VRTX API Support and Deviations

VRTX APIs	Support	API Deviations
sc_accept	Yes	
sc_acceptc	No	
sc_adelay	No	
sc_call	No	
sc_delay	Yes	
sc_elock	No	
sc_elock_init	No	
sc_fclear	Yes	
sc_fcreate	Yes	- dispatch of flag to pended task will be based on FIFO and not task priority order
sc_fdelete	Yes	
sc_finquiry	Yes	
sc_fpend	Yes	
sc_fpost	Yes	
sc_gblock	Yes	
sc_gclock	No	Use OS Abstractor API equivalents
sc_getc	No	
sc_gtime	Yes	
sc_gversion	Yes	
sc_halloc	Yes	
sc_hcreate	Yes	
sc_hdelete	Yes	
sc_hfree	Yes	
sc_hinquiry	Yes	
sc_lock	Yes	
sc_maccept	Yes	
sc_mcreate	Yes	
sc_mdelete	Yes	
sc_minquiry	Yes	
sc_mpend	Yes	
sc_mpost	Yes	
sc_pcreate	Yes	
sc_pdelete	Yes	
sc_pend	Yes	
sc_pextend	No	
sc_pinquiry	Yes	
sc_post	Yes	
sc_putc	Yes	
sc_qaccept	Yes	

VRTX APIs	Support	API Deviations
sc_qbrdcst	No	
sc_qcreate	Yes	- dispatch of queue messages to pended task will be based on FIFO and not task priority order - "qsize" parameter = 0 (as mailbox), not supported
sc_qdelete	Yes	- "opt" parameter = 1, will delete queue, but does not wake up pended tasks
sc_qecreate	Yes	- dispatch of queue messages to pended task will be based on FIFO and not task priority order - "qsize" parameter = 0 (as mailbox), not supported
sc_qinquiry	Yes	
sc_qjam	Yes	- (on Windows only), sc_qjam will behave like sc_qpost
sc_qpend	Yes	
sc_qpost	Yes	
sc_rblock	Yes	
sc_saccept	Yes	
sc_sclock	No	- Use OS Abstractor equivalents
sc_screate	Yes	
sc_sdelete	Yes	
sc_sinquiry	Yes	
sc_spend	Yes	
sc_spost	Yes	
sc_stime	No	
sc_tcreate	Yes	- "tid" parameter = 0 (special task), not supported.
sc_tdelete	Yes	- "code" parameter = 'A' (multiple tasks) not supported.
sc_tecreate	Yes	- "tid" parameter = 0 (special task), not supported. - "mode" parameter, TCMNOFP, TCMDEBUG, TCMINTRDIS, TCMINHDEB are ignored - based on TCMSUPER, if flag set use kernel stack size, otherwise use user stack size as task stack size - TCMSUPER for user/kernel will depend on underlying target OS (refer to OS Abstractor manuals)
sc_texcreate	No	
sc_tinquiry	Yes	
sc_tpriority	Yes	
sc_tresume	Yes	- "code" parameter = 'A' (multiple tasks) not supported.
sc_tslice	No	
sc_tsuspend	Yes	- "code" parameter = 'A' (multiple tasks) not supported.
sc_txcreate	No	
sc_unelock	No	
sc_unlock	Yes	
sc_vqaccept	No	
sc_vqcreate	No	
sc_vqdelete	No	
sc_vqinquiry	No	
sc_vqjam	No	

VRTX APIs	Support	API Deviations
sc_vqpend	No	
sc_vqpost	No	
sc_waitc	Yes	
ui_rxchr	No	
ui_timer	No	
ui_txrdy	No	
vqueue_init	No	
vrtx_go	No	
vrtx_init	Yes	

VRTX OS Changer Configuration

User Configuration Table (refer to vrtx_interface_usr.h)

Configuration	Description	Default Setting
CFUTSKCT	task id range specified by user	OS_TOTAL_SYSTEM_TASKS/2 which is 100/2 = 50 ID = 0, is reserved for internal use . User assigned task ID range can be 1 to 50
CFPARTCOUNT	partition id range specified by user	OS_TOTAL_SYSTEM_PM_POOLS/2 which is 100/2 = 50 User assigned partition ID range can only be 0 to 49
CFQUEUECOUNT	queue id range specified by user	OS_TOTAL_SYSTEM_QUEUES/2 which is 100/2 = 50 User assigned queue ID range can only be 0 to 49
INIT_VRTX_HEAP_TIERS_INFO	Used by sc_hcreate(). This contains the tiered memory pool config info consisting of levels and blocks within each level	{ 100,1024 },{ 50,128 },{ 20,64 },{ 0, 0 } The above means there are three levels (# of blocks, max block size) but can be configured to application needs. Note that it is required that the last level should always be {0,0}.
VRTX_MIN_STACK_SIZE	Used internally by sc_tcreate for the task size.	2*OS_MIN_STACK_SIZE Adjust this as per application needs

Configuration	Description	Default Setting
OS_VRTX_TASK_THREAD_POLICY	Usde within sc_tcreate() and sc_tecreate(). Applies to Linux target only. The Linux native real-time threads created via OS Changer will be one of the following: OS_FIFO_POLICY OS_RR_POLICY	OS_FIFO_POLICY The threads will not give control back to OS unless they make appropriate OS Changer calls. If your application is not very time critical then do not use this policy but use OS_RR_POLICY instead. This way your VRTX application will not use a lot of CPU

The full OS Abstractor configuration of kernel resources (cross_os_usr.h), please refer to OS Abstractor system config manual. Many of the VRTX configuration is derived from the OS Abstractor configuration so it is important that cross_os_usr.h is configured properly for VRTX OS Changer to function correctly.

VRTX OS Changer – Useful Macros

Macro	Description
VRTX2OSA_TASK_ID(x)	Gets you the OS Abstractor task id from VRTX task ID
VRTX2OSA_PARTITION_ID(x)	Gets you the OS Abstractor partition id from VRTX partition ID
VRTX2OSA_QUEUE_ID(x)	Gets you the OS Abstractor queue id from VRTX queue ID

The rest of the VRTX resources like, semaphore, mutex, flags, heap, etc the IDs at VRTX interface level is the same that of the OS Abstractor. Knowing OS Abstractor IDs will allow application developers be able to use the OS Abstractor APIs for advanced features (refer to OS Abstractor reference manual).

There are various other useful macros (data types, ticks_per_second, default_time_slice, etc.) that are available for use. Please refer to OS Abstractor system configuration manual.

VRTX OS Changer – Important Notes

- To see the real-time behavior of VRTX OS Changer, make sure to launch the Visual Studio 2015 and eclipse tools with Admin privilege. If you see any error saying “unable to change priority”, then the application is not running in admin mode. Even if you have a user account with admin privilege, you have to specify in VS properties that you want the IDE to be launched with admin
- Run the application in a single CPU core only. This can be done by passing the CPU mask during OS Changer initialization