

Run #2



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 oversight. 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.



Sure, I'll perform the evaluation based on the criteria you provided. To ensure clarity and precision, I will need the specific sets of System Requirements (SYSRS) and their corresponding Software Requirements (SWRS) to proceed with the review. Please provide the details of these requirements, and I will begin the analysis.

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 detailed rating criteria. To apply these criteria, I will evaluate the System Requirements (SYSRS) and their corresponding Software Requirements (SWRS) you provide. Please share the requirements sets that you'd like me to assess, and I'll review them against the defined checks (conditions, actions and responses, parameter sources, and additional conditions) to rate the **Meaningfulness**, **Completeness**, and **Overall Quality** of the links between each set.



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). "



Let's begin the evaluation of **SYSRS 001** and **SWRS 001** based on the checks you specified.

1. Condition Check:

- **SYSRS 001** does not explicitly mention a condition like "after event X" or "before event Y."
- **SWRS 001** introduces two conditions:
 - After the reception of the command `CRM_CMD 0x1` (Start Standard Measurement)

- Before the start of the measurement, the software sets the register.

Since **SYSRS 001** does not reference any condition about when the sampling frequency should be set, the conditions introduced in **SWRS 001** appear to be **additional** conditions that are not reflected in the system requirement. The **SWRS** does introduce conditions that help clarify when the action occurs, but this wasn't mentioned in the **SYSRS**. Therefore, this would be considered an additional condition.

2. Action and Response Check:

- **SYSRS 001** specifies the action of digitizing the transducer input signal using a specific sampling frequency formula.
- **SWRS 001** focuses on setting the register ``BRG.FS.f_s`` to the same sampling frequency formula (``CALIB_samplingFreq + YPX_samplingFreqOffset``).

The core action—setting the sampling frequency—is correctly implemented in the software requirement. However, the focus in **SWRS 001** is on configuring the register, whereas **SYSRS 001** refers to the digitization of the input signal at the IC pins, which is a higher-level system action. While the implementation of the action matches, the system requirement focuses on the external outcome (digitization), while the software requirement focuses on configuring the internal register. The alignment of actions is satisfactory, though they operate on slightly different levels of abstraction.

3. Parameter Sources Check:

- Both **SYSRS 001** and **SWRS 001** correctly reference the parameters ``CALIB_samplingFreq`` and ``YPX_samplingFreqOffset``. These are consistent across both requirements, and the software requirement specifies where the parameter is applied (``BRG.FS.f_s`` register).

There is no indication in **SWRS 001** about where these parameters are sourced from. While the parameters are mentioned, the source of these calibration values is not explicitly indicated, leaving a gap in clarity.

4. Additional Conditions Check:

- **SWRS 001** introduces additional conditions, such as waiting for the command (``CRM_CMD 0x1``) and setting the sampling frequency before starting the measurement. These conditions are not referenced in **SYSRS 001**.

Since **SYSRS 001** lacks any conditions about when or how to set the sampling frequency, these additional conditions in **SWRS 001** are considered new. Although they provide operational context, they don't appear directly relevant to the system requirement as no such event or command is mentioned in **SYSRS 001**.

Meaningfulness Rating: Medium

- **Reason:** The software requirement supports the system requirement, but it adds conditions that were not mentioned in the system requirement, making the alignment less direct. The focus on

setting the register aligns with the system goal of setting the sampling frequency, but the added conditions are extraneous.

Completeness Rating: Medium

- **Reason:** While the software requirement captures the core action of configuring the sampling frequency, it lacks details on sourcing the parameters, and the additional conditions could lead to confusion. The link could be more complete with clearer parameter sourcing and closer alignment with the system requirement's scope.

Overall Quality of the Link: Medium

- **Reason:** Both **Meaningfulness** and **Completeness** are rated as Medium. The link is generally supportive, but the added conditions and lack of clarity on parameter sourcing reduce the quality. To improve, the software requirement could clarify where the parameters come from and eliminate unnecessary conditions not mentioned in the system requirement.

Suggested Revision:

To better align **SWRS 001** with **SYRS 001**, I recommend the following changes:

1. Remove or explicitly justify the conditions that are not mentioned in the system requirement, such as waiting for the ``CRM_CMD 0x1`` command and setting the frequency before starting the measurement.
2. Include information on where the parameters ``CALIB_samplingFreq`` and ``YPX_samplingFreqOffset`` are sourced or validated to ensure the software can fully support the system requirement's intent.



SYRS 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



Let's evaluate **SYRS 002** and **SWRS 002** based on the checks you provided.

1. Condition Check:

- **SYRS 002** does not explicitly specify any conditional logic (e.g., "IF X THEN Y"). It simply states that the Sensor IC firmware shall have a watchdog to detect malfunction and reset the watchdog timer within the Fault Tolerant Time Interval (FTTI).
- **SWRS 002** introduces a condition: "IF a malfunction is present, perform a reset." This conditional logic was not mentioned in the system requirement but is relevant because it adds clarity to the watchdog's behavior in malfunction scenarios. Since detecting and responding to a malfunction is implied in **SYRS 002**, this additional condition in **SWRS 002** does not conflict with the system requirement, and it seems to appropriately clarify how the reset should be triggered.

2. Action and Response Check:

- **SYSRS 002** states the requirement for the firmware to "detect FW or HW malfunction by resetting the WDG time in appropriate time within the FTTI."
- **SWRS 002** captures the action by enabling the watchdog to detect a software or hardware malfunction and reset within the Fault Tolerant Time Interval (FTTI) of 20ms.

The actions described in **SWRS 002** align closely with **SYSRS 002**, as both involve enabling a watchdog for detecting malfunctions and resetting within a specific time. The software requirement also adds the exact FTTI value of 20ms, which is helpful and provides more detail. The reset action is explicitly mentioned in both, with the only difference being that **SWRS 002** adds clarity about how the malfunction leads to the reset action.

3. Parameter Sources Check:

- **SYSRS 002** refers to "FTTI" (Fault Tolerant Time Interval), but it doesn't specify a particular value.
- **SWRS 002** explicitly states the Fault Tolerant Time Interval as 20ms, providing a concrete value for the parameter. This is an example of **SWRS 002** adding relevant detail that was implied in **SYSRS 002**.

However, the system requirement did not mention where this FTTI value comes from or whether it is a fixed or configurable parameter. The software requirement uses a fixed value of 20ms, but there is no indication of where this value originates from or if it's consistent with any specific design requirement. A clarification about the source or flexibility of this FTTI parameter would improve the link.

4. Additional Conditions Check:

- The only additional condition in **SWRS 002** is the "IF malfunction is present" clause. This is logically relevant to the system requirement, as the intent of the watchdog is to reset after a malfunction. This additional condition does not conflict with **SYSRS 002** but rather provides a more precise behavioral description.

There are no irrelevant or extraneous conditions in **SWRS 002** that are unrelated to the system requirement.

Meaningfulness Rating: High

- **Reason:** The software requirement clearly implements the intent of the system requirement. The added condition ("IF malfunction is present") provides more operational detail, which is directly relevant to the system requirement's intent. Both requirements are strongly aligned in purpose and functionality.

Completeness Rating: Medium

- **Reason:** While **SWRS 002** generally supports the system requirement, there is a gap in detailing the source or flexibility of the FTTI value (20ms). The system requirement does not specify the

value, while the software requirement does, but there is no explanation of where this value comes from or if it's subject to configuration or adjustment. Including details on this would improve completeness.

Overall Quality of the Link: Medium

- **Reason:** Although the **Meaningfulness** rating is high, the **Completeness** is medium due to the lack of clarity about the FTTI value. This results in an overall medium rating, as more detail is needed about how the 20ms FTTI is derived or configured.

Suggested Revision:

To improve the alignment and completeness between **SYSRS 002** and **SWRS 002**, I recommend:

1. Clarifying in **SWRS 002** whether the 20ms Fault Tolerant Time Interval is a fixed value or configurable, and if configurable, where it is sourced from.
2. Optionally, **SYSRS 002** could specify a default or range for the FTTI value to ensure consistency between system and software requirements.



```
"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.

"
```



Let's evaluate **SYSRS 003** and **SWRS 003** based on the checks provided.

1. Condition Check:

- **SYSRS 003** includes a condition: **BEFORE Sending CRM_RCC After Reception of the CRM_FCC**. This condition specifies when the action should occur: it is triggered after receiving a CRM_FCC but before sending the CRM_RCC.
- **SWRS 003** introduces a similar condition: **BEFORE responding with the CRM_RSP**. However, **SWRS 003** does not reference the sequence of events around CRM_FCC and CRM_RCC that are specified in the system requirement. It only mentions that the software should set the parameter before responding with the CRM_RSP.

The condition in **SWRS 003** is related, but it omits important details from **SYSRS 003** regarding the context of CRM_FCC reception and CRM_RCC sending. This creates a misalignment between the two requirements in terms of the precise sequence of events.

2. Action and Response Check:

- **SYSRS 003** specifies that the Sensor IC should set the parameter ``CrmStatus = Status Value`` in the CRM_RSP before sending CRM_RCC.
- **SWRS 003** similarly states that the software shall include the ``CrmStatus`` parameter in the CRM_RSP. However, **SWRS 003** does not specify that the parameter should be set to ``Status Value``, which is a crucial detail in **SYSRS 003**.

The core action—setting the CRM_RSP parameter—is correctly implemented in both requirements, but **SWRS 003** lacks the explicit mention of setting the ``CrmStatus`` to a particular value (``Status Value``). This is a significant omission, as the system requirement clearly indicates that the value must be assigned.

3. Parameter Sources Check:

- **SYSRS 003** references the parameter ``CrmStatus`` and specifies that it should be set to a specific value (``Status Value``).
- **SWRS 003** mentions the inclusion of ``CrmStatus`` in the CRM_RSP but does not address the specific value it should be set to. It does not clarify where the value for ``CrmStatus`` comes from or how it is derived, leaving a gap in the details of the parameter sourcing.

The absence of the specific value setting (``Status Value``) in **SWRS 003** and the lack of clarity on where this value comes from suggest that parameter sourcing is incomplete.

4. Additional Conditions Check:

- **SWRS 003** does not introduce any additional conditions unrelated to the system requirement. The condition of setting ``CrmStatus`` before responding with the CRM_RSP is appropriate but simplified compared to **SYSRS 003**, which contains more detailed conditions (i.e., after CRM_FCC reception and before CRM_RCC sending).

There are no irrelevant or extraneous conditions in **SWRS 003**, but the simplification of conditions causes a mismatch with the more detailed system requirement.

Meaningfulness Rating: Medium

- **Reason:** While **SWRS 003** supports the system requirement by addressing the need to include the ``CrmStatus`` parameter in the CRM_RSP, it lacks the specific detail about setting the value to ``Status Value`` and does not fully capture the sequence of events described in **SYSRS 003**. The alignment is partial, but important aspects are missing.

Completeness Rating: Low

- **Reason:** **SWRS 003** does not specify the assignment of the ``CrmStatus`` parameter to ``Status Value``, nor does it address the more detailed sequence of conditions involving CRM_FCC reception and CRM_RCC sending, which are critical to understanding the system behavior. Additionally, the source of the ``Status Value`` is not addressed in **SWRS 003**, leaving the link incomplete.

Overall Quality of the Link: Low

- **Reason:** Since **Completeness** is rated as Low and **Meaningfulness** is rated as Medium, the overall quality of the link is Low. **SWRS 003** does not fully reflect the conditions or actions described in **SYSRS 003**, leading to a weak link between the two requirements.

Suggested Revision:

To improve the alignment and completeness between **SYSRS 003** and **SWRS 003**, I recommend the following changes:

1. **Clarify the Conditions:** Ensure that **SWRS 003** explicitly references the sequence of events involving CRM_FCC reception and CRM_RCC sending to match **SYSRS 003**.
2. **Specify the Parameter Value:** Add a statement in **SWRS 003** indicating that the ``CrmStatus`` parameter should be set to ``Status Value``, as specified in **SYSRS 003**.
3. **Provide Parameter Source Details:** Clarify where the ``Status Value`` comes from or how it is derived to complete the parameter sourcing information.

These changes would strengthen the link between the system and software requirements by ensuring that all critical details are properly aligned and addressed.



"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."



Let's evaluate **SYSRS 004** and **SWRS 004** based on the criteria you provided.

1. Condition Check:

- **SYSRS 004** includes two conditions:
 - **If performing Erase Backup**
 - **If addressed with the Unicast Command**

These conditions are clearly stated and require that both are met before the Sensor IC responds with a report acknowledgement (``CRM_RSP 0x0``).

- **SWRS 004** mirrors these two conditions exactly:
 - **If performing Erase Backup**
 - **If addressed with the Unicast Command**

The conditions in **SWRS 004** match **SYSRS 004** perfectly. There are no discrepancies or additional conditions introduced. The alignment between conditions is strong.

2. Action and Response Check:

- **SYSRS 004** specifies the action: If both conditions are met, the Sensor IC **shall respond** with ``CRM_RSP 0x0: Report Acknowledgement``.
- **SWRS 004** specifies that the software **shall prepare to respond** with the same report acknowledgement (``CRM_RSP 0x0``), but it does not explicitly state that the software actually sends the response. Instead, it stops at the preparation stage.

This creates a partial misalignment: **SYSRS 004** expects the Sensor IC to actually send the response, while **SWRS 004** only prepares to send it. The difference between “preparing to respond” and actually sending the response is significant because it leaves the final action (sending the response) ambiguous in **SWRS 004**. The software should explicitly state that the response is sent once preparation is complete to fully align with **SYSRS 004**.

3. Parameter Sources Check:

- Both **SYSRS 004** and **SWRS 004** reference the response type ``CRM_RSP 0x0: Report Acknowledgement``, and this is consistent between the two. There is no ambiguity regarding this parameter or any need for further sourcing information, as the response type is clearly defined in both requirements.

4. Additional Conditions Check:

- **SWRS 004** does not introduce any additional conditions that are not present in **SYSRS 004**. It strictly follows the two conditions specified in the system requirement, making it closely aligned without any irrelevant or extraneous conditions.

Meaningfulness Rating: Medium

- **Reason:** The software requirement generally aligns with the system requirement, but it stops short of fully implementing the specified action. While **SWRS 004** prepares to send the ``CRM_RSP 0x0``, it does not explicitly mention that the response is actually sent, as required by **SYSRS 004**. This reduces the clarity of the alignment and action between the two requirements.

Completeness Rating: Medium

- **Reason:** The software requirement captures the conditions and correctly identifies the response type, but it does not fully complete the expected action (sending the response). As a result, **SWRS 004** is missing a critical part of the system requirement, leading to incomplete implementation. No details are missing regarding the parameters, but the action itself is incomplete.

Overall Quality of the Link: Medium

- **Reason:** Since both **Meaningfulness** and **Completeness** are rated as Medium, the overall quality of the link is also Medium. The conditions and parameters are well-aligned, but the lack of explicit action to send the response weakens the overall quality.

Suggested Revision:

To improve the alignment between **SYSRS 004** and **SWRS 004**, I recommend the following changes:

1. **Explicitly State the Action:** In **SWRS 004**, change the phrasing from "prepare to respond" to "respond with `CRM_RSP 0x0`" to ensure that the software requirement explicitly matches the system requirement's action.
2. **Ensure Consistency in Action:** Make sure that the software not only prepares but also completes the response process as specified in **SYSRS 004**.

These changes would ensure that the system and software requirements are fully aligned in both conditions and actions.



"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. "



Let's evaluate **SYSRS 005** and **SWRS 005/SWRS 006** based on the provided criteria.

1. Condition Check:

- **SYSRS 005** includes a clear condition: **IF the Sensor IC generates more information than possible to store in the Event Buffer or Time Series Data Buffer**, indicating a scenario where the amount of data exceeds the storage capacity of the buffers.
- **SWRS 005** does not explicitly address this condition. Instead, it states that the software shall store information for **up to 5 Echo Events**, but it does not reference any scenario where more information is generated than can be stored in the buffer.
- **SWRS 006** introduces a condition related to memory constraints: **IF NOT possible regarding memory constraints**, it allows for a reduction in the number of stored time series data samples from 100 to 50. This is somewhat related to the system requirement's idea of memory constraints but does not specifically address the action of discarding information when capacity is exceeded, as described in **SYSRS 005**.

In conclusion, **SWRS 005/006** do not fully address the core condition in **SYSRS 005**—when buffer limits are exceeded. While **SWRS 006** discusses a reduction of data based on memory constraints, it does not directly address what happens if even the reduced limit is exceeded.

2. Action and Response Check:

- **SYSRS 005** describes the action: **THEN the Sensor IC shall forget the according information.** This indicates that when the event or time series data buffers are full, the Sensor IC should discard the extra information to prevent overflow.
- **SWRS 005** only specifies that the software stores **up to 5 Echo Events**, but it does not provide any guidance on what should happen if more than 5 events are generated.
- **SWRS 006** similarly mentions storing **up to 100 time series data samples** and a fallback mechanism to reduce this number to **50 samples** if memory constraints apply. However, it does not mention what happens if the memory limits for even 50 samples are exceeded.

The core action from **SYSRS 005**—discarding or forgetting data when buffer limits are exceeded—is **not** explicitly captured in either **SWRS 005** or **SWRS 006**. Therefore, the response described in the system requirement is not fully implemented in the software requirements.

3. Parameter Sources Check:

- **SYSRS 005** references two buffers: **Event Buffer** and **Time Series Data Buffer**.
- **SWRS 005** references storage for **up to 5 Echo Events**, which likely corresponds to the **Event Buffer** mentioned in the system requirement. However, it does not specify that this is the maximum capacity of the Event Buffer or provide any clarity on what happens when this buffer is full.
- **SWRS 006** references the **Time Series Data Buffer** in terms of storing up to **100 time series data samples**, with a fallback to **50 samples**. This seems to correspond to the **Time Series Data Buffer** mentioned in the system requirement.

While **SWRS 005/006** reference relevant parameters, they do not explain where these buffer limits (5 Echo Events and 100/50 time series samples) come from or how these limits were derived. There is no clear link to how these values were determined, leaving a gap in the sourcing of these parameters.

4. Additional Conditions Check:

- **SWRS 006** introduces the condition about reducing the number of time series samples based on memory constraints. This condition is not present in **SYSRS 005** but seems related. However, it does not address the action of forgetting information, which is the key response in **SYSRS 005**.

There are no other additional or irrelevant conditions in **SWRS 005** or **SWRS 006** that would conflict with the system requirement. However, the software requirements do not sufficiently cover the scenario of exceeding buffer limits and discarding information, which is central to the system requirement.

Meaningfulness Rating: Low

- **Reason:** The software requirements only partially address the system requirement. While **SWRS 005** and **SWRS 006** mention buffer sizes and memory constraints, they fail to include the crucial action described in **SYSRS 005**—discarding data when buffer limits are exceeded. As a result, the link between the system and software requirements is weak and lacks relevance to the main system requirement's intent.

Completeness Rating: Low

- **Reason:** The software requirements do not fully cover the core scenario described in the system requirement. **SYSRS 005** specifies that the Sensor IC should discard information when buffers are full, but this action is not mentioned in either **SWRS 005** or **SWRS 006**. Additionally, there is no clear sourcing for the buffer size limits in the software requirements. This leaves the link significantly incomplete.

Overall Quality of the Link: Low

- **Reason:** Both **Meaningfulness** and **Completeness** are rated as Low, which results in an overall Low rating. The software requirements do not address the main action of discarding data when buffer limits are exceeded, nor do they provide clear information on where the buffer limits come from.

Suggested Revision:

To improve the alignment and completeness between **SYSRS 005** and **SWRS 005/006**, I recommend the following changes:

1. **Explicitly Address the Action of Forgetting Information:** Add a statement in **SWRS 005** and **SWRS 006** to specify that if the event buffer or time series data buffer is full, the software shall discard the excess data as specified in **SYSRS 005**.
2. **Clarify the Parameter Sources:** Ensure that the software requirements explain where the buffer limits (5 Echo Events, 100/50 time series samples) are derived from and whether these limits are configurable or fixed.
3. **Address Scenarios Exceeding Buffer Limits:** Ensure that **SWRS 005/006** explicitly cover what should happen when more data is generated than the buffer can hold, as this is central to the system requirement.

These revisions would ensure that the software requirements fully implement the system requirement's intent.



```
"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").
"
```



Let's evaluate **SYSRS 006** and **SWRS 007** based on the provided criteria.

1. Condition Check:

- **SYSRS 006** specifies two conditions:
 1. **If Performing Memory Access in Unicast**
 2. **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 define when the action should be performed, specifically when a memory access occurs in unicast mode and the memory area is locked.

- **SWRS 007** mirrors these conditions exactly. It also specifies:
 1. **If Performing Memory Access in Unicast**
 2. **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**

The conditions are perfectly aligned between the system and software requirements, with no missing or additional conditions introduced.

2. Action and Response Check:

- **SYSRS 006** specifies the response: **THEN the Sensor IC shall respond with CRM_RSP 0x8: Report Lock Status and CRM_RSP 0x8 Flag KeyRequest = 0b1 ("key requested")**. This defines that the Sensor IC should send a response indicating the lock status and set the `KeyRequest` flag to `0b1`, signaling that a key is requested.
- **SWRS 007** states: **the Software shall prepare to respond with CRM_RSP 0x8: Report Lock Status and CRM_RSP 0x8 Flag KeyRequest = 0b1 ("key requested")**. While it mentions preparation for the response, it does not explicitly state that the response will actually be sent, stopping at the preparation phase.

This is a similar issue seen in previous requirements: **SWRS 007** only indicates that the software will prepare for the response, whereas **SYSRS 006** requires the Sensor IC to actually send the response. The lack of clarity on whether the response is sent weakens the alignment.

3. Parameter Sources Check:

- **SYSRS 006** references two key parameters:
 - **CRM_CMD 0x8 Parameter MemoryArea**
 - **CRM_CMD 0x8 Flag KeyRequest**

These parameters are explicitly mentioned in **SWRS 007** as well, ensuring consistency in referencing them.

- However, neither the system requirement nor the software requirement provides additional information on where the values for these parameters are sourced from or how they are managed. While the parameters are referenced correctly, the source or derivation of these values (like how the memory area and key request flag are determined) is not fully explained.

4. Additional Conditions Check:

- **SWRS 007** does not introduce any additional conditions not present in **SYSRS 006**. The software requirement closely follows the conditions specified in the system requirement, ensuring no irrelevant or extraneous conditions are included.

There are no additional conditions in **SWRS 007** that would conflict with or deviate from **SYSRS 006**.

Meaningfulness Rating: Medium

- **Reason:** The software requirement aligns with the system requirement by specifying the same conditions and action. However, it lacks clarity on whether the response is actually sent, as it only mentions "preparing" to respond, which reduces the direct implementation of the system requirement. The meaning of the link is weakened by this ambiguity.

Completeness Rating: Medium

- **Reason:** While the software requirement captures the conditions and identifies the correct parameters, it does not fully complete the action of responding. The omission of the explicit step to send the response leaves the link incomplete. Additionally, there is no clarity on the sourcing or handling of the parameter values (MemoryArea and KeyRequest flag).

Overall Quality of the Link: Medium

- **Reason:** Since both **Meaningfulness** and **Completeness** are rated as Medium, the overall quality of the link is Medium. The software requirement supports the system requirement but falls short of fully implementing it due to the lack of clarity on sending the response and parameter handling.

Suggested Revision:

To improve the alignment and completeness between **SYSRS 006** and **SWRS 007**, I recommend the following changes:

1. **Explicitly State the Response Action:** In **SWRS 007**, change the wording from "prepare to respond" to "respond with CRM_RSP 0x8: Report Lock Status and CRM_RSP 0x8 Flag KeyRequest = 0b1" to ensure that the response action is explicitly completed, as required by **SYSRS 006**.
2. **Clarify Parameter Sourcing:** Include details on where the `MemoryArea` and `KeyRequest` flag values come from or how they are managed in the software to improve completeness and clarity.

These changes would ensure that the software requirement fully aligns with the system requirement in both actions and parameter handling.



"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 by CRM_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.



Let's evaluate **SYSRS 007** and **SWRS 008** based on the criteria you provided.

1. Condition Check:

- **SYSRS 007** specifies several conditions:
 1. **If Performing Memory Access**
 2. **If the Key is Set**
 3. **If Read Operation is Requested**
 4. **If Memory Area Is Set Read Accessible**

All these conditions must be met before the action (setting the memory area to the state "Locked") is performed.

- **SWRS 008** mirrors these conditions exactly:
 1. **If Performing Memory Access**
 2. **If the Key is Set**
 3. **If Read Operation is Requested**
 4. **If Memory Area Is Set Read Accessible**

The conditions in both the system and software requirements are identical, ensuring a perfect match with no missing or additional conditions.

2. Action and Response Check:

- **SYSRS 007** specifies the action: **THEN the Sensor IC shall set the state of the Memory Area identified by CRM_CMD 0x8 Parameter MemoryArea to the state Locked.** This requires that the Sensor IC change the state of the identified memory area to "Locked" when the specified conditions are met.
- **SWRS 008** mirrors 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 described in **SWRS 008** is consistent with the system requirement and clearly implements the expected behavior.

The alignment between the system and software requirements in terms of actions is strong, as both specify the same operation of locking the memory area when all conditions are met.

3. Parameter Sources Check:

- **SYSRS 007** references the parameter **CRM_CMD 0x8 Parameter MemoryArea**, which is critical for identifying the memory area whose state is to be changed.
- **SWRS 008** also references the same **CRM_CMD 0x8 Parameter MemoryArea**, ensuring consistency in parameter usage.

However, neither the system requirement nor the software requirement provides additional details on how the memory area is identified or where the value for the **Key** or **Read Accessible** state is sourced from. While the parameters are consistently referenced, their sourcing or validation is not explained, leaving a small gap in clarity.

4. Additional Conditions Check:

- **SWRS 008** does not introduce any additional conditions not present in **SYSRS 007**. It follows the conditions exactly as defined in the system requirement, ensuring no extraneous conditions are included.

There are no irrelevant or additional conditions in **SWRS 008**, which ensures that it stays focused on the system requirement's intent.

Meaningfulness Rating: High

- **Reason:** The software requirement fully supports the system requirement by implementing the same conditions and actions. Both requirements specify the same behavior—locking the memory area under specific conditions—making the link highly meaningful. There is no misalignment or unnecessary detail in **SWRS 008** that would weaken the connection to **SYSRS 007**.

Completeness Rating: Medium

- **Reason:** While the software requirement captures the essential conditions and actions described in the system requirement, it does not provide clarity on the source or validation of certain parameters, such as the **Key** or **MemoryArea** state. The absence of this information leaves a small gap in completeness, though the overall implementation is correct.

Overall Quality of the Link: High

- **Reason:** Despite the minor gap in parameter sourcing, both **Meaningfulness** and **Completeness** are rated sufficiently high to result in an overall High rating. The core actions and conditions are well-aligned, and the software requirement directly implements the system requirement's functionality.

Suggested Revision:

To further improve the alignment and completeness between **SYSRS 007** and **SWRS 008**, I recommend:

1. **Clarifying Parameter Sourcing:** Include information in **SWRS 008** on where the **Key** is validated and how the memory area's **Read Accessible** state is determined. This would ensure the software requirement is fully transparent regarding parameter management.

This change would eliminate any ambiguity regarding the parameters and improve the overall completeness of the link.



```
"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.
"
```



Let's evaluate **SYSRS 008** and **SWRS 009** based on the criteria provided.

1. Condition Check:

- **SYSRS 008** specifies several conditions:
 1. **If Performing Memory Access with Set Key**
 2. **If Write Operation is Requested**
 3. **If Memory Area Is Set Write Accessible**

All of these conditions must be true for the action to be performed (locking the memory area).

- **SWRS 009** introduces different conditions:
 1. **If Performing Memory Access** (same as SYSRS 008)
 2. **If the Key is NOT Set** (different from SYSRS 008, which requires the key to be set)
 3. **If Write Operation is Requested** (same as SYSRS 008)
 4. **If Memory Area Is Set Read Accessible** (different from SYSRS 008, which requires write access)

There is a significant difference in the conditions between **SYSRS 008** and **SWRS 009**:

- **SYSRS 008** expects the key to be set, while **SWRS 009** expects the key to **not** be set.
- **SYSRS 008** refers to the memory area being **write-accessible**, while **SWRS 009** refers to it being **read-accessible**.

Because of these differences, the conditions in **SWRS 009** do **not** match those in **SYSRS 008**, leading to a clear misalignment between the two requirements.

2. Action and Response Check:

- **SYSRS 008** specifies the action: **The Sensor IC shall set the state of the Memory Area identified by CRM_CMD 0x8 Parameter MemoryArea to the state Locked**. This action occurs when the specified conditions are met.
- **SWRS 009** specifies a similar action: **The Software shall set the state of the Memory Area identified by CRM_CMD 0x10 Parameter MemoryArea to the state Locked**.

However, there are discrepancies in the parameters used:

- **SYSRS 008** refers to **CRM_CMD 0x8 Parameter MemoryArea**, while **SWRS 009** refers to **CRM_CMD 0x10 Parameter MemoryArea**. This inconsistency suggests that different commands or parameters are being referenced, even though both requirements intend to perform the same action (locking the memory area). The command and parameter mismatch could result in a different operation or context being used in the software, leading to further misalignment.

The core action of locking the memory area is present in both, but the parameter inconsistency and different conditions make it unclear whether they are implementing the same functionality.

3. Parameter Sources Check:

- **SYSRS 008** refers to **CRM_CMD 0x8 Parameter MemoryArea**.
- **SWRS 009** refers to **CRM_CMD 0x10 Parameter MemoryArea**.

The difference in parameter references between **CRM_CMD 0x8** and **CRM_CMD 0x10** is significant. It suggests that different command structures are being used, which could impact the implementation of memory locking. Additionally, **SWRS 009** mentions **Read Accessible** memory areas, while **SYSRS 008** focuses on **Write Accessible** areas. These are fundamentally different conditions that point to different parameter handling or validation.

4. Additional Conditions Check:

- **SWRS 009** introduces a condition that is not present in **SYSRS 008**: **If the Key is NOT Set**. This is a direct contradiction of **SYSRS 008**, which requires the key to be set.
- Additionally, **SWRS 009** checks for **Read Accessible** memory areas, while **SYSRS 008** checks for **Write Accessible** memory areas, which again introduces an inconsistency.

These additional conditions in **SWRS 009** are not aligned with **SYSRS 008** and introduce scenarios that are different from the system requirement.

Meaningfulness Rating: Low

- **Reason:** The software requirement diverges significantly from the system requirement. The conditions in **SWRS 009** (key not set, read-accessible memory) do not align with **SYSRS 008** (key set, write-accessible memory). Additionally, the software requirement references a different command parameter (**CRM_CMD 0x10** vs. **CRM_CMD 0x8**), leading to further misalignment. This reduces the relevance of the software requirement to the system requirement.

Completeness Rating: Low

- **Reason:** **SWRS 009** does not fully or accurately implement the action described in **SYSRS 008**. The conditions are different, the referenced command parameters are different, and the core intent of the system requirement (locking a write-accessible memory area with a set key) is not reflected in the software requirement (which checks for a read-accessible memory area with an unset key). As a result, the link is significantly incomplete.

Overall Quality of the Link: Low

- **Reason:** Both **Meaningfulness** and **Completeness** are rated as Low due to the major discrepancies between the system and software requirements. The software requirement does not properly implement the system requirement, resulting in a weak and unclear link.

Suggested Revision:

To better align **SWRS 009** with **SYSRS 008**, the following changes are recommended:

1. **Correct the Conditions:** Ensure that **SWRS 009** matches the conditions outlined in **SYSRS 008**, specifically:
 - The key should be **set**, not unset.
 - The memory area should be **write-accessible**, not read-accessible.
2. **Use the Correct Parameter:** Update **SWRS 009** to reference **CRM_CMD 0x8 Parameter MemoryArea**, as specified in **SYSRS 008**, rather than **CRM_CMD 0x10**.
3. **Remove Contradictory Conditions:** Eliminate any conditions related to an unset key and read-accessible memory, as these contradict the system requirement.

These revisions would ensure that the software requirement accurately reflects the system requirement, improving the overall quality and alignment.



"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").



Let's evaluate **SYSRS 009** and **SWRS 010** based on the criteria provided.

1. Condition Check:

- **SYSRS 009** specifies the following conditions:
 1. **If Performing Memory Access in Broadcast**
 2. **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 are critical to triggering the action of responding with the appropriate report information.

- **SWRS 010** mirrors these conditions exactly:
 1. **If Performing Memory Access in Broadcast**
 2. **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**

The conditions in **SWRS 010** perfectly match those in **SYSRS 009**. There is no discrepancy between the system and software requirements in terms of conditions, ensuring alignment.

2. Action and Response Check:

- **SYSRS 009** specifies the action: **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")**. This defines the exact structure of the response when the conditions are met.
- **SWRS 010** states that **the Software shall prepare to respond** with the same message structure: **PDCM_RSP P0: Report CRM_RSP Information containing CRM_RSP 0x8: Report Lock Status and CRM_RSP 0x8 Flag KeyRequest = 0b1 ("key requested")**.

However, as in previous requirements, **SWRS 010** only specifies that the software shall **prepare to respond**, rather than explicitly stating that the response will be sent. While the preparation of the response is an essential step, the software requirement should ensure that the response is actually sent to fully align with **SYSRS 009**, which requires the Sensor IC to **respond**.

3. Parameter Sources Check:

- **SYSRS 009** references the following key parameters:
 - **CRM_CMD 0x8 Parameter MemoryArea**

- **CRM_CMD 0x8 Operation Flag**
 - **PDCM_RSP P0**
 - **CRM_RSP 0x8 Flag KeyRequest = 0b1**
- **SWRS 010** mirrors these references exactly, using the same parameters. Both requirements are consistent in their usage of **CRM_CMD 0x8** and the response format, ensuring that the software correctly implements the same parameter structure.

However, there is no additional detail in **SWRS 010** about how the **CRM_CMD 0x8** parameters are sourced or validated. While the parameters themselves are consistent, the lack of clarity on their source leaves room for improvement in the completeness of the link.

4. Additional Conditions Check:

- **SWRS 010** does not introduce any additional or irrelevant conditions. It strictly follows the conditions laid out in **SYSRS 009**, ensuring that no extraneous conditions are added.

There are no unnecessary conditions in **SWRS 010**, which ensures that the software requirement stays aligned with the system requirement.

Meaningfulness Rating: Medium

- **Reason:** The software requirement mirrors the conditions and parameters of the system requirement closely, but it stops short of fully implementing the required action. **SWRS 010** only specifies that the software will prepare to respond, whereas **SYSRS 009** explicitly requires that the Sensor IC actually sends the response. This slight difference weakens the overall alignment, though the intention of the software requirement is clear.

Completeness Rating: Medium

- **Reason:** While the software requirement captures the conditions, parameters, and core action of **SYSRS 009**, it does not fully complete the action of sending the response. Additionally, there is no clarification of how parameters like **CRM_CMD 0x8 MemoryArea** or **CRM_RSP 0x8 Flag KeyRequest** are sourced or validated, which leaves a small gap in completeness. The overall structure is correct, but further details would improve it.

Overall Quality of the Link: Medium

- **Reason:** Both **Meaningfulness** and **Completeness** are rated as Medium, leading to an overall Medium rating. The core intent of the system requirement is captured, but the software requirement lacks the explicit step of sending the response, and it could benefit from clearer parameter sourcing.

Suggested Revision:

To improve the alignment and completeness between **SYSRS 009** and **SWRS 010**, I recommend the following changes:

1. **Explicitly State the Response Action:** Update **SWRS 010** to explicitly state that the software **shall respond**, not just prepare to respond. This would ensure that the system requirement's action is fully implemented.

2. **Clarify Parameter Sourcing:** Provide additional information in **SWRS 010** about how the **CRM_CMD 0x8 MemoryArea** and **KeyRequest flag** are sourced or validated to improve the completeness of the requirement.

These revisions would bring the software requirement fully in line with the system requirement and ensure that all aspects are clearly addressed.



"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."



Let's evaluate **SYSRS 010**, **SYSRS 011**, and **SWRS 011** based on the provided criteria.

1. Condition Check:

- **SYSRS 010** and **SYSRS 011** both specify the same condition:
 1. **If Performing Memory Area Modification**
 2. **AFTER reception of CRM_CMD 0x9: Take Data or Send Data**

These conditions define when specific actions should be taken (caching data and incrementing the start address).

- **SWRS 011** mirrors the same conditions:
 1. **If Performing Memory Area Modification**
 2. **AFTER reception of CRM_CMD 0x9: Take Data or Send Data**

However, **SWRS 011** introduces an additional condition not mentioned in **SYSRS 010/011**: storing the data in RAM **until CRM_CMD 0xA: Perform Memory Operation with CRM_CMD 0xA**

Parameter MemoryOperation == 0x2 ("Write to NVM") is received. This additional condition is not mentioned in either system requirement, which makes it an extraneous condition.

2. Action and Response Check:

- **SYSRS 010** requires that the Sensor IC **cache the CRM_CMD 0x9 Parameter Data to the Data Cache**. This means that once the **CRM_CMD 0x9** is received, the data should be cached directly.
- **SYSRS 011** requires that the Sensor IC **increment the CRM_CMD 0x8 Parameter StartAddress by 1 word address** after receiving the **CRM_CMD 0x9** command.
- **SWRS 011** specifies that the software should **store the CRM_CMD 0x9 Parameter Data to RAM until CRM_CMD 0xA ("Write to NVM") is received**. This action deviates from **SYSRS 010** in two ways:
 1. **SYSRS 010** requires caching the data immediately to the **Data Cache**, but **SWRS 011** stores the data in **RAM**, which is a different type of memory.
 2. **SYSRS 011** requires incrementing the **StartAddress** by 1 word address, but this action is not addressed in **SWRS 011** at all.

There is a significant gap between the required actions in **SYSRS 010/011** and the actions described in **SWRS 011**. The software requirement does not mention incrementing the **StartAddress** (as required by **SYSRS 011**), nor does it align with the system requirement's instruction to cache data directly.

3. Parameter Sources Check:

- **SYSRS 010** and **SYSRS 011** reference key parameters:
 - **CRM_CMD 0x9 Parameter Data**
 - **CRM_CMD 0x8 Parameter StartAddress**
- **SWRS 011** references **CRM_CMD 0x9 Parameter Data**, but it does not mention **CRM_CMD 0x8 Parameter StartAddress**, which is important for fulfilling **SYSRS 011**.
- Additionally, **SWRS 011** introduces a new parameter related to **CRM_CMD 0xA** and its **MemoryOperation** parameter, which is not mentioned in **SYSRS 010/011**. This creates a disconnect between the parameter sets in the system and software requirements.

4. Additional Conditions Check:

- **SWRS 011** introduces an additional condition that is not present in **SYSRS 010/011**: the software stores data until **CRM_CMD 0xA ("Write to NVM")** is received. This is a new condition that adds complexity not specified in the system requirements.

SYSRS 010/011 do not mention the **CRM_CMD 0xA** command or any delayed operation involving **RAM** or **NVM**, so this additional condition in **SWRS 011** is not directly relevant to the system requirements and could be flagged as an unnecessary condition.

Meaningfulness Rating: Low

- **Reason:** **SWRS 011** does not align well with **SYSRS 010** and **SYSRS 011**. The software requirement introduces an additional condition (storing data in RAM until a subsequent command) that is not mentioned in the system requirements, and it does not implement the key action of incrementing the **StartAddress** required by **SYSRS 011**. Additionally, **SWRS 011** stores data in RAM rather than caching it, which is a fundamental deviation from **SYSRS 010**.

Completeness Rating: Low

- **Reason:** The software requirement does not cover all the actions described in the system requirements. Specifically:
 1. **SYSRS 011** requires incrementing the **StartAddress**, which is not mentioned in **SWRS 011**.
 2. **SWRS 011** introduces an unnecessary dependency on a subsequent command (**CRM_CMD 0xA**) and delays the operation of writing to **NVM**, which is not present in **SYSRS 010/011**.
 3. The software requirement stores data in **RAM**, which is a deviation from the system requirement's instruction to cache the data.

Overall Quality of the Link: Low

- **Reason:** Since both **Meaningfulness** and **Completeness** are rated as Low, the overall quality of the link is also Low. **SWRS 011** does not fully align with the actions or conditions specified in **SYSRS 010** and **SYSRS 011**, and it introduces unnecessary conditions and dependencies not mentioned in the system requirements.

Suggested Revision:

To better align **SWRS 011** with **SYSRS 010** and **SYSRS 011**, the following changes are recommended:

1. **Remove the CRM_CMD 0xA dependency:** Eliminate the additional condition about storing data in RAM until **CRM_CMD 0xA ("Write to NVM")** is received. This condition is not present in the system requirements and adds unnecessary complexity.
2. **Implement the StartAddress Increment:** Add the action to increment the **CRM_CMD 0x8 Parameter StartAddress by 1 word address** after receiving **CRM_CMD 0x9**, as required by **SYSRS 011**.
3. **Cache Data as Specified:** Ensure that the **CRM_CMD 0x9 Parameter Data** is cached (as specified in **SYSRS 010**) rather than stored in **RAM**.

These revisions would ensure that the software requirement properly reflects the system requirements' conditions and actions, resulting in a stronger and clearer link.



```
"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").
"
```



Let's evaluate **SYSRS 012** and **SWRS 012** based on the criteria provided.

1. Condition Check:

- **SYSRS 012** specifies the following conditions:
 1. **BEFORE Leaving the Discovery Mode**
 2. **IF the effective value of the parameter DSI3_physicalAddress is NOT equal to the default value of the parameter DSI3_physicalAddress**

These conditions must be met before the Sensor IC sets the error flag
err_discPhysAddrMismatch = 0x1.

- **SWRS 012** mirrors these conditions but adds more detail:
 1. **BEFORE Leaving the Discovery Mode**
 2. **IF the effective value of the Parameter DSI3_physicalAddress (RAM) is NOT equal to the default value of the Parameter DSI3_physicalAddress (NVM)**

The additional detail in **SWRS 012** introduces a distinction between the **RAM** and **NVM** values of the **DSI3_physicalAddress**, while **SYSRS 012** does not specify where these values are stored. This extra detail clarifies the implementation but could potentially lead to misinterpretation if the system requirement intended only one source for comparison. However, this detail seems relevant and does not introduce an unrelated condition.

2. Action and Response Check:

- **SYSRS 012** specifies that the Sensor IC shall set the error flag **err_discPhysAddrMismatch = 0x1** if the condition (mismatch in **DSI3_physicalAddress**) is met. This action is straightforward.
- **SWRS 012** mirrors this action but adds an additional behavior: **OTHERWISE, the software shall set err_discPhysAddrMismatch = 0x0 ("no error")**. While **SYSRS 012** does not mention what should happen if there is no mismatch, this additional action in **SWRS 012** adds clarity by explicitly stating that the error flag should be set to **0** (indicating no error) when there is no mismatch.

The additional action of setting **err_discPhysAddrMismatch = 0x0** in the absence of an error provides useful operational clarity and does not contradict the system requirement. It could be seen as a logical extension of **SYSRS 012**.

3. Parameter Sources Check:

- **SYSRS 012** refers to the parameter **DSI3_physicalAddress**, but it does not specify whether this is the value stored in **RAM** or **NVM**, leaving some ambiguity about the source of the parameter.
- **SWRS 012** explicitly differentiates between the **RAM** and **NVM** values of the **DSI3_physicalAddress**, which helps clarify the source of the parameters being compared.

This distinction in **SWRS 012** adds valuable detail, ensuring that the software compares the correct memory locations (RAM vs. NVM). While **SYSRS 012** does not specify the memory locations, this additional specificity does not contradict the system requirement and helps ensure correct implementation.

4. Additional Conditions Check:

- **SWRS 012** adds a condition that was not present in **SYSRS 012**: explicitly setting **err_discPhysAddrMismatch = 0x0** if there is no mismatch. This is a logical extension and provides operational clarity that is beneficial for the overall system behavior. Since **SYSRS 012** does not specify what happens in the "no error" case, this addition in **SWRS 012** is not irrelevant but instead completes the possible outcomes.

Meaningfulness Rating: High

- **Reason:** The software requirement implements the system requirement accurately and adds relevant details (distinction between **RAM** and **NVM**) that enhance the clarity of the comparison without contradicting the intent of the system requirement. The additional action of setting **err_discPhysAddrMismatch = 0x0** if there is no error also adds value without introducing irrelevant behavior.

Completeness Rating: High

- **Reason:** The software requirement not only covers the condition described in the system requirement but also addresses an additional scenario (no mismatch), ensuring that the behavior is fully defined. The distinction between **RAM** and **NVM** values of **DSI3_physicalAddress** further improves the completeness by ensuring the correct comparison is made.

Overall Quality of the Link: High

- **Reason:** Both **Meaningfulness** and **Completeness** are rated as High. The software requirement fully implements the system requirement and adds useful operational details that enhance clarity and precision without contradicting or diverging from the system requirement.

Suggested Revision:

No significant revision is necessary, but for full alignment, **SYSRS 012** could specify whether the **DSI3_physicalAddress** comparison involves **RAM** and **NVM**, as **SWRS 012** does. This would ensure there is no ambiguity in the system requirement itself regarding the source of the parameters.



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**).
"



Let's evaluate **SYSRS 013** and **SWRS 013**, **SWRS 014**, and **SWRS 015** based on the criteria provided.

1. Condition Check:

- **SYSRS 013** specifies no explicit conditions, but it does define that amplification must occur based on a combination of multiple parameters: **CALIB_analogGain**, **SPX_dynGainStartValue**, **CRM_CMD 0x4 Parameter CalibrationGain**, and **YPX_calibrationGainOffset**. The amplification process is triggered without reference to a specific event.
- **SWRS 013**, **SWRS 014**, and **SWRS 015** each introduce a condition: **AFTER reception of the command CRM_CMD 0x4: Start Calibration Measurement**. This indicates that amplification occurs only after this command is received.

The condition of waiting for the **CRM_CMD 0x4** command is not mentioned in **SYSRS 013**. This could be seen as an additional condition introduced in the software requirements, which adds clarity to when the amplification is triggered. Although this extra condition was not mentioned in **SYSRS 013**, it appears relevant and operationally useful for the implementation of the system requirement. However, it should be noted that this condition may need explicit clarification in the system requirement for better alignment.

2. Action and Response Check:

- **SYSRS 013** defines the action as the **amplification of the Transducer Input Signal** based on the sum of the following parameters:
 1. **CALIB_analogGain**
 2. **SPX_dynGainStartValue**
 3. **CRM_CMD 0x4 Parameter CalibrationGain**
 4. **YPX_calibrationGainOffset**

- **SWRS 013**, **SWRS 014**, and **SWRS 015** break this action into separate steps for configuring amplification:
 - **SWRS 013** sets the **CALIB_analogGain** to the **ASP_CFG.g_ana** register for analog amplification.
 - **SWRS 014** sets the **SPX_dynGainStartValue** to the **ENVP_AMPD_CFG.g_dig** register for digital amplification.
 - **SWRS 015** sets the **CRM_CMD 0x4 Parameter CalibrationGain** plus **YPX_calibrationGainOffset** to the **ENVP_GCAL.g_cal** register for calibration gain.

These software requirements split the overall amplification action into its constituent parts, assigning specific parameters to different registers. While **SYSRS 013** describes the action of amplifying the input signal using a combined sum of parameters, the software requirements implement this by individually setting the values of the parameters in different registers. This is a clear and logical breakdown of the system-level action into detailed software actions.

3. Parameter Sources Check:

- **SYSRS 013** lists four parameters involved in the amplification process:
 1. **CALIB_analogGain**
 2. **SPX_dynGainStartValue**
 3. **CRM_CMD 0x4 Parameter CalibrationGain**
 4. **YPX_calibrationGainOffset**
- **SWRS 013** correctly handles the **CALIB_analogGain** by setting it to **ASP_CFG.g_ana**.
- **SWRS 014** handles the **SPX_dynGainStartValue** by setting it to **ENVP_AMPD_CFG.g_dig**.
- **SWRS 015** handles the **CRM_CMD 0x4 Parameter CalibrationGain** plus **YPX_calibrationGainOffset** by setting them to **ENVP_GCAL.g_cal**.

All parameters mentioned in **SYSRS 013** are addressed in the corresponding software requirements. However, **SYSRS 013** implies that the parameters should be combined into a single formula for amplification, while the software requirements set each parameter in different registers. The software implementation is valid, as it provides more specific details about how the parameters are used, but it could benefit from clarifying the overall amplification process at the system level.

4. Additional Conditions Check:

- The software requirements add the condition of waiting for **CRM_CMD 0x4** before performing the amplification steps. This is an additional condition not mentioned in **SYSRS 013**, but it is logically relevant to the calibration process. Since **SYSRS 013** does not specify when the amplification should occur, this extra condition in **SWRS 013/014/015** provides necessary operational detail.

There are no irrelevant or unrelated conditions introduced in the software requirements.

Meaningfulness Rating: High

- **Reason:** The software requirements break down the system requirement into logical steps, each of which correctly implements part of the overall amplification process. Although **SWRS 013/014/015** split the action into analog, digital, and calibration gain settings, this approach directly supports the system requirement. The added condition of waiting for **CRM_CMD 0x4** provides additional operational context, making the implementation meaningful and relevant.

Completeness Rating: Medium

- **Reason:** While all the parameters and actions specified in **SYSRS 013** are covered by the software requirements, **SYSRS 013** implies that the amplification factors are combined, while **SWRS 013/014/015** treat them as separate values set in different registers. The lack of clarity in the system requirement regarding how the parameters should be combined (or whether they are set separately, as described in the software) leaves a small gap in completeness. Additionally, the introduction of the condition of **CRM_CMD 0x4** is helpful, but it should be explicitly mentioned in **SYSRS 013** to ensure full alignment.

Overall Quality of the Link: High

- **Reason:** Despite the slight difference in the way the amplification factors are applied, the software requirements faithfully implement the system requirement in a detailed manner. The core amplification process is correctly executed, and the software's additional details make the system's intent clearer.

Suggested Revision:

To improve alignment, consider clarifying **SYSRS 013** by explicitly stating whether the amplification factors should be combined into a single formula or applied separately (as is done in **SWRS 013/014/015**). Additionally, it would be helpful to mention the **CRM_CMD 0x4** command in **SYSRS 013** to indicate when the amplification process should occur. This would ensure better completeness and alignment between the system and software requirements.



"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.



Let's evaluate **SYSRS 014** and **SWRS 016** based on the criteria provided.

1. Condition Check:

- **SYSRS 014** specifies the condition:

1. **IF the Event_MAX is generated**—This condition triggers the action of measuring the event time and transmitting the timestamp.

- **SWRS 016** matches this condition by specifying that the software will send the **MAX events**. However, **SWRS 016** does not explicitly state that the software waits for **Event_MAX** to be generated, though it can be inferred that **MAX events** correspond to **Event_MAX**. The alignment here is adequate, but a clearer reference to the event generation trigger in **SWRS 016** would ensure more explicit consistency with the system requirement.

2. Action and Response Check:

- **SYSRS 014** specifies two main actions:
 1. **Measure the time of the occurrence of the event condition relative to the start of the measurement.**
 2. **Provide PDCM_RSP P12, P13 Parameter EventTimeStamp<i> for the <i>th Event transmitted in a DSI3 packet.**
- **SWRS 016** mirrors these actions:
 1. It specifies sending **MAX events with the measured time of the occurrence of the event** relative to the start of the measurement.
 2. It also mentions providing **PDCM_RSP P12, P13 Parameter EventTimeStamp<i> for the <i>th Event transmitted.**

The alignment between the actions in **SYSRS 014** and **SWRS 016** is strong, as both address measuring the event time and transmitting the timestamp. However, there is a minor discrepancy in the transmission medium:

- **SYSRS 014** mentions transmission via a **DSI3 packet**.
- **SWRS 016** mentions transmission in a **PDCM frame**.

While these could be referring to related or equivalent transmission formats, the terminology difference creates some ambiguity. It would be helpful to clarify whether **PDCM frame** and **DSI3 packet** are the same or interchangeable.

3. Parameter Sources Check:

- **SYSRS 014** specifies the use of the **PDCM_RSP P12, P13 Parameter EventTimeStamp<i> for the i-th Event**.
- **SWRS 016** correctly mirrors this by specifying that the software will provide the **PDCM_RSP P12, P13 Parameter EventTimeStamp<i> for the <i>th Event**. The parameter sourcing is consistent between the system and software requirements, ensuring that the correct timestamp parameters are used.

4. Additional Conditions Check:

- **SWRS 016** does not introduce any additional or unrelated conditions that are not present in **SYSRS 014**. The software requirement remains focused on the actions specified in the system requirement without adding unnecessary complexity.

There are no extraneous conditions in **SWRS 016**.

Meaningfulness Rating: High

- **Reason:** The software requirement directly supports the system requirement by mirroring the core actions of measuring the event time and transmitting the event timestamp. The condition of **Event_MAX** generation is generally aligned, though **SWRS 016** could be more explicit in referring to this specific event trigger. Overall, the link between the system and software requirements is meaningful.

Completeness Rating: Medium

- **Reason:** While the core actions and parameters are fully captured in the software requirement, there is a small discrepancy in terminology between **DSI3 packet** (used in **SYSRS 014**) and **PDCM frame** (used in **SWRS 016**). If these are indeed the same, it would be helpful to clarify this. Additionally, making the condition of **Event_MAX** generation more explicit in **SWRS 016** would improve completeness.

Overall Quality of the Link: High

- **Reason:** Despite the minor terminology difference and the slight ambiguity in how **Event_MAX** is referenced, the overall alignment between **SYSRS 014** and **SWRS 016** is strong. The software requirement effectively implements the system requirement, and both are focused on the same goal: measuring the event time and transmitting the timestamp.

Suggested Revision:

To improve clarity and completeness:

1. **Clarify Terminology:** Ensure that **PDCM frame** and **DSI3 packet** refer to the same concept or explicitly state any differences.
2. **Make Event Trigger Explicit:** In **SWRS 016**, explicitly reference **Event_MAX** as the trigger for the action, rather than simply referring to "MAX events."

These changes would ensure full alignment and remove any potential ambiguity between the system and software requirements.



"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; "



Let's evaluate **SYSRS 015** and **SWRS 017** based on the criteria provided.

1. Condition Check:

- **SYSRS 015** specifies the following condition:
 1. **AFTER Sending PDCM Packet After Reception of the BRC**—This condition defines the specific sequence of events that triggers the action: after the Sensor IC sends the PDCM packet following the reception of the BRC (Broadcast Command).
- **SWRS 017** mirrors this condition:
 1. **AFTER Sending PDCM Packet After Reception of the BRC**—The software requirement uses the exact same condition as the system requirement, ensuring full alignment.

2. Action and Response Check:

- **SYSRS 015** specifies the action: **The Sensor IC shall increment the PDCM_RSP Parameter KAC by 1; the counter is reset to 0 at overflow.**
- **SWRS 017** mirrors the same action: **The Software shall increment the PDCM_RSP Parameter KAC by 1.**

The core action—incrementing the **PDCM_RSP Parameter KAC** by 1—is identical in both the system and software requirements. However, there is a minor omission in **SWRS 017**: **SYSRS 015** mentions that the **KAC** counter is reset to 0 at overflow, while **SWRS 017** does not mention this. The software requirement should ideally include the reset behavior to fully capture the system requirement's intent. While the action of incrementing is correct, the missing detail about the reset introduces a small gap in completeness.

3. Parameter Sources Check:

- Both **SYSRS 015** and **SWRS 017** reference the same parameter: **PDCM_RSP Parameter KAC**. There is no ambiguity or discrepancy in parameter sourcing. Both requirements correctly focus on the **KAC** parameter and its incrementation, ensuring consistency.

4. Additional Conditions Check:

- **SWRS 017** does not introduce any additional or unrelated conditions. It stays focused on the same condition defined in **SYSRS 015** (after sending the PDCM packet after reception of the BRC), ensuring there are no extraneous conditions.

There are no additional or irrelevant conditions introduced in **SWRS 017**.

Meaningfulness Rating: High

- **Reason:** The software requirement directly implements the system requirement, correctly reflecting the key condition (after sending the PDCM packet following BRC reception) and the

action (incrementing the **KAC** parameter). The link between the system and software requirements is strong and meaningful.

Completeness Rating: Medium

- **Reason:** While **SWRS 017** correctly handles the incrementing of the **KAC** parameter, it omits the detail that the counter should be reset to 0 at overflow. This is a minor but important behavior described in **SYSRS 015** that ensures the full functionality of the counter. Including this behavior in **SWRS 017** would complete the implementation.

Overall Quality of the Link: High

- **Reason:** Despite the omission of the overflow reset detail, the software requirement is largely well-aligned with the system requirement. The core actions and conditions are accurately mirrored, making the link between the two requirements strong overall. The omission is minor, but it does not significantly reduce the quality of the link.

Suggested Revision:

To improve completeness, consider adding the following detail to **SWRS 017**:

- **Include the Reset Behavior:** Add a clause stating that the **KAC** counter should be **reset to 0 at overflow** to fully align with **SYSRS 015**.

This revision would ensure that the software requirement captures all aspects of the system requirement, resulting in a fully complete and aligned implementation.



"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."



Let's evaluate **SYSRS 016** and **SWRS 018** based on the criteria provided.

1. Condition Check:

- **SYSRS 016** specifies the condition:

1. **After Reception of the Valid CRM_FCC_CMD**—This condition is the trigger for setting the **KAC** parameter.

- **SWRS 018** mirrors this condition in one part:

1. **After Reception of the Valid CRM_FCC_CMD**—However, while the condition is the same, the action triggered is different, as explained below. The **SWRS 018** also includes an additional condition about incrementing **KAC** after sending the PDCM packet, which is unrelated to **SYSRS 016**. This creates a mismatch in the scope of the conditions, as **SYSRS 016** focuses only on the valid **CRM_FCC_CMD** event.

2. Action and Response Check:

- **SYSRS 016** specifies the action: **The Sensor IC shall set the PDCM_RSP Parameter KAC = 0x0** after receiving the valid **CRM_FCC_CMD**.
- **SWRS 018** specifies a different action: **The Software shall set the PDCM_RSP Parameter KAC = 0x3** after receiving the valid **CRM_FCC_CMD**. This is a clear discrepancy because the system requirement expects **KAC** to be set to **0x0**, but the software requirement sets it to **0x3**. This is a significant misalignment and would lead to incorrect behavior in the software compared to what is expected by the system.

Additionally, **SWRS 018** includes an unrelated action of incrementing the **KAC** parameter after sending the PDCM packet, which is not mentioned in **SYSRS 016**. This action is not problematic in itself but is outside the scope of **SYSRS 016**, which only focuses on the valid **CRM_FCC_CMD** event.

3. Parameter Sources Check:

- Both **SYSRS 016** and **SWRS 018** refer to the same parameter: **PDCM_RSP Parameter KAC**. The parameter is consistent across both requirements, so there is no issue with sourcing.
- However, there is a critical difference in the value assigned to this parameter: **SYSRS 016** sets **KAC = 0x0**, while **SWRS 018** sets **KAC = 0x3**. This difference in the value assigned to **KAC** is a major inconsistency.

4. Additional Conditions Check:

- **SWRS 018** introduces additional actions and conditions (incrementing **KAC** after sending the PDCM packet after BRC reception), which are not present in **SYSRS 016**. While this does not directly contradict **SYSRS 016**, it adds complexity that is not relevant to the system requirement.

These additional actions could cause confusion or potential misalignment with the broader system behavior if not carefully managed.

Meaningfulness Rating: Low

- **Reason:** The software requirement diverges from the system requirement in terms of the action taken after receiving the **CRM_FCC_CMD**. **SWRS 018** sets the **KAC** parameter to **0x3**, while **SYSRS**

016 requires it to be set to **0x0**. This difference significantly reduces the relevance of the software requirement to the system requirement. Additionally, the introduction of unrelated actions (incrementing **KAC**) adds unnecessary complexity.

Completeness Rating: Low

- **Reason:** **SWRS 018** does not fully implement the system requirement. The expected behavior of setting **KAC = 0x0** is not present, and instead, the software requirement incorrectly sets **KAC = 0x3**. Moreover, the extra condition about incrementing **KAC** is not necessary for fulfilling **SYSRS 016** and creates unnecessary additional actions that are not related to the system requirement.

Overall Quality of the Link: Low

- **Reason:** Both **Meaningfulness** and **Completeness** are rated as Low due to the significant discrepancy in how the **KAC** parameter is set and the introduction of unrelated conditions. The link between **SYSRS 016** and **SWRS 018** is weak and misaligned, and the software requirement does not fulfill the system requirement's intent.

Suggested Revision:

To improve the alignment and completeness between **SYSRS 016** and **SWRS 018**, the following changes are recommended:

1. **Correct the KAC Value:** Ensure that **SWRS 018** sets the **PDCM_RSP Parameter KAC = 0x0** after receiving the valid **CRM_FCC_CMD**, as required by **SYSRS 016**.
2. **Remove Unrelated Conditions:** Remove the condition about incrementing **KAC** after sending the PDCM packet following BRC reception, as this is not relevant to **SYSRS 016** and introduces unnecessary complexity.

These revisions would ensure that the software requirement fully and accurately implements the system requirement, leading to a stronger and clearer link between the two.



"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")."



Let's evaluate **SYSRS 017** and **SWRS 019** based on the criteria provided.

1. Condition Check:

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

These conditions together trigger the action of setting the error flag **err_burstGenerationStopped = 0x1**.

- **SWRS 019** simplifies the conditions:
 1. **IF in Main-Measurement**
 2. **IF Break Burst Generation occurs**

While the condition of being in a measurement mode (**Main-Measurement**) loosely aligns with **Standard Path Measurement** in **SYSRS 017**, the condition of **Break Burst Generation occurs** is much broader and less specific compared to **SYSRS 017**'s more detailed condition about stopping an ongoing measurement and incomplete functionality in the US-Transducer Control. The system requirement is more granular, focusing specifically on the **Non-Modulated Sine Burst** process not being completed, which is not reflected in **SWRS 019**.

2. Action and Response Check:

- **SYSRS 017** specifies that **the Sensor IC shall set Error: err_burstGenerationStopped = 0x1** if all the conditions are met. This defines that an error should be flagged when a measurement is stopped before the non-modulated sine burst has completed.
- **SWRS 019** specifies that **the software shall set Error: err_burstGenerationStopped = 0x1** when a **Break Burst Generation** occurs. This part of the software action aligns with **SYSRS 017** in terms of setting the error, but **SWRS 019** uses a more generic condition of burst generation breaking, without the detailed conditions about stopping a measurement and the incomplete **US-Transducer Control** functionality.

3. Parameter Sources Check:

- **SYSRS 017** references the **US-Transducer Control (Non-Modulated Sine Burst)** and the measurement process in detail, which helps specify when the error should be triggered.
- **SWRS 019** does not reference these specific aspects but instead uses broader terms like **Break Burst Generation**. This lack of specificity makes it unclear whether **SWRS 019** fully captures the precise context of the system requirement. The software requirement does not mention the non-modulated sine burst, leaving a gap in parameter handling and context.

4. Additional Conditions Check:

- **SWRS 019** simplifies the conditions significantly compared to **SYSRS 017**. While it stays focused on the error related to burst generation, it does not address the specifics about the stopped measurement and the non-completed burst generation process outlined in **SYSRS 017**. The

broader conditions in **SWRS 019** could lead to the error being triggered in more situations than intended by **SYSRS 017**, which is more narrowly focused.

There are no completely irrelevant conditions in **SWRS 019**, but the simplification leads to a potential over-triggering of the error, which diverges from the intent of the system requirement.

Meaningfulness Rating: Medium

- **Reason:** The software requirement generally supports the system requirement in that it triggers the **err_burstGenerationStopped = 0x1** error when there is an issue with burst generation. However, the software requirement uses much broader conditions than the system requirement, which may lead to over-triggering of the error or failure to capture the specific scenario described in **SYSRS 017**. The alignment is partial but lacks the necessary detail to be fully meaningful.

Completeness Rating: Low

- **Reason:** **SWRS 019** does not fully capture the detailed conditions specified in **SYSRS 017**, particularly the focus on stopping the measurement and the incomplete **Non-Modulated Sine Burst** functionality. The broader terms used in **SWRS 019** do not fully reflect the specific behavior required by the system requirement, leading to an incomplete implementation. Additionally, **SWRS 019** does not reference the **US-Transducer Control** functionality, which is a key part of **SYSRS 017**.

Overall Quality of the Link: Medium

- **Reason:** While the software requirement captures the general idea of triggering the error for burst generation issues, it lacks the specificity and completeness needed to fully align with the system requirement. The broader conditions in **SWRS 019** reduce the overall quality of the link.

Suggested Revision:

To improve the alignment and completeness between **SYSRS 017** and **SWRS 019**, the following changes are recommended:

1. **Include Detailed Conditions:** Update **SWRS 019** to reflect the specific conditions in **SYSRS 017**, including stopping an ongoing measurement and the incomplete **Non-Modulated Sine Burst** process in **US-Transducer Control**. This would ensure the error is triggered only under the correct circumstances.
2. **Clarify Parameter Sources:** Ensure that **SWRS 019** includes a reference to the **Non-Modulated Sine Burst** in **US-Transducer Control**, as specified in **SYSRS 017**, to match the system requirement's level of detail.

These revisions would ensure that the software requirement fully implements the specific scenario described in the system requirement, leading to a stronger and clearer link between the two.



"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."



Let's evaluate **SYSRS 018** and **SWRS 020** based on the criteria provided.

1. Condition Check:

- **SYSRS 018** specifies two conditions:
 1. **If Performing Memory Access**
 2. **If the Key is NOT set**

These conditions trigger the action of setting the memory area state to **Unlocked**.

- **SWRS 020** specifies similar but opposite conditions:
 1. **If Performing Memory Access**
 2. **If the Key is set**

The conditions in **SWRS 020** are logically opposite to **SYSRS 018** because **SWRS 020** describes the state when the key **is set**, while **SYSRS 018** describes the state when the key **is not set**. This reflects a key difference in behavior: one condition handles the unlocked state, while the other handles the locked state.

While **SWRS 020** does not directly conflict with **SYSRS 018**, it is dealing with a different condition (key set vs. key not set). Therefore, **SWRS 020** is not directly related to **SYSRS 018**, and there is a mismatch in the conditions they handle.

2. Action and Response Check:

- **SYSRS 018** specifies the action: **The Sensor IC shall set the state of the Memory Area identified by CRM_CMD 0x8 Parameter MemoryArea to the state: Unlocked**. This action occurs when the key is not set.
- **SWRS 020** specifies a different action: **The Software shall set the state of the Memory Area identified by CRM_CMD 0x8 Parameter MemoryArea to the state: locked**. This action is triggered when the key is set.

The two actions are opposite in nature: **SYSRS 018** focuses on unlocking the memory area when no key is set, while **SWRS 020** focuses on locking the memory area when the key is set. These actions are not directly related; they describe opposite states of memory access control based on different key conditions. Thus, **SWRS 020** does not implement the action described in **SYSRS 018**, but instead handles a different scenario.

3. Parameter Sources Check:

- Both **SYSRS 018** and **SWRS 020** reference the same parameter: **CRM_CMD 0x8 Parameter MemoryArea**. The memory area is consistently referenced across both the system and software requirements, ensuring that the correct memory area is being modified.
- However, the difference lies in the states that are being set. **SYSRS 018** sets the state to **Unlocked**, while **SWRS 020** sets it to **Locked**. Although the parameter source is the same, the states being applied are opposite.

4. Additional Conditions Check:

- **SWRS 020** introduces no additional or irrelevant conditions, but it addresses a different scenario compared to **SYSRS 018**. The software requirement handles the **locked** state, while the system requirement handles the **unlocked** state. This difference means that while **SWRS 020** does not add unnecessary conditions, it also does not fully implement the scenario described in **SYSRS 018**.

Meaningfulness Rating: Low

- **Reason:** **SWRS 020** addresses the opposite scenario compared to **SYSRS 018**. While **SYSRS 018** focuses on unlocking the memory area when no key is set, **SWRS 020** handles locking the memory area when the key is set. These two actions are related in terms of memory access control, but they address opposite conditions. As such, **SWRS 020** does not meaningfully support or implement **SYSRS 018**.

Completeness Rating: Low

- **Reason:** **SWRS 020** does not implement the condition or action specified in **SYSRS 018**. While it correctly handles a different scenario (locking the memory area when the key is set), it does not address the scenario where the key is not set and the memory area needs to be unlocked. Therefore, the software requirement is incomplete in terms of implementing the system requirement.

Overall Quality of the Link: Low

- **Reason:** Since **SWRS 020** addresses the opposite scenario from **SYSRS 018**, there is little alignment between the two requirements. The conditions and actions described in **SWRS 020** are not related to **SYSRS 018**, leading to a weak link.

Suggested Revision:

To ensure alignment between the system and software requirements, I recommend the following changes:

1. **Create a Software Requirement for the Unlocked State:** There should be a corresponding software requirement that handles the condition where the key is not set and the memory area is unlocked, as specified in **SYSRS 018**.
2. **Clarify the Link Between Locking and Unlocking:** If **SWRS 020** is intended to complement **SYSRS 018**, the relationship between locking and unlocking should be made clear, with each scenario handled in separate but related software requirements.

These revisions would ensure that the software requirement properly implements both the locked and unlocked states, resulting in a stronger and clearer link between the system and software requirements.



"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")."



Let's evaluate **SYSRS 019** and **SWRS 021** based on the criteria provided.

1. Condition Check:

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

These conditions define when the action of setting the **Status Value** to **0x2 ("invalid")** should occur.

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

The conditions in **SWRS 021** are identical to those in **SYSRS 019**, ensuring perfect alignment. There are no missing or additional conditions introduced in the software requirement.

2. Action and Response Check:

- **SYSRS 019** specifies the action: **The Sensor IC shall set the parameter Status Value = 0x2 ("invalid")**. This action is triggered when the specified conditions are met.
- **SWRS 021** mirrors this action exactly: **The Software shall set the parameter Status Value = 0x2 ("invalid")**.

The core action of setting the **Status Value** to **0x2** is identical in both the system and software requirements. There is no discrepancy in how the action is implemented, ensuring that the software requirement correctly implements the system requirement.

3. Parameter Sources Check:

- Both **SYSRS 019** and **SWRS 021** refer to the same parameter: **Status Value**. There is no ambiguity or discrepancy in parameter sourcing, as both requirements handle the same parameter with the same value (**0x2**) being set.

The parameter sourcing and handling are consistent across both the system and software requirements.

4. Additional Conditions Check:

- **SWRS 021** does not introduce any additional or irrelevant conditions. It stays focused on the conditions defined in **SYSRS 019**, ensuring that no extraneous conditions are added.

There are no unnecessary or irrelevant conditions introduced in **SWRS 021**, ensuring alignment with the system requirement.

Meaningfulness Rating: High

- **Reason:** The software requirement perfectly mirrors the system requirement in terms of both conditions and actions. The conditions are identical, and the core action of setting the **Status Value** to **0x2** is fully supported by the software requirement. The link between the system and software requirements is highly meaningful.

Completeness Rating: High

- **Reason:** **SWRS 021** fully implements **SYSRS 019** without any omissions or discrepancies. All conditions and actions are present, and the software requirement correctly handles the parameter as specified by the system requirement. There is no missing information or incomplete implementation.

Overall Quality of the Link: High

- **Reason:** Both **Meaningfulness** and **Completeness** are rated as High, resulting in a strong overall quality of the link. **SWRS 021** accurately and completely implements **SYSRS 019**, ensuring perfect alignment between the system and software requirements.

Suggested Revision:

No revisions are necessary, as **SWRS 021** perfectly implements **SYSRS 019**. The software requirement captures all aspects of the system requirement, resulting in a well-aligned and complete link.



"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")."



Let's evaluate **SYSRS 020** and **SWRS 022** based on the criteria provided.

1. Condition Check:

- **SYSRS 020** specifies the following conditions:
 1. **If Performing Memory Access in Broadcast**
 2. **AFTER reception of the CRM_FCC_CMD**
 3. **If Addressed with the Unicast Command**

These conditions define when the action of setting the **Status Value** to **0x2 ("invalid")** should occur, specifically when a unicast command is addressed.

- **SWRS 022** specifies similar conditions:
 1. **If Performing Memory Access in Broadcast**
 2. **AFTER reception of the CRM_FCC_CMD**
 3. **If Addressed with the Broadcast Command**

There is a **key difference** in the third condition:

- **SYSRS 020** triggers the action if addressed with a **Unicast Command**.
- **SWRS 022** triggers the action if addressed with a **Broadcast Command**.

This is a significant mismatch in the condition. **SYSRS 020** focuses on the **Unicast Command**, while **SWRS 022** focuses on the **Broadcast Command**. The software requirement is not aligned with the system requirement in terms of the specific condition for addressing the command.

2. Action and Response Check:

- **SYSRS 020** specifies the action: **The Sensor IC shall set the parameter Status Value = 0x2 ("invalid")**. This action occurs when the specified conditions are met, particularly when addressed with the unicast command.
- **SWRS 022** mirrors the same action: **The Software shall set the parameter Status Value = 0x2 ("invalid")**, but it triggers this action when addressed with the broadcast command instead of the unicast command.

The core action of setting the **Status Value** to **0x2** is correctly implemented, but the condition under which it is triggered is incorrect in **SWRS 022**. This misalignment leads to incorrect behavior since the system requirement explicitly states that the action should occur upon receiving a unicast command.

3. Parameter Sources Check:

- Both **SYSRS 020** and **SWRS 022** reference the same parameter: **Status Value**, with the correct value of **0x2**. The parameter sourcing is consistent, but the discrepancy lies in when this value is set (unicast vs. broadcast addressing).

4. Additional Conditions Check:

- **SWRS 022** introduces no additional or irrelevant conditions, but it focuses on the wrong addressing condition (broadcast instead of unicast). This is a misalignment, as **SYSRS 020** is specific to unicast addressing.

Meaningfulness Rating: Low

- **Reason:** The software requirement incorrectly implements the condition for setting the **Status Value**. **SYSRS 020** specifies that the action should be triggered when addressed with a **unicast command**, but **SWRS 022** applies the action to a **broadcast command**. This discrepancy leads to a lack of relevance between the system and software requirements, as the core condition is misinterpreted.

Completeness Rating: Low

- **Reason:** **SWRS 022** does not fully implement **SYSRS 020**. The condition related to addressing the command is incorrect, leading to an incomplete fulfillment of the system requirement. The software does not capture the intended scenario, which is focused on unicast command handling.

Overall Quality of the Link: Low

- **Reason:** Since both **Meaningfulness** and **Completeness** are rated as Low, the overall quality of the link is Low. The software requirement does not implement the correct condition from the system requirement, resulting in a misaligned and incomplete link.

Suggested Revision:

To align **SWRS 022** with **SYSRS 020**, the following change is necessary:

1. **Correct the Addressing Condition:** Update **SWRS 022** to trigger the action when addressed with a **Unicast Command**, as specified in **SYSRS 020**, instead of a broadcast command.

This revision would ensure that the software requirement fully captures the condition specified in the system requirement, leading to a stronger and more accurate link between the two.



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"



Let's evaluate **SYSRS 021**, **SWRS 023**, and **SWRS 024** based on the criteria provided.

1. Condition Check:

- **SYSRS 021** specifies that the firmware should allow **configurability** for selecting retries for **open** and **short** faults independently. There is no specific condition or trigger for when this configurability is applied; it simply states that the functionality should be available.
- **SWRS 023** introduces the following condition:
 1. **IF the software detects an open or short error**
 2. **AND retry is configured as enabled in the DEFAULT OPEN SHORT RETRY value**

While **SWRS 023** does involve a retry mechanism for open or short errors, it conditions this behavior on the detection of an error and the retry being enabled through the default configuration. This provides some configurability but does not fully align with the independent configurability aspect mentioned in **SYSRS 021**. **SYSRS 021** seems to imply that users can independently configure retries for open and short faults, but **SWRS 023** does not explicitly differentiate between open and short faults regarding configurability.

SWRS 024 references the **DEFAULT OPEN SHORT RETRY** setting but does not introduce any additional conditions. It simply lists this diagnostic configuration item, which supports the software's ability to configure retries, though it does not directly address the independent configurability for open and short faults.

2. Action and Response Check:

- **SYSRS 021** specifies that the firmware should allow **configurability** to independently select retries for open and short faults. This means that the user should be able to configure retries for open faults and short faults separately.

- **SWRS 023** specifies that if an open or short error is detected and the retry is enabled in the configuration, the affected channel is re-enabled for one **TIMESLICE** at least every 200 milliseconds. This action addresses the retry mechanism for faults but does not mention configurability for open and short faults separately. The software requirement groups both faults under the same retry configuration.
- **SWRS 024** lists **DEFAULT OPEN SHORT RETRY** as a diagnostic configuration item, which implies configurability for the retry mechanism, but it does not clarify whether open and short faults can be configured independently. There is no clear indication that the retry configuration can be separately adjusted for open and short faults, as **SYSRS 021** specifies.

3. Parameter Sources Check:

- **SYSRS 021** refers to allowing configurability for retry selection, but it does not mention specific parameters or sources for this configuration.
- **SWRS 023** references the **DEFAULT OPEN SHORT RETRY** value in the **COMPILE TIME CONFIGURATION** data, which seems to serve as the source of the retry setting. However, this configuration item appears to control retries for both open and short faults as a single unit, rather than allowing separate configuration for each fault type.
- **SWRS 024** lists **DEFAULT OPEN SHORT RETRY** as a configuration item but does not provide further details on its usage or whether it can be applied independently to open and short faults.

4. Additional Conditions Check:

- **SWRS 023** introduces the condition of retrying only if the retry is enabled in the **DEFAULT OPEN SHORT RETRY** setting, which is not explicitly mentioned in **SYSRS 021**. This additional condition makes the retry mechanism dependent on a compile-time configuration setting, which seems appropriate but doesn't fully address the independent configurability of retries for open and short faults as outlined in **SYSRS 021**.

There are no irrelevant or unnecessary conditions in **SWRS 023** or **SWRS 024**, but the key condition of independent configurability is missing.

Meaningfulness Rating: Medium

- **Reason:** The software requirements partially address the system requirement by providing a retry mechanism for open and short faults. However, the core concept of **independent configurability** for retries (open vs. short) is not fully captured in **SWRS 023** or **SWRS 024**. The software requirements group the two faults under a single retry setting, which diverges from the system requirement's intent for independent control.

Completeness Rating: Low

- **Reason:** **SWRS 023** and **SWRS 024** do not fully implement **SYSRS 021** because they do not allow separate configurability for open and short fault retries. While they introduce a retry mechanism and reference a configuration setting, the ability to configure retries independently for open and

short faults is missing. The system requirement calls for more granular control, which is not reflected in the software requirements.

Overall Quality of the Link: Medium

- **Reason:** The software requirements cover part of the system requirement by implementing a retry mechanism for open and short faults, but they fail to deliver the independent configurability that **SYSRS 021** specifies. As a result, the overall link is only partially meaningful and incomplete.

Suggested Revision:

To improve alignment between **SYSRS 021** and **SWRS 023/024**, the following changes are recommended:

1. **Introduce Independent Configurability:** Modify **SWRS 023** and **SWRS 024** to explicitly allow independent configurability for open and short faults, as required by **SYSRS 021**. This could be achieved by introducing separate configuration items for open and short retries.
2. **Clarify Diagnostic Configuration:** In **SWRS 024**, add details to indicate how **DEFAULT OPEN SHORT RETRY** can be configured independently for open and short faults, ensuring that users can select retries separately for each fault type.

These revisions would ensure that the software requirements fully capture the system requirement's intent for independent configurability of retries.

This conversation is archived. To continue, please unarchive it first.

 Unarchive