ThreadX 5.0 Services

ThreadX Entry

VOID **tx_kernel_enter**(VOID);

Thread Control

UINT **tx_thread_create**(TX_THREAD *thread_ptr, CHAR *name_ptr,
VOID (*entry_function)(ULONG), ULONG entry_input,
VOID *stack_start, ULONG stack_size,
UINT priority, UINT preempt_threshold,
ULONG time_slice, UINT auto_start);
UINT **tx_thread_delete**(TX_THREAD *thread_ptr);
UINT **tx_thread_entry_exit_notify**(TX_THREAD *thread_ptr,
VOID (*thread_entry_exit_notify)(TX_THREAD *, UINT)); TX_THREAD
***tx_thread_identify**(VOID);
UINT **tx_thread_info_get**(TX_THREAD *thread_ptr, CHAR **name, UINT *state,
ULONG *run_count, UINT *priority, UINT *preemption_threshold,
ULONG *time_slice, TX_THREAD **next_thread, TX_THREAD **next_suspended_thread);
UINT **tx_thread_performance_info_get**(TX_THREAD *thread_ptr, ULONG *resumptions, ULONG *suspensions,
ULONG *solicited_preemptions, ULONG *interrupt_preemptions, ULONG *priority_inversions,
ULONG *time_slices, ULONG *relinquishes, ULONG *timeouts,
ULONG *wait_aborts, TX_THREAD **last_preempted_by);
UINT **tx_thread_performance_system_info_get**(ULONG *resumptions, ULONG *suspensions,
ULONG *solicited_preemptions, ULONG *interrupt_preemptions, ULONG *priority_inversions,
ULONG *time_slices, ULONG *relinquishes, ULONG *timeouts, ULONG *wait_aborts,
ULONG *non_idle_returns, ULONG *idle_returns);
UINT **tx_thread_preemption_change**(TX_THREAD *thread_ptr, UINT new_threshold, UINT *old_threshold);
UINT **tx_thread_priority_change**(TX_THREAD *thread_ptr, UINT new_priority, UINT *old_priority);VOID
tx_thread_relinquish(VOID);
UINT **tx_thread_reset**(TX_THREAD *thread_ptr);
UINT **tx_thread_resume**(TX_THREAD *thread_ptr);
UINT **tx_thread_sleep**(ULONG timer_ticks);
UINT **tx_thread_stack_error_notify**(VOID (*stack_error_handler)(TX_THREAD *));
UINT **tx_thread_suspend**(TX_THREAD *thread_ptr);
UINT **tx_thread_terminate**(TX_THREAD *thread_ptr);
UINT **tx_thread_time_slice_change**(TX_THREAD *thread_ptr, ULONG new_time_slice, ULONG *old_time_slice);
UINT **tx_thread_wait_abort**(TX_THREAD *thread_ptr);

Thread Communication

UINT **tx_queue_create**(TX_QUEUE *queue_ptr, CHAR *name_ptr, UINT message_size,
VOID *queue_start, ULONG queue_size);
UINT **tx_queue_delete**(TX_QUEUE *queue_ptr);
UINT **tx_queue_flush**(TX_QUEUE *queue_ptr);
UINT **tx_queue_info_get**(TX_QUEUE *queue_ptr, CHAR **name, ULONG *enqueued, ULONG *available_storage,
TX_THREAD **first_suspended, ULONG *suspended_count, TX_QUEUE **next_queue);
UINT **tx_queue_performance_info_get**(TX_QUEUE *queue_ptr, ULONG *messages_sent, ULONG *messages_received,
ULONG *empty_suspensions, ULONG *full_suspensions, ULONG *full_errors, ULONG *timeouts);
UINT **tx_queue_performance_system_info_get**(ULONG *messages_sent, ULONG *messages_received,
ULONG *empty_suspensions, ULONG *full_suspensions, ULONG *full_errors, ULONG *timeouts);
UINT **tx_queue_receive**(TX_QUEUE *queue_ptr, VOID *destination_ptr, ULONG wait_option);
UINT **tx_queue_send**(TX_QUEUE *queue_ptr, VOID *source_ptr, ULONG wait_option);
UINT **tx_queue_send_notify**(TX_QUEUE *queue_ptr, VOID (*queue_send_notify)(TX_QUEUE *));
UINT **tx_queue_front_send**(TX_QUEUE *queue_ptr, VOID *source_ptr, ULONG wait_option);
UINT **tx_queue_prioritize**(TX_QUEUE *queue_ptr);

Thread Synchronization

UINT **tx_event_flags_create**(TX_EVENT_FLAGS_GROUP *group_ptr, CHAR *name_ptr);
UINT **tx_event_flags_delete**(TX_EVENT_FLAGS_GROUP *group_ptr);
UINT **tx_event_flags_get**(TX_EVENT_FLAGS_GROUP *group_ptr, ULONG requested_flags,
UINT get_option, ULONG *actual_flags_ptr, ULONG wait_option);
UINT **tx_event_flags_info_get**(TX_EVENT_FLAGS_GROUP *group_ptr, CHAR **name, ULONG *current_flags,
TX_THREAD **first_suspended, ULONG *suspended_count,
TX_EVENT_FLAGS_GROUP **next_group);
UINT **tx_event_flags_performance_info_get**(TX_EVENT_FLAGS_GROUP *group_ptr, ULONG *sets, ULONG *gets,
ULONG *suspensions, ULONG *timeouts);
UINT **tx_event_flags_performance_system_info_get**(ULONG *sets, ULONG *gets,
ULONG *suspensions, ULONG *timeouts);
UINT **tx_event_flags_set**(TX_EVENT_FLAGS_GROUP *group_ptr, ULONG flags_to_set, UINT set_option);
UINT **tx_event_flags_set_notify**(TX_EVENT_FLAGS_GROUP *group_ptr,
VOID (*events_set_notify)(TX_EVENT_FLAGS_GROUP *));

UINT **tx_mutex_create**(TX_MUTEX *mutex_ptr, CHAR *name_ptr, UINT inherit);
UINT **tx_mutex_delete**(TX_MUTEX *mutex_ptr);
UINT **tx_mutex_get**(TX_MUTEX *mutex_ptr, ULONG wait_option);
UINT **tx_mutex_info_get**(TX_MUTEX *mutex_ptr, CHAR **name, ULONG *count, TX_THREAD **owner,
TX_THREAD **first_suspended, ULONG *suspended_count, TX_MUTEX **next_mutex);
UINT **tx_mutex_performance_info_get**(TX_MUTEX *mutex_ptr, ULONG *puts, ULONG *gets,
ULONG *suspensions, ULONG *timeouts, ULONG *inversions, ULONG *inheritances);
UINT **tx_mutex_performance_system_info_get**(ULONG *puts, ULONG *gets, ULONG *suspensions, ULONG *timeouts,
ULONG *inversions, ULONG *inheritances);
UINT **tx_mutex_prioritize**(TX_MUTEX *mutex_ptr);
UINT **tx_mutex_put**(TX_MUTEX *mutex_ptr);

UINT **tx_semaphore_ceiling_put**(TX_SEMAPHORE *semaphore_ptr, ULONG ceiling);
UINT **tx_semaphore_create**(TX_SEMAPHORE *semaphore_ptr, CHAR *name_ptr, ULONG initial_count);
UINT **tx_semaphore_delete**(TX_SEMAPHORE *semaphore_ptr);
UINT **tx_semaphore_get**(TX_SEMAPHORE *semaphore_ptr, ULONG wait_option);
UINT **tx_semaphore_info_get**(TX_SEMAPHORE *semaphore_ptr, CHAR **name, ULONG *current_value,
TX_THREAD **first_suspended, ULONG *suspended_count, TX_SEMAPHORE **next_semaphore);
UINT **tx_semaphore_performance_info_get**(TX_SEMAPHORE *semaphore_ptr, ULONG *puts, ULONG *gets,
ULONG *suspensions, ULONG *timeouts);
UINT **tx_semaphore_performance_system_info_get**(ULONG *puts, ULONG *gets,
ULONG *suspensions, ULONG *timeouts);
UINT **tx_semaphore_prioritize**(TX_SEMAPHORE *semaphore_ptr);
UINT **tx_semaphore_put**(TX_SEMAPHORE *semaphore_ptr);
UINT **tx_semaphore_put_notify**(TX_SEMAPHORE *semaphore_ptr, VOID (*semaphore_put_notify)(TX_SEMAPHORE *));

Interrupt Control

UINT **tx_interrupt_control**(UINT new_posture);

Timer Facilities

ULONG **tx_time_get**(VOID);VOID
tx_time_set(ULONG new_time);

UINT **tx_timer_activate**(TX_TIMER *timer_ptr);
UINT **tx_timer_change**(TX_TIMER *timer_ptr, ULONG initial_ticks, ULONG reschedule_ticks);
UINT **tx_timer_create**(TX_TIMER *timer_ptr, CHAR *name_ptr,
VOID (*expiration_function)(ULONG), ULONG expiration_input, ULONG initial_ticks,
ULONG reschedule_ticks, UINT auto_activate);
UINT **tx_timer_deactivate**(TX_TIMER *timer_ptr);
UINT **tx_timer_delete**(TX_TIMER *timer_ptr);
UINT **tx_timer_info_get**(TX_TIMER *timer_ptr, CHAR **name, UINT *active, ULONG *remaining_ticks,
ULONG *reschedule_ticks, TX_TIMER **next_timer);
UINT **tx_timer_performance_info_get**(TX_TIMER *timer_ptr, ULONG *activates, ULONG *reactivates,
ULONG *deactivates, ULONG *expirations, ULONG *expiration_adjusts);
UINT **tx_timer_performance_system_info_get**(ULONG *activates, ULONG *reactivates,
ULONG *deactivates, ULONG *expirations, ULONG *expiration_adjusts);

Memory Management

UINT **tx_block_allocate**(TX_BLOCK_POOL *pool_ptr, VOID **block_ptr, ULONG wait_option);
UINT **tx_block_pool_create**(TX_BLOCK_POOL *pool_ptr, CHAR *name_ptr, ULONG block_size,
VOID *pool_start, ULONG pool_size);
UINT **tx_block_pool_delete**(TX_BLOCK_POOL *pool_ptr);
UINT **tx_block_pool_info_get**(TX_BLOCK_POOL *pool_ptr, CHAR **name, ULONG *available_blocks,
ULONG *total_blocks, TX_THREAD **first_suspended,
ULONG *suspended_count, TX_BLOCK_POOL **next_pool);
UINT **tx_block_pool_performance_info_get**(TX_BLOCK_POOL *pool_ptr, ULONG *allocates, ULONG *releases,
ULONG *suspensions, ULONG *timeouts);
UINT **tx_block_pool_performance_system_info_get**(ULONG *allocates, ULONG *releases,
ULONG *suspensions, ULONG *timeouts);
UINT **tx_block_pool_prioritize**(TX_BLOCK_POOL *pool_ptr);
UINT **tx_block_release**(VOID *block_ptr);

UINT **tx_byte_allocate**(TX_BYTE_POOL *pool_ptr, VOID **memory_ptr, ULONG memory_size, ULONG wait_option);
UINT **tx_byte_pool_create**(TX_BYTE_POOL *pool_ptr, CHAR *name_ptr, VOID *pool_start, ULONG pool_size);
UINT **tx_byte_pool_delete**(TX_BYTE_POOL *pool_ptr);
UINT **tx_byte_pool_info_get**(TX_BYTE_POOL *pool_ptr, CHAR **name, ULONG *available_bytes,
ULONG *fragments, TX_THREAD **first_suspended,
ULONG *suspended_count, TX_BYTE_POOL **next_pool);
UINT **tx_byte_pool_performance_info_get**(TX_BYTE_POOL *pool_ptr, ULONG *allocates, ULONG *releases,
ULONG *fragments_searched, ULONG *merges, ULONG *splits,
ULONG *suspensions, ULONG *timeouts);
UINT **tx_byte_pool_performance_system_info_get**(ULONG *allocates, ULONG *releases,
ULONG *fragments_searched, ULONG *merges, ULONG *splits,
ULONG *suspensions, ULONG *timeouts);
UINT **tx_byte_pool_prioritize**(TX_BYTE_POOL *pool_ptr);
UINT **tx_byte_release**(VOID *memory_ptr);

expresslogic

ThreadX Execution States

TX_READY	0
TX_COMPLETED	1
TX_TERMINATED	2
TX_SUSPENDED	3
TX_SLEEP	4
TX_QUEUE_SUSP	5
TX_SEMAPHORE_SUSP	6
TX_EVENT_FLAG	7
TX_BLOCK_MEMORY	8
TX_BYTE_MEMORY	9
TX_TCP_IP	12
TX_MUTEX_SUSP	13

ThreadX API Return Values

TX_ACTIVATE_ERROR	0x17
TX_CALLER_ERROR	0x13
TX_CEILING_EXCEEDED	0x21
TX_DELETE_ERROR	0x11
TX_DELETED	0x01
TX_FEATURE_NOT_ENABLED	0xFF
TX_GROUP_ERROR	0x06
TX_INHERIT_ERROR	0x1F
TX_INVALID_CEILING	0x22
TX_MUTEX_ERROR	0x1C
TX_NO_EVENTS	0x07
TX_NO_INSTANCE	0x0D
TX_NO_MEMORY	0x10
TX_NOT_AVAILABLE	0x1D
TX_NOT_DONE	0x20
TX_NOT_OWNED	0x1E
TX_OPTION_ERROR	0x08
TX_POOL_ERROR	0x02
TX_PRIORITY_ERROR	0x0F
TX_PTR_ERROR	0x03
TX_QUEUE_EMPTY	0x0A
TX_QUEUE_ERROR	0x09
TX_QUEUE_FULL	0x0B
TX_RESUME_ERROR	0x12
TX_SEMAPHORE_ERROR	0x0C
TX_SIZE_ERROR	0x05
TX_START_ERROR	0x10
TX_SUCCESS	0x00
TX_SUSPEND_ERROR	0x14
TX_SUSPEND_LIFTED	0x19
TX_THREAD_ERROR	0x0E
TX_THRESH_ERROR	0x18
TX_TICK_ERROR	0x16
TX_TIMER_ERROR	0x15
TX_WAIT_ABORT_ERROR	0x1B
TX_WAIT_ABORTED	0x1A
TX_WAIT_ERROR	0x04

ThreadX API Parameters

TX_AND	2
TX_AND_CLEAR	3
TX_AUTO_ACTIVATE	1
TX_AUTO_START	1
TX_DONT_START	0
TX_FALSE	0
TX_INHERIT	1
TX_NO_INHERIT	0
TX_NO_TIME_SLICE	0
TX_NO_WAIT	0
TX_NULL	0
TX_OR	0
TX_OR_CLEAR	1
TX_NO_ACTIVATE	0
TX_STACK_FILL	0xEFEFEFEFUL
TX_THREAD_ENTRY	0
TX_THREAD_EXIT	1
TX_TRUE	1
TX_WAIT_FOREVER	0xFFFFFFFFUL

ThreadX Conditional Compilation Defines

TX_DISABLE_ERROR_CHECKING
TX_DISABLE_NOTIFY_CALLBACKS
TX_DISABLE_PREEMPTION_THRESHOLD
TX_DISABLE_REDUNDANT_CLEARING
TX_DISABLE_STACK_FILLING
TX_ENABLE_STACK_CHECKING
TX_INCLUDE_USER_DEFINE_FILE
TX_MAX_PRIORITIES
TX_MINIMUM_STACK
TX_REACTIVATE_INLINE
TX_TIMER_PROCESS_IN_ISR
TX_TIMER_THREAD_STACK_SIZE
TX_TIMER_THREAD_PRIORITY

TX_BLOCK_POOL_ENABLE_PERFORMANCE_INFO
TX_BYTE_POOL_ENABLE_PERFORMANCE_INFO
TX_EVENT_FLAGS_ENABLE_PERFORMANCE_INFO
TX_MUTEX_ENABLE_PERFORMANCE_INFO
TX_QUEUE_ENABLE_PERFORMANCE_INFO
TX_SEMAPHORE_ENABLE_PERFORMANCE_INFO
TX_THREAD_ENABLE_PERFORMANCE_INFO
TX_TIMER_ENABLE_PERFORMANCE_INFO

ASCII Character Codes in HEX

most significant nibble

least significant nibble	0_	1_	2_	3_	4_	5_	6_	7_
_0	NUL	DLE	SP	0	@	P	‘	p
_1	SOH	DC1	!	1	A	Q	a	q
_2	STX	DC2	“	2	B	R	b	r
_3	ETX	DC3	#	3	C	S	c	s
_4	EOT	DC4	$	4	D	T	d	t
_5	ENQ	NAK	%	5	E	U	e	u
_6	ACK	SYN	&	6	F	V	f	v
_7	BEL	ETB	‘	7	G	W	g	w
_8	BS	CAN	(	8	H	X	h	x
_9	HT	EM	)	9	I	Y	i	y
_A	LF	SUB	*	:	J	Z	j	z
_B	VT	ESC	+	;	K	[	K	}
_C	FF	FS	,	<	L	\	l	l
_D	CR	GS	-	=	M	]	m	}
_E	SO	RS	.	>	N	^	n	~
_F	SI	US	/	?	O	_	o	DEL

Power	Decimal Value	Hexadecimal Value	Power	Decimal Value	Hexadecimal Value
0	1	1	17	131,072	0x20000
1	2	2	18	262,144	0x40000
2	4	4	19	524,288	0x80000
3	8	8	20	1,048,576	0x100000
4	16	0x10	21	2,097,152	0x200000
5	32	0x20	22	4,194,304	0x400000
6	64	0x40	23	8,388,608	0x800000
7	128	0x80	24	16,777,216	0x1000000
8	256	0x100	25	33,554,432	0x2000000
9	512	0x200	26	67,108,864	0x4000000
10	1,024	0x400	27	134,217,728	0x8000000
11	2,048	0x800	28	268,435,456	0x10000000
12	4,096	0x1000	29	536,870,912	0x20000000
13	8,192	0x2000	30	1,073,741,824	0x40000000
14	16,384	0x4000	31	2,147,483,647	0x80000000
15	32,768	0x8000	32	4,294,967,295	0x100000000
16	65,536	0x10000			

ANSI C Format Specifications

%d or %i	decimal integer
%ld or %li	long decimal integer
%hd or %hi	short decimal integer
%o	octal integer
%lo	long octal integer
%ho	short octal intger
%x or %X	hexadecimal integer
%lx or %lX	long hexadecimal integer
%hx or %hX	short hexadecimal integer
%u	unsigned decimal integer
%lu	long unsigned decimal integer
%hu	short unsigned decimal integer
%c	ASCII character
%s	ASCII string, null terminated
%f	floating point
%e or %E	double precision floating point
%g or %G	double precision floating point
%p	void pointer

THREADX

Programmers Guide

expresslogic