Digital documentation in ReqIF™ format

User manuals and errata sheets

About this document

Scope and purpose

This document introduces the ReqIF™ XML format, the digital and machine-readable format that will be provided together with PDF versions of the user manual and errata sheet. An explanation of how PDF-based document content is mapped into the ReqIF™ format is provided, including relevant tools that can be used to further analyze or process Infineon digital documentation.

Intended audience

The intended audience for this document is software engineers, requirement engineers, application engineers, developers of electronic systems, and other technicians using the microcontroller documentation in automatized ways or aiming to integrate it into their own environment.

Table of contents

1 ReqIF™ XML format high level overview

ReqIF™ is an XML-based Requirement Interchange Format that provides a common interface of requirements. With its roots in the automotive industry, the ReqIF™ format provides lossless requirement exchange and workflow integration.

As a meta model, this format provides an XML schema that is used as a common data model across supporting tool environments. Workflows of different companies can be interconnected by this format; this format provides status information about deliverables in a way that the receiving company can use dependent linking on certain features.

1.1 The inventors: Object Management Group (OMG)

The ReqIF™ format was originally introduced by the OMG to provide a method of requirement interchange between different companies, having limited access to the original requirement databases.

For the current used version of the format (ReqIF™ 1.2), OMG shares the specification, normative and informative documents on their webpage.

1.2 Example scenario

Company A provides a certain functionality that is used as a fundamental part in the development of Company B's product. An early exchange of information in the ReqIF™ format can provide preliminary objects that can be used to further refine the development process in Company B.

Especially in the automotive industry, where the development is often tied to norms like the functional safety standard ISO 26262, this early available exchange can provide a basic structure for development of customer parts in advance and provide objects that can be used in the V-development model proposed by ISO 26262.

ReqIF™ files provided with the PDF version of a user manual or an errata sheet uses the structure defined by the ReqIF™ format and wrap chapters into these requirement containers. As a result, these documents are machine-readable. This enables customers to use them either directly as linkable objects for further development or to use algorithms for analyzing and automating the (pre-)processing for content analysis.

Providing requirement containers of the documentation on a chapter level enables more granular tracking of the documents content. To leverage integration into machine processing and requirement engineering, certain elements within the content is marked with additional meta-data. Main purpose of these metadata is, to easily identify and track certain elements and make them visible by conversion tooling. This supports actual information about what a certain element like a bitfield means, without the need of heuristic approaches to identify them. By doing so, robustness of both: the file format and further processing is increased. Focus for this metadata marking are registers. Elements within a register container are tagged based on the IP-XACT standard. This standard is provided by the SPIRIT consortium and has the goal to make IP description and configuration compatible across different vendors.

2 Proposed tools to use ReqIF™

For ReqIF™ files, the main tool designed for reading and manipulating the content is the ReqIF™ Studio from ReqIF™ Academy. It provides a common overview of the requirements provided by the ReqIF™ format and allows a structured view of provided requirements. In addition, requirement engineering tools can be directly used to integrate the objects from ReqIF™ into a requirement environment.

For automating in a more generic way, XML-focused processing engines like XSLT or XML parsers allow extracting the information or its subset for further processing. An outlook on potential applications is given at the end of this document.

2.1 ReqIF™ Studio

Infineon will use the ReqIF™ XML format as a container to wrap our document information. ReqIF™ Studio will show one requirement entry per chapter or register of the underlying document.

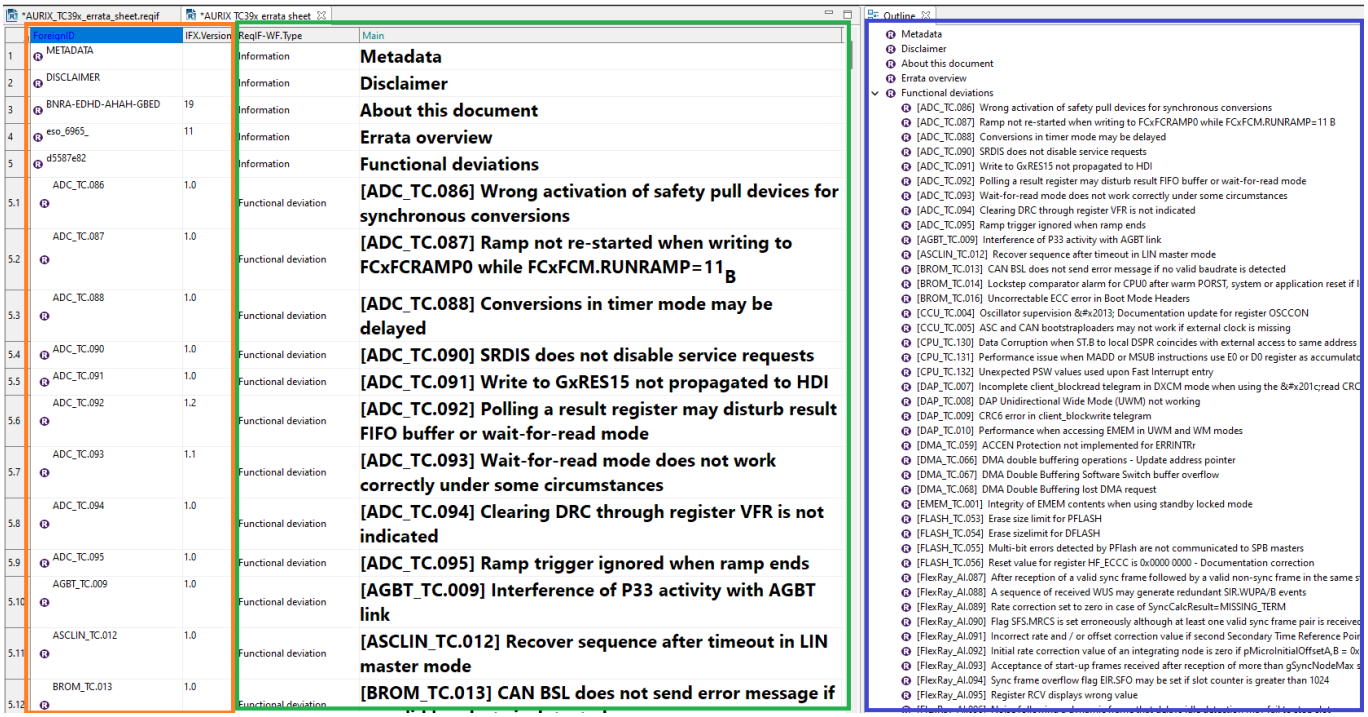


Figure 1 AURIX™ TC39x errata sheet opened as ReqIF™ in ReqIF™ Studio

Beside a complete machine-readable version of the document, every chapter is split into its own container (highlighted in green), providing a document-unique identifier and version (highlighted in orange). This allows easy tracing of changes within the document itself and enables to create linkable objects in requirement databases that support the ReqIF™ format.

In addition, the full structure that reflects the document's table of contents (ToC) is listed (highlighted in blue) for easy navigation. Because registers are the focus to become accessible for processing as requirements, every register is put into its own wrapping container. This results in one unique requirement object per register.

2.2 Requirement engineering tools

To provide a requirement-based work environment, many tools are compatible with the ReqIF™ format. Seamless integration of external provided requirements in these tools provide easy access of traceability measures based on exchanged requirements.

As the choice of the right requirement engineering tooling highly depends on individual environments, multiple tools are available. A short overview of some of the most used tools, supporting the ReqIF™ standard is provided in the following list:

- ReqIF Academy - ReqIF™ Studio
- IBM – DOORS® family
- ptc - Codebeamer
- Jama Software - Jama Connect®
- Siemens - Polarion®

3 Information classifier based on the IP-XACT standard

Because the ReqIF™ format does not have the IP-XACT standard built in, but only provides certain structures to hold the metadata, the documents content is structured in XHTML using attribute-based classification of the content. This enables using more specific classifiers that are connected to the very minimum content it describes.

The document's register content is wrapped in metadata, which is based on the 2018 IP-XACT standard. Because the main point of view for the IP-XACT standard is on the bitfield level while our documentation is focused on the register level, additional identifiers that are not part of the original IP-XACT standard maybe required. These identifiers will be added as part of the customized extensions and allow to map the content between bitfield-based and register-based views.

Metadata which is used in our ReqIF™ XML files are shown in Chapter 4 – Mapping of document content.

3.1 Example use case

As shown in **Figure 2**, the granularity of the requirement provided in ReqIF™ is at the register level. This helps to easily make registers linkable as objects in requirement databases; however, this lacks details if requirements need to be linked to bitfields, for example. This is important especially when registers share layouts or are addressable via different bus addresses. In such cases, it is hard to differentiate between them when dealing only at the register level.

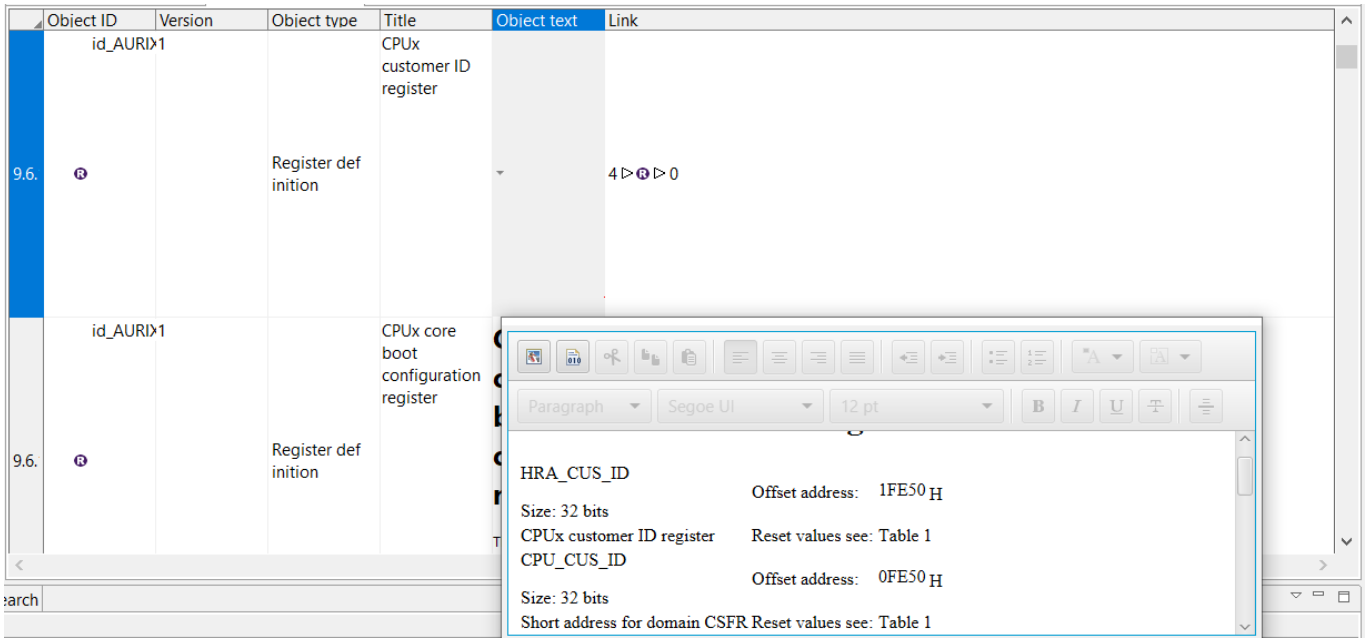


Figure 2 Default granularity by having one requirement container per register

If ReqIF™ provides metadata only on the requirement level, a classification applies to the whole paragraph of the requirement. For a register description that refers to multiple bitfields, the granularity is not fine enough to allow the classification of each referred bitfield individually; instead, it only provides a summary of these elements inside the requirement.

For better machine readability and processing capability, this metadata is moved into the XHTML domain of the ReqIF™ format, allowing to map the additional information directly to these elements. Because this view is used for machine processing, it allows better integration for automated analysis when working at the file level.

In general, providing a capability to further process files in an automatic way improves usability and integrateability when compared to manual integration of large documents.

Figure 3 ReqIF™ shown on XML level in Oxygen XML Editor providing more details

As **Figure 3** shows, making use of the wrapping structures inside the ReqIF™ format allows to add more granular metadata. When comparing the XML view to the ReqIF™ Studio view, multiple registers that share the same layout are much more than the same requirement now. It can easily be distinguished between different registers, even when sharing the same layout and shown as the same register view in the user manual. The metadata at the XML level allows to distinguish between these registers (highlighted in green). In addition, coherent content is delimited by the XML structure and provides disjunct blocks of content for further processing. As an example, the offset address of each corresponding register is structured at the same level (highlighted in blue) as the corresponding register name (green), but within one wrapping parent element.

3.2 IP-XACT classifier used

The following IP-XACT classifiers are used in the ReqIF™ format for the user manual and errata sheet:

Table 1 Used IP-XACT classifier

IP-XACT metadatum	Used for	Structure
`Ipxact_longName`	Full name	Register
`Ipxact_description`	Description text	Register
`Ipxact_name`	Short name	Register
`Ipxact_addressOffset`	Offset address	Register
`Ipxact_field`	Bitfield definition	Bitfield

IP-XACT metadatum	Used for	Structure
`Ipxact_name`	Name	Bitfield
`Ipxact_description`	Description text	Bitfield
`Ipxact_access`	Access type	Bitfield
`Ipxact_bitoffset_bitwidth`	Bitfield offset and range	Bitfield
`Ipxact_enumeratedValue`	Enumeration of bitfield values	Bitfield

because the standard is focused on bitfield view, there are some classifiers which slightly deviate from the original. As an example, `ipxact_bitoffset_bitwidth` is actually a combination of two identifiers: `ipxact_bitoffset` and `ipxact_bitwidth`. In Infineon documentation, bits and bitfields are specified as the offset inside the register and when spanning multiple bits, as an array [x:y]: there is no separate value for the actual size. Instead, the size can be derived as the difference between starting and ending bits.

4 Mapping of document content into ReqIF™ format

This section provides a better insight on how the actual information is stored in the ReqIF™ format in comparison with the user manual or the errata sheet, a mapping overview is provided, highlighting which information of the PDF document will be mapped to which part of the ReqIF™ format. This enables to use the ReqIF™ format provided to interpret or further process the information for integration into a customer-specific environment.

For easier navigation, the structure of an example erratum and an example register from the user manual is shown here, providing a split into sections.

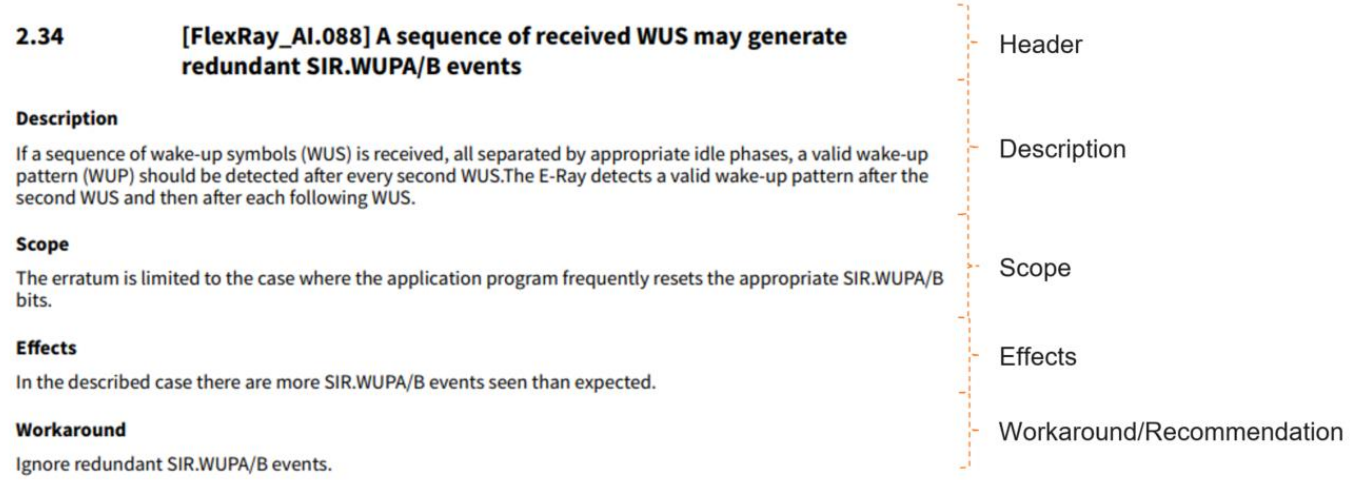
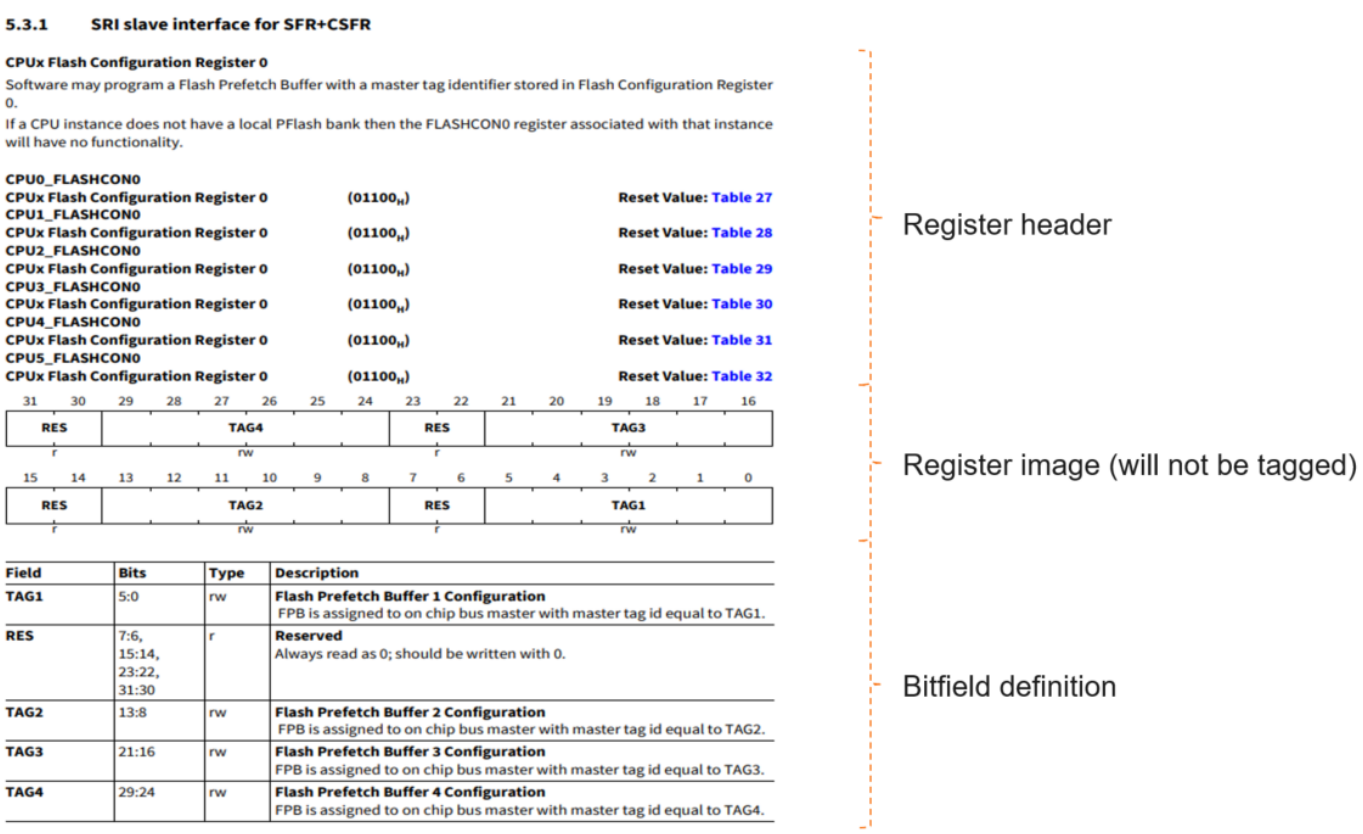


Figure 4 Sections in an erratum



5.3.1 SRI slave interface for SFR+CSFR

CPUx Flash Configuration Register 0

Software may program a Flash Prefetch Buffer with a master tag identifier stored in Flash Configuration Register 0.

If a CPU instance does not have a local PFlash bank then the FLASHCON0 register associated with that instance will have no functionality.

CPU0_FLASHCON0
CPUx Flash Configuration Register 0 (01100_H) **Reset Value:** Table 27
CPU1_FLASHCON0
CPUx Flash Configuration Register 0 (01100_H) **Reset Value:** Table 28
CPU2_FLASHCON0
CPUx Flash Configuration Register 0 (01100_H) **Reset Value:** Table 29
CPU3_FLASHCON0
CPUx Flash Configuration Register 0 (01100_H) **Reset Value:** Table 30
CPU4_FLASHCON0
CPUx Flash Configuration Register 0 (01100_H) **Reset Value:** Table 31
CPU5_FLASHCON0
CPUx Flash Configuration Register 0 (01100_H) **Reset Value:** Table 32

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES				TAG4				RES		TAG3					
r				rw				r		rw					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES				TAG2				RES		TAG1					
r				rw				r		rw					

Field	Bits	Type	Description
TAG1	5:0	rw	**Flash Prefetch Buffer 1 Configuration** FPB is assigned to on chip bus master with master tag id equal to TAG1.
RES	7:6, 15:14, 23:22, 31:30	r	**Reserved** Always read as 0; should be written with 0.
TAG2	13:8	rw	**Flash Prefetch Buffer 2 Configuration** FPB is assigned to on chip bus master with master tag id equal to TAG2.
TAG3	21:16	rw	**Flash Prefetch Buffer 3 Configuration** FPB is assigned to on chip bus master with master tag id equal to TAG3.
TAG4	29:24	rw	**Flash Prefetch Buffer 4 Configuration** FPB is assigned to on chip bus master with master tag id equal to TAG4.

Figure 5 Sections of a register description

For easier accessibility, each element of the document is described first with its meaning and then further mapped into the IP-XACT standard, showing how these are represented in the ReqIF™ format. These sections are used to support the orientation inside the PDF structure, which should be known already.

The following mapped elements that are available in the documents are described with their abstract meaning and how these are mapped into the ReqIF™ format using XHTML structures and metadata as XML attributes.

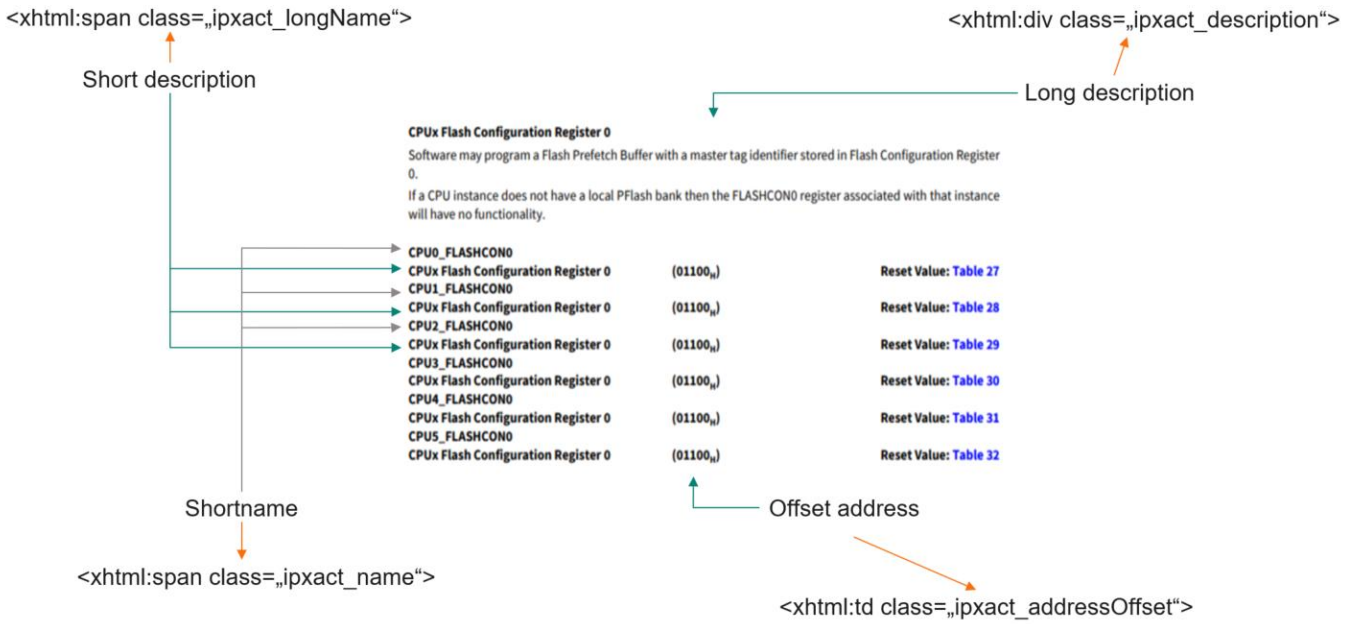


Figure 6 User manual register-header section mapped into ReqIF™

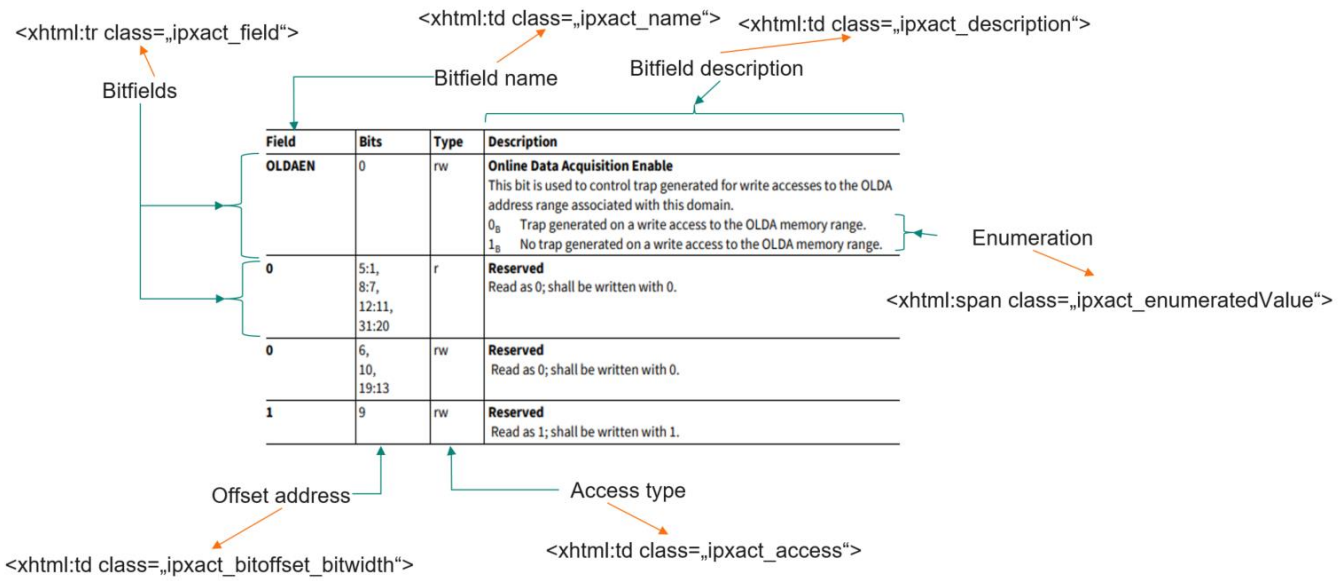


Figure 7 User manual bitfield definition section mapped into ReqIF™

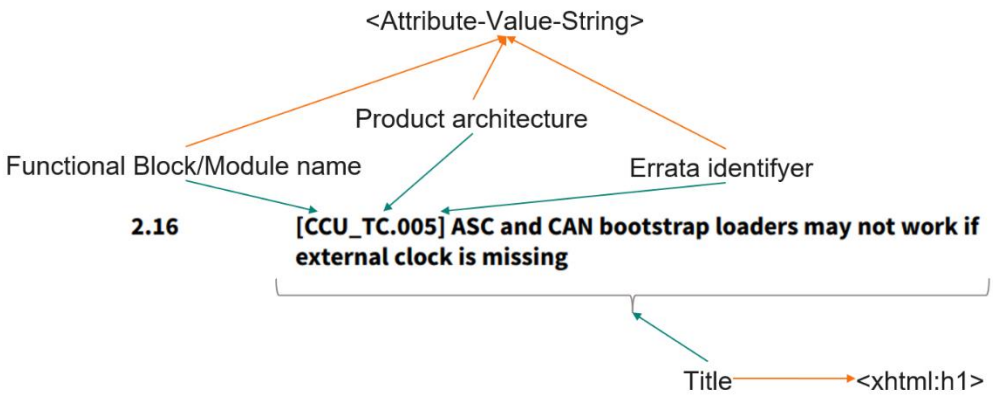


Figure 8 Errata sheet headline mapping into ReqIF™

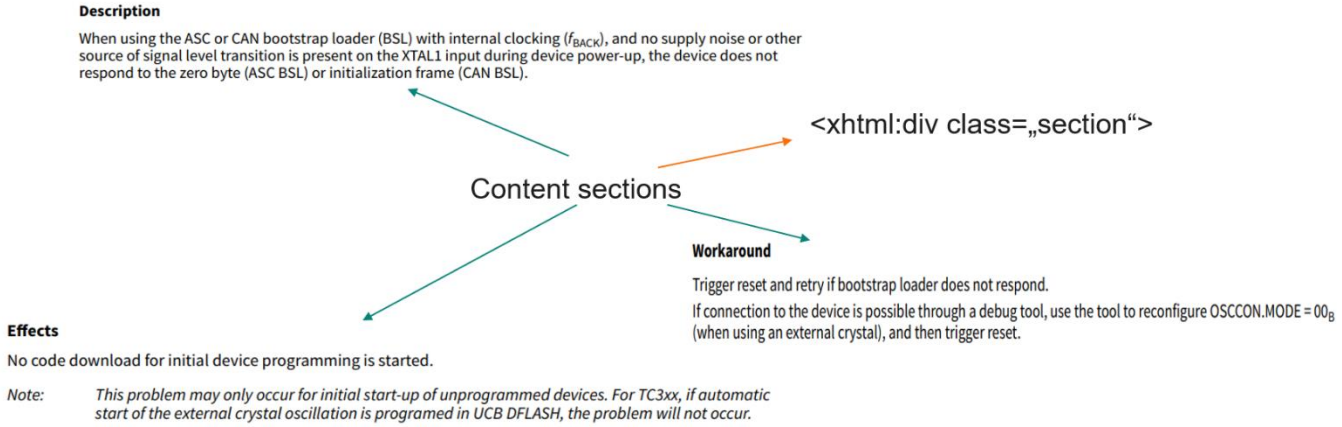


Figure 9 Errata sheet description mapping into ReqIF™

5 Outlook: Using ReqIF™ for automation

This section gives an overview of benefits when using digital documentation in the ReqIF™ format using most obvious applications. Additional use cases are possible.

5.1 Integration for requirement-driven development

As the documentation is provided as requirements in the ReqIF™ format, it can be imported into requirement tools. This provides linkable targets for requirement development based on Infineon microcontrollers. Main benefits here are finer granularity when it comes to traceability and less manual work.

Instead of creating linkable objects in a requirement environment out of the PDF version of the documentation, the documentation can simply be imported. This reduces the effort and time in going through the PDF document and manually migrating the content into the databases and is also less error prone due to little manual impact.

5.2 How to identify if a document update is affecting implemented applications

When making use of identifiable elements like registers, signals, or bitfields, tooling can be implemented to process a new delivery of a document. By identifying these structures, which are used by an implementation in user code or as part of the requirement engineering, it can easily be compared with a tool-based generated delta of deliveries. This helps to identify and highlight affected updates in an application. A significant benefit here is that the documents can be parsed by a program and basic information of affecting changes for different structures can be provided.

An example is when an application uses signal X and a register Y. Tool-based processing of the ReqIF™ delivery can easily identify if either X (signal) or Y (register) are affected without having to manually go through the whole document. As the documents can be imported into a requirement environment, this allows traceability of an application change due to documentation with already provided requirements.

5.3 Early feedback and alignment between customers and deliverables

By providing documentation in a machine-readable format, integration at the customer side can be improved, reducing possibility of errors introduced by manual work and allows easier traceability between different devices and versions.

Having this in an early stage of development, open points and mismatches between application requirements and device implementation can be identified. This enables early feedback and deeper development integration.

Revision history

Document revision	Date	Description of changes
V01_00	2023-12-07	Initial version

Trademarks
All referenced product or service names and trademarks are the property of their respective owners.

Edition 2023-12-07

Published by

Infineon Technologies AG

81726 Munich, Germany

© 2023 Infineon Technologies AG. All Rights Reserved.

Do you have a question about this document?

Email: erratum@infineon.com

Document reference

AN0000

Important notice

The information contained in this application note is given as a hint for the implementation of the product only and shall in no event be regarded as a description or warranty of a certain functionality, condition or quality of the product. Before implementation of the product, the recipient of this application note must verify any function and other technical information given herein in the real application. Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind (including without limitation warranties of non-infringement of intellectual property rights of any third party) with respect to any and all information given in this application note.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

Warnings

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.