Small Satellite Research Laboratory
Franklin College of Arts and Sciences
UNIVERSITY OF GEORGIA

Scheduler Component Documentation

Prepared for
The Small Satellite Research Laboratory
Fall 2024 - Spring 2025

Principal Investigator
Dr. Deepak Mishra

The University of Georgia
Geography-Geology Department
Athens, Georgia
United States of America

Prepared by
The MEMESat-1 CDH Team
University of Georgia
Athens, Georgia
United States of America

Contents

1 List of Acronyms

BBS	Bulletin Board System
CDH	Command and Data Handling
EPS	Electrical Power System
GDS	Ground Data System
FSM	Finite State Machine
FSW	Flight Software
MCU	Microcontroller Unit
MS-1	MEMESat-1
OBC	On-Board Computer
OSAL	Operating System Abstraction Layer
OTA	Over-The-Air
OTAU	Over-The-Air Updater
TLM	Telemetry
UART	Universal Asynchronous Receiver-Transmitter
UHF	Ultra High Frequency

2 Revision History

Changes	**Authors**	**Version**
Initial Draft	Aiden Hammond	1.0.0
Updating ports and general overview	Aiden Hammond	1.0.1
[2024-01-14] Completing and editing code	Nektaria Karagiannis	1.0.2.
[2024-03-04] Unit tested to 94% line coverage and 100% function coverage.	Nektaria Karagiannis	1.0.3
[2024-03-28] Implemented Scheduling Format, Dependecies, and Diagrams sections.	Nektaria Karagiannis	1.0.4
[2024-04-03] Reviewed and updated for Feasibility Review. Added block diagrams and dependencies sections. Cleaned up format	Olivia Beattie	1.1.3

3 Purpose

This document overviews how to use the Scheduler component. It is important to understand that the component uses the external libcron library (stored in MEMESat-FSW/obc/ms1/libs/libcron). The Scheduler component is meant to be a simpler alternative for a scheduler and a work-around for rate groups, allowing for long-term schedules.

4 Overview

The component interacts with a static libcron cron object defined within the file (Components/Scheduler/Scheduler.cpp). The Scheduler acts similarly to a rate group, in that it holds two arrays, one wherein components are able to connect to the Scheduler and gives it their schedules, and one wherein the Scheduler calls on the component by their schedules.

The Scheduler is the task delegator of the system; alongside the rate groups defined in the F Prime architecture, the component causes actions to occur on a schedule that is defined later in a table in this document. The Scheduler is used to program tasks that happen at a rate slower than 1Hz (the limit of the rate groups). The Scheduler works by using a libcron 'tick' to invoke a port connected to the desired component. Each component connected to the Scheduler is instantiated through a 'preamble' function predefined by F Prime to allow for an active component's initialization functionality.

Each component connected to the Scheduler should have a 'sendSchedule' output port of type SchedulerModle.SendSchedulePort and a 'scheduledHandler' input port of type SchedulerModule.RunSchedulePort. The output port sends necessary data to the Scheduler, which, upon being received and validated by the Schedulers' 'getSchedule' input port, becomes added to the libcron:Cron objects' schedule. There is only one schedule allowed to be on the Scheduler's task list per component. The schedule is connected to the Scheduler's 'runSchedule' output port, which is connected back to the connected component's 'scheduledHandler' input port. At the times specified by the input schedulee, Cron will call on the internal schedule's task and invoke the RunSchedulePort.

Description of directory files:

- Scheduler.fpp: Contains the FPrimePrime language used to auto-generate files upon build.
- Scheduler.hpp: Contains headers for overridden functions auto-generated by FPrime.
- Scheduler.cpp: Contains implementations of overridden functions and custom functions.

4.1 Scheduling Format

Referencing the PerMalmberg/libcron github page and altered based on testing:

```
+ --------------------- second (0-59)
| + --------------------- minute (0-59)
| | + --------------------- hour (0-23)
| | | + --------------------- day of month (1-31)
| | | | + --------------------- month (1-12 or JAN-DEC)
| | | | | + --------------------- day of week (0-6 or SUN-SAT)
| | | | | |
| | | | | |
* * * * * *
```

4.1.1 Accepted Formats

- Special characters: '*', meaning the entire range.
- '?' used to ignore day of month/day of week as noted below.
- Ranges: 1,2,4-6 - result: 1, 2, 4, 5, 6
- Using strings instead of numbers:
  - 1-12 = JAN-DEC
  - 0-6 = SUN-SAT

4.1.2 Failed Formats

- Day of month and day of week are mutually exclusive so ensure that there is not an impossible mix of these fields.
- White spaces are used to separate parts of the schedule, and should not be used within the respective fields.
- "Convenience scheduling" as referred to in the PerMalberg/libcron github page is rejected by the current component configuration.

5 Design

5.1 Requirements

Requirement	**Description**	**Verification Method**
SCH-001	The scheduler must be able to call functions from other components via ports based on a given schedule for that function.	Unit Test
SCH-002	Schedules on the scheduler must be mutable	Unit Test
SCH-003	The scheduler must support up to 1 hz (executions/second)	Unit Test
SCH-004	The scheduler must be non-blocking	Implementation Test
SCH-005	Schedules on the scheduler must be available for downlink	Unit Test and Implementation Test
SCH-006	The scheduler must minimize the latency between the expected execution and actual execution of functions connected to the scheduler.	Implementation Test

5.2 Ports

The Scheduler component has four ports: two output ports, and two input ports. The DownlinkCurrentSchedules port which downlinks the schedules currently running. The runSchedule port invoked handlers on the components whose schedules are being invoked. The getSchedule input ports send schedules to the Scheduler from other components. This is connected to the TlmChanWrapper and the FileRecycler's sendSchedule port. The tick should be invoked once every second as per required by Libcron.

Port Data Type	Name	Direction	Kind	Usage
Svc SendFileRequest	DownlinkCurrentSchedules	Output	N/A	Downlinks all schedules currently running on the libcron::Cron object.
SchedulerModule RunSchedulePort	runSchedule	Output	N/A	Runs the handlers on connected components.
SchedulerModule SendSchedulePort	getSchedule	Input	Asynchronous	Adds a new schedule or changes a cron job in the libcron::Cron object by removing it and reading it under a new schedule. It has priority over other ports and has a queue size of 10 (Default queue size) and will block incoming calls once that queue has been reached until those calls have been handled.
Svc.Sched	tick	Input	Synchronous	Ticks the libcron::Cron object which then recalculates when the schedules need to be called.

5.3 Custom Types

The Scheduler component has four custom types. the ScheduleOp enum is defined to be either STOP or START depending on the schedule's intended operation. ScheduleStatus is logged in response to the Scheduler's success in adding or running the schedule.

Type	Name	Argument/Parameter	Description
Port	SendSchedulePort	name schedule action	Name of the schedule for the libcron task. Schedule to run the task on in Cron format. Defined by ScheduleOp
Port	RunSchedulePort	status	Defined by ScheduleStatus. Allows the component t invoke the handlers of connected components.
Enum	ScheduleOp	STOP START	
Enum	ScheduleStatus	RUNNING STOPPED FAILED	

6 Implementation

6.1 Accepted Formats: Examples

Schedule Expression	Description
0 0 12 * * MON-FRI	Every weekday at noon.
0,3,40-50 * * * * ?	Seconds specified by list and range, ignoring day of week.
0 0 12 * * ?	Daily at noon, day of week ignored.
0 0 12 ? * MON	Every monday at noon, the day of the month is ignored.
0 0 * * * MON-THU,SAT	Every Monday to Thursday and Saturday at midnight.
* * * ? * *	Every second.
0 0 * 15 * ?	15th day of every month at midnight, day of week ignored.

6.2 Failed Formats: Examples

Schedule Expression	Reason for failure
0 0 12-23 * * *	Specifies both the day of month and day of week. Does not use '?' for one of these two fields.
0, 3, 40-50 * * * * ?	Invalid white space characters within the seconds range.

Once the schedule has been send to the Scheduler's getSchedule port, the Scheduler determines is the task name has been previously established as a task. If it is preexisting, the schedule will be removed and then added. If it is new, it will just be added to the task list. Regardless, the format of the schedule is parsed and its validity is ensured by Libcron, and the result of the add_schedule's operation is logged if unsuccessful. If the ScheduleOp is ScheduleOp::STOP, the schedule is removed completely.

6.3 Dependencies

The Scheduler depends on the PerMalmberg/libcron library. This library is held in the MEMESat-FSW/obc/ms1/libs/libcron directory. The Scheduler uses libcron to store its scheduled tasks, and calls tick() once every second in order to update the timing and successfully invoke the connected components to complete their tasks.