Internal Specification

Development of ADC Interrupt Router (AIR) model for RH850/E2x

(v1.4)

Summary:

This document describes the Detail Design Specification of ADC Interrupt Router (AIR) model for RH850/E2x.

Reference Manuals

No.	Title name	Document number	Description	Path
1	SC-HEAP_E3 PYTHON I/F function specification (v2.0)	LLWEB-00105192 MSS-SG-12-0062-02	The document describes how to use python interface (***File***: SC-HEAP_E3 Python IF_t.pdf)	-
2	REQ-SLD-16004_E2x_ADCH_models (Rev 1.0)	-	RH850 E2x/ADCH model development Requirement Specification	**DMS:** Documents/1. General Documents/010_ENG/140_FrontEnd/Project/01_SLD/2_SLD_Project/Model_Documents/01_Project_Document_Management/REQ/2016
3	NoSecurity_r01uh0641ej0010_rh850e2x-fcc1(rev 0.10)	-	RH850/E2x-FCC1 Hardware manual	-
4	INT-SLD-16006_refman.pdf	-	Documentation of Ex2/AIR model source code	**DMS:** Documents/1. General Documents/010_ENG/140_FrontEnd/Project/01_SLD/2_SLD_Project/Model_Documents/01_Project_Document_Management/INT/2016
5	E2x_FCC2/AIR target specification	-	Target specification of E2x_FCC2/AIR (***File***: E34_ADCH.docx)	**Server:** /shsv/sld/ipp/From_RT/E2x_FCC2/

Table of Contents

Index of Figures

Index of Tables

1. Model summary

- The AIR is a model that selects signals from all A/D related resources, and outputs interrupt requests and DMA requests.
- This model is developed for RH850/E2x_FCC1 and E2x_FCC2 products.
- Registers of model can be accessed via TLM common class.
- Both loosely time mode (LT) and approximately time (AT) mode are supported.
- This model supports little endian mode as the endian of APB bus interface.

Note: *Hereafter, ADC Interrupt Router model is simply called "AIR".*

2. Supported features

Table 2-1: Feature of AIR model

Feature	Description		HWM chapter
	Hardware	Model	
Max frequency of APB bus clock (CLK_LSB clock)	40 MHz	Unlimited frequency. There is no setting condition.	*ref[3]/15.2/Table 15.4*
Read/Write registers	Use APB bus interface	Use TLM common interface	*ref[3]/34.1.3/Table 34.30*
Reset	Hardware reset (assert/deassert reset signal)	Hardware reset (assert/deassert reset signal) Software reset (set by python command AssertReset) (*)	*ref[3]/34.1.5/Table 34.56*
Main function	AIR selects interrupt requests from resource 0 (output of ADCH, DSADC and CADC) and resource 1 (output of ABFG).	<-	*ref[3]/34.11.1.1* *ref[5]/34.12*
	AIR selects DMA requests from resource 0 0 (output of ADCH, DSADC and CADC) and resource 1 (output of ABFG).	<-	

Notes:

- *(*) These commands are described in Chapter 6.4*
- *The symbol "<-" means that these features are supported as description in the hardware manual.*

3. Block diagram

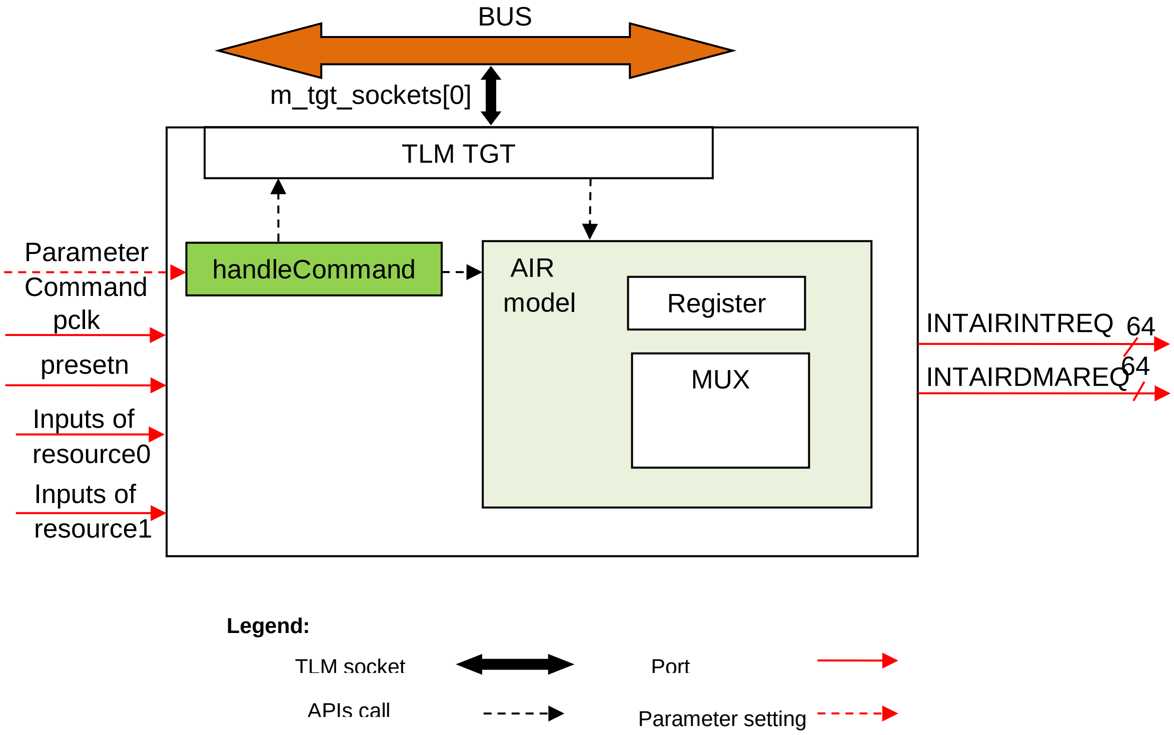


Figure 3-1: Block diagram of AIR model

Explanation:

(1) AIR model has one target socket. Data from BUS is transferred to AIR model via TLM target interface (target socket).

(2) AIR model has input ports: pclk, presetn, Input if resource0, Input of resource1. When AIR model receive signal from resource 0 or signal from resource 1, according to registers value the AIR model output interrupt/DMA request. Please refer to Chapter 5.3, 5.4 for detail description of input ports of resource 0/1.

(3) handleCommand is used for setting output messages, period of clock, and so on. Refer to chapter 6.4 for more detail.

(4) AIR model outputs are 64 interrupt request and 64 DMA request.

(5) When presetn signal active (Low), some registers, ports and all control variables are reset. Refer Table 4-2 for list of registers and corresponding reset port.

4. List of implemented registers

Table 4-1: List of AIR registers

Register name	Address offset	Read/ Write Access size (bit)	R/W	Bit position	Bit name	Explanation
AIR_ISELR0 Interrupt Request Select Register 0	0x00	8/16/32	R/W	31:0	ISEL31 ~ ISEL0	Interrupt Request Select These bits select from resource 0 and resource 1.
AIR_ISELR1 Interrupt Request Select Register 1	0x04	8/16/32	R/W	31:0	ISEL63 ~ ISEL32	Interrupt Request Select These bits select from resource 0 and resource 1.
AIR_ DSELR0 DMA Request Select Register 0	0x08	8/16/32	R/W	31:0	DSEL31 ~ DSEL0	DMA Request Select These bits select from resource 0 and resource 1.
AIR_ DSELR1 DMA Request Select Register 1	0x0C	8/16/32	R/W	31:0	DSEL63 ~ DSEL32	DMA Request Select These bits select from resource 0 and resource 1.

Note: *All registers described in HWM (ref[3]/Chapter 34.11.2, ref[5]/Chapter 34.12.2) are supported in model.*

Table 4-2: Reset value of AIR registers

Register	Reset port	Reset value
AIR_ISELR0	presetn	0x0000 0000
AIR_ISELR1	presetn	0xFF80 0000
AIR_ DSELR0	presetn	0x0F00 0000
AIR_ DSELR1	presetn	0xFF80 0000

5. Port behavior

5.1. List of implemented ports

Table 5-1: List of AIR ports

Port name	I/O	Type	Initial	Active	Synchronous clock	Description
CLK_LSB	In	sc_in<sc_dt::uint64>	-	-	-	Register access clock
presetn	In	sc_in<bool>	-	Low	-	Reset port
ADMPXI0	In	sc_in<bool>	-	Low	-	MPX interrupt from ADC channel 0
ADMPXI1	In	sc_in<bool>	-	Low	-	MPX interrupt from ADC channel 1
ADMPXI2	In	sc_in<bool>	-	Low	-	MPX interrupt from ADC channel 2
ADMPXI3	In	sc_in<bool>	-	Low	-	MPX interrupt from ADC channel 3
ADI00	In	sc_in<bool>	-	Low	-	Scan group 0 end interrupt from ADC channel 0
ADI01	In	sc_in<bool>	-	Low	-	Scan group 1 end interrupt from ADC channel 0
ADI02	In	sc_in<bool>	-	Low	-	Scan group 2 end interrupt from ADC channel 0
ADI03	In	sc_in<bool>	-	Low	-	Scan group 3 end interrupt from ADC channel 0
ADI04	In	sc_in<bool>	-	Low	-	Scan group 4 end interrupt from ADC channel 0
ADI10	In	sc_in<bool>	-	Low	-	Scan group 0 end interrupt from ADC channel 1
ADI11	In	sc_in<bool>	-	Low	-	Scan group 1 end interrupt from ADC channel 1
ADI12	In	sc_in<bool>	-	Low	-	Scan group 2 end interrupt from ADC channel 1
ADI13	In	sc_in<bool>	-	Low	-	Scan group 3 end interrupt from ADC channel 1
ADI14	In	sc_in<bool>	-	Low	-	Scan group 4 end interrupt from ADC channel 1
ADI20	In	sc_in<bool>	-	Low	-	Scan group 0 end interrupt from ADC channel 2
ADI21	In	sc_in<bool>	-	Low	-	Scan group 1 end interrupt from ADC channel 2
ADI22	In	sc_in<bool>	-	Low	-	Scan group 2 end interrupt from ADC channel 2
ADI23	In	sc_in<bool>	-	Low	-	Scan group 3 end interrupt from ADC channel 2
ADI24	In	sc_in<bool>	-	Low	-	Scan group 4 end interrupt from ADC channel 2
ADI30	In	sc_in<bool>	-	Low	-	Scan group 0 end interrupt

						from ADC channel 3
ADI31	In	sc_in<bool>	-	Low	-	Scan group 1 end interrupt from ADC channel 3
ADI32	In	sc_in<bool>	-	Low	-	Scan group 2 end interrupt from ADC channel 3
ADI33	In	sc_in<bool>	-	Low	-	Scan group 3 end interrupt from ADC channel 3
ADI34	In	sc_in<bool>	-	Low	-	Scan group 4 end interrupt from ADC channel 3
ADE0	In	sc_in<bool>	-	Low	-	A/D error interrupt Upper/lower limit error interrupt from ADC channel 0
ADE1	In	sc_in<bool>	-	Low	-	A/D error interrupt Upper/lower limit error interrupt from ADC channel 1
ADE2	In	sc_in<bool>	-	Low	-	A/D error interrupt Upper/lower limit error interrupt from ADC channel 2
ADE3	In	sc_in<bool>	-	Low	-	A/D error interrupt Upper/lower limit error interrupt from ADC channel 3
DSADEm (1*)	In	sc_in<bool>	-	Low	-	A/D error interrupt Upper / lower limit error interrupt from DSADCm
DSADIm (1*)	In	sc_in<bool>	-	Low	-	A/D conversion end interrupt from DSADCm
CADE00	In	sc_in<bool>	-	Low	-	A/D error interrupt Upper / lower limit error interrupt from CADC00
CADI00	In	sc_in<bool>	-	Low	-	A/D conversion end interrupt from CADC00
ASI0	In	sc_in<bool>	-	Low	-	Ch0 integration end interrupt from ASF
ASI1	In	sc_in<bool>	-	Low	-	Ch1 integration end interrupt from ASF
ASI2	In	sc_in<bool>	-	Low	-	Ch2 integration end interrupt from ASF
ASI3	In	sc_in<bool>	-	Low	-	Ch3 integration end interrupt from ASF
ASI4	In	sc_in<bool>	-	Low	-	Ch4 integration end interrupt from ASF
ASI5	In	sc_in<bool>	-	Low	-	Ch5 integration end interrupt from ASF
ASI6	In	sc_in<bool>	-	Low	-	Ch6 integration end interrupt from ASF
ASI7	In	sc_in<bool>	-	Low	-	Ch7 integration end interrupt from ASF
ASI8	In	sc_in<bool>	-	Low	-	Ch8 integration end interrupt from ASF
ASI9	In	sc_in<bool>	-	Low	-	Ch9 integration end interrupt from ASF
ASI10	In	sc_in<bool>	-	Low	-	Ch10 integration end interrupt from ASF

ASI11	In	sc_in<bool>	-	Low	-	Ch11 integration end interrupt from ASF
ASI12	In	sc_in<bool>	-	Low	-	Ch12 integration end interrupt from ASF
ASI13	In	sc_in<bool>	-	Low	-	Ch13 integration end interrupt from ASF
ASI14	In	sc_in<bool>	-	Low	-	Ch14 integration end interrupt from ASF
ASI15	In	sc_in<bool>	-	Low	-	Ch15 integration end interrupt from ASF
BFPw (1*)	In	sc_in<bool>	-	Low	-	Boundary flag pulse w interrupt from ABFG
INTAIRINTREQn (n = 0 ~ 63)	Out	sc_out<bool>	false	Low	CLK_LSB (2*)	Interrupt Request (0 ~ 63)
INTAIRDMAREQn (n = 0 ~ 63)	Out	sc_out<bool>	false	Low	CLK_LSB (2*)	DMA request (0 ~ 63)

Notes: - (1*)

w: 0 -> 47 (E2x_FCC1: w[40->47] are not used);

m: 00, 10, 11, 12, 13, 14, 15, 20, 21, 22 (E2x_FCC1: m[14,15,21,22] are not used);

Refer to Chapter 6.3/step 3 for the connecting of each product in detail.

- (2*) The output timing is depend on the timing input changed (after 1 CLK_LSB clock). Please refer to Figure 5-1 for more detail.

+ If input synchronizes with CLK_LSB, output synchronizes with CLK_LSB

+ If input does not synchronize with CLK_LSB, output does not synchronize with CLK_LSB

5.2. Timing of output ports

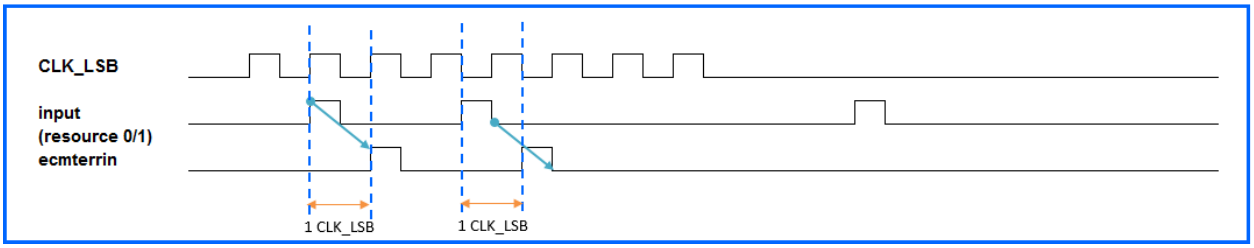


Figure 5-1: Timing of output ports

Explanation:

- When the input port active, according to setting of registers, the output port is active or not after 1 CLK_LSB clock.
- When CLK_LSB clock frequency is zero,
  - If input port is active, model does not detect the change of input port, no info/warning/error message is dumped.
  - If the output port is active, model keep the output port active, does not de-active it.
- User should active reset port presetn after set CLK_LSB to 0 when model is operating.

5.3. Interrupt select requests

Table 5-2: Interrupt request select

No.	Interrupt request	Register name	Bit name	Input ports as resource 0	Input ports as resource 1
0	INTAIRINTREQ0	AIR_ISELR0	ISEL[0]	ADMPXI0	
1	INTAIRINTREQ1	AIR_ISELR0	ISEL[1]	ADMPXI2	
2	INTAIRINTREQ2	AIR_ISELR0	ISEL[2]	ADMPXI1	BFP18
3	INTAIRINTREQ3	AIR_ISELR0	ISEL[3]	ADMPXI3	BFP19
4	INTAIRINTREQ4	AIR_ISELR0	ISEL[4]	ADI00	
5	INTAIRINTREQ5	AIR_ISELR0	ISEL[5]	ADI01	
6	INTAIRINTREQ6	AIR_ISELR0	ISEL[6]	ADI02	
7	INTAIRINTREQ7	AIR_ISELR0	ISEL[7]	ADI03	
8	INTAIRINTREQ8	AIR_ISELR0	ISEL[8]	ADI04	
9	INTAIRINTREQ9	AIR_ISELR0	ISEL[9]	ADI20	
10	INTAIRINTREQ10	AIR_ISELR0	ISEL[10]	ADI21	BFP46
11	INTAIRINTREQ11	AIR_ISELR0	ISEL[11]	ADI22	BFP47
12	INTAIRINTREQ12	AIR_ISELR0	ISEL[12]	ADI23	
13	INTAIRINTREQ13	AIR_ISELR0	ISEL[13]	ADI24	
14	INTAIRINTREQ14	AIR_ISELR0	ISEL[14]	ADI10	BFP20
15	INTAIRINTREQ15	AIR_ISELR0	ISEL[15]	ADI11	BFP21
16	INTAIRINTREQ16	AIR_ISELR0	ISEL[16]	ADI12	BFP22
17	INTAIRINTREQ17	AIR_ISELR0	ISEL[17]	ADI13	BFP23
18	INTAIRINTREQ18	AIR_ISELR0	ISEL[18]	ADI14	BFP24
19	INTAIRINTREQ19	AIR_ISELR0	ISEL[19]	ADI30	BFP25
20	INTAIRINTREQ20	AIR_ISELR0	ISEL[20]	ADI31	BFP26
21	INTAIRINTREQ21	AIR_ISELR0	ISEL[21]	ADI32	BFP27
22	INTAIRINTREQ22	AIR_ISELR0	ISEL[22]	ADI33	BFP28
23	INTAIRINTREQ23	AIR_ISELR0	ISEL[23]	ADI34	BFP29
24	INTAIRINTREQ24	AIR_ISELR0	ISEL[24]	ADE0	-
25	INTAIRINTREQ25	AIR_ISELR0	ISEL[25]	ADE2	-
26	INTAIRINTREQ26	AIR_ISELR0	ISEL[26]	ADE1	BFP13
27	INTAIRINTREQ27	AIR_ISELR0	ISEL[27]	ADE3	BFP14
28	INTAIRINTREQ28	AIR_ISELR0	ISEL[28]	DSADE00	
29	INTAIRINTREQ29	AIR_ISELR0	ISEL[29]	DSADE10	
30	INTAIRINTREQ30	AIR_ISELR0	ISEL[30]	DSADE20	
31	INTAIRINTREQ31	AIR_ISELR0	ISEL[31]	DSADE12	BFP15
32	INTAIRINTREQ32	AIR_ISELR1	ISEL[0]	DSADE13	BFP16
33	INTAIRINTREQ33	AIR_ISELR1	ISEL[1]	DSADE11	BFP17
34	INTAIRINTREQ34	AIR_ISELR1	ISEL[2]	DSADE15	BFP9
35	INTAIRINTREQ35	AIR_ISELR1	ISEL[3]	DSADE14	BFP10
36	INTAIRINTREQ36	AIR_ISELR1	ISEL[4]	DSADE22	BFP11
37	INTAIRINTREQ37	AIR_ISELR1	ISEL[5]	DSADE21	BFP12
38	INTAIRINTREQ38	AIR_ISELR1	ISEL[6]	CADE00	
39	INTAIRINTREQ39	AIR_ISELR1	ISEL[7]	ASI0	BFP30

40	INTAIRINTREQ40	AIR_ISELR1	ISEL[8]	ASI1	BFP31
41	INTAIRINTREQ41	AIR_ISELR1	ISEL[9]	ASI2	BFP32
42	INTAIRINTREQ42	AIR_ISELR1	ISEL[10]	ASI3	BFP33
43	INTAIRINTREQ43	AIR_ISELR1	ISEL[11]	ASI4	BFP34
44	INTAIRINTREQ44	AIR_ISELR1	ISEL[12]	ASI5	BFP35
45	INTAIRINTREQ45	AIR_ISELR1	ISEL[13]	ASI6	BFP36
46	INTAIRINTREQ46	AIR_ISELR1	ISEL[14]	ASI7	BFP37
47	INTAIRINTREQ47	AIR_ISELR1	ISEL[15]	ASI8	BFP38
48	INTAIRINTREQ48	AIR_ISELR1	ISEL[16]	ASI9	BFP39
49	INTAIRINTREQ49	AIR_ISELR1	ISEL[17]	ASI10	BFP40
50	INTAIRINTREQ50	AIR_ISELR1	ISEL[18]	ASI11	BFP41
51	INTAIRINTREQ51	AIR_ISELR1	ISEL[19]	ASI12	BFP42
52	INTAIRINTREQ52	AIR_ISELR1	ISEL[20]	ASI13	BFP43
53	INTAIRINTREQ53	AIR_ISELR1	ISEL[21]	ASI14	BFP44
54	INTAIRINTREQ54	AIR_ISELR1	ISEL[22]	ASI15	BFP45
55	INTAIRINTREQ55	AIR_ISELR1	ISEL[23]		BFP0
56	INTAIRINTREQ56	AIR_ISELR1	ISEL[24]		BFP1
57	INTAIRINTREQ57	AIR_ISELR1	ISEL[25]		BFP2
58	INTAIRINTREQ58	AIR_ISELR1	ISEL[26]		BFP3
59	INTAIRINTREQ59	AIR_ISELR1	ISEL[27]		BFP4
60	INTAIRINTREQ60	AIR_ISELR1	ISEL[28]		BFP5
61	INTAIRINTREQ61	AIR_ISELR1	ISEL[29]		BFP6
62	INTAIRINTREQ62	AIR_ISELR1	ISEL[30]		BFP7
63	INTAIRINTREQ63	AIR_ISELR1	ISEL[31]		BFP8

5.4. DMA select requests

Table 5-3: DMA request select

No.	DMA request	Register name	Bit name	Input ports as resource 0	Input ports as resource 1
0	INTAIRDMAREQ0	AIR_DSELR0	DSEL[0]	ADMPXI0	
1	INTAIRDMAREQ1	AIR_DSELR0	DSEL[1]	ADMPXI2	
2	INTAIRDMAREQ2	AIR_DSELR0	DSEL[2]	ADMPXI1	BFP18
3	INTAIRDMAREQ3	AIR_DSELR0	DSEL[3]	ADMPXI3	BFP19
4	INTAIRDMAREQ4	AIR_DSELR0	DSEL[4]	ADI00	
5	INTAIRDMAREQ5	AIR_DSELR0	DSEL[5]	ADI01	
6	INTAIRDMAREQ6	AIR_DSELR0	DSEL[6]	ADI02	
7	INTAIRDMAREQ7	AIR_DSELR0	DSEL[7]	ADI03	
8	INTAIRDMAREQ8	AIR_DSELR0	DSEL[8]	ADI04	
9	INTAIRDMAREQ9	AIR_DSELR0	DSEL[9]	ADI20	
10	INTAIRDMAREQ10	AIR_DSELR0	DSEL[10]	ADI21	BFP46
11	INTAIRDMAREQ11	AIR_DSELR0	DSEL[11]	ADI22	BFP47
12	INTAIRDMAREQ12	AIR_DSELR0	DSEL[12]	ADI23	
13	INTAIRDMAREQ13	AIR_DSELR0	DSEL[13]	ADI24	

14	INTAIRDMAREQ14	AIR_DSELR0	DSEL[14]	ADI10	BFP20
15	INTAIRDMAREQ15	AIR_DSELR0	DSEL[15]	ADI11	BFP21
16	INTAIRDMAREQ16	AIR_DSELR0	DSEL[16]	ADI12	BFP22
17	INTAIRDMAREQ17	AIR_DSELR0	DSEL[17]	ADI13	BFP23
18	INTAIRDMAREQ18	AIR_DSELR0	DSEL[18]	ADI14	BFP24
19	INTAIRDMAREQ19	AIR_DSELR0	DSEL[19]	ADI30	BFP25
20	INTAIRDMAREQ20	AIR_DSELR0	DSEL[20]	ADI31	BFP26
21	INTAIRDMAREQ21	AIR_DSELR0	DSEL[21]	ADI32	BFP27
22	INTAIRDMAREQ22	AIR_DSELR0	DSEL[22]	ADI33	BFP28
23	INTAIRDMAREQ23	AIR_DSELR0	DSEL[23]	ADI34	BFP29
24	INTAIRDMAREQ24	AIR_DSELR0	DSEL[24]		BFP9
25	INTAIRDMAREQ25	AIR_DSELR0	DSEL[25]		BFP10
26	INTAIRDMAREQ26	AIR_DSELR0	DSEL[26]		BFP11
27	INTAIRDMAREQ27	AIR_DSELR0	DSEL[27]		BFP12
28	INTAIRDMAREQ28	AIR_DSELR0	DSEL[28]	DSADI00	
29	INTAIRDMAREQ29	AIR_DSELR0	DSEL[29]	DSADI10	
30	INTAIRDMAREQ30	AIR_DSELR0	DSEL[30]	DSADI20	
31	INTAIRDMAREQ31	AIR_DSELR0	DSEL[31]	DSADI12	BFP17
32	INTAIRDMAREQ32	AIR_DSELR0	DSEL[0]	DSADI13	
33	INTAIRDMAREQ33	AIR_DSELR0	DSEL[1]	DSADI11	
34	INTAIRDMAREQ34	AIR_DSELR0	DSEL[2]	DSADI15	BFP13
35	INTAIRDMAREQ35	AIR_DSELR0	DSEL[3]	DSADI14	BFP14
36	INTAIRDMAREQ36	AIR_DSELR0	DSEL[4]	DSADI22	BFP15
37	INTAIRDMAREQ37	AIR_DSELR0	DSEL[5]	DSADI21	BFP16
38	INTAIRDMAREQ38	AIR_DSELR0	DSEL[6]	CADI00	
39	INTAIRDMAREQ39	AIR_DSELR0	DSEL[7]	ASI0	BFP30
40	INTAIRDMAREQ40	AIR_DSELR0	DSEL[8]	ASI1	BFP31
41	INTAIRDMAREQ41	AIR_DSELR1	DSEL[9]	ASI2	BFP32
42	INTAIRDMAREQ42	AIR_DSELR1	DSEL[10]	ASI3	BFP33
43	INTAIRDMAREQ43	AIR_DSELR1	DSEL[11]	ASI4	BFP34
44	INTAIRDMAREQ44	AIR_DSELR1	DSEL[12]	ASI5	BFP35
45	INTAIRDMAREQ45	AIR_DSELR1	DSEL[13]	ASI6	BFP36
46	INTAIRDMAREQ46	AIR_DSELR1	DSEL[14]	ASI7	BFP37
47	INTAIRDMAREQ47	AIR_DSELR1	DSEL[15]	ASI8	BFP38
48	INTAIRDMAREQ48	AIR_DSELR1	DSEL[16]	ASI9	BFP39
49	INTAIRDMAREQ49	AIR_DSELR1	DSEL[17]	ASI10	BFP40
50	INTAIRDMAREQ50	AIR_DSELR1	DSEL[18]	ASI11	BFP41
51	INTAIRDMAREQ51	AIR_DSELR2	DSEL[19]	ASI12	BFP42
52	INTAIRDMAREQ52	AIR_DSELR2	DSEL[20]	ASI13	BFP43
53	INTAIRDMAREQ53	AIR_DSELR2	DSEL[21]	ASI14	BFP44
54	INTAIRDMAREQ54	AIR_DSELR2	DSEL[22]	ASI15	BFP45
55	INTAIRDMAREQ55	AIR_DSELR2	DSEL[23]		BFP0
56	INTAIRDMAREQ56	AIR_DSELR2	DSEL[24]		BFP1
57	INTAIRDMAREQ57	AIR_DSELR2	DSEL[25]		BFP2

58	INTAIRDMAREQ58	AIR_DSELR2	DSEL[26]		BFP3
59	INTAIRDMAREQ59	AIR_DSELR2	DSEL[27]		BFP4
60	INTAIRDMAREQ60	AIR_DSELR2	DSEL[28]		BFP5
61	INTAIRDMAREQ61	AIR_DSELR3	DSEL[29]		BFP6
62	INTAIRDMAREQ62	AIR_DSELR3	DSEL[30]		BFP7
63	INTAIRDMAREQ63	AIR_DSELR3	DSEL[31]		BFP8

6. Direction for users

6.1. File structures

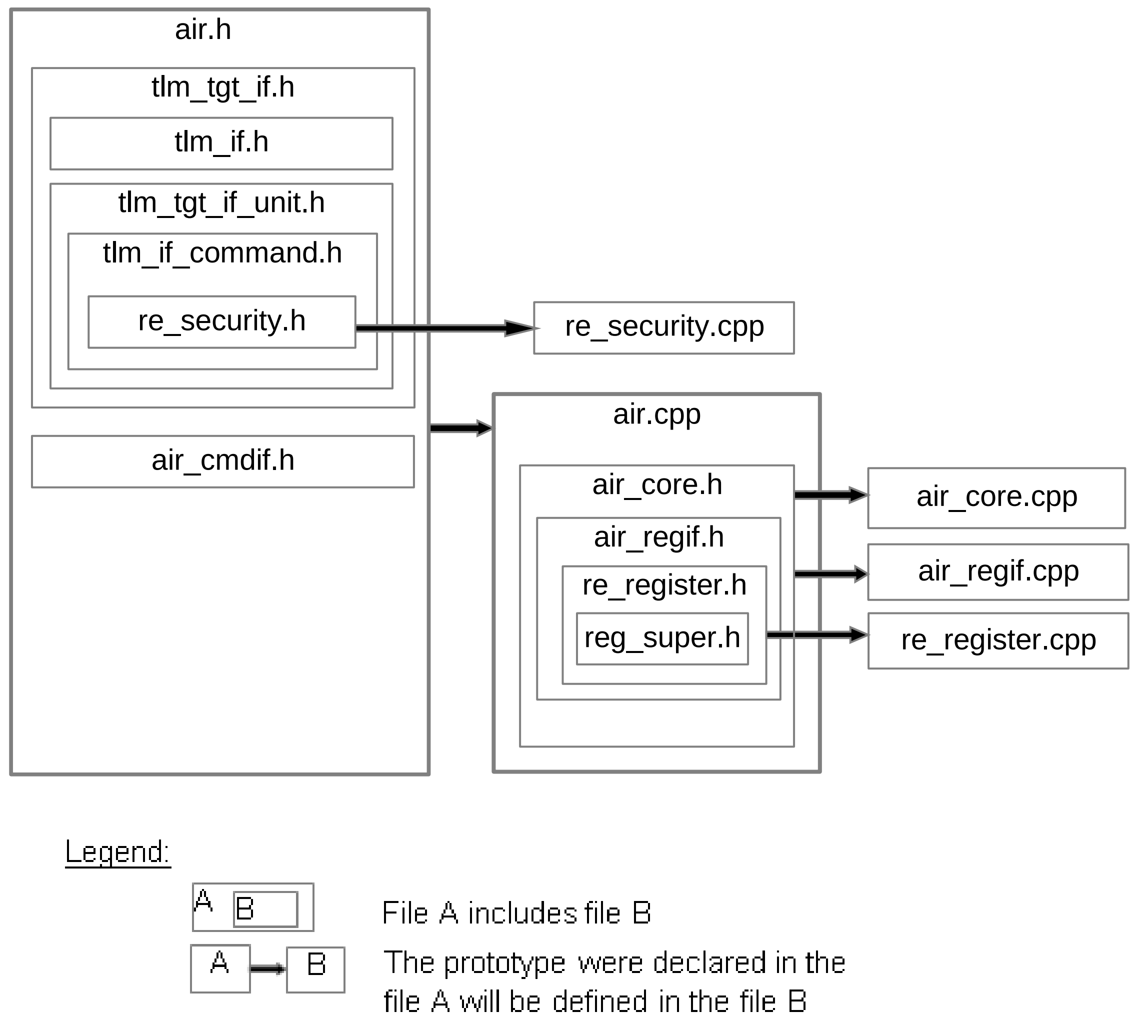


Figure 6-1: File structure of AIR model

Table 6-1: File description

No.	File name	CVS tag	Developed/ reused	Description
1	re_register.h	v2013_05_28	Reused	Header file of the re_register class.
2	re_register.cpp		Reused	Implement the attributes and the operations of common register class.
3	reg_super.h		Reused	General class for models to access to the memory array.
4	tlm_tgt_if.h	v2016_08_11_b_frm_v2014_04_02	Reused	Header file of the tlm_tgt_if class.
5	tlm_if.h		Reused	Header file of the tlm_if class.
6	tlm_tgt_if_unit.h		Reused	Header file of the tlm_tgt_if_unit class.
7	tlm_if_command.h		Reused	Header file of the tlm_if_command class.
8	air_info.txt	v2016_08_31	Developed	Input file of register IF generator for AIR model.
9	air_regif.h		Generated*	Header file of Register IF of AIR model.
10	air_regif.cpp		Generated*	Implementation file of Register IF of AIR model.
11	air.h		Developed	Header file of AIR model.
12	air.cpp		Developed	Implementation file of AIR model.
13	air_core.h		Developed	Header file of AIR core functions.
14	air_core.cpp		Developed	Implementation file of AIR core functions.
18	air_cmdif.h		Generated*	Command interface of AIR model.
22	PY_AIR.h		Generated*	Header file of Python IF of AIR model.
23	PY_AIR.cpp		Generated*	Implementation file of Python IF of AIR model.
24	re_security.h	v100419	Reused	Additional file of tlm_ini_if class and tlm_tgt_if class.
25	re_security.cpp		Reused	Additional file of tlm_ini_if class and tlm_tgt_if class.

Notes:

- Files air_regif.h/.cpp, air_regif.h/.cpp are generated from Register IF Generator (v2014_10_07). File air_cmdif.h are generated from Command IF Generator (v2015_02_12).

6.2. Input/Output file

There is no input or output file.

6.3. How to connect Verification Environment

There are 3 basic steps to connect AIR model to a verification environment.

- Step 1: Declare an instance of AIR class: Cair (<module_name>)
- Step 2: Bind the target socket named “m_tgt_sockets[0]”.
- Step 3: Bind the input and output ports listed in Table 5.1.
    - E2x_FCC1: The input ports (DSADEm, DSADIm, BFPw with m = [14, 15, 21, 22] and w = [40 -> 47]) are unused ports. Other input and output ports are used.
    - E2x_FCC2: All input and output ports are used.
    - For unused ports, the connection is requested. But, they should be connected to unused signals only.

6.4. Commands and parameters

Refer to ref[1]/Chapter 6.4 for supported commands and parameters.

Table 6-2: List of parameters

Parameter	Type	Default	Configure method	Description
MessageLevel (*)	string	fatal| error	Python IF	Select debug message level (“fatal”, “error”, “warning” and “info”). One or more than levels can be connected by vertical bar (Example “fatal|error”)
DumpRegisterRW (**)	string	false	Python IF	Enable/disable dumping access register. + false: Not dump register access information + true: Dump register access information

Notes:

- *(*) The setting value MessageLevel is not affected when REGIF_SC_REPORT macro is defined.(refer to ticket #23710 on Redmine).*

- *(**) The message belong to dumping register information is not effected by setting of MessageLevel parameter but DumpRegisterRW parameter.*

Table 6-3: List of commands

Command	Type	Argument	Configure method	Description
AssertReset	void	reset_name, start-time, period	Python IF	Assert and deassert reset signal + std::string <rst_name>: name of reset signal (“presetn”) + 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"
SetCLKFreq	void	clk_name, freq, unit	Python IF	Set clock frequency value + std::string <clk_name>: name of clock signal ("CLK_LSB") + sc_dt::uint64 <freq>: clock frequency + std::string <unit>: frequency unit ("Hz", "KHz", "MHz" or "GHz")
help	void	type	Python IF	Dump the direction how to use python interface parameters and commands + std::string <type>: "parameters" or "commands"

How to use: Below example describes how to use commands/parameters of python interface.

```
SCHEAP.setFreq(100,"MHz")
SCHEAP.MessageLevel("RH850.air","info|error|warning|fatal")
SCHEAP.SetCLKfreq("RH850.air","CLK_LSB",2, "MHz")
SCHEAP.GetCLKfreq("RH850.air","CLK_LSB")
SCHEAP.WriteRegister("RH850.air","AIR_ISELR0",0x0000FFFF)
SCHEAP.sc_start(1000)
```

Figure 6-2: An example of python interface usage

6.5. Message style

6.5.1. Register RW messages

Table 6-4: Dumping Register RW message description

<table>
<tr><td><b>Condition</b></td><td>This message is output when AIR registers are accessed and parameter DumpRegisterRW is set "true".</td></tr>
<tr><td><b>Output</b></td><td>This message's kind is printed to standard output (console).</td></tr>
<tr><td colspan="2"><b>Format</b>: 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>

<b>Example</b>:
Info: AIR: [2900000 ps] REG [AIR_ISELR0] R Size= 4 Addr= 0XFF460000 Data= 0x00000000
Info: AIR: [3270000 ps] REG [AIR_ISELR0] W Size= 4 Addr= 0XFF460000 Data= 0x0000FFFF : 0x0000000 => 0x0000FFFF</td></tr>
<tr><td><b>Tag name</b></td><td><b>Description</b></td></tr>
<tr><td>hier_instance_name</td><td>Hierarchy instance name of AIR model is being used.</td></tr>
<tr><td>time</td><td>Simulation time</td></tr>
<tr><td>reg_name</td><td>Name of accessed register.</td></tr>
<tr><td>size</td><td>Register size.</td></tr>
<tr><td>address</td><td>Register address.</td></tr>
<tr><td>value</td><td>Register value.</td></tr>
<tr><td>old_value</td><td>Register's value before writing.</td></tr>
<tr><td>new_value</td><td>Register's value after writing.</td></tr>
</table>

6.5.2. DumpStatInfo message style

Table 6-5: Statistical information message description

<table>
<tr><td><b>Condition</b></td><td>This message is dumped out when “DumpStatInfo” is called.</td></tr>
<tr><td><b>Output</b></td><td>This message is printed to standard output (console).</td></tr>
<tr><td colspan="2"><b>Format</b>:
PROFILE(StatInfo): AIR: ###START OF STATISTIC INFO OF AIR MODEL ######
PROFILE(StatInfo): AIR: Index of INTAIRINTREQ/INTAIRDMAREQ   Numbers of INTAIRINTREQ   Numbers of INTAIRDMAREQ
PROFILE(StatInfo): AIR: <index>   <number of INTERRUPT request>   <number of DMA request>
……
PROFILE(StatInfo): AIR: ### END OF STATISTIC INFO OF AIR MODEL ######
<b>Example</b>:
PROFILE(StatInfo): AIR: ###START OF STATISTIC INFO OF AIR MODEL ######
PROFILE(StatInfo): AIR: Index of INTAIRINTREQ/INTAIRDMAREQ   Numbers of INTAIRINTREQ   Numbers of INTAIRDMAREQ
PROFILE(StatInfo): AIR: 0   0   0
……
PROFILE(StatInfo): AIR: 63   1   1
PROFILE(StatInfo): AIR: ### END OF STATISTIC INFO OF AIR MODEL ######</td></tr>
<tr><td><b>Tag name</b></td><td><b>Description</b></td></tr>
<tr><td>index</td><td>The index of INTAIRINTREQ/INTAIRDMAREQ port</td></tr>
<tr><td>number of INTERRUPT request</td><td>The number of interrupt request is output from corresponding port</td></tr>
<tr><td>number of DMA request</td><td>The number of DMA request is output from corresponding port</td></tr>
</table>

6.5.3. Help messages

6.5.3.1. Parameter help messages

Table 6-6: Dumping parameter help message description

<table>
<tr><td><b>Condition</b></td><td>This message is dumped out when command AIR_Help is called with “parameters” argument.</td></tr>
<tr><td><b>Output</b></td><td>This message's kind is printed to standard output (console).
The help message is used for python interface.</td></tr>
<tr><td colspan="2">--- parameters ---
MessageLevel   <fatal|error|warning|info> Set debug message level (default: fatal|error).
DumpRegisterRW <true/false> Enable/disable dumping access register (default: false).</td></tr>
</table>

6.5.3.2. *Command help messages*

Table 6-7: Dumping command help message description

Condition	This message is dumped out when command AIR_Help is called with “commands” argument.
Output	This message's kind is printed to standard output (console). The help message is used for python interface.
--- commands --- AssertReset <start_time> <period> Assert and de-assert reset operation of presetn port. SetCLKfreq <clk_name> <clk_freq> Set clock frequency to CLK_LSB clocks. DumpStatInfo Dump statistic information: number of INTERRUPT/DMA request of each output ports	

6.5.4. Error and debugging messages

6.5.4.1. *Error and debugging messages style*

Table 6-8: 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 MessageLevel. Detailed conditions are described in the “Description” column of Table 6-9
Output	This message's kind is printed to standard output (console).
Format: <severity>: <hier_instance_name>: [<time><unit>] <Message content> **Example**: Error: AIR: [0 ps] CLK_LSB frequency is not set.	
Tag name	**Description**
severity	Kind of message's severity.
hier_instance_name	Hierarchy instance name of AIR model is being used.
time	Simulation time.
unit	Simulation time's unit.
Message content	Message content (message list is described in Table 6-9)

6.5.4.2. *List of error and debugging messages*

Table 6-9: Error and debug messages of AIR model

No.	Type	Severity	Message	Description
1	error	user	Invalid access address <address>	Users access registers of model with invalid address.
2	error	user	Invalid access address <address> with access size <size> bytes	Users access registers of model with wrong size at an address
3	error	user	Invalid access size 0 byte	Users access registers of model with size is 0
4	error	user	Writing access size to <register_name> at address <address> is wrong: <size> byte(s).	Users write the value to registers with invalid size.
5	error	user	Reading access size to <register_name> at address	Users read the value from registers with invalid size.

			<address> is wrong: <size> byte(s).	
6	error	user	Invalid access address <address> with access size <size> bytes	Users access registers of model with wrong size
7	warning	user	Cannot write during reset period	Dump this message when a register is written during reset period.
8	warning	user	Cannot launch call-back function during reset period	Dump this message when register which supported call back function is accessed during reset period.
9	warning	user	Should read all bit in a register	Dump this message when only some bits of register are read.
10	warning	user	<register name> is blocked writing from Bus I/F.	Dump this message when a register which was forced before is written.
11	warning	user	Clock name is invalid	Dump this message when the clock name is not correct.
12	warning	user	The <clock name> period is less than 1 unit time of system	Dump this message when period of clock is not correct compared with unit time of system.
13	warning	user	Reset is in progress.	Dump this message when reset is being active.
14	info	user	Initialize <register name> (<initialized value of register>)	Dump the initialize of each register.
15	info	user	The <clock name> is set with a frequency as %f.	Dump this message when the clock is updated successfully.
16	info	user	Reset signal is asserted.	Dump this message when presetn is low or reset of AssertReset command is called.
17	info	user	Reset signal is negated.	Dump this message when presetn is high or AssertReset command is ended.
18	info	user	AIR will reset for <reset time> ns after <time> ns.	Dump this message when the AssertReset command is called.
19	info	user	Input port <input port name> is <ACTIVE/DEACTIVE>	Dump this message when an input port is change to ACTIVE or DEACTIVE
20	info	user	INTAIRDMAREQ[<index of DMA] is <ACTIVE/DEACTIVE>	Dump this message when output port INTAIRDMAREQ is change to ACTIVE or DEACTIVE
21	info	user	INTAIRINTREQ[<index of interrupt] is <ACTIVE/DEACTIVE>	Dump this message when output port INTAIRINTREQ is change to ACTIVE or DEACTIVE

Table 6-10: Error and debugging message of handleCommand

No.	Type	Severity	Message	Description
1	Users	Error	wrong number of arguments (<command>) : Type reslx.AIR help	Dump this message when a command is called with wrong number of arguments.
2	Users	Error	wrong argument: <argument> (<command>) : Type reslx.AIR help	Dump this message when a command is called with illegal argument.

No.	Type	Severity	Message	Description
3	Internal	Error	command name "<command name>" is invalid.	This message is returned when an invalid command is called. (Command handler model dumps "Unknown command name" instead if this message is returned)
4	Users	Error	<command name> has too much arguments.	Dump this message when a command of Register IF is called with wrong number of arguments.
5	Users	Error	<command name> command needs an argument [<valid value>]	Dump this message when a command of Register IF is called with illegal argument.
6	Users	Error	[<register name>] Invalid force value	Dump this message when a register is forced an invalid value.
7	Users	Error	Wrong command : <reg ...>	Dump this message when an invalid command of Register IF is called.
8	Users	Error	[<register name>] Invalid write value	Dump this message when a register is written an invalid value through Register IF.
9	Internal	Error	Register name is invalid	This message is returned when an unknown register is access through Register IF command. (Command handler model dumps "Unknown command name" instead if this message is returned)

6.6. Defined macro and template

- There is no template in model.
- There are three macros REGIF_SC_REPORT, and REGIF_NOT_USE_SYSTEMC in the model
  - If users define the macro REGIF_SC_REPORT, the SC_REPORT function is used. Otherwise, the "printf" function is used. This macro should be not defined if users defined REGIF_NOT_USE_SYSTEMC.
  - Users can define macro REGIF_NOT_USE_SYSTEMC to remove SystemC part.

7. Flow diagram

7.1. Sequence flow

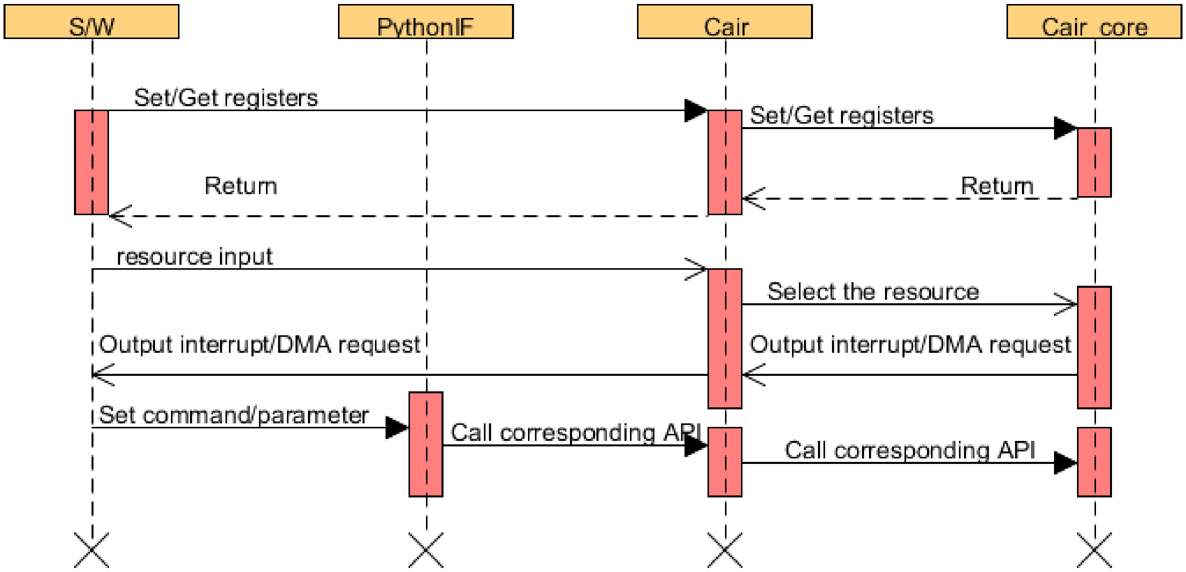


Figure 7-1: Sequence flow

Explanation:

- Users interact with parameters/commands of the AIR model via Python IF and Heap configuration. (The commands and parameters are described in ref[1]/Table 6.2 and Table 6.3).
- Users access the registers of the AIR via the target socket TLM target socket.
- Users set registers, when input signal is active, according to value of registers, the interrupt/DMA request is generated.

7.2. State diagram

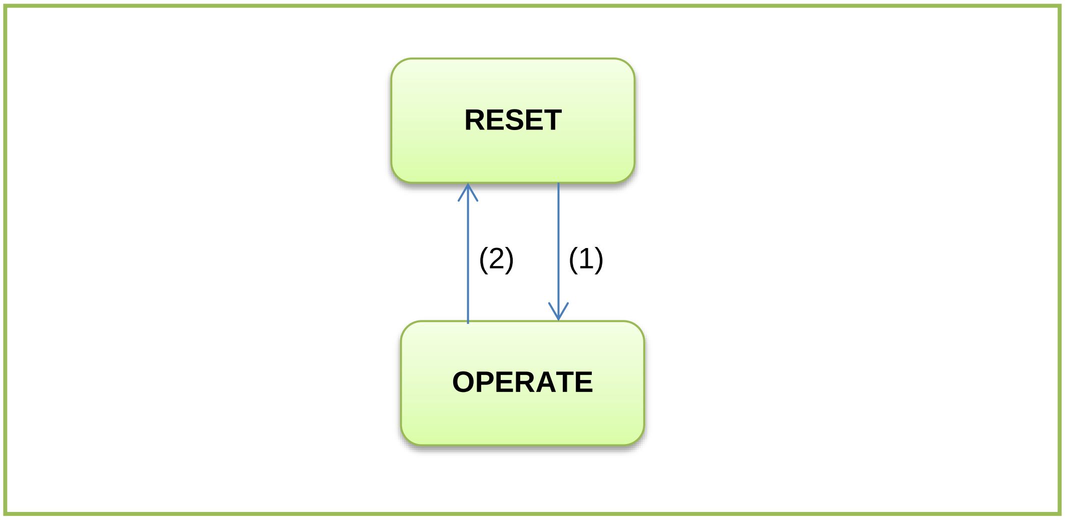
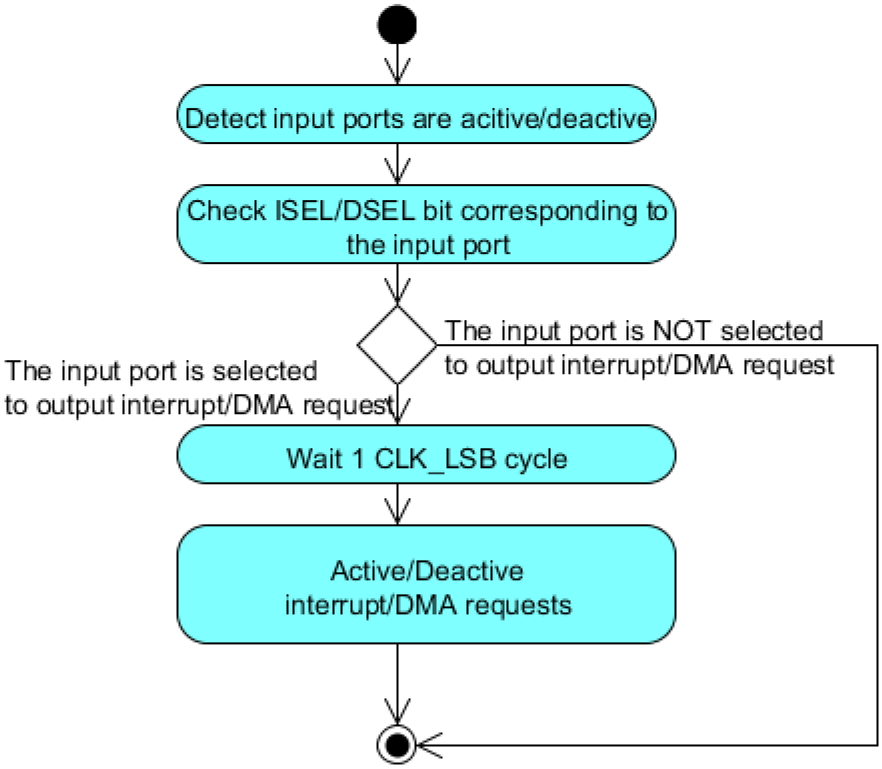


Figure 7-2: State diagram

Explanation:

- OPERATE: Transition of OPERATE state to other states is described as below:
  - OPERATE → RESET: If the reset is asserted (by hardware reset signal or python command), the state of the model is changed from OPERATE state to RESET state.
- RESET: Transition of RESET state to OPERATE state is described as below:
  - RESET → OPERATE: The state of the model is changed from RESET state to OPERTATE state after reset is deassert.

7.3. Operation flow



Detect input ports are acitive/deactive
Check ISEL/DSEL bit corresponding to the input port
The input port is NOT selected to output interrupt/DMA request
The input port is selected to output interrupt/DMA request
Wait 1 CLK_LSB cycle
Active/Deactive interrupt/DMA requests

Figure 7-3: Operation flow

Explanation:

- When external input signal (resource 0/1) is active, if corresponding bit of this resource is selected, corresponding interrupt/DMA request is output after 1 CLK_LSB cycle.

7.4. Python IF and Heap configuration operation flow

Refer to ref[1]/Chapter 7

7.5. Reset flow

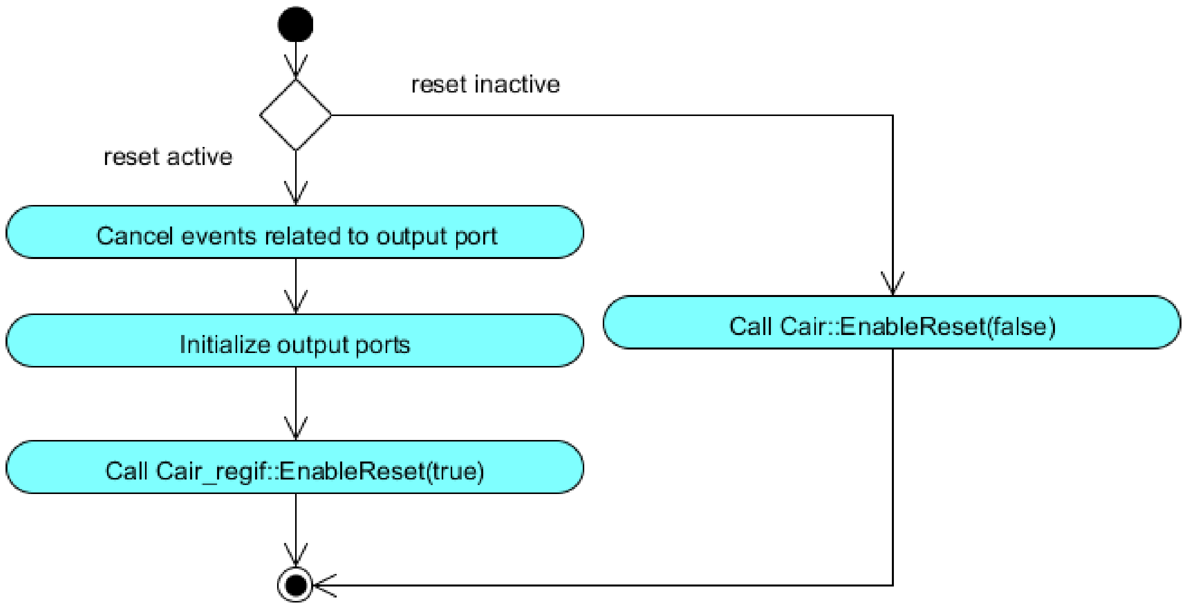

reset inactive
reset active
Cancel events related to output port
Initialize output ports
Call Cair_regif::EnableReset(true)
Call Cair::EnableReset(false)

Figure 7-4: Reset flow

Explanation:

- The AIR has hardware reset (presetn) and corresponding software resets, users can reset AIR by software reset via Python IF. The Figure 7-5 shows the relationship between software reset and hardware reset.
- If the reset is active, the AIR operates as following:
  - Cancel all operation events related to output port
  - Initialize all registers.
  - Initialize all internal variables and output signals.

Note:

- All registers cannot be accessed (read/write) during reset presetn period.

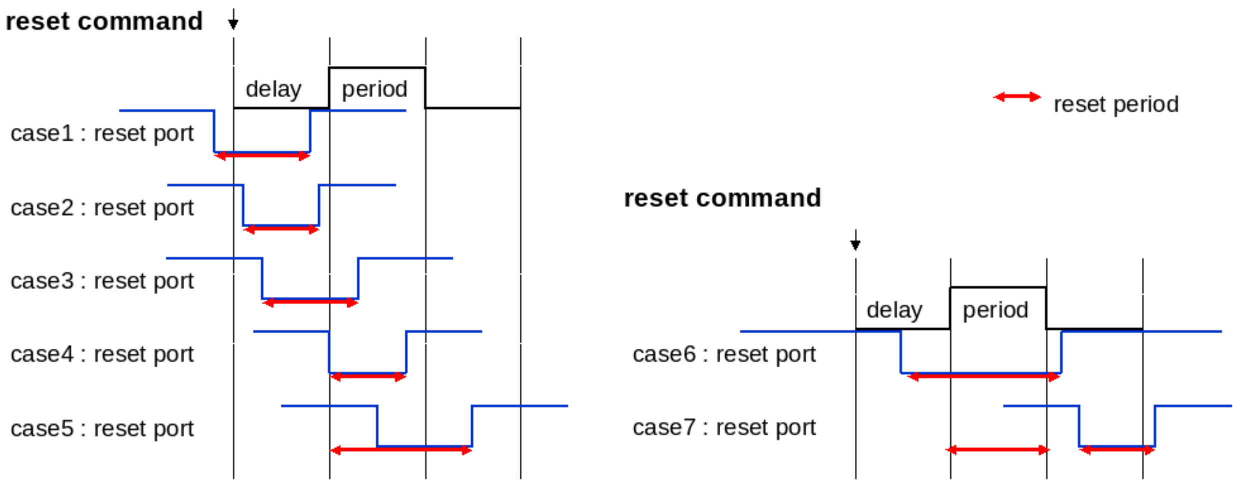


Figure 7-5: Relationship between software reset and hardware reset

8. Functions description

Please refer to INT-SLD-17003_AIR_refman.pdf file (ref[7]) for detail of functions of Cair and Cair_core classes.

9. Limitation

- None

Revision History

Version	Modified points	Agreed by Customer	Approved by RVC	Checked by	Created by
1.0	- Created new	-	Yen Nguyen 08/03/2016	Duc Duong 08/03/2016	Hong Lam 08/02/2016
1.1	- Update chapter 6.5 to correct messages of AIR model - Update chapter 8 to refer INT-SLD-16006_refman.pdf file	-	Yen Nguyen 09/08/2016	Duc Duong 09/08/2016	Hong Lam 08/26/2016
1.2	Chapter 5.1, revise the description of DSADE, DSADI, DSADE, CADE and CADEI.	-	Yen Nguyen 10/18/2016	Duc Duong 10/18/2016	Hong Lam 10/18/2016
1.3	Add Note for Table 4-1	-	Yen Nguyen 11/11/2016	Duc Duong 11/11/2016	Hong Lam 11/11/2016
1.4	- Updated document number “D-SLD-M40-0059-01” -> “D-SLD-M40-0059-02” - Updated to support for E2x_FCC2 product: + Add ref[5] + Update Chapter 1 to add information about supported products “E2x_FCC1 and E2x_FCC2 products” + Update Table 5-1, add note (1*): change DSADC unit number into “m”, change ABFG output number into “w” + Update the Note in Table 5-1 and Step 3 in Chapter 6.3 to add more detail about unused ports in E2x_FCC1 product + Update Table 5-2, Table 5-3: * No.10, 11, 49 -> 54: add resource 1 * No.34 -> 37: add resource 0 + Update Figure 5-1/Explanation and add note (2*) in Table 5-1: add detail about the time output changes “The output timing is depend on the timing input changed (after 1 CLK_LSB clock)”		Yen Nguyen 06/07/2017	Uyen Le 06/07/2017	Ngan Tran 05/29/2017