

Chapter 1

Introduction to the FRODO Virtual Machine

1.1 Safety and Time-Triggered Architectures

Safety-critical cyber-physical systems (CPS) are an integral component of normal human life. People use, and rely upon, flight control systems and pacemakers every day and demand for them to always work as expected.

The FRODO virtual machine

Chapter 2

FRDO Public Types and Structures

2.1 Defined Data Types

Abcdefg

2.2 Defined Data Structures

Abcdefg

Chapter 3

Core FRODO Events and Errors

Each of the major areas of functionality within the FRODO virtual machine (VM) creates or responds to a set of events and possibly generates a set of errors. This section documents the events and errors related to each of the major subsystems within FRODO. Each subsystem corresponds to a given `EVENTTYPE` enumerated value. For example, the on-line scheduler is related to the `SCHEDULER` enum value. Each subsection title contains the name of the subsystem and the corresponding enum value.

For each event listed below the name of the event is followed by the **enum value** used within the codebase to represent the event. These are followed by the **pertinent variables** that are associated with an occurrence of the event. Finally, you will find a textual description of what can cause the event to be issued.

For each error listed below...

3.1 On-line Scheduler : Scheduler

The FRODO online scheduler fundamentally implements time-triggered architecture (TTA) execution semantics. All safety-critical tasks must utilize a static TTA schedule. In addition to TTA, the FRODO on-line scheduler implements a sporadic task scheduler that can be utilized for non-critical tasks.

3.1.1 Scheduler Events

The FRODO scheduler issues a number of events as it progresses through the normal execution life-cycle. The following are the expected events generated by the scheduler:

Execution Start : **ExecutionStart** – (actual absolute time)

Event is issued when the scheduler begins its cyclic execution. Time before this includes the duration of time from when the high-resolution clock is initialized (via the logger being initialized) to when the cyclic execution begins by calling FS_Execute (SchedExecute). This time includes any setup, such as task creation, channel creation, port creation, etc.

Hyperperiod Start : **HyperperiodStart** – (actual absolute time)

Event is issued at the beginning of every hyperperiod from the start of execution through the end of execution.

Task Start : **TaskStart** – (task ID, actual absolute time, actual interval time, expected interval time)

Event is issued when the scheduler releases a task for execution (via FS_SignalExecution - SchedSignalExecution). There is typically a short delay (2us on benchmark linux platform) between when the scheduler posts to the task semaphore and goes to sleep (via NanoSleep) and when the task awakens and begins execution.

Task End : **TaskEnd** – (task ID, actual absolute time, actual interval time, expected interval time)

Event is issued when a task signals it has completed its execution. There is typically a short delay (2us on our benchmark Linux platform) between when the task posts to the scheduler semaphore and waits on its own semaphore (then goes to sleep) and when the scheduler awakens from its wait on its semaphore.

Sporadic Job Creation : **SporadicJobCreation** – (task ID, actual absolute time, actual interval time)

Event is issued any time a sporadic job request is created. Currently, the name or ID of the task requesting the job creating is not recorded.

Sporadic Job Removal : **SporadicJobRemoval** – (task ID, actual absolute time, actual interval time)

Event occurs when the online scheduler prunes a sporadic job from the queue due to there not being enough time to execute the task before the jobs stated deadline.

Hyperperiod End : **HyperperiodEnd** – (actual absolute time, actual interval time)

Event occurs at the end of each hyperperiod from the start of execution through the end of execution.

Execution End : **ExecutionEnd** – (actual absolute time)

Event is issued when FS_Execute (SchedExecute) is finishes running. This can occur for many reasons: elapsed number of hyperperiods, fatal error, halt signal via Ctrl-C, halt signal received via channel.

3.1.2 Scheduler Errors

Sleep Deadline Miss (**FS_ERR_SLEEPDEADLINEMISS**) - Occurs when a call to the NanoSleep function results in the calling task resuming more than DELTA milliseconds from the desired awakening time.

Task WCET Overrun (**FS_ERR_TASKWCETOVERRUN**) - Occurs when a task thread notifies the scheduler thread of execution completion later than the task start time + task WCET.

Sporadic Deadline Miss (**FS_ERR_SPORADICDEADLINEMISS**) - Occurs when a sporadic task is pruned from the list of ready sporadic tasks due to it not being able to execute before its desired deadline.

Sporadic Too Often (**FS_ERR_SPORADICTOOOFTEN**) - Occurs when a sporadic tasks is requested to execute more often that is declared frequency.

Periodic Create Failure (**FS_ERR_PERIODICCREATEFAILURE**) - Occurs when the scheduler fails to properly create a periodic task.

Sporadic Create Failure (`FS_ERR_SPORADICCREATEFAILURE`) - Occurs when the scheduler fails to properly create a sporadic task.

Initialization Failure (`FS_ERR_INITFAILURE`) - Occurs when the scheduler fails to properly initialize.

Execution Time Outside Hyperperiod (`FS_ERR_EXEETIMEOUTSIDEHYPERPERIOD`) - Occurs when a periodic task is given a scheduled start time that will result in its execution beyond the end of a hyperperiod when WCET is accounted for (start time + WCET \geq hyperperiod length).

Sched_FIFO Unavailable (`FS_ERR_SCHEDFIFOUNAVAILABLE`) - Occurs when (on pthreads-based platforms) the scheduler thread is not able to set the underlying OS scheduling policy to `SCHED_FIFO`. This may indicate that the FRODO process is not running as root, that the OS does not allow `SCHED_FIFO` processes, or that the OS does not support `SCHED_FIFO` processes.

Sched Priority Not Set (`FS_ERR_SCHEDPRIORITYNOTSET`) - Occurs when the scheduler thread is not able to set the process priority of the overall FRODO process to the maximum allowed priority. This may indicate that the FRODO process is not running as root.

Set Affinity Failure (`FS_ERR_SETAFFINITYFAIL`) - Occurs when the scheduler thread is not able to set the processor affinity to processor 0 (FRODO currently assumes a single-core execution model).

3.2 Peripheral/Communication Channels

Independed nodes (computers) are connected via communication channels. These channels may be of a variety of peripheral device types (e.g. AD/DA, UDP, serial, I^2C). Each device type must implement or respond to a number of events and may generate a set of errors. The events are designed to be generic across all peripheral types as are a number of the error conditions. Additionally, each peripheral type may generate a number of additional type-specific errors.

3.2.1 Peripheral Events

Synchronous Send Begin : **SyncSendBegin** – (channel ID, message ID, actual absolute time, actual interval time, expected interval time)
Desc.

Synchronous Send End : **SyncSendEnd** – (channel ID, message ID, actual absolute time, actual interval time, expected interval time)
Desc.

Synchronous Receive Begin : **SyncReceiveBegin** – (channel ID, message ID, actual absolute time, actual interval time, expected interval time)
Desc.

Synchronous Receive End : **SyncReceiveEnd** – (channel ID, message ID, actual absolute time, actual interval time, expected interval time)
Desc.

Asynchronous Send Begin : **AsyncSendBegin** – (channel ID, message ID, actual absolute time, actual interval time)
Desc.

Asynchronous Send End : **AsyncSendEnd** – (channel ID, message ID, actual absolute time, actual interval time)
Desc.

Receive Begin : **ReceiveBegin** – (channel ID, message ID, actual absolute time, actual interval time)
Desc.

Receive End : **ReceiveEnd** – (channel ID, message ID, actual absolute time, actual interval time)
Desc.

Asynchronous Receive : **AsyncReceive** – (channel ID, message ID, actual absolute time, actual interval time)

Desc.

3.2.2 Peripheral Errors

Channel Create Failure (**FP_ERR_CHANNELCREATEFAIL**) - Occurs when a fatal error occurs while trying to create a channel. The exact nature of the error is channel-type (e.g. UDP) dependent.

Channel Destroy Failure (**FP_ERR_CHANNELDESTROYFAIL**) - Occurs when a fatal error occurs while trying to shutdown a channel. The exact nature of the error is channel-type (e.g. UDP) dependent.

P (**FP_ERR_UNEXPECTEDMSGIDRCV**) - Occurs when

P (**FP_ERR_UNEXPECTEDMSGIDSEND**) - Occurs when

P (**FP_ERR_WRONGMSGSIZE**) - Occurs when

P (**FP_ERR_FAILURETOSEND**) - Occurs when

P (**FP_ERR_FAILURETORECEIVE**) - Occurs when

3.2.3 UDP Channel Events

None - Implements the generic peripheral events on a UDP channel

UDP Channel Errors:

P (**UDP_ERR_BINDSOCKETFAIL**) - Occurs when

P (**UDP_ERR_CREATERECVSOCKETFAIL**) - Occurs when

P (**UDP_ERR_CREATESENDSOCKETFAIL**) - Occurs when

P (**UDP_ERR_CLOSESOCKETFAIL**) - Occurs when

P (**UDP_ERR_SELECTFAIL**) - Occurs when

P (**UDP_ERR_RECVFROMFAIL**) - Occurs when

P (**UDP_ERR_SENDTOFAIL**) - Occurs when

P (**UDP_ERR_PORTREADFAIL**) - Occurs when

3.2.4 Channel Synchronization Events

Sync Ready Send : **SyncReadySend** – (variables)

Desc.

Sync Ready Receive : **SyncReadyReceive** – (variables)

Desc.

Sync Acknowledge Send : **SyncAckSend** – (variables)
Desc.

Sync Acknowledge Receive : **SyncAckReceive** – (variables)
Desc.

Sync Start Send : **SyncStartSend** – (variables)
Desc.

Sync Start Receive : **SyncStartReceive** – (variables)
Desc.

Sync Halt Send : **SyncHaltSend** – (variables)
Desc.

Sync Halt Receive : **SyncHaltReceive** – (variables)
Desc.

Sync Suggest Offset : **SyncSuggestOffset** – (variables)
Desc.

3.2.5 Channel Synchronization Errors

Synchronization Failure (FP_ERR_SYNCFAIL) - Occurs when a fatal error occurs during channel synchronization