Run #4

You are a Requirements Engineer for the "Standard Firmware" project. I will provide a set of linked System Requirements and their corresponding Software Requirements. Your task is to thoroughly evaluate whether every aspect of the system requirement is sufficiently detailed in its linked software requirements. Note that the system and software requirements are restricted to their scope system and software respectively hence can have different levels of abstraction. The software requirements can contain additional details than its linked system requirements as long as it adds more details and is not unrelated to its system requirement. If the additional condition is not at all mentioned in the system requirement, then flag it. It is acceptable that one SYSRS can be linked to multiple SWRS or vice versa. Please stick to the information provided in the requirements alone. Don't assume anything that is not explicitly mentioned. Please provide a single combined rating for each set of System Requirements (SYSRS) and their corresponding Software Requirements (SWRS).

Condition Check: Verify if the conditions such as 'After event X occurs', 'before event Y', 'IF X THEN' in the system requirements are matched in the software requirements. Only these constitute to the term 'conditions'.
Action and Response Check: Confirm if the actions and responses described in the system requirements are correctly implemented in the software requirements.
Parameter Sources Check: Ensure that all parameters mentioned in the system requirements are addressed in the software requirements including information on where these parameters can be found or how they are sourced. If the source is not mentioned, indicate that it needs to be identified.
Additional Conditions Check: Identify if there are any additional conditions in the software requirements that are not specified in the system requirements. If they are not relevant to the system requirement, then flag it. Ensure that all these checks are given equal priority to avoid over sight. Every observation is critical, be it major or minor. Use this to provide rating:
For each set, provide the following ratings and reasons:
Then use your checks to rate the quality of the link.
Meaningfulness Rating (High/Medium/Low):
Reason for Meaningfulness Rating: List all the issues.
Completeness Rating (High/Medium/Low):
Reason for Completeness Rating: List all the issues.
Overall Quality of the Link (High/Medium/Low):
Reason for Overall Quality Rating:
If any aspects are not fully addressed or require clarification, suggest specific changes or revisions to the software requirements to better align with the system requirements.

Please provide the specific system requirements (SYSRS) and their corresponding software requirements (SWRS) so that I can conduct a thorough evaluation based on the guidelines you outlined.

Meaningfulness of the link:
High: The software requirement clearly and directly supports or implements the system requirement, adding significant value and relevance. The link between them is strong and logical.
Medium: The software requirement generally supports the system requirement, but there may be a need for better alignment or more specific details.
Low: The software requirement is poorly aligned or irrelevant to the system requirement, making the link weak or unclear.

Completeness of the link:
High: The link between the system and software requirement is fully detailed, covering all necessary aspects needed for implementation. No significant information is missing.
Medium: The link is somewhat complete but lacks some critical details or clarifications that would improve understanding or implementation.
Low: The link is significantly incomplete or lacks essential details, making it difficult to see how the software requirement fully supports the system requirement.

Formula-Based Overall Quality:
High: If both Meaningfulness and Completeness are rated as High, the Overall Quality is High.
Medium: If either Meaningfulness or Completeness is Medium while the other is High, or if both are Medium, the Overall Quality is Medium.
Low: If either Meaningfulness or Completeness is Low, the Overall Quality is Low.

Thank you for providing the clarification regarding the rating structure. Please provide the specific System Requirements (SYSRS) and their corresponding Software Requirements (SWRS) for evaluation so I can apply the checks and provide ratings accordingly.

SYSRS 021: Diagnostic - Retry configuration: The firmware shall allow configurability to select the retry for open and short faults independently "SWRS 023: Open/Short retry: IF the software detects an open or short error AND retry is configured as enabled in the DEFAULT OPEN SHORT RETRY value in the COMPILE TIME CONFIGURATION data, the affected channel shall be re-enabled for one TIMESLICE with the following setup at least every 200 milliseconds

SWRS 024: Diagnostic configuration items : DEFAULT OPEN SHORT RETRY"

Evaluation of System Requirement (SYSRS 021) and Corresponding Software Requirements (SWRS 023 & SWRS 024)

Condition Check:

- **SYSRS 021** specifies that the firmware should allow configuration for selecting retries for open and short faults independently.

- **SWRS 023** mentions a conditional retry mechanism for open and short errors, where the retry is contingent on the software detecting the error and a "DEFAULT OPEN SHORT RETRY" configuration being enabled. However, it does not explicitly mention configurability for independent selection of retries for open and short faults.
    - The condition of retrying based on the configuration is covered, but there is no clarity if retries for "open" and "short" are handled independently. The system requirement implies separate configurability for each fault type, while the software requirement groups them together without clarification of independent settings.

Action and Response Check:

- **SYSRS 021** expects that the system allows independent configuration for retrying open and short faults.
- **SWRS 023** describes the action of re-enabling the affected channel for a specific timeslice upon detecting a fault, but it does not specify independent handling for open and short faults. It only mentions retries if the "DEFAULT OPEN SHORT RETRY" configuration is enabled, which doesn't fully implement the independent configuration requirement stated in SYSRS 021.

Parameter Sources Check:

- **SYSRS 021** mentions "retry" for open and short faults, but it does not specify where the retry values come from or how they are set.
- **SWRS 023** does provide the source of retry configuration as part of the "DEFAULT OPEN SHORT RETRY" value in the compile-time configuration, which is good. However, it doesn't clarify how the retry configuration for open and short faults can be independently set, as required by the system requirement.
- **SWRS 024** introduces a "DEFAULT OPEN SHORT RETRY" parameter, but again, this parameter does not clarify independent control for retries on open and short faults. It needs to clarify whether it applies to both types of faults together or allows separate control.

Additional Conditions Check:

- **SWRS 023** introduces the condition that the affected channel shall be re-enabled for a timeslice of 200 milliseconds. This detail is an additional implementation specification that is not mentioned in **SYSRS 021**. While it adds more information about how retries will be managed, it does not contradict the system requirement.
- However, the additional condition of a single timeslice of 200 ms is not irrelevant but should be clarified in the system requirement if it is critical to the diagnostic functionality.

Ratings:

Meaningfulness Rating: Medium

- **Reason:** The software requirements generally support the system requirement, but there is a lack of clarity about whether retries for open and short faults are independently configurable. The software requirements lump both fault types together without explicitly detailing independent retry configuration, which weakens the alignment. The additional detail about timeslice configuration adds meaningful information but needs more alignment with the system requirement’s intent.

Completeness Rating: Medium

- **Reason:** The software requirements describe retry handling and configuration but miss the crucial aspect of independent retry configuration for open and short faults as stated in the system requirement. The parameters are partially addressed, and more detail is needed on how the retry configuration can be independently set for different fault types.

Overall Quality of the Link: Medium

- **Reason:** While the software requirements are somewhat aligned with the system requirement, they lack specific details about independent fault handling, which reduces completeness and clarity. More clarity and additional details are needed in the software requirements to fully meet the expectations of the system requirement.

Suggested Revisions:

1. **SWRS 023** should be updated to explicitly mention independent configuration for retries of open and short faults, as required by **SYSRS 021**.
2. **SWRS 024** should clarify whether the "DEFAULT OPEN SHORT RETRY" applies to both fault types together or independently, and if independent, provide separate configuration items for open and short faults.
3. **SYSRS 021** could be expanded to include more specifics on retry behavior, such as timeslice duration (200 ms) and the retry mechanism, if they are critical to the system's operation.

"SYSRS 012: BEFORE Leaving the Discovery Mode
AND IF the effective value of the parameter DSI3_physicalAddress is NOT equal to the default value of the Parameter DSI3_physicalAddress
THEN
the Sensor IC shall
set Error: err_discPhysAddrMismatch = 0x1 (""error detected"").
"	"SWRS 012 : BEFORE Leaving the Discovery Mode
AND IF the effective value of the Parameter DSI3_physicalAddress (RAM) is NOT equal to the default value of the Parameter DSI3_physicalAddress (NVM)
THEN
the Software shall
set Error: err_discPhysAddrMismatch = 0x1 (""error detected"").
OTHERWISE
the Software shall
set Error: err_discPhysAddrMismatch = 0x0 (""no error"").
"

Evaluation of System Requirement (SYSRS 012) and Corresponding Software Requirement (SWRS 012)

Condition Check:

- **SYSRS 012** specifies the condition **BEFORE Leaving the Discovery Mode** and **IF the effective value of the parameter DSI3_physicalAddress** is not equal to the default value, then the sensor

IC must set an error. The condition is clearly stated with the comparison between the current (effective) and default values of the `` `DSI3_physicalAddress` ``.

- **SWRS 012** mirrors this condition accurately. It also checks **BEFORE Leaving the Discovery Mode** and compares the effective value of `` `DSI3_physicalAddress` `` in RAM with the default value stored in non-volatile memory (NVM). The RAM/NVM comparison adds a small clarification regarding where these values are stored, which aligns with the intent of the system requirement.
  - The condition logic is correct, and the additional specificity of checking RAM versus NVM values helps the software implementation without conflicting with the system requirement.

Action and Response Check:

- **SYSRS 012** expects the Sensor IC to set the `` `err_discPhysAddrMismatch` `` error to `` `0x1` `` (indicating error detected) if the values don't match.
- **SWRS 012** correctly implements this by setting the error to `` `0x1` `` if the RAM and NVM values of `` `DSI3_physicalAddress` `` do not match. Additionally, **SWRS 012** introduces an **OTHERWISE** condition, which sets the error to `` `0x0` `` (no error).
  - This additional action of clearing the error (`` `0x0` ``) when no mismatch is found is logical and provides more comprehensive behavior, but it is not explicitly mentioned in the system requirement. It does not contradict SYSRS 012 and adds useful error handling, though it could be seen as an extension of the expected behavior.

Parameter Sources Check:

- **SYSRS 012** mentions the parameter `` `DSI3_physicalAddress` `` without specifying where these values are stored. It simply requires checking the effective value against the default.
- **SWRS 012** improves upon this by clarifying that the effective value is in RAM and the default is in NVM. This added information is crucial for understanding how the system compares the values and does not conflict with SYSRS 012. The software requirement provides a clear explanation of where these parameters are sourced, making it more detailed and helpful for implementation.

Additional Conditions Check:

- **SWRS 012** introduces an additional condition: the `` `OTHERWISE` `` clause where the software sets the error to `` `0x0` `` if no mismatch is found. This is an enhancement that adds completeness to the error-handling process. Although this condition is not mentioned in SYSRS 012, it does not detract from the requirement; rather, it adds an appropriate handling case for when no error is detected.

Ratings:

Meaningfulness Rating: High

- **Reason:** The software requirement aligns well with the system requirement. It directly implements the condition and action expected in SYSRS 012. The additional clarification regarding RAM and NVM sources for the `` `DSI3_physicalAddress` `` parameter is meaningful and enhances understanding without deviating from the system requirement's intent. The introduction of the `` `OTHERWISE` `` condition adds robustness but does not undermine the original requirement.

Completeness Rating: High

- **Reason:** The software requirement fully covers the system requirement and provides additional useful detail regarding memory sources (RAM vs. NVM). The additional `` `OTHERWISE` `` clause, while not mentioned in SYSRS 012, adds completeness by handling the case where no error is detected. The parameter source is clearly addressed, and all conditions are covered.

Overall Quality of the Link: High

- **Reason:** The software requirement effectively implements the system requirement with additional clarifications that do not contradict the system-level intent. It provides full coverage of conditions and actions while adding details that improve the software's behavior.

Suggested Revisions:

1. **SYSRS 012** could be updated to explicitly mention what should happen if no mismatch is found (e.g., setting the error to `` `0x0` ``), aligning it with the additional detail provided in **SWRS 012**.
2. If applicable, the system requirement could also specify that the comparison is between the values stored in RAM and NVM, as detailed in **SWRS 012**, for better alignment.

"SYSRS 010: If Performing Memory Area Modification
AND AFTER reception of CRM_CMD 0x9: Take Data or Send Data
THEN
the Sensor IC shall
cache the CRM_CMD 0x9 Parameter Data to the Data Cache.

SYSRS 011: If Performing Memory Area Modification
AND AFTER reception of CRM_CMD 0x9: Take Data or Send Data
THEN
the Sensor IC shall
increment the CRM_CMD 0x8 Parameter StartAddress by 1 word address.

"   "SWRS 011: If Performing Memory Area Modification
AND AFTER reception of CRM_CMD 0x9: Take Data or Send Data
THEN
the Software shall store the CRM_CMD 0x9 Parameter Data to RAM until the command CRM_CMD 0xA: Perform Memory Operation with CRM_CMD 0xA Parameter MemoryOperation == 0x2 ""Write to NVM"" is received."

Evaluation of System Requirements (SYSRS 010 & SYSRS 011) and Corresponding Software Requirement (SWRS 011)

Condition Check:

- **SYSRS 010** and **SYSRS 011** both specify the condition: **If Performing Memory Area Modification AND AFTER reception of CRM_CMD 0x9: Take Data or Send Data**. These are consistent across both system requirements.

- **SWRS 011** mirrors this condition accurately. It specifies the same condition for memory modification and reception of CRM_CMD 0x9 before taking any action. The condition is implemented correctly in the software requirement.

Action and Response Check:

- **SYSRS 010** specifies that the Sensor IC should **cache the CRM_CMD 0x9 Parameter Data to the Data Cache** after receiving CRM_CMD 0x9.
  - **SWRS 011** describes a similar action where the software stores the parameter data into **RAM** (instead of explicitly mentioning the "Data Cache") until the command CRM_CMD 0xA with a specific operation (write to NVM) is received. This adds an additional step regarding CRM_CMD 0xA but introduces a slight mismatch with SYSRS 010, as caching in "RAM" may not precisely match caching to "Data Cache." Further clarification is needed regarding whether RAM is synonymous with the intended data cache.
- **SYSRS 011** specifies that after receiving CRM_CMD 0x9, the Sensor IC shall **increment the CRM_CMD 0x8 Parameter StartAddress by 1 word address**.
  - **SWRS 011** does not mention this action of incrementing the CRM_CMD 0x8 StartAddress. This is a significant omission, as the action is critical to the correct implementation of SYSRS 011. The software requirement focuses only on caching the CRM_CMD 0x9 data and misses the start address incrementing action, which creates a gap in alignment with the system requirement.

Parameter Sources Check:

- **SYSRS 010** and **SYSRS 011** both refer to CRM_CMD 0x9 for the parameter data and CRM_CMD 0x8 for the start address.
  - **SWRS 011** correctly handles the CRM_CMD 0x9 parameter data, specifying that it is stored in RAM (or cache) until a subsequent command (CRM_CMD 0xA) is received. However, the source of the parameter (CRM_CMD 0x8) for incrementing the start address, as required in **SYSRS 011**, is not addressed in **SWRS 011**.
  - The software requirement does not reference CRM_CMD 0x8 at all, which creates a significant gap in the handling of the parameter start address specified by the system requirement.

Additional Conditions Check:

- **SWRS 011** introduces an additional condition: the parameter data stored in RAM remains there until CRM_CMD 0xA with a specific operation (MemoryOperation == 0x2) is received. This is an additional implementation detail not mentioned in SYSRS 010.
  - While this extra condition makes sense in a software context and does not contradict the system requirement, it adds complexity that was not anticipated in the system requirement. This is a potentially relevant condition that enhances how data is handled, but the system requirement does not mention this interaction with CRM_CMD 0xA, which may need to be clarified if this is important for overall behavior.

Ratings:

Meaningfulness Rating: Medium

- **Reason:** The software requirement is generally aligned with the system requirements but has several issues:

1. **SWRS 011** handles the data caching, but it introduces the use of RAM rather than an explicit "Data Cache" as stated in **SYSRS 010**. This could be acceptable depending on whether RAM functions as the intended cache, but the discrepancy needs clarification.
2. **SWRS 011** omits the critical action from **SYSRS 011** of incrementing the CRM_CMD 0x8 StartAddress, which is a necessary part of the system's intended behavior.
3. The additional condition related to CRM_CMD 0xA introduces relevant details but was not anticipated in the system requirements. It does not weaken the link but adds complexity.

Completeness Rating: Low

- **Reason:** The software requirement misses a key part of **SYSRS 011** regarding the increment of the CRM_CMD 0x8 StartAddress by 1 word. This omission is critical to the behavior outlined in **SYSRS 011** and affects the overall completeness of the link.
  - Additionally, while the handling of CRM_CMD 0x9 is detailed in **SWRS 011**, the mismatch between caching in "RAM" versus "Data Cache" (as stated in **SYSRS 010**) needs further clarification to determine if it fully meets the system's requirement.

Overall Quality of the Link: Low

- **Reason:** While **SWRS 011** correctly implements parts of **SYSRS 010** (data caching) and introduces useful additional conditions, the failure to address the core action in **SYSRS 011** (incrementing the CRM_CMD 0x8 StartAddress) significantly weakens the link. The additional condition related to CRM_CMD 0xA also adds complexity that wasn't anticipated in the system requirements, and further alignment is needed.

Suggested Revisions:

1. **SWRS 011** should be updated to include the action of incrementing the CRM_CMD 0x8 Parameter StartAddress by 1 word, as required by **SYSRS 011**.
2. Clarify whether storing data in RAM (as mentioned in **SWRS 011**) is equivalent to caching in the Data Cache, or modify the software requirement to directly refer to the Data Cache if that is what is intended by **SYSRS 010**.
3. Consider updating **SYSRS 010** to acknowledge the interaction with CRM_CMD 0xA (MemoryOperation == 0x2) introduced in **SWRS 011**, if that operation is relevant to the broader system behavior.

"SYSRS 009: If Performing Memory Access in Broadcast
AND IF the state if the Memory Area identified by CRM_CMD 0x8 Parameter MemoryArea for the operation as determined by CRM_CMD 0x8 Operation Flag is in the state Locked,
THEN
the Sensor IC shall
respond with PDCM_RSP P0: Report CRM_RSP Information containing CRM_RSP 0x8: Report Lock Status and CRM_RSP 0x8 Flag KeyRequest = 0b1 (""key requested"").

" "SWRS 010: If Performing Memory Access in Broadcast
AND IF the state if the Memory Area identified by CRM_CMD 0x8 Parameter MemoryArea for the operation as determined by CRM_CMD 0x8 Operation Flag is in the state Locked,
THEN
the Software shall prepare to

respond with PDCM_RSP P0: Report CRM_RSP Information containing CRM_RSP 0x8: Report Lock Status and CRM_RSP 0x8 Flag KeyRequest = 0b1 (""key requested"").
"

Evaluation of System Requirement (SYSRS 009) and Corresponding Software Requirement (SWRS 010)

Condition Check:

- **SYSRS 009** specifies that **If Performing Memory Access in Broadcast** and **IF the state of the Memory Area identified by CRM_CMD 0x8 Parameter MemoryArea for the operation, as determined by CRM_CMD 0x8 Operation Flag, is in the state Locked**, then a specific response should be triggered.
- **SWRS 010** accurately reflects this condition, using the same condition language regarding memory access, broadcast, and checking the locked state of the memory area as determined by CRM_CMD 0x8.
  - The condition is correctly mirrored without any divergence from the system requirement, indicating a proper alignment between the system and software requirements in this regard.

Action and Response Check:

- **SYSRS 009** requires that the **Sensor IC shall respond with PDCM_RSP P0: Report CRM_RSP Information containing CRM_RSP 0x8: Report Lock Status and CRM_RSP 0x8 Flag KeyRequest = 0b1** (indicating a "key requested" status).
- **SWRS 010** indicates that the **Software shall prepare to respond** with the same information: **PDCM_RSP P0: Report CRM_RSP Information containing CRM_RSP 0x8: Report Lock Status and CRM_RSP 0x8 Flag KeyRequest = 0b1**.
  - The main difference here is the wording in **SWRS 010**: it says the software shall "prepare to respond," while **SYSRS 009** directly instructs the sensor IC to "respond." The "prepare to respond" wording introduces ambiguity about whether the response is actually sent or just prepared. The system requirement clearly expects an immediate response, so the software requirement needs to specify that the response is actually sent, not just prepared.

Parameter Sources Check:

- **SYSRS 009** refers to **CRM_CMD 0x8 Parameter MemoryArea** and **CRM_CMD 0x8 Operation Flag** to determine the locked state of the memory area.
- **SWRS 010** mirrors this by also referencing **CRM_CMD 0x8 Parameter MemoryArea** and **CRM_CMD 0x8 Operation Flag** for checking the state of the memory area, aligning with the system requirement.
  - The parameters are fully addressed in both the system and software requirements, and the sources are clearly defined.

Additional Conditions Check:

- **SWRS 010** does not introduce any new conditions. It aligns with the original condition in **SYSRS 009** regarding performing memory access in broadcast and checking the locked state. There are no additional conditions that need flagging.

Ratings:

Meaningfulness Rating: Medium

- **Reason:** The software requirement mostly supports the system requirement. However, the use of the phrase **"prepare to respond"** introduces ambiguity. **SYSRS 009** expects the Sensor IC to **actually respond**, while **SWRS 010** suggests a preparation stage, which does not guarantee that the response is sent. This weakens the link and causes some misalignment.

Completeness Rating: Medium

- **Reason:** The software requirement captures most of the system requirement's intent, including the correct condition and parameter usage. However, it does not ensure the actual response action is completed, which is a critical part of **SYSRS 009**. This reduces the completeness since the system requirement expects a definite response, not just preparation for one.

Overall Quality of the Link: Medium

- **Reason:** While **SWRS 010** mostly mirrors **SYSRS 009**, the ambiguity around whether the software is just preparing or actually sending the response creates a gap. The conditions and parameters are well aligned, but the missing explicit action weakens the overall quality of the link.

Suggested Revisions:

1. **SWRS 010** should be revised to state that the software shall **actually respond**, rather than just **prepare to respond**. This change would ensure the software requirement fully aligns with the expected behavior of **SYSRS 009**.
2. Consider making the distinction between "preparing" and "responding" explicit if there is a staged process, but ensure that the final action of responding with the PDCM_RSP is guaranteed and described in the software requirement.

"SYSRS 008: If Performing Memory Access With Set Key
AND If Write Operation is Requested
AND If Memory Area Is Set Write Accessible
THEN
the Sensor IC shall
set the state of the Memory Area identified by CRM_CMD 0x8 Parameter MemoryArea to the state Locked." "SWRS 009: If Performing Memory Access
AND If the Key is NOT Set
AND If Write Operation is Requested
AND If Memory Area Is Set Read Accessible
THEN
the Software shall
set the state of the Memory Area identified by CRM_CMD 0x10 Parameter MemoryArea to the state Locked.
"

Evaluation of System Requirement (SYSRS 008) and Corresponding Software Requirement (SWRS 009)

Condition Check:

- **SYSRS 008** defines the condition **If Performing Memory Access With Set Key**, **If Write Operation is Requested**, and **If the Memory Area Is Set Write Accessible**. It specifies that under these conditions, the Sensor IC should set the memory area to a locked state.
- **SWRS 009** differs from **SYSRS 008** in a significant way. It introduces the condition **If the Key is NOT Set**, which directly contradicts the system requirement that specifies performing the memory access **With Set Key**. This is a critical mismatch. Additionally, **SWRS 009** checks if the **Memory Area is Set Read Accessible**, while **SYSRS 008** refers to the memory area being **Set Write Accessible**. These differences in conditions create a significant divergence between the system and software requirements.

Action and Response Check:

- **SYSRS 008** specifies that under the stated conditions, the Sensor IC shall **set the state of the Memory Area identified by CRM_CMD 0x8 Parameter MemoryArea to the state Locked**.
- **SWRS 009** also specifies that under its conditions, the software shall **set the state of the Memory Area identified by CRM_CMD 0x10 Parameter MemoryArea to the state Locked**. However, there are two major differences:
  1. **CRM_CMD 0x10** is referenced in **SWRS 009**, whereas **SYSRS 008** refers to **CRM_CMD 0x8**. This suggests that the two are potentially addressing different commands or contexts, which causes confusion.
  2. The conditions for locking the memory area are different (Set Key vs. Key NOT Set, Write Accessible vs. Read Accessible), which means that the software requirement is not directly implementing the action under the same conditions as required by the system.

Parameter Sources Check:

- **SYSRS 008** uses **CRM_CMD 0x8 Parameter MemoryArea** as the source to identify the memory area.
- **SWRS 009** instead uses **CRM_CMD 0x10 Parameter MemoryArea**. This change in the parameter source further complicates alignment since the system and software requirements are referencing different command parameters.

Additional Conditions Check:

- **SWRS 009** introduces an additional condition where it checks if the key is **NOT Set**, while **SYSRS 008** requires the key to be **Set**. This additional condition is not only irrelevant but directly contradicts the condition in the system requirement, creating a significant misalignment.
- The software requirement also introduces the condition that the memory area is **Read Accessible** instead of **Write Accessible**, which again deviates from the system requirement.

Ratings:

Meaningfulness Rating: Low

- **Reason:** The software requirement does not meaningfully support the system requirement due to significant differences in conditions (Key Set vs. Key NOT Set, Write Accessible vs. Read Accessible). Additionally, the software requirement references a different command (**CRM_CMD 0x10** vs. **CRM_CMD 0x8**), further weakening the connection. The introduced conditions in **SWRS 009** are unrelated and contradictory to those in **SYSRS 008**.

Completeness Rating: Low

- **Reason:** The software requirement fails to address the system requirement properly. It alters the conditions for when the memory area should be locked and references different parameters (CRM_CMD 0x10 vs. CRM_CMD 0x8). These omissions and mismatches in conditions significantly reduce completeness, as the expected behavior in **SYSRS 008** is not implemented.

Overall Quality of the Link: Low

- **Reason:** The software requirement is poorly aligned with the system requirement. It changes key conditions, uses a different parameter source, and introduces unrelated conditions, leading to a low overall quality rating. There is insufficient linkage between the actions expected in **SYSRS 008** and the behavior described in **SWRS 009**.

Suggested Revisions:

1. **SWRS 009** should be updated to match the conditions in **SYSRS 008**:
   - Ensure that the software operates **With Set Key** instead of **If the Key is NOT Set**.
   - Ensure that the software checks if the memory area is **Set Write Accessible** instead of **Read Accessible**.
2. **SWRS 009** should reference **CRM_CMD 0x8** to align with the system requirement instead of **CRM_CMD 0x10** unless there is a valid reason for the command change, in which case that reason should be explained or justified.
3. Any additional conditions, such as checking if the key is not set or changing accessibility types, should be either removed or clarified if they are necessary to meet other system behaviors.

"SYSRS 007: If Performing Memory Access
AND If the Key is Set
AND If Read Operation is Requested
AND If Memory Area Is Set Read Accessible
THEN
the Sensor IC shall
set the state of the Memory Area identified byCRM_CMD 0x8 Parameter MemoryArea to the state Locked.
"   SWRS 008: If Performing Memory Access AND If the Key is Set AND If Read Operation is Requested   AND If Memory Area Is Set Read Accessible THEN the Software shall set the state of the Memory Area identified by CRM_CMD 0x8 Parameter MemoryArea to the state Locked.

Evaluation of System Requirement (SYSRS 007) and Corresponding Software Requirement (SWRS 008)

Condition Check:

- **SYSRS 007** specifies the condition **If Performing Memory Access**, **If the Key is Set**, **If Read Operation is Requested**, and **If Memory Area Is Set Read Accessible**. These conditions are clearly outlined and establish the context in which the action should occur.
- **SWRS 008** mirrors the same conditions exactly: **If Performing Memory Access**, **If the Key is Set**, **If Read Operation is Requested**, and **If Memory Area Is Set Read Accessible**. There is no divergence in the conditions between the system and software requirements, indicating strong alignment in this respect.

Action and Response Check:

- **SYSRS 007** specifies that under these conditions, the **Sensor IC shall set the state of the Memory Area identified by CRM_CMD 0x8 Parameter MemoryArea to the state Locked**.
- **SWRS 008** implements the same action: the **Software shall set the state of the Memory Area identified by CRM_CMD 0x8 Parameter MemoryArea to the state Locked**. The action is identical to the system requirement, and there is no deviation in how the software responds to the condition.

Parameter Sources Check:

- **SYSRS 007** refers to **CRM_CMD 0x8 Parameter MemoryArea** for identifying the memory area that is affected.
- **SWRS 008** also uses **CRM_CMD 0x8 Parameter MemoryArea**, ensuring consistency in how the memory area is referenced. The parameter source is clearly addressed in both the system and software requirements, and no discrepancies are present.

Additional Conditions Check:

- **SWRS 008** does not introduce any additional conditions. It adheres strictly to the conditions outlined in **SYSRS 007** without adding irrelevant or extraneous conditions, ensuring full alignment.

Ratings:

Meaningfulness Rating: High

- **Reason:** The software requirement directly supports the system requirement. It mirrors the conditions and actions outlined in **SYSRS 007** without any deviations. The link between the two is strong, logical, and meaningful.

Completeness Rating: High

- **Reason:** The software requirement fully addresses all aspects of the system requirement, covering all necessary conditions and actions. There is no missing information, and the parameter sources are correctly referenced.

Overall Quality of the Link: High

- **Reason:** The software requirement fully implements the system requirement without any discrepancies. The conditions, actions, and parameters are all correctly aligned, making the overall quality of the link high.

Suggested Revisions:

- No revisions are needed, as **SWRS 008** is fully aligned with **SYSRS 007**. Both requirements are consistent and complete.

"SYSRS 006: If Performing Memory Access in Unicast
AND IF the state of the Memory Area identified by CRM_CMD 0x8 Parameter MemoryArea for the operation as determined by CRM_CMD 0x8 Operation Flag is in the state Locked,
THEN
the Sensor IC shall
respond with CRM_RSP 0x8: Report Lock Status and CRM_RSP 0x8 Flag KeyRequest = 0b1 (""key requested"").
"	"SWRS 007: If Performing Memory Access in Unicast
AND IF the state of the Memory Area identified by CRM_CMD 0x8 Parameter MemoryArea for the operation as determined by CRM_CMD 0x8 Operation Flag is in the state Locked,
THEN
the Software shall prepare to
respond with CRM_RSP 0x8: Report Lock Status and CRM_RSP 0x8 Flag KeyRequest = 0b1 (""key requested"").
"

Evaluation of System Requirement (SYSRS 006) and Corresponding Software Requirement (SWRS 007)

Condition Check:

- **SYSRS 006** specifies the condition **If Performing Memory Access in Unicast** and **IF the state of the Memory Area identified by CRM_CMD 0x8 Parameter MemoryArea for the operation, as determined by CRM_CMD 0x8 Operation Flag, is in the state Locked**. These conditions provide clear criteria under which the action should be executed.
- **SWRS 007** mirrors these conditions exactly: **If Performing Memory Access in Unicast** and **IF the state of the Memory Area identified by CRM_CMD 0x8 Parameter MemoryArea, as determined by CRM_CMD 0x8 Operation Flag, is in the state Locked**. The conditions are fully aligned with **SYSRS 006**, indicating consistency in the implementation.

Action and Response Check:

- **SYSRS 006** states that, under the specified conditions, the **Sensor IC shall respond with CRM_RSP 0x8: Report Lock Status and CRM_RSP 0x8 Flag KeyRequest = 0b1 ("key requested")**. This is a clear instruction that the Sensor IC must take this action.
- **SWRS 007** introduces a slight difference in wording: **the Software shall prepare to respond with CRM_RSP 0x8: Report Lock Status and CRM_RSP 0x8 Flag KeyRequest = 0b1 ("key**

requested"). The key distinction here is the phrase "prepare to respond" rather than actually responding. This introduces ambiguity about whether the response is definitively sent or just prepared for transmission. **SYSRS 006** clearly expects the IC to respond, while **SWRS 007** suggests a preparation stage, which might not ensure that the response is completed.

Parameter Sources Check:

- **SYSRS 006** and **SWRS 007** both refer to **CRM_CMD 0x8 Parameter MemoryArea** and **CRM_CMD 0x8 Operation Flag** to determine the memory area and its locked state.
- Both the system and software requirements correctly handle the parameter sources, referencing the same command and parameters for determining the operation and memory area involved.

Additional Conditions Check:

- **SWRS 007** does not introduce any additional conditions that would alter or expand on the conditions in **SYSRS 006**. It remains fully aligned with the system requirement's conditions without introducing irrelevant or unnecessary details.

Ratings:

Meaningfulness Rating: Medium

- **Reason:** While the software requirement generally supports the system requirement, the phrase "prepare to respond" introduces ambiguity. The system requirement clearly expects an actual response, but the software requirement leaves room for interpretation, potentially weakening the meaningfulness of the link.

Completeness Rating: Medium

- **Reason:** The software requirement captures most of the system requirement's intent, including the correct conditions and parameters. However, the difference between "preparing" and "responding" reduces the completeness since **SYSRS 006** expects a definitive action, while **SWRS 007** stops short of guaranteeing the response is sent.

Overall Quality of the Link: Medium

- **Reason:** Although **SWRS 007** largely reflects **SYSRS 006**, the ambiguity regarding whether the software just prepares to respond or actually responds creates a gap. The conditions and parameters are aligned, but the software requirement falls short in ensuring the full execution of the action.

Suggested Revisions:

1. **SWRS 007** should be updated to state that the software shall **actually respond** rather than merely **prepare to respond**. This would remove the ambiguity and align the software requirement more closely with **SYSRS 006**.
2. If the preparation stage is necessary (e.g., in staged operations), **SWRS 007** should explicitly mention that the response is guaranteed after preparation to ensure that the expected behavior is fulfilled.

"SYSRS 005: IF the Sensor IC generates more information than possible to store in Event Buffer or Time Series Data Buffer
THEN
the Sensor IC shall forget the according information." "SWRS 005: - The software shall store information for up to 5 Echo Events.

SWRS 006: - The software shall hold at least the information for up to 100 time series data samples (time stamp and value)
IF NOT possible regarding memory constraints could be reduced up to 50 time series data samples. "

Evaluation of System Requirement (SYSRS 005) and Corresponding Software Requirements (SWRS 005 & SWRS 006)

Condition Check:

- **SYSRS 005** states that **IF the Sensor IC generates more information than possible to store in the Event Buffer or Time Series Data Buffer**, the system should "forget" the corresponding information. This implies that when the buffer limits are exceeded, the system must discard any excess information.
- **SWRS 005** and **SWRS 006** describe specific storage capacities for different types of data:
  - **SWRS 005** mentions storing up to **5 Echo Events**.
  - **SWRS 006** mentions holding **at least 100 time series data samples** but allows for a reduction to **50 samples** if memory constraints occur.

  While both software requirements introduce constraints for data storage, neither explicitly mirrors the condition in **SYSRS 005**, where the system must forget data when the buffer exceeds capacity. The software requirements talk about memory limits, but there is no explicit mention of handling excess data (i.e., forgetting or discarding it), which is critical in **SYSRS 005**.

Action and Response Check:

- **SYSRS 005** specifies a clear action: when the buffer limit is exceeded, the Sensor IC must **forget the corresponding information** (i.e., discard data).
- **SWRS 005** and **SWRS 006** do not describe any specific action when the buffer is full or overfilled. While they set limits for how much data can be stored, they do not specify what happens when more data is generated than can be stored in the defined buffers. This is a significant omission because the system requirement expects the system to forget or discard data in such cases, but the software requirements do not address this behavior.

Parameter Sources Check:

- **SYSRS 005** refers to two specific data buffers: the **Event Buffer** and the **Time Series Data Buffer**.
- **SWRS 005** indirectly addresses the **Event Buffer** by specifying a storage capacity for **Echo Events** but does not explicitly name the buffer.
- **SWRS 006** addresses the **Time Series Data Buffer**, specifying its capacity, though it also introduces a condition where the capacity can be reduced due to memory constraints. Both

software requirements refer to the relevant data types, but neither addresses what happens when buffer capacities are exceeded, as required in **SYSRS 005**.

Additional Conditions Check:

- **SWRS 006** introduces an additional condition: the buffer for time series data samples can be reduced from 100 samples to 50 if memory constraints require it. While this is a reasonable limitation for handling memory issues, it does not align with **SYSRS 005**, which expects the system to forget data once buffer capacity is exceeded. The reduction condition does not conflict with the system requirement, but it is not a substitute for addressing what happens when memory constraints lead to data loss.

Ratings:

Meaningfulness Rating: Low

- **Reason:** While **SWRS 005** and **SWRS 006** define storage limits, they do not address the core aspect of **SYSRS 005**, which is the need to discard or forget information when buffer capacities are exceeded. This omission makes the link weak, as the software requirements do not fully implement the system requirement's critical action.

Completeness Rating: Low

- **Reason:** The software requirements are incomplete because they do not address the expected behavior when the buffer is full (i.e., forgetting or discarding excess data). While they provide some limits and flexibility around memory constraints, they do not fulfill the system requirement's instruction to discard data when limits are reached.

Overall Quality of the Link: Low

- **Reason:** The software requirements fail to address the key action described in the system requirement. The omission of handling buffer overflow and the absence of any mechanism for discarding excess data results in a weak and incomplete link between the system and software requirements.

Suggested Revisions:

1. **SWRS 005** and **SWRS 006** should be updated to include a mechanism for discarding or "forgetting" data when the respective buffers (Event Buffer and Time Series Data Buffer) are full, as required by **SYSRS 005**.
2. Consider explicitly mentioning what happens when buffer limits are exceeded, especially in cases where more than 5 Echo Events or more than the allowed number of time series data samples are generated.
3. The additional condition in **SWRS 006** about reducing the number of stored samples to 50 is acceptable but should be supplemented with instructions on how to handle data overflow if even the reduced limit is reached.

"SYSRS 004:   If performing Erase Backup
AND If Addressed with the Unicast Command

THEN
the Sensor IC shall
respond with CRM_RSP 0x0: Report Acknowledgement." "SWRS 004: If performing Erase Backup
AND If Addressed with the Unicast Command
THEN
the Software shall prepare to
respond with CRM_RSP 0x0: Report Acknowledgement."

Evaluation of System Requirement (SYSRS 004) and Corresponding Software Requirement (SWRS 004)

Condition Check:

- **SYSRS 004** specifies the conditions under which the action should occur: **If performing Erase Backup** and **If Addressed with the Unicast Command**. These conditions are clear and define the circumstances where the Sensor IC must respond.
- **SWRS 004** mirrors these conditions exactly: **If performing Erase Backup** and **If Addressed with the Unicast Command**. The conditions in the software requirement are identical to those in the system requirement, indicating strong alignment in this aspect.

Action and Response Check:

- **SYSRS 004** states that, under the given conditions, the Sensor IC must **respond with CRM_RSP 0x0: Report Acknowledgement**. This is a clear and specific action that the system expects.
- **SWRS 004** introduces a slight difference: it states that the software shall **prepare to respond** with the same message (CRM_RSP 0x0: Report Acknowledgement). The phrase "prepare to respond" introduces ambiguity, as it implies a preparatory step rather than the actual action of responding. **SYSRS 004** clearly expects the Sensor IC to respond immediately, whereas **SWRS 004** suggests that the response may not be immediate or certain. This difference in wording is significant because it impacts whether the action (the response) is definitively carried out.

Parameter Sources Check:

- Both **SYSRS 004** and **SWRS 004** reference the same response message, **CRM_RSP 0x0: Report Acknowledgement**, and the same conditions involving the **Unicast Command**. There are no discrepancies in how the parameters are used, and the source of the response is correctly identified in both requirements.

Additional Conditions Check:

- **SWRS 004** does not introduce any additional conditions that alter or complicate the ones stated in **SYSRS 004**. It adheres to the same conditions without adding any irrelevant or conflicting details.

Ratings:

Meaningfulness Rating: Medium

- **Reason:** While **SWRS 004** generally aligns with **SYSRS 004**, the phrase **"prepare to respond"** creates ambiguity. The system requirement expects an immediate response, but the software requirement implies a preparatory stage, which reduces the meaningfulness of the link. The intended action of responding is not fully guaranteed by the software requirement.

Completeness Rating: Medium

- **Reason:** The software requirement captures most of the system requirement's intent, but the difference between "preparing" and "responding" reduces completeness. **SYSRS 004** expects the response to occur, while **SWRS 004** only ensures preparation for the response, which is not sufficient to fulfill the system requirement.

Overall Quality of the Link: Medium

- **Reason:** The conditions and response parameters are aligned between the system and software requirements. However, the action described in **SWRS 004** (preparing to respond) falls short of the system requirement's expectation for an immediate response, resulting in a medium overall quality rating.

Suggested Revisions:

1. **SWRS 004** should be updated to state that the software shall **respond**, not just **prepare to respond**, with **CRM_RSP 0x0: Report Acknowledgement**. This would remove the ambiguity and ensure the software requirement fully implements **SYSRS 004**.
2. If the preparatory stage is necessary, it should be clarified that the response will definitely follow after preparation, ensuring the intent of the system requirement is met.

"SYSRS 003: BEFORE Sending CRM_RCC After Reception of the CRM_FCC,
THEN
the Sensor IC shall
set CRM_RSP Parameter CrmStatus = Status Value.
"	"SWRS 003: BEFORE responding with the CRM_RSP,
the Software shall include the CRM_RSP Parameter CrmStatus into the CRM_RSP.

"

Evaluation of System Requirement (SYSRS 003) and Corresponding Software Requirement (SWRS 003)

Condition Check:

- **SYSRS 003** specifies that **BEFORE Sending CRM_RCC After Reception of the CRM_FCC**, the Sensor IC must set the **CRM_RSP Parameter CrmStatus** to a status value. This condition is clear in defining the sequence of events: first, the CRM_FCC is received, then the CRM_RCC is sent, and before this happens, the status value is set in the CRM_RSP parameter.

- **SWRS 003** mirrors this condition closely but changes the timing slightly: **BEFORE responding with the CRM_RSP**, the software must include the **CRM_RSP Parameter CrmStatus**. The focus here is on preparing the CRM_RSP message before it is sent, which aligns conceptually with **SYSRS 003**. The slight difference is that **SWRS 003** does not explicitly mention the sequence involving **CRM_RCC** and **CRM_FCC**, though the general idea of preparing the CRM_RSP before sending it is the same.

Action and Response Check:

- **SYSRS 003** specifies that the **Sensor IC shall set the CRM_RSP Parameter CrmStatus = Status Value**. This action means that the IC must define the status value in the CRM_RSP before sending the CRM_RCC message.
- **SWRS 003** implements a similar action: the software must **include the CRM_RSP Parameter CrmStatus** in the CRM_RSP before sending it. While the wording differs slightly (SWRS 003 refers to "including" rather than "setting"), the intended action is effectively the same: ensuring that the CrmStatus parameter is part of the CRM_RSP before the response is sent.

Parameter Sources Check:

- **SYSRS 003** refers to **CRM_RSP Parameter CrmStatus** and requires that this parameter be set to a status value.
- **SWRS 003** also references the **CRM_RSP Parameter CrmStatus** and requires that it be included in the CRM_RSP. Both the system and software requirements reference the same parameter and handle it in a consistent manner, ensuring alignment regarding the parameter source.

Additional Conditions Check:

- **SWRS 003** does not introduce any additional conditions. It adheres to the sequence and action described in **SYSRS 003** without adding any irrelevant or conflicting conditions.

Ratings:

Meaningfulness Rating: High

- **Reason:** The software requirement aligns well with the system requirement. It effectively captures the essence of **SYSRS 003** by ensuring that the **CrmStatus** parameter is included in the CRM_RSP before the response is sent. While there is a slight difference in wording (setting vs. including), the intent and action are the same, making the link meaningful.

Completeness Rating: High

- **Reason:** The software requirement fully addresses the system requirement. It ensures that the **CrmStatus** parameter is included in the CRM_RSP before the response is sent, fulfilling the system requirement's directive to set the status value before sending the CRM_RCC message. There is no missing information, and the necessary conditions and parameters are fully covered.

Overall Quality of the Link: High

- **Reason:** The software requirement closely aligns with the system requirement in both action and condition. While the wording is slightly different, the meaning and implementation are consistent, resulting in a strong overall link between the two.

Suggested Revisions:

- No revisions are needed, as **SWRS 003** is well-aligned with **SYSRS 003**. Both requirements are consistent and complete in their handling of the CRM_RSP parameter and timing.

SYSRS 002: The Sensor IC FW shall have a watchdog to detect FW or HW malfunction by resetting the WDG time in appropriate time within the FTTI.   SWRS 002: The software shall enable a watchdog to detect software or hardware malfunction AND IF a malfunction is present, perform a reset in an appropriate time within the Fault Tolerant Time Interval of 20ms

Evaluation of System Requirement (SYSRS 002) and Corresponding Software Requirement (SWRS 002)

Condition Check:

- **SYSRS 002** specifies that **the Sensor IC firmware shall have a watchdog to detect firmware or hardware malfunction** by **resetting the watchdog (WDG) time within the appropriate Fault Tolerant Time Interval (FTTI)**. This focuses on ensuring that the watchdog is reset within the designated interval to detect malfunctions.

- **SWRS 002** closely mirrors this by stating that **the software shall enable a watchdog to detect software or hardware malfunction**. However, **SWRS 002** introduces an additional condition: **IF a malfunction is present, perform a reset within the Fault Tolerant Time Interval of 20ms**. This addition includes both detecting malfunctions and performing a reset if a malfunction is detected.

  While **SWRS 002** includes additional steps (resetting in case of a malfunction), the condition of resetting the watchdog timer within the FTTI is aligned with the system requirement. The inclusion of the reset action is a logical extension but introduces more detail than is strictly required by **SYSRS 002**, which only specifies resetting the WDG time.

Action and Response Check:

- **SYSRS 002** focuses on ensuring that the **watchdog is reset within the appropriate time interval (FTTI)** to detect malfunctions. The system requirement does not mention what should happen if a malfunction is detected, focusing only on the regular resetting of the watchdog timer to avoid malfunctions.
- **SWRS 002** extends this by also specifying that **IF a malfunction is detected, a reset should occur within the 20ms FTTI**. This is an additional action not mentioned in the system requirement but adds a layer of response handling in the event of malfunction detection. While this addition provides more comprehensive functionality, it is not strictly required by **SYSRS 002**, which does not mention any specific actions in the case of a malfunction.

Parameter Sources Check:

- **SYSRS 002** references the **FTTI** without specifying a specific duration for this time interval. It simply mentions that the watchdog should be reset within the appropriate time.
- **SWRS 002** provides additional clarity by specifying that the **Fault Tolerant Time Interval is 20ms**. This additional detail enhances understanding of the system behavior, providing a concrete number for the FTTI that is left abstract in **SYSRS 002**.

Additional Conditions Check:

- **SWRS 002** introduces the condition of performing a reset **if a malfunction is present**, which is not mentioned in **SYSRS 002**. This is an additional action that enhances the software's robustness but does not conflict with the system requirement. It could be considered a natural extension of what would happen if a malfunction is detected, though it is not explicitly required by the system requirement.

Ratings:

Meaningfulness Rating: Medium

- **Reason:** The software requirement aligns with the system requirement in terms of enabling the watchdog and resetting it within the Fault Tolerant Time Interval (FTTI). However, the introduction of the additional action (performing a reset upon detecting a malfunction) is not mentioned in **SYSRS 002**. While this extra step is useful, it adds more functionality than what is strictly required by the system requirement, creating a partial misalignment.

Completeness Rating: Medium

- **Reason:** While **SWRS 002** covers the core function of enabling the watchdog and resetting it within the FTTI, it introduces additional functionality (resetting in case of malfunction) that is not covered in **SYSRS 002**. Additionally, **SWRS 002** provides a specific time interval (20ms) for the FTTI, which adds clarity, but this additional information is not included in the system requirement. Therefore, **SWRS 002** exceeds what is necessary for completeness, which could lead to confusion about whether the system requirement intends for this extra functionality to be present.

Overall Quality of the Link: Medium

- **Reason:** While **SWRS 002** addresses the core function described in **SYSRS 002**, the introduction of extra actions (such as resetting the system when a malfunction is detected) creates some misalignment. The software requirement goes beyond the scope of the system requirement, making the link between the two only moderately clear.

Suggested Revisions:

1. **SWRS 002** could be revised to explicitly match **SYSRS 002** by focusing on **resetting the watchdog** within the FTTI rather than introducing the additional action of performing a reset in case of malfunction. This would create a more direct alignment.
2. If the extra functionality (performing a reset upon malfunction detection) is required, **SYSRS 002** should be updated to include this behavior so that the system requirement and software requirement are fully aligned in terms of both detection and response.

SYSRS 001: The Sensor IC shall digitize the Transducer Input Signal at the IC Pins with sampling frequency of (CALIB_samplingFreq plus YPX_samplingFreqOffset).   "SWRS 001: AFTER reception of the command CRM_CMD 0x1: Start Standard Measurement
AND
BEFORE the start of the Measurement the software shall set the register BRG.FS.f_s to (CALIB_samplingFreq plus YPX_samplingFreqOffset). "

Evaluation of System Requirement (SYSRS 001) and Corresponding Software Requirement (SWRS 001)

Condition Check:

- **SYSRS 001** specifies that the **Sensor IC shall digitize the Transducer Input Signal at the IC Pins with a sampling frequency of (CALIB_samplingFreq plus YPX_samplingFreqOffset)**. The condition in the system requirement does not specify an event trigger but only focuses on the digitization process with the correct sampling frequency.
- **SWRS 001** adds specific timing conditions: **AFTER reception of the command CRM_CMD 0x1: Start Standard Measurement** and **BEFORE the start of the Measurement**, the software shall set the register for the sampling frequency. These conditions provide more context than **SYSRS 001**, linking the action to the initiation of a measurement sequence and setting the frequency before the measurement starts. This additional context is logical and aligns with the overall process but introduces conditions that are not explicitly mentioned in **SYSRS 001**.

Action and Response Check:

- **SYSRS 001** focuses on the action of **digitizing the Transducer Input Signal** with the specified sampling frequency. This is the main action required by the system.
- **SWRS 001** implements the action of **setting the register BRG.FS.f_s** to the specified sampling frequency **before the measurement starts**. This action ensures that the correct sampling frequency is used during the digitization process. While **SWRS 001** does not explicitly mention digitizing the input signal, setting the correct frequency before the measurement effectively supports the intent of **SYSRS 001**. Therefore, the software requirement addresses the necessary setup for digitization without directly repeating the system-level action of digitization.

Parameter Sources Check:

- **SYSRS 001** refers to the parameters **CALIB_samplingFreq** and **YPX_samplingFreqOffset** as components of the final sampling frequency.
- **SWRS 001** correctly uses the same parameters, **CALIB_samplingFreq** and **YPX_samplingFreqOffset**, to set the sampling frequency in the register. This ensures consistency between the system and software requirements, as the correct parameters are applied to the same calculation.

Additional Conditions Check:

- **SWRS 001** introduces the additional conditions of performing the action **AFTER reception of CRM_CMD 0x1** and **BEFORE the start of the measurement**, which are not mentioned in **SYSRS**

001. These conditions do not conflict with the system requirement but add more context to when the action takes place. If the system requirement's intent is for the digitization to occur during measurements, then these additional conditions are reasonable and useful.

Ratings:

Meaningfulness Rating: High

- **Reason:** The software requirement supports the system requirement by ensuring the correct sampling frequency is set before the measurement starts. While **SWRS 001** introduces more specific timing conditions, they do not conflict with the system requirement and provide additional context for proper operation. The link between the two is logical and meaningful.

Completeness Rating: Medium

- **Reason: SWRS 001** focuses on setting the sampling frequency in preparation for the measurement but does not explicitly mention the digitization process, which is central to **SYSRS 001**. While the software requirement ensures that the necessary setup is completed, it stops short of confirming that the digitization of the input signal occurs. This reduces the completeness slightly, as the full action expected in the system requirement is not fully detailed.

Overall Quality of the Link: High

- **Reason:** Despite the slight gap in explicitly mentioning digitization, the software requirement effectively implements the necessary setup for ensuring the correct sampling frequency, making the overall quality of the link high. The additional conditions provide useful context without detracting from the system-level requirement.

Suggested Revisions:

1. **SWRS 001** could be revised to explicitly state that the **digitization of the transducer input signal** will occur using the set sampling frequency. This would ensure that the software requirement fully addresses the action expected in **SYSRS 001**.
2. If necessary, **SYSRS 001** could be expanded to include the conditions **after receiving the CRM_CMD 0x1** and **before starting the measurement**, to better align with the software requirement's detailed timing sequence.

"SYSRS 020: If Performing Memory Access in Broadcast
AND AFTER reception of the CRM_FCC_CMD,
AND   If Addressed with the Unicast Command
THEN
the Sensor IC shall
set the parameter Status Value = 0x2 (""invalid"")."   "SWRS 022: If Performing Memory Access in Broadcast
AND AFTER reception of the CRM_FCC_CMD,
AND If Addressed with the Broadcast Command
THEN

the Software shall
set the parameter Status Value = 0x2 (""invalid"")."

Evaluation of System Requirement (SYSRS 020) and Corresponding Software Requirement (SWRS 022)

Condition Check:

- **SYSRS 020** specifies the following conditions:
    - **If Performing Memory Access in Broadcast**
    - **AFTER reception of the CRM_FCC_CMD**
    - **If Addressed with the Unicast Command** (i.e., the memory access is directed to a single entity)

- **SWRS 022** mirrors the first two conditions:
    - **If Performing Memory Access in Broadcast**
    - **AFTER reception of the CRM_FCC_CMD**

  However, there is a critical difference in the third condition:

    - **SWRS 022** specifies **If Addressed with the Broadcast Command**, which directly contradicts the **Unicast Command** condition in **SYSRS 020**. This creates a significant misalignment in the type of command that triggers the action.

Action and Response Check:

- **SYSRS 020** specifies that under the given conditions, the **Sensor IC shall set the parameter Status Value = 0x2 ("invalid")**.
- **SWRS 022** correctly implements the same action: **the software shall set the parameter Status Value = 0x2 ("invalid")**. The action itself is aligned between the system and software requirements, with both requiring the same status value to be set.

Parameter Sources Check:

- Both **SYSRS 020** and **SWRS 022** refer to the **Status Value** parameter, and both set it to **0x2 ("invalid")**. There is no discrepancy in the parameter usage, and the source is handled consistently across both requirements.

Additional Conditions Check:

- **SWRS 022** introduces no additional conditions that would further complicate the situation. However, the shift from **Unicast Command** to **Broadcast Command** represents a significant condition change, leading to a misalignment.

Ratings:

Meaningfulness Rating: Low

- **Reason:** The critical difference in the condition related to the type of command (Unicast vs. Broadcast) makes the software requirement poorly aligned with the system requirement. **SYSRS 020** explicitly requires the action to occur when addressed with a **Unicast Command**, while **SWRS 022** performs the same action under a **Broadcast Command**. This difference weakens the link, making the software requirement not meaningfully implement the system requirement.

Completeness Rating: Medium

- **Reason:** While the action of setting the **Status Value = 0x2** is correctly implemented in **SWRS 022**, the conditions for when this action takes place are not aligned with **SYSRS 020**. The software requirement is incomplete because it triggers the action under a different scenario (Broadcast instead of Unicast).

Overall Quality of the Link: Low

- **Reason:** The difference in the command type (Unicast vs. Broadcast) leads to a significant misalignment. Even though the core action of setting the status value is consistent, the conditions for triggering this action are different, resulting in a low overall quality rating.

Suggested Revisions:

1. **SWRS 022** should be revised to trigger the action when the system is addressed with a **Unicast Command**, aligning it with **SYSRS 020**. This would bring the conditions in line and ensure proper implementation of the system requirement.
2. If the **Broadcast Command** condition is also necessary for some scenarios, a separate software requirement should be created to handle the behavior under **Broadcast Command** conditions, ensuring both scenarios are covered without conflict.

"SYSRS 019: If Performing Memory Access in Unicast
AND AFTER reception of the CRM_FCC_CMD,
AND If Addressed with the Broadcast Command
THEN
the Sensor IC shall
set the parameter Status Value = 0x2 (""invalid"")."   "SWRS 021: If Performing Memory Access in Unicast
AND AFTER reception of the CRM_FCC_CMD,
AND If Addressed with the Broadcast Command
THEN
the Software shall
set the parameter Status Value = 0x2 (""invalid"")."

Evaluation of System Requirement (SYSRS 019) and Corresponding Software Requirement (SWRS 021)

Condition Check:

- **SYSRS 019** specifies the following conditions:
  - **If Performing Memory Access in Unicast**
  - **AFTER reception of the CRM_FCC_CMD**
  - **If Addressed with the Broadcast Command**

  These conditions create a specific scenario where the memory access is unicast, but the system is addressed with a broadcast command after receiving the CRM_FCC_CMD.

- **SWRS 021** mirrors these conditions exactly:
  - **If Performing Memory Access in Unicast**
  - **AFTER reception of the CRM_FCC_CMD**
  - **If Addressed with the Broadcast Command**

  There is no difference in the conditions between the system and software requirements, indicating strong alignment in this aspect.

Action and Response Check:

- **SYSRS 019** specifies that under the given conditions, the **Sensor IC shall set the parameter Status Value = 0x2 ("invalid")**.

- **SWRS 021** implements the same action: **the software shall set the parameter Status Value = 0x2 ("invalid")**.

  The action is identical between the system and software requirements, with both requiring that the **Status Value** is set to **0x2 ("invalid")** under the specified conditions.

Parameter Sources Check:

- Both **SYSRS 019** and **SWRS 021** refer to the **Status Value** parameter, and both set it to **0x2 ("invalid")**. There is no discrepancy in the parameter usage, and the source is handled consistently across both requirements.

Additional Conditions Check:

- **SWRS 021** introduces no additional conditions beyond those mentioned in **SYSRS 019**. The software requirement remains aligned with the system requirement in terms of when the action should be triggered.

Ratings:

Meaningfulness Rating: High

- **Reason:** The software requirement fully supports the system requirement by implementing the same conditions and actions. There is no ambiguity, and the link between the system and software requirements is clear and logical. The conditions and actions directly match, making the link meaningful.

Completeness Rating: High

- **Reason:** The software requirement covers all aspects of the system requirement. It addresses all the conditions (memory access, reception of CRM_FCC_CMD, and addressing via broadcast command) and specifies the correct action to be taken. There is no missing information or critical detail.

Overall Quality of the Link: High

- **Reason:** The software requirement is fully aligned with the system requirement. The conditions and actions are identical, making the overall quality of the link strong. The software requirement correctly implements the behavior expected by the system requirement.

Suggested Revisions:

- No revisions are needed, as **SWRS 021** is fully aligned with **SYSRS 019**. Both requirements are consistent and complete, ensuring that the correct actions are taken under the specified conditions.

"SYSRS 018: - Unlocked, If No Key Is Set
If Performing Memory Access
AND If the Key is NOT set
THEN
the Sensor IC shall
set the state of the Memory Area identified by CRM_CMD 0x8 Parameter MemoryArea to the state Memory Area State: Unlocked." "SWRS 020: - locked, If Key Is Set
If Performing Memory Access
AND If the Key is set
THEN
the Software shall
set the state of the Memory Area identified by CRM_CMD 0x8 Parameter MemoryArea to the state Memory Area State: locked."

Evaluation of System Requirement (SYSRS 018) and Corresponding Software Requirement (SWRS 020)

Condition Check:

- **SYSRS 018** specifies that **If Performing Memory Access** and **If the Key is NOT set**, the **Sensor IC shall set the state of the Memory Area identified by CRM_CMD 0x8 Parameter MemoryArea to the state: Unlocked**. This system requirement describes the condition where no key is set, and as a result, the memory area should be unlocked.
- **SWRS 020** specifies a different condition: **If Performing Memory Access** and **If the Key is set**, the **software shall set the state of the Memory Area identified by CRM_CMD 0x8 Parameter MemoryArea to the state: Locked**. This software requirement focuses on the opposite condition, where the key is set, and the memory area should be locked.

The conditions in **SYSRS 018** and **SWRS 020** are completely different:

  - **SYSRS 018** deals with the scenario when **the key is NOT set** and requires the memory area to be **unlocked**.
  - **SWRS 020** deals with the scenario when **the key is set** and requires the memory area to be **locked**.

These are opposing conditions, and therefore, there is no alignment between the two in terms of conditions.

Action and Response Check:

- **SYSRS 018** specifies that the **Sensor IC** should set the memory area state to **Unlocked** when the key is **NOT set**.

- **SWRS 020** specifies that the **software** should set the memory area state to **Locked** when the key **is set**.

  The actions required in these requirements are opposite:

  - **SYSRS 018** requires the memory area to be **unlocked**.
  - **SWRS 020** requires the memory area to be **locked**.

  These actions are contradictory, and **SWRS 020** does not address the condition described in **SYSRS 018** (key not set, resulting in an unlocked state).

Parameter Sources Check:

- Both **SYSRS 018** and **SWRS 020** refer to the **CRM_CMD 0x8 Parameter MemoryArea** to identify the memory area whose state is being modified. This aspect is consistent, but since the conditions and actions are opposite, the parameter handling is not sufficient to make these requirements aligned.

Additional Conditions Check:

- **SWRS 020** introduces no additional conditions that would align it with **SYSRS 018**. It focuses entirely on the scenario where the key is set, whereas **SYSRS 018** addresses the scenario where the key is not set.

Ratings:

Meaningfulness Rating: Low

- **Reason:** The conditions and actions in **SWRS 020** do not align with **SYSRS 018**. The system requirement focuses on unlocking the memory area when the key is not set, while the software requirement focuses on locking the memory area when the key is set. These are opposite scenarios, making the link weak and not meaningful.

Completeness Rating: Low

- **Reason: SWRS 020** does not address the key condition described in **SYSRS 018**. The system requirement requires the memory area to be unlocked when no key is set, but the software requirement covers a different scenario (locking when the key is set). This leaves the condition and action described in **SYSRS 018** unaddressed.

Overall Quality of the Link: Low

- **Reason:** The software requirement does not implement the behavior expected by the system requirement. The two requirements describe opposing conditions and actions, which significantly weakens the link. As a result, the overall quality of the link is low.

Suggested Revisions:

1. **SWRS 020** should be revised to include the condition where **the key is NOT set** and ensure that the memory area is set to **Unlocked** when this condition is met, as described in **SYSRS 018**.
2. If both conditions (key set and key not set) need to be addressed, it would be useful to have two separate software requirements: one handling the **key not set** condition (unlocking the memory area) and one handling the **key set** condition (locking the memory area). This would ensure that both scenarios are covered appropriately.

"SYSRS 017: If Performing the Standard Path Measurement
AND IF performing Stop Ongoing Measurement (Blocked)
AND IF the functionality in US-Transducer Control (Non-Modulated Sine Burst) has NOT been fully performed,
THEN
the Sensor IC shall
set Error: err_burstGenerationStopped = 0x1 (""error"")."   "SWRS 019: IF in Main-Measurement
AND IF Break Burst Generation occurs
THEN
the software shall set
Error: err_burstGenerationStopped = 0x1 (""error"")."

Evaluation of System Requirement (SYSRS 017) and Corresponding Software Requirement (SWRS 019)

Condition Check:

- **SYSRS 017** specifies the following conditions:
  - **If Performing the Standard Path Measurement**
  - **AND IF performing Stop Ongoing Measurement (Blocked)**
  - **AND IF the functionality in US-Transducer Control (Non-Modulated Sine Burst) has NOT been fully performed**

  These conditions create a scenario where the ongoing measurement is stopped before completing a specific functionality (the non-modulated sine burst), leading to an error being set.

- **SWRS 019** specifies different but related conditions:

  - **IF in Main-Measurement**
  - **AND IF Break Burst Generation occurs**

  SWRS 019 condenses the conditions, specifically focusing on a situation where **break burst generation** occurs during a main measurement, without explicitly mentioning the specific circumstances (e.g., Stop Ongoing Measurement or the incomplete performance of US-Transducer Control).

  The conditions in **SWRS 019** are simplified, and key parts of **SYSRS 017**—like the incomplete sine burst functionality and stopping the ongoing measurement—are missing. While both refer to burst generation being interrupted, **SWRS 019** does not fully reflect the detailed conditions outlined in **SYSRS 017**.

Action and Response Check:

- **SYSRS 017** specifies that under the given conditions, the **Sensor IC shall set Error: err_burstGenerationStopped = 0x1 ("error")**.

- **SWRS 019** correctly implements the same action: **the software shall set Error: err_burstGenerationStopped = 0x1 ("error")** when the break burst generation occurs.

  The action of setting the error is aligned, but **SWRS 019** lacks clarity regarding when the action is triggered due to the simplified condition handling.

Parameter Sources Check:

- Both **SYSRS 017** and **SWRS 019** use the same parameter, **err_burstGenerationStopped**, and correctly set its value to **0x1 ("error")** when the conditions are met. The parameter handling is consistent, but the missing conditions in **SWRS 019** affect the clarity of when this error should be set.

Additional Conditions Check:

- **SWRS 019** introduces no additional conditions but simplifies the ones presented in **SYSRS 017**. Key details, like performing a stop on the ongoing measurement and the incomplete sine burst, are missing from the software requirement.

Ratings:

Meaningfulness Rating: Medium

- **Reason:** While **SWRS 019** addresses a related scenario (burst generation being interrupted), it simplifies the conditions significantly, omitting important parts of the system requirement (like the stop measurement condition and the incomplete sine burst). This makes the software requirement only partially aligned with the system requirement, reducing its meaningfulness.

Completeness Rating: Medium

- **Reason: SWRS 019** captures the key action of setting the error, but it lacks important conditions mentioned in **SYSRS 017**. The missing details—such as stopping the ongoing measurement and the incomplete US-Transducer Control functionality—mean that the software requirement does not fully implement the system requirement.

Overall Quality of the Link: Medium

- **Reason:** The overall quality is reduced by the simplification of the conditions in **SWRS 019**. While the action of setting the error is consistent, the lack of detailed condition handling results in a moderate link between the system and software requirements.

Suggested Revisions:

1. **SWRS 019** should be revised to include more detailed conditions, specifically:
   - The scenario where **Stop Ongoing Measurement** occurs.
   - The condition where the **US-Transducer Control (Non-Modulated Sine Burst)** has **NOT been fully performed**.

   These changes would ensure that **SWRS 019** fully aligns with the detailed conditions described in **SYSRS 017**.
2. Consider clearly differentiating between the types of measurements (Standard Path vs. Main-Measurement) to ensure that the software requirement properly reflects the scope and context of the system requirement.

"SYSRS 016: After Reception of the Valid CRM_FCC_CMD,
THEN
the Sensor IC shall
set the PDCM_RSP Parameter KAC = 0x0."   "SWRS 018:   Increment KAC
AFTER Sending PDCM Packet After Reception of the BRC
THEN
the Software shall
increment the PDCM_RSP Parameter KAC by 1;

After Reception of the Valid CRM_FCC_CMD,
THEN
the Software shall
set the PDCM_RSP Parameter KAC = 0x3."

Evaluation of System Requirement (SYSRS 016) and Corresponding Software Requirement (SWRS 018)

Condition Check:

- **SYSRS 016** specifies that **After Reception of the Valid CRM_FCC_CMD**, the **Sensor IC shall set the PDCM_RSP Parameter KAC = 0x0**. This indicates that once a valid command is received, the KAC parameter must be reset to 0.

- **SWRS 018** introduces two distinct conditions:

1. **After Sending PDCM Packet After Reception of the BRC**, the software shall increment the KAC parameter by 1.
2. **After Reception of the Valid CRM_FCC_CMD**, the software shall set the KAC parameter to **0x3**.

There is a significant difference between the two requirements:

- **SYSRS 016** specifies setting the KAC parameter to **0x0** after receiving the valid CRM_FCC_CMD.
- **SWRS 018** instead specifies setting the KAC parameter to **0x3** under the same condition (after reception of the valid CRM_FCC_CMD), which directly contradicts **SYSRS 016**.

Action and Response Check:

- **SYSRS 016** requires the **Sensor IC to set the KAC parameter to 0x0** after receiving a valid CRM_FCC_CMD, which implies a reset or initialization of the parameter.

- **SWRS 018** specifies two actions:
  1. Incrementing the KAC parameter after sending the PDCM packet following the reception of the BRC. This action is not mentioned in **SYSRS 016**, and it appears to be an additional condition unrelated to the system requirement.
  2. Setting the KAC parameter to **0x3** after receiving the valid CRM_FCC_CMD, which conflicts with the system requirement's instruction to set it to **0x0**.

  The actions in **SWRS 018** diverge significantly from **SYSRS 016** in both the value assigned to the KAC parameter and the introduction of the unrelated condition about incrementing KAC after sending the PDCM packet.

Parameter Sources Check:

- Both **SYSRS 016** and **SWRS 018** refer to the **PDCM_RSP Parameter KAC**, but they differ in how this parameter is handled:
  - **SYSRS 016** requires the parameter to be reset to **0x0**.
  - **SWRS 018** refers to incrementing the parameter and setting it to **0x3** under different conditions, diverging from the expected behavior.

Additional Conditions Check:

- **SWRS 018** introduces an additional condition (incrementing KAC after sending the PDCM packet following the BRC), which is not mentioned in **SYSRS 016**. This additional condition is not relevant to the system requirement and introduces unnecessary complexity.

Ratings:

Meaningfulness Rating: Low

- **Reason: SWRS 018** does not meaningfully align with **SYSRS 016**. The core condition of setting the KAC parameter to **0x0** after receiving the CRM_FCC_CMD is contradicted by the software requirement, which sets it to **0x3** instead. Additionally, the introduction of unrelated conditions (incrementing KAC after sending the PDCM packet) weakens the link.

Completeness Rating: Low

- **Reason: SWRS 018** fails to implement the primary action of **SYSRS 016** (setting KAC to 0x0). Instead, it introduces different actions and conditions that are not part of the system requirement. This results in an incomplete implementation of the system's expected behavior.

Overall Quality of the Link: Low

- **Reason:** The overall link between the system and software requirements is poor. The software requirement contradicts the system requirement by setting the wrong value for the KAC parameter and introduces additional, unrelated conditions. The core functionality described in **SYSRS 016** is not implemented in **SWRS 018**.

Suggested Revisions:

1. **SWRS 018** should be revised to ensure that the software **sets the KAC parameter to 0x0** after receiving a valid CRM_FCC_CMD, as specified in **SYSRS 016**. This would align the software behavior with the system requirement.
2. If the condition of incrementing KAC after sending the PDCM packet is necessary for other functionality, it should be separated into its own software requirement to avoid confusion and ensure that the core behavior of **SYSRS 016** is properly addressed.

"SYSRS 015: Increment KAC
AFTER Sending PDCM Packet After Reception of the BRC
THEN
the Sensor IC shall
increment the PDCM_RSP Parameter KAC by 1; the counter is reset to 0 at overflow" "SWRS 017: Increment KAC
AFTER Sending PDCM Packet After Reception of the BRC
THEN
the Software shall
increment the PDCM_RSP Parameter KAC by 1; "

Evaluation of System Requirement (SYSRS 015) and Corresponding Software Requirement (SWRS 017)

Condition Check:

- **SYSRS 015** specifies that **after sending a PDCM packet after reception of the BRC**, the **Sensor IC shall increment the PDCM_RSP Parameter KAC by 1**. Additionally, it states that the **counter is reset to 0 at overflow**.

- **SWRS 017** matches the initial condition of **after sending a PDCM packet after reception of the BRC**, and similarly, it requires the software to **increment the PDCM_RSP Parameter KAC by 1**.

However, **SWRS 017** does not mention the overflow condition or resetting the counter to 0 at overflow.

The primary condition of when to increment the KAC parameter is aligned between the system and software requirements. However, the overflow reset mentioned in **SYSRS 015** is missing in **SWRS 017**, which creates a slight misalignment regarding how the counter behaves after an overflow.

Action and Response Check:

- **SYSRS 015** requires the **Sensor IC to increment the KAC parameter by 1** after sending the PDCM packet and specifies an additional behavior: **the counter is reset to 0 at overflow**.
- **SWRS 017** similarly requires the software to **increment the KAC parameter by 1** but does not specify the behavior related to overflow. The lack of an overflow reset condition in **SWRS 017** means that the software may not handle the KAC counter in the same way as the system requirement.

  While both requirements handle the incrementing of the KAC parameter, **SWRS 017** is incomplete because it omits the reset condition at overflow, which is an important part of the behavior described in **SYSRS 015**.

Parameter Sources Check:

- Both **SYSRS 015** and **SWRS 017** refer to the **PDCM_RSP Parameter KAC**, ensuring consistency in the parameter being modified. There are no discrepancies in the parameter usage between the system and software requirements.

Additional Conditions Check:

- **SWRS 017** does not introduce any additional conditions but omits the important condition regarding resetting the counter at overflow, as stated in **SYSRS 015**. This is a key aspect missing from the software requirement, as the behavior of the counter could be different without the overflow condition.

Ratings:

Meaningfulness Rating: Medium

- **Reason:** The software requirement mostly aligns with the system requirement in terms of when to increment the KAC parameter. However, the omission of the **overflow reset condition** means that **SWRS 017** does not fully implement the behavior expected by **SYSRS 015**. This weakens the meaningfulness of the link.

Completeness Rating: Medium

- **Reason:** While the core action of incrementing the KAC parameter is correctly captured, **SWRS 017** is incomplete because it does not handle the reset to 0 at overflow. This missing behavior is critical for the proper functioning of the counter, reducing the completeness of the software requirement.

Overall Quality of the Link: Medium

- **Reason:** The software requirement implements the core functionality (incrementing the KAC parameter), but the omission of the overflow reset behavior creates a gap in how fully the software addresses the system requirement. Therefore, the overall quality of the link is medium.

Suggested Revisions:

1. **SWRS 017** should be updated to include the condition that **the KAC counter is reset to 0 at overflow**, as specified in **SYSRS 015**. This will ensure that the software correctly handles both the increment and the overflow behavior.
2. Clarify how the software manages the overflow to ensure it behaves identically to the system-level expectation, particularly in scenarios where the counter exceeds its limit.

"SYSRS 014: IF the Event_MAX is generated,
THEN
the Sensor IC shall
measure the time of the occurrence of the event condition relative to the start of the measurement and provide PDCM_RSP P12, P13 Parameter EventTimeStamp<i> for the <i>th Event transmitted in a DSI3 packet." SWRS 016: The software shall send the MAX events with the measured time of the occurrence of the event, condition relative to the start of the measurement, and provide the PDCM_RSP P12,P13 Parameter EventTimeStamp<i> for the <i>th Event transmitted in a PDCM frame.

Evaluation of System Requirement (SYSRS 014) and Corresponding Software Requirement (SWRS 016)

Condition Check:

- **SYSRS 014** specifies the condition: **IF the Event_MAX is generated**. This triggers the measurement of the event's occurrence time relative to the start of the measurement, followed by the provision of the event timestamp in the form of **PDCM_RSP P12, P13 Parameter EventTimeStamp<i>** for the corresponding event.

- **SWRS 016** mirrors this condition well, referring to sending **MAX events** with the measured time relative to the start of the measurement, followed by the provision of the event timestamp (**PDCM_RSP P12, P13 Parameter EventTimeStamp<i>**).

  The conditions are well aligned. Both requirements specify that the event's time is measured relative to the start of the measurement and that the timestamp is transmitted using the same parameters. The only minor difference is that **SWRS 016** uses the term "PDCM frame" instead of "DSI3 packet," but this does not affect the overall meaning, as both are part of the same communication process.

Action and Response Check:

- **SYSRS 014** specifies that the **Sensor IC shall measure the time of the event's occurrence** and provide the timestamp for the event via **PDCM_RSP P12, P13**.
- **SWRS 016** describes the same action: the software **sends the MAX events with the measured time** and provides the timestamp via **PDCM_RSP P12, P13**.

  The action described in both requirements is identical. The system requirement focuses on the Sensor IC performing the measurement, while the software requirement specifies the action of sending the event data with the timestamp. Both requirements align perfectly in terms of what needs to happen when the event is generated.

Parameter Sources Check:

- Both **SYSRS 014** and **SWRS 016** refer to the **PDCM_RSP P12, P13 Parameter EventTimeStamp<i>** for the timestamp. The same parameters are referenced, and the software correctly implements the system's need to transmit this timestamp for the event.

Additional Conditions Check:

- **SWRS 016** does not introduce any additional conditions or actions that would complicate or deviate from the core behavior described in **SYSRS 014**. It adheres to the same action of measuring and transmitting the event timestamp.

Ratings:

Meaningfulness Rating: High

- **Reason:** The software requirement aligns well with the system requirement. The condition of the event generation, measurement, and the provision of the timestamp is fully captured in both requirements, and the minor terminology difference between **DSI3 packet** and **PDCM frame** does not impact the meaning or intent.

Completeness Rating: High

- **Reason: SWRS 016** fully implements the action described in **SYSRS 014**. The measurement of the event time, relative to the start of the measurement, and the provision of the timestamp via the correct parameters are all included, ensuring that no critical aspects are missing.

Overall Quality of the Link: High

- **Reason:** The software requirement mirrors the system requirement in both condition and action. The functionality is implemented without missing any details, and the small difference in terminology is not significant enough to impact the quality of the link.

Suggested Revisions:

- No revisions are needed, as **SWRS 016** is fully aligned with **SYSRS 014**. Both requirements are consistent and complete in handling the event timestamp process. If there is a need for clearer terminology consistency (e.g., **DSI3 packet** vs. **PDCM frame**), it can be clarified, but it does not affect the overall alignment.

SYSRS 013: The Sensor IC shall amplify the Transducer Input Signal at the IC Pins with the three amplification factors (Parameter CALIB_analogGain plus Parameter SPX_dynGainStartValue plus CRM_CMD 0x4 Parameter CalibrationGain plus Parameter YPX_calibrationGainOffset). "SWRS 013: Analog Amplification: AFTER reception of the command CRM_CMD 0x4: Start Calibration Measurement the software shall set the register the register ASP_CFG.g_ana to Parameter CALIB_analogGain .

SWRS 014: Digital Amplification: AFTER reception of the command CRM_CMD 0x4: Start Calibration Measurement the software shall set the register ENVP_AMPD_CFG.g_dig to Parameter SPX_dynGainStartValue .

SWRS 015: Calibration Gain: AFTER reception of the command CRM_CMD 0x4: Start Calibration Measurement
the software shall set the register ENVP_GCAL.g_cal to (CRM_CMD 0x4 Parameter CalibrationGain + Parameter YPX_calibrationGainOffset).
"

Evaluation of System Requirement (SYSRS 013) and Corresponding Software Requirements (SWRS 013, SWRS 014, SWRS 015)

Condition Check:

- **SYSRS 013** specifies that the **Sensor IC shall amplify the Transducer Input Signal** using three amplification factors:
  - **Parameter CALIB_analogGain**
  - **Parameter SPX_dynGainStartValue**
  - **CRM_CMD 0x4 Parameter CalibrationGain plus Parameter YPX_calibrationGainOffset**

  This condition requires that the amplification process includes all three factors.

- **SWRS 013**, **SWRS 014**, and **SWRS 015** collectively address these factors:
  - **SWRS 013** handles **CALIB_analogGain** by setting the register **ASP_CFG.g_ana** after the reception of **CRM_CMD 0x4: Start Calibration Measurement**.
  - **SWRS 014** addresses **SPX_dynGainStartValue** by setting the register **ENVP_AMPD_CFG.g_dig** after the reception of **CRM_CMD 0x4**.
  - **SWRS 015** addresses the **CalibrationGain** and **YPX_calibrationGainOffset** by setting the register **ENVP_GCAL.g_cal** to the sum of **CRM_CMD 0x4 Parameter CalibrationGain + Parameter YPX_calibrationGainOffset** after the reception of **CRM_CMD 0x4**.

  All three software requirements reference **CRM_CMD 0x4** and execute actions based on the command. However, while each SWRS addresses one aspect of the amplification process, they are broken into separate components rather than being unified, as described in **SYSRS 013**.

Action and Response Check:

- **SYSRS 013** specifies that all three amplification factors should be applied together to amplify the transducer input signal.

- **SWRS 013**, **SWRS 014**, and **SWRS 015** each handle setting the respective amplification factors to different registers:
    - **SWRS 013** sets **CALIB_analogGain** to **ASP_CFG.g_ana**.
    - **SWRS 014** sets **SPX_dynGainStartValue** to **ENVP_AMPD_CFG.g_dig**.
    - **SWRS 015** sets the sum of **CRM_CMD 0x4 Parameter CalibrationGain + YPX_calibrationGainOffset** to **ENVP_GCAL.g_cal**.

  Each software requirement implements the respective component of the system requirement, but **SYSRS 013** implies that the amplification is a combined process. The software requirements handle these factors separately, which may cause some ambiguity about whether the amplification is applied as a unified step.

Parameter Sources Check:

- **SYSRS 013** lists the parameters:
    - **CALIB_analogGain**
    - **SPX_dynGainStartValue**
    - **CRM_CMD 0x4 Parameter CalibrationGain**
    - **YPX_calibrationGainOffset**

  These parameters are all correctly referenced in the corresponding software requirements. However, **SYSRS 013** implies that they should be applied together, while the software requirements address them in isolation.

Additional Conditions Check:

- All three software requirements (**SWRS 013**, **SWRS 014**, and **SWRS 015**) introduce no additional conditions beyond the start of calibration (**CRM_CMD 0x4: Start Calibration Measurement**). This aligns with **SYSRS 013**, where the amplification process is also tied to calibration. However, **SYSRS 013** suggests a single unified process for applying the gains, while the software requirements split the process into separate steps.

Ratings:

Meaningfulness Rating: Medium

- **Reason:** While the software requirements each handle a portion of the amplification process, they are presented in isolation, which slightly diverges from **SYSRS 013**'s implication of a unified process. The software requirements address the correct actions but do so in a fragmented way, reducing the overall clarity of how the amplification is applied as a whole.

Completeness Rating: Medium

- **Reason:** The software requirements cover all three amplification factors mentioned in **SYSRS 013**, but they handle them as separate actions. The system requirement implies a more cohesive process, and the separation of these steps in the software requirements might lead to confusion about whether the amplification is applied correctly in a unified manner. Additionally, there's no mention in **SWRS 013–015** about how these amplifications interact or combine.

Overall Quality of the Link: Medium

- **Reason:** The software requirements correctly implement the individual amplification factors, but the lack of integration or clarity about how the factors combine in a unified process weakens the overall link. The system requirement's intent of amplifying the transducer input signal using multiple factors as a single action isn't fully reflected in the software requirements.

Suggested Revisions:

1. Consider revising the software requirements to indicate that these three amplification factors are applied together as part of a unified amplification process, ensuring alignment with **SYSRS 013**.
2. Clarify how the software handles the combination of these amplification factors, either by creating a single software requirement that unifies the three steps or by explaining how the separate steps lead to a cohesive amplification process.
3. Ensure that the software documentation makes clear how these different gains are applied in sequence or combination during the calibration measurement to avoid ambiguity.