Core Flight System (cFS) Training

Community Apps:

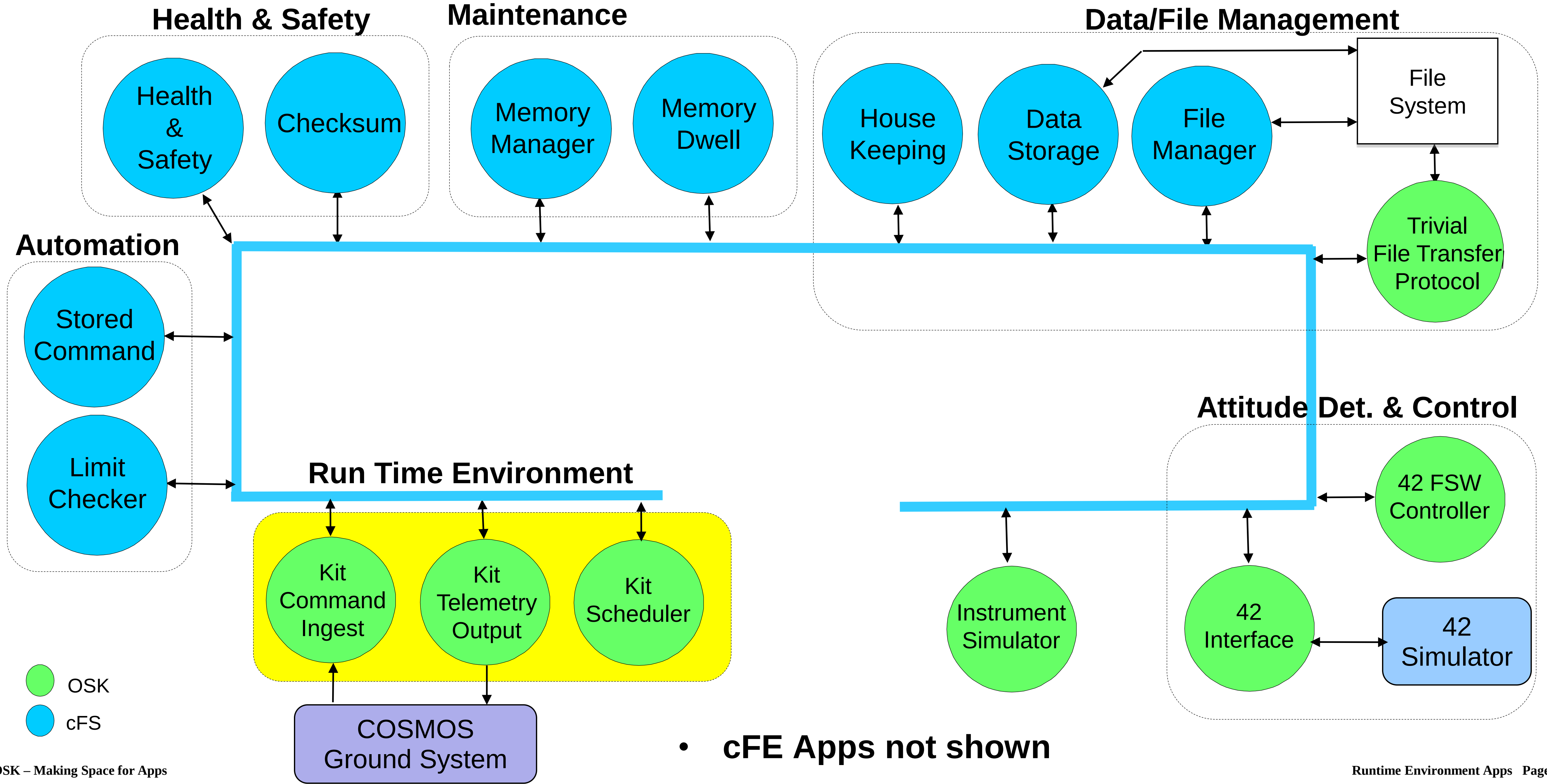
Runtime Environment

Runtime Environment App Agenda

- Runtime Environment App Overview
- Integrating Apps into the Runtime Environment
- Operational Scenarios
- OSK-to-Flight Transition
- NASA’s Open Source Scheduler(SCH) App

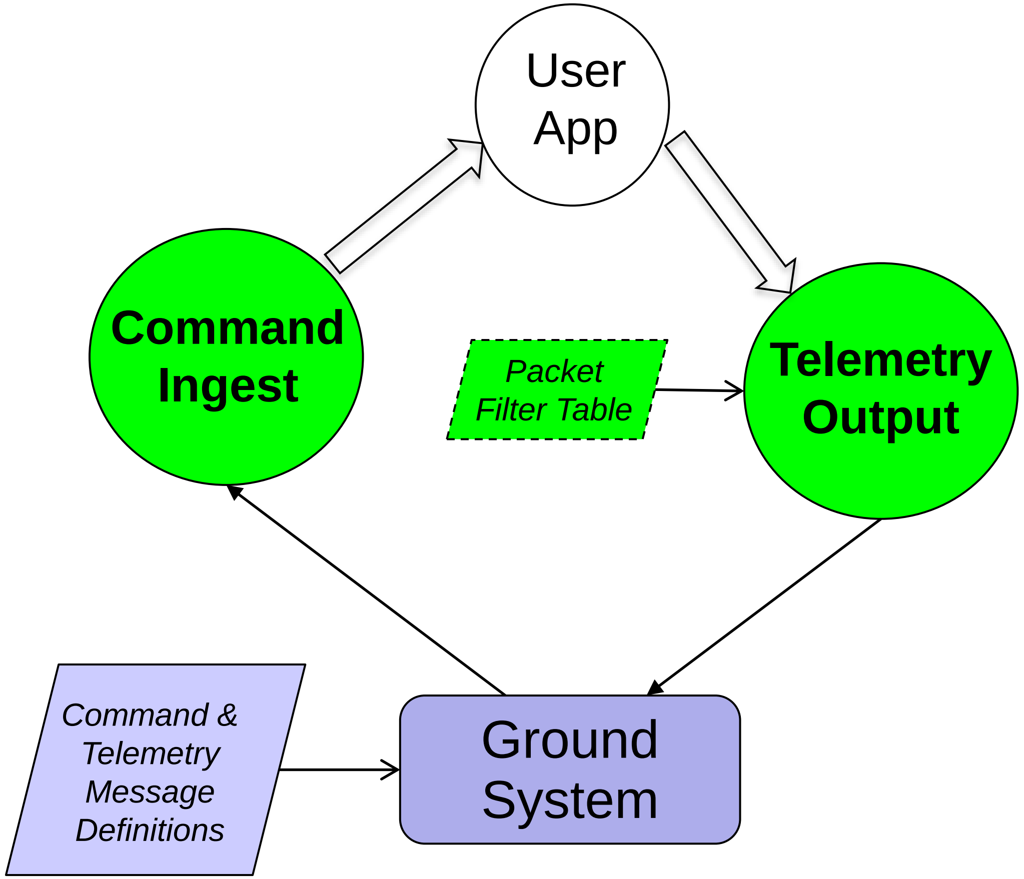
Runtime Environment Application Overview

OSK FSW SimSat Applications



- cFE Apps not shown

Command & Telemetry Context



- **Command Ingest (CI) App**
  - Receives commands from an external source, typically the ground system, and sends them on the software bus
- **Telemetry Output (TO) App**
  - Receives telemetry packets from a the software bus and sends them to an external source, typically the ground system
  - Optional *Packet Filter Table* that provides parameters to algorithms that select which messages should be output on the external communications link

- **Different versions of CI and TO used on different platforms**
  - cFS Bundle includes ‘lab’ versions that use UDP for the external comm
  - JSC released versions that use a configurable I/O library for a different external comm links
  - OSK versions use UDP and a JSON tables
  - ITAR-restricted flight versions typically used inflight

KIT_TO Packet Table

KIT_TO Packet Table Excerpt

```
"packet": {
   "name": "CFE_EVS_EVENT_MSG_MID",
   "stream-id": "\u0808",
   "dec-id": 2056,
   "priority": 0,
   "reliability": 0,
   "buf-limit": 40,
   "filter": { "type": 2, "X": 1, "N": 1, "O": 0}
},
"packet": {
   "name": "CFE_EVS_HK_TLM_MID",
   "stream-id": "\u0801",
   "dec-id": 2049,
   "priority": 0,
   "reliability": 0,
   "buf-limit": 4,
   "filter": { "type": 2, "X": 1, "N": 1, "O": 0}
},
"packet": {
   "name": "CFE_SB_HK_TLM_MID",
   "stream-id": "\u0803",
   "dec-id": 2051,
   "priority": 0,
   "reliability": 0,
   "buf-limit": 4,
   "filter": { "type": 2, "X": 1, "N": 1, "O": 0}
},
```

- **Default file: *cfs/osk_defs/cpu1_osk_to_pkt_tbl.json***
- **Array of packet objects**
- **KIT_TO subscribes to receive all packets defined in the table using id, priority, reliability, and buf-limit**

- **Filter allows "N of X" packets to be sent starting at offset 0**

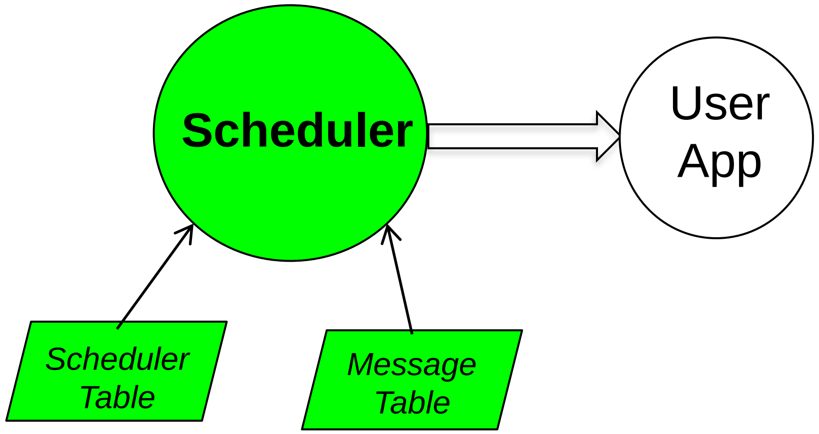
  1 = Time based (rare)

  2 = Sequence count based

  ((sequence count + O) modulo X) < N

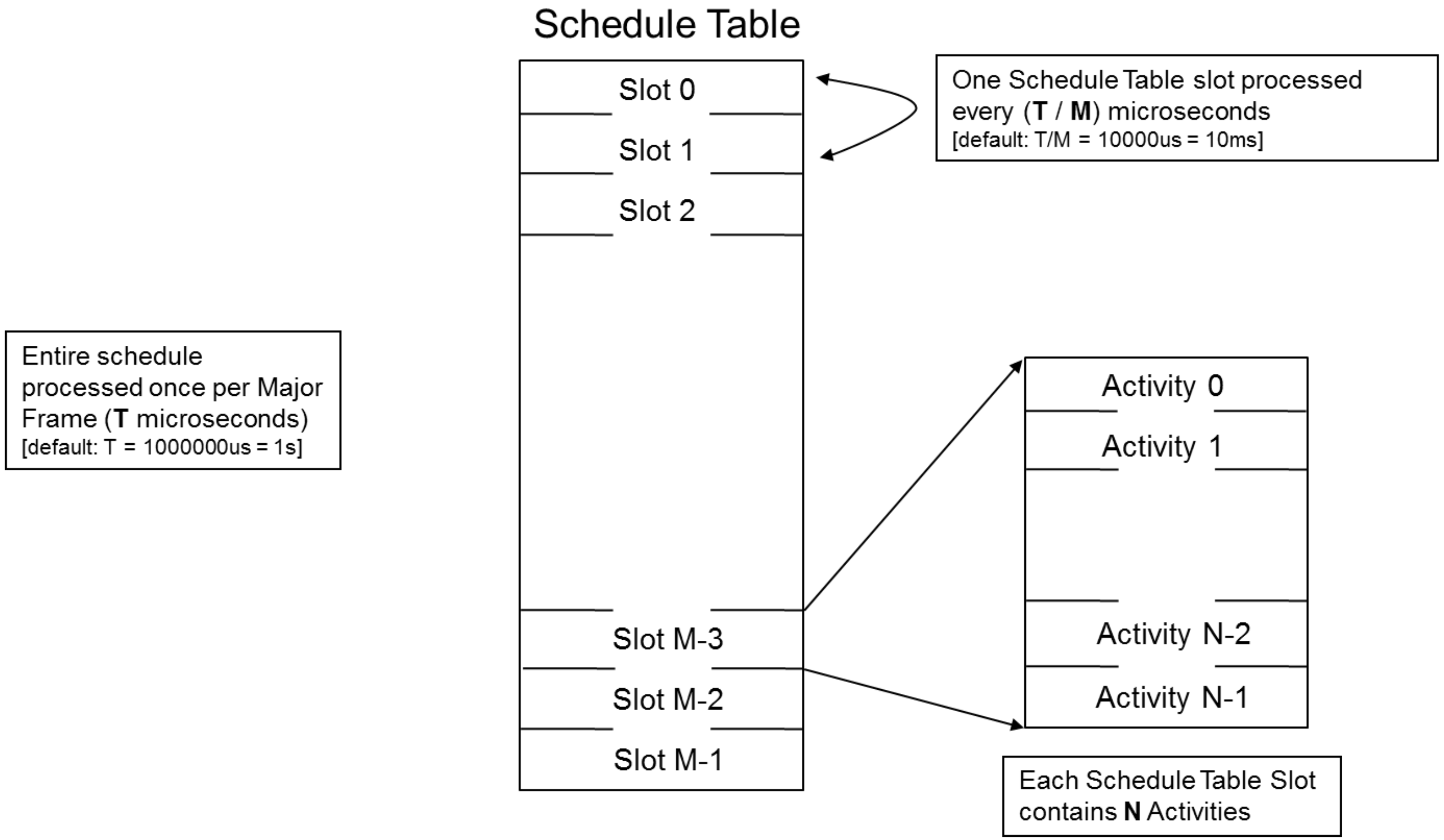
- **If (X=0) then do not send the packet**

Application Scheduling Context



- **Scheduler (SCH) App**
  - Synchronizes execution with clock's 1Hz signal
  - Sends software bus messages defined in the *Message Table* at time intervals defined in the *Scheduler Table*

- **Application Control Flow Options**
  - Pend indefinitely on a *SB Pipe* with subscriptions to messages from the Scheduler
    - This is a common way to synchronize the execution of most of the apps on a single processor
    - Many apps send periodic "Housekeeping" status packets in response to a "Housekeeping Request message from Scheduler
  - Pend indefinitely on a message from another app
    - Often used when an application is part of a data processing pipeline
  - Pend with a timeout
    - Used in situation with loose timing requirements and system synchronization Is not required
    - The SB timeout mechanism uses the local oscillator so the wakeup time may drift relative to the 1Hz



- SCH maintains a count of number of times entire table is processed and reports it in housekeeping telemetry

KIT_SCH Scheduler Table

(Slot 1 Excerpt)

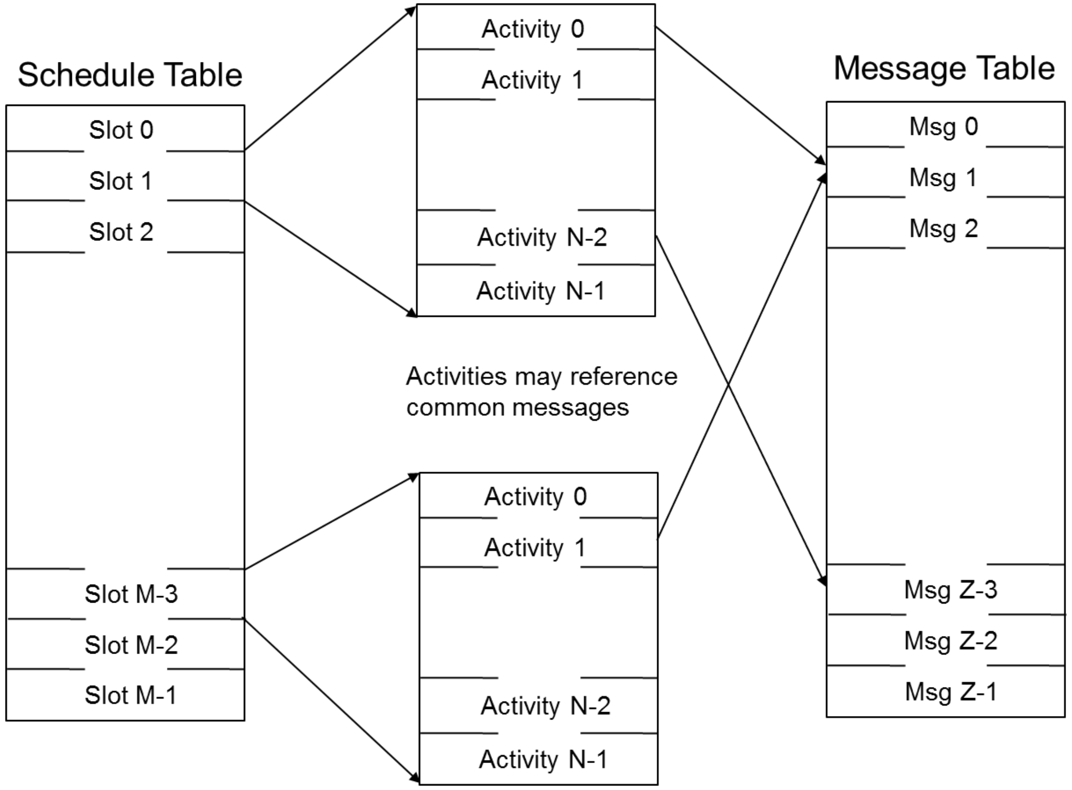
```
{"slot": {

   "index": 1,
   "activity-array" : [

      {"activity": {
         "name":     "LC Execution(Sample AP)",
         "descr":    "",
         "index":    0,
         "enabled":  "false",
         "period":   1,
         "offset":   0,
         "msg-idx":  17
      }},

      {"activity": {
         "name":     "HK Combined Pkt 1",
         "descr":    "Period can be shortened
         "index":    1,
         "enabled":  "true",
         "period":   4,
         "offset":   2,
         "msg-idx":  24
      }},
```

SCH – Activity Messages



KIT_SCH Message Table

```
{"message": {
   "name":  "LC_SAMPLE_AP_MID",
   "descr": "0x18A6(6310), 0xC000(49152), 0x0005, [fc|cs, Start Inc
   "id": 17,
   "stream-id": 6310,
   "seq-seg": 49152,
   "length": 7
   "data-words": "0,0,10,0"
}},

{"message": {
   "name":  "MD_SEND_HK_MID",
   "descr": "0x1891(6289), 0xC000(49152), 0x0001",
   "id": 18,
   "stream-id": 6289,
   "seq-seg": 49152,
   "length": 1
}},

{"message": {
   "name":  "MD_WAKEUP_MID",
   "descr": "0x1892(6290), 0xC000(49152), 0x0001",
   "id": 19,
   "stream-id": 6290,
   "seq-seg": 49152,
   "length": 1
}},
```

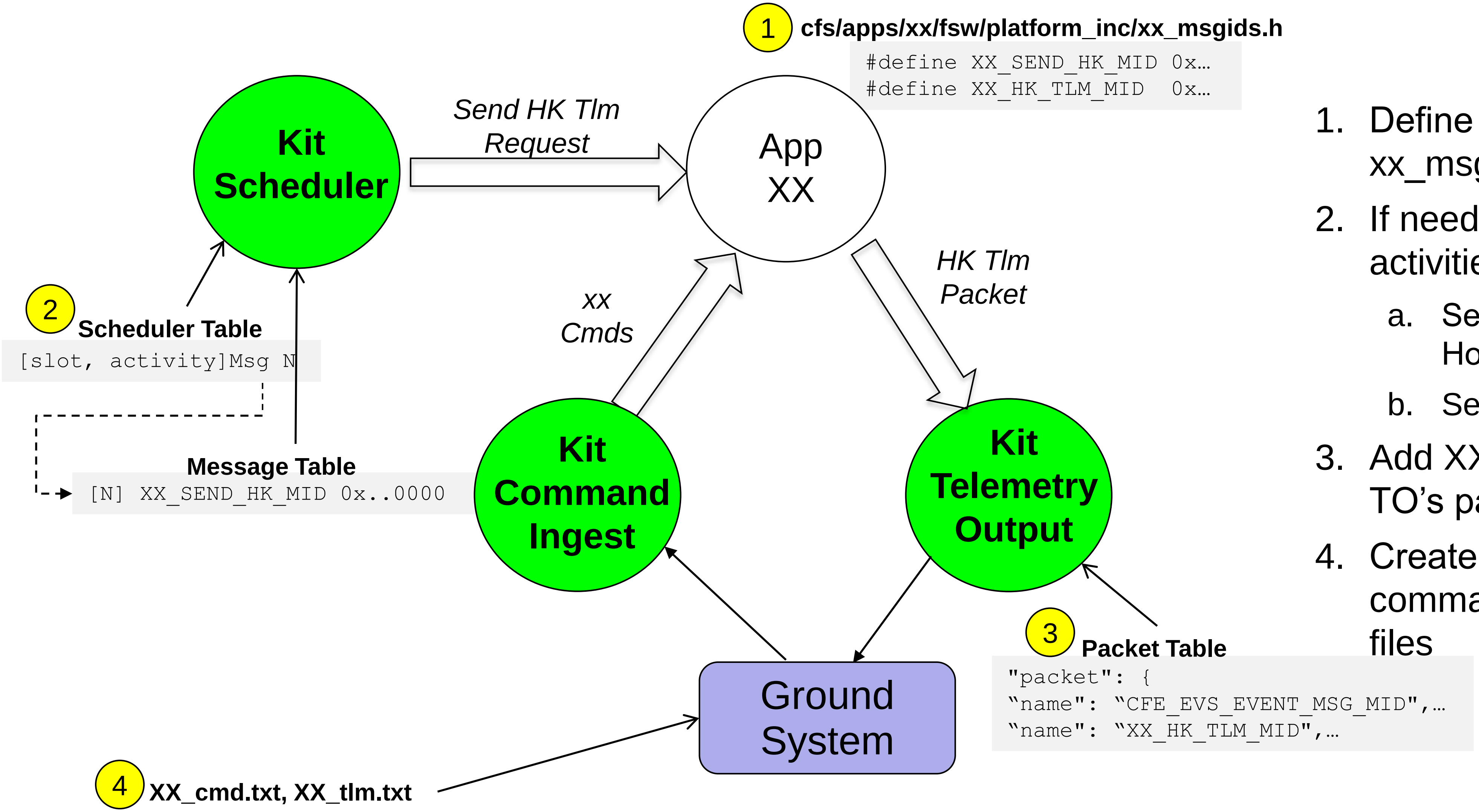
Limit Checker messages sent by KIT_SCH

Integrating Apps into the Runtime Environment

Integrating a single app

Design a scheduler-based system

Integrating a Single App



1. Define XX message ID's in its xx_msgids.h file
2. If needed, add messages & activities to Scheduler tables
   a. Send XX's "Send Housekeeping Tlm Request"
   b. Send XX wakeup/execute
3. Add XX telemetry messages to TO's packet table
4. Create XX's COSMOS command & telemetry definition files

```
COMMAND <%= @APP_PREFIX_STR %> <%= Osk::CMD_STR_NOOP %> <%= Osk::Cfg.processor_endian %> "Generate an event message with app version"
<%= Osk::Cfg.cmd_hdr(@APP_PREFIX_STR, @CMD_MID_STR, 0, 0) %> • • •
TELEMETRY XX HK_TLM_PKT <%= Osk::Cfg.processor_endian %> "Housekeeping Packet"
<%= Osk::Cfg.tlm_hdr(@APP_PREFIX_STR, @HK_TLM_MID_STR) %> • • •
```

OSK Message Definition Files

Core Flight System

cfs/osk_defs/

cpu1_osk_sch_msg_tbl.json

cpu1_osk_to_pkt_tbl.json

cfs/apps/xx/fsw/platform_inc/

xx_msgids.h

COSMOS

cosmos/config/targets/XX/cmd_tlm/

cosmos/lib

fsw_msg_id.rb

xx_cmd.txt

xx_tlm.txt

Manual process to keep files in synch

Long term goal is for a single message definition location and automatic file generation for all dependencies

Designing a Scheduler-based System

- **What are the hard real-time requirements?**
  - They’re often driven by
    - Attitude Determination and Control (ADC) and Guidance Navigation and Control (GNC) controller frequencies
    - Payload control and data management cycles
- **What control and data flows are involved with the real-time requirements?**
  - What are the hardware devices and how do they interface with the processor?
  - Does the scheduler app need to coordinate I/O?
    - Some architectures use a distributed1Hz timing pulse to synchronize the processor with sensors, actuators and payloads
    - Since the schedular app can be synchronized to the 1Hz timing pulse it can be used to manage a synchronous system
- **Are there relative hardware timing requirements?**
  - For example a rate measurement may need to correlated to a star tracker measurement
- **What apps require scheduled messages?**
  - Are there optimal relative timings between apps?
- **After the initial system is designed and prototyped it can measured and tuned**

Main Loop Control for Community Apps

Application	Main Loop Control	Control Notes
CF – CFDP	Pend Forever	Scheduler wakeup and HK request
CS – Checksum	Pend Forever	Scheduler wakeup and HK request
DS - Data Storage	Pend Forever	Subscribed message wakeup and HK request
F42 - 42 FSW Controller	Pend with timeout	Pends for sensor data packet from I42
FM – File Manager	Pend Forever	Ground Command, Scheduler HK request
HK - Housekeeping	Pend Forever	Scheduler combo pkt request and HK request
HS – Health & Safety	Pend with timeout	Scheduler HK request, no scheduler control
I42 – 42 Simulator I/F	Synched with 42	Flight equivalent depends upon H/W interfaces
KIT_CI – Command Ingest	Task Delay, Socket	
KIT_SCH – Scheduler	Synched with CFE_TIME	
KIT_TO – Telemetry Output	Pend with timeout	Subscribed message wakeup and HK request
LC – Limit Checker	Pend Forever	Scheduler wakeup and HK request
MD – Memory Dwell	Pend Forever	Scheduler wakeup and HK request
MM – Memory Manager	Pend Forever	Ground Command, Scheduler HK request
SC – Stored Command	Pend Forever	Scheduler wakeup and HK request
TFTP	Task Delay, Socket	Simulation environment (see CF for flight app)

SimSat Scheduler Table

Activities[2]

Slot 0	Slot 1	Slot 2	Slot 3
Collect Sensor Data[1]	Run Limit Checker	Collect Sensor Data[1]	Run Limit Checker
Run Stored Command	Send HK Combo Pkt	Run Health & Safety	Send HK Combo Pkt
Run Memory Dwell		Run Memory Dwell	
Checksum		Collect Payload Data	
	Run File Transfer App		Run File Transfer App
Send App HK Requests[3]	Send App HK Requests	Send App HK Requests	Send App HK Requests

1. Simplified assumption that sensor data initiates a ‘pipeline’ of app executions to perform attitude determination & control and guidance navigation & control
   a. Sensor and actuator interfaces drive the
2. All activities in a slot are executed immediately without delays
   a. Slots are used to control delays
3. Offsets can be used to help load balance a particular slot for activities with a period longer than 1 Hz
   a. HK requests are often on the order of 3-5 seconds

KIT_CI

Parameter	Description	Default Value
KIT_CI_RUNLOOP_DELAY	Delay in milliseconds between main loop executions	500
KIT_CI_RUNLOOP_MSG_READ	Number of messages read during each execution	10
KIT_CI_PORT	UDP port	1234
UPLINK_RECV_BUFF_LEN	Maximum bytes in an uplink message	1024

KIT_SCH

Parameter	Description	Default Value
SCHTBL_SLOTS	Number of divisions within a Major Frame (1s period)	4
SCHTBL_ACTIVITIES_PER_SLOT	Number of activities performed per slot	15
MSGTBL_MAX_ENTRIES	Maximum number of messages defined in message table	200
MSGTBL_MAX_MSG_WORDS	Maximum number of words defined in a message table message	8

Configuration Parameters (2 of 2)

KIT_TO

Parameter	Description	Default Value
KIT_TO_RUN_LOOP_DELAY_MS	Delay in milliseconds between main loop executions	500
KIT_TO_MIN_RUN_LOOP_DELAY_MS	Minimum commanded value for run loop delay	200
KIT_TO_MAX_RUN_LOOP_DELAY_MS	Maximum commanded value for run loop delay	10000
KIT_TO_TLM_PORT	UDP port	1235

Operational Scenarios

FSW Runtime Environments (1 of 2)

Software in the Loop (SIL)

Simulation

HW Cmds

HW Data

FSW

Cmds

Tlm

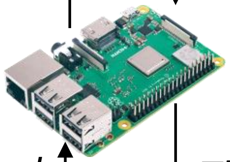
"Ground System"
(Test Scripts)

Processor in the Loop (PIL)

Simulation

HW Cmds

HW Data

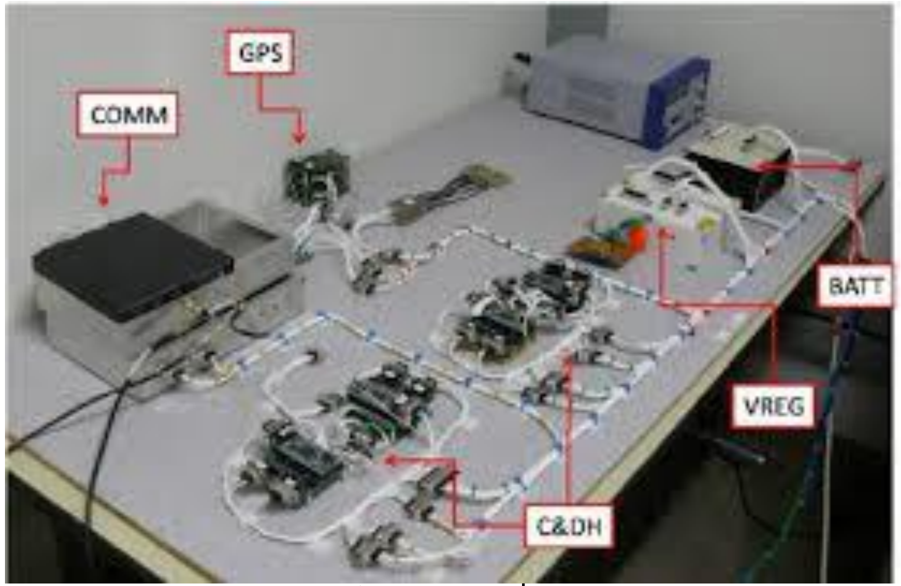


Cmds

Tlm

"Ground System"
(Test Scripts)

FlatSat



Cmds

Tlm

Ground System
(Test Scripts)

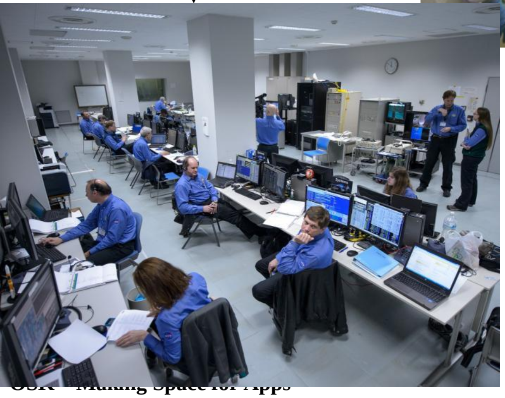
FSW Runtime Environments (2 of 2)

Spacecraft Integration & Test

Serial Link

Embedded Processor Host

Comm Hardware



Operational Ground System

Flight Operations

Cmds

Tlm

Ground System
(Operational Scripts)

FSW Test Environment Considerations

- Objectives
- Fidelity
- Interfaces
- Test Reuse & Repeatability

Kit App Operational Considerations

- **The runtime apps are configured during the build process for each test/operational environment and typically don't require much operational reconfiguration**
- **TO's packet filter table is the one exception**
  - Multiple TO packet tables can be created for different downlink rates and real-time packet selection
  - These table definitions are coordinated with the stored packet definitions managed by an app such as Data Storage
- **KIT_TO commands**
  - Enable telemetry, Set run loop delay
  - Add/remove packet, Update a packet's filter
  - Send a packet table entry telemetry packet
- **KIT_SCH commands**
  - Enable/disable scheduler [slot, activity] entry
  - Load message/scheduler table entry
  - Send message/scheduler table entry telemetry packet
- **NASA's SCH app commands**
  - Resynch to 1Hz, Enabled/disable groups

Packet Table Filter Demo

KIT_SCH: Scheduler Table
(tst1_to_sch_tbl.json)

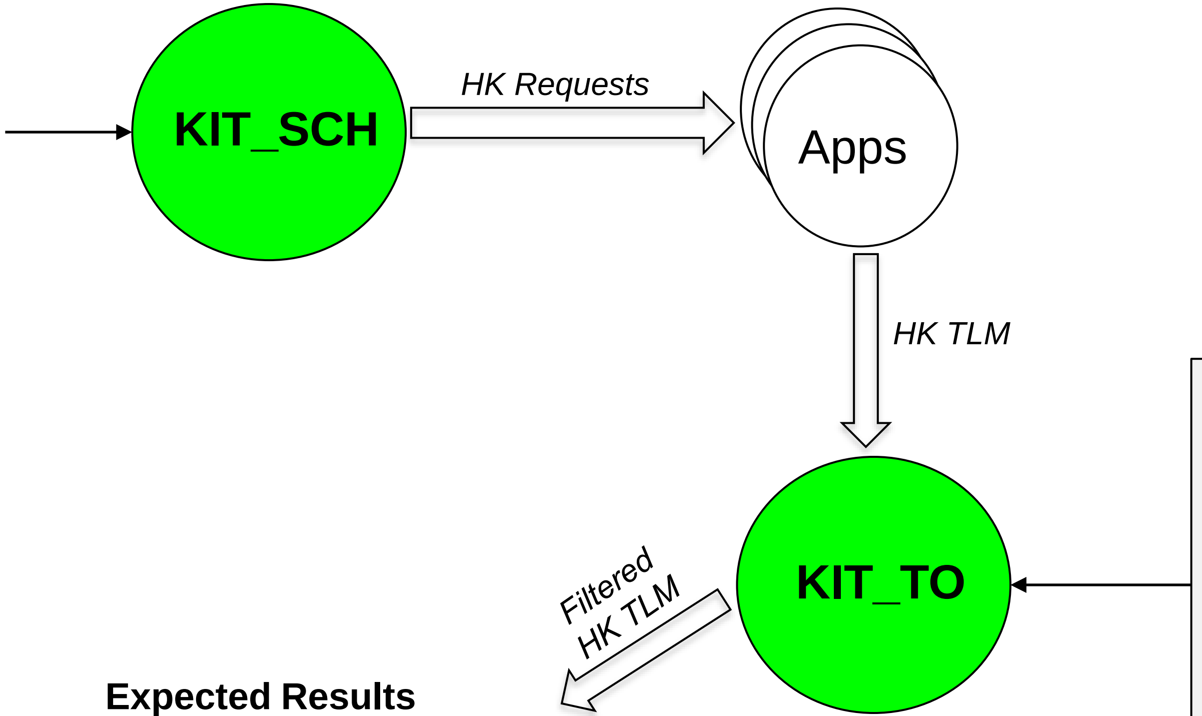
Slot 0
- cFE ES HK Request
- cFE EVS HK Request

Slot 1
- cFE SB HK Request
- cFE TBL HK Request

Slot 2
- cFE TIME HK Request
- cFE KIT_CI HK Request

Slot 3
- cFE KIT_SCH HK Request
- cFE KIT_TO HK Request

All HK requests sent at 1Hz



KIT_TO: Packet Table
(tst1_to_sch_tbl.json, tst4_to_sch_tbl.json)

Packet Array
- CFE_ES_HK_TLM_MID
- CFE_EVS_HK_TLM_MID
- CFE_SB_HK_TLM_MID
- CFE_TBL_HK_TLM_MID
- CFE_TIME_HK_TLM_MID
- CFE_KIT_CI_HK_TLM_MID
- CFE_KIT_SCH_TLM_MID
- CFE_KIT_TO_TLM_MID

Two test tables:
Tst1_ - Sends all packets
Tst4 – - Sends 1 of 4 packets

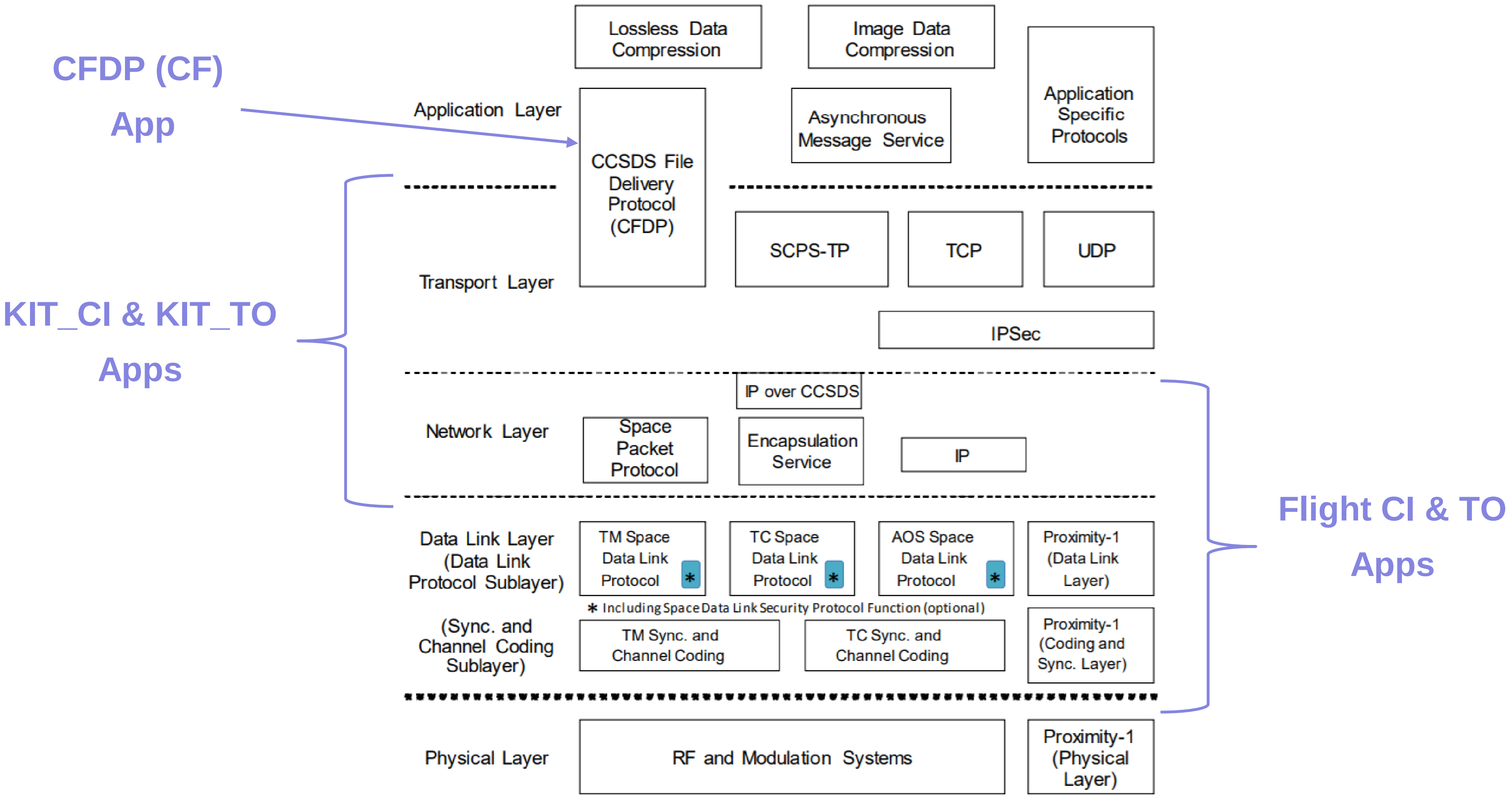
Expected Results

Test Case	Packets Per Sec	Bytes Per Sec
1Hz	8	842
0.25Hz	2	210

cosmos/config/targets/SIMSAT/procedures/demo_runtime_to_stats.rb

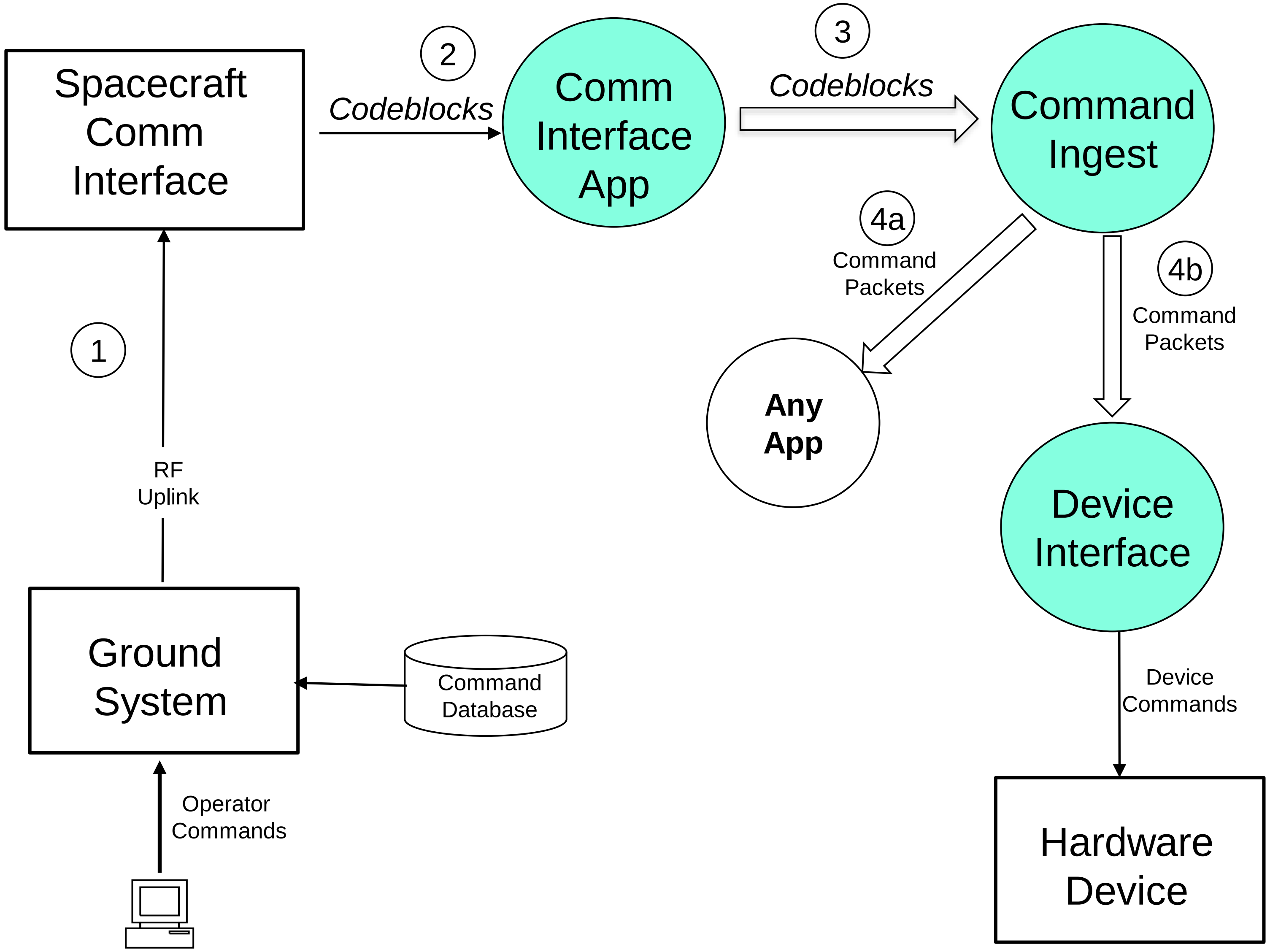
OSK-to-Flight Guidelines

CCSDS Space Communication Protocol Overview



Command Ingest Flight Context

1) Commands sent from ground system are received by Comm Interface
2) Comm Interface processes Communications Link Transmission Units and sends Code Blocks to Comm Interface App
3) Comm Interface App extracts, validates and forwards Code Blocks to CI
4) CI constructs, validates and processes transfer frames, then extracts, validates and sends Command Packets on the software bus
   a) Command packets processed by apps
   b) Command packets processed by hardware devices. A device interface routes the commands to the device and performs any necessary reformatting if the device does not interpret CCSDS command packets

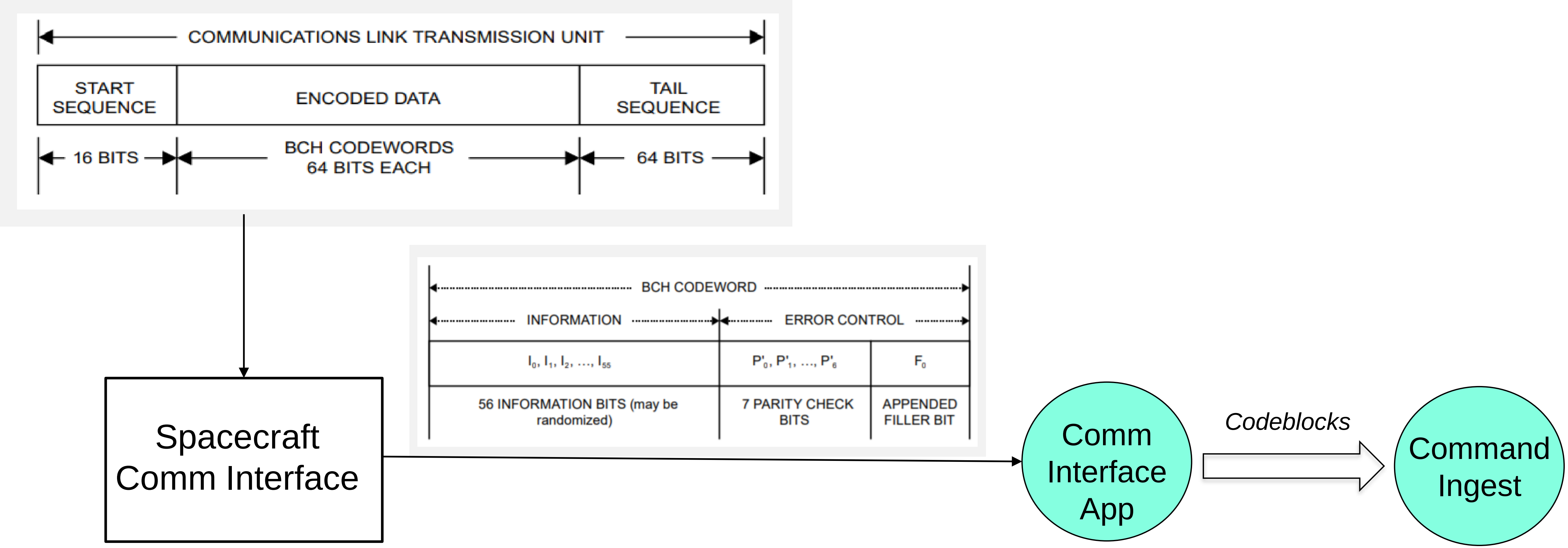


Flight Command Ingest (CI) Overview

- These slides describe a 4 step process that a flight CI app would perform to implement a minimal implementation of the CCSDS telecommand standards
- The CCSDS standards provide several implementation variations that trade quality attributes such as scalability, efficiency, security and complexity
- There are many heritage ground and flight systems that can be used as starting points
- Each step should provide buffers that retain all intermediate data for troubleshooting & diagnostics

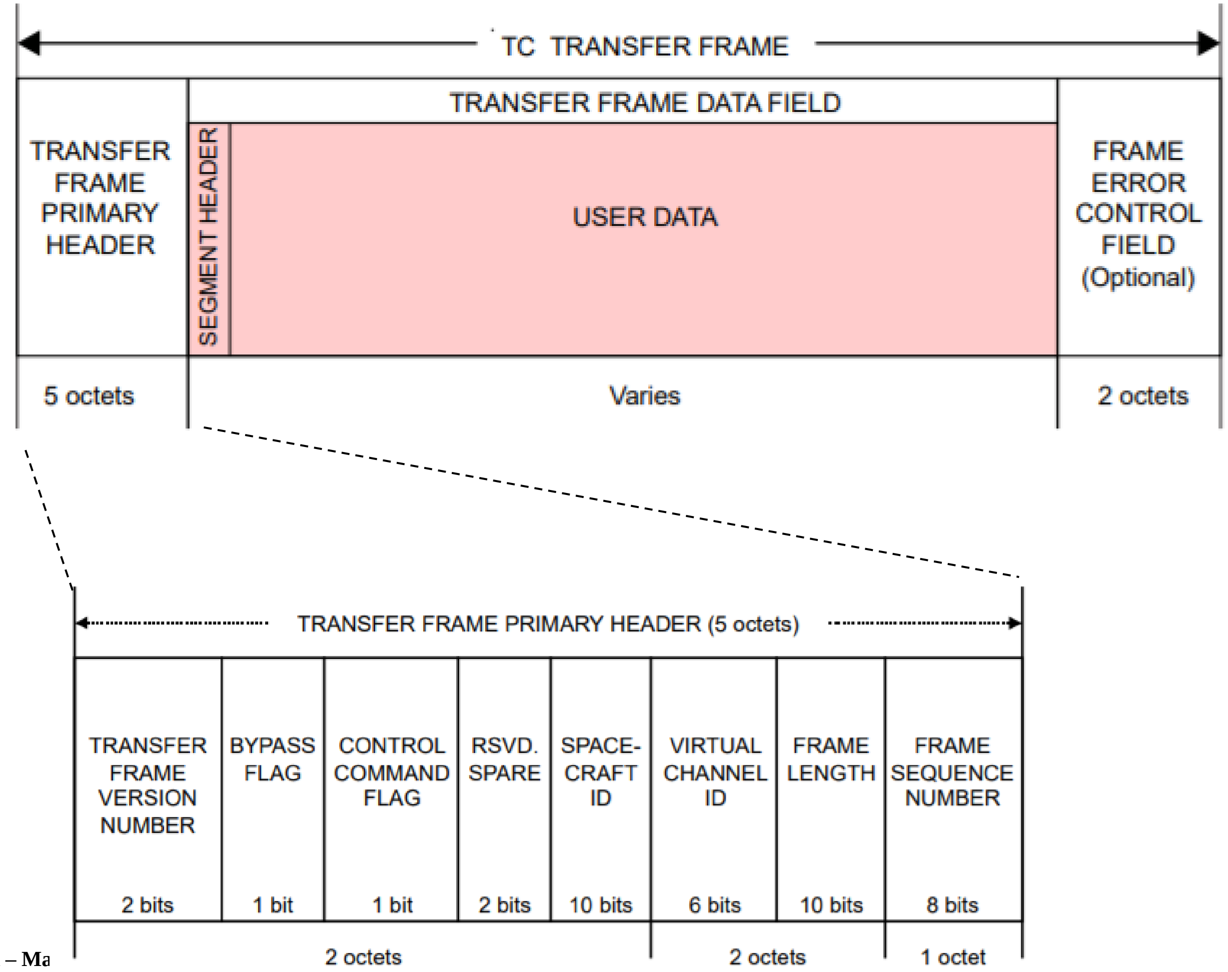
Flight Command Ingest Step 1

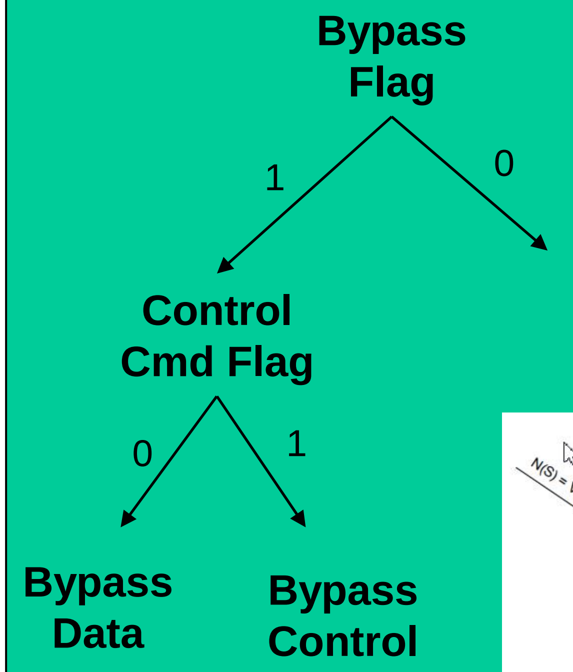
1. Receive and validate code blocks
   a. Defined in CCSDS 231.0-B-3, Telecommand Synchronization and Channel Coding
   b. A separate comm interface app is typically required that sends “BCH Codewords” aka Codeblocks to CI



Flight Command Ingest Step 2

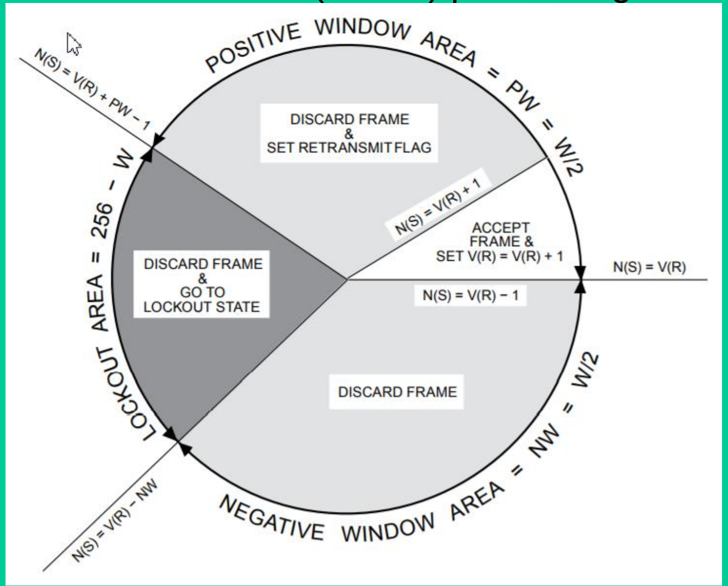
2. Construct, validate and process transfer frame
   a. Transfer frame defined in CCSDS 232.0-B-3, Telecommand Space Data Link Protocol
   b. Data acceptance algorithms defined CCSDS 232.1-B-2, Communication Operation Procedure-1





Acceptance Data

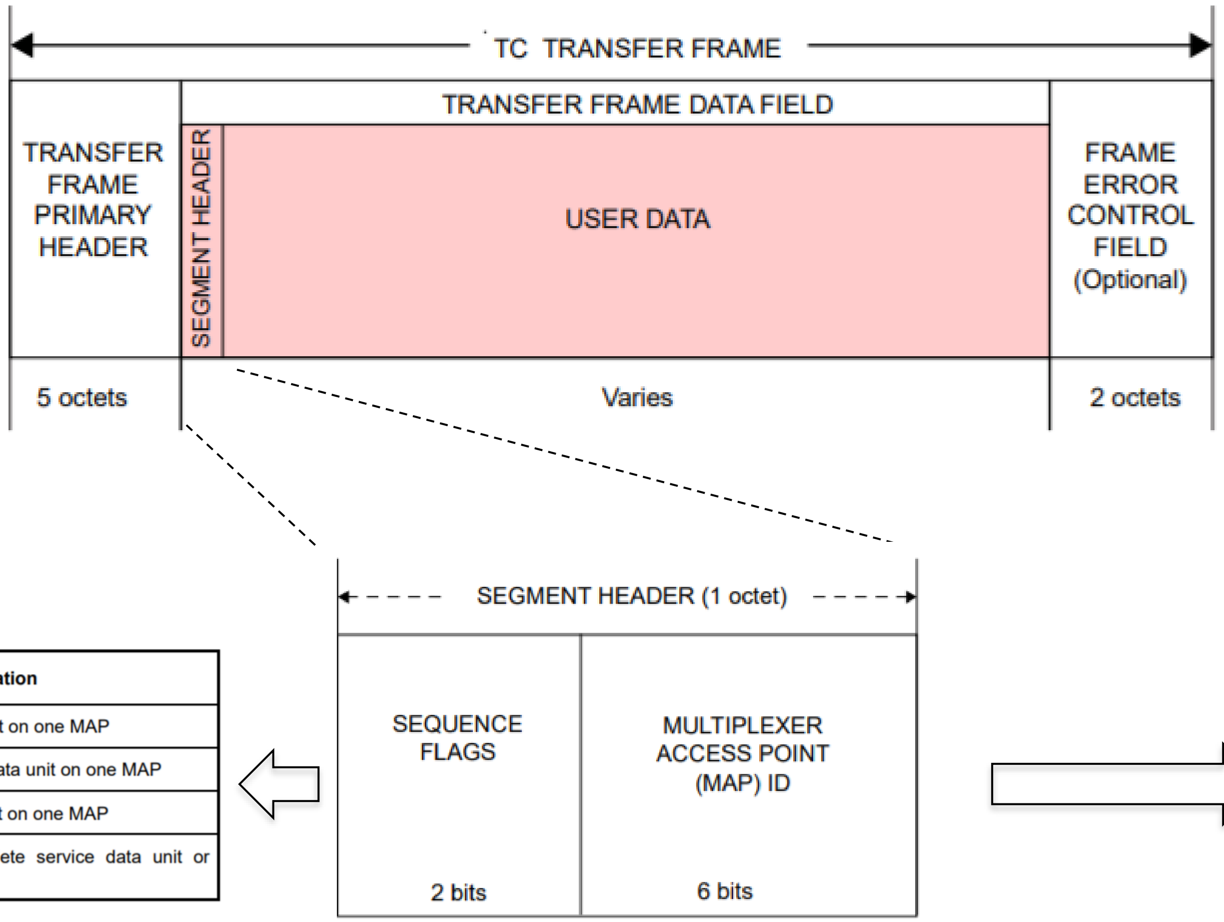
Perform Command Operation Procedure (COP-1) Frame Acceptance and Reporting Mechanism (FARM) processing



Flight Command Ingest Step 3

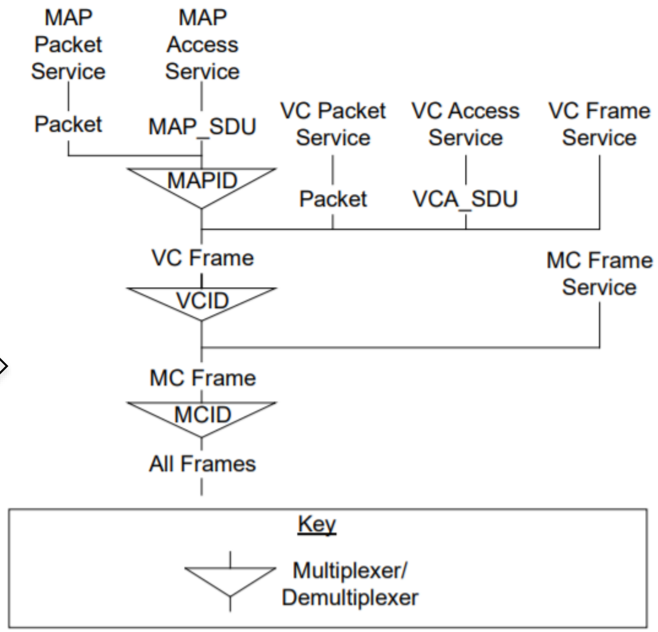
3. Validate and process segmentation header
   a. Defined in CCSDS 232.0-B-3, Telecommand Space Data Link Protocol

Simplest approach is no segmentation, single MAP ID and Virtual Channel



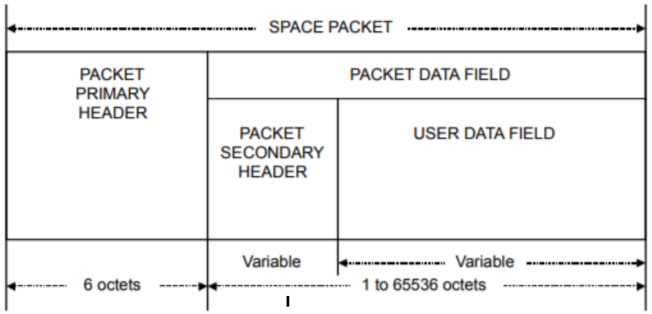
Sequence Flags

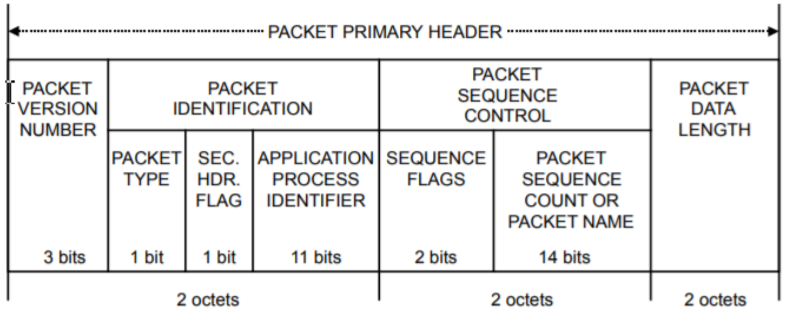
Bit 0	Bit 1	Interpretation
0	1	First portion of service data unit on one MAP
0	0	Continuing portion of service data unit on one MAP
1	0	Last portion of service data unit on one MAP
1	1	No segmentation (one complete service data unit or multiple Packets)



Flight Command Ingest Step 4

4. Command packet validation
   a. Defined in CCSDS 133.0-B-2, Space Packet Protocol
   b. Validate packet length
   c. Validate checksum defined in cFS command secondary header
   d. Send command packet on the software bus



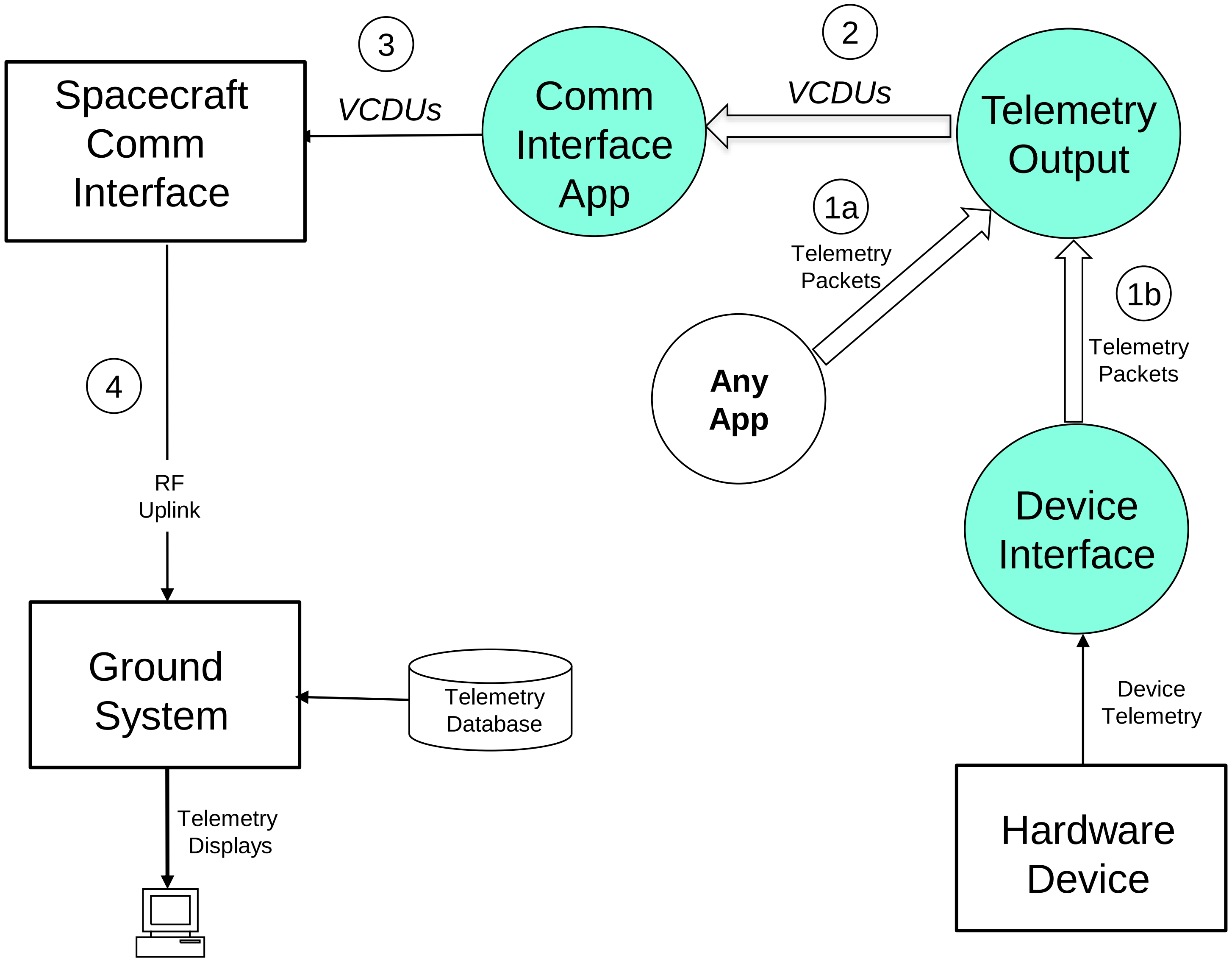


cfs/cfe/fsw/cfe-core/src/inc/ccsds.h

```
/*----- CCSDS command secondary header. -----*/
typedef struct {
   uint16  Command;     /* command secondary header */
      /*  bits  shift   ------------ description ---------------- */
      /* 0x00FF    0  : checksum, calculated by ground system     */
      /* 0x7F00    8  : command function code                    */
      /* 0x8000   15  : reserved, set to 0                       */
} CCSDS_CmdSecHdr_t;
```

Telemetry Output Flight Context

1) **Telemetry routed to TO on the software bus**
   a) **Apps send telemetry packets**
   b) **Device Interface apps collect hardware telemetry and package it into CCSDS telemetry packets if needed**
2) **TO collects, filters, builds and sends real-time AOS Transfer Frame on the software bus**
3) **Comm Interface App packages and sends AOS Transfer Frames to Comm Interface**
4) **Telemetry is received by the ground system from communication hardware bus**



Flight Telemetry Output (TO) Overview

- Additional considerations
  - Virtual channel management
  - Security
  - Fill frames
  - Real-time vs playback data
- Data rate management between and between apps sending data to TO
- Defined in CCSDS 133.0-B-2, Space Packet Protocol
- Defined in CCSDS 732.0-B-3, Advance Orbiting System (AOS) Space Data Link Protocol
- Historical Documents
  - VCDU

- **Telemetry Packet**

Packet Identification				Packet Sequence Control		Packet Length	Secondary Header			Packet Data
Version (3)	Type (1)	Sec Hdr Flag (1)	Application ID (11)	Segmentation Flag (2)	Source Sequence Count (14)	Pkt Length (16)	Res. (1)	Seconds (31)	Sub Seconds (16)	Application Data (n*8)

. . . .

- **VCDU Data Field**

X’0000’ – Pkt Hdr at first byte

X’07FF’ – No Pkt Hdr in frame

- **X’07FE’ – Fill Pkt only**

M_PDU Header		M_PDU Packet Zone
Spare (5)	First Header Ptr (11)	Source Packet Data 1770 Octets (1770 x 8 = 14160 bits)

Real-Time VCDU (X’7FC000000000’)

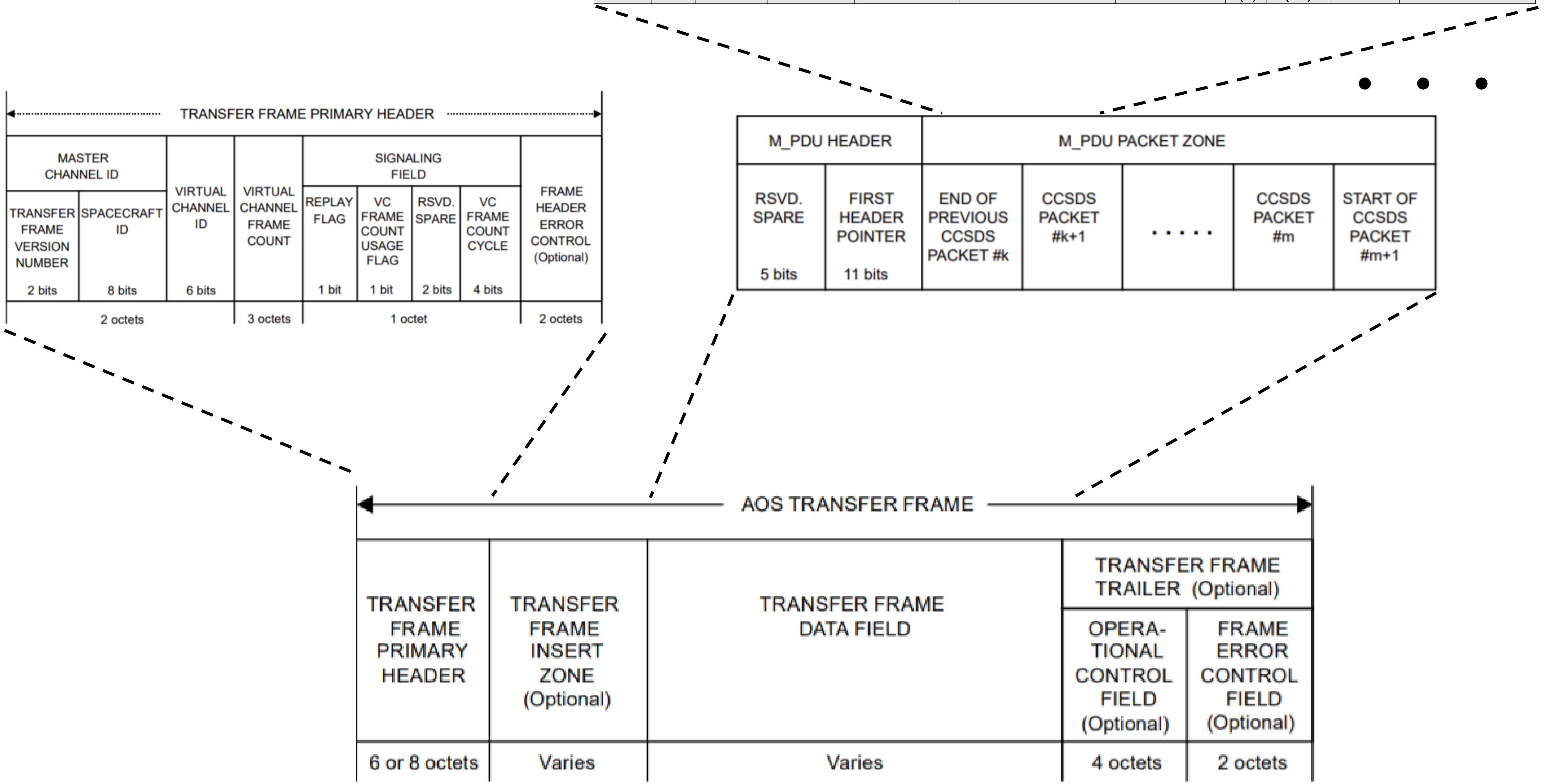
VCDU Primary Header							VCDU Data	VCDU Trailer	
Version AOS=1 (2)	VCDU ID		VCDU Counter (24)	Replay Flag RT=0 (1)	Time Corr Bit (1)	Spare (6)	1772 Octets (1772 x 8 = 14176 bits)	CLCW (32)	CRC (16)
	Space-craft ID (8)	VC ID (6)							

Packet Header	SpaceWire Cargo	
	Synchronization Marker 0x1ACFFCID (4 Bytes)	Downlink Data (1784 Bytes)

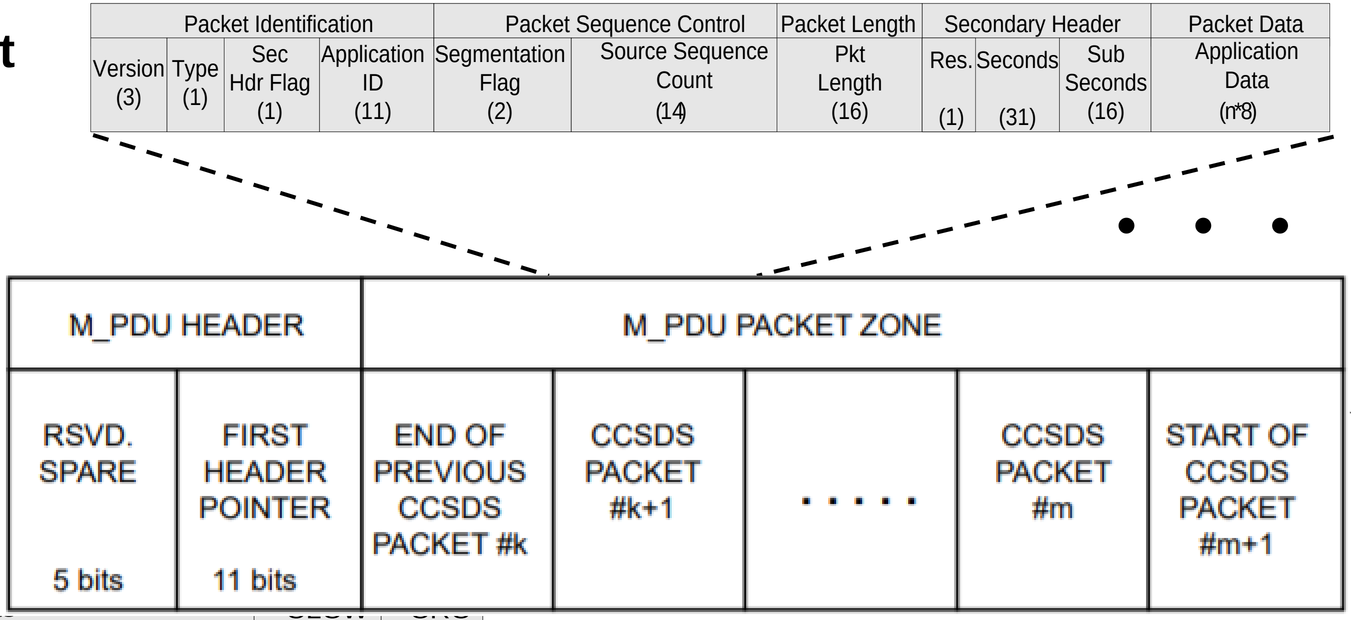
Message Sent to SW application

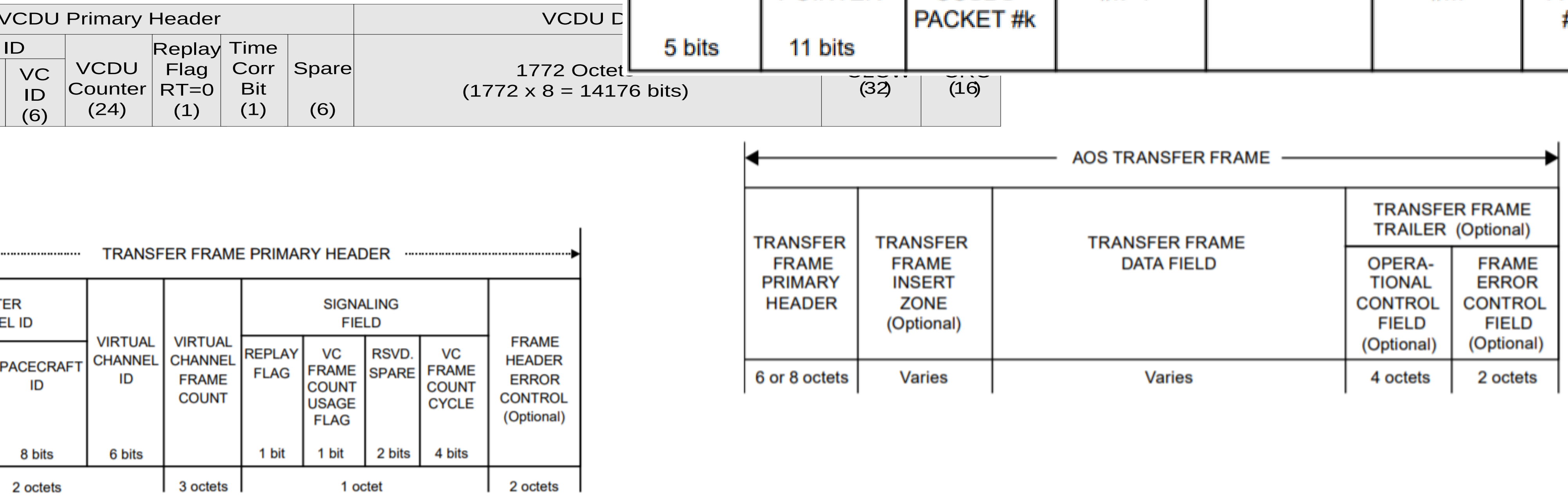
Telemetry Packet

Packet Identification				Packet Sequence Control		Packet Length	Secondary Header			Packet Data
Version (3)	Type (1)	Sec Hdr Flag (1)	Application ID (11)	Segmentation Flag (2)	Source Sequence Count (14)	Pkt Length (16)	Res. (1)	Seconds (31)	Sub Seconds (16)	Application Data (n*8)



Telemetry Packet





Historical Changes

- **732.0.B-1 AOS Space Data Link Protocol**
- **701.0.B-3 Network Data Link Architecture Spec, VCDU?**

NASA's
Open Source Scheduler (SCH) App

https://github.com/nasa/SCH

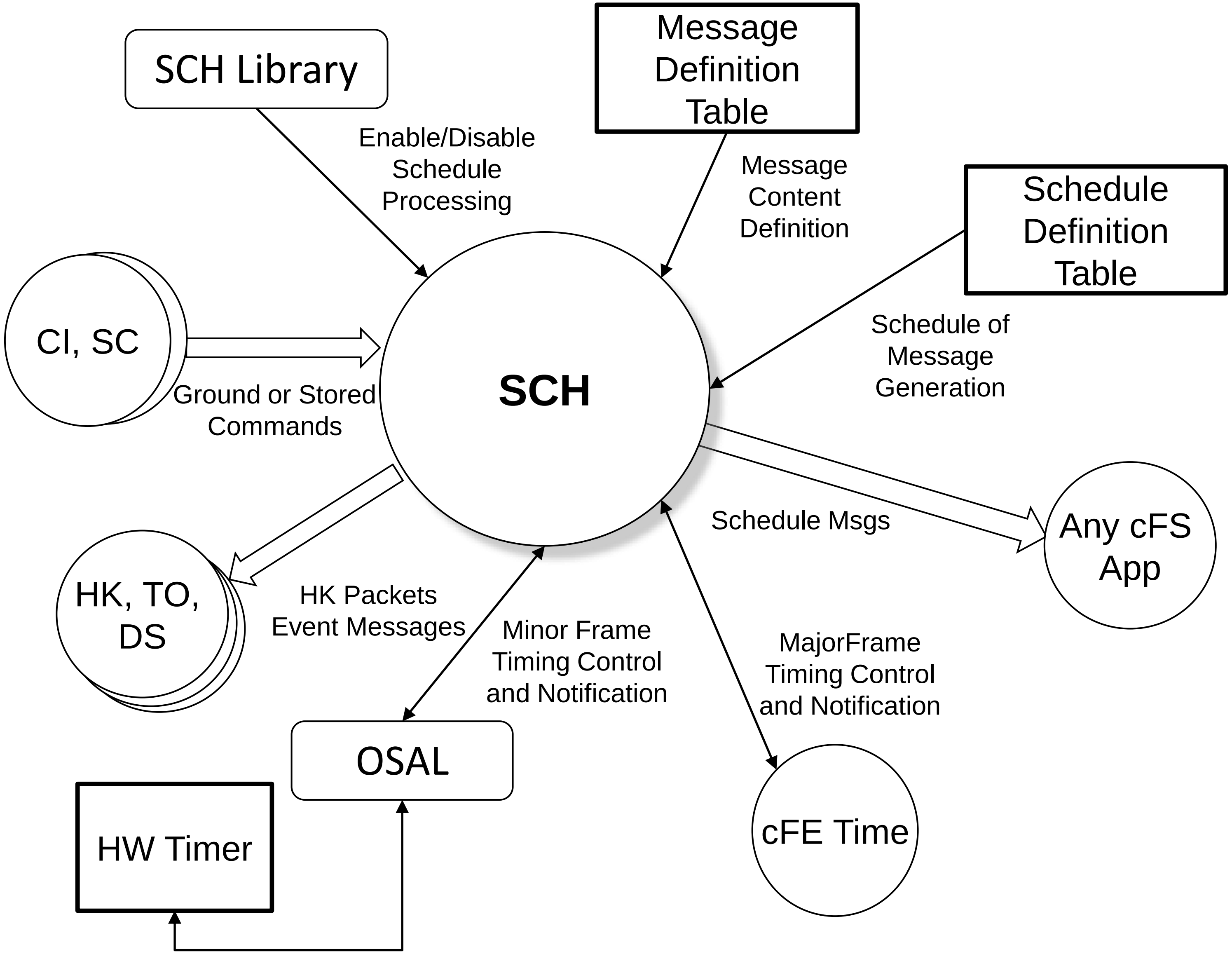
SCH Overview

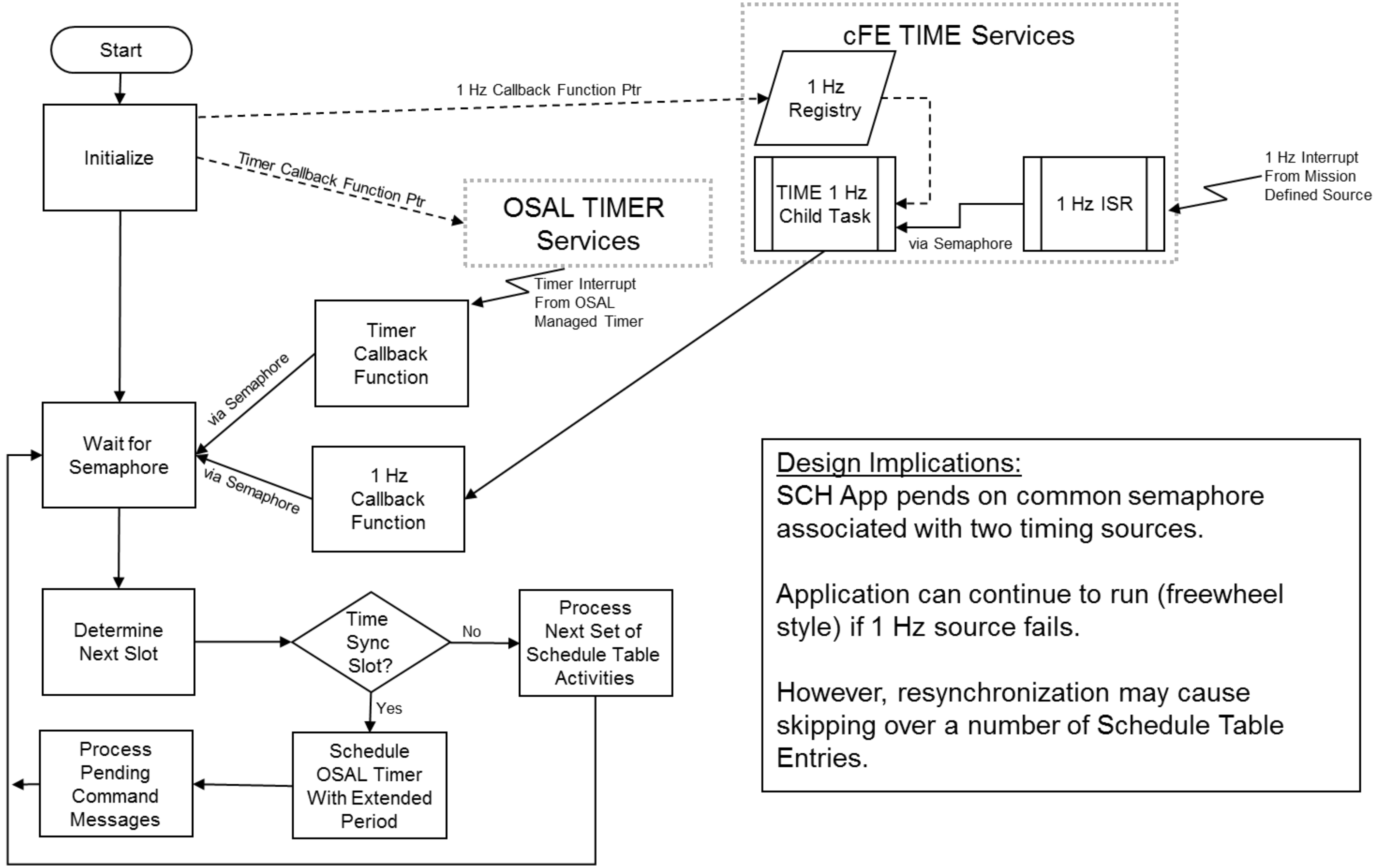
- **Provides method of generating messages at pre-determined timing intervals**
  - Operates in a Time Division Multiplexed (TDM) fashion with deterministic behavior
    - Synchronized to external Major Frame cFE TIME 1 Hz signal
    - Each Major Frame split into a platform configuration number of smaller slots (typically 100 slots of 10 milliseconds each)
      - Each slot can contain a platform configuration number of software bus messages that can be issued within that slot

Same table structure but binary

Groups are good for redundancy management

SCH Context Diagram





<u>Design Implications:</u>

SCH App pends on common semaphore associated with two timing sources.

Application can continue to run (freewheel style) if 1 Hz source fails.

However, resynchronization may cause skipping over a number of Schedule Table Entries.

SCH Configuration Parameters (1 of 2)

Parameter	Description	Default Value
SCH_PIPE_DEPTH	Software bus command pipe depth	12
SCH_TOTAL_SLOTS	Minor Frame Frequency (in Hz)	100
SCH_ENTRIES_PER_SLOT	Maximum Activities per slot	5
SCH_MAX_MESSAGES	Maximum Number of Message Definitions in Message Table	128
SCH_MDT_MIN_MSG_ID	Minimum Message ID allowed in Message Definition Table	0
SCH_MDT_MAX_MSG_ID	Maximum Message ID allowed in Message Definition Table	CFE_SB_HIGHEST_VALID_MSGID
SCH_MAX_MSG_WORDS	Maximum Length, in Words, of a Message in the message table	64
SCH_MAX_LAG_COUNT	Maximum Number of slots allowed for catch-up before skipping	(SCH_TOTAL_SLOTS/2)
SCH_MAX_SLOTS_PER_WAKEUP	Maximum Number of Slots to be processed when in "Catch Up" mode	5
SCH_MICROS_PER_MAJOR_FRAME	Conversion factor for how many microseconds in a wake-up period	10000000

SCH Configuration Parameters (2 of 2)

Parameter	Description	Default Value
SCH_SYNC_SLOT_DRIFT_WINDOW	Additional time allowed in Sync Slot to wait for Major Frame Sync	5000
SCH_STARTUP_SYNC_TIMEOUT	Timeout on waiting for all applications to start at initialization	50000
SCH_STARTUP_PERIOD	Number of microseconds to attempt major frame synchronization	(5*SCH_MICROS_PER_MAJOR_FRAME)
SCH_MAX_NOISY_MAJORF	Maximum noisy major frames prior to desynchronization	2
SCH_LIB_PRESENCE	Presence of SCH Library	1
SCH_LIB_DIS_CTR	Processing disabled counter at startup	0
SCH_SCHEDULE_FILENAME	Default schedule table filename to load at startup	“/cf/apps/sch _def_schtbl.tbl”
SCH_MESSAGE_FILENAME	Default message table filename to load at startup	“/cf/apps/sch _def_msgtbl.tbl"
SCH_MISSION_REV	Mission revision number	0

SCH Commands

Command	Description
No-op	Increments the Command Accepted Counter and sends a debug event message
Reset Counters	Initializes housekeeping counters to zero
Enable Entry	Enables an entry in the Schedule Definition Table
Disable Entry	Disables an entry in the Schedule Definition Table
Enable Group and/or Multi-Group(s)	Enables a group and/or multi-group(s) of entries in the Schedule Definition Table
Disable Group and/or Multi-Group(s)	Disables a group and/or multi-group(s) of entries in the Schedule Definition Table
Enable Sync	Enables usage of Major Frame Signal if previously autonomously disabled for being "noisy"
Send Diagnostic Tim	Generates and sends the SCH Diagnostic Telemetry Packet that contains the current state of all activities defined in the Schedule Definition Table

SCH Housekeeping Telemetry Message (1 of 2)

Telemetry Point	Description
CommandCounter	Number of accepted ground commands
CommandErrCounter	Number of rejected ground commands
ScheduleActivitySuccessCounter	Number of scheduled activities processed
ScheduleActivityFailureCounter	Number of scheduled activities failed due to error
SlotsProcessedCounter	Number of schedule slots processed
SlotsSkippedCounter	Number of instances when one or more slots were skipped
MultipleSlotsCounter	Number of instances when two or more slots were processed at once
SameSlotCounter	Number of instances when SCH woke up in the same time slot as previously
BadTableDataCount	Number of table entries with an error that have been encountered
TableVerifySuccessCount	Number of successful table verifications performed
TableVerifyFailureCount	Number of failed table verifications performed
TablePassCounter	Number of times Schedule Table was completely processed
ValidMajorFrameCount	Number of Valid Major Frame Signals received
MissedMajorFrameCount	Number of Major Frame Signals that did not occur when expected
UnexpectedMajorFrameCount	Number of Major Frame Signals that occurred when nor expected to occur
MinorFramesSinceTone	Number of Minor Frames processed since last Major Frame

SCH Housekeeping Telemetry Message (2 of 2)

Telemetry Point	Description
NextSlotNumber	The next slot to be processed in the Schedule Definition Table
LastSyncMETSlot	Slot Number when last Time Synchronization occurred
IgnoreMajorFrame	Major Frame Signals are ignored because they are deemed "noisy"
UnexpectedMajorFrame	Last Major Frame Signal occurred when not expected
SyncToMET	Minor Frames are synchronized to MET.
MajorFrameSource	Identifies the source of the Major Frame Signal (timer, MET, etc)