Exercise 2: HW/SW co-design of Analog-to-Digital Converter (ADC) in SystemC/AMS

In this exercise you should model a ADC peripheral (Only) of a typical ARM Cortex M3 Processor (ATSAM3S4A). In an embedded software development the Programmers View (PV) is described by the Register name and Address map. In virtual prototyping this register and address mapping can be used to create abstract models. Moreover with the SystemC AMS extensions you can model the Analog part using Timed Dataflow model as described in lecture. As you proceed with a **PV**, **you can reuse the same software (SW) for end product, thus motivating a virtual and early software development.** When this model is ready it should connect with the Master module through TLM interface. You are free to choose right TLM methods (either a blocking or non-blocking transport) for your implementation and also a simple master module. A master module should connect to the ADC slave peripheral through TLM initiator -> target techniques. The initiator and target is in master and ADC slave respectively. It should also execute required software functions through the **PV** as below:

ADC_Configuration - configure the ADC module (all registers), Programmable Gain Amplifier (PGA), Sampling Time and Channel using the **PV**

ADC_GetStatus - return current configuration for use of other SW functions

ADC_GetData – returns the sampled data from ADC data register for a channel

For this task configure channels 3, 5, 7 of ADC and connect analog signals Sine, cosine wave and a saw tooth wave respectively. You should use SystemC AMS TDF for these input signals. You could reuse the same examples sent by us for TLM, Sine wave, ADC and PGA.

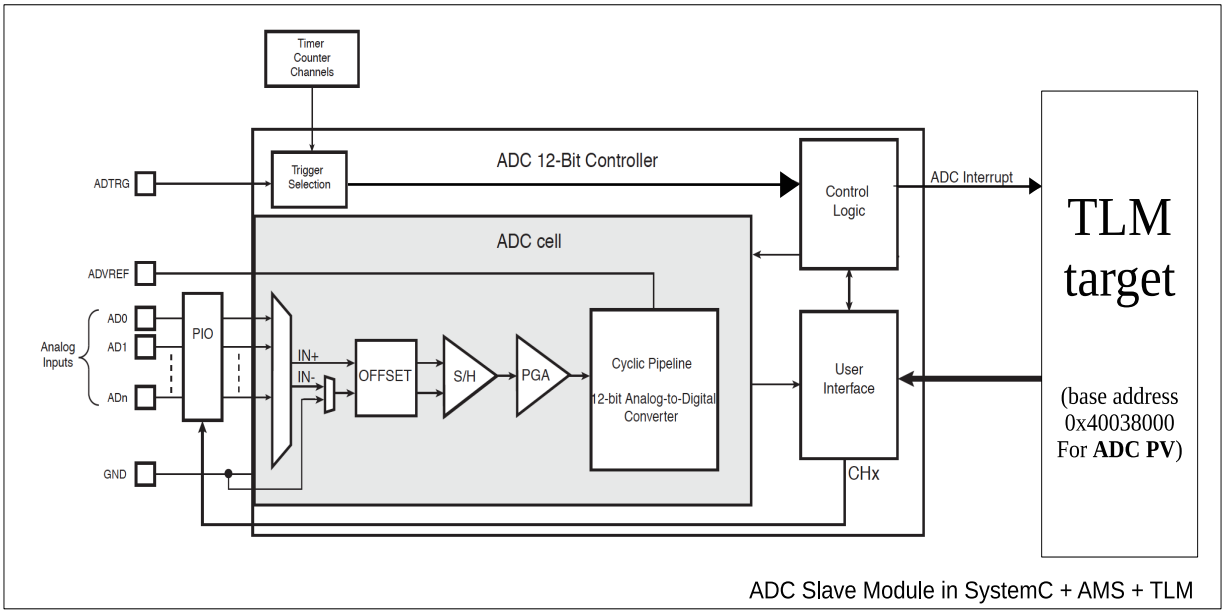
For the ADC slave and Master modules, below block diagram should be followed. The interrupt and timer counter logic is not needed in this exercise. Use a header file with set of registers for ADC starting at base address 0x40038000. This header file is for SW functions to access the ADC. You need to implement the registers following the figure.

When there is execution of these ADC functions in master module, It should trigger transactions to ADC slave based on TLM 2.0 protocol. Trace the analog input signals and its corresponding digital values in VCD file for each channel (here AD3, AD5, AD7). Also print the log where ever possible to a text file about register values. Its needed to handle only the single ended inputs.

The shown ADC is based on a 12-bit ADC configured by an ADC controller. The 16/1 multiplexer enables the analog to digital conversions of 16 analog lines.

Analog Pin Description:

ADVREF – Reference Voltage
The conversion from analog to digital value is performed on a full range between 0 V and the reference voltage ADVREF. The reference voltage is 3.3 V.



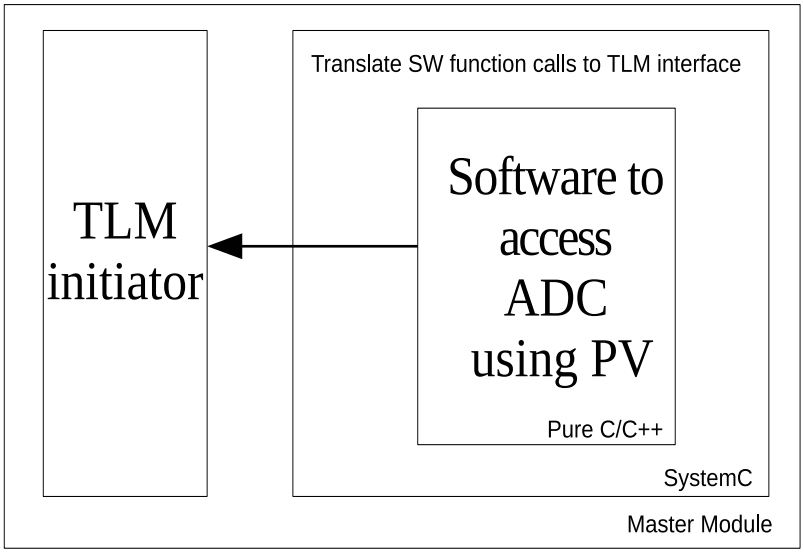


Figure 1. The block diagram for HW/SW co-design of Analog-to-Digital Converter

AD0-AD15 – Analog Input Channels
In this exercise use only three analog channels AD3, AD5 and AD7.Connect the following analog signals to the input channels (refer example about PGA and analog ports) :

1. To **AD3** connect sine wave signal of amplitude 2V
2. To **AD5** connect a cosine wave signal of amplitude 2.5V
3. To **AD7** connect a SAW tooth wave signal of amplitude 2.5V

ADTRG – External trigger
GND - Ground

The user interface is composed of registers, which configure and sample inputs of ADC. The User interface is also called Programmers View (**PV)**.

Table 1. Register mapping of user interface

Memory offset	Register	Name	Access
0x00	Control register	ADC_CR	Write-Only
0x04	Mode Register	ADC_MR	Read-write
0x10	Channel Enable Register	ADC_CHER	Write-only
0x14	Channel Disable Register	ADC_CHDR	Write-only
0x18	Channel Status Register	ADC_CHSR	Read-only
0x48	Channel Gain Register	ADC_CGR	Read-write
0x4C	Channel Offset Register	ADC_COR	Read-write
0x50	Channel Data Register 0	ADC_CDR0	Read-write
0x54	Channel Data Register 1	ADC_CDR1	Read-write
0x58	Channel Data Register 2	ADC_CDR2	Read-write
0x5C	Channel Data Register 3	ADC_CDR3	Read-write
.....	.....	.....	Read-write
0x8C	Channel Data Register 15	ADC_CDR15	Read-write
0xE4	Write Protect Mode Register	ADC_WPMR	Read-write

The exact address of a register is base address + offset from the table 1. The list of registers expected to model in this task is given below.

1.ADC Control Register: ADC_CR – 32 bit Write Only Register

7	6	5	4	3	2	1	0
–	–	–	–	–	–	START	SWRST

- **SWRST: Software Reset**

0 = No effect.

1 = Resets the ADC simulating a hardware reset.

- **START: Start Conversion**

0 = No effect.

1 = Begins analog-to-digital conversion.

The bits from 8-31 are not used.

2.ADC Mode Register: ADC_MR – 32 bit Read-Write

31	30	29	28	27	26	25	24
USEQ	–	TRANSFER		TRACKTIM			
23	22	21	20	19	18	17	16
ANACH	–	SETTLING		STARTUP			
15	14	13	12	11	10	9	8
PRESCAL							
7	6	5	4	3	2	1	0
FREERUN	FWUP	SLEEP	LOWRES	TRGSEL			TRGEN

Note: In this task we will not use the bits SLEEP, FWUP and FREERUN and hence their values should be set to 0.

USEQ bit is used to enable User Sequence mode. In this mode ADC respects the content of registers ADC_SEQR1 and ADC_SEQR2 during conversion from analog to digital value. In this exercise this mode should be disabled and conversion in a simple numeric order should be used. Hence, set USEQ to 0.

Note: TRGEN should be set to 0. Hence, hardware triggers are always disabled.

- **TRGEN: Trigger Enable**

Value	Name	Description
0	DIS	Hardware triggers are disabled. Starting a conversion is only possible by software.
1	EN	Hardware trigger selected by TRGSEL field is enabled.

- **TRGSEL: Trigger Selection**

Value	Name	Description
0	ADC_TRIG0	External trigger
1	ADC_TRIG1	TIO Output of the Timer Counter Channel 0
2	ADC_TRIG2	TIO Output of the Timer Counter Channel 1
3	ADC_TRIG3	TIO Output of the Timer Counter Channel 2
4	ADC_TRIG4	PWM Event Line 0
5	ADC_TRIG5	PWM Event Line 1
6	ADC_TRIG6	Reserved
7	–	Reserved

- **LOWRES: Resolution**

Value	Name	Description
0	BITS_12	12-bit resolution
1	BITS_10	10-bit resolution

- **PRESCAL: Prescaler Rate Selection**
  ADCClock = MCK / ((PRESCAL+1) * 2)

In this exercise use the value of 1 MHz for ADCClock.

- **STARTUP: Start Up Time**

Value	Name	Description
0	SUT0	0 periods of ADCClock
1	SUT8	8 periods of ADCClock
2	SUT16	16 periods of ADCClock
3	SUT24	24 periods of ADCClock
4	SUT64	64 periods of ADCClock
5	SUT80	80 periods of ADCClock
6	SUT96	96 periods of ADCClock
7	SUT112	112 periods of ADCClock
8	SUT512	512 periods of ADCClock
9	SUT576	576 periods of ADCClock
10	SUT640	640 periods of ADCClock
11	SUT704	704 periods of ADCClock
12	SUT768	768 periods of ADCClock
13	SUT832	832 periods of ADCClock
14	SUT896	896 periods of ADCClock
15	SUT960	960 periods of ADCClock

- **SETTLING: Analog Settling Time**

Value	Name	Description
0	AST3	3 periods of ADCClock
1	AST5	5 periods of ADCClock
2	AST9	9 periods of ADCClock
3	AST17	17 periods of ADCClock

- **ANACH: Analog Change**

Value	Name	Description
0	NONE	No analog change on channel switching: DIFF0, GAIN0 and OFF0 are used for all channels
1	ALLOWED	Allows different analog settings for each channel. See ADC_CGR and ADC_COR Registers

- **TRACKTIM: Tracking Time**
  Tracking Time = (TRACKTIM + 1) * ADCClock periods.

- **TRANSFER: Transfer Period**
  Transfer Period = (TRANSFER * 2 + 3) ADCClock periods.

- **USEQ: Use Sequence Enable**

Value	Name	Description
0	NUM_ORDER	Normal Mode: The controller converts channels in a simple numeric order.
1	REG_ORDER	User Sequence Mode: The sequence respects what is defined in ADC_SEQR1 and ADC_SEQR2 registers.

Settling time is a very important parameter, which must be considered when multiple channels are used. Switching between different channels requires some time and hence the ADC conversion should not start before this time. **Tracking time** is the time the ADC must see the input voltage in order to provide the specified accuracy. This time is defined as a hold time of Sample and Hold (S/H) (do possible abstraction here) circuit. The **transfer period** is the time necessary to transfer the sampled input analog signal to the input of Cyclic Pipelined ADC shown in Figure 1. It is the combination of S/H sampling and hold time.

3. ADC Channel Enable Register: ADC_CHER 32-bit Write-Only Register

The bits from 16 to 31 are not used.

15	14	13	12	11	10	9	8
CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8
7	6	5	4	3	2	1	0
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

Note: In this task use only the three channels CH3, CH5 and CH7.

- **CHx: Channel x Enable**

0 = No effect.

1 = Enables the corresponding channel.

4.ADC Channel Disable Register : ADC_CHDR 32 bit Write-Only Register

The bits from 16 to 31 are not used.

15	14	13	12	11	10	9	8
CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8
7	6	5	4	3	2	1	0
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

- **CHx: Channel x Disable**

0 = No effect.

1 = Disables the corresponding channel.

5.ADC Channel Status Register: ADC_CHSR – 32-bit Read-Only Register

15	14	13	12	11	10	9	8
CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8
7	6	5	4	3	2	1	0
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

- **CHx: Channel x Status**

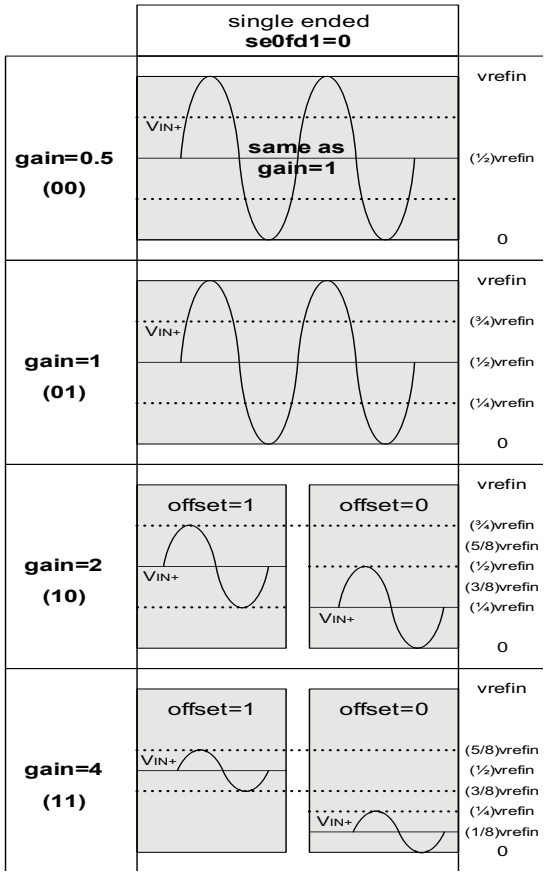
0 = Corresponding channel is disabled.

1 = Corresponding channel is enabled.

The bits from 16 to 31 are not used.

6.ADC Channel Gain Register: 32-bit Read-Write Register

31	30	29	28	27	26	25	24
GAIN15		GAIN14		GAIN13		GAIN12	
23	22	21	20	19	18	17	16
GAIN11		GAIN10		GAIN9		GAIN8	
15	14	13	12	11	10	9	8
GAIN7		GAIN6		GAIN5		GAIN4	
7	6	5	4	3	2	1	0
GAIN3		GAIN2		GAIN1		GAIN0	



Gains should be applied on each channel:

GAINx		Gain applied when DIFFx = 0	Gain applied when DIFFx = 1
0	0	1	0.5
0	1	1	1
1	0	2	2
1	1	4	2

The meaning of the parameter DIFFx mentioned in the table is given in the following register.

OFFSET Bit	OFFSET (DIFF = 0)	OFFSET (DIFF = 1)
0	0	0
1	(G-1)Vrefin/2	

7. ADC Channel Offset Register: ADC_COR – 32 bit Read-Write register

31	30	29	28	27	26	25	24
DIFF15	DIFF14	DIFF13	DIFF12	DIFF11	DIFF10	DIFF9	DIFF8
23	22	21	20	19	18	17	16
DIFF7	DIFF6	DIFF5	DIFF4	DIFF3	DIFF2	DIFF1	DIFF0
15	14	13	12	11	10	9	8
OFF15	OFF14	OFF13	OFF12	OFF11	OFF10	OFF9	OFF8
7	6	5	4	3	2	1	0
OFF7	OFF6	OFF5	OFF4	OFF3	OFF2	OFF1	OFF0

- **DIFFx: Differential inputs for channel x**

0 = Single Ended Mode.

1 = Fully Differential Mode.

Note: In this exercise only single ended mode should be used. Hence, all DIFFx should always be set to 0.

8. ADC Channel Data Register ADC_CDRx for each channel x [x=0..14] – 32-bit RW register

Converted data is placed in the first 10 (12) bits depending on the resolution, which defined, by LOWRES in ADC_MR.

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	DATA			
7	6	5	4	3	2	1	0
DATA							

After the successful completion of modeling part, run appropriate test SW and SystemC test cases to prove that the ADC virtual prototype behaves as expected through Programmers View (Register Map). Justify your results with VCD and log files of simulation !!!

Reference:

http://www.atmel.com/Images/Atmel_6500_32-bit-Cortex-M3-Microcontroller_SAM3S_Datasheet.pdf