Core Flight System (cFS) Training

Community Apps:

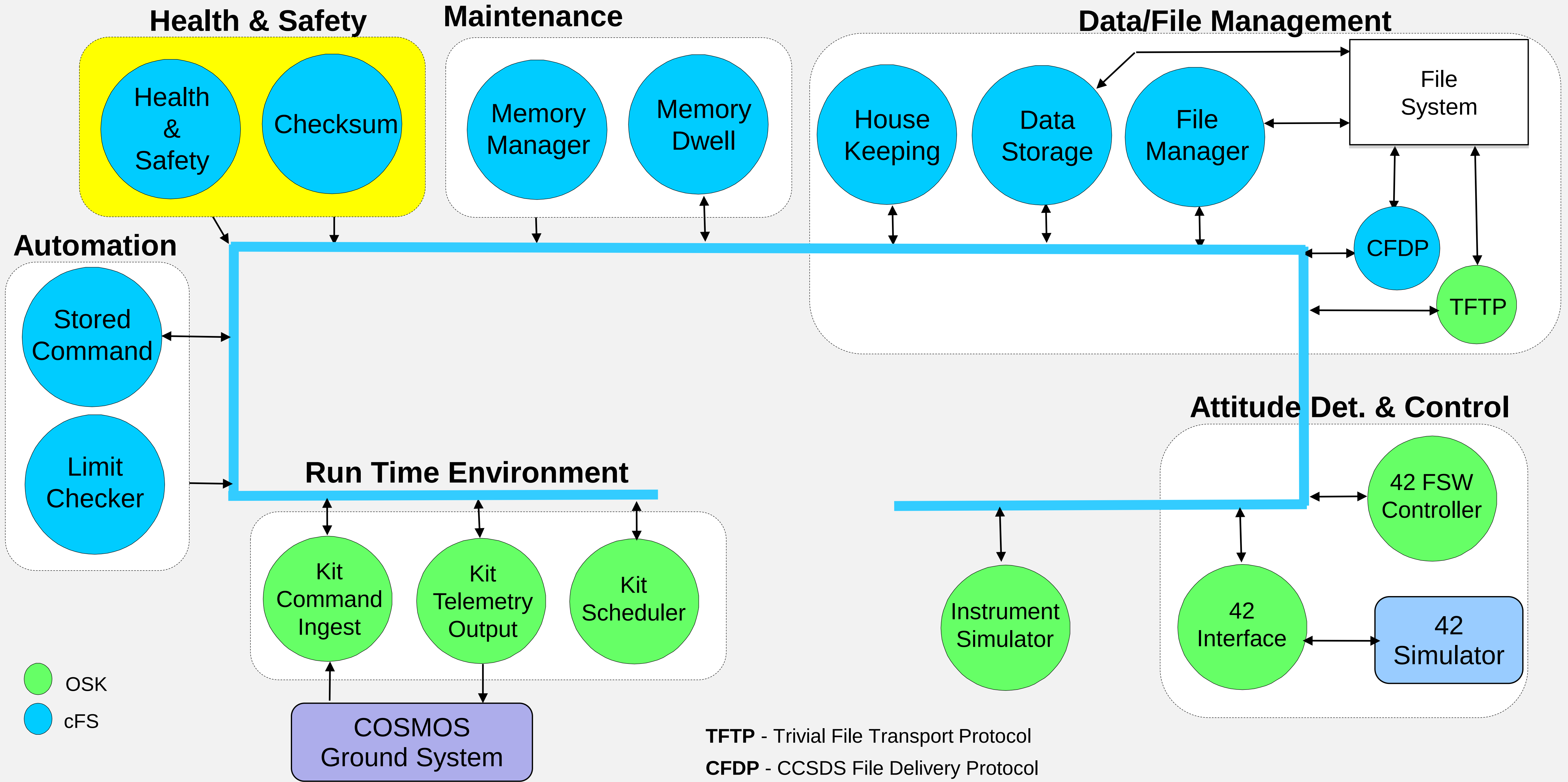
Health & Safety

Health & Safety App Group Agenda

- **TODO**

Health & Safety Application Group Overview

OSK FSW SimSat Applications



TFTP - Trivial File Transport Protocol

CFDP - CCSDS File Delivery Protocol

Health & Safety App Group Objectives

- **System Integrity**
  - “The quality that a **system** has when it performs its intended function in an unimpaired manner, free from unauthorized manipulation of the **system**, whether intentional or accidental.”**
- **Limit Checker & Stored Command part of**
- **This app group plays one a may roles in the activity of sustaining engineering**
  - **Integrity & Diagnostics, Implementation of new functionality**

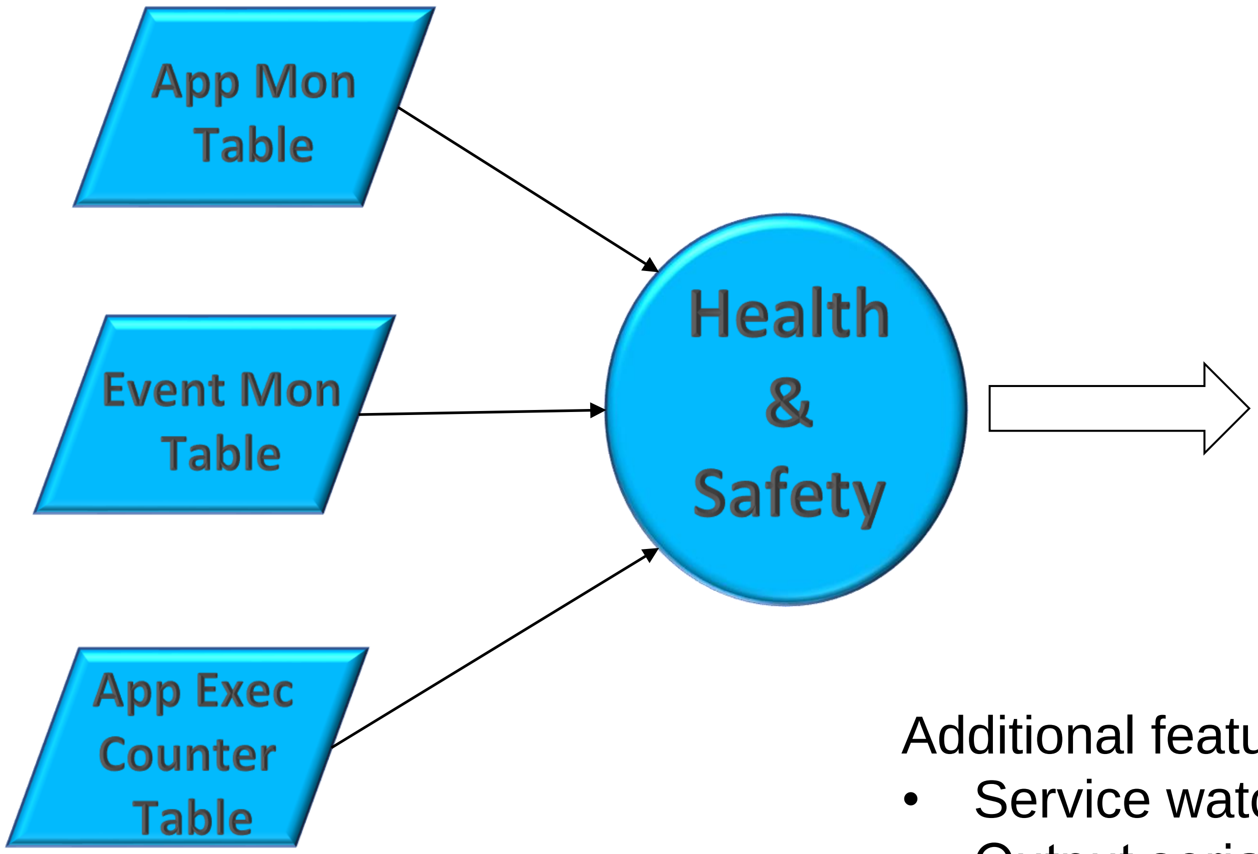
** National Information Technology Laboratory (NIST) Computer Security Resource Center, https://csrc.nist.gov/glossary/term/system_integrity

System Participant Overview

- **Identify cFE services and apps that participate in the bulk of Sustaining activities as identified in the scenarios & use cases**
- **Show chart with HS & CS features and which ones will be exercised in the demo. Show simplified context for HS & CS. Challenge will be to keep simple but allow viewer to understand what's going on.**
- HS used for FSW app H&S and limit checker. Use ISIM fault example

Health & Safety App Overview

Provides services that monitor the health of the FSW system



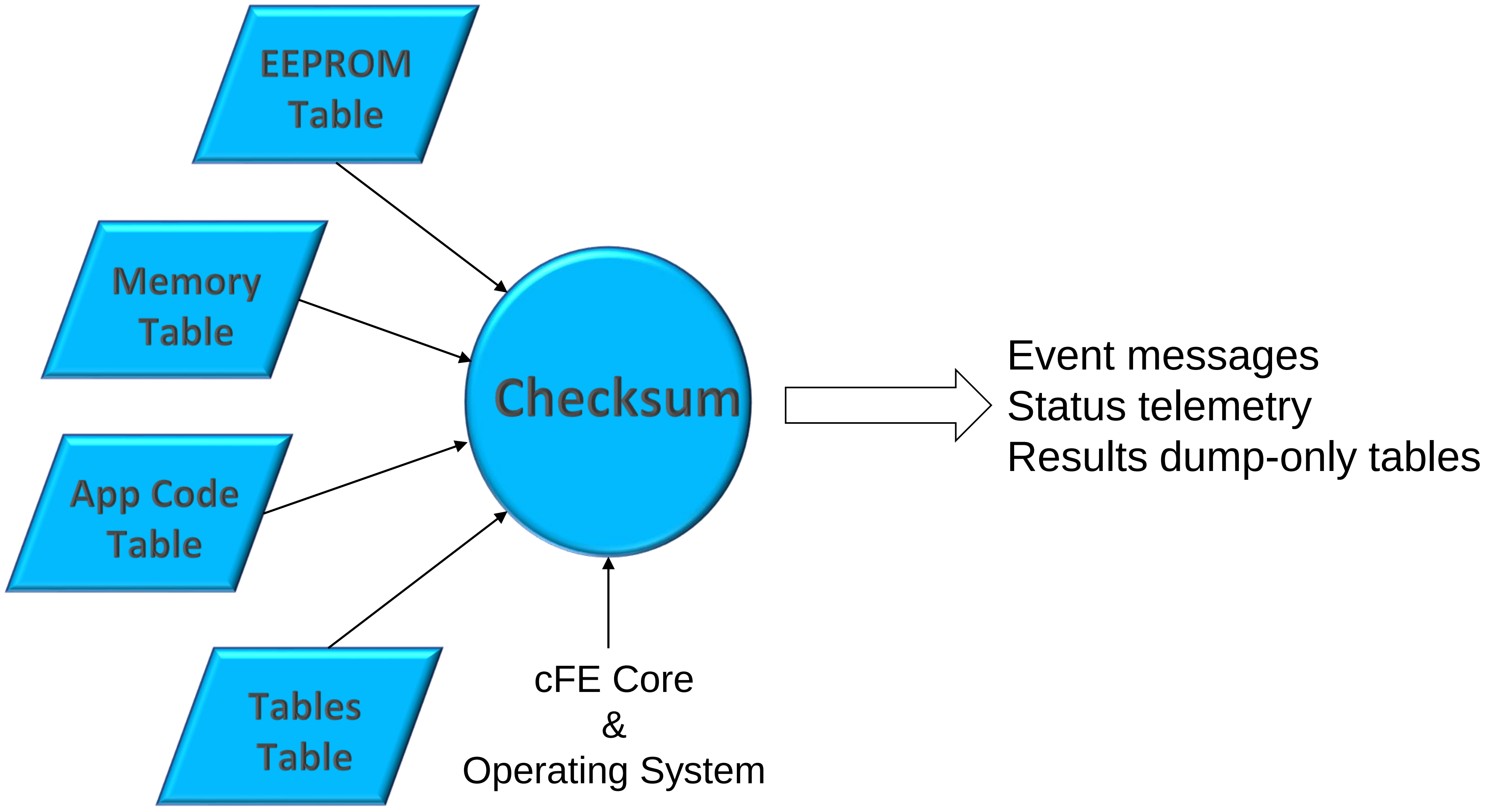
Event message
Restart an app
Reset processor
Table-defined SB message

Additional features:

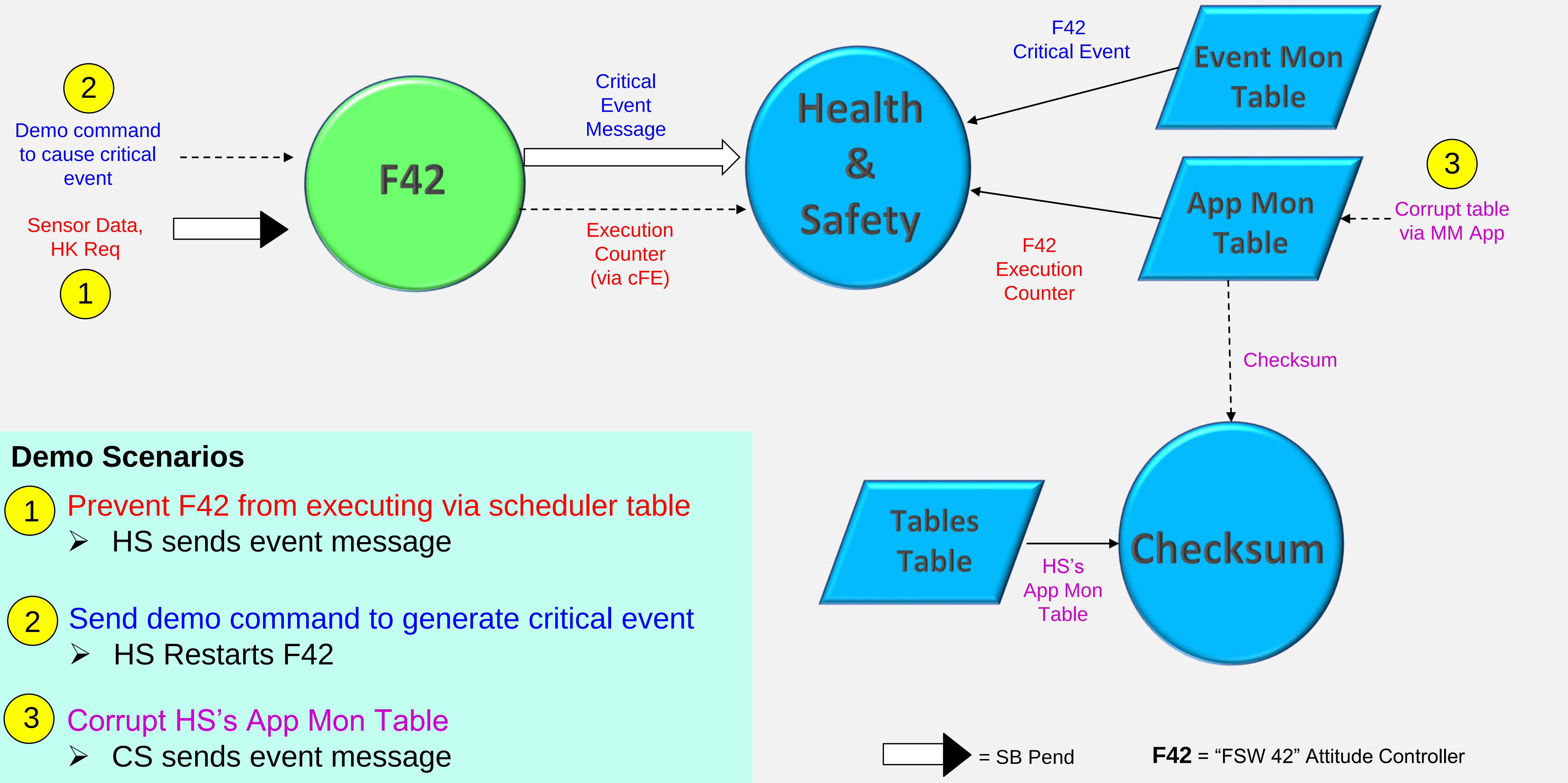
- Service watchdog
- Output serial port ‘aliveness’
- Hooks for custom CPU utilization function

Checksum App Overview

Monitor memory integrity by verifying checksums for table-defined regions



H&S Demo Context



F42 = "FSW 42" Attitude Controller

Demo Scenarios

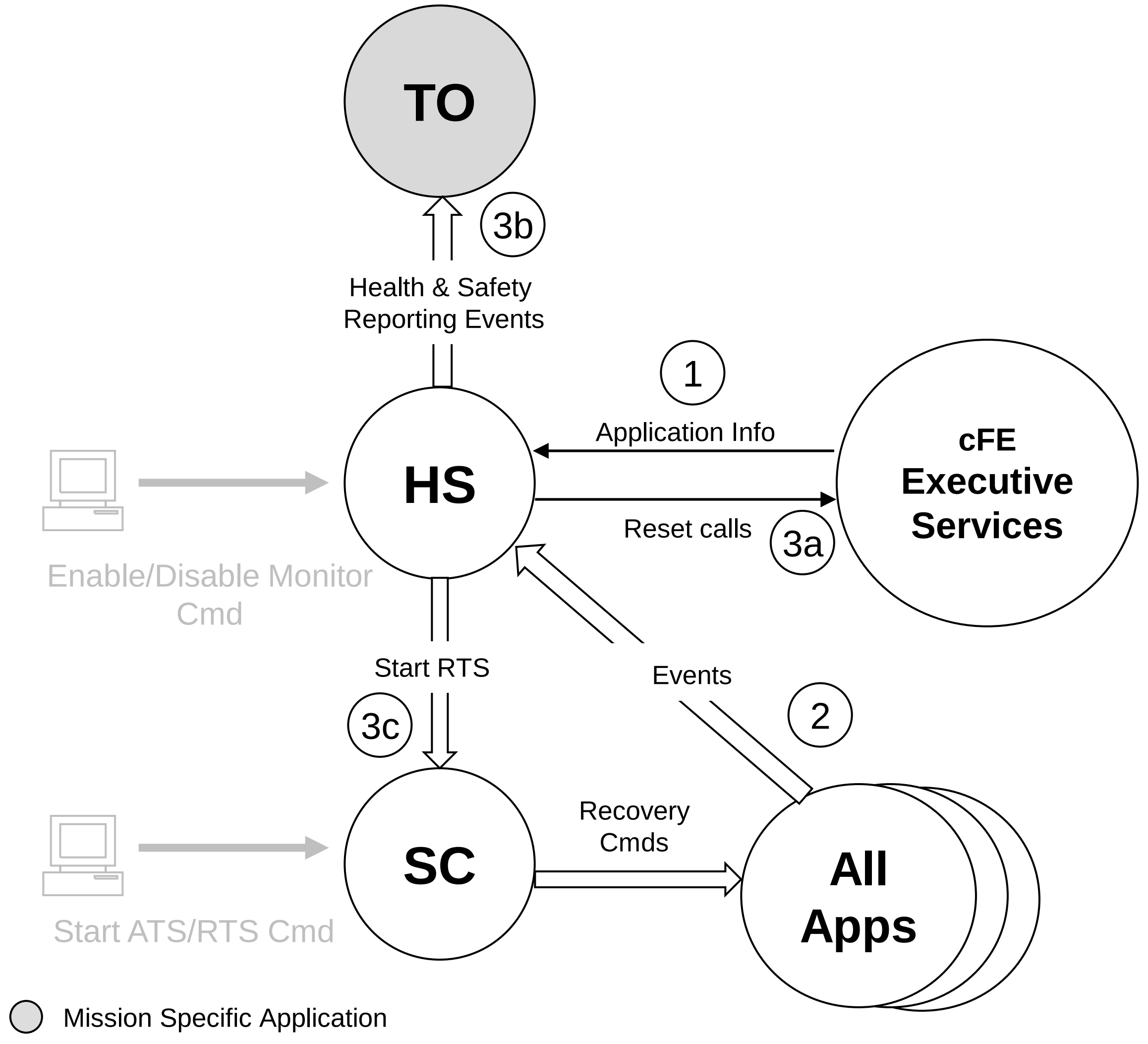
1. Prevent F42 from executing via scheduler table
   - HS sends event message
2. Send demo command to generate critical event
   - HS Restarts F42
3. Corrupt HS's App Mon Table
   - CS sends event message

Health & Safety App Overview

- EEPROM & App Table not supported in Linux environment

Scenarios

1) **HS monitors applications**
2) **HS monitors event messages**
3) **HS Table specified actions are taken in response to application and event monitoring:**
   a) **Reset applications or the processor**
   b) **Send Event message**
   c) **Initiate Stored Command (SC) recovery sequence**



Not pictured: HS manages watchdog, reports CPU utilization & detects hogging, and outputs aliveness heartbeat to UART.

Health & Safety

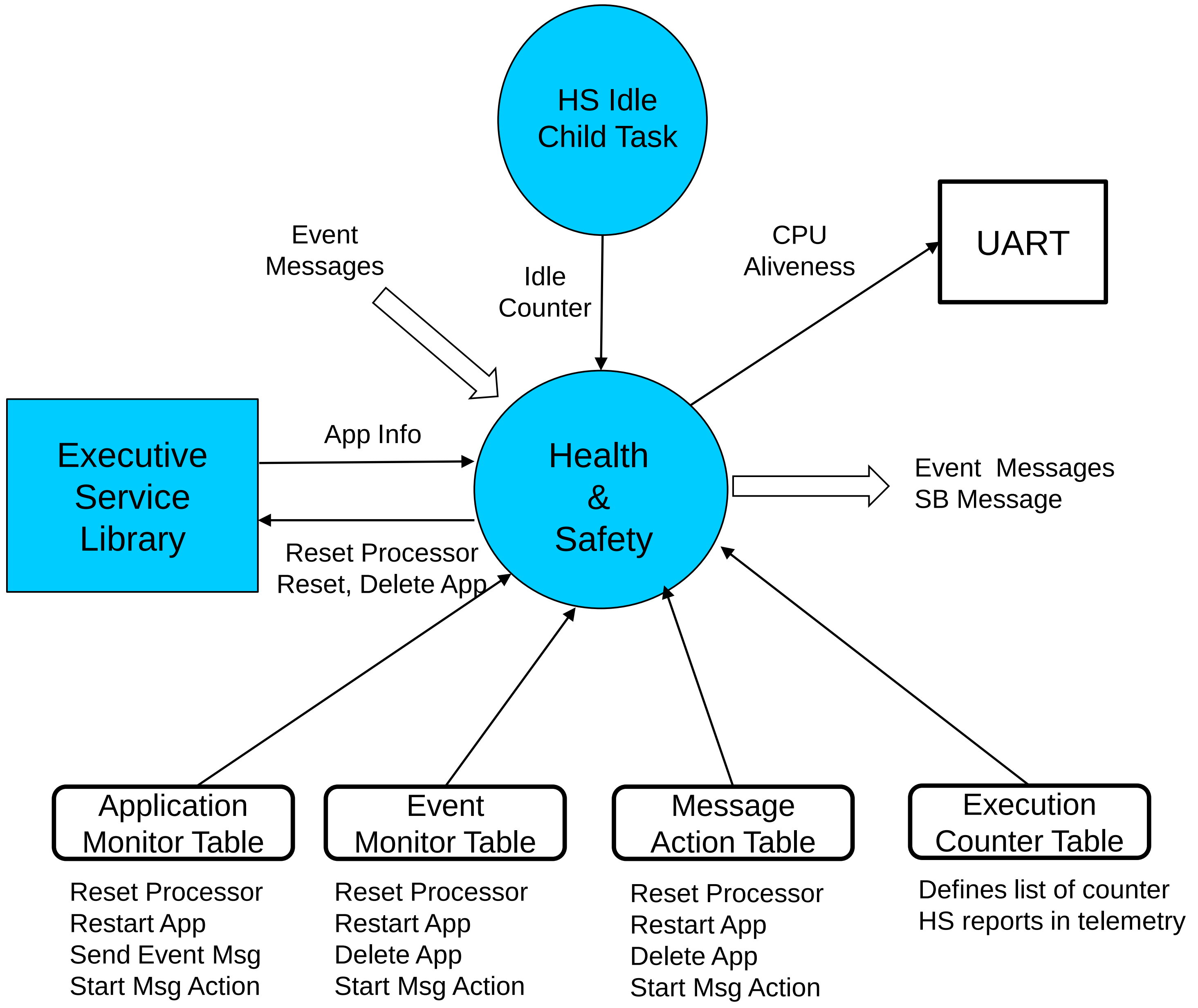
https://github.com/nasa/HS

Health & Safety Objectives

Overview

- **Performs Application Monitoring**
  - Detects when critical applications are not running and take a table-defined action
- **Performs Event Monitoring**
  - Detects critical events and take a table defined action
- **Manages Watchdog**
  - Initializes and service the watchdog.
  - Withholds servicing of the watchdog if certain conditions are not met.
- **Manages CPU**
  - Reports CPU Utilization
  - Detects CPU Hogging and take appropriate action
  - Provides CPU Aliveness Indication
- **Reports Table-Defined Execution Counters**
  - Can include Application Main Tasks, Child Tasks, ISRs, and Device Drivers

Context


HS Idle Child Task
Idle Counter
Event Messages
CPU Aliveness
UART
Executive Service Library
App Info
Reset Processor Reset, Delete App
Health & Safety
Event Messages
SB Message
Application Monitor Table
Event Monitor Table
Message Action Table
Execution Counter Table
Reset Processor
Restart App
Send Event Msg
Start Msg Action
Reset Processor
Restart App
Delete App
Start Msg Action
Reset Processor
Restart App
Delete App
Start Msg Action
Defines list of counter HS reports in telemetry

Application Monitoring

- **Monitors the health of table specified applications**
  - Both cFE core applications and any cFS application
- **How are they monitored?**
  - Use the counters maintained by ES in the CFE_ES_RunLoop function.
    - Applications must call CFE_ES_RunLoop to increment the execution counter and let the system know they are active.
- **What are the response options for an application not running?**
  1. Perform Processor reset
     - Sets Service_watchdog flag to FALSE
  2. Restart Application
  3. Send an Event message
  4. Send Software Bus Message

- **What happens if an app goes away or is restarted?**
  - There should be sufficient time to restart an app before it's flagged as missing
  - Application monitoring can be disabled during application updates/maintenance
  - Application monitoring table can be reloaded if an application is permanently deleted

Event Monitoring

- **Subscribe to all event messages**
  - Monitored events are table specified
- **HS can take one of the following actions on events:**
  - Processor Reset
  - Reset Application
  - Delete Application
  - Send Software Bus Message
- **Event monitoring can be turned on or off by command**

Watchdog Management

- **Watchdog must be initialized at startup**
  - BSP will program watchdog to a reasonable value to allow system to start
- **Watchdog will be serviced as long as “Service_watchdog” flag is TRUE**
- **If HS is not running, the watchdog will expire causing a CPU reset.**
- **The “Service_watchdog” flag is set to FALSE if**
  - There is CPU hogging for more than <TBD, Configuration parameter > seconds
  - There is a Critical Application Monitoring failure
- **One of the above conditions should be enough to restart the system before the watchdog expires**
  - Why bother with the watchdog then?
    - If the software gets “stuck” then the watchdog will make sure it is reset.
- **OS API will supply get timeout, set timeout, and service functions separately**
  - HS will set the timeout to the default value when initializing, and service every cycle

CPU Management

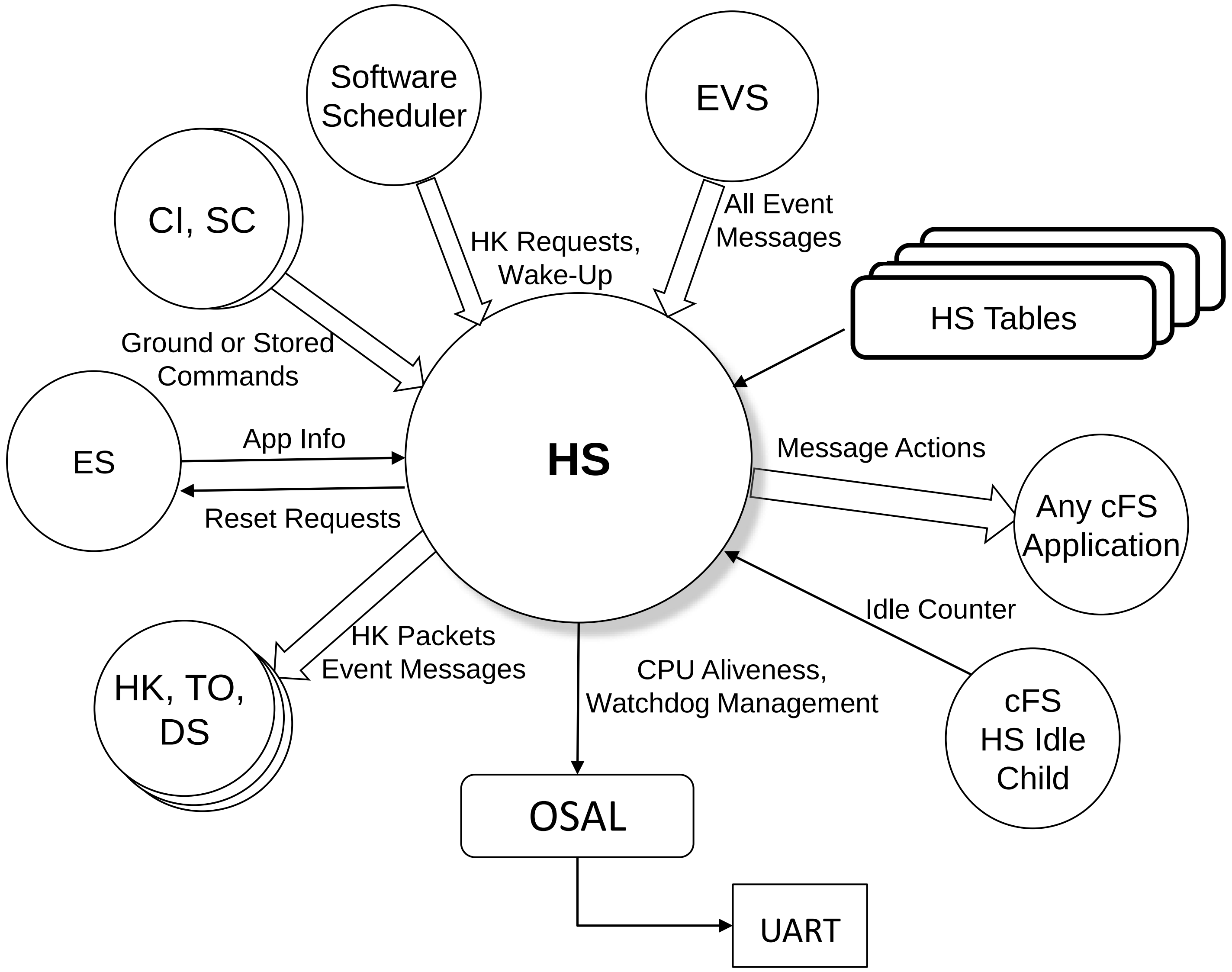
- **HS will perform the following CPU related functions:**
  - Reports CPU utilization information
    - The CPU information comes from OS/BSP
      - May have different implementations on different platforms
    - Collects and report average CPU utilization over <TBD, Configuration Parameter> time
    - Collects and report peak CPU utilization over <TBD, Configuration Parameter> time
  - Provides CPU hogging indication
    - If the CPU is at 100% start the “hogging” counter
    - If hogging counter reaches <TBD, Configuration parameter > limit
      - Send event / make syslog entry
      - Set “Service_watchdog flag” to FALSE
      - Processor Reset
  - Provides CPU aliveness indication (output characters to UART)
    - Enable/disable Command to output periodic heartbeat to UART
    - Output characters are <TBD, Configuration parameter >

Execution Counter Reporting

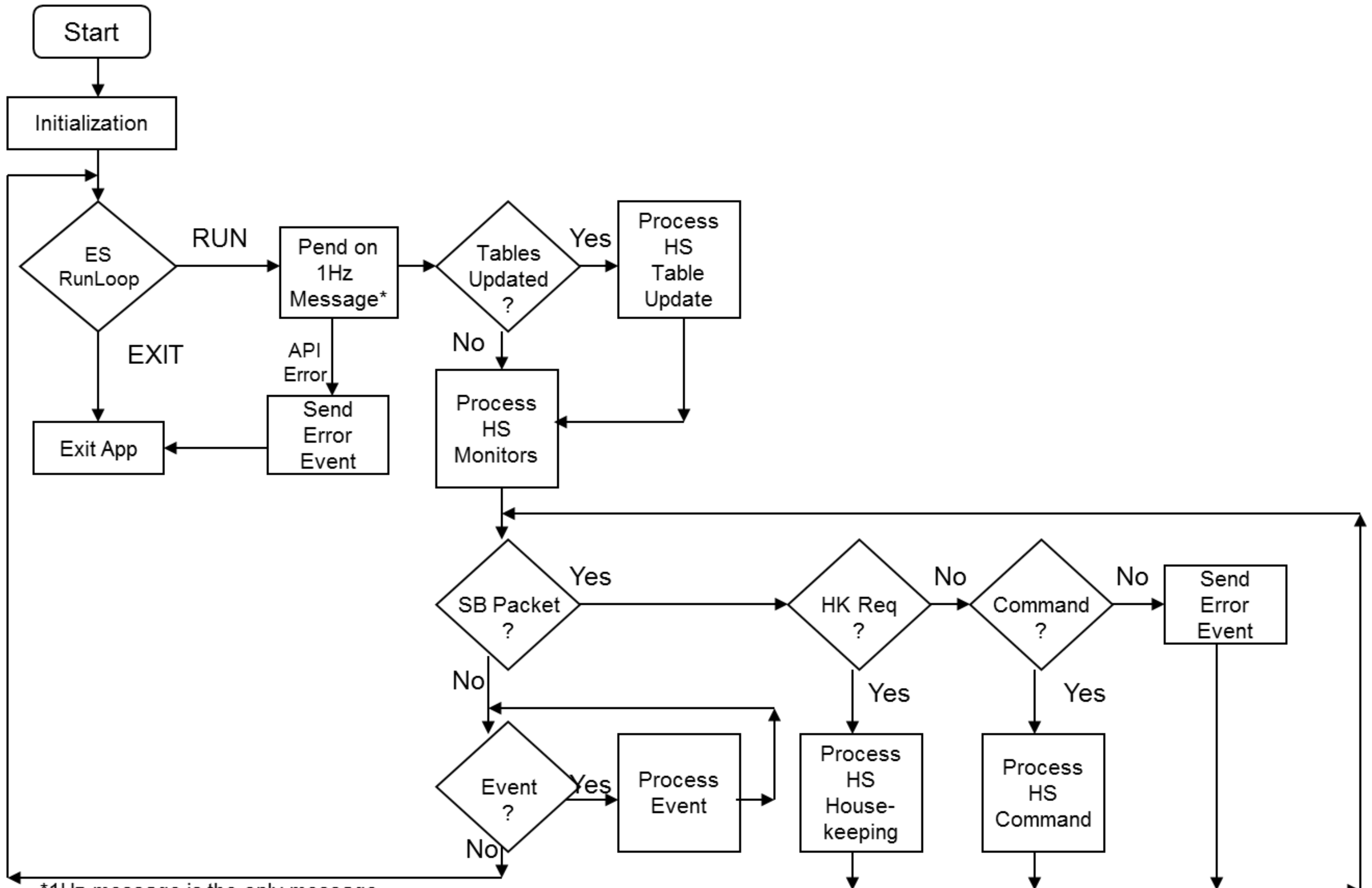
- **HS has a configurable number of execution counter entries in it's housekeeping telemetry message**
  - A Table specifies the counters that will be reported in in the housekeeping telemetry message.
  - On every housekeeping request, HS copies the requested execution counters into the packet

- **Where does HS get the execution counter status?**
  - ES Maintains execution counters for:
    - cFE Core Apps
    - cFS Apps
    - Child Tasks
    - Device Drivers/ISRs
  - App Main Counters are incremented by calling CFE_ES_RunLoop
    - The counters are different from heritage counters that incremented before and after the software bus call.
  - Child task counters are incremented by using an ES counter API
  - Device Driver and ISR counters can be incremented with an ES counter API
  - Counters are fetched by calling ES GetAppInfo and GetTaskInfo API functions

Context


Software Scheduler
EVS
CI, SC
HK Requests, Wake-Up
All Event Messages
HS Tables
Ground or Stored Commands
ES
App Info
Reset Requests
HS
Message Actions
Any cFS Application
Idle Counter
HK Packets Event Messages
HK, TO, DS
CPU Aliveness, Watchdog Management
cFS HS Idle Child
OSAL
UART

App Flow Control



*1Hz message is the only message that shows up on that pipe, and is a pend with timeout.

Configuration Parameters - 1

Parameter	Description	Default Value
HS_MAX_EXEC_CNT_SLOTS	Maximum Number of Execution Counters to be Reported	32
HS_MAX_MSG_ACT_TYPES	Maximum Number of Message Action types	8
HS_MAX_MSG_ACT_SIZE	Maximum Size of Message Action Message	16
HS_MAX_CRITICAL_APPS	Maximum Number of Critical Applications to Monitor	32
HS_MAX_CRITICAL_EVENTS	Maximum Number of Critical Events to Monitor	16
HS_WATCHDOG_TIMEOUT_VALUE	Default Watchdog timeout value in milliseconds to be set when initializing	10000
HS_CPU_ALIVE_STRING	String to output on UART	”·”
HS_CPU_ALIVE_PERIOD	How often to output CPU aliveness indicator	5
HS_MAX_RESTART_ACTIONS	How many times a Processor Reset can be performed by a monitor failure	3
HS_CMD_PIPE_DEPTH	Software bus command pipe depth	12
HS_IDLE_TASK_PRIORITY	Priority of the Idle Task being used for CPU Utilization Monitoring	252

HS Configuration Parameters - 2

Parameter	Description	Default Value
HS_UTIL_CALLS_PER_MARK	Number of (1 Hz) calls between capturing the Idle Task Count	1
HS_UTIL_CYCLES_PER_INTERVAL	Number of HS cycles between calculating CPU Utilization	1
HS_UTIL_PER_INTERVAL_TOTAL	Number that signifies full utilization during one period	10000
HS_UTIL_PER_INTERVAL_HOGGING	Number that signifies CPU is being hogged in terms of full utilization	9900
HS_UTIL_CONV_MULT1 **HS_UTIL_CONV_DIV** **HS_UTIL_CONV_MULT2**	Utilization = Full Utilization —(((Idle Task Cycles * MULT1) / DIV) * MULT2)	Determined by Calibration
HS_UTIL_HOGGING_TIMEOUT	Number of Intervals for which hogging threshold must be exceeded to result in hogging event message	5
HS_UTIL_PEAK_NUM_INTERVAL	Number of intervals over which to report the peak value	64
HS_UTIL_AVERAGE_NUM_INTERVAL	Number of intervals over which to report the average value	4
HS_UTIL_DIAG_MASK	Used for calibration (how frequently to record time)	0xFFFFFFFF
HS_UTIL_DIAG_ARRAY_POWER	Used for calibration (how many time recordings are stored)	4

HS Commands - 1

Command	Description
Noop	Increment commands accepted counter and send event message
Reset Counters	Reset housekeeping telemetry counters
Disable Critical Application Monitor	Disables the monitoring and actions related to the critical application monitor.
Enable Critical Application Monitor	Enables and reinitializes the monitoring and actions related to the critical application monitor.
Disable Critical Event Monitor	Disables the critical event monitor function in HS
Enable Critical Event Monitor	Enables the critical event monitor function in HS
Disable CPU Aliveness Indicator	Stops the periodic output of characters to the UART.
Enable CPU Aliveness Indicator	Starts the periodic output of characters to the UART.
Set Max Processor Resets	Sets the max number of processor resets HS can perform to provided parameter value
Reset Processor Resets Counter	Resets the current count of HS performed Processor Resets.
Disable CPU Hogging Indicator	Stops the Hogging event from being sent
Enable CPU Hogging Indicator	Allows the Hogging event to be sent

HS Commands - 2

Command	Description
Report Utilization Diagnostics	Reports the current Utilization Diagnostics information in an event message
Set Utilization Parameters	Sets the calibration parameters used for Utilization Monitoring to the specified parameters
Set Utilization Diagnostics Mask	Sets the mask value being used for collecting Utilization Diagnostics information to a specified parameter

HS Housekeeping Telemetry Message

Telemetry Point	Description
HS CMDPC	Count of valid commands received
HS CMDEC	Count of invalid commands received
HS APPMONSTATE	Status of Critical Application Monitor (enabled, disabled)
HS EVTMONSTATE	Status of Event Monitor (enabled, disabled)
HS CPUALIVESTATE	Status of Aliveness Indicator output (enabled, disabled)
HS CPUHOGSTATE	State of CPU Hogging Indicator output (enable, disabled)
HS STATUSFLAGS	Status flags for table loaded and CDS available states
HS PRRESETCNT	Number of resets HS has performed so far
HS MAXRESETCNT	Max number of resets HS is allowed to perform
HS EVTMONCNT	Number of events monitored by the Event Monitor
HS INVALIDEVTAPPCNT	Number of entries in Event Monitor Table that have unresolvable task names
HS APPMONENABLE[TBD]	Application Monitor enable status by table entry
HS MSGACTCTR	Number of message actions sent
HS CPUUTILAVG	Average CPU Utilization
HS CPUUTILPEAK	Peak CPU Utilization
HS EXECOUNT[TBD]	Execution Counter Array

Checksum

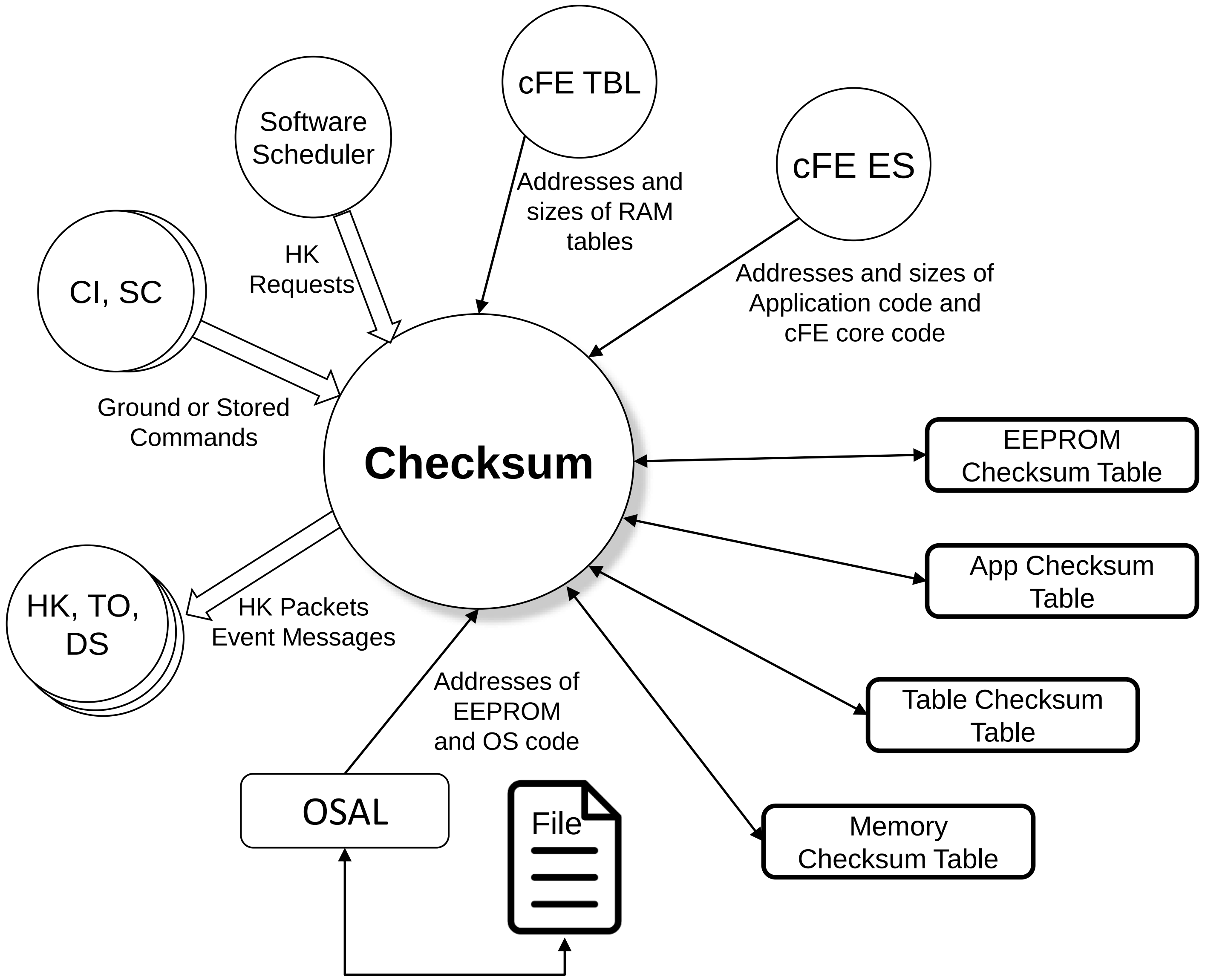
https://github.com/nasa/CS

Checksum Objectives

CS Overview

- **Monitors the static code/data specified by the users and the OS and cFE code segments.**
- **Uses four different user defined tables**
  - Table of Apps to be checkummed
  - Table of Tables to be checksummed
  - Table of EEPROM to be checksummed
  - Table of other memory areas (“Memory”) to be checksummed
- **Reports all checksum miscompares as errors.**
- **Scheduled to wakeup on a 1Hz schedule**
- **Byte-limited per cycle to prevent CPU hogging**

CS Context


cFE TBL
Software Scheduler
cFE ES
Addresses and sizes of RAM tables
HK Requests
CI, SC
Addresses and sizes of Application code and cFE core code
Ground or Stored Commands
Checksum
EEPROM Checksum Table
App Checksum Table
HK, TO, DS
HK Packets Event Messages
Addresses of EEPROM and OS code
Table Checksum Table
Memory Checksum Table
OSAL
File

CS Checksum Activities

- **Background Cycle**
  - On wake-up from a “background cycle” command, CS continues checksum calculations through the four definitions tables, OS code segment, and cFE core.
- **Recompute**
  - On receipt of a “recompute” command, CS spawns a child task to compute a new baseline checksum for the selected area
  - Only one child task may be active at a time
    - Includes One-Shot
- **One Shot**
  - On receipt of a “one-shot” command, CS spawns a child task to compute a checksum on the specified memory area.
  - Only one child task may be active at a time
    - Includes Recomputes

CS Checksum Algorithm

- **The term ‘checksum’ is historical and does not actually mean we are using a checksumming algorithm, as it has been proven they are not safe enough.**

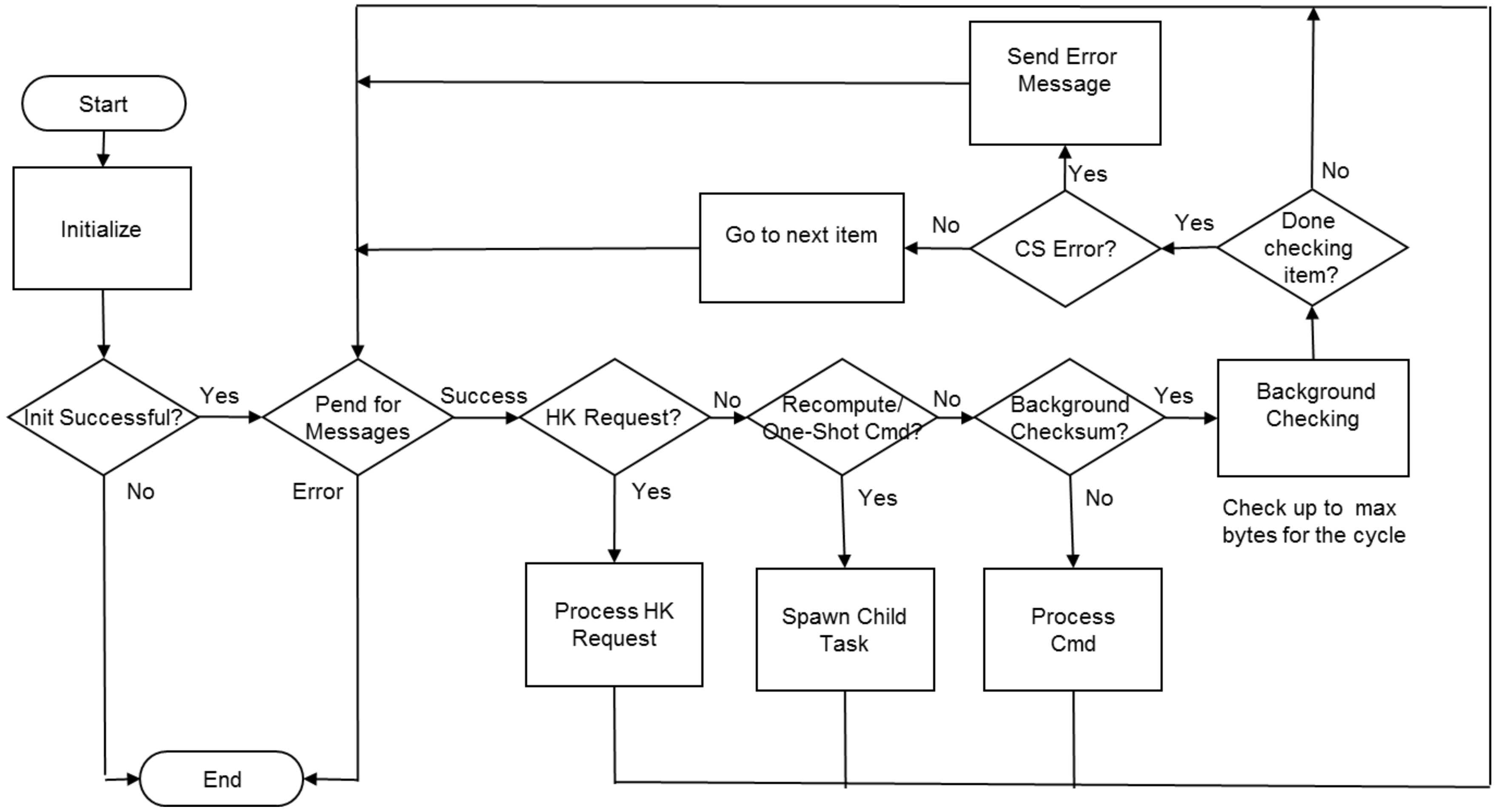
- **The algorithm that CS will use will be a Cyclical Redundancy Check (CRC) algorithm. It will be handled by a cFE ES function which specifies 8, 16, or 32 bit polynomial.**

- **By default, CS will use the cFE default CRC algorithm, but can be changed via a configuration parameter**

CS Checksum Regions

- **EEPROM**
  - Everything in EEPROM (file system, OS, bootstrap, etc)
    - Split up by user-defined regions
- **RAM**
  - OS code segment
  - cFE core code segment
  - Application code segments
  - Tables
  - User defined memory segments

CS App Control Flow



CS Telemetering Checksums

- **CS maintains a dump-only checksum working table for each checksum region defined by table in CS**
  - Updates checksum results for each checksum region on each checksum cycle
  - Users can obtain current checksum results by performing a table dump via a Table Services command

CS Configuration Parameters

Parameter	Description	Default Value
Default EEPROM Table Name	--	/cf/apps/cs_eepromtbl.tbl
Default Memory Table Name	--	/cf/apps/cs_memorytbl.tbl
Default Tables Table Name	--	/cf/apps/cs_tablestbl.tbl
Default Apps Table Name	--	/cf/apps/cs_apptbl.tbl
Pipe Depth	Command pipe depth	12
Max # of EEPROM Entries	Maximum number of entries in the table to checksum	16
Max # of Memory Entries	Maximum number of entries in the table to checksum	16
Max # of Tables Entries	Maximum number of entries in the table to checksum	24
Max # of Apps Entries	Maximum number of entries in the table to checksum	24
Default Bytes per Cycle	# of bytes checksummed in a single cycle	16384 (16KB)
Child Task Priority	1 is highest priority. Child cannot be higher than CS.	200
Child Task Delay	Delay to prevent CPU hogging.	1000 ms
Startup Timeout	Time for CS to wait for other apps to start	60000 ms
Mission Revision	Mission-level revision number	0

CS Commands

Command	Description
No-op	Increments the Command Accepted Counter and sends an info event message
Reset Counters	Initializes housekeeping counters to zero
Disable Checksumming	Stop CS background checking
Enable Checksumming	Restart background checking
OneShot Checksum	Start at given address, compute checksum over size
Cancel Oneshot checksum	If a one shot CS is in progress, stop it
Report Baseline of cFE Core	Reports the baseline of the cFE Core code segment
Recompute Baseline of cFE Core	Recomputes the baseline of the cFE Core code segment
Report Baseline of OS	Reports the baseline of the OS code segment
Recompute Baseline of OS	Recomputes the baseline of the OS code segment
Disable Checksumming for cFE Core	Stop background checking cFE Core code segment
Enable Checksumming for cFE Core	Restart background checking cFE Core code segment
Disable Checksumming for OS	Stop background checking OS code segment
Enable Checksumming for OS	Restart background checking OS code segment

CS EEPROM Commands

Command	Description
Get Region ID for EEPROM Address	Retrieves EEPROM table entry ID for region that covers given address
Recompute baseline for EEPROM Region	Recompute the baseline checksum for the given EEPROM region ID
Report Baseline for EEPROM Region	Sends event message with baseline checksum for given EEPROM region ID
Disable Checksumming for EEPROM Region	Stop background checking for the given EEPROM region ID
Enable Checksumming for EEPROM Region	Restart background checking for the given EEPROM region ID
Disable Checksumming for EEPROM	Stop background checking entire EEPROM table
Enable Checksumming for EEPROM	Restart background checking entire EEPROM table

CS User Defined Memory Commands

Command	Description
Get Region ID for Memory Address	Retrieves Memory table entry ID for region that covers given address
Recompute baseline for Memory Region	Recompute the baseline checksum for the given Memory region ID
Report Baseline for Memory Region	Sends event message with baseline checksum for given Memory region ID
Disable Checksumming for Memory Region	Stop background checking for the given Memory region ID
Enable Checksumming for Memory Region	Restart background checking for the given Memory region ID
Disable Checksumming for Memory	Stop background checking entire Memory table
Enable Checksumming for Memory	Restart background checking entire Memory table

CS Application Commands

Command	Description
Recompute baseline for Application	Recompute the baseline checksum for the given App name
Report Baseline for Application	Sends event message with baseline checksum for given App name
Disable Checksumming for Application	Stop background checking for the given App name
Enable Checksumming for Application	Restart background checking for the given App name
Disable Checksumming for Apps	Stop background checking entire App table
Enable Checksumming for Apps	Restart background checking entire App table

CS Table Commands

Command	Description
Recompute baseline for Table	Recompute the baseline checksum for the given Table name
Report Baseline for Table	Sends event message with baseline checksum for given Table name
Disable Checksumming for Table	Stop background checking for the given Table name
Enable Checksumming for Table	Restart background checking for the given Table name
Disable Checksumming for Tables	Stop background checking entire Table table
Enable Checksumming for Tables	Restart background checking entire Table table

CS Housekeeping Telemetry Message - 1

Telemetry	Description
CmdCounter	Number of accepted commands
CmErrCounter	Number of rejected commands
ChecksumState	Enable/Disable status of background checksumming
EepromCSState	Enable/Disable status of EEPROM checksumming
MemoryCSState	Enable/Disable status of user-defined Memory checksumming
AppCSState	Enable/Disable status of Apps checksumming
TablesCSState	Enable/Disable status of Tables checksumming
OSCSState	Enable/Disable status of OS code segment checksumming
CfeCoreCSState	Enable/Disable status of cFE core checksumming
EepromCSErrCounter	Number of checksum errors reported in EEPROM
MemoryCSErrCounter	Number of checksum errors reported in checksummed area of memory
AppsCSErrCounter	Number of checksum errors reported in checksummed apps
TablesCSErrCounter	Number of checksum errors reported in checksummed tables
CfeCoreCSErrCounter	Number of checksum errors reported in cFE core code
OSCSErrCounter	Number of checksum errors reported in OS code

CS Housekeeping Telemetry Message - 2

Telemetry	Description
CurrentCSTable	Current table being checksummed (cFE Core, OS, EEPROM, Memory, Apps, Tables)
CurrentEntryInTable	Current entry ID in the table currently being checksummed
EepromBaseline	Current baseline checksum of entire EEPROM
OSBaseline	Current baseline checksum of OS code segment
CfeCoreBaseline	Current baseline checksum of cFE Core code segment
LastOneShotAddress	Start address used in the last One Shot checksum command
LastOneShotSize	Number of bytes used in the last One Shot checksum command
LastOneShotChecksum	Calculated checksum by the last One Shot checksum command
PassCounter	Number of times CS has passed through all of its tables

CS

CS