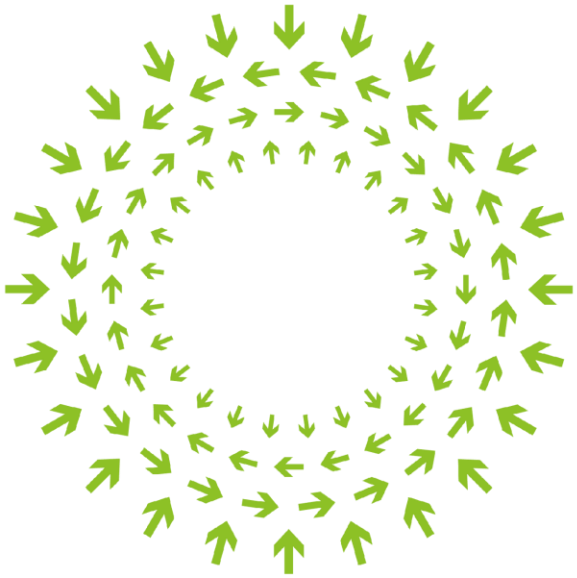

OPEN

Compute Project

Open Boot and Management Framework – Interface Consolidation Protocol (OBMF-ICP)

0.7.1

Base Specification

Effective XXXX, 2025

Table of Contents

1. License

THE UPDATED DEFAULT CONTRIBUTOR LICENSE AGREEMENT (CLA) IS **OWFa 0.9**. PLEASE VERIFY THE CORRECT CLA/FSA IS USED AND EXECUTED FOR THIS CONTRIBUTION.

1.1. Open Web Foundation (OWF) CLA

Contributions to this Specification are made under the terms and conditions set forth in **Modified Open Web Foundation Agreement 0.9 (OWFa 0.9)**. (As of October 16, 2024) ("Contribution License") by:

- Advanced Micro Devices
- Arm Limited
- ASPEED Technology, Inc.
- Intel Corporation
- Microsoft Corporation
- Nuvoton Technology Israel Ltd.
- NVIDIA

Usage of this Specification is governed by the terms and conditions set forth in **Modified OWFa 0.9 Final Specification Agreement (FSA)** (As of October 16, 2024) **("Specification License").**

You can review the applicable Specification License(s) referenced above by the contributors to this Specification on the OCP website at https://www.opencompute.org/contributions/templates-agreements.

For actual executed copies of either agreement, please contact OCP directly.

NOTWITHSTANDING THE FOREGOING LICENSES, THIS SPECIFICATION IS PROVIDED BY OCP "AS IS" AND OCP EXPRESSLY DISCLAIMS ANY WARRANTIES (EXPRESS, IMPLIED, OR OTHERWISE), INCLUDING IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR A PARTICULAR PURPOSE, OR TITLE, RELATED TO THE SPECIFICATION. NOTICE IS HEREBY GIVEN, THAT OTHER RIGHTS NOT GRANTED AS SET FORTH ABOVE, INCLUDING WITHOUT LIMITATION, RIGHTS OF THIRD PARTIES WHO DID NOT EXECUTE THE ABOVE LICENSES, MAY BE IMPLICATED BY THE IMPLEMENTATION OF OR COMPLIANCE WITH THIS SPECIFICATION. OCP IS NOT RESPONSIBLE FOR IDENTIFYING RIGHTS FOR WHICH A LICENSE MAY BE REQUIRED IN ORDER TO IMPLEMENT THIS SPECIFICATION. THE ENTIRE RISK AS TO IMPLEMENTING OR OTHERWISE USING THE SPECIFICATION IS ASSUMED BY YOU. IN NO EVENT WILL OCP BE LIABLE TO YOU FOR ANY MONETARY DAMAGES WITH RESPECT TO ANY CLAIMS RELATED TO, OR ARISING OUT OF YOUR USE OF THIS SPECIFICATION, INCLUDING BUT NOT LIMITED TO ANY LIABILITY FOR LOST PROFITS OR ANY CONSEQUENTIAL, INCIDENTAL, INDIRECT, SPECIAL OR PUNITIVE DAMAGES OF ANY CHARACTER FROM ANY CAUSES OF ACTION OF ANY KIND WITH RESPECT TO THIS

SPECIFICATION, WHETHER BASED ON BREACH OF CONTRACT, TORT (INCLUDING NEGLIGENCE), OR OTHERWISE, AND EVEN IF OCP HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

1.2. Trademarks

Names and brands may be claimed as trademarks by their respective companies. I2C® is a trademark of NXP Semiconductor. PCIe® and PCI Express® are the registered trademarks of PCI-SIG. I3C is a trademark of MIPI Alliance, Inc.

2. References

References to other standards:

- SMBus Management Interface Forum. System Management Bus (SMBus) Specification. System Management Interface Forum, Inc, Version 2.0, August 3rd, 2000
- MIPI alliance Specification for I3C Basic^{SM} Version 1.1.1. – Released July 2021
- USB Implementers Forum. Universal Serial Bus Specification, Revision 2.0, April 27th, 2000
- PCI-SIG PCI Express Base Specification Revision 6.3, January 15, 2025
- DMTF Standard. Management Component Transport Protocol (MCTP) SMBus/I2C Transport Binding Specification, Rev 1.2.0, Apr 6, 2020
- DMTF Standard. DSP0283, Management Component Transport Protocol (MCTP) Universal Serial Bus (USB) Transport Binding Specification, Rev 1.0.1, May 21st, 2024
- DMTF Standard. DSP0233, Management Component Transport Protocol (MCTP) I3C Transport Binding Specification, Rev 1.0.1, Mar 25, 2024
- DMTF Standard. DSP0283, Management Component Transport Protocol (MCTP) PCIe VDM Transport Binding Specification, Rev 1.3.0, Oct 4, 2024

References to other OBMF-specific documents (all documents summarized on https://github.com/opencomputeproject/ocp-obmf/wiki):

- DMTF Standard. DSP0295, OBMF-ICP over Management Component Transport Protocol (MCTP) Binding Specification, to be defined by DMTF (not available at the time of publication of this document)
- OBMF-ICP over USB Class Specification. https://github.com/opencomputeproject/ocp-obmf/blob/main/OBMF-ICP%20over%20USB/OBMF-ICP%20over%20USB%20revision%200.7.pdf
- [RDL_FILES] OBMF-ICP RDL Channel definitions: https://github.com/opencomputeproject/ocp-obmf/tree/main/RDL_Files

3. Acronyms

For the purposes of the DC-SCM specification, the following acronyms apply:

Acronym	Definition
ASIC	Application Specific Integrated Circuit
BMC	Baseboard Management Controller
DC-SCM	Data Center Secure Control Module
DC-SCI	Data Center Secure Control Interface
ESPI	Enhanced Serial Peripheral Interface
FRU	Field Replaceable Unit
I/O	Input / Output
HPM	Host Processor Module
I2C	Inter-Integrated Circuit - two wire serial protocol
I3C	MIPI Alliance Improved Inter-Integrated Circuit – two wire serial protocol
LPC	Low Pin Count bus
OCP	Open Compute Project

4. Change Log

Date	Revision	Comment
10-Oct-2025	0.7	Initial public revision (draft)
30-Oct-2025	0.7.1	Minor editorial corrections: • Arm missed on the contributors list • Security protections are out-of-scope of this spec

5. Goals

OBMF-ICP specification has been driven by the need to standardize the protocol for use during boot, configuration, and runtime communication between Host Processor or other devices on the Host Processor Module (HPM) and Baseboard Management Controller (BMC) on the DC-SCM Module. But OBMF-ICP is not tied to DC-SCM and can be used in other circumstances when consolidation of other interfaces is necessary.

OBMF-ICP protocol shall meet the following goals:

- Host Processor (CPU) vendor agnostic
- Applicable to non-DC-SCM use cases
- Reusable for other devices e.g. PCIe Cards, CXL Cards, etc.
- Should be physical interface agnostic protocol (e.g., MCTP/PLDM)
- Recommended physical interface choices for DC-SCM use cases
- Supporting adoption model for vendor specific features (vendor extensions)
- Leveraging OCP standards & DMTF standards
- Align to existing security standards and requirements (e.g., SPDM)

OBMF-ICP allows establishing multiple communication channels, each of them transporting a different use case over a single physical connection. Use case examples are: TPM, RTC, UART, or BIOS POST code reporting. OBMF-ICP can be used to consolidate these various types of traffic.

Cryptographic content protection, such as encryption, is out of scope of OBMF-ICP specification. Every protocol that requires protection, such as TPM, should provide their own protections against the targeted threats (such as man-in-the-middle attacks) or use pre-existing mechanisms, such as wrapping the OBMF-ICP communication in a SPDM session.

6. Requirements

The following are the key requirements of the OBMF-ICP protocol:

No.	Title	Description
1	Host Processor (CPU) vendor agnostic	OBMF-ICP shall be agnostic to the host processor (CPU) vendor, ensuring compatibility across different processor architectures without requiring vendor-specific adaptations.
2	Applicable to non-DC-SCM use cases	OBMF-ICP shall be applicable in DC-SCM and non-DC-SCM based designs.
3	Reusable for other devices	OBMF-ICP shall be applicable across multiple device types e.g. GPUs, PCIe Cards, CXL Cards, etc., ensuring consistency and minimizing redundant implementations.
4	Physical interface agnostic protocol	OBMF-ICP shall define a protocol that is independent of the underlying physical interface.
5	Transport layer agnostic	OBMF-ICP protocol shall be defined independently of the transport layer, with specific bindings defined for each supported transport (e.g., MCTP via DMTF specifications).

6	Recommended physical interface choices for DC-SCM	OBMF-ICP should provide recommended physical interface options specifically tailored for DC-SCM implementations (e.g., USB, I3C), while maintaining broader interface agnosticism.
7	Supporting adoption model for vendor specific features (vendor extensions)	OBMF-ICP shall support a structured adoption model for vendor-specific extensions, allowing proprietary features to coexist with standard functionality.
8	Leveraging OCP standards & DMTF standards	OBMF-ICP shall leverage existing OCP and DMTF standards, with clearly defined dependencies and integration points to ensure alignment and interoperability.
9	Align to existing security standards and requirements	OBMF-ICP shall align with existing industry security standards, such as SPDM, to ensure secure communication and system integrity.
10	Support Multi-Node and Multi-Partitions architectures	OBMF-ICP shall support multi-node and multi-partition system architectures in alignment with DC-SCM 3.0 specifications.
11	Request-response model	OBMF-ICP shall employ a request-response communication model to ensure synchronous interaction between applications and components.
12	Multiple communication channels between different applications	OBMF-ICP shall support multiple independent communication channels between different applications operating on the same endpoint. Delays in one application shall not block communications in other channels. Application/channels exampples: - TPM - UART - GPIO
13	One outstanding request per channel	OBMF-ICP shall allow only one outstanding request per communication channel to simplify message management and ensure predictable behavior.
14	Low complexity allowing HW-offload options.	OBMF-ICP shall be designed with low complexity to enable hardware offload and efficient implementation in constrained environments.
15	Prioritization of channels	OBMF-ICP shall allow prioritization of communication channels. The enforcement of prioritization policies shall be implementation-dependent.

16	Latency constrains and mitigations for time-sensitive channels	OBMF-ICP should define timing and latency evaluation, especially for performance-sensitive applications such as TPM and UART.
17	No changes required to Host BIOS/OS/Drivers	OBMF-ICP shall not require modifications to host BIOS, operating systems, or drivers for interfaces it consolidates, ensuring seamless integration.
18	Leveraging existing security standards	OBMF-ICP shall not define native security solutions. Existing solutions such as SPDM shall be used to provide secure channels when needed.
19	Leveraging existing security protection for TPM traffic	If secure communication is required for the TPM channel, OBMF-ICP shall rely on Project Kirkland to ensure traffic protection.
10	Open: Recommended OBMF-ICP driver model	OBMF-ICP specification should include or be accompanied by an application note or driver model recommendation to guide implementation.
21	Open: Predefined standard channels	OBMF-ICP specification should define a list of standardized channels such as UART, Virtual Wires etc. and allow for extensions with vendor-specific channels.
22	Leveraging existing security protection for TPM traffic	If secure communication is required for the TPM channel, OBMF-ICP shall rely on Project Kirkland to ensure traffic protection.
23	64kB message sizes	OBMF-ICP shall allow exchanging application messages of size up to 64 kB.

7. Assumptions

The following are the key assumptions about the environment or use cases:

1. The underlying OBMF-ICP transport shall guarantee message delivery, that is, OBMF-ICP messages shall not be dropped nor duplicated.
2. The underlying OBMF-ICP transport shall guarantee in-order delivery, that is, OBMF-ICP messages are never reordered.
3. The underlying OBMF-ICP transport shall guarantee message integrity.
4. OBMF-ICP does not define priority differentiation mechanisms. If needed, the underlying OBMF-ICP transport layer can be used to differentiate priorities.

Note that MCTP alone (DSP0236 and related standard protocols) does not provide the above guarantees. For this reason, on top of OBMF-ICP over MCTP Binding, there is a MCTP-specific OBMF-ICP reliable delivery layer in the protocol stack – see Figure 1.

8. Introduction

8.1. OBMF-ICP Protocol Stack Overview

This specification defines the OBMF-ICP protocol that is part of a protocol stack as shown in the red color in Figure 1:

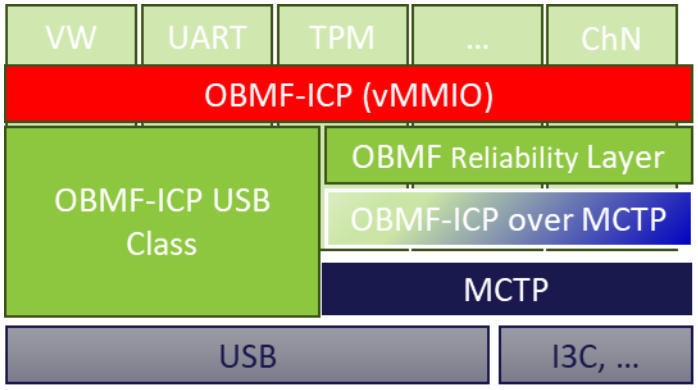

Figure 1: OBMF-ICP Protocol Stack

This specification defines message formats and encoding of the OBMF-ICP layer and relies on the underlying transport specifications, such as:

- OBMF-ICP USB Class specification (please see https://github.com/opencomputeproject/ocp-obmf/wiki for access) – defines how OBMF-ICP messages can be carried over USB bus,
- OBMF-ICP over MCTP protocol stack (please see https://github.com/opencomputeproject/ocp-obmf/wiki for access) – multiple specifications define how OBMF-ICP messages can be carried over any MCTP-capable bus,
- collateral RDL files (please see at [RDL_FILES] for access) – define specific structures that are being modeled to support specific use cases, such as UART, TPM, VW (Virtual Wires), etc.

Figure 2 shows the specification dependencies. Documents published by OCP OBMF workstream as part of HW Management Project are marked with the appropriate symbol:

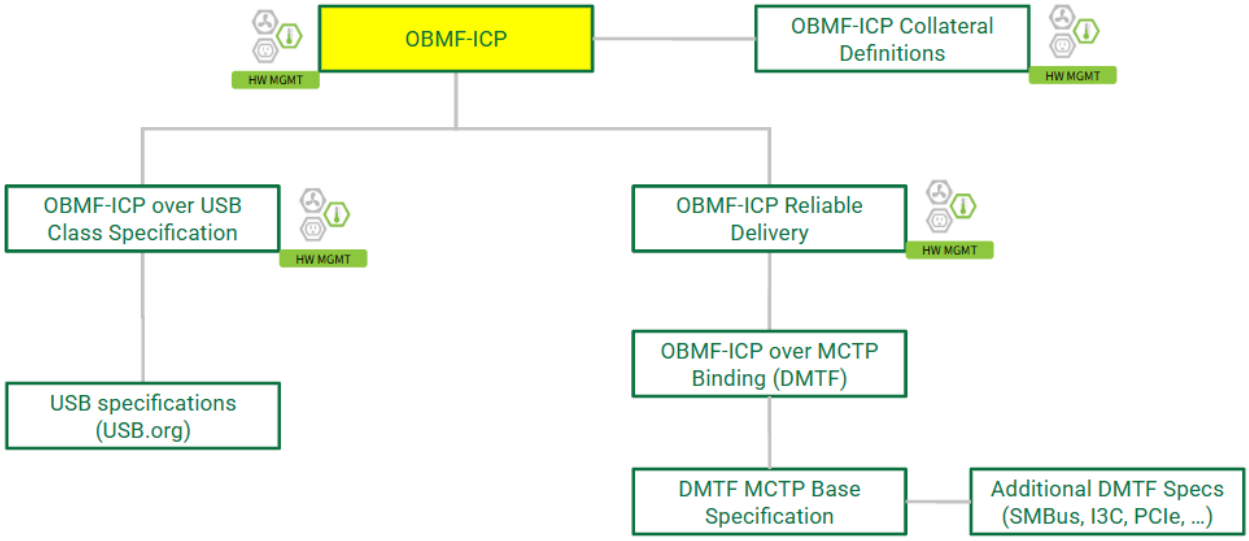

Figure 2: Specifications related to OBMF-ICP

8.2. OBMF-ICP Component Roles

OBMF-ICP is designed to enable communication between two entities that are traditionally connected using a multitude of legacy interfaces. OBMF-ICP replaces all these legacy interfaces with a new protocol.

The two communicating entities will be referred to as:

- **OBMF-ICP Primary** – typically a platform management controller (BMC)
- **OBMF-ICP Secondary** – such as a processor or a GPU device

Figure 3 shows the typical platform topology with OBMF-ICP Primary and OBMF-ICP Secondary roles:

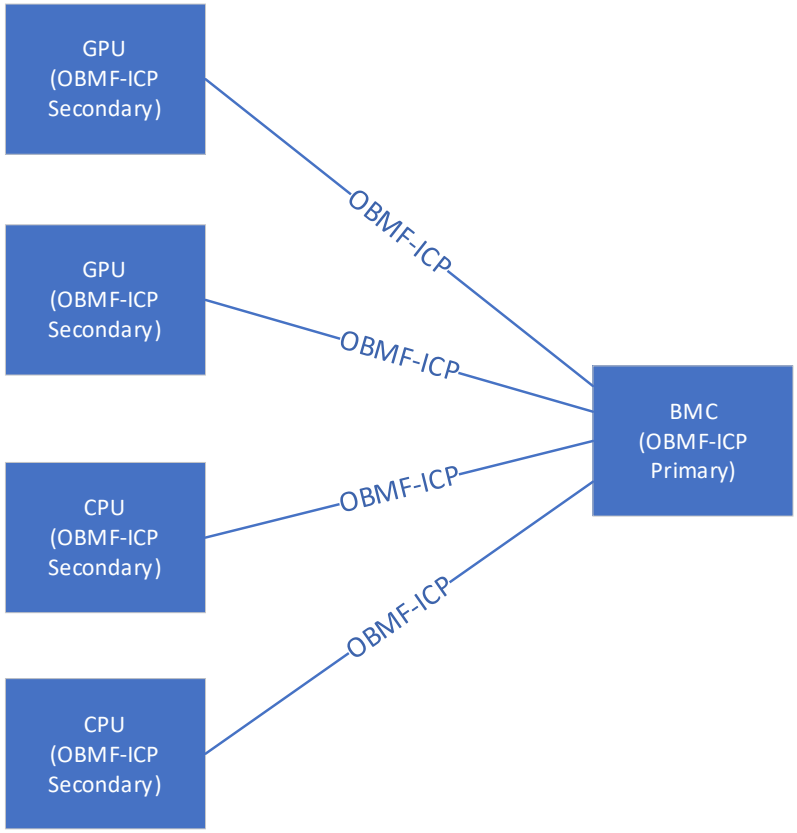

Figure 3: Typical OBMF-ICP Primary and OBMF-ICP Secondary Roles

The legacy interfaces are now replaced with OBMF-ICP Channels. For example, there may be an OBMF-ICP Channel to provide Real-Time Clock (RTC) access: a processor can ask the BMC to provide the current RTC information by reading data from a well-defined RTC data structure maintained by the BMC. For each channel, OBMF-ICP distinguishes two roles:

- **OBMF-ICP Producer** – hosts a well-defined data structure or a service represented by this structure
- **OBMF-ICP Consumer** – uses the service provided by the OBMF-ICP Producer

In the example about RTC service above, BMC would be the OBMF-ICP Producer while the processor would be the OBMF-ICP Consumer. Note that at the same time the roles may be reversed for a different channel – for example, a processor may be an OBMF-ICP Producer or FRU information and BMC may be requesting FRU information as an OBMF-ICP Consumer.

OBMF-ICP channel list and supported services are implementation dependent, with one exception: there is a special Control Channel 0 that must always be implemented. For Channel 0, OBMF-ICP Secondary device (such as a processor or a GPU) acts as OBMF-ICP Producer.

Channel 0 is used to communicate the list of other expected channels and any other important discovery information.

This specification defines a set of standard data structure definitions (see Appendix I: OBMF-ICP Standard Channel Definitions but also published in the form of RDL files available at [RDL_FILES]) that typical implementations would follow. A specific vendor may define and publish (preferably using the same RDL format) additional extensions. The published definitions (standard and vendor-defined extensions) allow vendors (typically BMC HW/FW vendors) to implement the proper support of necessary services.

The concept of channels and associated data structures is shown in Figure 4:

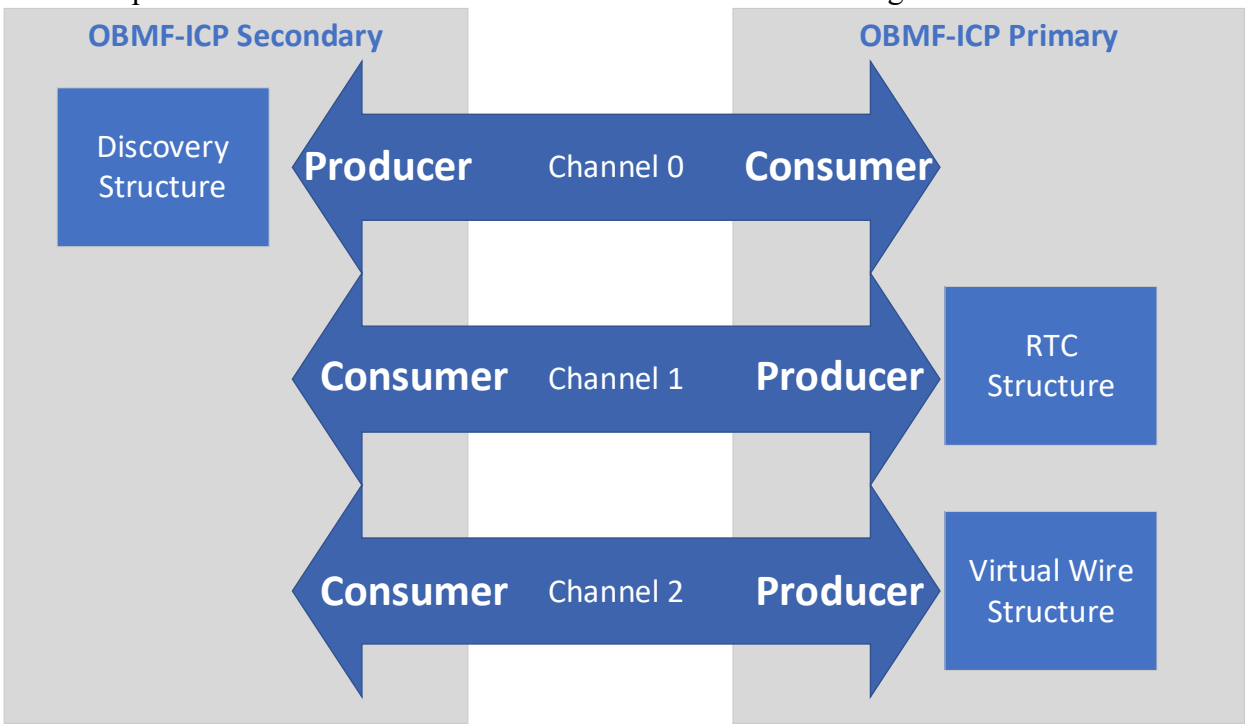

Figure 4: Concept of OBMF-ICP Channels and Data Structures

In order to enable data exchange, OBMF-ICP Protocol defines a set of Read and Write transactions that may be used to read or write the content of these data structures for each channel – see section 9.1. For each transaction, there are two phases of communication and we can define two roles of the communicating entities:

- **OBMF-ICP Requester** – the entity that initiates the read or write request
- **OBMF-ICP Responder** – the entity that handles the request and provides a response

In the majority of situations, it is the OBMF-ICP Consumer that is the OBMF-ICP Requester: initiates the Read or Write requests to access the service. However, some OBMF-ICP Channels also allow the OBMF-ICP Producer to initiate the Write request to the OBMF-ICP Consumer in order to notify about some asynchronous changes. For example, consider a BMC acting as

OBMF-ICP Producer to provide Virtual Wire (VW) channel service to a processor (OBMF-ICP Consumer). The processor may want to initiate Read transactions to read the state of an input wire. It may also initiate a Write transaction to update the output state of another wire. Additionally, BMC may also initiate a Write transaction to notify the processor about GPIO state changes. This example is shown in Figure 5:

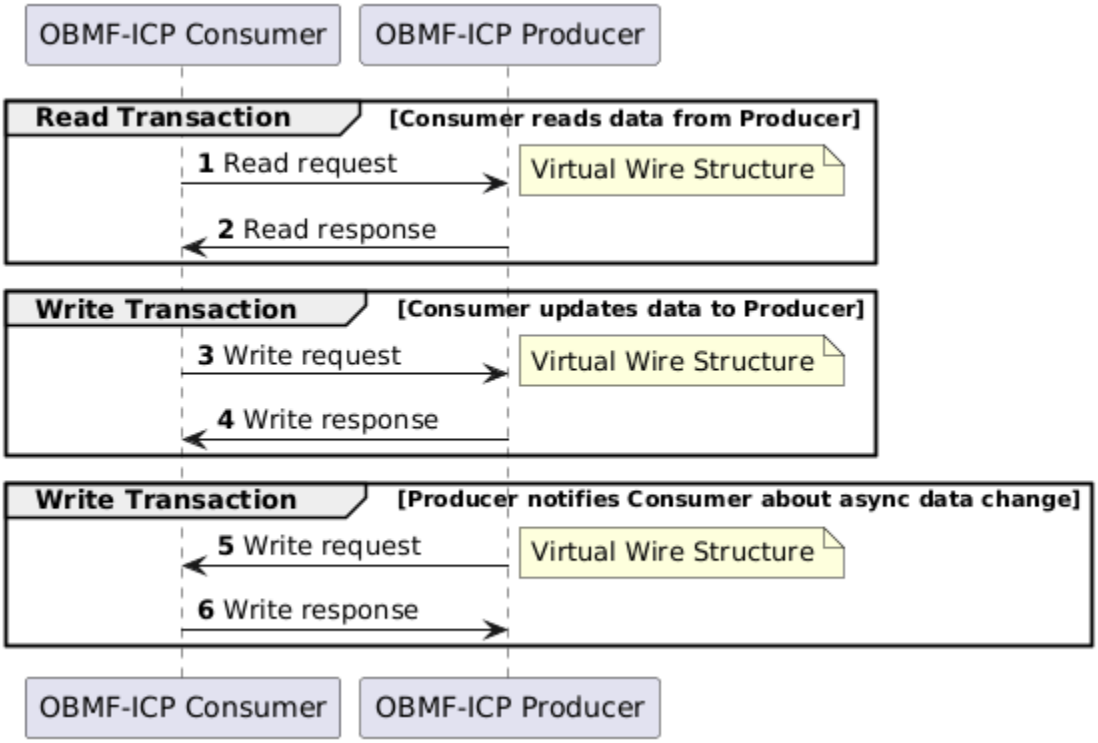

Figure 5: Sample Read and Write Transaction Uses

8.3. Typical Use Case Example

Summarizing, the typical use case is as follows:

Precondition:

1. OBMF-ICP Secondary device vendor follows the published OBMF-ICP standard definitions (see [RDL_FILES]) or defines and publishes their own vendor-defined extensions.
2. BMC HW/FW (OBMF-ICP Primary) implements services to meet the defined expectations.

System operation sequence:

3. Device (OBMF-ICP Secondary) and BMC (OBMF-ICP Primary) discover and negotiate their capabilities, especially the set of OBMF-ICP channels/services expected by the

OBMF-ICP Secondary – please see section 10.1 for more information. Note that some channels/services may be mandatory, while some may be optional.

4. BMC enables the necessary OBMF-ICP channels.
5. OBMF-ICP Secondary device uses the available services by sending OBMF-ICP requests on a specific OBMF-ICP channel. These are Read or Write accesses, as defined in section 9, to the data structures defined with RDL. BMC handles the Read or Write requests and provides requested responses.
6. If supported by a specific channel/service, the Device can enable notifications (see section 10.1). That is, if data is updated on the BMC side, a notification (in a form of a write to the structure location) is sent by the BMC to the device.

9. OBMF-ICP Protocol Definition

As stated in section 8.2, OBMF-ICP protocol defines the concept of channels to enable bi-directional communication between an OBMF-ICP Producer and an OBMF-ICP Consumer. Each OBMF-ICP Channel supports a service offered by OBMF-ICP Producer – each service is represented by a data structure (a register map) that OBMF-ICP Producer and OBMF-ICP Consumer share.

Each channel is independent, that is, transactions initiated by OBMF-ICP Requester on one channel are expected to be serviced by the OBMF-ICP Responder independently of the state of the other channels. For example, a processor acting as OBMF-ICP Consumer may request services over RTC channel, while waiting for a previous request to be fulfilled over a Flash access channel.

Standard channel definitions are published in Appendix I: OBMF-ICP Standard Channel Definitions and can be downloaded from [RDL_Files].

9.1. OBMF-ICP Message Format

The following types of messages are defined by OBMF-ICP protocol:

- Read – allows the requester to get content of data offered by a service
- Write – allows the requester to set content of writeable data offered by a service

As explained in more detail in section 8.2, one party initiates a Read request to a data structure location and the other party is expected to handle the request and provide a Read Response. Note that a specific channel may allow Read requests in one direction only. Similarly, if allowed by a specific OBMF-ICP Channel, the same interaction applies to Write transactions. Write transactions, if allowed by a specific channel, may be sent in the other direction as an asynchronous notification.

OBMF-ICP Message format is defined in the table below. OBMF-ICP messages include header and message payload as shown in Figure 6:

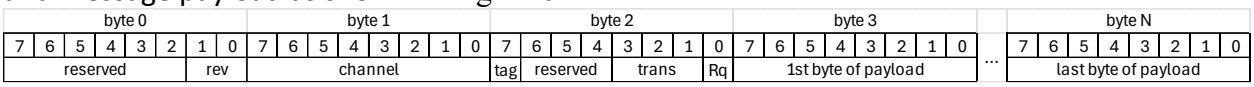

Figure 6: OBMF-ICP Message Format

Field definitions are as follows:

Bytes[Bits]	Field	Description
0[1:0]	OBMF-ICP Revision	OBMF-ICP revision number • 0: protocol defined in this document Remaining values are reserved for future use.
0[7:2]	Reserved	Reserved
1[7:0]	OBMF-ICP Channel	OBMF-ICP Channel number: • Channel zero is reserved for discovery purposes – see section 11.1 All other channel assignments are as per descriptor definitions.
2[0]	RqResp	Indicates wheter the OBMF-ICP message is a request or a response: • 0: Request • 1: Response
2[3:1]	OBMF-ICP Transaction	The following transaction types are defined: • 0: Short Read • 1: Short Write • 2: Long Read • 3: Long Write • 4: Short Notify • 5: Long Notify
2[6:4]	Reserved	Reserved
2[7]	Tag	Subsequent transactions must alternate the value between 0 and 1.
3:N	Payload	Payload related to each type of OBMF-ICP transaction as defined in subsequent sections. Note: Payload size is detemined by the size of the transport layer message.

The following subsections define the Payload portion the Read and Write requests and responses. Note that each message has a short and a long format. Short format is used when the actual Read or Write payload is shorter than 256 bytes, while the Long format is used when the payloads are longer.

9.2. Short Read Transaction

9.2.1. Short Read Request

OBMF-ICP Read Request message field definitions are as follows:

Bytes[Bits]	Field	Description
7:0[63:0]	Address	64-bit address of data to read (no alignment restrictions). Address definiitons for each OBMF-ICP Channel are defined in [RDL_FILES] (as well as included in Appendix I of this document).
8[7:0]	Size	Read size. Maximum size is discovered using Channel 0 Discovery Structure

9.2.2. Short Read Response

OBMF-ICP Read Response message field definitions are as follows:

Bytes[Bits]	Field	Description
0	Status	Indicates the completion status. Completion codes are defined in section 9.6
N:1	Data	If Status = Success, this field carries *Size* bytes of the MMIO Data returned in response to the request If Status <> Success, this field is omitted

9.3. Short Write Transaction

9.3.1. Short Write Request

OBMF-ICP Write Request message field definitions are as follows:

Bytes[Bits]	Field	Description
7:0	Address	64-bit address of data to write (no alignment restrictions). Address definiitons for each OBMF-ICP Channel are defined in [RDL_FILES] (as well as included in Appendix I of this document).
8[7:0]	Size	Write size (no alignment restrictions). Maximum size is discovered using Channel 0 Discovery Structure
9+Size-1:9	Data	Data to be written at the address location

9.3.2. Short Write Response

OBMF-ICP Write Response message field definitions are as follows:

Bytes[Bits]	Field	Description
0	Status	Indicates the completion status. Completion codes are defined in section 9.6

9.4. Long Read Transaction

9.4.1. Long Read Request

OBMF-ICP Read Request message field definitions are as follows:

Bytes[Bits]	Field	Description
7:0[63:0]	Address	64-bit address of data to read (no alignment restrictions). Address definiitons for each OBMF-ICP Channel are defined in [RDL_FILES] (as well as included in Appendix I of this document).
9:8	Size	Read size. Maximum size is discovered using Channel 0 Discovery Structure

9.4.2. Long Read Response

OBMF-ICP Read Response message field definitions are as follows:

Bytes[Bits]	Field	Description
0	Status	Indicates the completion status. Completion codes are defined in section 9.6
N:1	Data	If Status = Success, this field carries *Size* bytes of the MMIO Data returned in response to the request If Status <> Success, this field is omitted

9.5. Long Write Transaction

9.5.1. Long Write Request

OBMF-ICP Write Request message field definitions are as follows:

Bytes[Bits]	Field	Description
7:0[63:0]	Address	64-bit address of data to read (no alignment restrictions). Address definiitons for each OBMF-ICP Channel are defined in [RDL_FILES] (as well as included in Appendix I of this document).
9:8	Size	Write size. Maximum size is discovered using Channel 0 Discovery Structure
10+Size-1:10	Data	Data to be written at the address location

9.5.2. Long Write Response

OBMF-ICP Write Response message field definitions are as follows:

Bytes[Bits]	Field	Description
0	Status	Indicates the completion status. Completion codes are defined in section 9.6.

9.6. Completion Codes

Common completion codes used in OBMF-ICP protocol are:

- 0x00: Success – see Data field for requested MMIO data
- 0x01: Unknown OBMF-ICP Channel / Channel not supported
- 0x02: Command not supported for this Channel
- 0x03: Permanent error for this Channel
- 0x04: Channel not ready at this moment
- 0x05: Request rejected due to insufficient privilege
- 0x06: Address Out-of-range
- 0x07: Other error (used when more specific codes are not available)

TBD: do we need any other error codes?

10. OBMF-ICP Protocol Flows and Sequences

10.1. Initialization Flow

This section documents the steps to be followed during initialization. Initialization relies on Channel 0 – Discovery Structure. This description assumes that the OBMF-ICP Secondary device is the OBMF-ICP Producer for Channel 0 while the OBMF-ICP Primary device is typically the OBMF-ICP Consumer on Channel 0.

Precondition: the transport layer is initialized (e.g., USB bus is initialized and USB devices are discovered and enumerated) in order to enable OBMF-ICP communication.

1. As the first step, OBMF-ICP Primary should use Channel 0 Reads to retrieve the Discovery Structure from OBMF-ICP Secondary. Discovery Structure allows to verify/discover:
    a. OBMF-ICP Revision – compatibility verification

   b. Expected channel list. This includes information which channels are OBMF-ICP Secondary requires (mandatory channels) and which channels are just optional.
   c. Maximum transaction sizes – OBMF-ICP requires reads and writes to allow 64 byte payloads by default but implementations can optionally increase the sizes to optimize communication
2. OBMF-ICP Primary shall perform the following for each OBMF-ICP Channel it supports:
   a. Verify channel definition, especially the definition revision
   b. Enable the selected channel – this is done by Writing to the "Enabled" fields in the Discovery Structure

OPEN: This means that the Secondary device will have to support Write transactions for Channel 0. We could allow channels to be enabled by default and this would allow simplifying the device implementations.

3. Once a channel is enabled:
   a. The Secondary device (usually the OBMF-ICP Consumer for other channels) can start using the read and write transactions to access the services via other channels.
   b. If desired, and if the channel definition supports this, OBMF-ICP Consumer may enable notifications. This allows OBMF-ICP Producer to initiate Write transactions to provide asynchronous updates to the Consumer. Only selected channels define such notifications.

The above flow is illustrated in Figure 7:

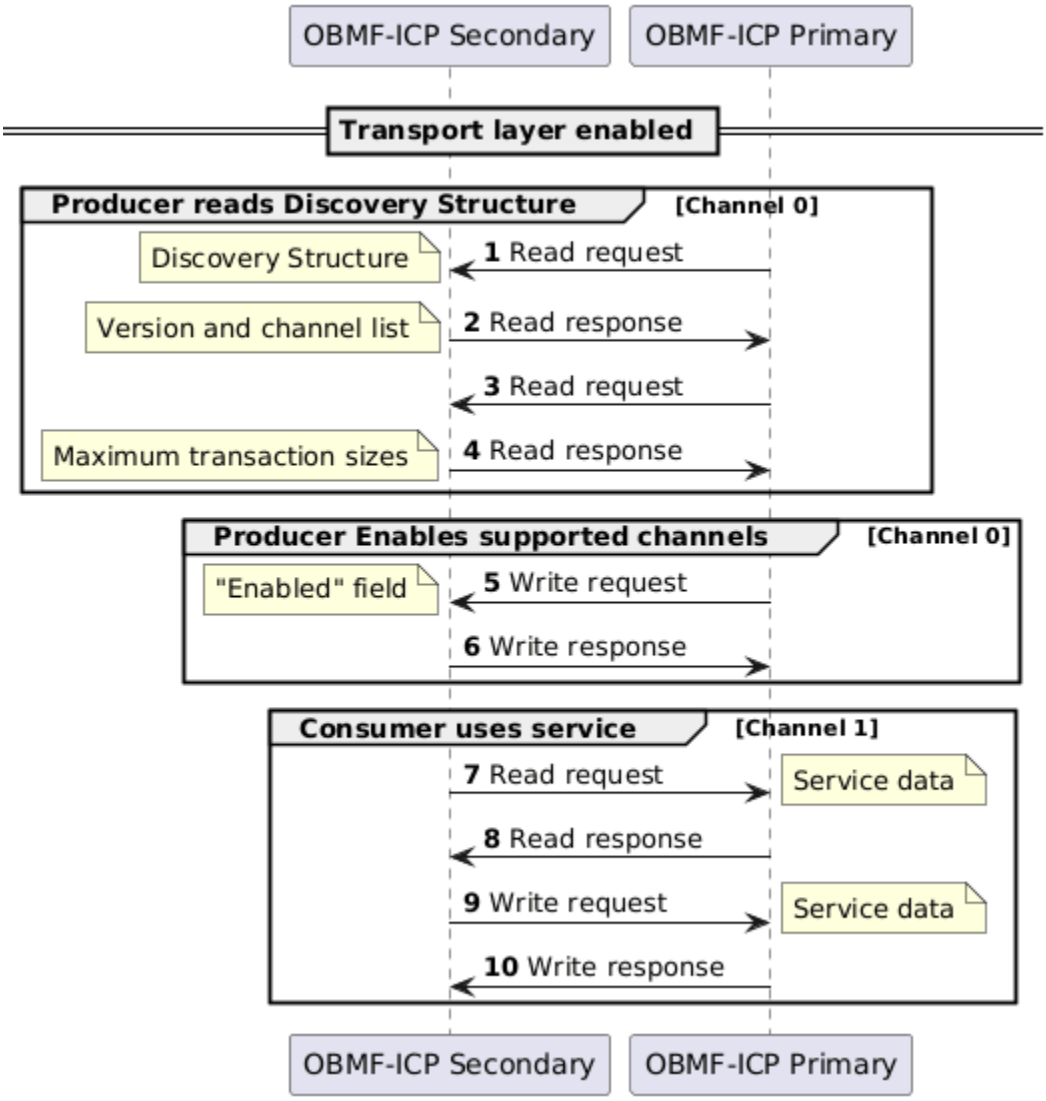

Figure 7: Initial Sequence

10.2. Communication Flows

OBMF-ICP Requester shall follow the following rules:

1. After sending one request for a specific OBMF-ICP Channel, the Requester waits for the corresponding response before sending another request for the same OBMF-ICP Channel. In other words, only one outstanding request is allowed per OBMF-ICP Channel.

2. OBMF-ICP Channels are independent, meaning that the Requester may send a new request on a channel, provided there is no outstanding request on this channel, even if there are outstanding requests on other channels.
3. Communication in the opposing directions (i.e., Notify messages vs read/write messages) are independent. That is, Notify messages may be sent regardless whether there is an outstanding Read/Write request in the opposing direction.
4. Each OBMF-ICP Responder operates independently, so the Requester may have concurrent outstanding requests to different OBMF-ICP Responders.
5. When sending the request, the first request message on a channel in a given direction shall use Tag = zero and the subsequent requests shall use alternating Tag values.
6. The response messages are expected to have the same Tag value as in the request. If a non-matching Tag is detected, it should shutdown the channel as it indicates a fatal error.

OBMF-ICP Responder takes the following actions when a new request arrives:

1. Upon receiving a request, OBMF-ICP Responder shall verify if the Tag value matches the expectation on a given Channel: the first message on a channel in a given direction shall use Tag = zero and the subsequent requests shall use alternating Tag values. If a non-matching Tag is detected, it should shutdown the channel as it indicates a fatal error.
2. If Tag the received request is matching the expected value, the OBMF-ICP Request is delivered to the application layer for processing
3. After the application layer prepares the response, it is sent back with the same Tag value as in the request.

Communication in each direction over a specific channel is independent – for example, if a Write request is pending in one direction, this should not stop sending a Write request in the other direction (assuming such Write requests are generally allowed in both directions for this specific channel).

10.3. Channel “Not Ready”

Services that are tunneled over OBMF-ICP may be in two general states:

- Not-ready
- Ready

“Not ready” state is reported as a completion code and it indicates that the service on this channel is not available at a given moment. It may be an expected state (e.g. low power state) or it may be due to an error. In the “ready” state, all the other completion codes defined in in section 9.6 apply.

10.4. Transport Guarantees

OBMF-ICP does not implement integrity verification mechanisms not mechanisms to detect and/or recover an unresponsive or misbehaving communication endpoint. OBMF-ICP depends on the underlying transport layers to provide guaranteed service.

OBMF-ICP does not implement timeouts. In order to deal with scenarios such as no response from the communicating endpoint, timeouts may need to be implemented by the upper layers that use OBMF-ICP (that is typically a practice today, e.g. over eSPI).

If the underlying transport reports a fatal error (e.g. USB endpoint disconnected), OBMF-ICP implementation should report the error to the higher layers in an implementation specific way.

11. Appendix I: OBMF-ICP Standard Channel Definitions

~~This section lists the normative definition of each type of channel.~~ *Note: Prior to revision 1.0, this draft document only includes the sample structures. Normative content is still to be created.*

Note: The content of this section has been generated from RDL files. It may be more convenient for the reader to use the original RDL files published at [RDL_FILES] . Markup or HTML content generated from RDL is also available at the same location.

11.1. OBMF-ICP Control Channel – Channel 0

This structure must be provided by the OBMF-ICP Secondary on Channel 0. The Primary can read this content to obtain the list of all the other channels that the OBMF-ICP Secondary expects to be supported by the Primary. Some of these channels may be mandatory, while some are optional. The Primary should enable all the necessary channels by writing to the "Enable" fields in this data structure. In addition to the channel list, this discovery structure allows OBMF-ICP endpoints to establish OBMF-ICP protocol version compatibility and other capabilities.

Offset	Identifier	Name
0x000	OBMF_VER	OBMF-ICP Protocol Version
0x008	READ_SIZE	Maximum Read Size
0x010	WRITE_SIZE	Maximum Write Size
0x018	MAX_CHANNEL_NO	Maximum Channel Number
0x100	CHANNEL_1	Channel 1
0x200	CHANNEL_2	Channel 2
0x300	CHANNEL_3	Channel 3
0x400	CHANNEL_4	Channel 4
0x500	CHANNEL_5	Channel 5

11.1.1. OBMF_VER register

- Absolute Address: 0x0
- Base Offset: 0x0
- Size: 0x4

OBMF-ICP Protocol Version supported by the device

Bits	Identifier	Access	Reset	Name
31:0	OBMF_VER	r	0x0	—

- **OBMF_VER field**

OBMF-ICP Protocol Version supported by the device

11.1.2. READ_SIZE register

- Absolute Address: 0x8
- Base Offset: 0x8
- Size: 0x8

Defines the maximum Read response payload size that should be used. READ_SIZE_PRI and READ_SIZE_SEC define the capabilities of both parties and the actual size used is the minimum of both values. The same size is used in both directions of communication.

Bits	Identifier	Access	Reset	Name
31:0	READ_SIZE_SEC	r	0x40	—
63:32	READ_SIZE_PRI	rw	0x40	—

- **READ_SIZE_SEC field**

Indicates the total number of bytes for a Read transaction that the OBMF-ICP Secondary supports. OBMF-ICP Primary must obey this limit.

- **READ_SIZE_PRI field**

Indicates the total number of bytes for a Read transaction that the OBMF-ICP Primary supports. A minimum size that every OBMF-ICP Primary shall support is 64-bytes and this should be the default value of this register. OBMF-ICP Primary may write to this field to allows larger Read sizes to be initiated by OBMF-ICP Secondary.

11.1.3. WRITE_SIZE register

- Absolute Address: 0x10
- Base Offset: 0x10
- Size: 0x8

Defines the maximum Write request payload size that should be used. WRITE_SIZE_PRI and WRITE_SIZE_SEC define the capabilities of both parties and the actual size used is the minimum of both values. The same size is used in both directions of communication.

Bits	Identifier	Access	Reset	Name
31:0	WRITE_SIZE_SEC	r	0x40	—
63:32	WRITE_SIZE_PRI	rw	0x40	—

- **WRITE_SIZE_SEC field**

Indicates the total number of bytes for a Write transaction that the OBMF-ICP Secondary supports.

- **WRITE_SIZE_PRI field**

Indicates the total number of bytes for a Write transaction that the OBMF-ICP Primary supports. A mimimum size that every OBMF-ICP Primary shall support is 64-bytes and this should be the default value of this register.

11.1.4. MAX_CHANNEL_NO register

- Absolute Address: 0x18
- Base Offset: 0x18
- Size: 0x4

The last channel number listed in the channel list of this Discovery Structure. Note that Channel 0 is reserved for Control Channel and does not require discovery while all other channels are sequentially numbered from 1 to MAX_CHANNEL_NO.

Bits	Identifier	Access	Reset	Name
31:0	MAX_CHANNEL_NO	r	0x0	—

- **MAX_CHANNEL_NO field**

The last channel number listed in the channel list of this Discovery Structure. Note that channel 0 is reserved for Control Channel and does not require discovery while all other channels are numbered from 1 to MAX_CHANNEL_NO.

11.1.5. CHANNEL_1 register file

- Absolute Address: 0x100
- Base Offset: 0x100
- Size: 0x14

Channel 1 Discovery Structure

Offset	Identifier	Name
0x00	CHANNEL_GUID	Channel GUID
0x10	CHANNEL_CFG	Channel Configuration

11.1.5.1. CHANNEL_GUID register

- Absolute Address: 0x100
- Base Offset: 0x0
- Size: 0x10

Channel GUID - indicates the type of channel/service that should be supported by OBMF-ICP Primary.

Bits	Identifier	Access	Reset	Name
127:0	GUID	r	0x123423168094F00180900124567890AB	—

- **GUID field**

Channel GUID - Please see the proper file with this GUID for detailed definition of this structure.

11.1.5.2. CHANNEL_CFG register

- Absolute Address: 0x110
- Base Offset: 0x10
- Size: 0x4

Channel Configuration

Bits	Identifier	Access	Reset	Name
7:0	CHANNEL_NO	r	0x1	—
8	MANDATORY	r	0x1	—
9	ENABLED	rw	0x0	—

- **CHANNEL_NO field**

Channel Number

- **MANDATORY field**

Indicates whether this channel is mandatory for the OBMF-ICP Secondary to be fully functional.

- **ENABLED field**

Indicates whether this channel is enabled: 0 = DISABLED, 1 = ENABLED. OBMF-ICP Primary should write to this field to enable the communication over this channel.

11.1.6. CHANNEL_2 register file

- Absolute Address: 0x200
- Base Offset: 0x200
- Size: 0x14

Channel 2 Discovery Structure

Offset	Identifier	Name
0x00	CHANNEL_GUID	Channel GUID
0x10	CHANNEL_CFG	Channel Configuration

11.1.6.1. CHANNEL_GUID register

- Absolute Address: 0x200
- Base Offset: 0x0
- Size: 0x10

Channel GUID - indicates the type of channel/service that should be supported by OBMF-ICP Primary.

Bits	Identifier	Access	Reset	Name
127:0	GUID	r	0x899ABC228902345189ABBC5609BC450F	—

- **GUID field**

Channel GUID - Please see the proper file with this GUID for detailed definition of this structure.

11.1.6.2. CHANNEL_CFG register

- Absolute Address: 0x210
- Base Offset: 0x10
- Size: 0x4

Channel Configuration

Bits	Identifier	Access	Reset	Name
7:0	CHANNEL_NO	r	0x1	—
8	MANDATORY	r	0x1	—
9	ENABLED	rw	0x0	—

- **CHANNEL_NO field**

Channel Number

- **MANDATORY field**

Indicates whether this channel is mandatory for the OBMF-ICP Secondary to be fully functional.

- **ENABLED field**

Indicates whether this channel is enabled: 0 = DISABLED, 1 = ENABLED. OBMF-ICP Primary should write into this field to enable the communication over this channel.

11.1.7. CHANNEL_3 register file

- Absolute Address: 0x300
- Base Offset: 0x300
- Size: 0x14

Channel 3 Discovery Structure

Offset	Identifier	Name
0x00	CHANNEL_GUID	Channel GUID
0x10	CHANNEL_CFG	Channel Configuration

11.1.7.1. CHANNEL_GUID register

- Absolute Address: 0x300
- Base Offset: 0x0
- Size: 0x10

Channel GUID - indicates the type of channel/service that should be supported by OBMF-ICP Primary.

Bits	Identifier	Access	Reset	Name
127:0	GUID	r	0x34568866AB34567F89AB324212356789	—

- **GUID field**

Channel GUID - Please see the proper file with this GUID for detailed definition of this structure.

11.1.7.2. CHANNEL_CFG register

- Absolute Address: 0x310
- Base Offset: 0x10
- Size: 0x4

Channel Configuration

Bits	Identifier	Access	Reset	Name
7:0	CHANNEL_NO	r	0x1	—
8	MANDATORY	r	0x1	—
9	ENABLED	rw	0x0	—

- **CHANNEL_NO field**

Channel Number

- **MANDATORY field**

Indicates whether this channel is mandatory for the OBMF-ICP Secondary to be fully functional.

- **ENABLED field**

Indicates whether this channel is enabled: 0 = DISABLED, 1 = ENABLED. OBMF-ICP Primary should write into this field to enable the communication over this channel.

11.1.8. CHANNEL_4 register file

- Absolute Address: 0x400
- Base Offset: 0x400
- Size: 0x14

Channel 4 Discovery Structure

Offset	Identifier	Name
0x00	CHANNEL_GUID	Channel GUID
0x10	CHANNEL_CFG	Channel Configuration

11.1.8.1. CHANNEL_GUID register

- Absolute Address: 0x400
- Base Offset: 0x0
- Size: 0x10

Channel GUID - indicates the type of channel/service that should be supported by OBMF-ICP Primary.

Bits	Identifier	Access	Reset	Name
127:0	GUID	r	0xC143A289047340809C421E8C941535B2	—

- **GUID field**

Channel GUID - Please see the proper file with this GUID for detailed definition of this structure.

11.1.8.2. CHANNEL_CFG register

- Absolute Address: 0x410
- Base Offset: 0x10
- Size: 0x4

Channel Configuration

Bits	Identifier	Access	Reset	Name
7:0	CHANNEL_NO	r	0x1	—
8	MANDATORY	r	0x1	—
9	ENABLED	rw	0x0	—

- **CHANNEL_NO field**

Channel Number

- **MANDATORY field**

Indicates whether this channel is mandatory for the OBMF-ICP Secondary to be fully functional.

- **ENABLED field**

Indicates whether this channel is enabled: 0 = DISABLED, 1 = ENABLED. OBMF-ICP Primary should write into this field to enable the communication over this channel.

11.1.9. CHANNEL_5 register file

- Absolute Address: 0x500
- Base Offset: 0x500
- Size: 0x14

Channel 5 Discovery Structure

Offset	Identifier	Name
0x00	CHANNEL_GUID	Channel GUID
0x10	CHANNEL_CFG	Channel Configuration

11.1.9.1. CHANNEL_GUID register

- Absolute Address: 0x500
- Base Offset: 0x0
- Size: 0x10

Channel GUID - indicates the type of channel/service that should be supported by OBMF-ICP Primary.

Bits	Identifier	Access	Reset	Name
127:0	GUID	r	0x2354AB229871543A89ABBC5609BC7567	—

- **GUID field**

Channel GUID - Please see the proper file with this GUID for detailed definition of this structure.

11.1.9.2. CHANNEL_CFG register

- Absolute Address: 0x510
- Base Offset: 0x10
- Size: 0x4

Channel Configuration

Bits	Identifier	Access	Reset	Name
7:0	CHANNEL_NO	r	0x1	—
8	MANDATORY	r	0x1	—

Bits	Identifier	Access	Reset	Name
9	ENABLED	rw	0x0	—

- **CHANNEL_NO field**

Channel Number

- **MANDATORY field**

Indicates whether this channel is mandatory for the OBMF-ICP Secondary to be fully functional.

- **ENABLED field**

Indicates whether this channel is enabled: 0 = DISABLED, 1 = ENABLED. OBMF-ICP Primary should write into this field to enable the communication over this channel.

11.2. OBMF-ICP Flash Channel

FLASH Registers

Offset	Identifier	Name
0x0000	FLASH_SPACE[0]	FLASH SPACE
0x1000	ERASE_START_ADDRESS	ERASE_START_ADDRESS
0x1004	ERASE_SIZE	ERASE_SIZE

11.2.1. FLASH_SPACE register

- Absolute Address: 0x4000
- Base Offset: 0x0
- Size: 0x80
- Array Dimensions: [1]
- Array Stride: 0x80
- Total Size: 0x80

Flash space.

Bits	Identifier	Access	Reset	Name
1023:0	FLASH_SPACE	rw	—	—

- **FLASH_SPACE field**

Flash space

11.2.2. ERASE_START_ADDRESS register

- Absolute Address: 0x5000
- Base Offset: 0x1000

- Size: 0x4

Erase cycle start address (note: devices typically require erase to start on block boundary).

Bits	Identifier	Access	Reset	Name
31:0	ERASE_START_ADDRESS	rw	0x0	—

- **ERASE_START_ADDRESS field**

Erase start address

11.2.3. ERASE_SIZE register

- Absolute Address: 0x5004
- Base Offset: 0x1004
- Size: 0x4

Erase size in bytes (note: devices usually only support specific erase sizes).

Bits	Identifier	Access	Reset	Name
31:0	ERASE_SIZE	rw	0x0	—

- **ERASE_SIZE field**

Erase size in bytes

11.3. OBMF-ICP Virtual Wires Channel

Virtual Wires Registers

Offset	Identifier	Name
0x0	VW_CFG	Configuration of Virtual Wires Channel
0x4	VW_0_STATE	Virtual Wires 0 State
0x5	VW_1_STATE	Virtual Wires 1 State
0x6	VW_2_STATE	Virtual Wires 2 State
0x7	VW_3_STATE	Virtual Wires 2 State
0x8	VW_0_DIRECTION	Virtual Wires 0 Direction
0x9	VW_1_DIRECTION	Virtual Wires 1 Direction
0xA	VW_2_DIRECTION	Virtual Wires 2 Direction
0xB	VW_3_DIRECTION	Virtual Wires 3 Direction

11.3.1.1. VW_CFG register

- Absolute Address: 0x8000
- Base Offset: 0x0
- Size: 0x4

Configuration of Virtual Wires Channel

Bits	Identifier	Access	Reset	Name
7:0	VW_NO	r	0x0	—
8	VW_NOTIFICATION	rw	0x0	—
31:9	reserverd	r	0x0	—

- **VW_NO field**

Total number of VWs supported by OBMF-ICP Producer

- **VW_NOTIFICATION field**

Flag indicating of notifications from OBMF-ICP Producer to OBMF-ICP Secondary are enabled. OBMF-ICP Secondary may update this field to enable/disable. 0 - DISABLED, 1 - ENABLED, default = 0.

- **reserverd field**

Reserved

11.3.1.2. VW_0_STATE register

- Absolute Address: 0x8004
- Base Offset: 0x4
- Size: 0x1

Virtual Wires State

Bits	Identifier	Access	Reset	Name
0	VW_STATE	rw	0x0	—
7:1	reserverd	r	0x0	—

- **VW_STATE field**

Virtual Wires State.Indicates the state of the vGPIO: 0 - LOW , 1 - HIGH

- **reserverd field**

Reserved

11.3.1.3. VW_1_STATE register

- Absolute Address: 0x8005
- Base Offset: 0x5
- Size: 0x1

Virtual Wires State

Bits	Identifier	Access	Reset	Name
0	VW_STATE	rw	0x0	—
7:1	reserverd	r	0x0	—

- **VW_STATE field**

Virtual Wires State.Indicates the state of the vGPIO: 0 - LOW , 1 - HIGH

- **reserverd field**

Reserved

11.3.1.4. VW_2_STATE register

- Absolute Address: 0x8006
- Base Offset: 0x6
- Size: 0x1

Virtual Wires State

Bits	Identifier	Access	Reset	Name
0	VW_STATE	rw	0x0	—
7:1	reserverd	r	0x0	—

- **VW_STATE field**

Virtual Wires State.Indicates the state of the vGPIO: 0 - LOW , 1 - HIGH

- **reserverd field**

Reserved

11.3.1.5. VW_3_STATE register

- Absolute Address: 0x8007
- Base Offset: 0x7
- Size: 0x1

Virtual Wires State

Bits	Identifier	Access	Reset	Name
0	VW_STATE	rw	0x0	—
7:1	reserverd	r	0x0	—

- **VW_STATE field**

Virtual Wires State.Indicates the state of the vGPIO: 0 - LOW , 1 - HIGH

- **reserverd field**

Reserved

11.3.1.6. VW_0_DIRECTION register

- Absolute Address: 0x8008
- Base Offset: 0x8
- Size: 0x1

Virtual Wires Direction

Bits	Identifier	Access	Reset	Name
1:0	VW_DIRECTION	rw	0x0	—

- **VW_DIRECTION field**

Virtual Wires Direction. Indicates if the VW is an input or output: 0 - Input, 1 - Output, 2 - Hi-Z, 3 - Input & Output

11.3.1.7. VW_1_DIRECTION register

- Absolute Address: 0x8009
- Base Offset: 0x9
- Size: 0x1

Virtual Wires Direction

Bits	Identifier	Access	Reset	Name
1:0	VW_DIRECTION	rw	0x1	—

- **VW_DIRECTION field**

Virtual Wires Direction. Indicates if the VW is an input or output: 0 - Input, 1 - Output, 2 - Hi-Z, 3 - Input & Output

11.3.1.8. VW_2_DIRECTION register

- Absolute Address: 0x800A
- Base Offset: 0xA
- Size: 0x1

Virtual Wires Direction

Bits	Identifier	Access	Reset	Name
1:0	VW_DIRECTION	rw	0x2	—

- **VW_DIRECTION field**

Virtual Wires Direction. Indicates if the VW is an input or output: 0 - Input, 1 - Output, 2 - Hi-Z, 3 - Input & Output

11.3.1.9. VW_3_DIRECTION register

- Absolute Address: 0x800B
- Base Offset: 0xB
- Size: 0x1

Virtual Wires Direction

Bits	Identifier	Access	Reset	Name
1:0	VW_DIRECTION	rw	0x3	—

- **VW_DIRECTION field**

Virtual Wires Direction. Indicates if the VW is an input or output: 0 - Input, 1 - Output, 2 - Hi-Z, 3 - Input & Output

11.4. OBMF-ICP RTC Channel

RTC Registers

Offset	Identifier	Name
0x0	SEC	SEC
0x1	SEC_ALARM	SEC_ALARM
0x2	MINUTES	MINUTES
0x3	MINUTESALARM	MINUTESALARM
0x4	HOURS	HOURS
0x5	HOURSALARM	HOURSALARM
0x6	DAYOFWEEK	DAYOFWEEK
0x7	DAYOFMONTH	DAYOFMONTH
0x8	MONTH	MONTH
0x9	YEAR	YEAR
0xA	REGISTERA	REGISTERA
0xB	REGISTERB	REGISTERB
0xC	REGISTERC	REGISTERC
0xD	REGISTERD	REGISTERD

Offset	Identifier	Name
0xE	REGISTERE	REGISTERE

11.4.1. SEC register

- Absolute Address: 0xC000
- Base Offset: 0x0
- Size: 0x1

RTC Index: 00h. Attribute: Read/Write. Size: 8-bit. This register stores current time second. None of the bits are affected by any reset signal. Only cleared at RTC Power loss

Bits	Identifier	Access	Reset	Name
7:0	SEC	rw	0x0	—

- **SEC field**

Time Seconds. The time in seconds can be represented in either BCD or Binary format depending on the value in RegB.Data Mode.

11.4.2. SEC_ALARM register

- Absolute Address: 0xC001
- Base Offset: 0x1
- Size: 0x1

RTC Index: 01h. Attribute: Read/Write. Size: 8-bit. This register stores Seconds Alarm. None of the bits are affected by any reset signal. Only cleared at RTC Power loss

Bits	Identifier	Access	Reset	Name
7:0	SEC_ALARM	rw	0x0	—

- **SEC_ALARM field**

Seconds field of the Alarm.

11.4.3. MINUTES register

- Absolute Address: 0xC002
- Base Offset: 0x2
- Size: 0x1

RTC Index: 02h. Attribute: Read/Write. Size: 8-bit. This register stores current time Minutes. None of the bits are affected by any reset signal. Only cleared at RTC Power loss.

Bits	Identifier	Access	Reset	Name
7:0	MINUTES	rw	0x0	—

- **MINUTES field**

Time Minutes. The time in minutes can be represented in either BCD or Binary format depending on the value in RegB.Data Mode

11.4.4. MINUTESALARM register

- Absolute Address: 0xC003
- Base Offset: 0x3
- Size: 0x1

RTC Index: 03h. Attribute: Read/Write. Size: 8-bit. This register stores Alarm Minutes. None of the bits are affected by any reset signal. Only cleared at RTC Power loss.

Bits	Identifier	Access	Reset	Name
7:0	MINUTESALARM	rw	0x0	—

- **MINUTESALARM field**

Minutes field of the Alarm

11.4.5. HOURS register

- Absolute Address: 0xC004
- Base Offset: 0x4
- Size: 0x1

RTC Index: 04h. Attribute: Read/Write. Size: 8-bit. This register stores current time Hours. None of the bits are affected by any reset signal. Only cleared at RTC Power loss.

Bits	Identifier	Access	Reset	Name
7:0	HOURS	rw	0x12	—

- **HOURS field**

Time Hours. The time in hours can be represented in either BCD or Binary format depending on the value in RegB.Data Mode. It can also be represented in either 12-hour mode or 24-hour mode depending on the value in RegB.Hour Format.

11.4.6. HOURSALARM register

- Absolute Address: 0xC005
- Base Offset: 0x5
- Size: 0x1

RTC Index: 05h. Attribute: Read/Write. Size: 8-bit. This register stores Alarm Hours. None of the bits are affected by any reset signal. Only cleared at RTC Power loss.

Bits	Identifier	Access	Reset	Name
7:0	HOURSALARM	rw	0x12	—

- **HOURSALARM field**

Hours field of the Alarm.

11.4.7. DAYOFWEEK register

- Absolute Address: 0xC006
- Base Offset: 0x6
- Size: 0x1

RTC Index: 06h. Attribute: Read/Write. Size: 8-bit. This register stores current Day of Week. None of the bits are affected by any reset signal. Only cleared at RTC Power loss.

Bits	Identifier	Access	Reset	Name
7:0	DAYOFWEEK	rw	0x7	—

- **DAYOFWEEK field**

This field indicates current Day of Week. 1-Sunday 2-Monday 3-Tuesday 4-Wednesday 5-Thursday 6-Friday 7-Saturday. The value is the same regardless of the Data Mode.

11.4.8. DAYOFMONTH register

- Absolute Address: 0xC007
- Base Offset: 0x7
- Size: 0x1

RTC Index: 07h. Attribute: Read/Write. Size: 8-bit. This register stores current Day of Month. None of the bits are affected by any reset signal. Only cleared at RTC Power loss.

Bits	Identifier	Access	Reset	Name
7:0	DAYOFMONTH	rw	0x1	—

- **DAYOFMONTH field**

This field indicates current Day of Month. The day of month can be represented in either BCD or Binary format depending on the value in RegB.Data Mode.

11.4.9. MONTH register

- Absolute Address: 0xC008
- Base Offset: 0x8
- Size: 0x1

RTC Index: 08h. Attribute: Read/Write. Size: 8-bit. This register stores current Month. None of the bits are affected by any reset signal. Only cleared at RTC Power loss.

Bits	Identifier	Access	Reset	Name
7:0	MONTH	rw	0x1	—

- **MONTH field**

This field indicates current Month. The month can be represented in either BCD or Binary format depending on the value in RegB.Data Mode.

11.4.10. YEAR register

- Absolute Address: 0xC009
- Base Offset: 0x9
- Size: 0x1

RTC Index: 09h. Attribute: Read/Write. Size: 8-bit. This register stores current Year. None of the bits are affected by any reset signal. Only cleared at RTC Power loss.

Bits	Identifier	Access	Reset	Name
7:0	YEAR	rw	0x0	—

- **YEAR field**

This field indicates current Year. The year can be represented in either BCD or Binary format depending on the value in RegB.Data Mode.

11.4.11. REGISTERA register

- Absolute Address: 0xC00A
- Base Offset: 0xA
- Size: 0x1

RTC Index: 0Ah. This register is used for general configuration of the RTC functions.

Bits	Identifier	Access	Reset	Name
6:4	DV	rw	0x2	—
7	UIP	r	0x0	—

- **DV field**

Controls the divider chain for the oscillator, and are not affected by any other reset signal.02h: Normal Operation.06h/07h: Divider in Reset. RTC stops.Others: Ignored. Keep last State.Note: Write a 06/07h, will get read back as 06h.

- **UIP field**

When set, an update is is in progress. When cleared, the update cycle will not start for at least 488 us. Always return '0' in Integrated Boot.

11.4.12. REGISTERB register

- Absolute Address: 0xC00B
- Base Offset: 0xB
- Size: 0x1

RTC Index: 0Bh. Attribute: Read/Write This register is used for general configuration of the RTC functions.

Bits	Identifier	Access	Reset	Name
0	DSE	r	0x0	—
1	HF	rw	0x0	—
2	DM	rw	0x0	—
3	SQWE	r	0x0	—
4	UIE	r	0x0	—
5	AIE	rw	0x0	—
6	PIE	r	0x0	—
7	SET	rw	0x0	—

- **DSE field**

Not implemented.

- **HF field**

When set, twenty-four hour mode is selected. When cleared, twelve-hour mode is selected. In twelve hour mode, the seventh bit represents AM (cleared) and PM (set).

- **DM field**

When set, represents binary representation. When cleared, denotes BCD.

- **SQWE field**

Not implemented.

- **UIE field**

When set and C.UF is set, an interrupt is generated. Not Implemented.

- **AIE field**

When set, and C.AF is set, an interrupt is generated.

- **PIE field**

When set, and C.PF is set, an interrupt is generated. Not Implemented.

- **SET field**

When cleared, an update cycle occurs once each second. If set, a current update cycle will abort and subsequent update cycles will not occur until SET is returned to zero. When set, SW may initialize time and calendar bytes safely.Note: Ignored by Integrated boot. Always allow to update Time. Reset on Global reset

11.4.13. REGISTERC register

- Absolute Address: 0xC00C
- Base Offset: 0xC
- Size: 0x1

RTC Index: 0Ch. Attribute: Read-Only (Writes have no effect). All bits in this register are cleared when this register is read.

Bits	Identifier	Access	Reset	Name
4	UF	r	0x0	—
5	AF	rw	0x0	—
6	PF	r, rclr	0x0	—
7	IRQF	r	0x0	—

- **UF field**

Updated-ended flag will be high immediately following an update cycle for each second. The bit is cleared upon AUX_PWRGOOD deassertion or a read of Register C.Note: in Integrated boot, Always read as '0'.

- **AF field**

Alarm Flag will be high after all Alarm values match the current time. This bit is cleared upon RTCRST# assertion or a read of Register C.

- **PF field**

Periodic interrupt Flag will be one when the tap as specified by the RS bits of register A is one. If no taps are specified, this flag bit will remain at zero. This bit is cleared upon AUX_PWRGOOD deassertion or a read of Register C. Note: In Integrated boot, always read as '0'.

- **IRQF field**

Interrupt Request Flag = (PF * PIE) + (AF * AIE) + (UF * UIE). This also causes the RTC Interrupt to be asserted.Note: In integrated boot, as PIE, UIE are not supported, It = (AF * AIE).

11.4.14. REGISTERD register

- Absolute Address: 0xC00D
- Base Offset: 0xD
- Size: 0x1

RTC Index: 0Dh.

Bits	Identifier	Access	Reset	Name
5:0	DA	r	0x0	—
7	VRT	r	0x1	—

- **DA field**

These bits store the date of month alarm value. If set to 000000, then a don't care state is assumed. Data alarm not supported in Bootable CPU, these bits will return zeros to mimic the functionality of the Motorola 146818B.

- **VRT field**

This bit should always be written as a 0 for write cycle, however it will return a 1 for read cycles.

11.4.15. REGISTERE register

- Absolute Address: 0xC00E
- Base Offset: 0xE
- Size: 0x1

Remaining 114 Bytes of Lower User RAM. Each byte in this bank share the same description as shown below. RAM default values are undetermined and the last written value will be retained until RTC Clear

Bits	Identifier	Access	Reset	Name
5:0	LOWER_USER_RAM	r	0x0	—

- **LOWER_USER_RAM field**

These bits store the date of month alarm value. If set to 000000, then a don't care state is assumed. Data alarm not supported in Bootable CPU, these bits will return zeros to mimic the functionality of the Motorola 146818B.

11.5. OBMF-ICP UART Channel

UART Registers

Offset	Identifier	Name
0x0	RBR_THR_DLL	RBR_THR_DLL
0x1	IER_DLH	IER_DLH
0x2	IIR_FCR	IIR_FCR
0x3	LCR	LCR
0x4	MCR	MCR
0x5	LSR	MCR
0x6	MSR	MSR
0x7	SCR	SPR

11.5.1. RBR_THR_DLL register

- Absolute Address: 0x10000
- Base Offset: 0x0
- Size: 0x1

RBR = Recieve buffer register - RO. This register contains the data byte received on the serial input port (sin). The data in this register is valid only if the Data Ready (LSR(0) is set to 1). If FIFOs are disabled (FCR(0) is cleared to 0) the data in the RBR must be read before the next data arrives, otherwise it will be overwritten, resulting in an overrun error. If FIFOs are enabled (FCR(0) set to 1) this register accesses the head of the receive FIFO. If the receive FIFO is full, and this register is not read before the next data character arrives, then the data already in the FIFO will be preserved but any incoming data will be lost. An overrun error will also occur. THR = Transmit hold register - WO. This register contains data to be transmitted on the serial output port (sout). Data should only be written to the THR when the THR Empty bit (LSR(5) is set to1). If FIFOs are disabled (FCR(0) is set to 0) and THRE is set to 1, writing a single character to the THR clears the THRE. Any additional writes to the THR before the THRE is set again causes the THR data to be overwritten. If FIFO's are enabled (FCR(0) is set to 1) and THRE is set, the FIFO can be filled up to a preconfigured depth (FIFO_DEPTH). Any attempt to write data when the FIFO is full results in the write data being lost. DLL = Divisor Latch low. - RW. This register makes up the lower 8-bits of a 16-bit, Read/write, Divisor Latch register that contains the baud rate divisor for the UART. This register may only be accessed when the DLAB bit (LCR(7) is set to 1). The output baud rate is equal to the system clock (clk) frequency divided by sixteen times the value of the baud rate divisor, as follows: baud rate = (system clock freq) / (16 * divisor) Note: With the Divisor Latch Registers (DLL and DLH) set to zero, the baud clock is disabled and no serial communications will occur. Also, once the DLL is set, at least 8 system clock cycles should be allowed to pass before transmitting or receiving data.

Bits	Identifier	Access	Reset	Name
7:0	RBR_THR_DLL	rw	0x0	—

- **RBR_THR_DLL field**

When DLAB = 0, this 8-bit value is used as Received Buffer Register as Read, and is used as Transmit Holding Register as Write. When DLAB is 1, this register is used as DLL (Divisor Latch Low Byte Register).

11.5.2. IER_DLH register

- Absolute Address: 0x10001
- Base Offset: 0x1
- Size: 0x1

IER = Interrupt Enable Register - RW. The ier_dlh (Interrupt Enable Register) may only be accessed when the DLAB bit (7) of the LCR Register is set to 0. Allows control of the Interrupt Enables for transmit and receive functions. refer to UART Interrupt Enable register for details. DLH = Divisor Latch High - RW. The Divisor Latch High Register is accessed when the DLAB

bit (LCR(7) is set to 1). Bits[7:0] contain the high order 8-bits of the baud rate divisor. The output baud rate is equal to the system clock (clk) frequency divided by sixteen times the value of the baud rate divisor, as follows: baud rate = (system clock freq) / (16 * divisor). Note: With the Divisor Latch Registers (DLL and DLH) set to zero, the baud clock is disabled and no serial communications will occur. Also, once the DLL is set, at least 8 system clock cycles should be allowed to pass before transmitting or receiving data.

Bits	Identifier	Access	Reset	Name
7:0	IER_DLH	rw	0x0	—

- **IER_DLH field**

This is a multi-function register. This register enables/disables receive and transmit interrupts and also controls the most significant 8-bits of the baud rate divisor. When DLAB = 0, this 8-bit value is used as Interrupt Enable Register. When DLAB = 1, this 8-bit value is used as DLH (Divisor Latch High Byte Register).

11.5.3. IIR_FCR register

- Absolute Address: 0x10002
- Base Offset: 0x2
- Size: 0x1

IIR = Interrupt Identification Register - RO. The IIR register is read to determine the type and source of UART interrupts. To be 16550 compatible, the lower 4 bits (0-3) of the IIR register are priority encoded. If two or more interrupts occurs which are represented by bits (0-3), only the interrupt with the highest priority is displayed. The upper 4 bits (4-7) are not priority encoded. These bits will assert/deassert independently of the lower 4 bits. Bit 0 (nIP) is used to indicate the existence of an interrupt in the priority encoded bits (0-3) of the IIR register. A low signal on this bit indicates an encoded interrupt is pending. If this bit is high, no encoded interrupt is pending regardless of the state of the other 3 bits. nIP has no effect ,or association with, the upper bits four bits (4-7) which assert/deassert independently of nIP. Refer to UART Interrupt Identification register for details. FCR = FIFO Control Register - WO. FCR is a write-only register that is located at the same address as the IIR (IIR is a read-only register). FCR enables/disables the transmitter/receiver FIFOs, clears the transmitter/receiver FIFOs, sets the Receiver FIFO trigger level. Refer to FIFO Control register for details.

Bits	Identifier	Access	Reset	Name
7:0	IIR_FCR	rw	0x1	—

- **IIR_FCR field**

When read, this 8-bit value is used as Interrupt Identification Register. When written, this 8-bit value is used as FIFO Control Register.

11.5.4. LCR register

- Absolute Address: 0x10003
- Base Offset: 0x3
- Size: 0x1

In the Line Control register, system programmers specify the format of the asynchronous data communications exchange. The serial data format consists of a start bit (logic 0), five to eight data bits, an optional parity bit, and one or two stop bits (logic 1). The LCR has bits for accessing the Divisor Latch registers and causing a Break condition. Programmers can also read the contents of the Line Control register. The read capability simplifies system programming and eliminates the need for separate storage in system memory.

Bits	Identifier	Access	Reset	Name
1:0	LCR_WLS	rw	0x0	—
2	LCR_STB	rw	0x0	—
3	LCR_PEN	rw	0x0	—
4	LCR_EPS	rw	0x0	—
5	LCR_STKYP	rw	0x0	—
6	LCR_SB	rw	0x0	—
7	LCR_DLAB	rw	0x0	—

- **LCR_WLS field**

The word length select bits specify the number of data bits (five to eight bits are allowed) in each transmitted or received serial character. 00: 5-bit character (default) 01: 6-bit character, 10: 7-bit character, 11: 8-bit character

- **LCR_STB field**

This bit specifies the number of stop bits transmitted and received in each serial character. 0: 1 stop bit, 1: 2 stop bits; except for 5-bit character, then 1 bits, The receiver checks the first stop bit only, regardless of the number of stop bits selected.

- **LCR_PEN field**

When PEN is logic 1, a parity bit is generated (transmit data) or checked (receive data) between the last data-word bit and stop bit of the serial data. 0: No parity function. 1: Allows parity generation and checking.

- **LCR_EPS field**

EPS is only valid when the parity is enabled (PEN = 1). 0: Sends and checks for odd parity, 1: Sends and checks for even parity.

- **LCR_STKYP field**

This bit is the sticky-parity bit, which can be used in multiprocessor communications. 0: No effect on parity bit. 1: Forces parity bit to be opposite of EPS bit value. When PEN and STKYP are logic 1, the bit that is transmitted in the parity-bit location (the bit just before the stop bit) is

the complement of the EPS bit. If EPS is 0, then the bit at the parity-bit location will be transmitted as a 1. In the receiver, if STKYP and PEN are 1, then the receiver compares the bit that is received in the parity-bit location with the complement of the EPS bit. If the values being compared are not equal, the receiver sets the parity-error bit in LSR, and causes an Error Interrupt if Line-Status Interrupts were enabled. For example, if EPS is 0, the receiver expects the bit received at the parity-bit location to be 1. If it is not, then the parity-error bit is set. By forcing the bit value at the parity-bit location, rather than calculating a parity value, a system with a Driver/Initiator transmitter and multiple receivers can identify some transmitted characters as receiver addresses and the rest of the characters as data. If PEN = 0, STKYP is ignored.

- **LCR_SB field**

The set-break control bit transmits a Break condition to the receiving UART. When SB is set to logic 1, the serial output (TXD) is forced to the Spacing (logic 0) state and remains there until SB is set to logic 0. This bit acts only on the TXD pin and has no effect on the transmitter logic. 0: No effect on TXD output. 1: Forces TXD output to 0 (space). This feature enables the processor to alert a terminal in a computer communications system. If the following sequence is executed, no erroneous characters will be transmitted because of the break: Load 0x00 in the Transmit Holding register in response to a TDRQ Interrupt. After TDRQ goes high (indicating that 0x00 is being shifted out), set the break bit before the parity or stop bits reach the TXD pin. Wait for the transmitter to be idle (TEMT = 1), and clear the break bit when normal transmission has to be restored. During the Break, the transmitter can be used as a character timer to accurately establish the Break duration. In FIFO mode, wait for the transmitter to be idle (TEMT=1) to set and clear the break bit.

- **LCR_DLAB field**

The divisor-latch access bit must be set high (logic 1) to access the divisor latches of the baud rate generator during a read or write operation. It must be set low (logic 0) to access the receiver buffer, the Transmit Holding register, or the Interrupt Enable register. This bit does not need to be set when using auto-baud. 0: Access Transmit Holding register (THR), Receive Buffer register (RBR) and Interrupt Enable register. 1: Access Divisor Latch registers (DLL and DLH)

11.5.5. MCR register

- Absolute Address: 0x10004
- Base Offset: 0x4
- Size: 0x1

This register controls the interface with the modem or data set (or a peripheral device emulating a modem)

Bits	Identifier	Access	Reset	Name
0	MCR_DTR	rw	0x0	—
1	MCR_RTS	rw	0x0	—
2	MCR_OUT1	rw	0x0	—

Bits	Identifier	Access	Reset	Name
3	MCR_OUT2	rw	0x0	—
4	MCR_LOOP	rw	0x0	—

- **MCR_DTR field**

This bit controls the Data Terminal Ready output. 0: Primary output \"uart_dtr\" pin is forced to logic 1. 1: Primary output uart_dtr pin is forced to logic 0. The uart_dtr output of the UART may be applied to an EIA inverting line driver (such as the DS1488) to obtain the proper polarity input at the succeeding modem or data set.

- **MCR_RTS field**

When Auto Flow is disabled: MCR.AFE = 0: 0: Primary output uart_rts pin is forced to logic 1 | 1: Primary output uart_rts pin is forced to logic 0 When Auto Flow is enabled: MCR.AFE = 1: | 0: Auto-RTS is disabled | 1: Auto-RTS is enabled

- **MCR_OUT1 field**

The auxiliary output OUT1 bit is only used in Loopback Test mode. 0: uart_ri UART pin input set low when in Loop-back mode 1: uart_ri UART pin input set high when in Loop-back mode Note: OUT1 is not used in IBL.

- **MCR_OUT2 field**

The function of the auxiliary output OUT2 bit differs depending on the mode of the UART. 0: uart_dcd UART pin input set low when in Loop-back mode 1: uart_dcd UART pin input set high when in Loop-back mode. Note: OUT2 is not used in IBL.

- **MCR_LOOP field**

This bit provides a local loopback feature for diagnostic testing of the UART. This feature allows the processor to verify the UART transmit and receive data paths. The Transmit, Receive, and Modem Control interrupts are operational in this mode. 0: Normal UART operation. 1: Loopback Test mode operation. When LOOP is set to logic 1, the following will occur: The TXD (transmitter output) pin is set to a logic-1 state. The RXD (receiver input) pin is disconnected. The output of the Transmitter Shift register is \"looped back\" into the Receiver-Shift register input. The four modem-control inputs (uart_cts, uart_dsr, uart_dcd, and uart_ri) are disconnected from the pins and the modem-control output pins (uart_rts and uart_dtr) are forced to their inactive state. The lower four bits of the Modem Control register (MCR) are connected to the upper four modem status register (MSR) bits. Flow control can be tested. DTR = 1 forces DSR to a 1 RTS = 1 forces CTS to a 1 OUT1 = 1 forces RI to a 1 OUT2= 1 forces DCD to a 1 Coming out of the Loopback Test mode may result in unpredictable activation of the delta bits (bits 3:0) in the Modem Status register (MSR). It is recommended that MSR is read once to clear the delta bits in the MSR.

11.5.6. LSR register

- Absolute Address: 0x10005
- Base Offset: 0x5
- Size: 0x1

This register provides status information to the processor concerning the data transfers. Bits 5 and 6 indicate the status of the transmitter. The remainder of the bits contains information about the receiver. In non-FIFO mode, three of the LSR register bits parity error, framing error, and break interrupt indicate the error status of the character that has just been received. In FIFO mode, these three status bits are stored with each received character in the FIFO. These bits in the line status register are associated with the character at the top of the FIFO. These bits are not cleared by reading the erroneous byte from the FIFO or receive buffer. They are cleared only by reading LSR. In FIFO mode, the Line Status Interrupt occurs only when the erroneous byte is at the top of the FIFO. If the erroneous byte being received is not at the top of the FIFO, an interrupt is generated only after the previous bytes are read and the erroneous byte is moved to the top of the FIFO.

Bits	Identifier	Access	Reset	Name
0	LSR_DR	r	0x0	—
1	LSR_OE	r	0x0	—
2	LSR_PE	r	0x0	—
3	LSR_FE	r	0x0	—
4	LSR_BI	r	0x0	—
5	LSR_TDRQ	r	0x1	—
6	LSR_TEMT	r	0x1	—
7	LSR_FIFOE	r	0x0	—

- **LSR_DR field**

DR is set to logic 1 when a complete incoming character has been received and transferred into the Receiver Buffer register or the FIFO. In non-FIFO mode, DR is reset to 0 when the receive buffer is read. In FIFO mode, DR is reset to logic 0 if the FIFO is empty (last character has been read from RBR) or the RESETRF bit is set in FCR.

- **LSR_OE field**

0: No overflow error. Data has not been lost. 1: Overflow error. Receive data has been lost. In non-FIFO mode, OE indicates that data in the Receiver Buffer register was not read by the processor before the next character was received; the new character is lost. In FIFO mode, OE indicates that all 64 bytes of the FIFO are full and the most recently received byte has been discarded. The OE indicator is set to logic 1 upon detection of an overflow condition and reset when the processor reads the Line Status register.

- **LSR_PE field**

0: No Parity error. 1: Parity error has occurred. PE indicates that the received character does not have the correct even or odd parity, as selected by the Even Parity Select bit. The PE is set to logic 1 upon detection of a parity error and is reset to logic 0 when the processor reads the Line Status register. In FIFO mode, PE shows a parity error for the character at the bottom of the FIFO, not the most recent character received.

- **LSR_FE field**

0: No Framing error. 1: Framing error has occurred. FE indicates that the received character did not have a valid stop bit. FE is set to logic 1 when the bit following the last data bit or parity bit is detected as logic 0 bit (spacing level). If the Line Control register had been set for two stop bits, the receiver does not check for a valid second stop bit. The FE indicator is reset when the processor reads the Line Status register. The UART will resynchronize after a framing error by assuming that the framing error was due to the next start bit. Therefore it samples this start bit twice and then takes in the data. In FIFO mode, FE shows a framing error for the character at the bottom of the FIFO, not for the most recent character received.

- **LSR_BI field**

0: No break signal has been received. 1: Break signal has been received. BI is set to logic 1 when the received data input is held in the Spacing (logic 0) state for longer than a full character transmission time (that is, the total time of start bit + data bits + parity bit + stop bits). The Break Indicator is reset when the processor reads the Line Status register. In FIFO mode, only one Break character (equal to 0x00) is loaded into the FIFO regardless of the length of the Break condition. BI shows the Break condition for the character at the bottom of the FIFO, not the most recent character received.

- **LSR_TDRQ field**

If THRE mode is disabled (IER(7) set to zero). 0: The UART is NOT ready to receive data for transmission. 1: The UART is ready to receive data for transmission. The assertion of TDRQ causes the UART to interrupt the processor when the Transmit Data Request Interrupt Enable IER.TIE is set high. TDRQ is set to 1 when the new symbol is copied from Transmit Holding register into the Transmit Shift register. The bit is reset to logic 0 concurrently with the loading of the Transmit Holding register by the processor. In FIFO mode, TDRQ is set to logic 1 when the FIFO is empty. It is cleared when the FIFO is more than half full. If more than 32 characters are loaded into the FIFO, the excess characters are lost. If THRE mode is enabled (IER(7) set to one), the functionality is switched to indicate the transmitter FIFO is full, and no longer controls THRE interrupts, which are then controlled by the FCR[5:4] threshold setting.

- **LSR_TEMT field**

0: There is data in the Transmit Shift register, the Transmit Holding register, or the FIFO (in FIFO mode). 1: All the data in the transmitter has been shifted out.

- **LSR_FIFOE field**

0: No character has error inside the Receive FIFO. 1: At least one character in Receiver FIFO has errors. In non-FIFO mode, this bit is always set to 0. In FIFO mode, FIFOE is set to 1 when there is at least one parity error, framing error, or break indication for any of the characters in the FIFO. A processor read to the Line Status register does not reset this bit. FIFOE is reset when all error bytes have been read from the FIFO.

11.5.7. MSR register

- Absolute Address: 0x10006
- Base Offset: 0x6
- Size: 0x1

The MSR provides the current state of the control lines from the modem or data set (or a peripheral device emulating a modem) to the processor. In addition to this current state information, four bits of the MSR provide change information. These bits, 3:0, are set to logic 1 when the associated control Input from the remote modem changes state. They are reset to logic 0 when the processor reads the MSR. When bits 0, 1, 2, or 3 are set to logic 1, a Modem Status Interrupt IIR.IID is generated if bit MIE of the Interrupt Enable register is set.

Bits	Identifier	Access	Reset	Name
0	MSR_DCTS	r	0x0	—
1	MSR_DDSR	r	0x0	—
2	MSR_TERI	r	0x0	—
3	MSR_DDCR	r	0x0	—
4	MSR_CTS	r	0x0	—
5	MSR_DSR	r	0x0	—
6	MSR_RI	r	0x0	—
7	MSR_DCD	r	0x0	—

- **MSR_DCTS field**

0: No change in uart_cts pin since last read of MSR 1: uart_cts pin has change state When DCTS is set the modem status interrupt will be generated if enabled in IER.

- **MSR_DDSR field**

0: No change in uart_dsr pin since last read of MSR 1: uart_dsr pin has changed state When DDSR is set the modem status interrupt will be generated if enabled in IER.

- **MSR_TERI field**

0: No change in uart_ri pin since last read of MSR 1: uart_ri pin has changed state When TERI is set the modem status interrupt will be generated if enabled in IER.

- **MSR_DDCR field**

0: No change in uart_dcr pin since last read of MSR 1: uart_dcr pin has changed state When DDCR is set the modem status interrupt will be generated if enabled in IER.

- **MSR_CTS field**

Loopback is disabled, MCR.LOOP = 0 | 0: Primary input uart_cts is logic 1 | 1: Primary input uart_cts is logic 0 Loopback is enabled, MCR.LOOP = 1 | 0: MCR.RTS is logic 0 | 1: MCR.RTS is logic 1

- **MSR_DSR field**

Loopback is disabled, MCR.LOOP = 0 | 0: Primary input uart_dsr is logic 1 | 1: Primary input uart_dsr is logic 0 Loopback is enabled, MCR.LOOP = 1 | 0: MCR.DTR is logic 0 | 1: MCR.DTR is logic 1

- **MSR_RI field**

Loopback is disabled, MCR.LOOP = 0 | 0: Primary input uart_ri is logic 1 | 1: Primary input uart_ri is logic 0 Loopback is enabled, MCR.LOOP = 1 | 0: MCR.OUT1 is logic 0 | 1: MCR.OUT1 is logic 1

- **MSR_DCD field**

Loopback is disabled, MCR.LOOP = 0 | 0: Primary input uart_dcd is logic 1 | 1: Primary input uart_dcd is logic 0 Loopback is enabled, MCR.LOOP = 1 | 0: MCR.OUT2 is logic 0 | 1: MCR.OUT2 is logic 1

11.5.8. SCR register

- Absolute Address: 0x10007
- Base Offset: 0x7
- Size: 0x1

This is the SPR register

Bits	Identifier	Access	Reset	Name
7:0	SPR	rw	0x0	—

- **SPR field**

Writing to this field does not affect the operation of the UART in any way.

11.6. OBMF-ICP MMIO Channel

Memory Mapped I/O Space

Offset	Identifier	Name
0x0	MMIO_SPACE[0]	MMIO_SPACE

11.6.1. MMIO_SPACE register

- Absolute Address: 0x14000
- Base Offset: 0x0
- Size: 0x80
- Array Dimensions: [1]
- Array Stride: 0x80
- Total Size: 0x80

Memory Mapped I/O Space

Bits	Identifier	Access	Reset	Name
1023:0	MMIO_SPACE	rw	—	—

MMIO_SPACE field
MMIO space

11.7. OBMF-ICP TPM Channel

Trusted Platform Module Space. TPM Spec: PC Client Work Group PC Client Specific TPM Interface Specification (TIS), Version 1.21, Revision 1.00.: https://trustedcomputinggroup.org/wp-content/uploads/TCG_PCClientTPMSpecification_1-21_1-00_FINAL.pdf

Offset	Identifier	Name
0x00	TPM_SPACE[0]	TPM SPACE

11.7.1. TPM_SPACE register

- Absolute Address: 0x18000
- Base Offset: 0x0
- Size: 0x40
- Array Dimensions: [1]
- Array Stride: 0x40
- Total Size: 0x40

TPM space.

Bits	Identifier	Access	Reset	Name
511:0	TPM_SPACE	rw	—	—

1 TPM_SPACE field
TPM space

11.8. OBMF-ICP Legacy I/O Channel

Post Code I/O Channel for system diagnostics and debugging.

Offset	Identifier	Name
0x80	POST_CODE_SPACE[0]	POST CODE SPACE
0x81	POST_CODE_SPACE[1]	POST CODE SPACE

11.8.1. POST_CODE_SPACE register

- Absolute Address: 0x1C000
- Base Offset: 0x80
- Size: 0x1
- Array Dimensions: [2]
- Array Stride: 0x1
- Total Size: 0x2

Post Code space.

Bits	Identifier	Access	Reset	Name
7:0	POST_CODE_SPACE	rw	—	—

- **POST_CODE_SPACE field**

Post Code space

11.8.2. POST_CODE_SPACE register

- Absolute Address: 0x1C001
- Base Offset: 0x81
- Size: 0x1
- Array Dimensions: [2]
- Array Stride: 0x1
- Total Size: 0x2

Post Code space.

Bits	Identifier	Access	Reset	Name
7:0	POST_CODE_SPACE	rw	—	—

- **POST_CODE_SPACE field**

Post Code space