**Tuning Selection Management**

**FDD #ES-400A**

[1. High Level Description 4](#_Toc448583732)

[2. Derived Requirements 4](#_Toc448583733)

[3. Sub-Function Data Flow 4](#_Toc448583734)

[4. Design Rationale 5](#_Toc448583735)

[4.1. Design Overview 5](#_Toc448583736)

[4.2. Flash Calibration Table 5](#_Toc448583737)

[4.3. RAM Calibration Table 6](#_Toc448583738)

[4.4. Calibration Components 6](#_Toc448583739)

[4.4.1. Flash Memory Location 6](#_Toc448583740)

[4.4.2. Calibration Usage 7](#_Toc448583741)

[4.4.3. Calibration Indexing 7](#_Toc448583742)

[4.4.4. Online Calibration Grouping 7](#_Toc448583743)

[4.5. Changing Calibration Indexes 7](#_Toc448583744)

[4.6. Copying Calibrations to RAM for Online Calibration 9](#_Toc448583745)

[5. Sub-Functions 11](#_Toc448583746)

[5.1.1. Initialization (TunSelnMngtInit1) 11](#_Toc448583747)

[5.1.1.1. Design Rationale 11](#_Toc448583748)

[5.1.1.2. Inputs 11](#_Toc448583749)

[5.1.1.3. Operation 11](#_Toc448583750)

[5.1.1.4. Outputs 12](#_Toc448583751)

[5.1.2. Periodic (TunSelnMngtPer1) 13](#_Toc448583752)

[5.1.2.1. Design Rationale 13](#_Toc448583753)

[5.1.2.2. Inputs 13](#_Toc448583754)

[5.1.2.3. Operation 13](#_Toc448583755)

[5.1.2.4. Outputs 14](#_Toc448583756)

[5.1.3. Sub-Function: CopyCalPageReq 15](#_Toc448583757)

[5.1.3.1. Design Rationale 15](#_Toc448583758)

[5.1.3.2. Inputs 15](#_Toc448583759)

[5.1.3.3. Operation 15](#_Toc448583760)

[5.1.3.4. Outputs 15](#_Toc448583761)

[5.1.4. Sub-Function: GetCalPageReq 16](#_Toc448583762)

[5.1.4.1. Design Rationale 16](#_Toc448583763)

[5.1.4.2. Inputs 16](#_Toc448583764)

[5.1.4.3. Operation 16](#_Toc448583765)

[5.1.4.4. Outputs 16](#_Toc448583766)

[5.1.5. Sub-Function: GetSegInfoReq 17](#_Toc448583767)

[5.1.5.1. Design Rationale 17](#_Toc448583768)

[5.1.5.2. Inputs 17](#_Toc448583769)

[5.1.5.3. Operation 17](#_Toc448583770)

[5.1.5.3.1. Mode Decision 17](#_Toc448583771)

[5.1.5.3.2. SegModAdrInfo 17](#_Toc448583772)

[5.1.5.3.2.1. Operation 17](#_Toc448583773)

[5.1.5.3.3. SegModStdInfo 18](#_Toc448583774)

[5.1.5.3.3.1. Operation 18](#_Toc448583775)

[5.1.5.3.4. SegModAdrMpg 18](#_Toc448583776)

[5.1.5.3.4.1. Operation 18](#_Toc448583777)

[5.1.5.4. Outputs 18](#_Toc448583778)

[5.1.6. Sub-Function: OnlineTunRamAdrMpg 19](#_Toc448583779)

[5.1.6.1. Design Rationale 19](#_Toc448583780)

[5.1.6.2. Inputs 19](#_Toc448583781)

[5.1.6.3. Operation 19](#_Toc448583782)

[5.1.6.4. Outputs 19](#_Toc448583783)

[5.1.7. Sub-Function: SetCalPageReq 20](#_Toc448583784)

[5.1.7.1. Design Rationale 20](#_Toc448583785)

[5.1.7.2. Inputs 20](#_Toc448583786)

[5.1.7.3. Operation 20](#_Toc448583787)

[5.1.7.4. Outputs 20](#_Toc448583788)

[5.1.8. Sub-Function: IdxChgMngt 21](#_Toc448583789)

[5.1.8.1. Design Rationale 21](#_Toc448583790)

[5.1.8.2. Inputs 21](#_Toc448583791)

[5.1.8.3. Operation 21](#_Toc448583792)

[5.1.8.4. Outputs 22](#_Toc448583793)

[5.1.9. Sub-Function: MemCopy32Bit and MemCopy8Bit 23](#_Toc448583794)

[5.1.9.1. Design Rationale 23](#_Toc448583795)

[5.1.9.2. Inputs 23](#_Toc448583796)

[5.1.9.3. Operation 23](#_Toc448583797)

[5.1.9.4. Outputs 23](#_Toc448583798)

[5.1.10. Sub-Function: SwtCalIdx 24](#_Toc448583799)

[5.1.10.1. Design Rationale 24](#_Toc448583800)

[5.1.10.2. Inputs 24](#_Toc448583801)

[5.1.10.3. Operation 24](#_Toc448583802)

[5.1.10.4. Outputs 24](#_Toc448583803)

[6. Timing / Execution Constraints 25](#_Toc448583804)

[6.1. Rationale / Comments 25](#_Toc448583805)

[6.2. Rates and State Execution 25](#_Toc448583806)

[7. Serial Communications Interfaces 25](#_Toc448583807)

[8. Additional Information 25](#_Toc448583808)

[9. Revision Record & Change Approval 26](#_Toc448583809)

# High Level Description

This document describes the design of the tuning selection management software component.

# Derived Requirements

None

# Sub-Function Data Flow

The following table describes the data flow in and out of this module.

|  |  |
| --- | --- |
| Type | Name |
| Input | DesIninIdx |
| Input | DesRtIdx |
| Output | ActvGroup |
| Output | ActvIninIdx |
| Output | ActvRtIdx |
| Client | Calc32BitCrc\_u32 |
| Client | RtCalChgReq |
| Client | SetNtcSts |
| Server | CopyCalPageReq |
| Server | GetCalPageReq |
| Server | GetSegInfoReq |
| Server | OnlineTunRamAdrMpg |
| Server | SetCalPageReq |

# Design Rationale

## Design Overview

Tuning selection management creates two copies of the RTE generated calibration table. This table contains pointers to all the calibration software components integrated for a given software application. The component manages the pointers to provide the ability for initialization and runtime calibration changes based on inputs provided by outside applications. These inputs can be, but are not limited to, NvM values, serial communication inputs, or inputs from other software components. This component also manages a RAM buffer for online calibration over XCP.

The following sub sections will show an example of how this component operates.

## Flash Calibration Table

The flash table is generated by the RTE and uses the double pointer method defined by AUTOSAR for accessing calibrations. This provides a base pointer at the start of the flash table, and allows software to index into the calibration tables by an array index instead of knowing the direct names of the calibration structures. When the calibration source ports are connected to the receiver port within another software module, a linkage is generated by the RTE through the base pointer with the software components generated header file. An overview of this configuration is shown in the image below.



## RAM Calibration Table

Tuning Selection Management creates two copies of the flash memory calibration table. Along with each copy a CRC is maintained over each of the copies memory to ensure that the values are correct and were not modified by an outside source.

It is important to note that the cross functional team needs to ensure that the flash calibrations generated by the RTE can be used independently of the EPS system variants, such as but not limited to, motor sizing and C-factor. In the event of a RAM failure or CRC error, the default fault response is to go back using the flash calibration table and the default response is to remove assist immediately. If it can be ensured that the flash defaults are safe to drive for all program variants, the NTC can be reduced to an informative fault to and keep assist active. However, this needs to have all cross functional teams ensure that all criteria are met to make that change.



## Calibration Components

The RTE generated table of pointers point to structures of calibrations that are represented by calibration components. These components do not have any run-time actions (such as initialization or periodic functions) and only provide source ports for the calibrations. Based on the configuration provided by the program team, the description of these components is generated by an outside tool. The names of these calibration components are created based on the following sections. An example of a calibration component name is as follows:

CalRegn01Inin00GroupA

<Flash Memory Location><Calibration Usage><Calibration Indexing><Online Calibration Grouping>

### Flash Memory Location

The flash memory location describes where the calibration is located in flash memory from Nexteer calibration memory regions to customer locations. The region is identified by the prefix “CalRegnXX”, where XX represents the region number. These regions are generic in name, but are defined by the cross-functional team to identify which locations these calibrations are located. For example, CalRegn00 could represent Nexteer calibrations and CalRegn01 could represent customer calibrations.

### Calibration Usage

The usage part of the name identifies of it is an initialization (Inin), a runtime (Rt) calibration, or a common calibration (Cmn). Common calibrations are common among all initialization and runtime calibrations. Initialization calibrations are selected at start up and cannot be changed during operation. Runtime calibrations are selected at startup and can be changed during operation. These indexes can be changed according to the rules that govern those indexes described in the section 4.5.

### Calibration Indexing

The calibration indexing describes that index of the desired initialization and runtime ports a given calibration component belongs. The initialization and runtime indexes are designed to be independent of each other to give systems engineering more flexibility in defining calibrations that need to change at start up or during operation.

### Online Calibration Grouping

The online calibration grouping defines the segment for XCP access. This section of the name is optional and if no grouping is defined then the calibration contained with that calibration component are not allowed to be tuned by XCP.

## Changing Calibration Indexes

During initialization or during runtime, the application may require an index to be changed. Calibrations marked as initialization calibrations can only be changed during initialization and will remain the same values for the rest of the ignition cycle. Runtime calibrations can change at initialization and during runtime operations.

Below is an example of calibrations in the RAM table. The calibration components highlighted in yellow represent the current active components. In this example, the desired run-time and initialization indexes are both zero (0). It is also important to note that only one of the RAM tables is active at any given time by the ECU. This allows tuning selection management the ability to modify the table contents of the unused table and update the base pointer to point to the unused table once all the changes are in place and the checksum is recalculated.



If we assume that the desired runtime index changes from a zero (0) to one (1), tuning selection management will update unused index to reflect the changes as highlighted in red below.



Once all the changes are completed, the base pointer will be updated to point to ram memory index 1 and index 0 will become the ‘scratchpad’ for any further updates.



It is important to note that the references made by the software components do not change the index they are configured to look in to. In the example above if we assume Cal Port 1 is part of the calibration components CalRegnXXRtXXGroupA, the pointer will also point to the same index. As a result, tuning selection management will need to modify the pointer to point from CalRegn01Rt00GroupA to CalRegn01Rt01GroupA to allow the software component to read the new value of the calibration.

## Copying Calibrations to RAM for Online Calibration

When a segment is enabled for online calibration, the software will move the active indexes within the requested group into the XCP RAM buffer. In the example previously provided, if we assume index zero (0) was active for both initialization and runtime calibrations the memory layout would look as follows. Note that CalRegn01Rt01GroupA is not in RAM, because it is not the active index.



XCP can provide the following access during online calibration. This design requires that two pages exists, Flash and RAM. By default, ECU and XCP access both read from Flash and XCP services will need to be executed to change the access to RAM. These services are not covered in this document. For clarification, ECU access simply means where the software components are looking for their calibration values. XCP access simple means where XCP will perform actions, such as read and write.

|  |  |
| --- | --- |
| ECU Access | XCP Access |
| Flash | Flash |
| Flash | RAM |
| RAM | Flash |
| RAM | RAM |

When the ECU is given access to the RAM buffer for software component access, the pointers will change to point to the RAM image instead of the flash table. This will allow the user to modify the calibration values during operation of the ECU.



# Sub-Functions

### Initialization (TunSelnMngtInit1)

***#REQ: The following requirement(s) are met by the design feature below: Requirement ID: ES400A\_48, ES400A\_50, ES400A\_51***

#### Design Rationale

The tuning select management RAM shall be initialized according to the following pseudo code.

#### Inputs

None

#### Implementation

Read DesIninIdx Port

Read DesRtIdx Port

Set all PIMs to default values (0).

Set Page access to Flash for XCP and ECU access

Copy flash table into both MngtRamTbl indexes with MemCopy32Bit

Calculate CRC32Bit over the Flash table, and update both CRC values for the MngtRamTbl with calculated result

IF DesIninIdx not equal to PIM value:

IninIdxFound = IdxChngMngt

IF (IninIdxFound equal TRUE):

Set NTC 1F6, parameter 0 to passed

Update PIM value

Write ActiveIninIdx Port value with DesIninIdx

ELSE:

Set NTC 1F6, parameter 1

ELSE:

Set NTC 1F6, parameter 0 to passed

Write ActiveIninIdx Port value with DesIninIdx

ENDIF

IF DesRtIdx not equal to PIM value:

RtIdxFound = IdxChngMngt

IF (RtIdxFound equal TRUE):

Set NTC 1F7, parameter 0 to passed

Update PIM value

Write ActiveRtIdx Port value with DesRtIdx

ELSE:

Set NTC 1F7, parameter 1

ELSE:

Set NTC 1F7, parameter 0 to passed

Write ActiveRtIdx Port value with DesRtIdx

ENDIF

IF IninIdxFound equals TRUE OR RtIdxFound equals TRUE:

SwtCalIdx()

ENDIF

#### Outputs

None

### Periodic (TunSelnMngtPer1)

***#REQ: The following requirement(s) are met by the design feature below: Requirement ID: ES400A\_18, ES400A\_51, ES400A\_69***

#### Design Rationale

This function queues the request to move calibrations from flash to the RAM buffer for online calibration activities during the next periodic run of tuning selection management. It shall also capture the active initialization and runtime calibration indexes and the selected group (or segment).

#### Inputs

None

#### Operation

Read DesRtIdx Port

Calculate CRC32Bit over active MngtRamTbl

IF CalcCRC not equal to MngtRamTblCRC:

Set NTC 1F8, parameter 1

ELSE:

Set NTC 1F8 to pass

CrcFlt equals FALSE

Call RtCalChgReq\_Oper

ENDIF

IF Cal Copy Status equals Pending:

FOREACH Online calibration component:

IF Online calibration Componet Group Equals Active Group:

Move cal values into RAM buffer with MemCopy8Bit

ENDIF

ENDFOREACH

Set Cal Copy Status to Complete

ENDIF

IF ( (CrcFalt equals FALSE) **AND (**RtCalChgReq\_Oper equals OK) **AND**

**(** Previous Runtime Index does not equal Desired Runtime Index **OR**

ActiveGroup Page Acccess modified **OR**

Cal Copy Status to Complete)):

RtIdxFound = IdxChgMngt()

IF RtIdxFound equals FALSE:

Set NTC 1F7, parameter 1

ELSE:

SwtCalIdx()

Set NTC 1F7, parameter 0 to passed

Write ActvRtIdx port to Desired Runtime Index

ENDIF  
ELSE:

/\* CRC fault, set back to flash table \*/

Rte pointer equals Flash Address

ENDIF

Write ActvGroup port to Current Active Group

Write CalCopySts Port with Cal Copy Status

#### Outputs

None

### Sub-Function: CopyCalPageReq

***#REQ: The following requirement(s) are met by the design feature below: Requirement ID: ES400A\_85***

#### Design Rationale

This function queues the request to move calibrations from flash to the RAM buffer for online calibration activities during the next periodic run of tuning selection management. It shall also capture the active initialization and runtime calibration indexes and the selected group (or segment).

#### Inputs

IN: Seg\_Arg – Segment (or online calibration group) to be copied into RAM.

#### Operation

IF Seg\_Arg is less than MaxNumberOfSegements:

Write Seg\_Arg to PIM

Write Active Initialization Index to PIM

Write Active Runtime Index to PIM

Set Copy Status to Pending

Return OK

ELSE:

Return NOT\_OK

ENDIF

#### Outputs

Return: OK or NOT\_OK

### Sub-Function: GetCalPageReq

***#REQ: The following requirement(s) are met by the design feature below: Requirement ID: ES400A\_80***

#### Design Rationale

This function returns the page of the requested access mode for a requested segment.

#### Inputs

IN: Seg\_Arg – Requested segment for searching.

IN: Mod\_Arg – Mode to search each page for.

IN/OUT: Page\_Arg – Page the requested access mode was found

IN/OUT: Rtn\_Arg – Return value of the function.

#### Operation

IF Mod\_Arg is non-zero and a value of XCP, ECU, or XCP & ECU access:

IF Seg\_Arg is less than MaxNumberOfSegements:

LOOP Each page of for the requested segment until match is found:

IF Mod\_Arg equals Page Access of page in segment:

Page\_Arg equals LoopIndex

Rtn\_Arg = XCP\_CMD\_OK

ELSE:

Rtn\_Arg = PAGE\_MODE\_NOT\_VALID

ENDIF

ENDLOOP

ELSE:

Rtn\_Arg = SEGMENT\_NOT\_VALID

ENDIF

ELSE:

Rtn\_Arg = PAGE\_MODE\_NOT\_VALID

ENDIF

#### Outputs

None

### Sub-Function: GetSegInfoReq

***#REQ: The following requirement(s) are met by the design feature below: Requirement ID: ES400A\_84***

#### Design Rationale

This function returns information of the requested segment.

#### Inputs

IN: Mod\_Arg – Requested mode based on the XCP protocol specification (Get Segment Address Info, Get Segment Standard Info, Get Segment Address Mapping)

IN: Seg\_Arg – Segment to perform the operation on.

IN: SegInfo\_Arg – Sub function option for the commands

IN: MpgIdx\_Arg – Mapping Index mode (only used if Get Segment Address Mapping mode is used)

IN/OUT: Resp\_Arg – Buffer for the command response

IN/OUT: RespLen\_Arg – Length of the command response

IN/OUT: Rtn\_Arg – Return value of the function.

#### Operation

##### Mode Decision

IF Seg\_Arg is less than MaxNumberOfSegements:

If Mod\_Arg equals Get Segment Address Info:

Rtn\_Arg = SegModAdrInfo(Seg\_Arg, SegInfo\_Arg, Resp\_Arg,

RespLen\_Arg)

ELSE IF Mod\_Arg equals Get Segment Standard Info:

Rtn\_Arg = SegModStdInfo(Seg\_Arg, Resp\_Arg, RespLen\_Arg)

ELSE IF Mod\_Arg equals Get Segment Address Mapping:

Rtn\_Arg = SegModAdrMpg(Seg\_Arg, SegInfo\_Arg, MpdIdx\_Arg,

Resp\_Arg, RespLen\_Arg)

ELSE:

Rtn\_Arg = OUT\_OF\_RANGE

ENDIF

ELSE:

Rtn\_Arg = SEGMENT\_NOT\_VALID

ENDIF

##### SegModAdrInfo

###### Operation

Rtn\_Arg = CMD\_OK

IF SegInfo\_Arg equals Segment Address:

ReturnData = Starting Address of XCP RAM buffer location

ELSE IF SegInfo\_Arg equals Segment Length:

ReturnData = Number of bytes of the entire segment

ELSE:

Rtn\_Arg = OUT\_OF\_RANGE

ENDIF

IF Rtn\_Arg equals CMD\_OK:

Populate Resp\_Arg with ReturnData per the XCP protocol specification

Resp\_Arg = 8

ENDIF

##### SegModStdInfo

###### Operation

Set Counter to 0

FOREACH calibration component:

IF Seg\_Arg equals the calibration component grouping:

Increment Counter

ENDIF

ENDFOREACH

Populate Resp\_Arg with the counted values in Counter per the XCP protocol specification

Resp\_Arg = 6

##### SegModAdrMpg

###### Operation

Since calibration components may not be adjacent within the flash generated table, the following for loop shall create a smaller array containing the indexes of each calibration component within the selected segment. Also note that MpgIdxInfo\_Arg is the same as SegInfo\_Arg. However, per the XCP specification the values have different meanings in the different subfunctions.

Set Counter to 0

FOREACH calibration component:

IF Seg\_Arg equals the calibration component grouping:

Write Loop Index into SubArray

Increment ArrayIndex

ENDIF

ENDFOREACH

IF MpgIdx\_Arg is less than Items in SubArray:

Rtn\_Arg = CMD\_OK

IF MpgIdxInfo\_Arg equals Source Address:

ReturnData equals the selected cal component flash address

ELSE IF MpgIdxInfo\_Arg equals Destination Address:

ReturnData equals the selected cal component RAM address

ELSE IF MpgIdxInfo\_Arg equals Length:

ReturnData equals the selected cal components length

Populate Resp\_Arg with the ReturnData per the XCP protocol specification

Resp\_Arg = 8

#### Outputs

None

### Sub-Function: OnlineTunRamAdrMpg

#### Design Rationale

This function translates a flash address to a RAM address for an active group that is in RAM. This function is for eTool and CANoe to use the flash values to modify and read tuning that is located in RAM.

Note, <<RAM range>> must be configurable. By default the range should only between that of the tuning select RAM buffer and reject writes everywhere else. However, a build option should be included to open the full RAM range for internal testing.

#### Inputs

IN: ReqAdr\_Arg – Requested segment for searching.

IN/OUT: CorrdAdr\_Arg – Mode to set the page.

IN: ReqTyp\_Arg – Read or Write request

#### Operation

Rtn = NOT\_OK

IF ReqAdr\_Arg <= MAX\_FLASH\_ADDRESS:

IF XCP Access on RAM page:

FOREACH Online Calibration Component:

IF (ReqAdr\_Arg < FlashTableBaseAdr + FlashTableSize **AND**

Active Group equals OnlineCalibrationGroup)

AdrOffs equals ReqAdr\_Arg – FlashTableBaseAdr

CorrdAdr\_Arg equals OnlineCalibrationGroupRAMAddress

+ AdrOffs

Rtn = OK

ENDIF

ENDFOREACH

ELSE:

IF ReqTyp\_Arg equals Read:

CorrdAdr\_Arg = ReqAdr\_Arg

Rtn = OK

ENDIF

ENDIF

ELSE:

IF ReqTyp\_Arg equals Write:

IF ReqAdr\_Arg within <<RAM range>>:

CorrdAdr\_Arg = ReqAdr\_Arg

Rtn = OK

ENDIF

ENDIF

ENDIF

#### Outputs

Rtn

### Sub-Function: SetCalPageReq

***#REQ: The following requirement(s) are met by the design feature below: Requirement ID: ES400A\_86***

#### Design Rationale

This function sets the page to the requested access mode for a requested segment.

#### Inputs

IN: Seg\_Arg – Requested segment for searching.

IN: Mod\_Arg – Mode to set the page.

IN/OUT: Page\_Arg – Page the requested access mode was found

#### Operation

IF (Mod\_Arg is non-zero and a value of XCP, ECU, or XCP & ECU access **AND**

Seg\_Arg is less than MaxNumberOfSegments **AND**

Page\_Arg is less than MaxNumberOfPages **AND**

Seg\_Arg is equal to the Active Group):

FOREACH page in the segment:

IF Page\_Arg equals LoopIndex:

Logic OR In Page Access (sets only the page access)

ELSE:

Logic AND to clear Page Access (keeps other access same)

ENDIF

ENDFOREACH

ENDIF

#### Outputs

None

### Sub-Function: IdxChgMngt

#### Design Rationale

Index change management moves calibration indexes to match the desired runtime and initialization values.

#### Inputs

IN/OUT: SeldIdx\_Arg – Selected runtime / initialization index

IN/OUT: GendCalTblSize\_Arg – Size of the calibration component

IN: GendCalTbl\_Arg – Pointer to the calibration data

#### Operation

Set Swt to unused index in the management RAM table (MngtRamTbl)

Calculate CRC32Bit on unused index of MngtRamTbl

IF CalcCRC does not equal CRC in MngtRamTbl:

Set NTC 1F8 with parameter 2

ELSE:

IF ( Calibration copy from Flash to RAM has completed **AND**

Page Access has been modified **AND**

Page Access for the RAM Page is active):

FOREACH Calibration Component:

If Active Group matches Calibration Component Group

MemCopy32Bit(MngtRamTbl for Cal Component Address,

XCP RAM buffer Address, 1)

END IF

ENDFOREACH

ELSE:

/\* Restore entire table to flash defaults \*/

MemCopy32Bit(MngtRamTbl, Flash Address, TblSize)

FOREACH Initialization Cal Component:

IF Active Initialization PIM value equals Cal Component:

MemCopy32Bit(MngtRamTbl for Cal Component, Flash

Address, 1)

ENDIF

ENDFOREACH

ENDIF

IndexFound equals FALSE

FOREACH Entry in GendCalTbl\_Arg:

IF GendCalTbl index equals SeldIdx\_Arg:

IF (Calibration copy from Flash to RAM has completed **AND**

Page Access has been modified **AND**

Active Runtime equals GendCalTbl index):

FOREACH Online Calibration Component:

MemCopy32Bit(MngtRamTbl CalComponent Address,

XCP RAM buffer Address, 1)

ENDFOREACH

ELSE:

IF MngtRamTbl SrAddr = DestAddr:

MemCopy32Bit(MngtRamTbl CalComp Addr,

Flash Address, 1)

ELSE

MemCopy32Bit(MngtRamTbl CalComp DestAddr,

MngtRamTbl CalComp SrcAddr, 1)

ENDIF

IndexFound equals TRUE

ENDIF

ENDIF

ENDFOREACH

IF IndexFound equals TRUE:

Calculate CRC32Biton on unused MngtRamTbl

ENDIF

ENDIF

#### Outputs

None

### Sub-Function: MemCopy32Bit and MemCopy8Bit

#### Design Rationale

In an effort to limit the amount of time while move calibrations from Flash to RAM or populating the RAM tables with the flash pointers, the memory copy functions follow the same pseudo code, but their access widths vary from 32-bit to 8-bit for atomic writes.

#### Inputs

IN/OUT: Dest\_Arg – Destination address

IN/OUT: Src\_Arg – Source address of data

IN: Len\_Arg – Number of bytes (8 bit function) or words (32 bit function) to copy

#### Operation

FOR 0 to Len\_Arg:

Dest\_Arg[LoopIndex] equals Src\_Arg[LoopIndex]

ENDFOR

#### Outputs

None

### Sub-Function: SwtCalIdx

#### Design Rationale

This sub-function shall be used whenever a change of RAM indexes occurs. The design will switch the pointer to the new table, and replace the old table with the new values so both RAM images match. The software implementation shall implement the following pseudo code.

#### Inputs

None

#### Operation

Set Swt to unused index in the management RAM table (MngtRamTbl)

MemCopy32Bit(Current Index in MngtRamTbl, Unused Index in MngtRamTbl,

MngtRamTblSize)

Update RAM Page Access with Current Page Access

Set Rte pointer to unused Index in MngtRamTbl

Update PIM for new Swt value

#### Outputs

None

# Timing / Execution Constraints

## Rationale / Comments

The functions defined in this document are synchronous functions and are not required to be scheduled to run at a periodic rate.

## Rates and State Execution

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sub-Function Name** | **Rate (ms)** | **Cold Init** | **Warm Init** | **Operate** | **Disable** |
| TunSelnMngtInit1 | N/A | Y\* | Do Not Execute | Do Not Execute | Do Not Execute |
| TunSelnMngtPer1 | N/A | Do Not Execute | Y | Y | Y |
| CopyCalPageReq | N/A | Do Not Execute | N/A | N/A | N/A |
| GetCalPageReq | N/A | Do Not Execute | N/A | N/A | N/A |
| GetSegInfoReq | N/A | Do Not Execute | N/A | N/A | N/A |
| OnlineTunRamAdrMpg | N/A | Do Not Execute | N/A | N/A | N/A |
| SetCalPageReq | N/A | Do Not Execute | N/A | N/A | N/A |

Y\* -- No calibration access can be performed by any software component unless this function is executed.

# Serial Communications Interfaces

None

# Additional Information

None

# Revision Record & Change Approval

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Rev** | **Date** | **Change Control #** | **Drw** | **Change Description** |
| 01.00.00 | 16-Apr-16 | EA4#1839 | KJS | Initial Release of this document |