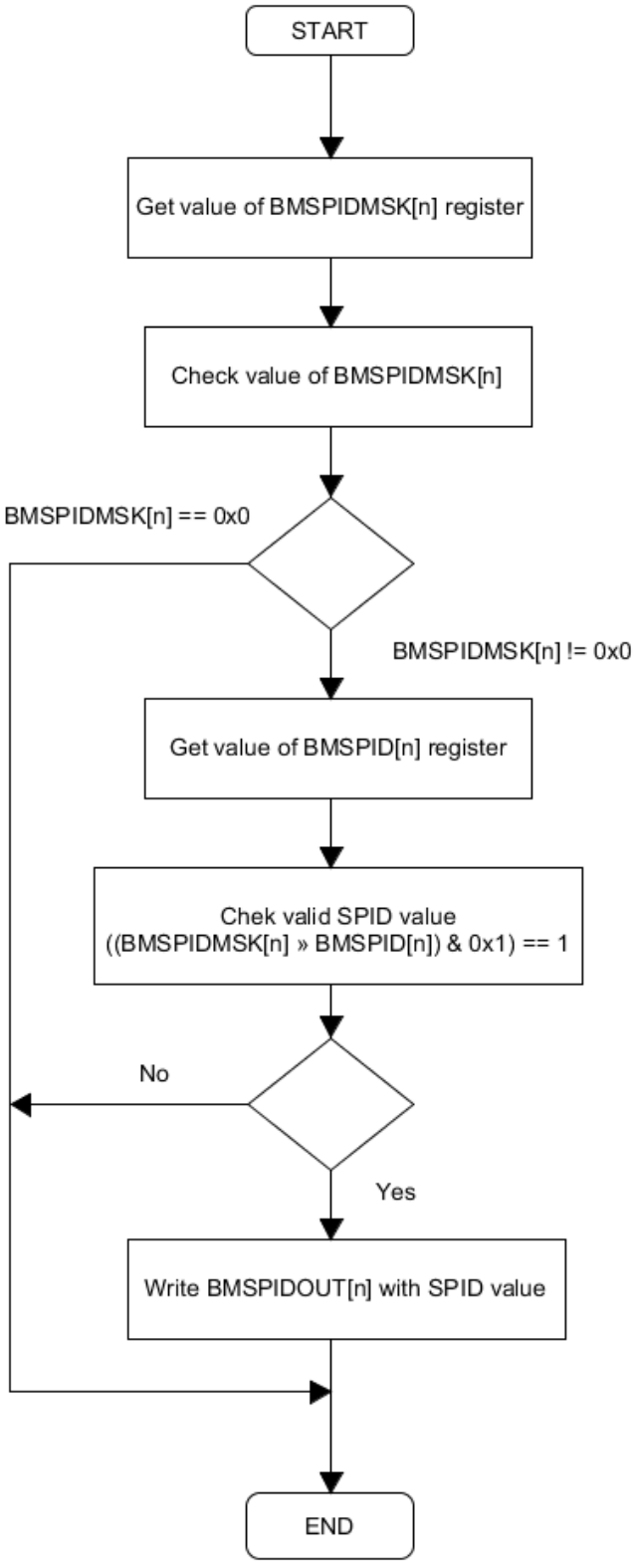

Figure 7-10: Flow process update SPID value.

Explanation:

(1) Figure above describes operations when model receives notify update SPID value.

(2) If BMSPIDMSK is zero, model does nothing.

(3) Else, check SPID whether valid or not. If valid, write SPID value into output port BMSPIDOUT. Else, model does nothing.

7.11.Flow of reset

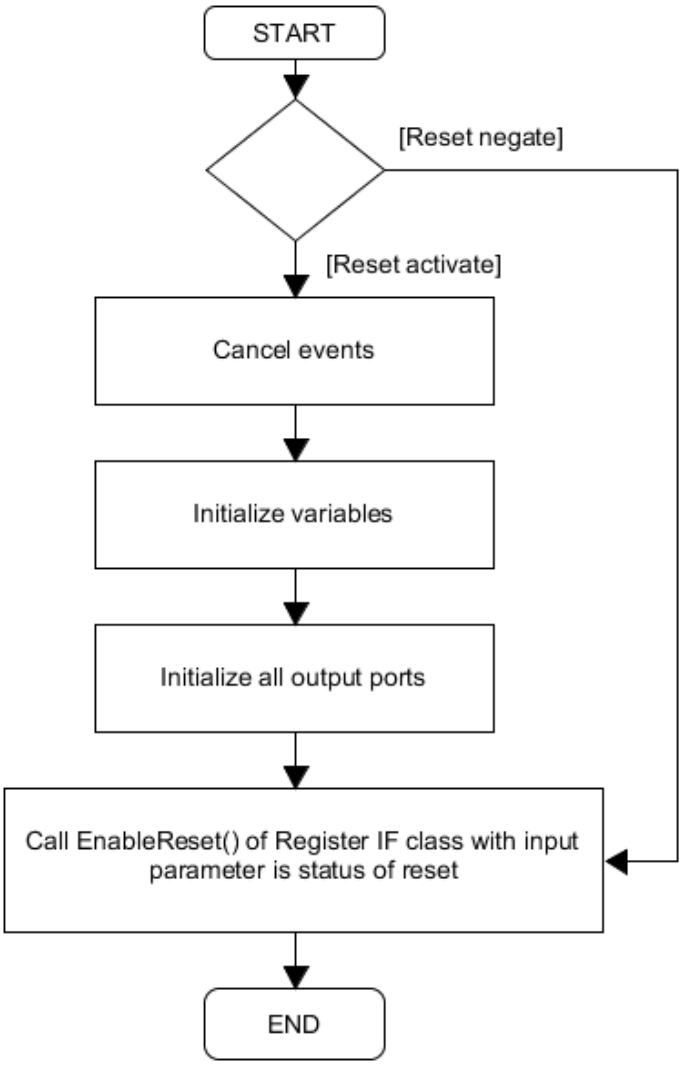

Figure 7-11: Flow of reset

Explanation:

(1) Figure above describes operation of SPIDCTL model when the Reset port is changed value.

(2) If Reset port is activated:

(2.1) All events related to AssertReset command are canceled. Events related to operations are canceled.

(2.2) Most of variables (except variable related to reset/clock) are initialized.

(2.3) Output port is written initial value.

(2.4) The EnableReset() function of register I/F class is called with input parameter is the status of reset.

7.12.Flow of Python IF

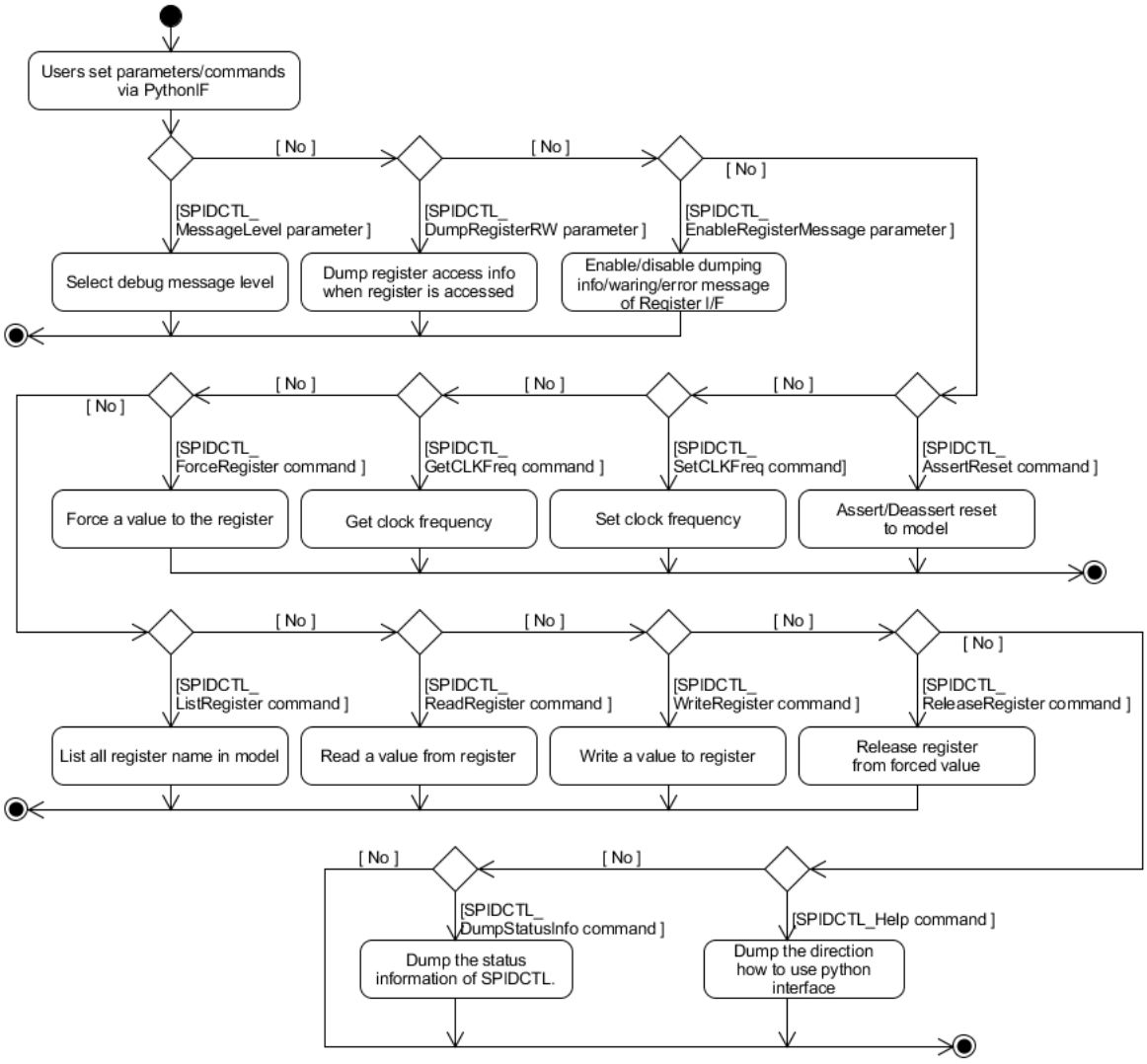

Figure 7-12: Flow of Python IF

Explanation:

(1) Figure above describes operation of Python IF when users set/call parameters/commands of SPIDCTL model.

(2) Refer to chapter **Commands and parameters** for more detail about functions of them.

8.Function description

8.1.List of public/private function in SPIDCTL class

Table 8.1: List of public functions in SPIDCTL class

No.	Function name	Description
1	SPIDCTL (sc_module_name name, const unsigned int rLatency, const unsigned int wLatency, const char *config_file);	Constructor of SPIDCTL class
2	~SPIDCTL ();	Destructor of SPIDCTL class
3	void SetMessageLevel (const std::string msg_lv);	Set message level (fatal, error, warning, info)
4	void DumpRegisterRW (const std::string is_enable);	Enable/ Disable dumping message when users access registers
5	void EnableRegisterMessage (const std::string is_enable);	Enable/disable info/warning/error message of register IF
6	void DumpStatusInfo (void);	Dump value of all status registers.
7	void AssertReset (const std::string reset_name, const double start_time, const double period);	Assert reset by software
8	void SetCLKFreq (const std::string clock_name, const sc_dt::uint64 freq, const std::string unit);	Set clock value and clock unit
9	void GetCLKFreq (const std::string clock_name);	Get clock value
10	void ForceRegister (const std::string reg_name, const unsigned int reg_value);	Force value to register
11	void ReleaseRegister (const std::string reg_name);	Release forced value after forcing registers
12	void WriteRegister (const std::string reg_name, const unsigned int reg_value);	Write value to registers by software
13	void ReadRegister (const std::string reg_name);	Read value of register by software
14	void ListRegister (void);	List registers name of whole model
15	void Help (const std::string type);	Dump help message of all parameters or commands

Table 8.2: List of private functions in SPIDCTL class

No.	Function name	Description
1	void HandlePCLKSignalMethod (void);	Handle PCLK clock
2	void HandleResetSignalMethod (void);	Handle Reset signal
3	void HandleResetHardMethod(void);	Process reset function when reset signal is active
4	void HandleResetCmdMethod (void);	Process reset function when reset command is active
5	void CancelResetCmdMethod (void);	Cancel reset function when reset command is active
6	void EnableReset (const bool is_active);	Enable/ Disable reset
7	void Initialize(void);	Initialize variables

8	void CancelEvents(void);	Cancel events when reset is assert
9	bool CheckClockPeriod(void);	Check a clock period is valid or equals to 0
10	bool GetResetStatus(void);	Get reset status
11	void NotifyUpdateSPIDValue(unsigned int Index);	Notify SPID and SPID mask registers are written
12	void HandleSPIDCTLMethod(unsigned int spid_id);	Handle update SPID value of model
13	bool IsKeyCodeEnable(void);	Get the KCE bit status in model.
14	void SetKeyCodeEnable(bool isEnable);	Set value the KCE bit status into model.
15	bool IsMaskLocked(unsigned int RegIndex);	Get the LOCK bit status in model.
16	void SetMaskLocked(unsigned int RegIndex, bool isLocked);	Set status of LOCK bit into model.
17	bool GetArrCfgValue(std::string s, unsigned int *arr)	Return array of interger from given string.
18	void ReadConfigFile(const char *config_file)	Get parameter from config file.
19	std::string GetStrBetweenTwoStr(const std::string &s, const std::string &start_delim, const std::string &stop_delim)	Get string between two string.

8.2.List of public/private function in SPIDCTL_Func class

Table 8.3: List of public functions in SPIDCTL_Func class

No.	Function name	Description
1	SPIDCTL_Func (std::string name, SPIDCTL_AgentController *SPIDCTLAgentControllerPtr);	Constructor of SPIDCTL_Func class
2	~SPIDCTL_Func ();	Destructor of SPIDCTL_Func class.
3	void EnableReset (const bool is_active);	Enable/ Disable reset
4	void RegisterHandler (const std::vector<std::string>& args);	Handle commands access to register.
5	void read (unsigned int offsetAddress, TlmBasicPayload& trans, BusTime_t* t, bool debug);	Overwrite virtual functions of BusSlaveFuncIf class
6	void write (unsigned int offsetAddress, TlmBasicPayload& trans, BusTime_t* t, bool debug);	Overwrite virtual functions of BusSlaveFuncIf class
7	void DumpStatusInfo(void);	Dump register info
8	unsigned int GetSPIDReg_SPIDBit(unsigned int index);	Get SPID value at SPID Register's index
9	unsigned int GetSPIDMskReg(unsigned int index);	Get SPID mask value at SPID Mask Register's index

Table 8.4: List of private functions in SPIDCTL_Func class

No.	Function name	Description

1	void Initialize(void);	Initialize variables
2	void cb_BMSPID_SPID(RegCBstr str);	Callback function of SPID registers.
3	void cb_BMSPIDMSK_SPIDMSK(RegCBstr str);	Callback function of SPID mask registers
4	void cb_BMSPIDMSKLOCK_LOCK(RegCBstr str);	Callback function of LOCK bit in SPID mask lock registers.
5	void cb_SPIDKCPROT_KCE (RegCBstr str);	Callback function of KCE bit in SPIDKCPROT register.
6	void InitializeBMSPIDn(void);	Set initial value for BMSPIDn registers

8.3.List of public function in SPIDCTL_AgentController class

Table 8.5: List of public functions in SPIDCTL_AgentController class

No.	Function name	Description
1	SPIDCTL_AgentController(std::string name): rvc_common_model(name);	Constructor of SPIDCTL_AgentController class.
2	virtual ~ SPIDCTL_AgentController(void);	Destructor of SPIDCTL_AgentController class.
3	virtual bool CheckClockPeriod(void) = 0;	Virtual function is used to check clock period value
4	virtual bool GetResetStatus(void) = 0;	Virtual function is used to check reset status
5	virtual void NotifyUpdateSPIDValue (unsigned int Index) = 0;	Virtual function is used to notify SPID and SPID mask registers are written
6	virtual bool IsKeyCodeEnable(void) = 0;	Virtual function is used to get the KCE bit status in model.
7	virtual void SetKeyCodeEnable(bool isEnable) = 0;	Virtual function is used to set value the KCE bit status into model.
8	virtual bool IsMaskLocked(unsigned int RegIndex) = 0;	Virtual function is used to get the LOCK bit status in model.
9	virtual void SetMaskLocked(unsigned int RegIndex, bool isLocked) = 0;	Virtual function is used to set value the LOCK bit status into model.

9.Limitation

(1) Table below list all limitation of this model (comparing with HW).

Table 9.1: Limitation of model

No.	HWM	Model	HW Reference
1	SPIDCTL module have I/F to output BMnSPIDMSK values to IPs which have internal-SPID-register.	SPIDCTL module does NOT have I/F to output BMnSPIDMSK values to IPs which have internal-SPID-register. This point will be update in next SPIDCTL maintenance.	[9] 55.5.3.3. List of Registers [5] 44.5.3.2. List of Registers [6] 40.5.2.2. List of Registers

10. Appendix

Initial value of SPID

Initial value of SPID	Bus master	Register location
0	CPU0	CPU0
1	CPU1	CPU1
2	CPU2	CPU2
3	CPU3*3	CPU3
4	CPU4*3*4	CPU4
5	CPU5*3*4	CPU5
6	Reserved	-
7	Reserved	-
8	Reserved	-
9	Reserved	-
10	GTM (Cluster 0 to Cluster 9)*8	SPID module(n=10-17)*7
11	EMU3S0 (EMU3S core and PSEQ)*4*5*6	SPID module(n=18, 19)*7
12	EMU3S1 (EMU3S core and PSEQ)*4*5*6	SPID module(n=20, 21)*7
13	R-Switch*3*4*5	SPID module(n=22)*7
14	ACEU0 *3*4	SPID module(n=23)*7
15	ACEU1 *3*4*5*6	SPID module(n=24)*7
16	Reserved	-
17	Reserved	-
18	RHSIF1*3*4*5	SPID module(n=27)*7
19	RHSIF0*3	SPID module(n=28)*7
20	Reserved	-
21	Reserved	-
22	Reserved	-
23	FlexRay0	SPID module(n=32)*7
24	ETND1 *3*4*6	SPID module(n=33)*7
25	ETND0*6	SPID module(n=34)*7
26	ICUMHB (ICUM core, AES and ICUM DMA)	SPID module(n=35-37)*7
27	sDMAC1	sDMAC1
28	sDMAC0	sDMAC0

Initial value of SPID	Bus master	Register location
29	DTS*1	DTS
30	DFP*3	DFP
31	MAU	MAU
	DFP*3, EMU3S*4*5*6 ICUM (debug master)	SPID module(n=43)*7

Figure 10-1: U2B Initial value of SPID

Initial value of SPID	Bus master	Register location
0	CPU0	CPU0
1	CPU1	CPU1
2	CPU2	CPU2
3	CPU3	CPU3
4 to 16	Reserved	-
17	Gb Ether	SPID module
18	Reserved	-
19	RHSIF0	SPID module
20	Reserved	-
21	Reserved	-
22	FlexRay1	SPID module
23	FlexRay0	SPID module
24	Fast Ether	SPID module
25	ICUM_AES0	SPID module
26	ICUMHA	SPID module
27	sDMAC1	sDMAC1
28	sDMAC0	sDMAC0
29	DTS*1	DTS
30	Reserved	-
31	MAU	MAU

Figure 10-2: U2A Initial value of SPID

Initial value of SPID	Bus master	Register location
0	CPU0	CPU0
1	CPU1	CPU1
2	CPU2	CPU2
3	CPU3	CPU3
4	CPU4*2	CPU4
5	CPU5*2	CPU5
6 to 18	Reserved	—
19	RHSIF0	SPID module
20	Reserved	—
21	HSSPI0	SPID module
22	FlexRay1*2	SPID module
23	FlexRay0	SPID module
24	Ethernet	SPID module
25	Reserved	—
26	ICU-M	SPID module
27	sDMAC1	sDMAC1
28	sDMAC0	sDMAC0
29	DTS*1	DTS
30	Reserved	—
31	MAU	MAU

Figure 10-3: E2x Initial value of SPID

Note: The default initial value in this model when config_file setting is not input is:

- BMnSPID = n when n = [0:31]
- BMnSPID = 31 when n > 31

Revision History				
Rev.	**Modified Contents**	**Approved by RVC**	**Checked**	**Created**
1.0	Create new.	Chuong Le Jan/15/2020	Khoa Nguyen Jan/09/2020	Anh Nguyen Jan/03/2020
1.1	**Update ticket Redmine #117906** Figure 3-1 : - Add note (*1) **Update after coding** Table 8.2 : - Add arguments for function NotifyUpdateSPIDValue. Table 8.5. - Add arguments for function NotifyUpdateSPIDValue.	Chuong Le Feb/04/2020	Khoa Nguyen Jan/17/2020	Anh Nguyen Jan/16/2020
1.2	**Update for U2B HWUM v0.40** - Add description for U2B in Cover page, summary, header and Model summary. - Update Note in Block diagram. - Add GTMSPIDAS0, GTMSPIDAS1 and update Note in Table 4.1 - Update Note, Reset port Active, and BMSPIDOUT range and Initial in Table 5.1 - Update rvc_common_model inherit, add Common.h and remove SPIDCTL_cmdif.h in Figure 6.1. - Add rvc_common_model.cpp, Common.h, and remove SPIDCTL_cmdif.h and OSCI2.h inTable 6.1 . - Add functions No. 17, 18, 19 in Table 8.2 - Add function No. 6 in Table 8.4. - Update Copyright - Update Reqtify ID and Reference.	Chuong Le Nov/17/2021	Duong Phan Nov/12/2021	Minh Ha Nov/10/2021
1.3	- Update message No. 15, remove message No. 13 and add message No. 34 in Table 6.8 - Add limitation in Table 9.1	Chuong Le Nov/22/2021	Duong Phan Nov/20/2021	Minh Ha Nov/18/2021