

TCE User Group Meeting Working With FLUSH

Presented by: Stephen Fu

31-Mar-2014

Overview

- What's FLUSH
- FLUSH Syntax
- Experiment on FLUSH API
 - Changes to Setup Data Without Flush
 - Changes to Hardware Without Flush
 - Changes to Setup Data With Flush
 - Changes to Hardware With Flush
 - Changes to Hardware With Flush APRM
 - Changes to Hardware With Functional Test
- Summary of FLUSH Effect
- APIs contained operation of FLUSH
- Things to Remember

What's FLUSH

FLUSH is a semi-automatic mean to minimize HW access and to optimize HW setups to achieve better test time.

- This API reflects the changes specified by the following APIs to the setup data or hardware:
 - ANALOG
 - ANALOG_STATUS
 - ROUTING_STATUS
- And also, this reflects the changes specified by the following APIs to the MCD (firmware level):
 - PRIMARY_STATUS
 - TIMING_SPEC / LEVEL_SPEC / SPECIFICATION

FLUSH Syntax

- Syntax

void FLUSH();

void FLUSH(const TM::FLUSH_POST_ACTION action);

- Parameters

action	<p>This parameter controls how the setup data will be downloaded to the hardware.</p> <p>You can leave it empty or use TM::APRM.</p> <p>When using the TM:APRM option, the APRM CNL firmware command will be issued following the normal FLUSH() actions.</p>
--------	---

APRM, APRM? - (Activate PRiMary sets)

The APRM command activates primary sets in the hardware

Experiment on FLUSH API

```

48 virtual void run()
49 {
50     //Add your test code here.
51
52     CONNECT();
53
54     //Primary_status ,timing_spec,level_spec,specification
55     Primary.getLevelSpec().change("vdd1",1);
56     Primary.getLevelSpec().change("vdd2",2);
57     Primary.getLevelSpec().change("vdd3",3);
58
59     Primary.getLevelSpec().change("vil_lcl",0.5);
60     Primary.getLevelSpec().change("vih_lcl",3.3);
61     Primary.getLevelSpec().change("voh_lcl",3);
62     Primary.getLevelSpec().change("vol_lcl",2);
63
64     Primary.getTimingSpec().change("t1",10);
65     Primary.getTimingSpec().change("t2",30);
66
67     //Routing_status
68     Routing.util("utl01").on();
69     Routing.pin("Ain_p").connect();
70
71     //Analog_status
72     Analog.AWG("Ain_p").enable();
73     ANALOG_SET AnalogSet_01("AnalogSet_01");
74     ANALOG_WAVEFORM Wave_Adc_Sine("ADC_Sine");
75     Wave_Adc_Sine.definition(TM::SINE).periods(7).phase(180).min(-1.25).max(1.25).samples(2048);
76     ANALOG_SEQUENCER Seq_ADC_Sine_seq("ADC_Sine_seq");
77     Seq_ADC_Sine_seq.add(TM::RPT,Wave_Adc_Sine,5);
78     Seq_ADC_Sine_seq.add(TM::RPT,Wave_Adc_Sine,5);
79     AnalogSet_01.CLOCK_DOMAIN(2,TM::ANALOG).clock(TM::MCLK_AUTO);
80     AnalogSet_01.AWG("Ain_p").clockDomain(2).rule(TM::FIXFS).frequency(100E6);
81     AnalogSet_01.AWG("Ain_p").coreFunction(TM::HF).attn(10).vOffset(0.1 V).filter("15M").sequencerProgram(Seq_ADC_Sine_seq);
82

```

In the testmethod, Change the SpecVariable values by timing_spec and level_spec API, specify the routing setting data by Routing API, Create Analog set using analog API.

Check the different effect to setup data and hardware with or without FLUSH and FLUSH APRM, Functional Test API.

Changes to Setup Data Without Flush

The image displays three screenshots of Advantest software windows, illustrating changes to setup data without a flush operation. A central callout bubble states: "Nothing changed to the DPS and digital pin level setup, timing setup".

Level Setup Window (Top): Shows the configuration for Level Setup. The "Eqn#" is 1, "Sps#" is 1, "Specification" is lv_sps1, and "Equation" is lv_eq1. The table below shows the setup data for various pins:

dps pin name	level [V]	current source [A]	current sink [A]	off cur. [A]	V Range [V]	I Range [A]	V Bump state	level [V]
Vcc	1.8000	0.1500000000	0.1500000000	min	7.0000			
Vdd	2.5000	0.1500000000	0.1500000000	min				
Vee	3.0000	0.1500000000	0.1500000000	min				

Level Setup Window (Middle): Shows the configuration for Level Setup. The "Eqn#" is 1, "Sps#" is 1, "Lset#" is 1 of 2, "Level" is lv_ls1, "Specification" is lv_sps1, and "Equation" is lv_eq1. The table below shows the setup data for various pins:

pin/group name	pin type	level [V] low	level [V] high	mode	lev/v3h	tol
input9	i	0.300	3.000			
input9	o	1.500	2.500	off		
input8	i	0.300	3.000			
input8	o	1.500	2.500	off		
input7	i	0.300	3.000			
input7	o	1.500	2.500	off		
input6	i	0.300	3.000			
input6	o	1.500	2.500	off		
input5	i	0.300	3.000			
input5	o	1.500	2.500	off		

Timing Setup Window (Bottom): Shows the configuration for Timing Setup. The "Eqn#" is 1, "Sps#" is 1, "Test#" is 1 of 2, "For" is 1, "Test" is 100.000000000ns (10.000000000MHz), and "Cyc" is 2. The table below shows the setup data for various pins:

pin/group	DevCyc	i/o	cycle
input0	BREAK	i	0
input0	0	o	1
input0	1	i	2
input0	1	o	1
input0	1	i	2
input0	1	o	1

Changes to Hardware Without Flush

The screenshot displays the Advantest software interface, which is divided into several panels. On the left, there is a 'Pin Browser' panel showing a list of pins (output1 to output9) and their configurations. Below this, there are sections for 'Analog' settings (Routing, Keep Alive) and 'Utility' settings (Chain, S). The main central panel shows a 'Hardware' configuration window with a schematic diagram of the hardware setup. This diagram includes components like 'Driver', 'Pin PMU', 'BADC', and 'High Prec PMU'. A red circle highlights a specific part of the schematic. To the right of the hardware panel, there is a 'Timing Diagram' panel. This panel shows a table of timing parameters for various inputs (input9 to input0) and cycles. A red circle highlights the 'Vector' and 'Time Delta [ns]' columns. An arrow points from the 'Vector' column to the 'Timing Diagram' panel.

Input	Port: Tim/Pat	Vector	Time Delta [ns]
input9	i		
input8	i		
input7	i		
input6	i		
input5	i		
input4	i		
input3	i		
input2	i		
input1	i		
input0	i		
Cycles			

The changes specified by the TIMING_SPEC, LEVEL_SPEC, and Routing, Analog APIs are not reflected to the hardware.

Changes to Setup Data With Flush

The screenshot displays two windows from the Advantest software: 'Level Setup' and 'Timing Setup'.

Level Setup Window:

- Eqn#: 1, Sps#: 1, RUN TIME
- Specification: lv_sps1, Equation: lv_eq1
- Table:

dps pin name	level [V]	current source [A]	current sink [A]	off cur.	V Range [V]	I Range [A]	V Bump state	level [V]
Vcc	1.0000	0.1500000000	0.1500000000	min				
Vdd	2.0000	0.1500000000	0.1500000000					
Vee	3.0000	0.1500000000	0.1500000000					

Timing Setup Window:

- Eqn#: 1, Sps#: 1, Lset#: 1 of 2, Level: lv_ls1
- Specification: lv_sps1, Equation: lv_eq1
- Table:

pin/group name	pin type	level [V] low	level [V] high	mode	lev/v3h	lamp [V]	high [V]
input9	i	0.500	3.300				
input9	o	2.000	3.000	off		0.000	7.000
input8	i	0.500	3.300			0.000	7.000
input8	o	2.000	3.000	off		0.000	7.000
input7	i	0.500	3.300			2.000	7.000
input7	o	2.000	3.000	off		-2.000	7.000
input6	i	0.500	3.300				
input6	o	2.000	3.000	off			
input5	i	0.500	3.300				

Timing Setup Window (continued):

- Eqn#: 1, Sps#: 1, Tset#: 1 of 2, RUN TIME
- Table:

d1	d2	d3	d4	d5	d6	d7	d8
10.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Timing Setup Window (continued):

- Table:

pin/group	DevCyc	i/o	cycle>	0	1
input0	BREAK	i	2		
input0	0	o	1		
input0	1	i	2		
input0		o	1		
input0		i	2		
input0		o	1		

A callout box states: "The changes specified by the TIMING_SPEC, LEVEL_SPEC, are reflected to the setup data".

Changes to Hardware With Flush

Changes not reflected to DPS

The screenshot displays the Advantest software interface, which is divided into several panels. On the left, there is a 'Pin Browser' panel showing digital and analog pins. Below it, there are three configuration panels: 'Analog', 'Routing', and 'DPS'. The 'DPS' panel is highlighted with a blue border and contains the following settings:

DPS	Voltage
Vcc	1.8 V
Vdd	2.5 V
Vee	3 V

Below the 'DPS' panel is the 'Utility' panel, which shows the 'Chan S' setting. The main part of the screenshot is a large schematic diagram of the hardware, showing various components like DACs, amplifiers, and PMUs. A red circle highlights a specific part of the schematic. On the right side, there is a 'Timing Diagram' panel showing a list of inputs and outputs, with a cursor marker indicating a specific point in time.

The changes specified by the TIMING_SPEC, LEVEL_SPEC, and Routing, Analog APIs are reflected to the hardware except for the changes to DPS.

Changes reflected to DPS

The changes specified by the TIMING_SPEC, LEVEL_SPEC, and Routing, Analog APIs are reflected to the setup data and hardware including the changes to DPS.



Summary of FLUSH Effect

	Execute without Flush		Execute with FLUSH()		Execute with FLUSH(TM::APRM)		Execute with FUNCTIONAL_TEST()	
	Changes reflect to Setup Data	Changes reflect to Hardware	Changes reflect to Setup Data	Changes reflect to Hardware	Changes reflect to Setup Data	Changes reflect to Hardware	Changes reflect to Setup Data	Changes reflect to Hardware
Change Spec with APIs								
Primary.getLevelSpec (for DPS)	No	No	Yes	No	Yes	Yes	Yes	Yes
Primary.getLevelSpec (for digital channel)	No	No	Yes	Yes	Yes	Yes	Yes	Yes
Primary.getTimingSpec	No	No	Yes	Yes	Yes	Yes	Yes	Yes
Routing.util	No	No	Yes	Yes	Yes	Yes	Yes	Yes
Analog set	No	No	Yes	Yes	Yes	Yes	Yes	Yes

APIs contained operation of FLUSH

The changes specified by the PRIMARY_STATUS, TIMING_SPEC, LEVEL_SPEC, and SPECIFICATION are actually reflected to the hardware when any of the following APIs is executed.

- EXECUTE_TEST
- START_TEST
- DIGITAL_CAPTURE_TEST
- NB_DIGITAL_CAPTURE_START
- FUNCTIONAL_TEST
- FLUSH then SPEC_SEARCH (execute() member function)
- FLUSH then SEQUENCER_STATUS (run() member function)
- FLUSH then PMU_IFVM (execute() member function)
- FLUSH then PMU_VFIM (execute() member function)
- FLUSH then TASK_LIST (execute() member function)
- FLUSH then DPS_VFIM (execute() member function)

The FLUSH also reads the buffered setting data in the current analog set and routing setting data in the primary routing set and defined by the [ACMD "CNCT"](#) command from the MCD, and downloads the contents to the hardware.

Things to Remember...

When using the FLUSH API, the following points should be considered:

- Each HW access takes time. Analog setups may get cached and activated (flushed) in a single shot when performing a FLUSH(). In addition the internal wait time can be optimized, which would be otherwise required for each single HW access.
- HW setup optimization means that FLUSH may recognize situations where it can avoid redundant setups (e.g. a disconnect, if the next test suite is performing a connection again).
- FLUSH() execution time is in a range of 1 to 10 and more ms dependant on the complexity of the HW changes.
- It scales with number of cores, actions, sequencer program length and tester period. MSE is in the range of 60%.
- Although FLUSH() tries to optimize execution time, unnecessary calls of FLUSH() will add test time.
- Debugging may require to put additional FLUSH()'s in the code to force immediate activation to allow visibility in debug tools, which needs to be removed later for production.

Example

Test program with 500 testsuites with one unnecessary FLUSH()
each may end up with an overhead of $500 \times 3\text{ms} = 1.5\text{s}$

Q/A