Contents

[1. Scope 2](#_Toc212214361)

[2. System overview 3](#_Toc212214362)

[3. MOFC interface definition 4](#_Toc212214363)

[3.1 Digital ICD 4](#_Toc212214364)

[3.2 Communication format 5](#_Toc212214365)

[3.2.1 SPI Interface Specifications 5](#_Toc212214366)

[3.2.2 SPI write transaction 5](#_Toc212214367)

[3.2.3 SPI Read transaction 6](#_Toc212214368)

[3.2.4 Timing constrains 6](#_Toc212214369)

[4. MOFC SPI COMMAND SET 7](#_Toc212214370)

[5. SPI interface module 8](#_Toc212214371)

[6. Software API, for interfacing with MOFC 9](#_Toc212214372)

[6.1 Global API functions 9](#_Toc212214373)

[6.2 Specific API functions 9](#_Toc212214374)

[7. Software test application for MOFC integration 10](#_Toc212214375)

[7.1 GUI test applications for MOFC assignment 10](#_Toc212214376)

[7.1.1 MOFC assignment 10](#_Toc212214377)

[7.1.2 Reading Status Reg and Versions 11](#_Toc212214378)

[7.1.3 Global read/write register 11](#_Toc212214379)

[7.2 GUI test applications for MOFC read configuration 12](#_Toc212214380)

Figures

**No table of figures entries found.**

Tables

**No table of figures entries found.**

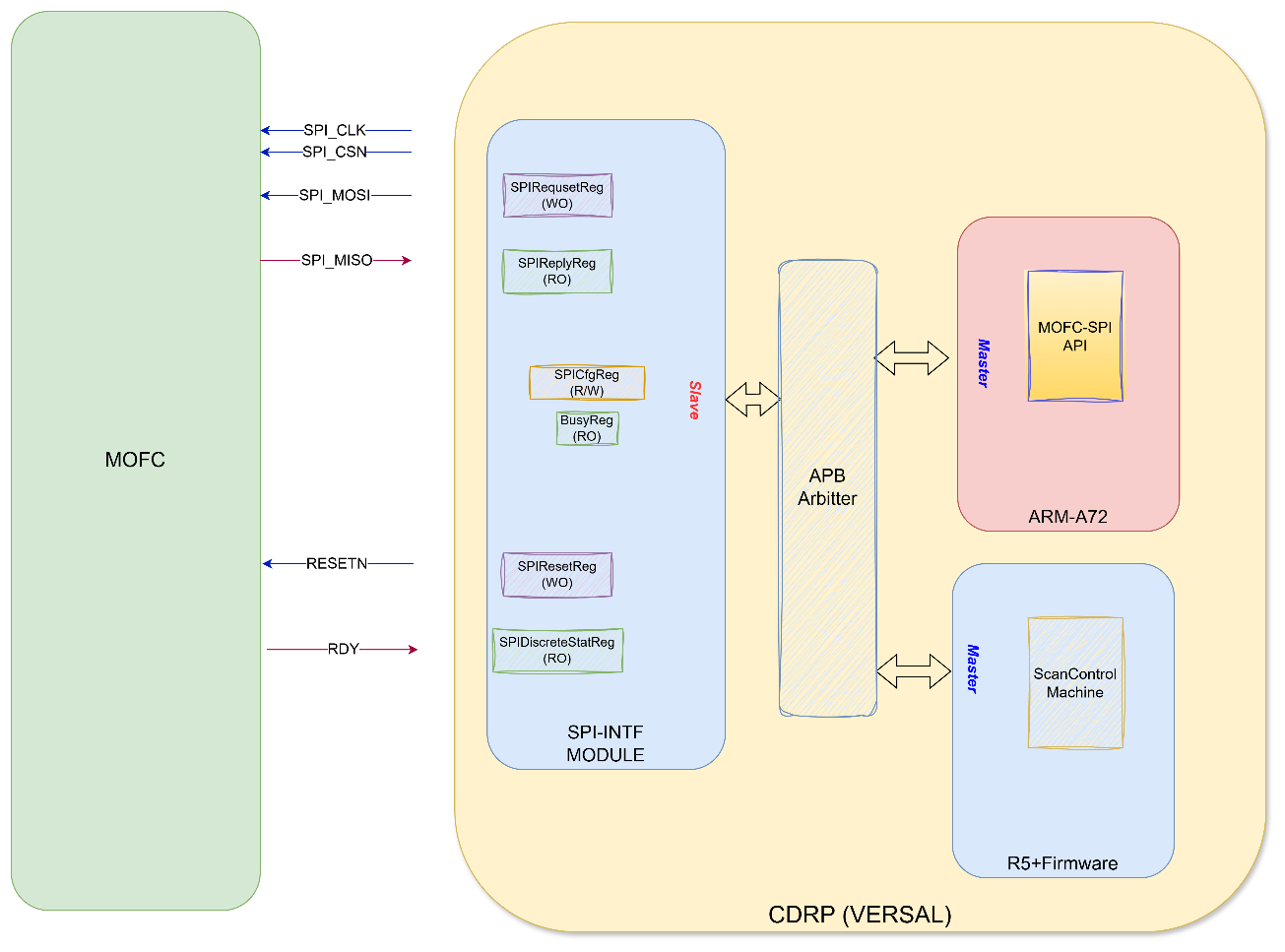
# Scope

This document defines the requirements for supporting the integration of the MOFC module withing the CDRP system, and cover the following aspects

* MOFC CDRP interface communication and discrete signal
* CDRP firmware requirements and architecture review for supporting the MOFC integration
* SW infrastructure API definition
* SW infrastructure test application for support testing and assignment This document serves as a reference for both the firmware development team implementing the SPI interface and the hardware design team ensuring compliance with the interface specifications

# System overview

The following block diagram provides a high-level overview of the CDRP and MOFC communication architecture.



The system is based on a master-slave architecture between the CDRP (Master) and the Miniature Octal Frequency Converter (MOFC) (Slave). The CDRP controls the MOFC's operational parameters

***Component Roles:***

MOFC (Miniature Octal Frequency Converter): An RF module operating as a slave device. It executes configuration commands received from the CDRP master.

CDRP : The master unit responsible for system control. It houses a dedicated SPI Interface Module to manage all communications with the MOFC.

CDRP internal architecture:

* CDRP SPI Interface Module: A firmware-based SPI Interface Module within the CDRP handles SPI transactions, sending the 32 bit of the SpiRequestReg toward the MOFC through MOSI (parallel to serial) and storing the reply arrive at MOSI at a reply register (serial to parallel), the module is accessed by two independent masters This module is accessed by two independent masters:
  + The SW API Application: Resides on ARM-A72 processor,
  + Firmware Scan control machine

.

* **Arbitration & Priority**: Since both masters can initiate SPI transactions concurrently, a hardware arbitration mechanism is required to prevent race conditions. The arbitration scheme grants priority access to the Firmware Scan Controller to ensure deterministic, real-time performance.

# MOFC interface definition

## Digital ICD

The following table describes the digital Interface Control Document (ICD) between the CDRP and the MOFC. All signal directions are from the CDRP's perspective

|  |  |  |  |
| --- | --- | --- | --- |
| Name | PIN | Function | Type |
| SPI\_Clk\_N | TBD | SPI Clock | RS485 input |
| SPI\_Clk\_P | TBD |
| SPI\_CSn\_N | TBD | Chip Select active low  Chip Select active low | RS485 input  RS485 input |
| SPI\_CSn\_P | TBD |
| SPI\_MOSI\_N | TBD | Master Out, Slave In | RS485 input |
| SPI\_MOSI\_P | TBD |
| SPI\_MISO\_N | TBD | Master In, Slave Out | RS485 output |
| SPI\_MISO\_P | TBD |
| Resetn\_N | TBD | System Reset active low | RS485 input |
| Resetn\_P | TBD |
| RF\_RDY\_N | TBD | Indicates the tuner RF is ready | RS485 output |
| RF\_RDY\_P | TBD |

Table 1: Digital ICD

## Communication format

The MOFC controller acts as the “SLAVE” and the main controller acts as the “MASTER”. The MOFC controller uses a typical SPI interface in terms of logic, but uses differential RS485 inputs/outputs. The SPI interface consists of four lines: SClk, CSn, MOSI, and MISO. The SPI interface uses 32-bit commands. All write commands transmitted to MOFC must be 32-bits long. Read commands contain the R/W bit set to 1 and the register address on MOSI while the returning data is clocked out on the MISO line during the data portion of the same 32-bit sequence.

### SPI Interface Specifications

The specified clock frequency for this application is 10MHz. Only the MASTER controller supplies the SPI 10MHz clock. The 10MHz clock is only supplied during SPI transactions.

### SPI write transaction

A typical Write transaction is shown below. The new command goes out on the MOSI line, and the data is registered by the SLAVE on each positive clock edge. The R/W bit is set to 0. The transaction is initiated when the MASTER pulls the CSn line low for a period of Tcs\_c before the first positive clock edge and held low for the remainder of the transaction. The MASTER then pulls the CSn line high Tc\_cs after the last negative clock edge. See timing constraints in the table below for minimum timing specifications.

A row of squares with numbers

Description automatically generated

Figure 1: SPI write transaction

### SPI Read transaction

A typical Read transaction is shown below. The Read transaction sequence starts out with the MASTER clocking out the R/W bit set to 1 and clocking out the address bits on each positive clock edge. The SLAVE then starts clocking out the read data on the next clock falling edge. The MASTER registers the 24 bits of data on the remaining 24 positive clock edges.

A diagram of a computer

Description automatically generated

Figure 2: SPI read transaction

### Timing constrains

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Description | Min | Max |
| SPI Clock Frequency | Maximum frequency of SPI Clock | 10 MHz |  |
| Tcs\_c | Time between CSn going low and the first positive clock edge | TBD[ns] |  |
| Tc\_cs | Time between last negative clock edge and CSn going high | TBD[ns] |  |
| SpiAccMinGap | Minimum Time gap between consecutive SPI access | 10[us] |  |

Table 2: Timing constarins

# MOFC SPI COMMAND SET

The MOFC memory space, described in the table below, contains **42 registers**, each **24 bits wide**. The registers have different access types — some are **write-only**, others **read-only**, and some are **read/write (R/W)**.

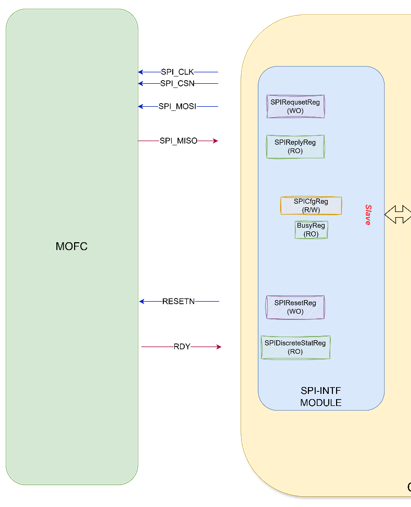
The **CDRP** configures the **MOFC** through multiple register writes. Once all required configurations are complete, writing to the **LOAD\_RF\_SETUP** register initiates the **MOFC assignment process**.

To ensure data integrity, a **checksum register** is implemented. The CDRP writes the calculated checksum value to this register, which triggers an internal verification process within the MOFC. The result of the checksum verification can be read from the **Status register**.

For more details on the register definitions, refer to the **MOFC\_IDD documentation**.



# SPI interface module



* ***Operation concept:*** when a write access to the SpiRequestReg is performed the 32 bit data is transferred toward the MOFC through the SPI\_MOSI pin, and the return MISO is stored in the SpiReplyReg
* **Busy**: This signal (a discrete line and/or a status register bit) indicates that an SPI transaction is in progress. It is asserted upon a write operation to the SpiRequestReg and is deasserted after a delay configured in the SPICfgReg. The default delay is 10 µs.
* **MOFC Reset:** The SPI\_INTF\_MODULE provides master access to the SPIResetReg. The value of this register is directly output to the MOFC's RESETN pin.
  + Since RESETN is an active-low signal, the SPIResetReg defaults to '1' to keep the MOFC out of reset.
  + The software API is responsible for generating a reset pulse by writing a '0' to the SPIResetReg, followed by a '1' after a short delay.
* ***RF Ready register***: This register is Read only register, it reflects the RF\_RDY discrete line arrive from MOFC
* ***advanced functionality requirement*** 
  + Default SPI\_CLK generated by the SPI\_INTF module is 10[mhz] for supporting long distance configuration option include 5[mhz] and 2.5[mhz]
  + The return data (MOSI) by default is samples at the rising of the SPI\_CLK, for supporting long distance the sample point can be delay in resolution of 10[ns] delay can be between 0 (default) and up 310[ns]
* ***Configuration fields***
  + ***SPI\_CLK\_RATE:*** 
    - 0: default 10[mhz],
    - 1: 5[mhz]
    - 2: 2.4[mhz]
  + ***SMPL\_POINT\_DLY***: 5 bits for 0 to 31, step =10[ns], minimum value 0, maximum 310[ns]
  + **MIN\_GAP\_BETWEEN\_SPI\_ACCESS**: 8 bits, step=1[us] default 10 (10[us))

# Software API, for interfacing with MOFC

Software API communicate with MOFC through SPI accesses, the MOFC is memory mapped and all access is perform using single SPI tarnsactions Red/Write of 32 bits.the structure of the 32 bits is

* Bit [31] R/W: indicate access is Read(0) or Write(1)
* Bits[30:24]: 7 bits of Register Address
* Bits[23:20]: 24 bit register data

## Global API functions

* SpiWrite(Addr,Data): perform Spi write transaction write Data to a register specify by Addr
* SpiRead(Addr): perform Spi Read return data from register specify by address
* SpiReadVerify(Addr,ExpectedData): perform SpiRead, and compare data to ExpexctedData, returns true is equal and False if not

## Specific API functions

* SpiGetStatusReg():
* SpiGetFPGAVerReg():
* SpiGetSerialNum():
* SpiSetGlobalCfg():
* SpiGetGlobalCfg():
* SpiSetBitCfg()
* SpiGetBitCfg()
* SpiSetChannelCfg(CHID)
* SpiGetChannelCfg(CHID)
* SpiWriteCheckSum(DataPayload)
* SpiReadCheckSum()
* SpiLoadRfSetup()

# Software test application for MOFC integration

GUI Test application will include 2 pages/Tab the first enable the user to assign the MOFC and verify the correct behavior of all API commands, the second TAB will be used by the user for reading and extract all MOFC registers

## GUI test applications for MOFC assignment

Figure below illustrates GUI first tab/page functionality for MOFC assignment



### MOFC assignment

* Configuration fields enable user to configure the
  + Global Config register
  + Bit Config register
  + 8 Channels configurations
* Buttons
  + WriteGlobalCfg: clicking this button initiate write to the global configuration register
  + SendConfigMessage – Clicking this button initiates the write sequence to the MOFC BitCfg register and the Channel Configuration registers.  
    The specific channels to be configured depend on the OpMode field within the GlobalCfg register:
    - Common Octal Mode: Only channel 0 is configured
    - Common Quad Mode: Channels 0 and 1 are configured
    - Single Channel Mode: All channels (0–7) are configured

After completing the Bit and Channel configuration, if the WriteCheckSum checkbox is selected, the GUI will perform a checksum write to the Checksum register.  
For more information on checksum generation, refer to the MOFC IDD document.

* + SendLoadRF: clicking this button initiate write to the LoadRF resister, and will initiate the MOFC assignment process

### Reading Status Reg and Versions

* Clicking on the ReadStatusReg will read, extract, and display the StatusReg fields
* Clicking on the Read FPGA Ver will read, extract, and display the FPGAVerReg fields
* Clicking on the Read Serail number will read, extract, and display the SerailNumn register fields

### Global read/write register

* Click the ReadReg button will initiate a SPI read request to the address define by the user, and will display the read data
* Clicking the WriteReg will initiate SPI write access address and data define by the user

## GUI test applications for MOFC read configuration

Figure below illustrates GUI second tab/page functionality for MOFC read configuration, clicking the Read Configuration button will present

* GlobalCfgReg fields
* BitConfigReg fields
* Channesl configuration (all channels) fields
* StatusReg fields
* FPGA\_Ver register fields
* SerailNum register fields

