MIPI® Alliance Specification for RF Front-End Control Interface

Version 1.10 – 26 July 2011

MIPI Board Approved 2-Nov-2011

Further technical changes to this document are expected as work continues in the RF Front-End Control Working Group

NOTICE OF DISCLAIMER

The material contained herein is not a license, either expressly or impliedly, to any IPR owned or controlled by any of the authors or developers of this material or MIPI®. The material contained herein is provided on an "AS IS" basis and to the maximum extent permitted by applicable law, this material is provided AS IS AND WITH ALL FAULTS, and the authors and developers of this material and MIPI hereby disclaim all other warranties and conditions, either express, implied or statutory, including, but not limited to, any (if any) implied warranties, duties or conditions of merchantability, of fitness for a particular purpose, of accuracy or completeness of responses, of results, of workmanlike effort, of lack of viruses, and of lack of negligence.

All materials contained herein are protected by copyright laws, and may not be reproduced, republished, distributed, transmitted, displayed, broadcast or otherwise exploited in any manner without the express prior written permission of MIPI Alliance. MIPI, MIPI Alliance and the dotted rainbow arch and all related trademarks, tradenames, and other intellectual property are the exclusive property of MIPI Alliance and cannot be used without its express prior written permission.

ALSO, THERE IS NO WARRANTY OF CONDITION OF TITLE, QUIET ENJOYMENT, QUIET POSSESSION, CORRESPONDENCE TO DESCRIPTION OR NON-INFRINGEMENT WITH REGARD TO THIS MATERIAL OR THE CONTENTS OF THIS DOCUMENT. IN NO EVENT WILL ANY AUTHOR OR DEVELOPER OF THIS MATERIAL OR THE CONTENTS OF THIS DOCUMENT OR MIPI BE LIABLE TO ANY OTHER PARTY FOR THE COST OF PROCURING SUBSTITUTE GOODS OR SERVICES, LOST PROFITS, LOSS OF USE, LOSS OF DATA, OR ANY INCIDENTAL, CONSEQUENTIAL, DIRECT, INDIRECT, OR SPECIAL DAMAGES WHETHER UNDER CONTRACT, TORT, WARRANTY, OR OTHERWISE, ARISING IN ANY WAY OUT OF THIS OR ANY OTHER AGREEMENT, SPECIFICATION OR DOCUMENT RELATING TO THIS MATERIAL, WHETHER OR NOT SUCH PARTY HAD ADVANCE NOTICE OF THE POSSIBILITY OF SUCH DAMAGES.

Without limiting the generality of this Disclaimer stated above, the user of the contents of this Document is further notified that MIPI: (a) does not evaluate, test or verify the accuracy, soundness or credibility of the contents of this Document; (b) does not monitor or enforce compliance with the contents of this Document; and (c) does not certify, test, or in any manner investigate products or services or any claims of compliance with the contents of this Document. The use or implementation of the contents of this Document may involve or require the use of intellectual property rights ("IPR") including (but not limited to) patents, patent applications, or copyrights owned by one or more parties, whether or not Members of MIPI. MIPI does not make any search or investigation for IPR, nor does MIPI require or request the disclosure of any IPR or claims of IPR as respects the contents of this Document or otherwise.

Questions pertaining to this document, or the terms or conditions of its provision, should be addressed to:

MIPI Alliance, Inc.
c/o IEEE-ISTO
445 Hoes Lane
Piscataway, NJ 08854
Attn: Board Secretary

Contents

Figures

Tables

Release History

Date	Release	Description
2010-05-03	v0.80.00	Second Draft Review Release
2010-07-29	v1.00.00	Board-Approved Release
2011-12-05	v1.10	Board-Approved Release

1 Introduction

1 The RF Front-End Control Interface (later referred to as RFFE) was developed to offer a common and wide-spread method for controlling RF front-end devices. There are a variety of front-end devices, including Power Amplifiers (PA), Low-Noise Amplifiers (LNA), filters, switches, power management modules, antenna tuners and sensors. These functions may be located either in separate devices or integrated into a single device, depending on the application.

2 RFFE should not be confused with the MIPI Alliance DigRF specifications, [MIPI01] and [MIPI02]. DigRF specifies the interface between the baseband and RF ICs, whereas RFFE is mainly an RF front-end-dedicated control interface. The key driver for DigRF is to offer a very high-speed interface for carrying digital RF IQ data and RF control information. RFFE, on the other hand, is targeted purely towards the control of RF front-ends, and therefore does not incorporate the signal paths associated with the front-end devices being controlled. DigRF provides only a point-to-point configuration, and thus requires multiple instantiations for complex configurations. In contrast to DigRF, RFFE supports point-to-multipoint connectivity, thus allowing for the control of complex RF systems comprising multiple front-end devices.

3 The trend in mobile radio communications is towards complex multi-radio systems comprised of several parallel transceivers. This implies a leap in complexity of the RF front-end design. Thus, the RFFE bus must be able to operate efficiently in configurations from the simplest one Master and one Slave configuration to potentially multi-Master configurations with tens of Slaves. The emphasis of this version of the specification is on configurations with only one Master, while also providing for future expansion to multiple Master configurations. Future versions of this specification might thus allow more complex configurations that provide for multiple Masters in addition to multiple Slaves.

4 RF front-end modules are sometimes developed in process technologies unlike bulk digital CMOS. Diverse technology choices are necessary to meet the functional and performance requirements of the application. The downside is that suitability for digital design might be quite low. In some of these technologies the implementation of digital logic might be costly, so a prerequisite of the RFFE design was to offer options to reduce Slave control complexity to a minimum (approximately 300 to 500 gates). Simplicity has been a core driver in RFFE development. The RFFE specification, positioned at the low complexity end of all interfaces, is optimized for Master and Slave implementation simplicity without sacrificing a broad set of features.

5 One challenge for RFFE is presented by the need in many radio applications for time-accurate control. This is addressed in RFFE by utilizing a relatively high bus clock frequency of 26 MHz and by the introduction of time-accurate triggering mechanisms to allow control of timing-critical functions in multiple devices. This is predicated on the expectation that a simple Slave lacks the required timing accuracy, and thus is command-driven.

6 The RFFE specification is based on *MIPI Alliance Specification for System Power Management Interface (SPMI)* [MIPI03] developed by the SPM Working Group. The intention has been to preserve compatibility with SPMI by selection of a reduced SPMI feature set in RFFE. RFFE-specific features have been added to that set. The relevant parts of the SPMI specification are copied into this document to make it a complete stand-alone specification. Compatibility to SPMI might be maintained, depending on the impact to the RFFE Specification, by updating the relevant sections in future releases.

1.1 Scope

7 The scope of this document is to specify the control interface for RF front-end devices. Analog signal paths required between front-end devices and other elements that control and utilize the devices, are outside the scope of this document.

8 A voltage reference is introduced as part of the control interface. The implementation of this voltage source is not specified, although a set of electrical characteristics are defined. This document also defines interface-specific procedures, and also provides alternative means to perform certain actions. Implementers may

determine which optional procedures and alternative means are supported by a device. Since a Master implementation supports all options, the functional choices are intended primarily for Slave device implementations.

1.2 Purpose

9 RFFE provides a low-complexity solution to meet the cost and performance targets of RF front-end components. It offers extensibility from simple configurations with one Slave on a single bus, all the way to complex configurations with many Slaves on a single bus, or distributed on multiple buses. This eases both the RF and front-end module design by requiring a mobile terminal to support only a single control interface. Ideally, this will lead to a broader range of control-compatible components, and to larger markets for RF front-end devices.

2 Terminology

The MIPI Alliance has adopted Section 13.1 of the IEEE Specifications Style Manual, which dictates use of the words “shall”, “should”, “may”, and “can” in the development of documentation, as follows:

The word *shall* is used to indicate mandatory requirements strictly to be followed in order to conform to the Specification and from which no deviation is permitted (*shall* equals is *required to*).

The use of the word *must* is deprecated and shall not be used when stating mandatory requirements; *must* is used only to describe unavoidable situations.

The use of the word *will* is deprecated and shall not be used when stating mandatory requirements; *will* is only used in statements of fact.

The word *should* is used to indicate that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required; or that (in the negative form) a certain course of action is deprecated but not prohibited (*should* equals is *recommended that*).

The word *may* is used to indicate a course of action permissible within the limits of the Specification (*may* equals *is permitted*).

The word *can* is used for statements of possibility and capability, whether material, physical, or causal (*can* equals *is able to*).

All sections are normative, unless they are explicitly indicated to be informative.

Numbers are decimal unless otherwise indicated. A prefix of 0x indicates a hexadecimal number, while a prefix of 0b indicates a binary number.

2.1 Definitions

Address Frame: A series of nine bits with eight bits representing address information and a single parity bit.

Broadcast: A procedure of sending a Command Sequence to multiple recipients simultaneously using either Broadcast ID or GSID.

Broadcast ID: A unique GSID defined as 0b0000 addressing all Slaves simultaneously.

Bus Idle: The RFFE bus is idle when both the SCLK and SDATA are at a logic level zero between the end of a Command Sequence.

Bus Park Cycle: A single clock cycle that occurs when the SDATA signal control may change between devices during, or at the end of, a Command Sequence.

Command Frame: A series of thirteen bits with four bits representing a Slave address, eight bits representing an RFFE command and a single parity bit.

Command Sequence: A bus transaction on the RFFE bus that begins with a SSC, a Command Frame, potentially Data and Address Frames and ends with a Bus Park Cycle.

Data Frame: A series of nine bits with eight bits of data and a single parity bit.

Group Slave ID: A 4-bit number assigned to one or more Slaves identifying them on the RFFE bus as a group.

Full Speed: Operating RFFE bus with a fundamental SCLK frequency between 13 MHz and 26 MHz.

Half Speed: Operating RFFE bus with a fundamental SCLK frequency between 32 kHz and 13 MHz.

Master: A device on the RFFE bus that drives the SCLK line and controls the transmission of all Command Sequences.

No Response Frame: A Data or Address Frame that is used when no applicable data is available.

Slave: A device on the RFFE bus that is not capable of driving the SCLK line, i.e. not a Master.

Slave ID: A 4-bit number assigned to a Slave. It may be either Unique Slave ID or Group Slave ID.

Unique Slave ID: Unique 4-bit number assigned to a Slave identifying it on the bus.

2.2 Abbreviations

e.g.	For example (Latin: exempli gratia)
i.e.	That is (Latin: id est)
High-Z	High impedance
SDATA	RFFE data
SCLK	RFFE clock
SCLKint	Internal serial clock used within a Master
VIO	RFFE Bus I/O Voltage Level

2.3 Acronyms

3GPP	3rd Generation Partnership Project
ASM	Antenna Switch Module
DDB	Device Descriptor Block
EDGE	Enhanced Data-Rates from GSM Evolution
EGPRS	Enhanced General Packet Radio System
EMI	Electromagnetic Interference
FEM	Front-End Module
GPRS	General Packet Radio System
GSID	Group Slave Identifier
GSM	Global System for Mobile Communications
HSPA	High Speed Packet Access
HW	Hardware
IC	Integrated Circuit
IEEE	Institute of Electrical and Electronics Engineers
ISTO	Industry Standards and Technology Organization
I/O	Input/Output
LSB	Least Significant Bit
LTE	Long Term Evolution

60	MIMO	Multiple Input Multiple Output
61	MIPI	Mobile Industry Processor Interface
62	MSB	Most Significant Bit
63	PA	Power Amplifier
64	PCB	Printed Circuit Board
65	RF	Radio Frequency
66	RFFE	RF Front-End Control Interface
67	RFIC	Radio Frequency Integrated Circuit
68	RX	Receiver
69	RCS	Request Capable Slave
70	SA	Slave Address
71	SW	Software
72	SID	Slave Identifier
73	SSC	Sequence Start Condition
74	SPM	System Power Management
75	SPMI	System Power Management Interface
76	TX	Transmitter
77	UMTS	Universal Mobile Telecommunications System
78	USID	Unique Slave Identifier

3 References

79 [MIPI01] *MIPI Alliance Specification for DigRF^{SM} v4*, Version 1.10, MIPI Alliance, Inc., (In Press)

80 [MIPI02] *MIPI Alliance Specification for Dual Mode 2.5G/3G Baseband/RFIC Interface*, Version 3.09.06, MIPI Alliance, Inc., 14 December 2010

81 [MIPI03] *MIPI Alliance Specification for System Power Management Interface (SPMI)*, Version 1.00.00, MIPI Alliance, Inc., 27 October 2008

82 [MIPI04] MIPI Alliance, Inc., Current Members - List of all MIPI Manufacturer IDs, “List of MIPI Manufacturer IDs”, < http://mid.mipi.org/>

83 [MIPI05] *MIPI Alliance Specification for Device Descriptor Block (DDB)*, Version 0.82.01, MIPI Alliance, Inc., 30 October 2008

84 [MIPI06] *MIPI Alliance Application Note for RF Front-End Control Interface*, Version 1.10, MIPI Alliance, Inc., 15 November 2011

4 Architecture and Operations Overview

85 This section is intended to convey an overview of the architecture and operational details of the RFFE interface.

4.1 Overview

86 RFFE is a two-wire, serial interface intended to be used to connect Radio Frequency ICs (RFIC) of a mobile terminal to their related Front-End Modules (FEM). The RFFE interface enables systems to efficiently control various FEMs in next generation mobile terminals with increased complexity of performance supporting multi-mode, multi-band and multiple antennas, all with a minimum number of wires and pins using a single RFFE bus. It is designed to support existing 3GPP standards such as LTE, EGPRS, UMTS, HSPA, etc. and also other, non-3GPP air interfaces. The RFFE interface is based on *MIPI Alliance Specification for System Power Management Interface (SPMI)* [MIPI03]. The RFFE interface is intended to be efficient, flexible, and extensible, accommodating many variations in the overlying system design, while providing interoperability at the interface level between compliant RFICs and FEMs. The ability to design one common control interface which may be reused for all of these modules helps reduce front-end complexity, and hence speed up the time-to-market for these terminals.

87 Within the mobile terminal, the RFIC is the Master and the FEMs are the Slaves on the RFFE bus. Command Sequences on the bus may only be initiated by the Master. A Slave shall not initiate Command Sequences on the bus. This specification defines the operating states, the Command Sequence set, the physical interface, and the protocol for data communication between RFFE devices on an RFFE bus to insure the compatibility of control and data transfers. The RFFE Command Sequence set includes Slave addressing, control of the Slave operating state, register read from and register write to Slaves, as well as Command Sequences supporting the use of Device Descriptor Block [MIPI05].

88 The key pillars of the RFFE design include the following considerations:

89 • Minimize the wiring effort in the front-end of a mobile terminal

90 • Minimize pin count

91 • Ease and optimize control flow

92 • Ensure minimal EMI contributions due to RFFE bus

93 • Minimize complexity for the Slave

94 • Add flexibility and scalability, allowing use of multiple receivers and transmitters simultaneously

95 The basic configuration of the RFFE interface and its bus structure are shown in Figure 1. As RFFE is based on the SPMI interface it shall have two signals, one serial bidirectional data signal (SDATA) and one clock signal (SCLK) controlled by the Master. Any additional signals present on an RFFE device shall not change the behavior of the RFFE interface protocol or prevent the operation of the RFFE bus described in this specification.

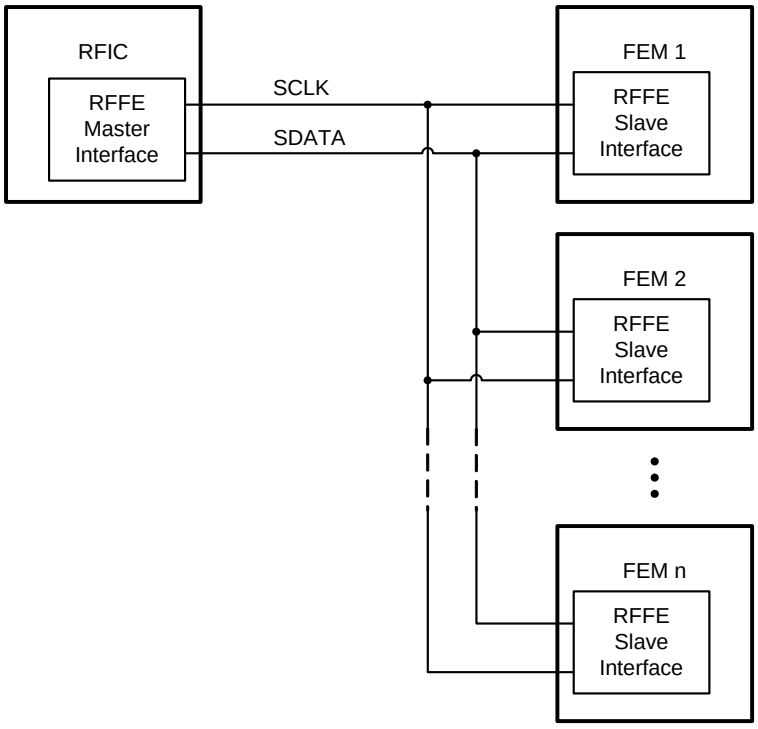

Figure 1 RFFE Interface and Bus Structure

4.1.1 Topology

4.1.1.1 Basic

96 Figure 2 shows a Basic Configuration of an RFFE bus implementation based on a minimal topology consisting of a RFIC with one RX and one TX path connected to one antenna. The main characteristic of the Basic Configuration is that there is only one RFFE bus where all front-end components are connected. The RFIC is the Master on the RFFE bus and all front-end components act as Slaves. The TX signal path starts at the RFIC and may comprise various outputs for different radio standards or frequency bands. Depending on the detailed architecture these analog outputs may be connected to a set of different gain or power amplifiers, which usually need to be controlled. These gain or power amplifiers may be separate for each output or may also be shared for several outputs. Following the TX direction towards the antenna there are bandlimiting filters, which may be configurable for different scenarios, the antenna switch used to select RX and TX directions as well as different bands, and finally the antenna tuning-module. In addition, these components may be accompanied by various sensors for temperature, power, voltage, etc. and adjustable power supplies for the front-end components like LDOs or DC/DC converters for PAs.

97 Complexity in terms of control functionality of these various front-end component types may be different as well as process technology and manufacturing requirements of such components. Therefore, the RFFE bus needs to cover a wide range of configurations and application complexity while simultaneously enabling a small implementation in low density process technologies. The various front-end components also may have very different requirements regarding real time control performance, number of parameters to be controlled, amount and frequency of data to be read back, etc.

98 Furthermore, depending on the topology of the system and the use cases to be supported several front-end components may need to receive control information at almost the same time. The absolute number of front-

end components in the overall system is a very important boundary condition since each component needs to be individually addressable.

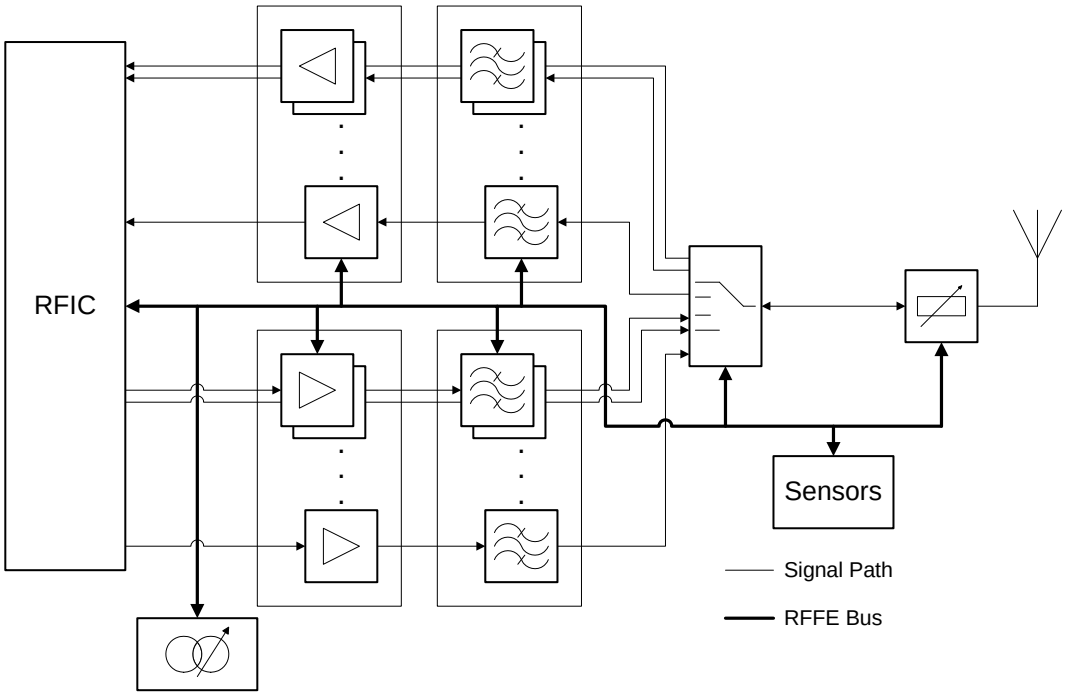

Figure 2 Basic Configuration

4.1.1.2 Diversity

99 Figure 3 shows a topology supporting receive diversity (RxDiv). The primary difference from the basic configuration shown in Figure 2 is the additional receive path with a separate antenna connected to one RFIC. The additional front-end components are connected to the same RFFE bus. This scenario might have increased requirements regarding bandwidth and addressable components for the RFFE bus.

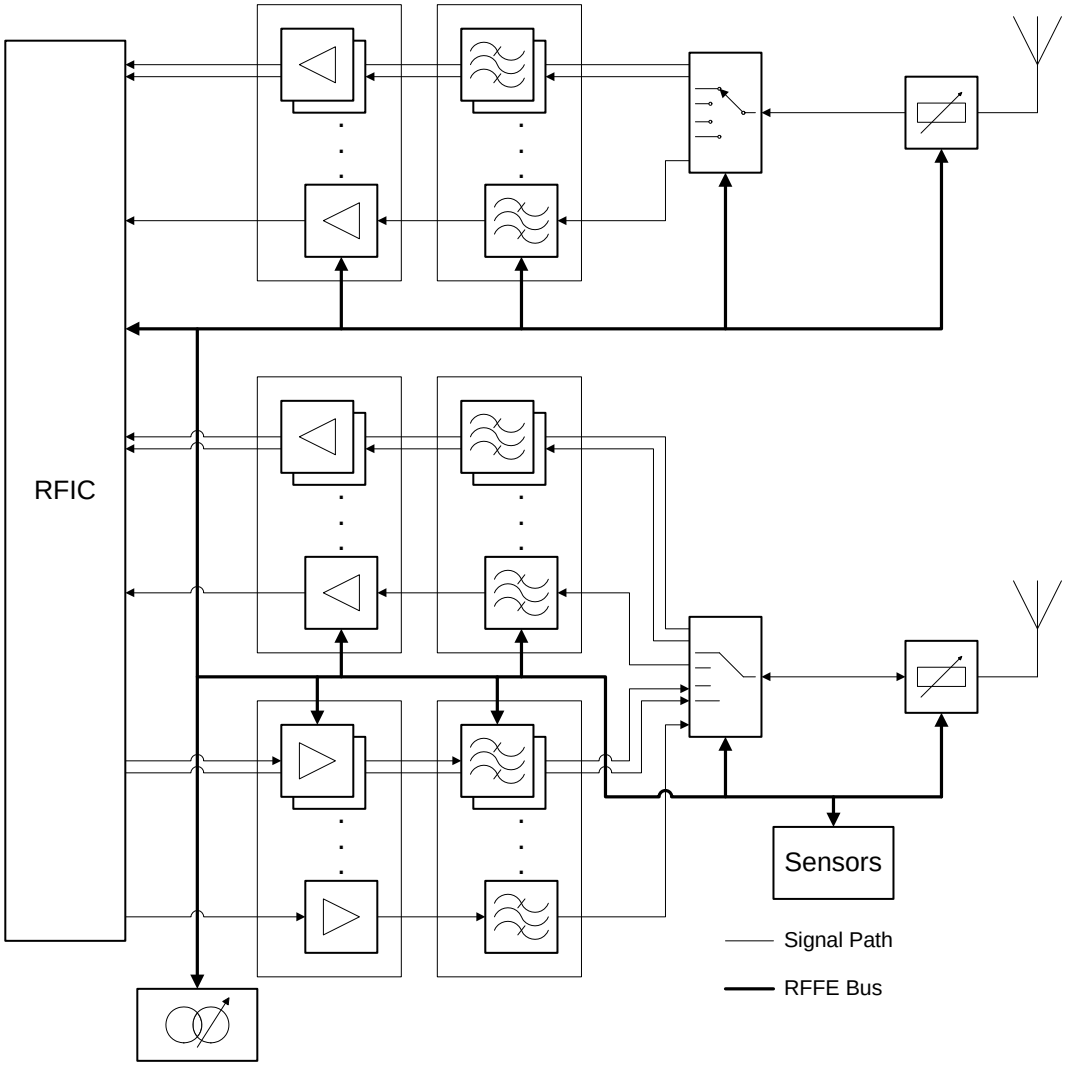

Figure 3 Diversity Configuration

4.1.1.3 MIMO

100 The MIMO configuration shown in Figure 4, as compared to the diversity configuration shown in Figure 3, represents a further step in terms of complexity for the RFFE bus. Now there are two complete and independent RX and two complete and independent TX paths, which allows MIMO operation. Therefore, the number of front-end components increases again. Higher traffic on the RFFE bus is expected and a higher number of front-end components need to be addressed.

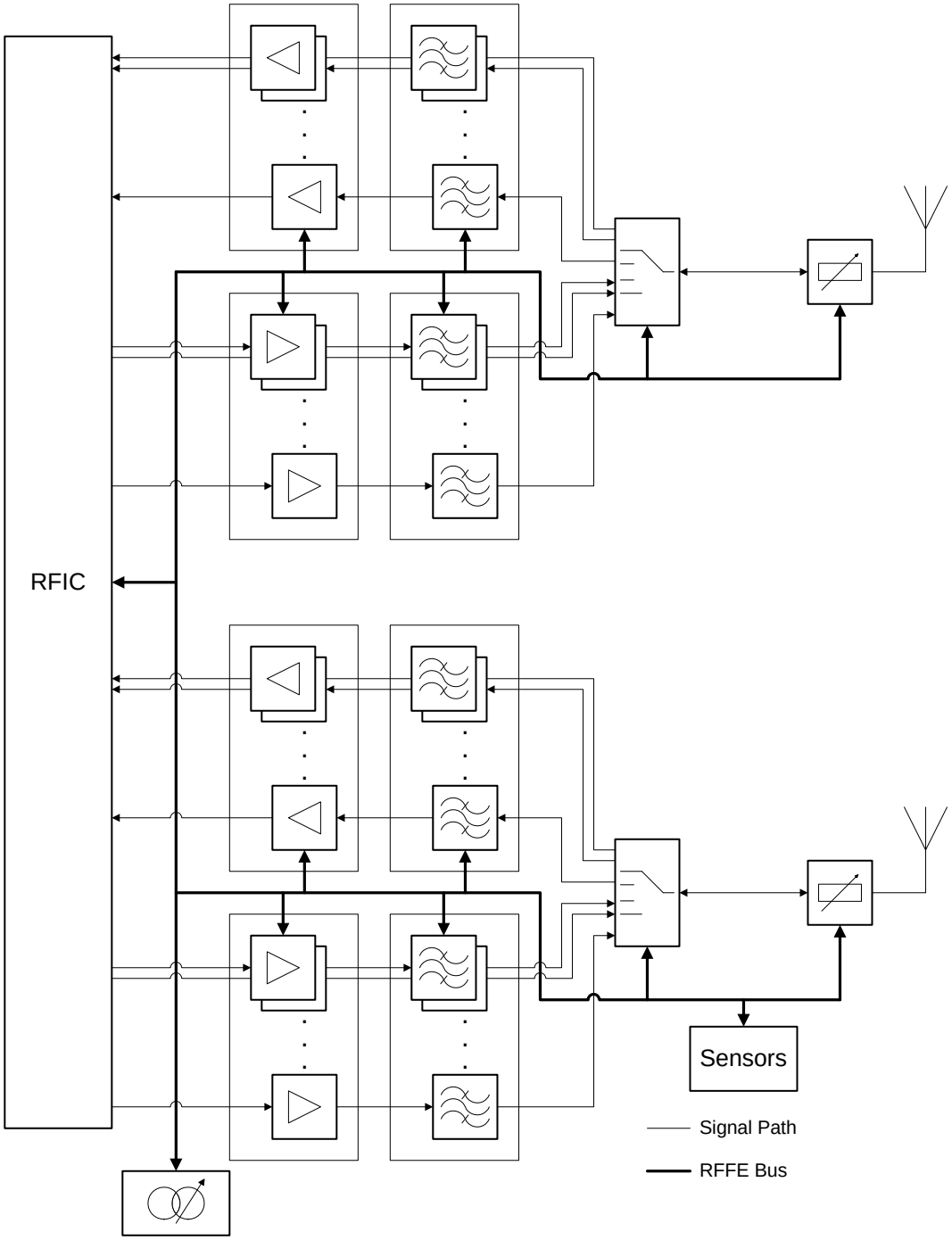

Figure 4 MIMO Configuration

4.1.1.4 Dual-Bus Basic

101 Figure 5 shows a topology employing two RFFE busses for receive and transmit front-end devices. The primary difference to the topology in the basic configuration shown in Figure 2 is that there are two RFFE busses connected to one RFIC. This configuration supports a higher number of front-end devices and keeps the capacitive load on a single bus low. The dual bus topology also allows reduction of crosstalk from the TX

RFFE bus activity into noise critical RX devices. The transmission of timing critical messages is improved with a reduced traffic load.

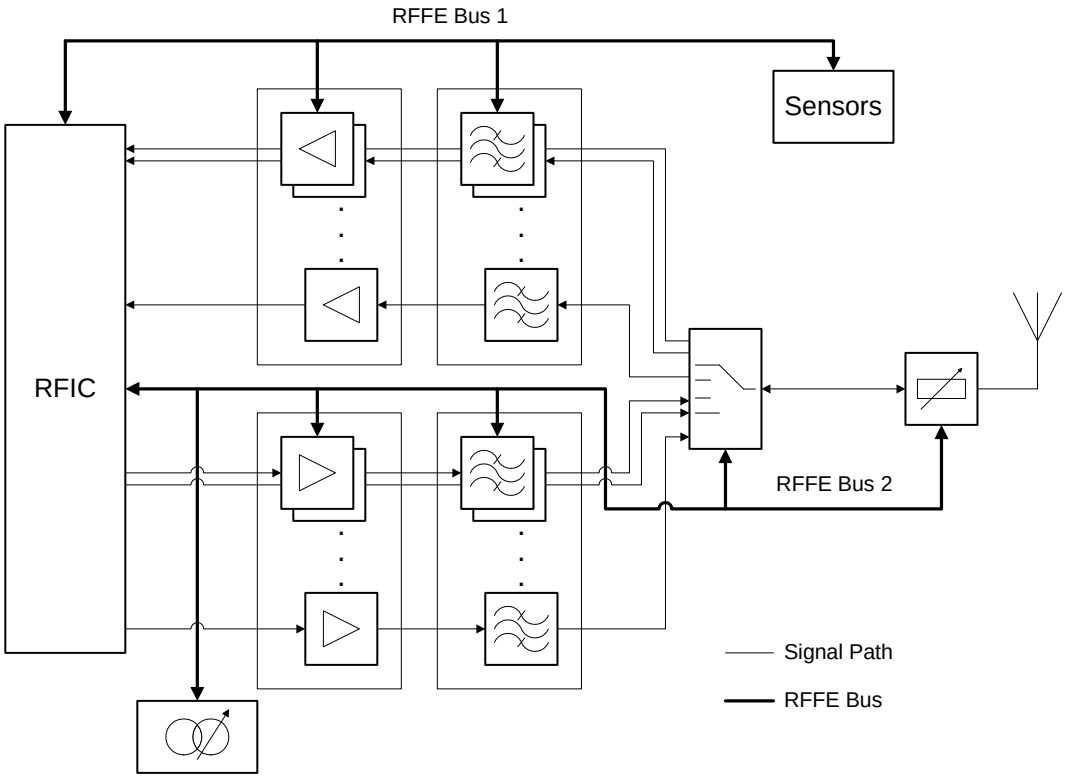

Figure 5 Dual-bus Basic Configuration

4.1.1.5 Dual-Bus MIMO

102 Figure 6 shows a topology employing two RFFE busses for a MIMO system. The primary difference to the MIMO configuration shown in Figure 4 is that there are two RFFE busses connected to one RFIC. Timing critical transmission is relaxed due to splitting the bus or due to advantageous allocation of devices to the separate buses.

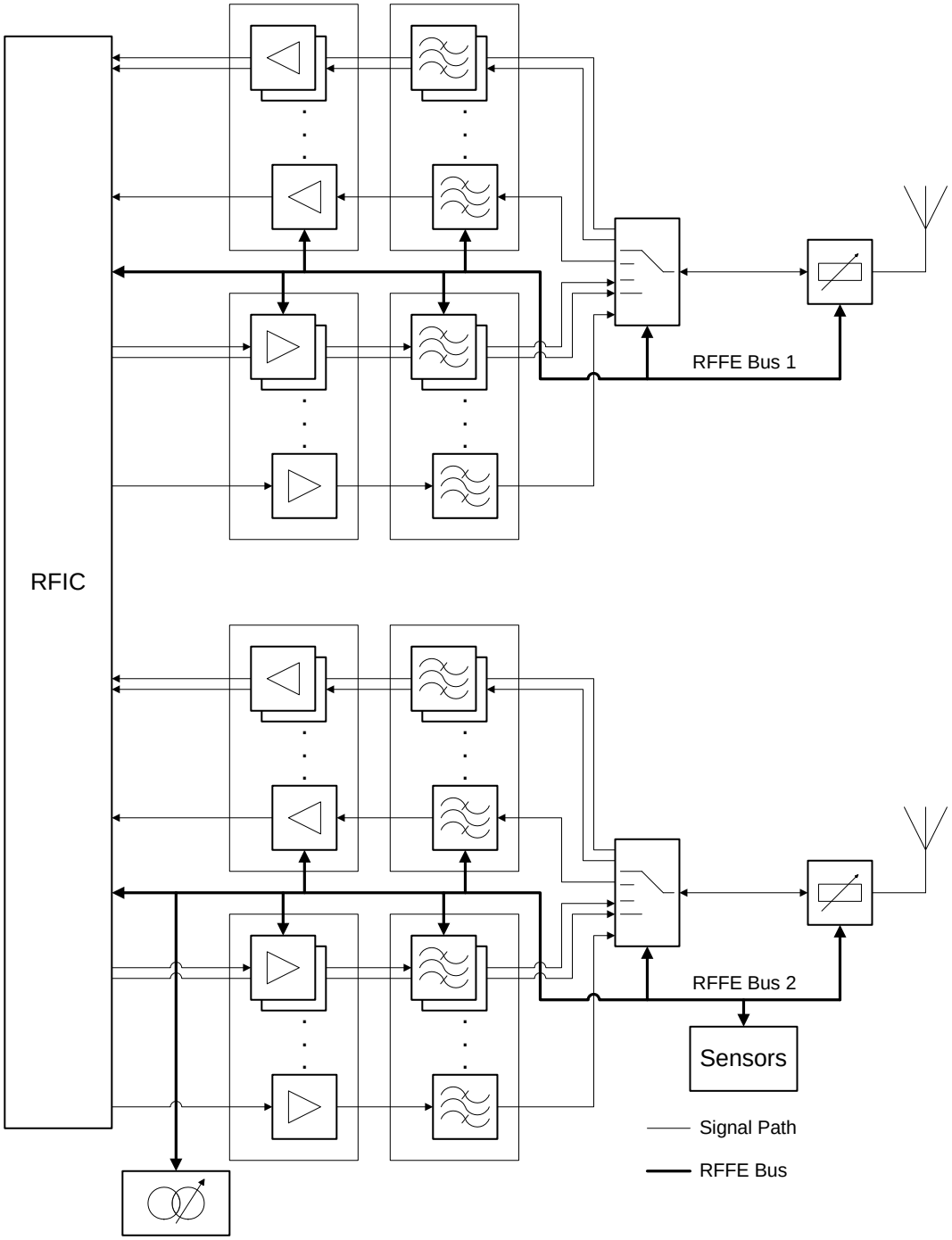

Figure 6 Dual-bus MIMO Configuration

4.1.2 Device Identification

103 Device Allocation is described in Section 6.8 and Section 6.9.

104 Figure 7 shows the state diagram of the dynamic Slave address allocation scheme for programming a new USID to a Slave. As explained in Section 6.8.3, a USID is programmed if both the selected PRODUCT_ID

and MANUFACTURER_ID match to the respective values of the Slave. All Slaves supporting programmable USID in a particular system must have a different combination of PRODUCT_ID, MANUFACTURER_ID and default USID in order to ensure that each device is uniquely addressable.

105 By issuing a reset issued by Command Sequence or signal, the USID shall be returned to its default value.

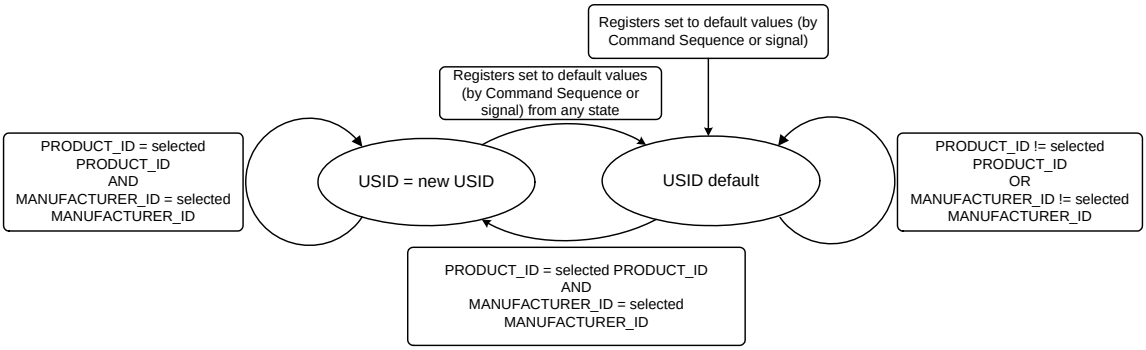

Figure 7 State Diagram of Programming a New USID

4.2 Read and Write Timing

106 The minimum transition time on the SCLK or SDATA line is directly related to the physical bus distance and the level of EMI generated from the bus lines (see Section 5.4). The maximum physical distance between a transmitter and a receiver is expected to be less than 15 cm, which implies a minimum transition time of 2.1 ns for both SCLK and SDATA. The lower the desired EMI and the longer this bus distance, the longer the transition time needed to generate a reliable clock signal without generating interference, reflections, voltage overshoot or undershoot.

4.2.1 RFFE Clock (SCLK)

107 The Master shall drive the RFFE clock signal. All clock waveforms shall start and end with the SCLK signal at logic level zero. A Slave shall not drive the SCLK signal.

108 The maximum operation frequency of SCLK is 26 MHz, although lower rates may be utilized. Timing requirements in Section 4.2.1 and Section 4.2.2 shall be fulfilled with any SCLK rate employed within the range specified in Table 1.

Table 1 RFFE SCLK Specification

Symbol	Description	Min	Max	Unit
F_{SCLK}	SCLK Frequency	0.032	26	MHz
T_{SCLK}	SCLK Period (1/FSCLK)	0.038	32	µs
F_{SCLK_HALF}	SCLK Half-Speed Frequency	0.032	13	MHz
T_{SCLK_HALF}	SCLK Half-Speed Period	0.077	32	µs

4.2.1.1 Specifications for the Master SCLK Driver

109 SCLK shall not toggle during idle and inactive time periods. SCLK shall only run while data is being transferred on the bus; otherwise SCLK is at logic level zero.

110 To reduce active power consumption, the SCLK line is not terminated. To avoid reflections and over voltage problems on such a bus system, the SCLK line shall be transition time controlled for Full Speed operation. This constraint makes a conventional CMOS I/O unsuitable for driving the SCLK line at Full Speed. The SCLK line may be driven at Half Speed during readback portion of the Command Sequence using a carefully matched CMOS driver with or without transition time control.

111 When driving the test load specified in Section 5.2 the SCLK line driver shall conform to the timing characteristics shown in Figure 8.

112 A Slave might not be able to support a 26 MHz clock frequency during a read operation. In this case, known as Half Speed operation, a 13 MHz clock frequency may be implemented for only the readback as described in Section 6.7.3.

113 Figure 8 shows the Clock Timing Diagram defining the clock output transition time ($T_{SCLKOTR}$) for rising from a voltage level of V_{OLmax} to V_{OHmin} and falling V_{OHmin} to V_{OLmax}. The durations for the signal above V_{OHmin} and below V_{OLmax} are defined by clock output high time (T_{SCLKOH}) and clock output low time (T_{SCLKOL}), respectively.

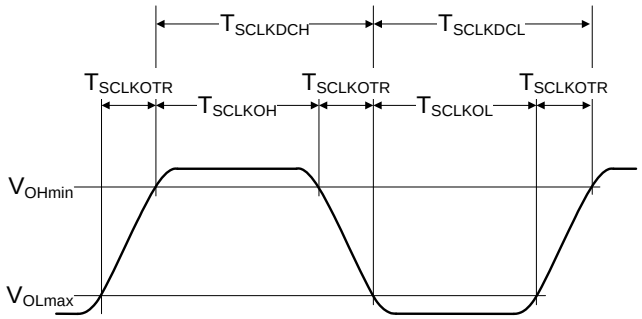

Figure 8 Clock Driver Output Waveform Constraints

114 The Clock timing specification is given in Table 2. Half Speed Device timing is valid for readback operation only as described in Section 6.7.3. V_{OHmin} and V_{OLmax} are defined in Table 8.

Table 2 Output Timing Characteristics

Symbol	Description	Half Speed Device		Full Speed Device		Units
		Min	Max	Min	Max	
T_{SCLKOH}	Clock Output High Time	24		11.25		ns
T_{SCLKOL}	Clock Output Low Time	24		11.25		ns
$T_{SCLKOTR}$	Clock Output Transition (Rise/Fall) Time[1]	3.5	10	3.5	6.5	ns
$T_{SCLKDCH}$	Clock Output Duty Cycle, High Time[2,3]	45	55	45	55	%
$T_{SCLKDCL}$	Clock Output Duty Cycle, Low Time[2,3]	45	55	45	55	%

1. *The minimum limit for $T_{SCLKOTR}$ applies for all valid SCLK frequencies, F_{SCLK}.*
2. *$T_{SCLKDCH}$ is defined to consist of T_{SCLKOH} + $T_{SCLKOTR}$(Fall); thus, it consists of the time that SCLK is a valid high level, plus the time until SCLK achieves a valid low level by exceeding V_{OLmax}. Similarly, $T_{SCLKDCL}$ is defined to consist of T_{SCLKOL} + $T_{SCLKOTR}$(Rise); it is thus the time that SCLK is a valid low level, combined with the time until SCLK exceeds V_{OHmin}.*
3. *$T_{SCLKDCH}$ and $T_{SCLKDCL}$ are expressed as a percentage of the SCLK period, T_{SCLK}. The limits expressed apply for any valid SCLK frequency, F_{SCLK}.*

115 The rise and fall times of the clock driver constrain both the maximum operating frequency and length of the SCLK line.

116 All timing characteristics are referenced to V_{OH} and V_{OL} in Section 5.1.1.

4.2.1.2 Specifications for SCLK Input

117 During a read, the critical path is determined by the SCLK Input High Time.

118 The timing requirements for correct operation of an RFFE device shall meet the minimum limits of the SCLK input high and low times given in Table 3. A glitch rejection filter may be used on the SCLK input. The clock receiver shall be capable of receiving slowly changing edges without glitching. An RFFE device shall implement inputs with hysteresis on the SCLK pin as defined in Table 7. It is important to note that the input thresholds shall employ threshold windows which are reduced and skewed relative to one another for low-to-high versus high-to-low input transitions, as shown in Table 7.

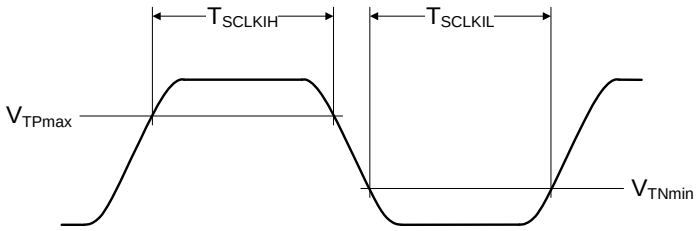

Figure 9 Received Clock Signal Constraints

119 Timings are referenced to V_{TPmax} and V_{TNmin}. Half Speed Device timing is valid for readback operation only as described in Section 6.7.3.

Table 3 Clock Input Timing Requirements

Symbol	Description	Half Speed Device		Full Speed Device		Units
		Min	Max	Min	Max	
T_{SCLKIH}	SCLK Input High Time	24		11.25		ns
T_{SCLKIL}	SCLK Input Low Time	24		11.25		ns

4.2.2 RFFE Data (SDATA)

120 The RFFE SDATA signal is bidirectional, driven by the Master or a Slave. Data shall be written on the rising edge (transition from logical level zero to logical level one) of the SCLK signal by both Master and Slaves. Each node on the bus, Master or Slave, shall read the data on the falling edge (transition from logical level one to logical level zero) of the SCLK signal.

4.2.2.1 Specifications for the SDATA Driver

121 The same signal integrity issues, maximum distance between any transmitter to any receiver device and signal transition time factors, affect the SDATA line as well as the SCLK line as described in Section 4.2. For this reason, the SDATA driver of a Full Speed device shall have transition time control to meet transition time specifications listed in Table 4.

122 The minimum slew time, $T_{SDATAOTRmin}$, is specified for a single device driving the test load described in Section 5.3. The output driver of a SDATA terminal shall drive the test loads specified in Section 5.3 with the dynamic specifications shown in Figure 10 and Table 4. The minimum transition time limit for a Slave readback is relaxed relative to the Master to permit a less complex Slave implementation.

123 The Data Timing Diagram is shown in Figure 10 where the Time for Data Output Valid from the rising clock edge T_D and the Data Output Transition Time $T_{SDATAOTR}$ are defined.

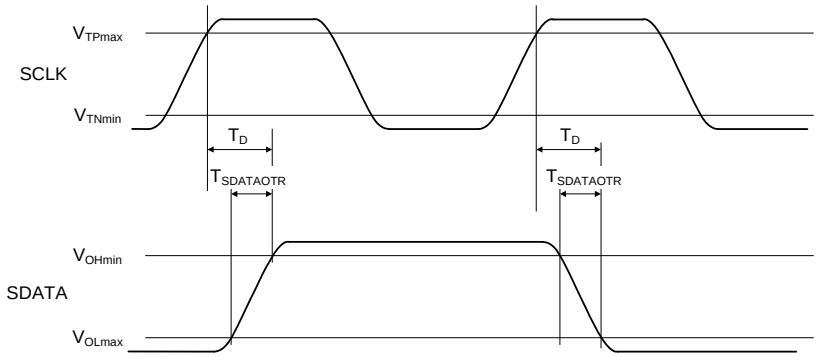

Figure 10 Bus Active Data Transmission Timing Specification

124 The data timing specification is given in Table 4. Half Speed Device timing is valid for readback operation only as described in Section 6.7.3.

Table 4 SDATA Output Timing Characteristics

Symbol	Description	Half Speed Device		Full Speed Device		Units
		Min	Max	Min	Max	
T_D	Time for Data Output Valid from SCLK rising edge	0	22	0	10.25	ns
$T_{SDATAOTR}$	SDATA Output Transition (Rise/Fall) Time (Master only)	N/A	N/A	3.5	6.5	ns
	SDATA Output Transition (Rise/Fall) Time (Slave only)	2.1	10	2.1	6.5	ns

125 Timing is referenced to V_{TPmax}, V_{OHmin} and V_{OLmax}.

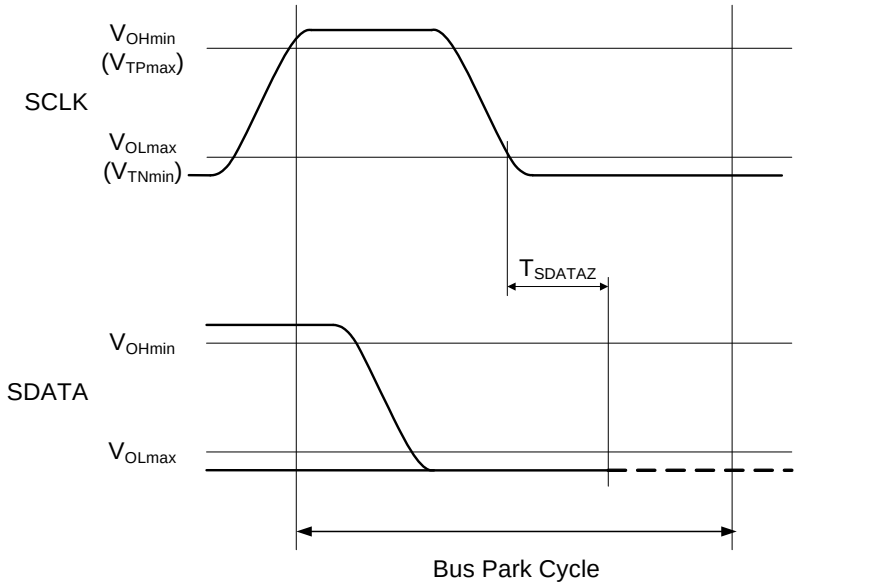

Figure 11 Bus Park Cycle Timing

126 The Bus Park Cycle as shown in Figure 11 is a special bus condition that facilitates the change of SDATA control for bus turnaround purposes. The SDATA line is driven to a logic level zero while SCLK is at a logic level one. The SDATA line is released on the falling edge of SCLK. The Bus Park Cycle is also used at the end of all Command Sequences.

Table 5 SDATA Release Timing Parameters

Symbol	Description	Half Speed Device		Full Speed Device		Units
		Min	Max	Min	Max	
T_{SDATAZ}	Data drive release time		18		10	ns

127 T_{SDATAZ} timing specification in Table 5 is referenced to V_{OLmax} and V_{TNmin} in Figure 11. Half Speed Device timing is valid for readback operation only as described in Section 6.7.3.

128 T_{SDATAZ} data signal specification is measured from the falling edge of SCLK (either from V_{OLmax} when the device is driving SCLK and SDATA lines, or from V_{TNmin} when the device is receiving SCLK and driving the SDATA lines).

4.2.2.2 Specifications for the SDATA Receiver

129 The SDATA receiver may use a glitch rejection filter on the SDATA input. The SDATA receiver shall be capable of receiving slowly changing edges without glitching. An RFFE component shall implement inputs with hysteresis on the SDATA pin as defined in Table 7. It is important to note that the input thresholds shall employ threshold windows which are reduced and skewed relative to one another for low-to-high versus high-to-low input transitions, as shown in Table 7.

130 Figure 12 defines the setup time, T_S, and hold time, T_H, of the data signal SDATA with respect to the falling edge of the clock, SCLK. V_{OHmin}, V_{OLmax}, and V_{TPmax}, V_{TNmin} are defined in Table 7 and Table 8, respectively.

131 Based on the SCLK High Time, T_{SCLKOH}, of 11.25 ns in Full Speed mode from Table 3, and the time for Data Output Valid from SCLK rising edge, T_D, of 10.25 ns from Table 4, the Data setup time, T_S, is 1 ns.

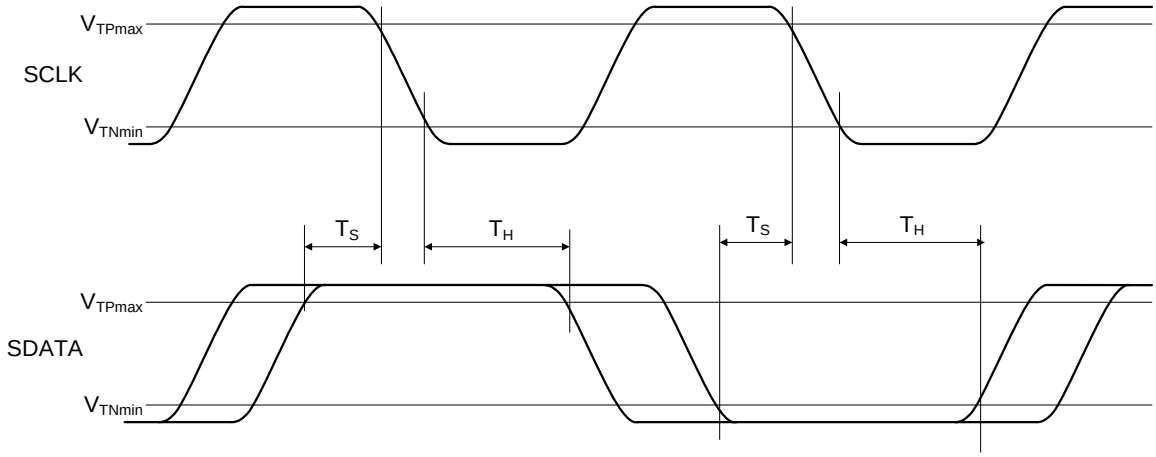

Figure 12 Bus Active Data Receiver Timing Requirements

132 The data setup and hold timing specification for Half Speed and Full Speed devices is given in Table 6. Half Speed Device timing is valid only for Slave readback operation.

Table 6 Data Setup and Hold Timing

Symbol	Description	Half Speed Device		Full Speed Device		Units
		Min	Max	Min	Max	
T_S	Data setup time	2		1		ns
T_H	Data hold time	5		5		ns

133 Timings are referenced to V_{TPmax} and V_{TNmin} defined in Table 7 and are measured at the input of the device.

4.2.3 Read/Write Access

134 All read and write data transfers are subject to meeting device specifications from Table 2, Table 4, and Table 6. For a complete description, see Section 6.7.

4.2.4 Write Access

135 All write data transfers from a Master have to meet Full Speed device specifications from Table 2, Table 4, and Table 6.

4.3 Operating States

136 Slaves shall have a minimum of three operating states, ACTIVE, SHUTDOWN and STARTUP as shown in Figure 13. An optional fourth LOW POWER state is shown in Figure 13. Separate SHUTDOWN and STARTUP states are provided to allow a synchronous RFFE Command Sequence-initiated reset using the

PWR_MODE register (see Section 6.9.1.4.1). VIO, as described in Section 5.2, has priority over any RFFE Command Sequences.

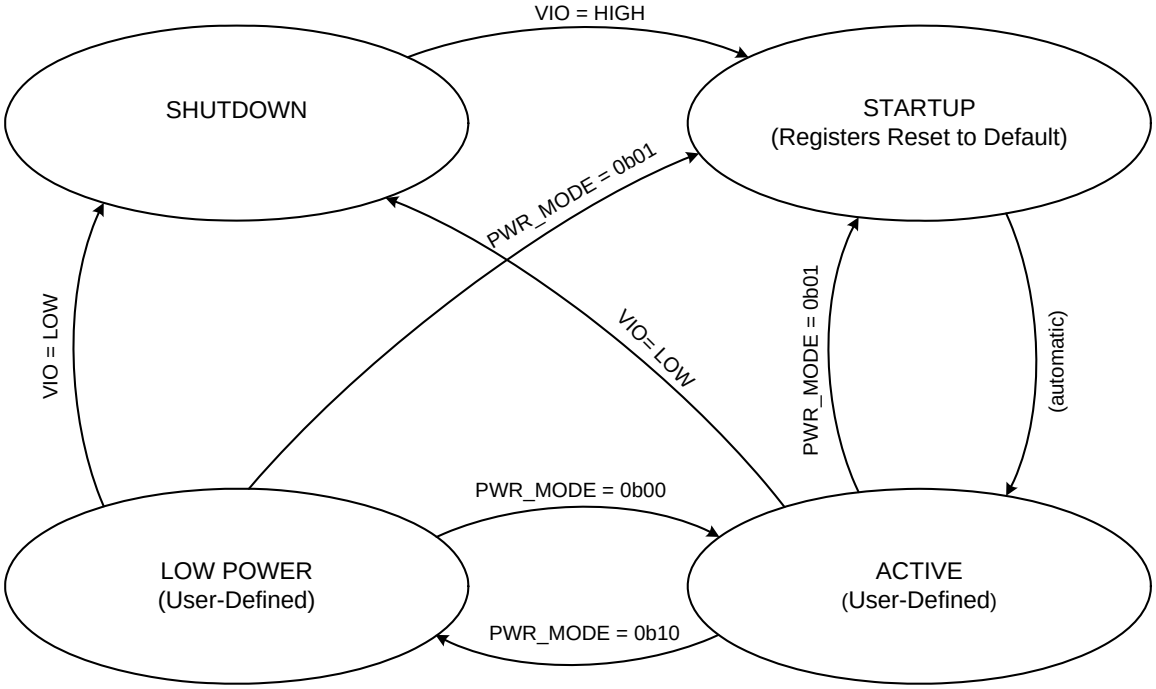

Figure 13 Slave State Diagram

4.3.1 STARTUP

137 The STARTUP state is the default condition after a reset. The STARTUP state is entered from SHUTDOWN when VIO voltage supply is applied (see Section 5.2.1) and from any other state by using the PWR_MODE register (see Section 6.9.1.4.1). The internal power-up (or power-down) process of the Slave is device-specific. The order in which functional blocks or sub-modules activate, and the time required to activate them, depend on the needs of the target application and component. From STARTUP, a Slave shall transition automatically into the ACTIVE state.

4.3.2 ACTIVE

138 The ACTIVE state is the normal operating condition of a Slave after the power-on procedure. The configuration of any functional block or sub-module on the Slave is either user-defined or specified by the device manufacturer. In the ACTIVE state, a Master may control operating values on the Slave by programming the corresponding Slave control registers.

4.3.3 LOW POWER

139 The LOW POWER state is another user-defined state and is similar to the ACTIVE state in that the configuration of any functional block or sub-module on the Slave is either user-defined or specified by the device manufacturer. The LOW POWER state is entered under software control by programming the PWR_MODE register for individual Slaves as described in Section 6.9.1.4.1.

140 During transition into the LOW POWER state, and out of LOW POWER state to ACTIVE state the Slave register information is maintained to allow for reduced initialization programming overhead.

141 In the LOW POWER state the Master may program the corresponding RFFE control registers in the Slave device. The only way to exit the LOW POWER state is to write to the PWR_MODE register or by toggling VIO.

4.3.4 SHUTDOWN

142 In the SHUTDOWN state, the Slave interface is off. (see Section 5.2.2)

143 SHUTDOWN state is entered from any other state when VIO is removed. Exit from the SHUTDOWN state to STARTUP is achieved when VIO voltage supply is applied.

4.3.5 Exceptional State Transitions

144 This document does not specify the external or environmental conditions required for RFFE operation, only the normal operation that occurs while external and environmental factors are within operating limits. Slaves may, for example, have operating limit monitoring, and this limit monitoring may affect RFFE component state, triggering state transitions such as transitioning to the SHUTDOWN state. New top-level states shall not be added into the Slave state behavior although sub-states may be added under the top level states. Examples of exceptional state transitions are shutdown triggered by excessive die temperature or exceeding a device's current protection limits.

5 Physical Layer

145 This section describes the physical layer. It is based on a subset of [MIPI03].

5.1 I/O Structures

146 The RFFE is an interface between RFFE devices that has two signals, a serial bidirectional data signal (SDATA) and a clock signal (SCLK) controlled by a Master. Figure 14 shows the I/O structures required for the data signal for both Master and Slave for a readback capable Slave. In Figure 15 the I/O cells for Master and non-readback capable Slave are shown.

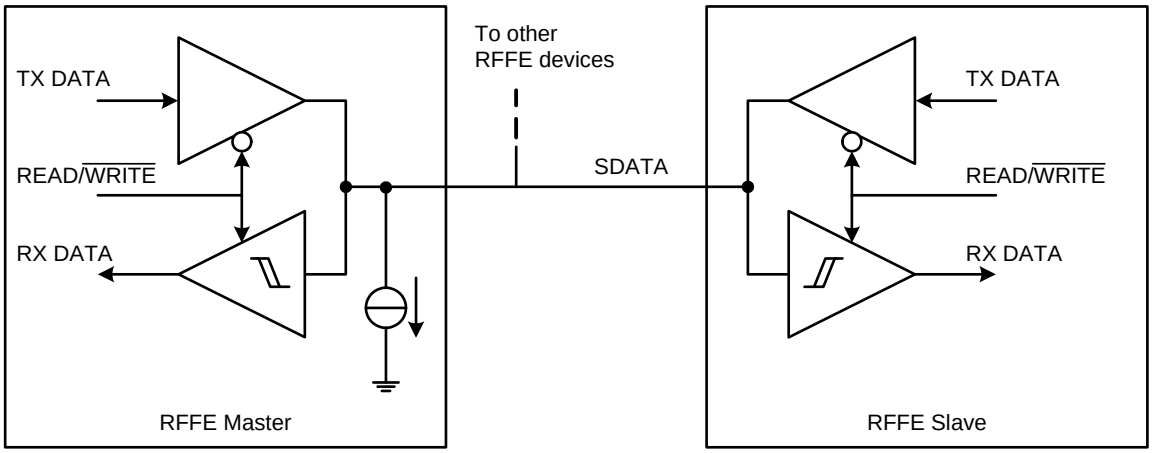

Figure 14 SDATA Master and Slave I/O Cells

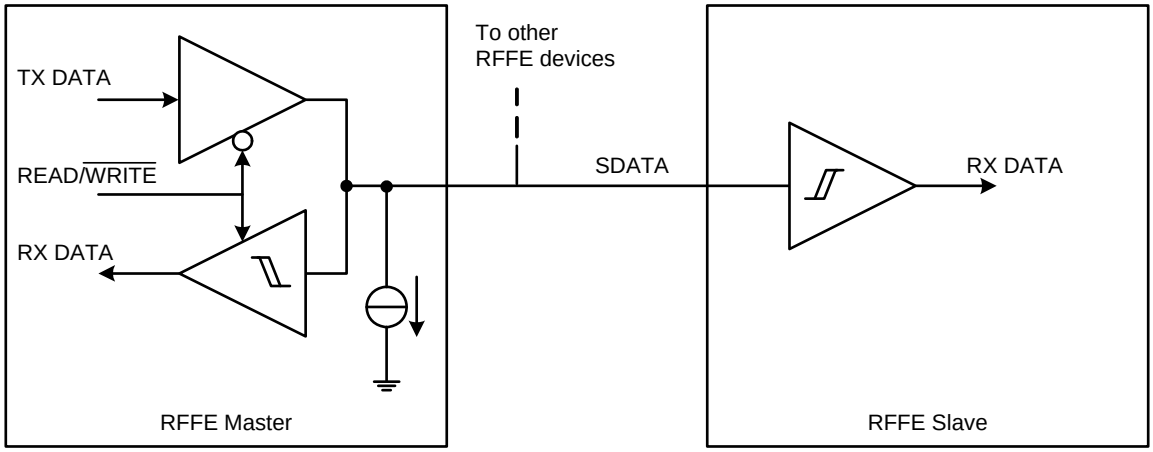

Figure 15 SDATA Master and Slave I/O Cells for an Non-readback Capable Slave

147 Figure 16 shows the I/O structures required for the clock signal for both Master and Slave.

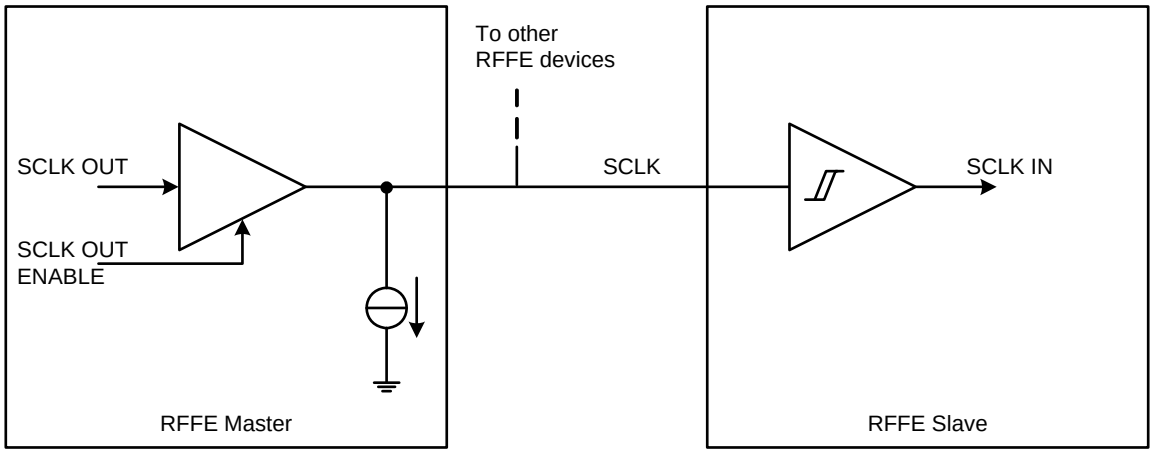

Figure 16 SCLK Master and Slave I/O Cells

148 SDATA and SCLK pull-downs may be implemented as internal or external components or current sources. Some examples are shown in Figure 14, Figure 15, and Figure 16. Internal pull-downs shall be implemented only on a Master.

149 The I/O cells shall be implemented with high impedance input structures and output drivers that are high impedance when not active. I/O cells with typical CMOS structures usually provide these characteristics.

5.1.1 Signaling Voltages

150 An RFFE component shall meet the requirements in Table 7 and Table 8 for 1.2 V or 1.8 V Bus operation.

151 A Master shall include, or have associated with it, a pull-down current (current sink) proportional to the number of Slaves supported. For example, a Master that supports sixteen devices on the bus (fifteen Slaves and one Master), each with a leakage current of -2 μA, needs a current sink of at least 32 μA to maintain a low input level, while a Master that supports five devices needs only 10 μA current sink. For the case of an external pull-down this requirement shall be met by the external pull-down device.The leakage current requirements for the SDATA pin are relaxed relative to the SCLK pin to accommodate the I/O functionality needed for readback.

152 Note:

153 *A positive value for input current in Table 8 denotes current into the pin, and a negative value denotes current out of the pin.*

Table 7 Signaling Parameters

Symbol	Description	Condition	Min	Max	Units
V_{TP}	Positive Going Threshold Voltage	1.2 V or 1.8 V Bus	0.4*VIO	0.7*VIO	V
V_{TN}	Negative Going Threshold Voltage	1.2 V or 1.8 V Bus	0.3*VIO	0.6*VIO	V
V_H	Hysteresis Voltage (V_{TP}-V_{TN})	1.2 V or 1.8 V Bus	0.1*VIO	0.4*VIO	V

Table 8 Static Electrical Characteristics for Signaling

Symbol	Description	Condition	Min	Max	Units
V_{OL}	Output Low Voltage	I_{OL}=2 mA	0	0.2*VIO	V
V_{OH}	Output High Voltage	I_{OH}=-2 mA	0.8*VIO	VIO	V
I_{IH}	Input Current High	SDATA = 0.8*VIO	0	+10, -2	µA
		SCLK = 0.8*VIO	0	+10, -1	µA
I_{IL}	Input Current Low	SDATA = 0.2*VIO	0	+1, -2	µA
		SCLK = 0.2*VIO	0	+1, -1	µA

5.1.2 I/O Configuration with Multiple Slaves

154 The RFFE specification supports up to fifteen logical Slaves on a single, or on multiple, physical Slave ICs. RFFE bus components are connected in parallel to the SCLK and SDATA, therefore the Master and the Slaves see the same loading. A line driver exists for both SCLK and SDATA in the Master, whereas only Slaves supporting readback need a line driver for SDATA. Each physical Slave has one SCLK input and one SDATA input or bidirectional interface. Any logical Slaves inside a physical Slave share the common I/O structures of the physical device (see Section 7.3.1).

155 If significantly less than the maximum fifteen physical Slaves are connected to the bus, then the capacitive load on the I/O lines may be reduced as well. When the overall capacitive load is reduced the peak current drive capability drops correspondingly.

156 Lower drive capability should be used when possible due to a significant reduction in EMI. The Master drive capability on SDATA and SCLK may be adjusted such that the total system timing specification is still met.

157 All components on a single RFFE bus instance shall share the same VIO from a common source. Figure 17 illustrates the more typical case where the VIO signal is provided by the Master. It is possible to use an external reference voltage source for VIO as shown in Figure 18.

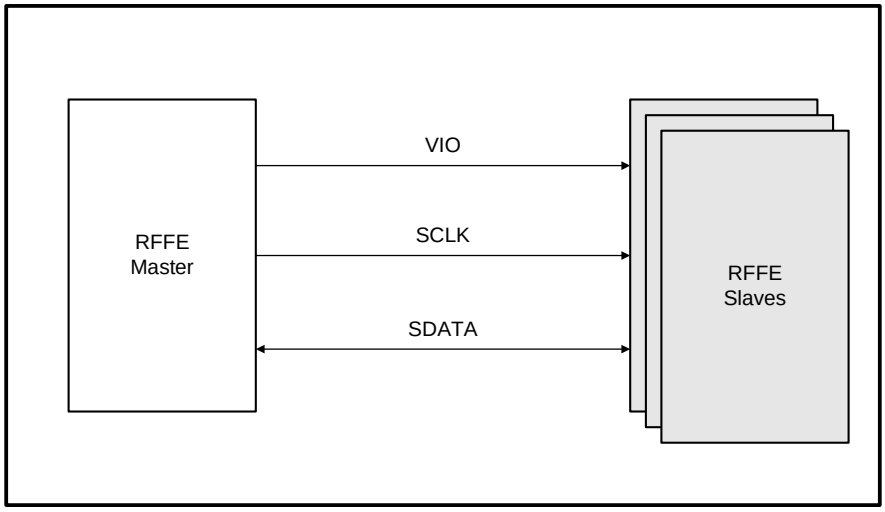

Figure 17 VIO Bus Supply

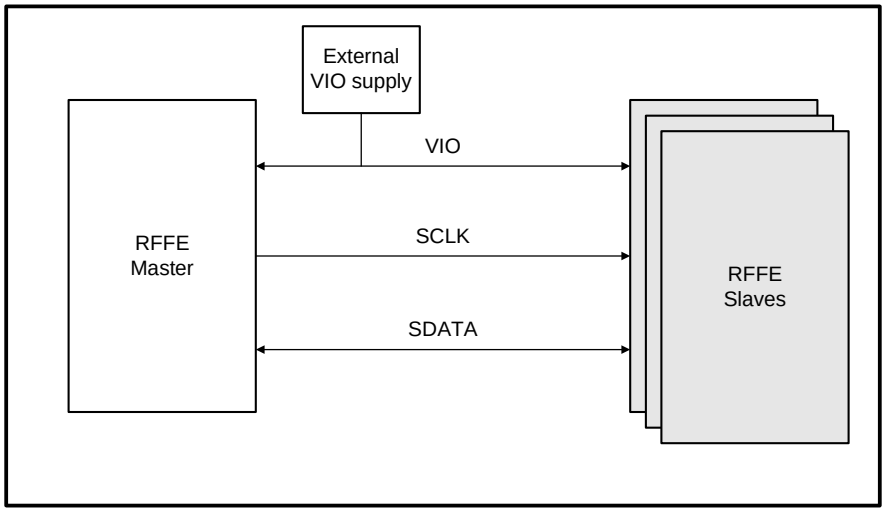

Figure 18 VIO External Bus Supply

5.2 VIO Supply

158 SCLK and SDATA are CMOS-like signals, i.e. single-ended, ground referenced, rail-to-rail, voltage mode signals. Therefore, electrical specifications in this document are given relative to the I/O supply voltage, VIO. The SCLK and SDATA terminals shall use the same signaling levels.

159 All components on a single RFFE bus instance shall share the same VIO from a common source. The VIO voltage also provides the power control for the RFFE interface blocks. Due to the wide variety of process geometries, the RFFE Slave digital interface may run at a different voltage level or off a different supply rail than other functional blocks within the Slave. The VIO voltage is separate and independent from any additional supplies or power domains in the Slave. In the case where a Slave is not active when VIO is high, it shall not interfere with the normal operation of the bus.

160 Most commonly, the Slave would be operating at a higher voltage than the VIO due to a larger process geometry. Power control schemes may also vary depending on the system implementation. Two different possible scenarios are shown in Figure 19 and Figure 20 where Vreg denotes the higher voltage rail.

161 In Figure 19, the Slave digital logic operates at VIO. It is relatively straightforward to generate a signal resetting RFFE registers by the VIO input signal. Voltage level translation may be done at an internal block interface level since the RFFE digital core is running on the VIO supply. In this case, isolation between VIO and Vreg power domains is necessary.

162 In Figure 20 the voltage level translation is done at the pad interface level since the RFFE digital core is running on the Vreg supply. If the Vreg is always on before the VIO, the register reset is easily generated within the digital block (ResetD). If Vreg is not always on before the VIO is powered a register reset may need to be provided from the Vreg functional block (ResetA).

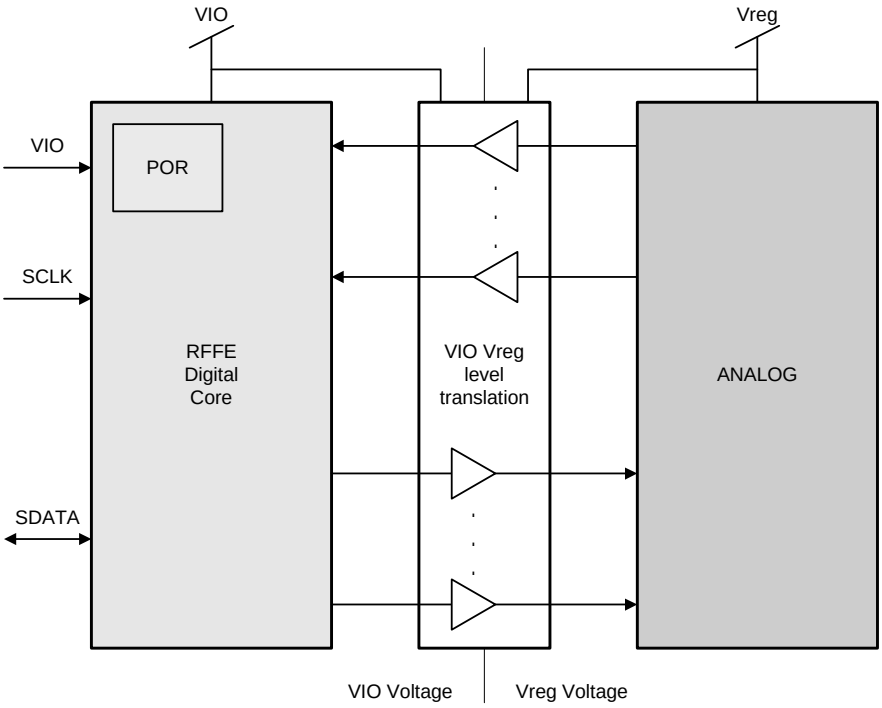

Figure 19 Slave VIO Digital

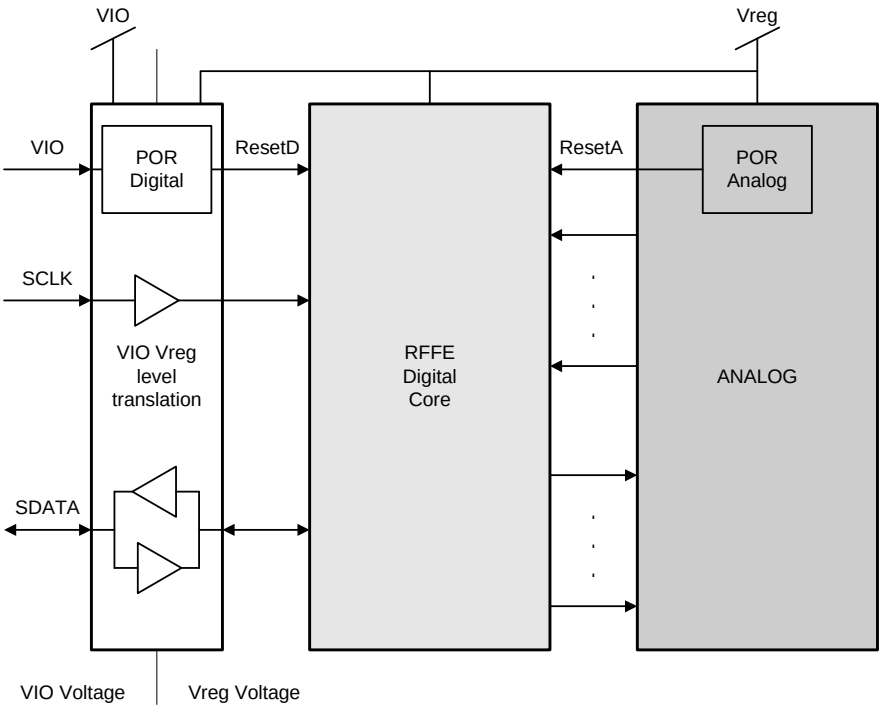

Figure 20 Slave Vreg Digital

163 VIO shall be either 1.2 V or 1.8 V and the voltage supply variation shall be less than or equal to the ± 8.33% voltage tolerance range in Table 9.

164 The device providing VIO, whether included in the bus instance or not, shall meet the VIO supply pin output current requirements in Table 9.

165 An RFFE component shall support either 1.2 V or 1.8 V operation and may support both voltage levels. If a component on an RFFE bus instance is using the common VIO voltage as source for the RFFE interface drivers it shall not exceed the maximum current loading requirements in Table 9.

Table 9 VIO Supply Pin Requirements

Symbol	Description	Condition	Min	Typ	Max	Units
VIO	VIO Voltage Level	1.8 V Bus	1.65	1.8	1.95	V
		1.2 V Bus	1.1	1.2	1.3	V
$I_{VIO-OUT}$	VIO Peak Output Current	VIO = HIGH, All devices on RFFE bus in Active Mode, C_{LOAD} = 50 pF, $T_{SCLKOTR}$ = $T_{SDATAOTR}$ = 5 ns		60		mA
I_{VIO-IN}	VIO Average Input Current	VIO = 1.8 V, Slave Device in write-only mode			1.25	mA

166 Peak power consumption (peak current draw) of the RFFE interface is dominated by two characteristics:

167 • The capacitive loading on each of the interface lines (SDATA and SCLK), and

168 • The voltage signal swing on each of these interface lines.

169 As an example, when operating from a 26 MHz timebase with a capacitive load on each interface line of 50 pF, the peak dynamic current required is 20 mA (for each line) when the transition time is restricted to 5 ns.

170 It is not economically desirable to burden the Slave with the need to support all possible RFFE bus capacitive loads for readback. A readback capable Slave might meet all the SDATA drive and edge rate requirements of the Master, but it is not required to do so. Specific system implementations might warrant designing the RFFE bus with a load capacitance of 25 pF, or less. The Slave's peak current drive requirements are lower as the maximum SDATA bus capacitance drops (Section A.2). Example criteria that might drive this decision include the maximum number of Slaves is known to be below the allowable maximum of fifteen; individual Slaves have lower input capacitances; a smaller physical layout reduces the RFFE bus routing trace capacitance.

5.2.1 Power-On (STARTUP)

171 The RFFE bus requires that VIO, a clock source (SCLK), and the common data line (SDATA) to all be available and operational before the bus may be initialized and before any devices may send Command Sequences. The RFFE bus is initialized when the Master is active. Additional to these signals other voltage supplies, clocks, and subsystems might be needed for full functionality of Master or Slaves; these are device-specific, and beyond the scope of the RFFE Specification.

172 VIO controls whether the RFFE interface is powered on, and thus active, or powered off, and disabled. Any additional discrete control signals that may be implemented by the Master or Slave shall not interfere with normal RFFE operation. (See Annex A)

173 When the VIO voltage supply is applied to the RFFE bus it shall enable the Slave interfaces and reset the user-defined Slave registers to the device manufacturer's default settings which may be fixed, adjustable, or programmable, depending on the Slave device capability.

174 SCLK and SDATA shall be at logic low levels when VIO is applied and shall remain low until the RFFE interface is active. The SCLK and SDATA logic levels shall stay low for a minimum signal reset delay time after VIO is applied as specified in Table 10, and shown in Figure 21.

175 At power-on, the reserved power mode and trigger register (PM_TRIG[7:0] at address 0x001C) shall be set to 0x00, effectively resetting PWR_MODE to normal operation (see Section 6.9.1.4.1) and clearing the trigger mask bits TRIG_REG (see Section 6.9.1.4.2).

176 The optional programmable USID register value (USID[3:0]) shall be set to the default value, which may be fixed, adjustable, or programmed, upon power-on (see Section 6.8.2).

5.2.2 Power-Off (SHUTDOWN)

177 A logic low level applied to VIO shall power-off the device interfaces on that particular RFFE bus instance. When VIO is low, both Master and Slave RFFE interfaces shall not consume any significant power from VIO, SCLK or SDATA. Also, any internal pull-ups on inputs shall be disabled. When the VIO supply is low, the RFFE VIO, SCLK and SDATA leakage current shall meet the signal OFF current specification in Table 10. The Master shall maintain the SDATA and SCLK signals at a logic low level when VIO is not applied.

Table 10 VIO Supply Reset Requirements

Symbol	Description	Condition	Min	Typ	Max	Units
$V_{VIO-RST}$	VIO Supply Reset Voltage Level	VIO Low			0.2	V
$T_{VIO-RST}$	VIO Supply Reset Timing	VIO Low	10			µs
T_{VIO-R}	VIO Supply Rise Time				400	µs
T_{SIGOL}	Input Signal Reset Delay Time	VIO >= VIO(min), SCLK, SDATA=Low	120			ns
$I_{SIG-OFF}$	Interface OFF State Leakage Current	VIO <= $V_{VIO-RST}$ Applies to VIO, SCLK, SDATA Inputs	-1		1	µA

178 If the Slave is off, the RFFE Slave interface is also off. If the Slave is active, the RFFE Slave interface is also likely active. However, an RFFE Slave does not have the concept of connecting or disconnecting from the RFFE bus. The Slave is simply active or not, depending on whether it responds to and receives Command Sequences on the bus it is connected to.

179 A Slave may turn off the RFFE interface when bus access is not needed by that device, even if the device is otherwise active and functioning. A Slave may autonomously turn off its RFFE interface as desired, provided that it does not interfere with other devices on the bus.

180 A Master or Slave may monitor RFFE bus signals even when it is not connected to the bus. For example, in the case of an external VIO reference source, the Master may monitor the VIO level to know when the Slaves will be asynchronously reset.

5.2.3 Reset

181 The VIO input to a Slave functions as an active low reset. When VIO is asserted or re-asserted, all of the Slaves attached to that RFFE bus shall enter the STARTUP state and shall be asynchronously reset.

182 When the VIO voltage supply is reapplied to the RFFE bus, it shall re-enable the Slave interface and set the user-defined Slave registers to the device manufacturer's default settings, whether these are fixed, adjustable, or programmable default values. After a reset, reapplying VIO initiates Slave transitions from SHUTDOWN to the STARTUP state and then automatically to the ACTIVE state. (see Section 4.3.1 and Section 5.2.1).

183 VIO is the only required reset. However, Slave registers may also be synchronously set to their default values, whether these are fixed, adjustable, or programmable default values, by register programming PWR_MODE to the Default Settings mode (see Section 6.9.1.4.1). Additionally, an optional Reset signal may be used (see Section A.1.1), or other device-specific reset functions, but these shall not interfere with the reset methods defined in the RFFE Specification.

184 A diagram depicting the requirements fromTable 10 for a VIO-initiated reset is shown in Figure 21.

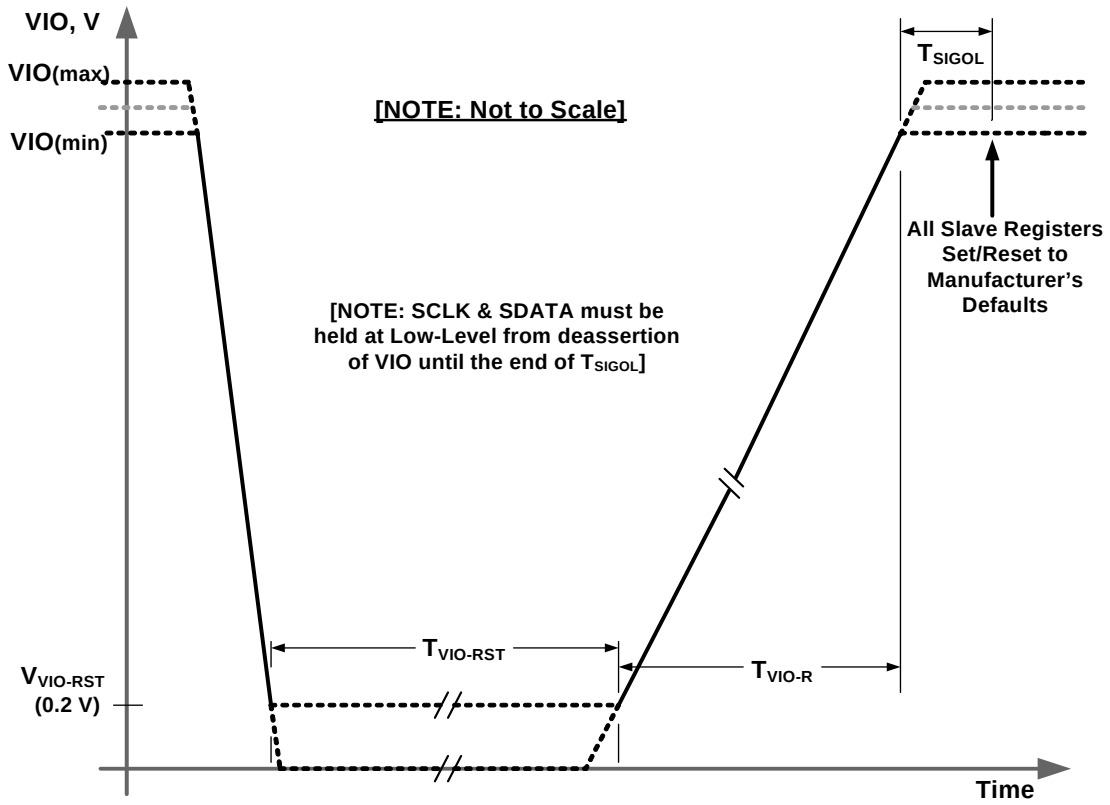

Figure 21 Requirements for VIO-Initiated Reset

185 As shown in Figure 21 and Table 10, certain requirements must be met by an assertion (or re-assertion) of VIO, following any VIO deassertion. For VIO deassertion the voltage level on VIO must attain a voltage of 0.2 V or less, and this must be maintained for a minimum of $T_{VIO-RST}$ during the dessertion of VIO.

186 Upon re-assertion the VIO voltage source shall be capable of returning VIO to the appropriate VIO(min) level within T_{VIO-R}. After VIO attains a voltage level of VIO(min), or greater, a minimum reset time of T_{SIGOL} must be allotted before any RFFE bus activity is initiated on the SCLK and SDATA lines. This time is

intended to allow for the asynchronous set/reset of all Slave registers to the manufacturer's default settings, whether the default settings are fixed, adjustable, or programmable. Note that the value given for T_{SIGOL} is a minimum requirement. Due to the fact that various implementation scenarios are possible, the time required for a specific device to be able to resume normal operations after re-assertion of VIO is implementation-specific, and shall be stated in each manufacturer's device data sheet if additional requirements are necessary.

5.3 Device Characterization

187 The device electrical characteristics for SCLK and SDATA shall be guaranteed using the device characterization circuit shown in Figure 22. The characterization trace impedance in the figure is only for characterization purpose, and does not reflect real trace impedance in an application that may be different.

188 The slew times ($T_{SCLKOTRmin}$ and $T_{SCLKOTRmax}$ for the SCLK line, $T_{SDATAOTRmin}$ and $T_{SDATAOTRmax}$ for the SDATA line), transient voltage, and transition time specifications of a Master and Slave shall be characterized across device manufacturer's specified load range through a transmission line of 75 Ω impedance and 15 cm physical length.

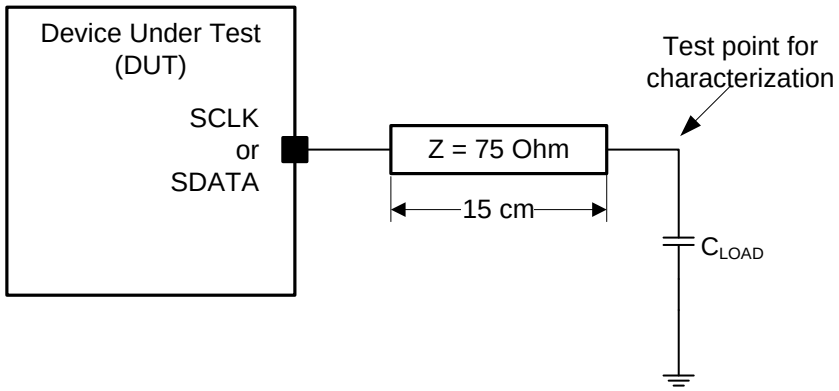

Figure 22 Device Characterization Circuit

5.4 EMI

189 EMI artifacts from the RFFE bus could potentially interfere with very low noise RF circuits. In particular, harmonics generated by the RFFE bus should be less than -180 dBm/Hz in the 3GPP bands (6 dB below the noise floor) to avoid degrading receiver LNA sensitivity.

190 The noise coupling from RFFE interface pins onto sensitive RF pins is highly dependant on package isolation and pin location. RFFE SCLK and SDATA signal pins should be located on the opposite side of the package from noise sensitive RF pins when possible.For this analysis, a minimum package isolation of 40 dB at 700 MHz was assumed.

191 Investigation by the RFFE Working Group shows the worst case for RFFE generated EMI interference occurs in the 700 MHz frequency bands when operating at VIO = 1.8 V.

192 The conducted emissions from each RFFE signal (SCLK and SDATA) shall be less than -90 dBm (voltage equivalent) measured in a 100 kHz RBW at frequencies greater than or equal to 700 MHz measured in the characterization circuit shown in Figure 22. A high impedance RF probe is required to perform the necessary impedance transformation to 50 Ω for this power measurement.

193 Typically, the roll-off of EMI at higher frequencies is greater than the reduction in package isolation. The RFFE interface lines may have additional low-pass filtering with cutoff frequency exceeding 100 MHz to

provide extra EMI filtering above 700 MHz. Finite source resistance on the line driver results in such a Lappish filter, though the cutoff frequency depends on the actual capacitance loading on each line.

5.4.1 EMI for Readback

194 The RFFE Working Group expects the occurrence of readback events to be low. A readback event is anticipated to be of short duration, therefore, EMI experienced from a readback event is expected to be tolerable and the minimum transition time limit for a Slave readback is relaxed as shown in Table 4.

6 Protocol Layer

195 This section describes the Protocol Layer.

6.1 Bit Ordering

196 Bits shall be sent on the bus MSB first. In all figures, bits and Frames are shown such that both individual bits and Frames are represented, in left-to-right, top-to-bottom reading order, as they would transfer across the interface.

6.2 Command Sequences

197 Command Sequences shall be comprised of the following three events that occur in order:

198 • Transmission of the Sequence Start Condition (SSC)

199 • Transmission of Frames (Command Frame and possibly one or more Data Frames)

200 • Transmission of a Bus Park Cycle

201 Three events, SSC, Frames and Bus Park Cycle, form the Command Sequence.

6.2.1 Sequence Start Condition

202 The Sequence Start Condition shall be a unique condition on the bus identified by a rising edge followed by a falling edge on SDATA while SCLK remains at a logic low level. The SSC is used by the Master to identify the start of a Command Sequence.

203 The Master shall generate the SSC by driving SDATA to logic level one for one SCLKint period, then to logic level zero for one SCLKint period while holding SCLK at logic level zero as shown in Figure 23.

204 A Command Frame shall start on the next SCLKint rising edge. A Slave never responds to anything unless the Command Sequence is correct and it is decoded as a Command Sequence for the Slave.

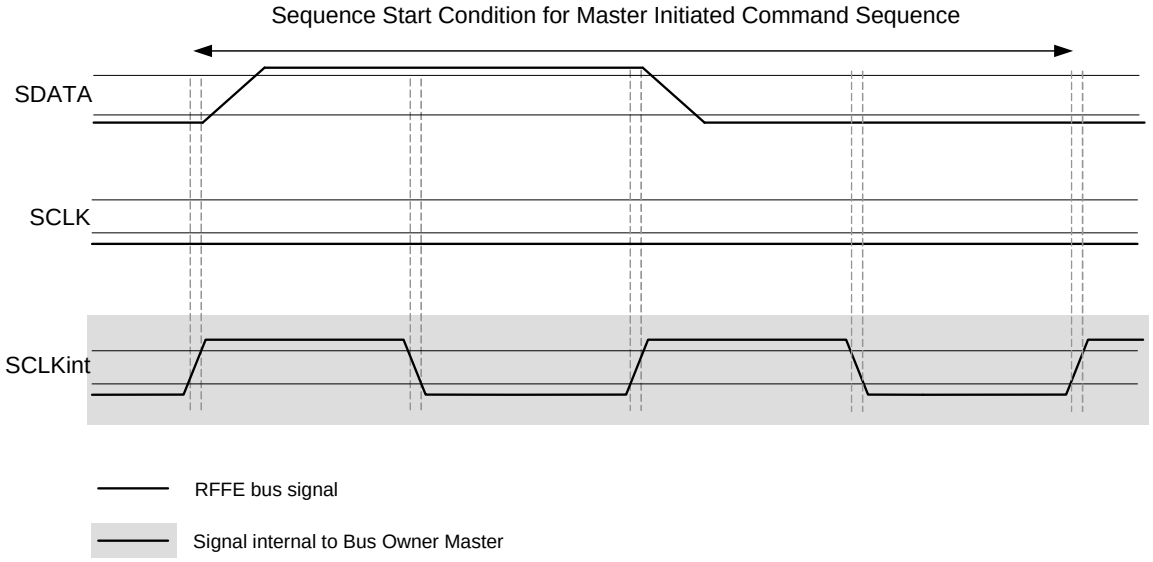

Figure 23 Sequence Start Condition

6.2.2 Frames

205 There are three basic types of Frames:

206 • Command Frame, thirteen bits. See Section 6.2.2.1.

207 • Data or Address Frames, nine bits. See Section 6.2.2.2.

208 • No Response Frame, nine bits. See Section 6.2.2.3.

209 All Frames consist of data, address or command bits and a parity bit.

6.2.2.1 Command Frame

210 A Command Frame shall consist of a 4-bit Slave address field, an 8-bit command payload field, and a single parity bit.

211 The first four bits of the address field are the SA Slave address bits followed by the eight command bits followed by the parity bit. The Command Frame structure is shown in Figure 24.

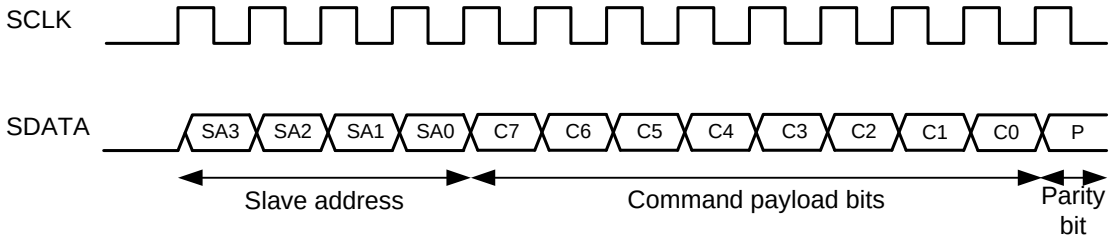

Figure 24 Command Frame

6.2.2.2 Data or Address Frame

212 A Data or Address Frame shall consist of eight data bits or eight address bits, respectively, and a single parity bit. The Frame structure is shown in Figure 25. The Frame is called an Address Frame when the payload bits carry address information and a Data Frame when the payload bits carry data. The type of data being carried is defined by the position of the Frame within the Command Sequence. See Section 6.7.2 for more information.

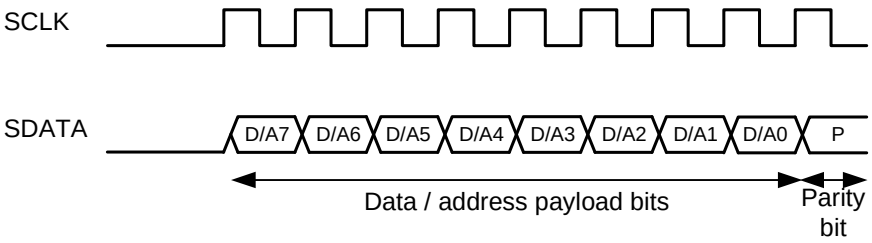

Figure 25 Data or Address Frame

6.2.2.3 No Response Frame

213 All bits, including the parity bit, of a No Response Frame shall be zero. This Frame may be generated by driving SDATA to logic level zero or passively allowing the Master to pull SDATA low for the duration of the Frame. A No Response Frame is nine bits long since it is a Data Frame. A No Response Frame is used as the response of a write-only Slave during a Read Command Sequence, or by a Slave during a read of an unused register location.

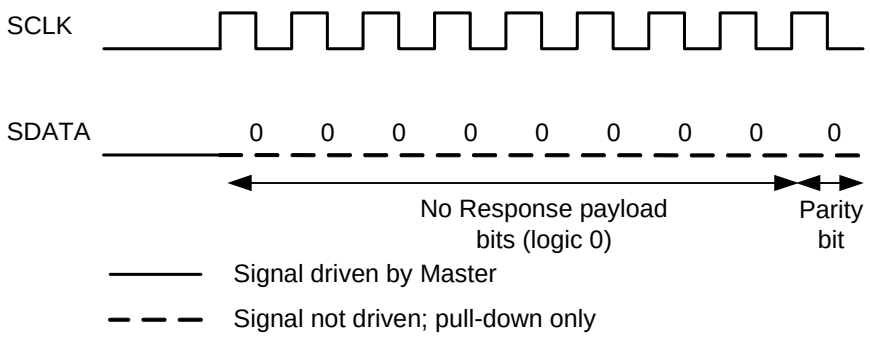

Figure 26 No Response Frame

6.2.3 Parity Bit

214 A Frame shall end with a single parity bit. The parity bit shall be driven such that the total number of bits in the Frame that are driven to logic level one, including the parity bit, is odd. For example, a Data Frame with data 0x63 (0b01100011) has a “1” parity bit and a Data Frame with data 0x4C (0b01001100) has a “0” parity bit.

6.2.3.1 Error Detection and Handling

215 In the event of error detection characterized by the cases in Table 11, each and every condition shall be handled according to the corresponding procedure. A Slave may record errors over the bus.

216 A Master shall always complete all Read Command Sequences. If a Slave ever finds a parity error it shall not take control of the SDATA line.

Table 11 Error Handling

Error Detected	Error Handling
Undefined Command Frame received	Slave ignores entire Command Sequence
Command Frame with parity error received	Slave ignores entire Command Sequence
Incompatible command length (command interrupted by SSC)	Slave executes Command Sequence up to the last correct and complete frame (multi-byte read and write operations)
Address Frame with parity error received	Slave ignores entire Command Sequence
Data Frame with parity error received	Slave ignores only the erroneous Data Frame of the Command Sequence
Read of unused register	Slave sends No Response Frame
Write of an unused register	Slave discards data being written, rest of Command Sequence proceeds as normal
Read using the Broadcast ID or a GSID	Slave ignores the Command Sequence

217 A Slave interprets the first thirteen bits after SSC as the Command Frame. If the Slave does not receive thirteen SCLK pulses it waits for the missing clocks to appear, unless another SSC occurs. If there are extra pulses after the Command Frame, the pulses are interpreted as part of the subsequent Address and Data

Frames. Any additional SCLK pulses after an otherwise correctly formatted, complete Command Sequence are ignored because the Slave is expecting an SSC, and it will synchronize itself to the next SSC when it appears. It is unlikely a Slave will be able to distinguish an incompatible Command Sequence length, but it shall be able to identify any unknown Command Sequences, parity bit errors and the occurrence of an SSC embedded in a Command Frame.

218 For the case of Extended Read and Extended Write Command Sequences, as described in Section 6.7.2, any Command Sequence shall be considered valid until the last completed byte in a multi-byte Command Sequence, prior to the receipt of an SSC. This does not prohibit a Slave from defining a set of registers that are updated only after an entire Extended Command Sequence has completed. However in general, it is assumed that a Slave updates each byte immediately after the byte is successfully received in a Data Frame.

6.2.4 Bus Park Cycle

219 A device with ownership of SDATA shall initiate a Bus Park Cycle on SDATA (see Figure 11 in the electrical specifications section) at the end of a Command Sequence or when the device transfers control of SDATA to another device. The purpose of the Bus Park Cycle is to put the RFFE bus into a known state in preparation for an SDATA signal control change, the start of a Command Sequence, or the start of the Bus Idle condition.

220 During the Bus Park Cycle, the device releasing SDATA shall drive the SDATA signal to a logic level zero during the first half of the SCLK clock cycle. Thereafter, the device releasing SDATA shall put its SDATA driver into the high impedance state. The Master controlling the Command Sequence shall provide a normal SCLK during the Bus Park Cycle.

221 A minimum of 10 ns shall be reserved between the Bus Park Cycle and SSC separating two subsequent Command Sequences.

6.3 Bus Idle Condition

222 The RFFE bus is in the “Idle” condition when both SCLK and SDATA signals are at logic level zero and all devices connected to the bus other than the Master have their bus driver outputs in a high impedance state. The Master is the only device on the bus that drives the SCLK line, maintaining a strong (low-impedance) low level on that line. A pull-down source on SDATA maintains a weak (high impedance) low level on that line. The bus is always in idle condition between the end of a Command Sequence and the beginning of a new Command Sequence. See Figure 27 for an illustration of the Bus Idle condition.

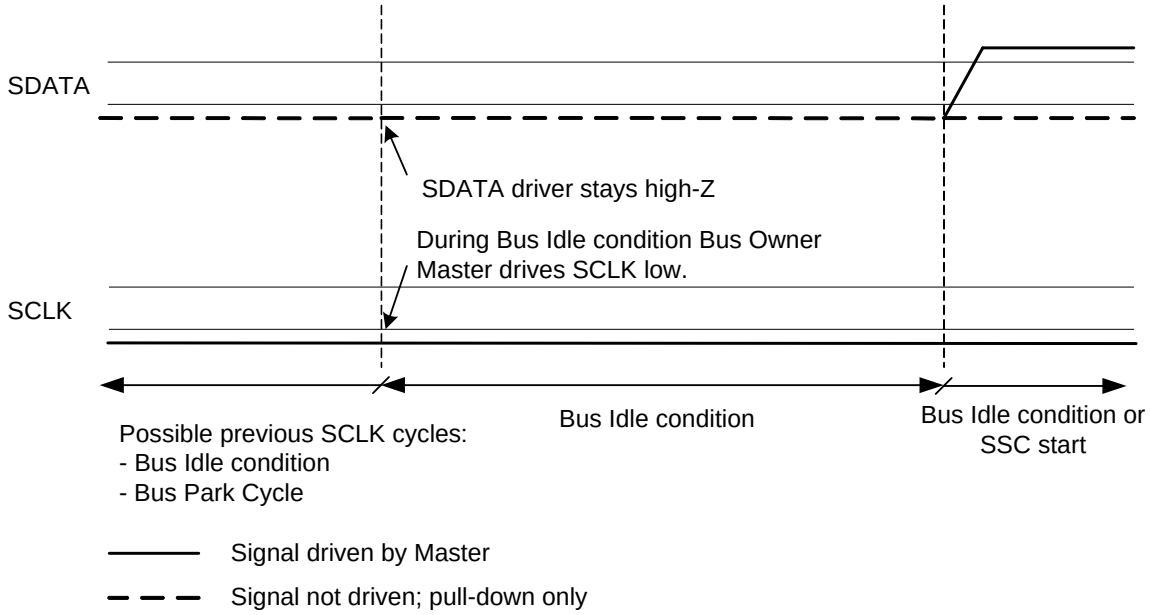

Figure 27 Bus Idle Condition

6.4 RFFE Command Sequences

223 RFFE supports a number of Read and Write Command Sequences. Single byte and multi-byte Command Sequences are supported by a Master and are optional for a Slave. RFFE supports the following Command Sequences:

224 • Register Write
225 • Register 0 Write
226 • Extended Register Write of up to four data bytes mandatory and up to sixteen data bytes optional
227 • Register Read
228 • Extended Register Read of up to four data bytes mandatory and up to sixteen data bytes optional
229 • Extended Register Write Long (Optional)
230 • Extended Register Read Long (Optional)

231 An RFFE Master shall support the RFFE mandatory Command Sequences, Register Write, Register 0 Write, Extended Register Write (from one to four bytes), Register Read and Extended Register Read (from one to four bytes). If the Extended Register Write Long and Extended Register Read Long Command Sequences are not supported by the Master, the address space for the registers in a Slave is limited to 8-bits, which supports 256 registers. A Slave may support certain Command Sequences for certain registers. For example, some registers might be read-only or write-only, or an application-specific function might be defined that uses an Extended Register Write Command Sequence only on certain registers. By doing this, the Slave gate count is kept to a minimum. Table 12 lists all the Master and Slave supported Command Sequences.

Table 12 RFFE Supported Command Sequences

Mandatory Commands		Information		SSC		Command Frame											Data Frame								
M	S	Hex	Description																					
Y	O	00 - 0F	Extended Register Write			SA[3:0]	0	0	0	0	0 0 0 0 BC[3:0] 1 1 1 1				P	Address[7:0]	P	Up to 16 Bytes of Data with Parity Master must support up to 4 bytes					BP
		10	(Reserved)				0	0	0	1	0	0	0	0	P	Undefined								
		11	(Reserved)				0	0	0	1	0	0	0	1	P	Undefined								
		12	(Reserved)				0	0	0	1	0	0	1	0	P	Undefined								
		13	(Reserved)				0	0	0	1	0	0	1	1	P	Undefined								
		14	(Reserved)				0	0	0	1	0	1	0	0	P	Undefined								
		15	(Reserved)				0	0	0	1	0	1	0	1	P	Undefined								
		16	(Reserved)				0	0	0	1	0	1	1	0	P	Undefined								
		17	(Reserved)				0	0	0	1	0	1	1	1	P	Undefined								
		18	(Reserved)				0	0	0	1	1	0	0	0	P	Undefined								
		19	(Reserved)				0	0	0	1	1	0	0	1	P	Undefined								
		1A	(Reserved)				0	0	0	1	1	0	1	0	P	Undefined								
		1B	(Reserved)				0	0	0	1	1	0	1	1	P	Undefined								
		1C	(Reserved)				0	0	0	1	1	1	0	0	P	Undefined								
		1D	(Reserved)				0	0	0	1	1	1	0	1	P	Undefined								
		1E	(Reserved)				0	0	0	1	1	1	1	0	P	Undefined								
		1F	(Reserved)	1	0		0	0	0	1	1	1	1	1	P	Undefined								
Y	O	20 - 2F	Extended Register Read			SA[3:0]	0	0	1	0	0 0 0 0 BC[3:0] 1 1 1 1				P	Address[7:0]	P	BP	Up to 16 Bytes of Data with Parity Master must support up to 4 bytes					BP
O	O	30 - 37	Extended Register Write Long			SA[3:0]	0	0	1	1	0	0 0 0 BC[2:0] 1 1 1			P	Address[15:8]	P	Address[7:0]			P	Up to 8 Bytes of Data with Parity		BP
O	O	38 - 3F	Extended Register Read Long			SA[3:0]	0	0	1	1	1	0 0 0 BC[2:0] 1 1 1			P	Address[15:8]	P	Address[7:0]			P	BP	Up to 8 Bytes of Data with Parity		BP
Y	O	40 - 5F	Register Write			SA[3:0]	0	1	0	0 0 0 0 0 Address[4:0] 1 1 1 1 1					P	Data[7:0]	P	BP						
Y	O	60 - 7F	Register Read			SA[3:0]	0	1	1	0 0 0 0 0 Address[4:0] 1 1 1 1 1					P	BP	Data[7:0]		P	BP					
Y	O	80 - FF	Register 0 Write			SA[3:0]	1	0 0 0 0 0 0 0 Data[6:0] 1 1 1 1 1 1 1							P	BP								

Notes
All non-implemented Command Sequences and register accesses are ignored, resulting in a null response.

Nomenclature
BC = Byte Count M = Master P = Parity Y = Mandatory
BP = Bus Park O = Optional SA = Slave Address S = Slave SSC = Sequence Start Condition

6.5 Broadcast Messages

232 The RFFE protocol supports Broadcast Messages over the RFFE interface. A Broadcast Message is simply a Command Sequence that is sent using a GSID for the USID portion of the Command Frame. A Broadcast Message is used to write a Command Sequence to all Slaves on the bus. The SID 0b0000 is reserved for Broadcast Messages. Other SIDs may be defined as GSIDs in order to create different groups. The power and trigger modes are broadcast-capable registers. Further, user-defined registers may also be implemented that utilize broadcast capabilities. Only a Write Command Sequence shall be sent using a GSID or the Broadcast ID. A Read Command Sequence using a GSID or the Broadcast ID shall be ignored by a Slave.

233 Other triggering mechanisms such as defining a write to a particular register address in order to configure all devices with a common GSID to a predefined mode may be user-defined. This is similar to a trigger operation without the shadow registers, thereby reducing the area overhead for register triggers.

6.6 RFFE Register Space

234 The RFFE specification defines a single register space that simplifies the use of registers in a Slave. While the RFFE register space is a merged register space of SPMI, it is also backwards-compatible with SPMI register spaces and SPMI register access Command Sequences. RFFE registers may be accessed by both single-byte and multi-byte register Command Sequences.

235 In Section B.4, the SPMI register space is discussed in order to illustrate the differences with the RFFE register space.

236 RFFE has a single register space that contains registers 0 to 65535. RFFE registers may be accessed using the Register 0 Write, Register Write, Register Read, Extended Register Write, Extended Register Read, Extended Register Write Long and Extended Register Read Long Command Sequences. The Register 0 Write Command Sequence may be used for accessing Register 0, while shorter Register Write and Register Read Command Sequences may be used for accessing registers 0 to 31. Extended Command Sequences may be used for accessing registers 0 to 255, while Extended Long Command Sequences may be used for accessing registers 0 to 65535.

237 By keeping RFFE registers in a single register space and backward-compatible with SPMI register spaces, the lower thirty-two registers can take advantage of both shorter single-byte and Extended multi-byte Command Sequences.

238 Figure 28 shows registers in the RFFE register space and Command Sequences that may be used for accessing these registers.

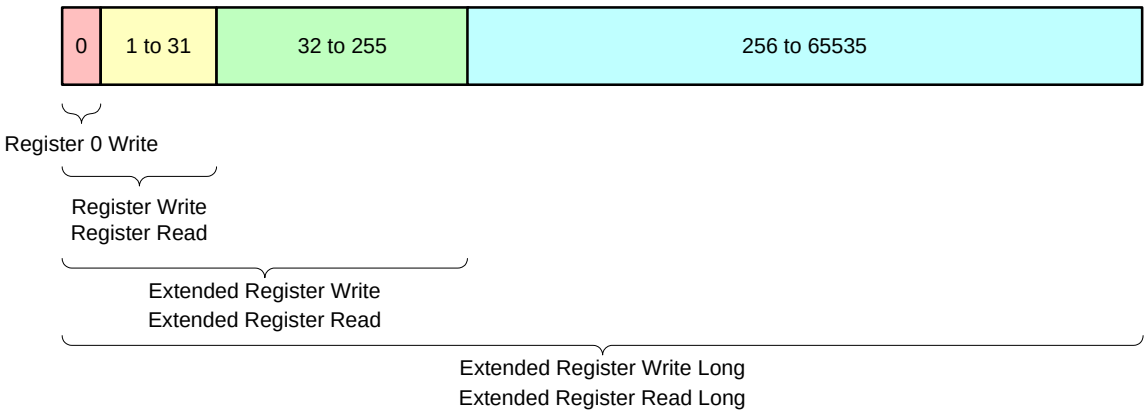

Figure 28 Slave Register Space, RFFE

6.7 Command Sequences

6.7.1 Command Sequence Summary

239 This section describes all Command Sequences defined in the RFFE protocol.

240 See Section 6.2 for more information about Command Sequences and other RFFE constructs. The coding of the Command Frame is shown in Table 12.

6.7.2 Command Sequence Descriptions

241 Each Command Sequence is described in detail in the following sections.

6.7.2.1 Extended Register Write Command Sequence

242 The Extended Register Write provides write access to the extended register space on a Slave. One to sixteen bytes of data shall be written in a single Command Sequence. The extended register address shall be supplied in a separate Address Frame in the Command Sequence.

243 Figure 29 shows the Extended Register Write Command Sequence. The Command Sequence starts with the SSC followed by the Extended Register Write Command Frame, an Address Frame and one or more Data Frames with the data to be written. Note that Data Frames immediately follow the Command Frame in a continuous Command Sequence. The Command Sequence ends with a Bus Park Cycle.

244 The four LSBs of the Extended Register Write Command Frame, BC[3:0] in Figure 29, indicate the number of bytes to be written by the Command Sequence. BC3 is the byte count MSB. 0b0000 indicates one byte shall be written and 0b1111 indicates sixteen bytes shall be written.

245 The register address in the Command Sequence contains the address of the first extended register to be written. If more than one byte is written in a single Command Sequence then the Slave's local extended register address shall be automatically incremented by one for each byte written up to address 0xFF, starting from the address indicated in the Address Frame. If the extended register address reaches 0xFF before the last Data Frame in the Command Sequence then the content of the register at address 0xFF is overwritten with the overflow data.

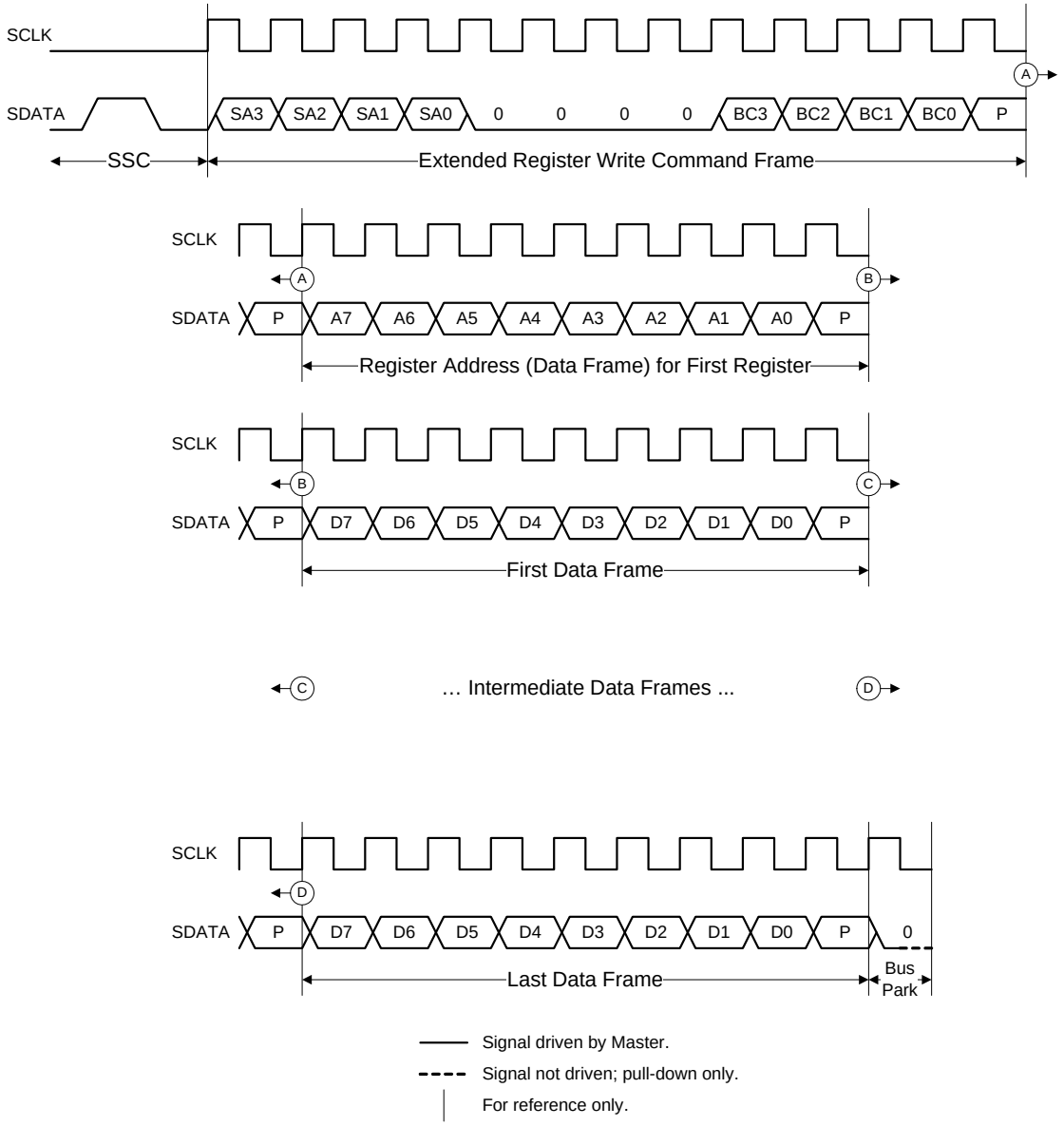

Figure 29 Extended Register Write Command Sequence

6.7.2.2 Extended Register Read Command Sequence

246 The Extended Register Read Command Sequence provides read access to the extended register space on a Slave. One to sixteen bytes of data shall be read in a single Command Sequence. The extended register address shall be supplied in a separate Address Frame in the Command Sequence.

247 Figure 30 shows the Extended Register Read Command Sequence. The Command Sequence starts with an SSC followed by the Extended Register Read Command Frame, an Address Frame and one or more Data Frames with the data read from the Slave. A Bus Park Cycle occurs between the Address Frame and the Data Frames. The Command Sequence ends with a Bus Park Cycle.

248 The four LSBs of the Extended Register Read Command Frame, BC[3:0] in Figure 30, indicate the number of bytes to be read in the Command Sequence. BC3 is the byte count MSB. 0b0000 indicates one byte shall be read and 0b1111 indicates sixteen bytes shall be read.

249 The register address in the Command Sequence contains the address of the first extended register to be read. If more than one byte is read in a single Command Sequence then the Slave's local extended register address shall be automatically incremented by one for each byte read up to address 0xFF, starting from the address indicated in the Address Frame. If the extended register address reaches 0xFF before the last Data Frame in the Command Sequence, then the content of the register at address 0xFF is read multiple times. An Extended Register Read Command Sequence by the Master to an unsupported Slave extended register address results in a No Response Frame from the Slave. If the address that results from auto-incrementing the Slave's local extended register address is for an unsupported register then the Slave sends a No Response Frame to the Master in place of the Data Frame. The Command Sequence continues from the next extended register address.

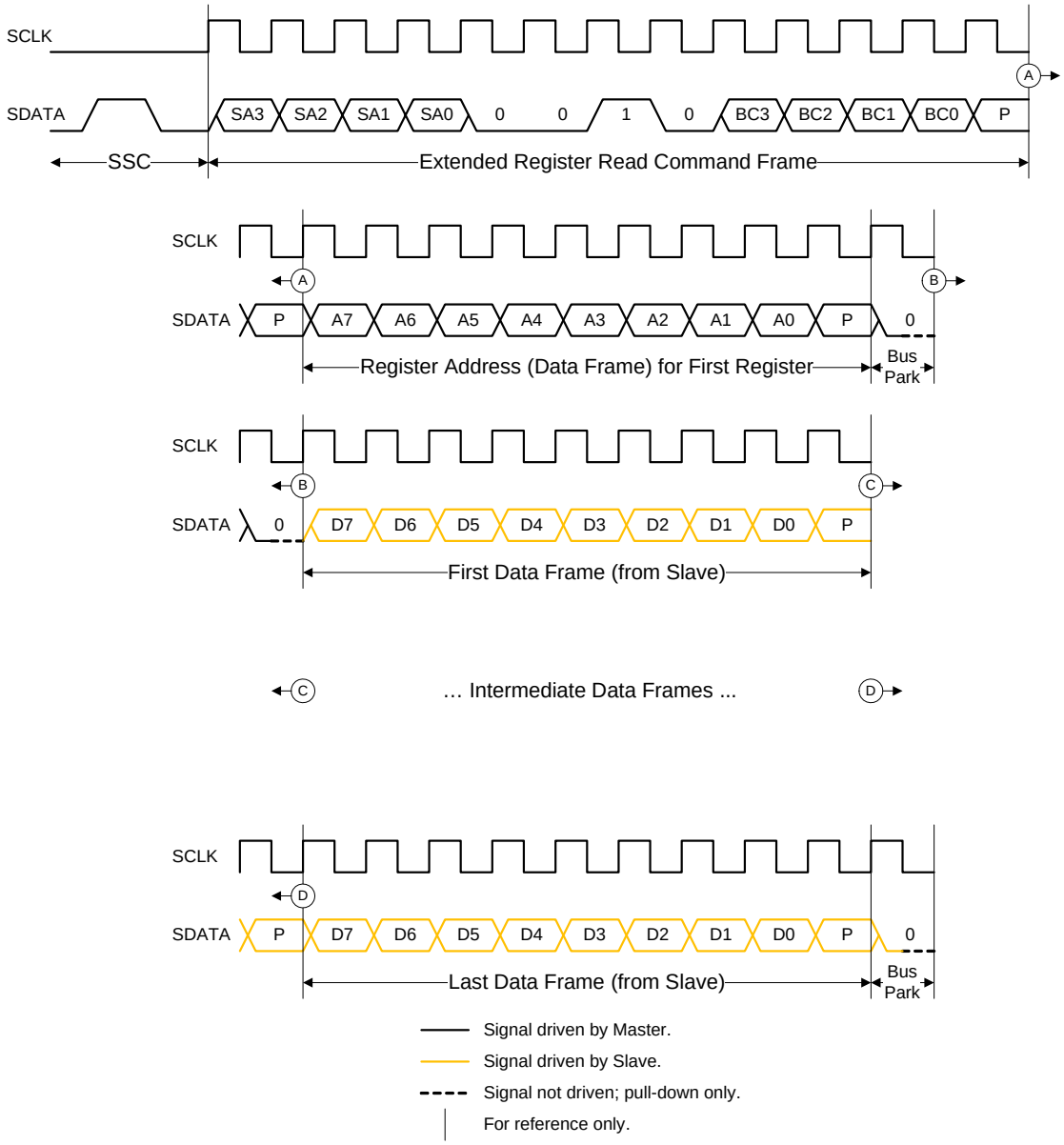

Figure 30 Extended Register Read Command Sequence

6.7.2.3 Extended Register Write Long Command Sequence

250 The Extended Register Write Long Command Sequence provides write access to the extended register space on a Slave using a 16-bit address. One to eight bytes of data may be written in a single Command Sequence. The extended register address is supplied in two Address Frames within the Command Sequence.

251 Figure 31 shows the Extended Register Write Long Command Sequence. The Extended Register Write Long Command Sequence starts with an SSC followed by the Extended Register Write Long Command Frame, two Address Frames and one or more Data Frames with the data to be written. The Command Sequence ends with a Bus Park Cycle.

252 The three LSBs of the Extended Register Write Command Frame, BC[2:0] in Figure 31, indicate the number of bytes to be written in the Command Sequence. BC2 is the byte count MSB. 0b000 indicates one byte shall be written and 0b111 indicates eight bytes shall be written.

253 The register address in the Command Sequence contains the address of the first extended register to be written. If more than one byte is written in a single Command Sequence, then the Slave's local extended register address shall be automatically incremented by one for each byte written up to address 0xFFFF, starting from the address indicated in the Address Frame. If the extended register address reaches 0xFFFF before the last Data Frame in the Command Sequence, then the content of the register at address 0xFFFF is overwritten with the overflow data.

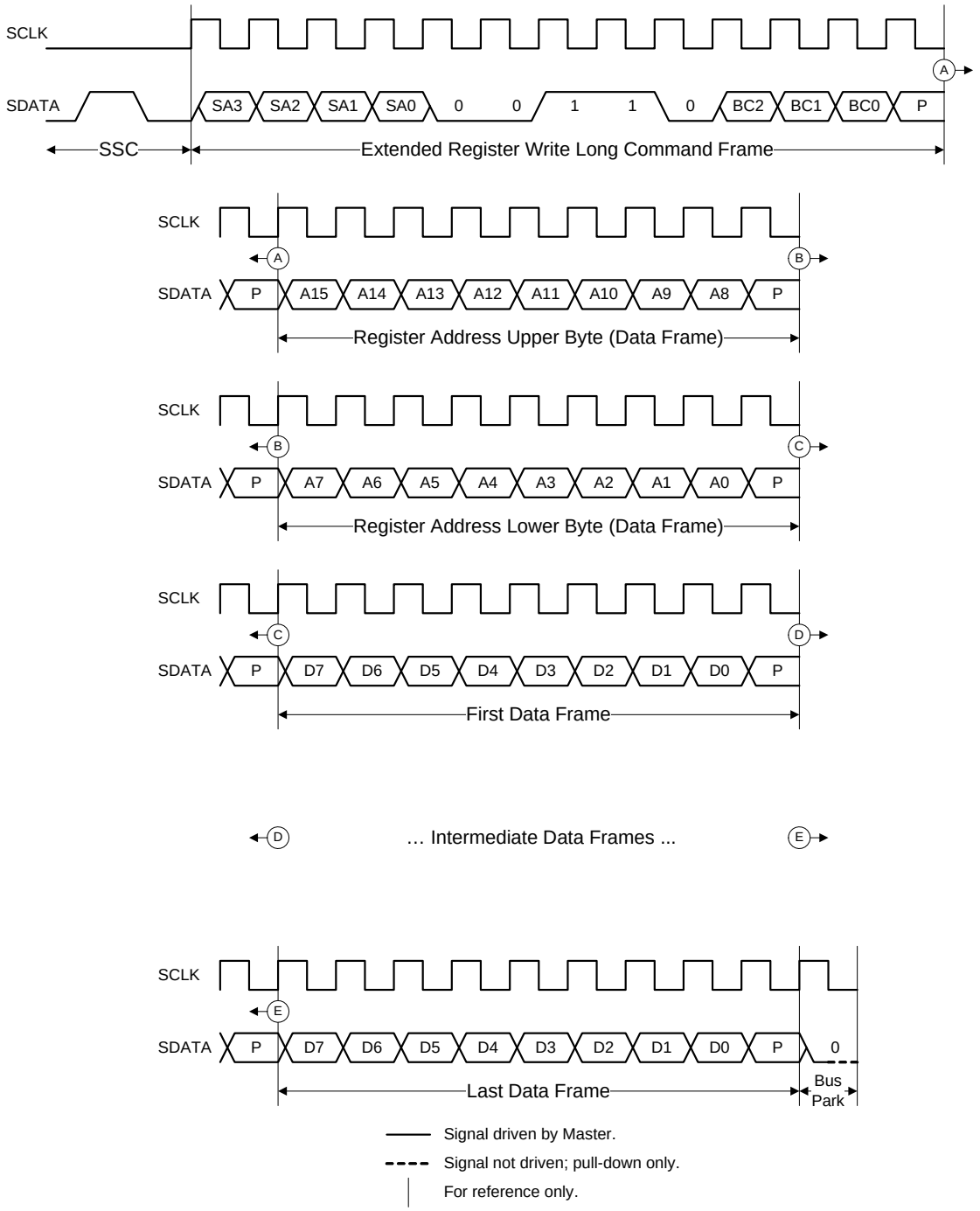

Figure 31 Extended Register Write Long Command Sequence

6.7.2.4 Extended Register Read Long Command Sequence

254 The Extended Register Read Long Command Sequence provides read access to the extended register space on a Slave using a 16-bit address. One to eight bytes of data may be read in a single Command Sequence. The extended register address is supplied in two Address Frames within the Command Sequence.

255 Figure 32 shows the Extended Register Read Long Command Sequence. The Command Sequence starts with an SSC followed by the Extended Register Read Long Command Frame, two Address Frames and one or more Data Frames with the data read from the Slave. A Bus Park Cycle occurs between the Address Frame and the Data Frames. The Command Sequence ends with a Bus Park Cycle.

256 The three LSBs of the Extended Register Read Long Command Frame, BC[2:0] in Figure 32, indicate the number of bytes to be read in the Command Sequence. BC2 is the byte count MSB. 0b000 indicates one byte shall be read and 0b111 indicates eight bytes shall be read.

257 The register address in the Command Sequence contains the address of the first extended register to be read. If more than one byte is read in a single Command Sequence, then the Slave's local extended register address shall be automatically incremented by one for each byte read up to address 0xFFFF, starting from the address indicated in the Address Frame. If the extended register address reaches 0xFFFF before the last Data Frame in the Command Sequence then the content of the register at address 0xFFFF is read multiple times. An Extended Register Read Long Command Sequence by the Master to an unsupported Slave extended register address results in a No Response Frame from the Slave. If the address that results from auto-incrementing the Slave's local extended register address is for an unsupported register, then the Slave sends a No Response Frame to the Master in place of the Data Frame. The Command Sequence continues from the next extended register address.

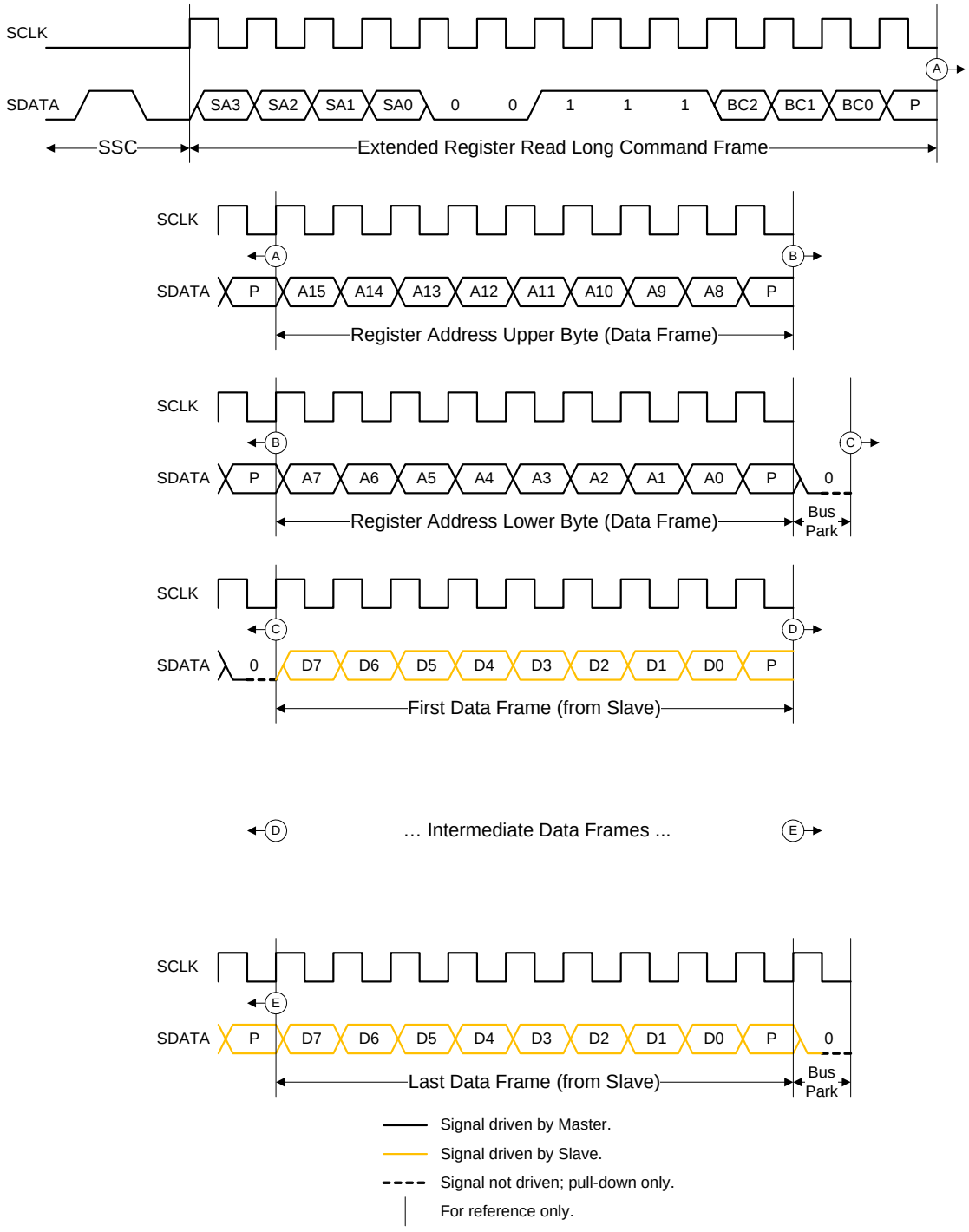

Figure 32 Extended Register Read Long Command Sequence

6.7.2.5 Register Write Command Sequence

258 Figure 33 shows the Register Write Command Sequence. The Command Sequence starts with an SSC, followed by the Write Command Frame containing the Slave address and the target register address and a Data Frame containing the data to be written. The Command Sequence ends with a Bus Park Cycle.

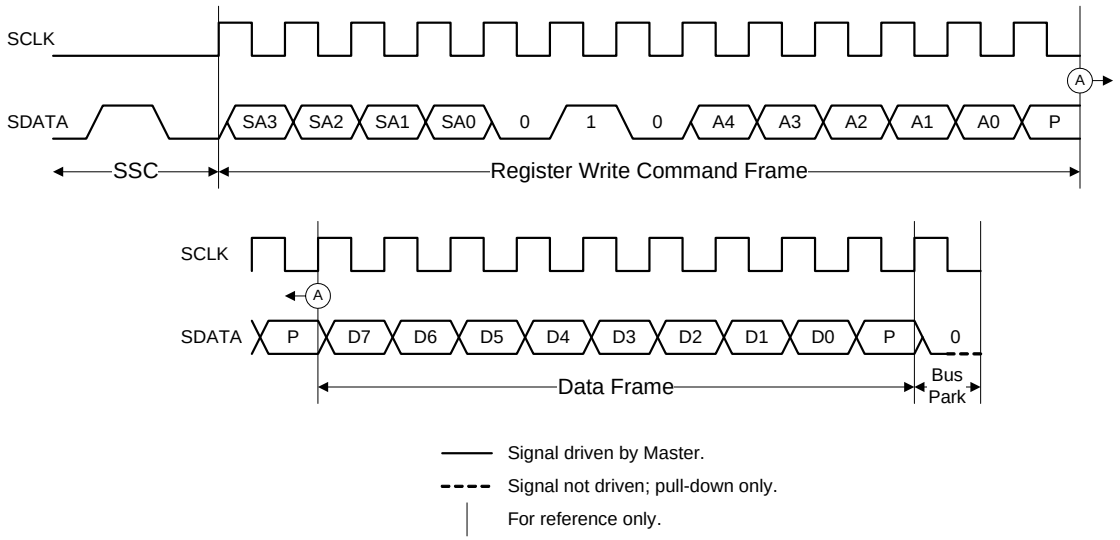

Figure 33 Register Write Command Sequence

6.7.2.6 Register Read Command Sequence

259 Figure 34 shows the Register Read Command Sequence. The Command Sequence starts with an SSC, followed by the Read Command Frame, a one-clock cycle Bus Park Cycle and a Data Frame sent by the Slave. The Command Sequence ends with another Bus Park Cycle.

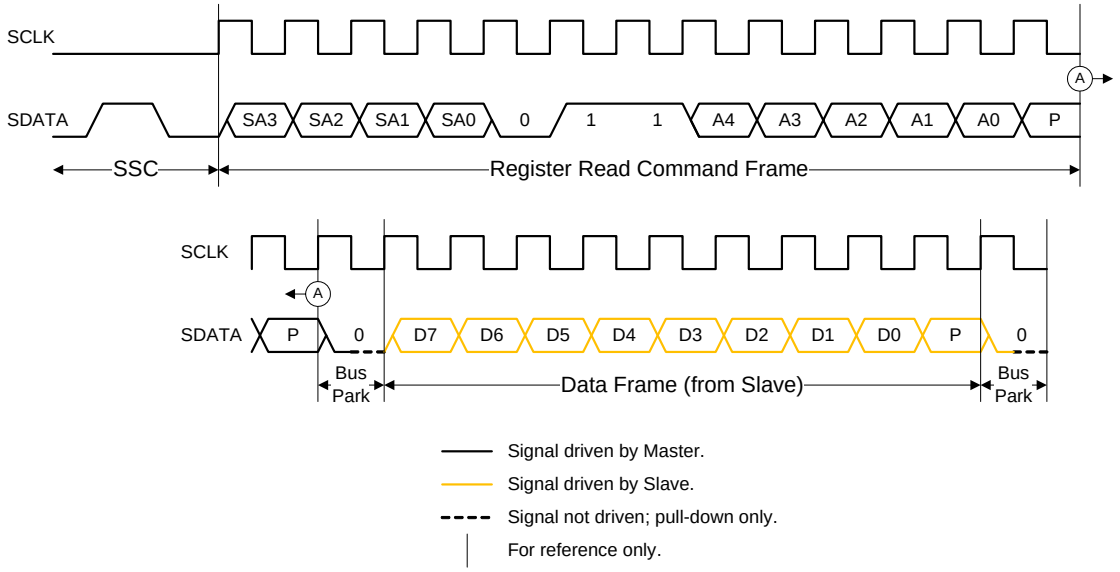

Figure 34 Register Read Command Sequence

6.7.2.7 Register 0 Write Command Sequence

260 Figure 35 shows the Register 0 Write Command Sequence. The Command Sequence starts with an SSC, followed by the Register 0 Write Command Frame containing the Slave address, a logic one, and a seven bit word to be written to Register 0. The Command Sequence ends with a Bus Park Cycle.

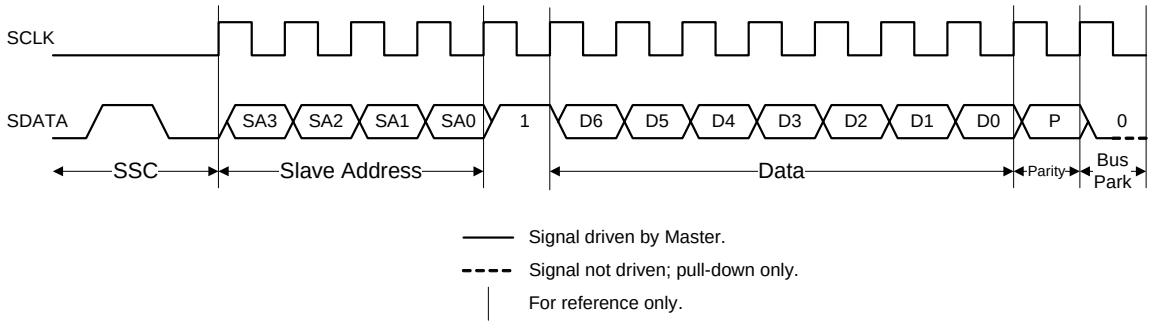

Figure 35 Register 0 Write Command Sequence

6.7.3 Half Speed Read Access

261 A Master shall support Half-speed read access. Half-speed data transfers shall be used with a Slave that is unable to provide data with the Full Speed Command Sequence timing. When a Slave is unable to meet the Full Speed electrical timing and waveform requirements of the SDATA Output Timing Parameter, T_D as defined in Table 4, the Master shall reduce the SCLK frequency to a maximum of 13 MHz to accommodate Slave operating characteristics. The Master may reduce the SCLK frequency to any supported frequency below 13 MHz. The Master needs to be aware of any Slaves that have this limitation.

262 Register reads from a Slave that requires additional time or additional SCLK cycles to fetch information from slow internal or external memory, external source, or for any other reason, may use one of the methods for delayed readback as explained in Section 6.10. Delayed readback and Half Speed read access may be utilized together to read a Slave with limited SDATA output timing and requirement for additional time for fetching information.

263 However, only the SCLK rate during the data portion of a Read Command Sequence needs to be reduced while the Command Frame and any addressing portion of the Data Frame may remain at the full SCLK speed. A system designer may choose to slow down the SCLK from the Master for the entire Read Command Sequence if the additional delay this introduces is acceptable.

264 Figure 36 shows a Half Speed Register Read. The clock rate for the SSC, Command Frame and first Bus Park Cycle is equal to SCLKint. The effective clock rate for the Data Frame and second Bus Park Cycle is equal to one-half the SCLKint frequency. The first Bus Park Cycle may be at the SCLKint rate because the Master is driving the SDATA line. The Slave is driving the SDATA line in the second Bus Park Cycle, so it occurs at the slower clock rate. The second Bus Park does not end until T_{SDATAZ} after the last falling edge of SCLK (see Section 4.2.2.1).

265 Requirements for Half Speed Register Read are extended to all other Read Command Sequences. Other Read Command Sequences are not shown.

266 The alignment between SCLKint and SCLK is shown as a reference only as the timing between SCLKint and SCLK is not specified and is implementation-specific.

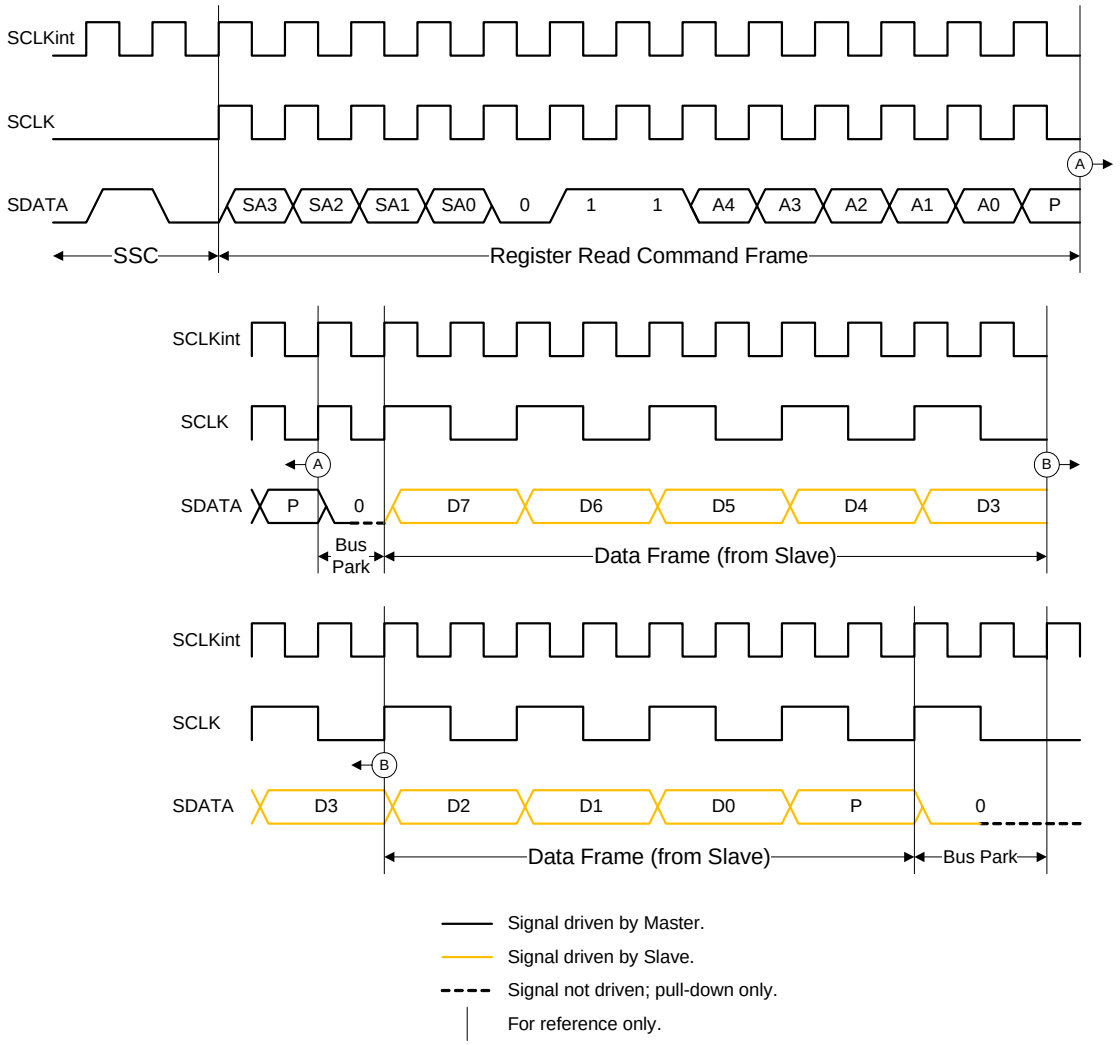

Figure 36 Read Half Speed

6.8 Device Enumeration

267 Devices on the RFFE bus are Master or Slaves. This section focuses on Slave enumeration. To be able to address devices on the bus Slave Identifier (SID) numbers are used to identify specific Slaves or groups of Slaves.

6.8.1 Slave Device Identifier

268 Each Slave on the RFFE bus shall have a Unique Slave Identifier (USID). The USID shall be assigned by the system integrator. This specification defines USIDs for a number of different types of Slaves that should be used if at all possible, allowing Slaves with fixed USIDs to be portable across multiple platforms.

269 SID address b0000 is reserved for Broadcast Messages and should not be used to identify an individual Slave. A Broadcast Message allows the Master to communicate to all Slaves in a single Command Sequence.

270 If there are less than fifteen Slaves connected to the RFFE bus, the system integrator may assign more than one Slave ID to a Slave, provided that the device supports such an assignment. Group Slave IDs (GSID) may

be assigned to multiple devices in a system. GSID are described in detail in Section 6.8.3. Unique Slave Identifier numbers and Group Slave Identifier numbers are collectively referred to as Slave Identifiers (SID).

271 In the case where the same Slave is used multiple times in the system on the same bus, Slaves need a method to identify themselves. Devices with pins or pads which may be tied or low to provide a unique USID is one option. Another option is to have multiple RFFE buses. If the total number of Slaves is fifteen or less, then SCLK could be shared. A second SCLK should be used for systems with sixteen or more Slaves. With a shared SCLK, only one SDATA may be active at a given time.

272 The predefined SIDs listed in Table 13 are a guideline for choosing IDs and are not mandatory. Guidelines are given so that a component may be used in multiple systems without requiring its USID to be programmed or programmable.

Table 13 Slave Identifiers

SID[3:0]	Description
1111	PA Module 1
1110	PA Module 2
1101	Spare (user-defined)
1100	Spare (user-defined)
1011	Antenna Switch Module 1
1010	Antenna Switch Module 2
1001	Spare (user-defined)
1000	Spare (user-defined)
0111	Antenna Tuning Module 1
0110	Spare (user-defined)
0101	Power Control Module 1
0100	Spare (user-defined)
0011	LNA Module 1
0010	Spare (user-defined)
0001	Spare (user-defined)
0000	Broadcast ID

6.8.2 Unique Slave Identifier

273 An RFFE bus may have up to fifteen Slaves. Each Slave on the bus shall be assigned a USID between 0b1111 and 0b0001, inclusive. USIDs do not have to be sequential. For example, in a system with four Slaves, one Slave might have a USID of 0b0001, the second Slave might have a USID of 0b1010, the third Slave might have a USID of 0b1110 and the fourth Slave might have a USID of 0b0011.

274 If a device supports GSIDs, then the USID registers may be individually user-defined as to which GSIDs the device supports, if any. The USID registers in a device may respond to a single GSID, multiple GSIDs, or only the USID depending on how the registers are defined.

275 The use of GSIDs limits the number of USIDs available to the system. For example, in a system that uses two GSIDs, not including the Broadcast ID, no more than thirteen Slaves with unique SIDs may be present. GSID (or the Broadcast ID) shall be used only with Write Command Sequences.

6.8.3 Programmable USID

276 Due to the wide range of Slaves and vendors for these Slaves an optional programmable USID feature exists. The programmable USID feature utilizes three reserved registers. MANUFACTURER_ID is a 10-bit value that is split across two bytes.

Table 14 Programmable USID Registers

Register Address	Bits	Register Name	Notes
0x001D	7:0	PRODUCT_ID[7:0]	This is a read-only register. However during the programming of the USID a Write Command Sequence is performed on this register even though the write does not change its value.
0x001E	7:0	MANUFACTURER_ID[7:0]	This is a read-only register. However during the programming of the USID a Write Command Sequence is performed on this register even though the write does not change its value.
0x001F	7:6	SPARE[1:0]	These are read-only bits that are reserved and yield a value of 0b00 at readback.
	5:4	MANUFACTURER_ID[9:8]	These bits are read-only. However during the programming of the USID a Write Command Sequence is performed on this register even though the write does not change its value.
	3:0	USID[3:0]	These bits are read-only for a device not supporting programmable USID; however, for programmable USID they are read/write. Performing a write to this register using the described programming sequences will program the USID in devices supporting this feature. These bits store the USID of the device.

277 The USID shall be programmed using only the programming procedures described in this section. A Master shall support both USID programming procedures. A Slave shall support at least one of the USID programming procedures.

278 The following programming procedure provides details for using Register Write Command Sequences to program the USID:

279 • Register Write Command Sequence to register 0x001D; the Data Frame needs to match PRODUCT_ID.

280 • Register Write Command Sequence to register 0x001E; the Data Frame needs to match MANUFACTURER_ID[7:0].

281 • Register Write Command Sequence to register 0x001F with SPARE[1:0], MANUFACTURER_ID[9:8], and the new USID in the Data Frame; if the PRODUCT_ID and the MANUFACTURER_ID match, then a new USID is programmed.

282 If the programming procedure Command Sequences do not occur in the order given, the Slave shall not update its USID.

283 If during the programming procedure, the Slave receives any other Command Sequence with an SA equal to the Slave's USID, or the Broadcast ID, the Slave shall not program its USID and the programming procedure shall be terminated. However, the Master may access other devices on the RFFE bus during the programming procedure. Command Sequences sent to other devices on the RFFE bus shall not cause the Slave being programmed to terminate the programming procedure.

284 The programming procedure for programming a new USID using the Register Write Command Sequence is shown in Figure 37.

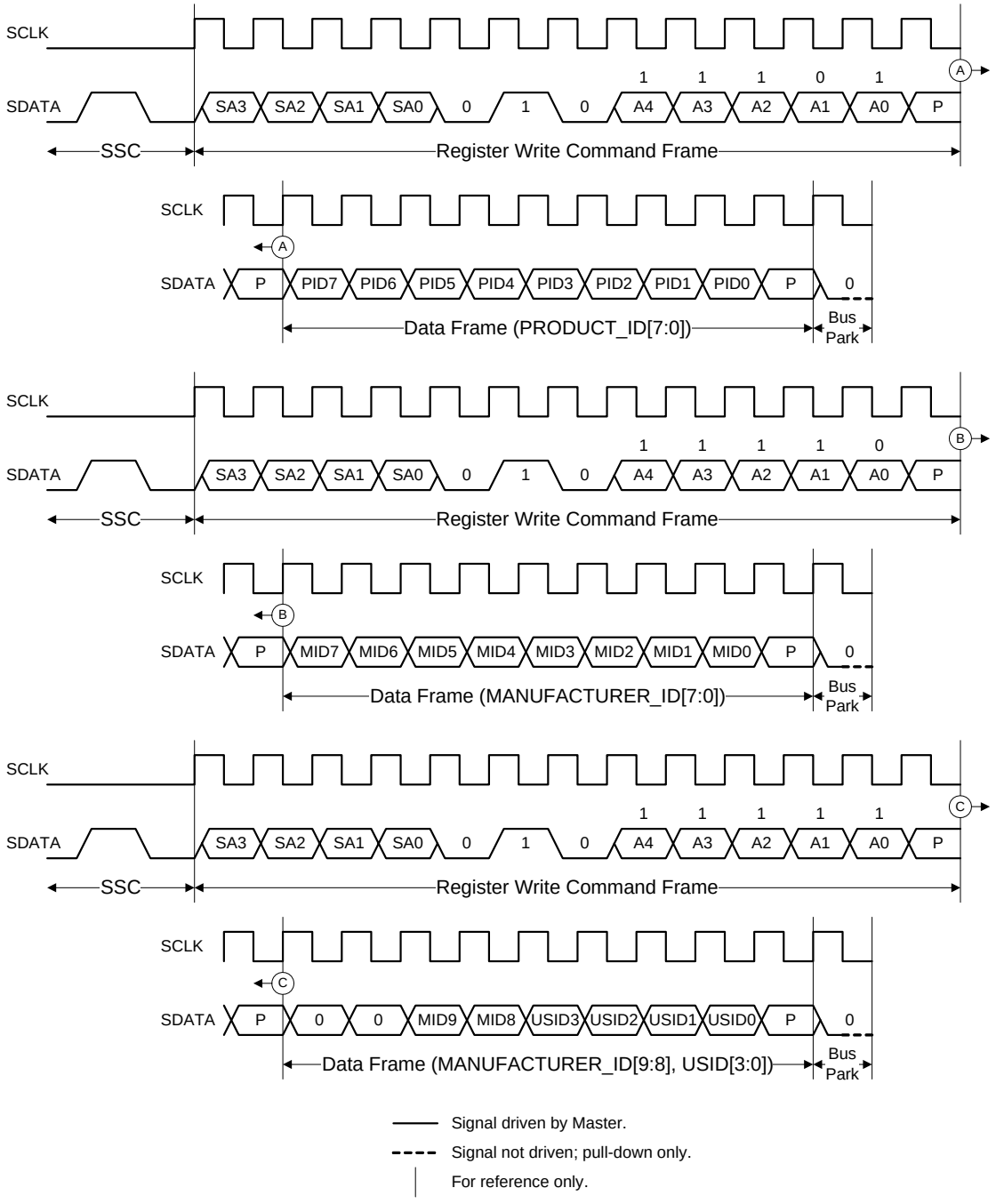

Figure 37 Register Write USID

285 The following programming procedure provides details for using an Extended Register Write Command Sequence to program the USID:

286 • Command Frame: Current USID, 0b0000, BC[3:0] and P
where BC[3:0] = 0b0010 = three bytes of data

287 • Data Frame:
Byte 1: Starting Address: Address of PRODUCT_ID (0b00011101 and P = 0b1)
Byte 2: Product ID (Value of Register 0x001D and P)
Byte 3: MANUFACTURER_ID[7:0] (Value of Register 0x001E and P)
Byte 4: 0b00, MANUFACTURER_ID[9:8], New USID, P and BP

288 If the PRODUCT_ID and the MANUFACTURER_ID match, then a new USID is programmed.

289 The programming procedure for programming a new USID using the Extended Register Write Command Sequence is shown in Figure 38.

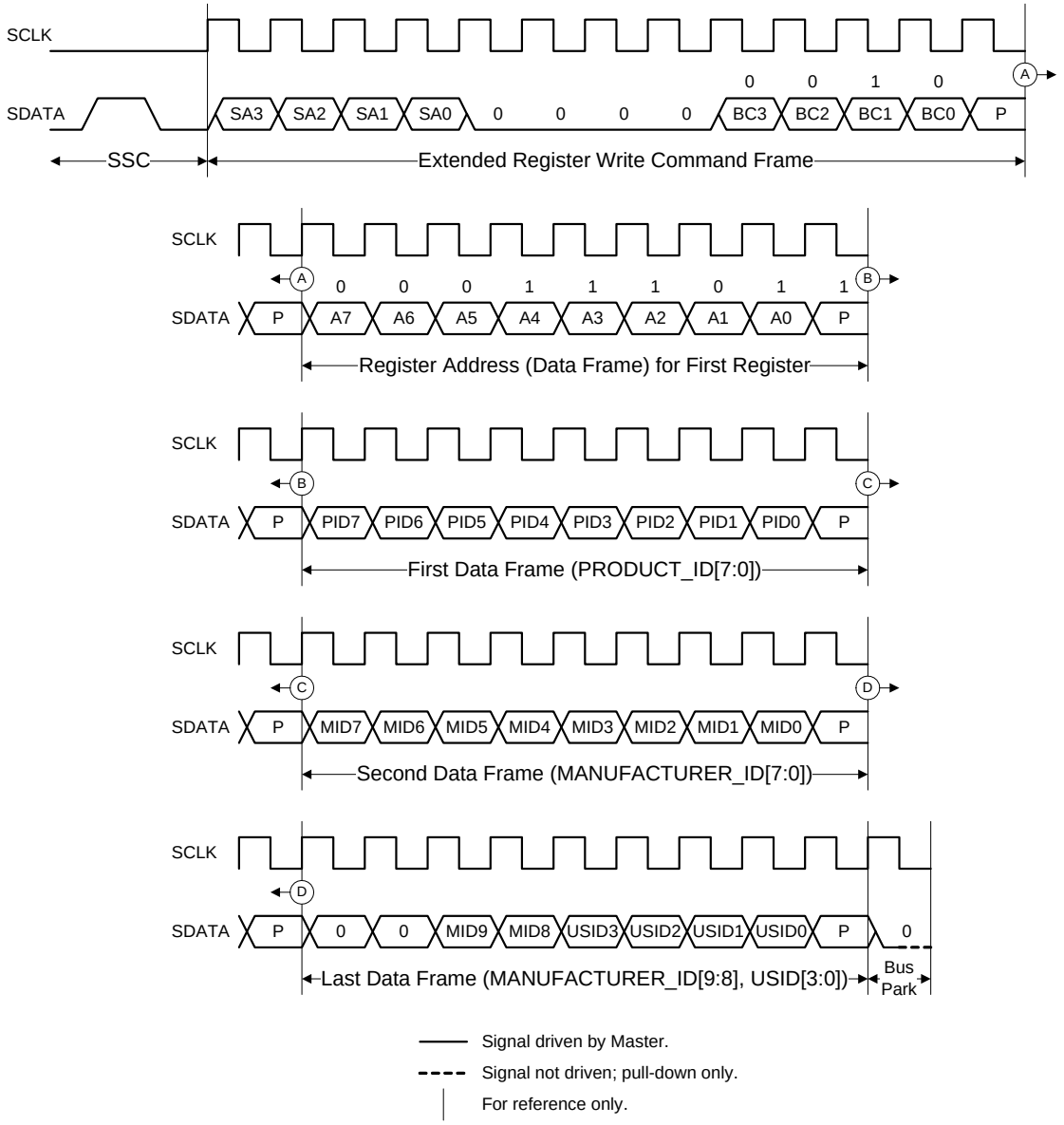

Figure 38 Extended Register Write USID

290 Since the Master supports both the Register Write and the Extended Register Write Command Sequences, a Slave may support either, or both, methods.

291 If a device supports an optional GSID, the GSID may be programmable or fixed. A programmable GSID is a user-defined register which may only be altered (reprogrammed) by a Command Sequence with a matching USID. The default value for a programmable GSID is 0b0000, which is the Broadcast ID. A programmable GSID may be turned off by programming the GSID to be equal to the USID.

292 When the device enters the STARTUP state the USID reverts back to its initial USID. However, if the device is placed in the LOW POWER state, the programmed USID remains the USID for the device. This also applies to GSIDs.

293 If a Slave does not support the feature of a programmable USID, it shall reserve address location 0x001F, and if it is a readable device, it could output the USID during a read operation of this address.

294 Read-supporting devices shall not reuse the same USID. Though not prohibited, write-only devices should not reuse the same USID. In the case where the same Slave is used multiple times, some type of mechanism such as having pins or pads which may be tied High or Low, or fuses within the device should be used to give each device a unique Slave address.

6.9 Slave Device Address Mapping

295 Registers in a device are byte addressable. Register Write and Register Read Command Sequences support five bits of address, which supports thirty-two addressable registers. The Extended Register Write and the Extended Register Read Command Sequences support eight bits of address which is 256 addressable registers. The optional Extended Register Write Long and the Extended Register Read Long Command Sequences have sixteen bits of address which supports 65536 register locations.

296 The physical register length of any user-defined register may be zero to eight bits in length. During a write the unused bits are not stored by the Slave, but are used for parity check purposes. A readback of a register less than eight bits should read back 0's for the bits that are not populated.

297 Register 0 is a user-defined register which may be written into using with a Register Write, an Extended Register Write, an Extended Register Write Long, or a Register 0 Write Command Sequence. Although all registers may be up to eight bits, a Register 0 Write Command Sequence only writes to the seven LSBs of Register 0. The Slave Register Mapping is shown in Table 15.

Table 15 Slave Register Mapping

Register Address	Register Name	Number of Bits	Notes
0x0000	REGISTER_0	0 to 8	User-defined register content
0x0001 to 0x001B	User-defined register	0 to 8	User-defined register name and content
0x001C	PM_TRIG	8	Power Mode and Trigger Register
0x001D	PRODUCT_ID	8	Product Identification
0x001E	MANUFACTURER_ID	8	Manufacturer Identification bits 7:0
0x001F	SPARE(1:0) MANUFACTURER_ID(9:8) USID(3:0)	8	Bits 7:6 Spare Bits 5:4 MANUFACTURER_ID(9:8) Bits 3:0 Programmable Unique Slave Identifier
0x0020 to 0xFFFF	User-defined register	0 to 8	User-defined register

6.9.1 Slave Device Reserved Addresses

298 There are four reserved address locations. These reserved addresses are used for power mode configuration, trigger functionality, reading the PRODUCT_ID, reading the MANUFACTURER_ID, and programming the USID. These register locations shall be reserved in a Slave. The implementation of the reserved registers functionality is optional. For example a write-only device would not support reading the PRODUCT_ID and MANUFACTURER_ID registers.

6.9.1.1 PRODUCT_ID

299 The PRODUCT_ID[7:0] register is eight bits, which allows for 256 unique device identifiers. These eight bits are defined by the vendors or systems outside the scope of this specification. For example, a vendor manufactures two different GSM PAs and one ASM. The vendor might define the PRODUCT_ID as shown in Table 16.

Table 16 Slave PRODUCT_ID Example

Device	USID[3:0]	MANUFACTURER_ID	PRODUCT_ID
GSM PA	1111	Company A	0b0000 0000
GSM PA	1111	Company A	0b0000 0001
ASM	1011	Company A	0b0000 0000

6.9.1.2 MANUFACTURER_ID

300 The MANUFACTURER_ID[9:0] register is ten bits in length, which allows for 1024 unique vendors. This list of manufactureres [MIPI04] is controlled by the MIPI Alliance. Members may be added to this list as needed. In each device this register is fixed and is a read-only register. This register is also used during the programming of the USID.

6.9.1.3 USID

301 The reserved register USID is used to program a new USID for a Slave. The details of this function are described in Section 6.8.3.

6.9.1.4 PM_TRIG

302 The PM_TRIG is a reserved register that is shared by the power modes and the trigger function. The PWR_MODES register occupies the two MSBs and the TRIG_REG are the six LSBs of the PM_TRIG register.

Table 17 PM_TRIG(7:0)

Bits	Register
7:6	PWR_MODE
5:0	TRIG_REG

6.9.1.4.1 POWER MODES

303 In RFFE the settings for the various Power Modes are available via the reserved PWR_MODE register bits, which are the upper two bits of the PM_TRIG register (PM_TRIG[7:6]). These two bits support three possible operating states for an RFFE Slave device, as shown in Table 18. Note that the fourth possible state for PWR_MODE(1:0) is 0b11, which is undefined (Reserved), and thus Slaves shall ignore a Write of this value to PWR_MODE(1:0). A Master shall not write this value to PWR_MODE(1:0).

Table 18 Power Modes

PWR_MODE(1:0)	Power Mode	Write Value	Read Value
00	Normal Operation (ACTIVE)	0b00	0b00
01	Default Settings (STARTUP)	0b01	0b00

Table 18 Power Modes (continued)

PWR_MODE(1:0)	Power Mode	Write Value	Read Value
10	Low Power (LOW POWER)	0b10	0b10
11	Reserved	N/A	N/A

304 When an RFFE Slave is initially powered up, and comes out of reset, the Slave registers shall be in their default settings, whether these defaults are fixed, adjustable, or programmable, and the PWR_MODE state is set to Normal Operation, which is the ACTIVE state.

305 PWR_MODE sets the Default Settings mode via a Command Sequence that synchronously puts the Slave registers to their default settings, whether these defaults are fixed, adjustable, or programmable, after which PWR_MODE goes to the Normal Operation Mode.

306 The Low Power Mode allows the Slave to put device-specific defined registers to their low power state value. The system user defines which registers have a low power state value. When PWR_MODE is switched out of the Low Power Mode back to the Normal Operation Mode, the registers shall output their ACTIVE state value. To exit the Low Power Mode into the Default Settings Mode, a value of 0b01 should be written to PWR_MODE(1:0). Not all Slaves and not all registers within a Slave will necessarily have a low power value. This is device- and system-specific.

307 Figure 39 illustrates an example implementation of the LOW POWER state value for a register. In this example a LOW POWER state value has been defined for bits 2 and 0 of Register A. When instructed to enter the LOW POWER state the values of RegisterA(2) and RegisterA(0) would take on the value of “Low Power Value” as shown in the diagram, while RegisterA(7:3) and RegisterA(1) would all retain their previous values for the duration of LOW POWER. Note that “Low Power Value” may be unique for each bit.

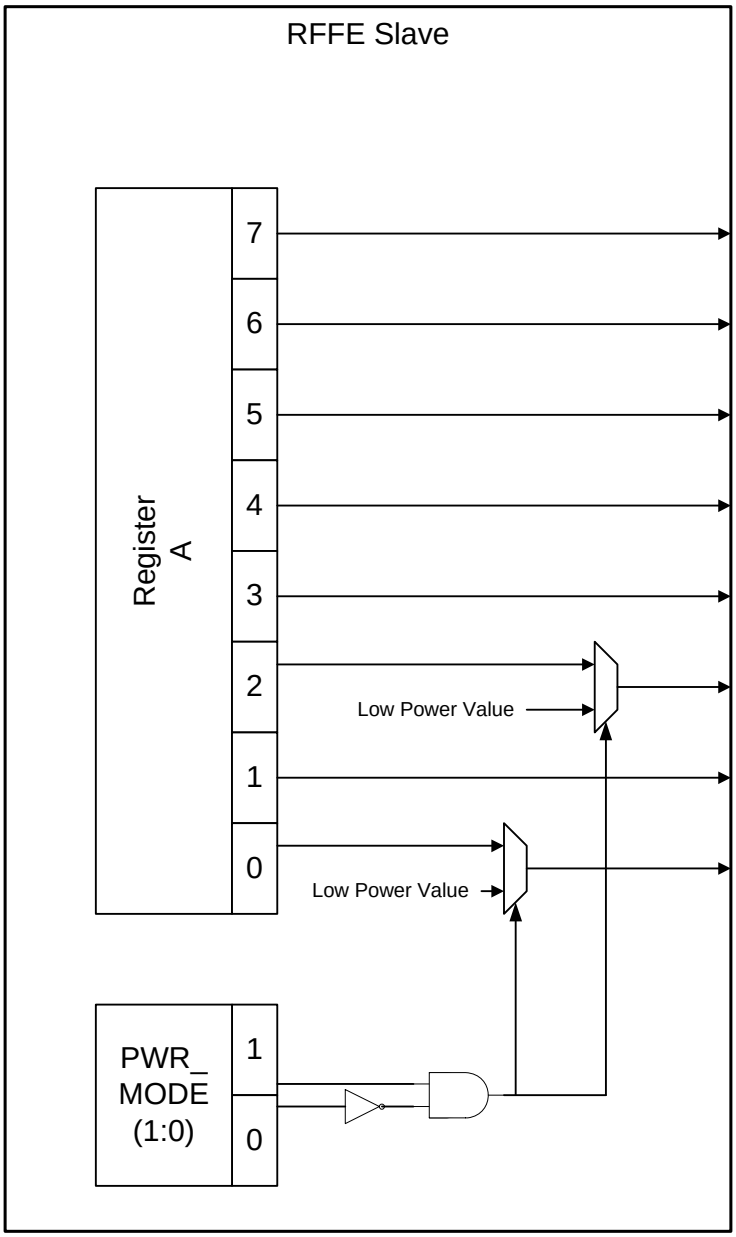

Figure 39 Low Power Mode Applied to Selected Register Bit Values

308 For more advanced Slaves that may require more Power Modes, it is possible to utilize one of the user-defined registers for this purpose; however, these additional states shall not replace or interfere with those defined herein for PWR_MODE. Such additional states could be used for system- or device-specific power modes beyond what is currently specified. This register may be designed to be broadcast-addressable or device-addressable, at the discretion of the implementer.

6.9.1.4.2 Trigger Modes

309 Group trigger functionality is essential for meeting precise timing requirements especially if risk for congestion using ordinary messages arises. Triggers provide a tool to solve bandwidth limitations, and to achieve simultaneous precise timing at multiple destinations.

310 The trigger function is not a mandatory function in every Slave, but may be used if multiple registers are required to be loaded at exactly the same time, or if the timing of the system requires multiple registers to be loaded in a timing critical window of time.

311 • Device trigger mode: If a register is more than eight bits, which is the maximum width of a register, multiple registers could be loaded and triggered to occur at the same time.

312 • Broadcast trigger mode: Multiple devices require registers to be updated at the same time.

313 • When reading back a register that is on a trigger the readback data is always the information stored in the destination register and not the shadow register.

314 • If multiple triggers are triggered in the same write, they occur at the same time.

315 For more advanced Slaves that require more triggers a user-defined register can be used for system/device specific triggers.

316 The reserved TRIG_REG register shares an 8-bit register with the PWR_MODE register. TRIG_REG is six bits and is the six LSBs of the PM_TRIG(5:0). The six bits support three separate triggers as shown in Table 19.

Table 19 TRIG_REG Definition

Bit	R/W	Name	Notes
5	R/W	Trigger Mask 2	If this bit is set trigger 2 is disabled. When disabled writing to a register that is associated to trigger 2 the data goes directly to the destination register.
4	R/W	Trigger Mask 1	If this bit is set trigger 1 is disabled. When disabled writing to a register that is associated to trigger 1 the data goes directly to the destination register.
3	R/W	Trigger Mask 0	If this bit is set trigger 0 is disabled. When disabled writing to a register that is associated to trigger 0 the data goes directly to the destination register.
2	W	Trigger 2	A write of a one to this bit loads trigger 2's registers.
1	W	Trigger 1	A write of a one to this bit loads trigger 1's registers.
0	W	Trigger 0	A write of a one to this bit loads trigger 0's registers.

317 All three triggers may be set for each device individually or may be broadcast to multiple devices. All triggers may be controlled individually or as a group.

318 The trigger masks may be set solely for each device individually. If a register is a trigger register by default the trigger is not masked. Only a write to a specific device using the USID shall change the mask control bits.

6.10 Slaves With Delayed Readback

319 The RFFE protocol does not support the concept of a data ready bit during a read operation. A Slave may require extra clock cycles or more time to retrieve data during a read operation. While the implementer is not limited to the options given here, the solution chosen shall comply with the RFFE Command Sequence set.

320 Some possible examples are:

321 • Repeat the Register Read Command Sequence

322 • Repeat the Register Read Command Sequence with multiple register reads

323 • Pre-write a Register

324 • Use an Extended Read Command Sequence

6.10.1 Repeat the Register Read Command Sequence

325 The first example involves repeating the Register Read Command Sequence. In this case, the data read by the Register Read Command Sequence is the data that was requested by the previous Register Read Command Sequence of the same register. In the case where the previous data is not available for any reason, a No Response Frame may be used.

326 A Slave that supports this method shall be capable of providing the data requested by a Register Read Command Sequence by the start of the Data Frame of the subsequent Register Read Command Sequence of the same register.

327 Other RFFE bus activities or Command Sequences may occur between Register Read Command Sequences of the same register. However, the Slave shall supply the data requested from the preceding Register Read Command Sequence in response to the second Register Read Command Sequence of the same register.

328 The Repeat Register Read Command Sequence method is illustrated in Figure 40.

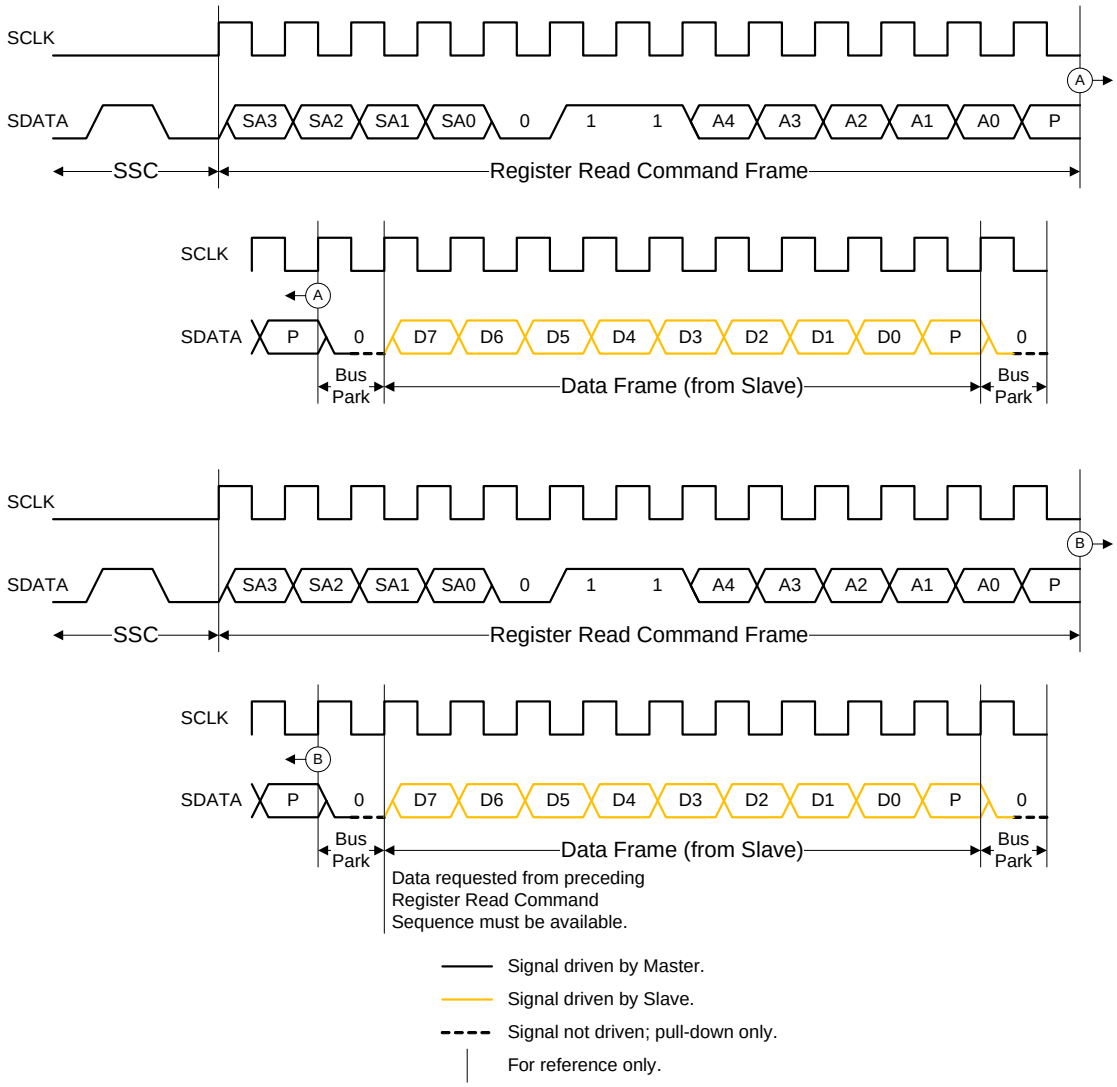

Figure 40 Repeat the Register Read Command Sequence

6.10.2 Repeat the Register Read Command Sequence with Multiple Register Reads

329 The second example is similar to the previous one, but may be used when multiple bytes of data need to be read. In this case, the data read by a Register Read Command Sequence is the data that was requested by the previous Register Read Command Sequence. In the case where the previous data is not available for any reason, a No Response Frame may be used.

330 The Master may ignore the data provided by the Slave in the first Register Read Command Sequence, and may repeat the Register Read Command Sequence of the last register to ensure it receives all the desired data correctly.

331 A Slave that supports this method shall be capable of providing the data requested by a Register Read Command Sequence by the start of the Data Frame of the subsequent Register Read Command Sequence.

332 Other RFFE bus activities or Command Sequences may occur between Register Read Command Sequences.

333 This method is suitable for a Slave that needs additional time to retrieve the requested data and has multiple bytes of data needed by the Master. This method is more bandwidth-efficient when the Master needs to read large amounts of data from the Slave.

334 The Repeat Register Read Command Sequence with Multiple Register Reads method is illustrated in Figure 41. The example in the figure shows consecutive addresses, which is not a requirement, but is used to illustrate one possible application of this method.

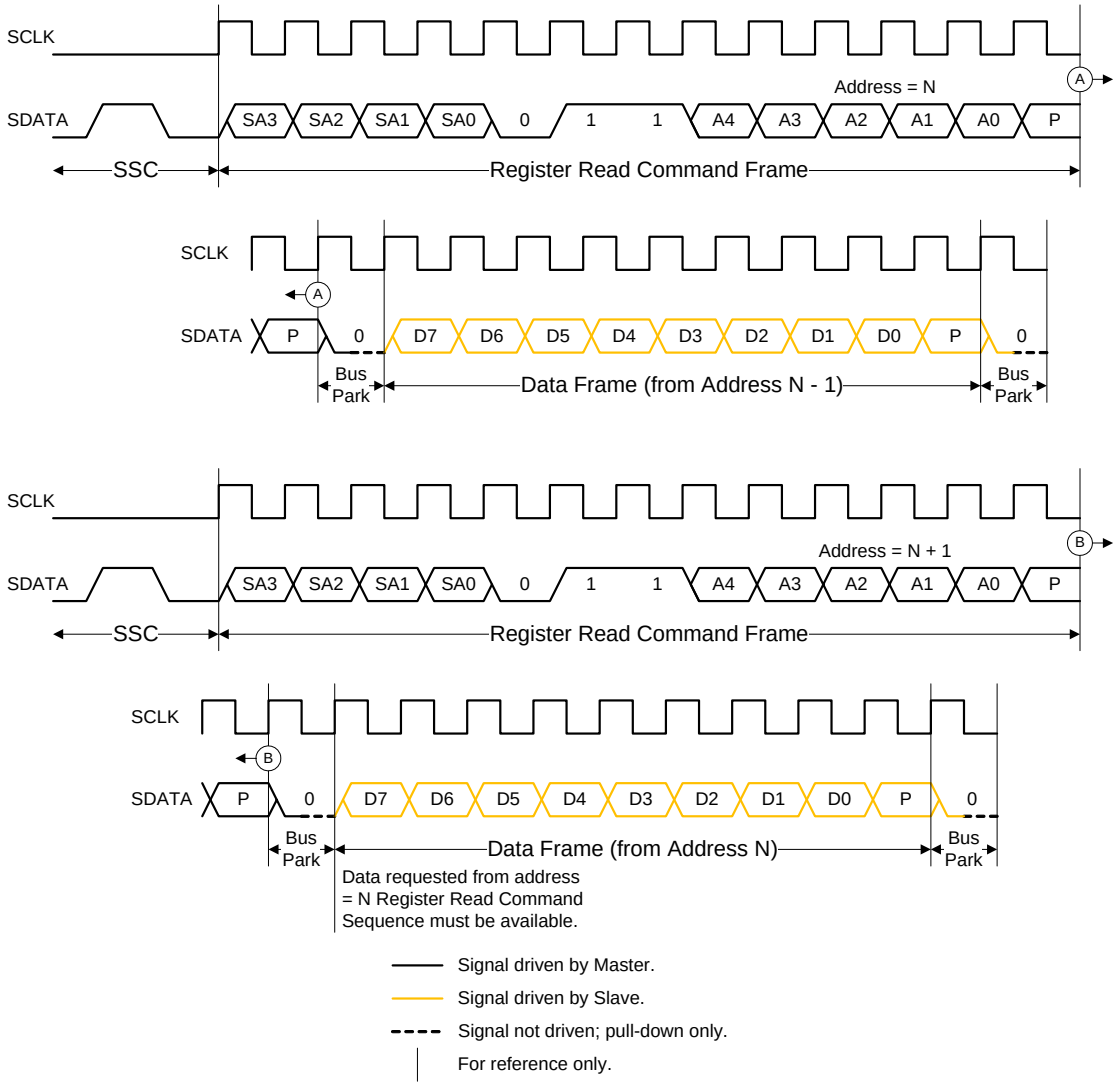

Figure 41 Slow Register Read with Multiple Read Command Sequences

6.10.3 Pre-Write a Register

335 The third example involves pre-writing a register using the Register Write Command Sequence to request the delayed readback data, which is then read with a Register Read Command Sequence. In this option the Register Write Command Sequence is used by the Slave to initiate the fetching of the delayed readback data. The register that is chosen in the Slave to initiate the fetching of the data may be the same register address as the register it wants to read, or it may be a different register address. In the case where the pre-write register address and the read register address are the same, the Register Write Command Sequence data is not stored

in the register location - it is just used to initiate the fetching of the data. The data read by the Register Read Command Sequence is the delayed readback data that was requested by the Register Write Command Sequence. In the same address case the data is read-only for this register address, as the Write Command only initiates the fetching of the data.

336 A Slave that supports this method shall be capable of providing the data requested by the Register Write Command Sequence by the start of the Data Frame of the subsequent Register Read Command Sequence.

337 Other RFFE bus activities or Command Sequences may occur between the Register Write Command Sequence and the Register Read Command Sequence used by this method. However, the Slave shall supply the data requested by the Register Write Command Sequence in response to the Register Read Command Sequence.

338 The pre-write a Register method is illustrated in Figure 42.

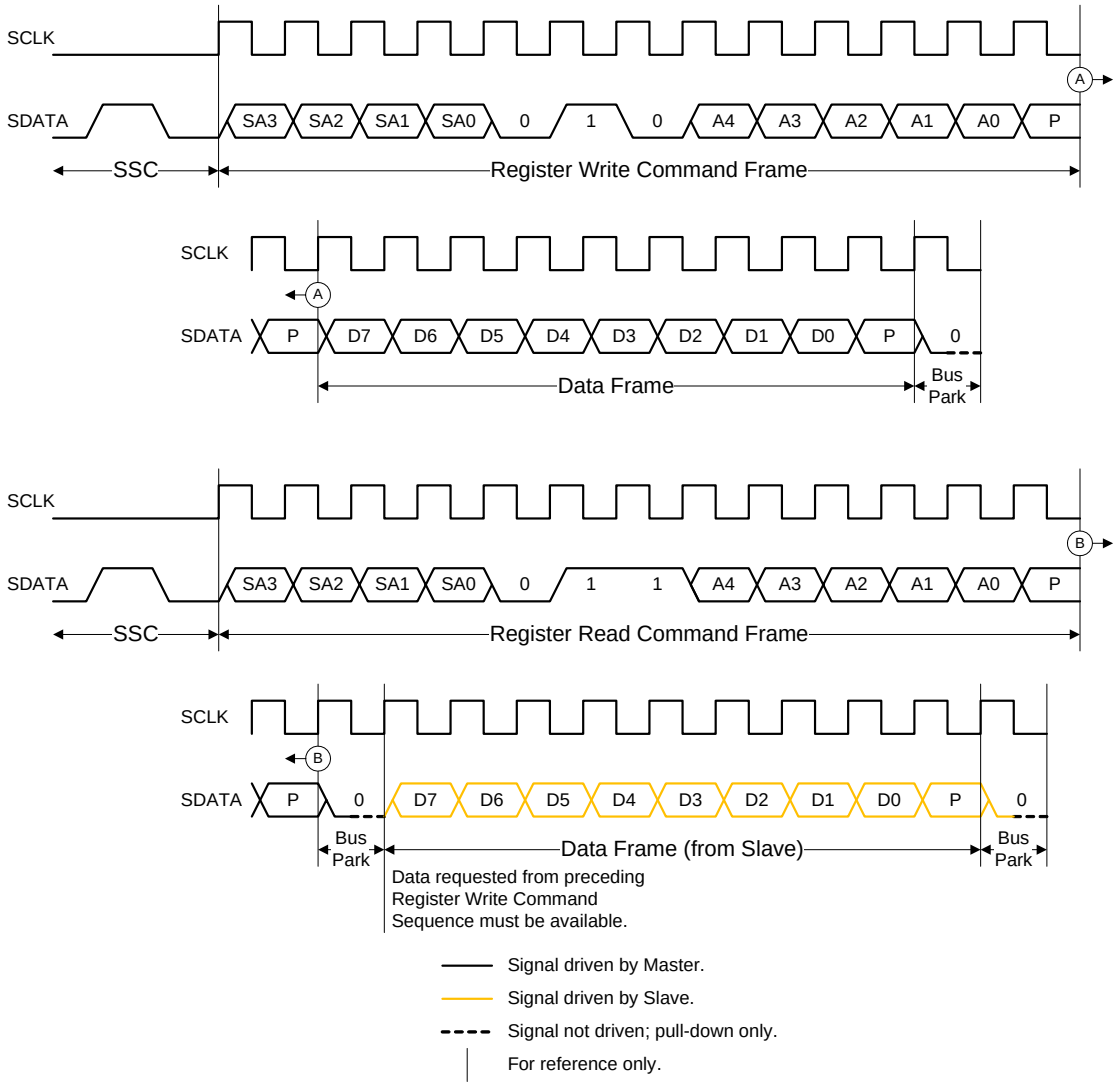

Figure 42 Pre-Write a Register

6.10.4 Use an Extended Register Read Command Sequence

339 If the Slave supports the Extended Register Read Command Sequence it could be used for a register with a slow readback. Using the method in this example multiple bytes of data may be read in a single Command Sequence. The Command Sequence may be extended if necessitated by the amount of time or clock cycles required to fetch the data. The number of clocks and the amount of time available is bounded by the length of the Command Sequence. The read triggering mechanism uses the first address and number of bytes argument to trigger the data request. The data in preceding Frames is application-specific.

340 The following description is an example of using an Extended Register Read Command Sequence.

341 Register 0x0009 is a slow readback register and requires sixteen clock cycles to fetch the data.

342 The Master issues an Extended Register Read Command Sequence with a data width of three bytes and a starting address of 0x0007. Depending on whether or not addresses 0x0007 and 0x0008 are used, the Slave returns either actual data from those registers, which may be defined as all zeroes with a parity bit of “1”, or a No Response Frame if the respective register address is not supported. The Master may process or ignore each returning Data Frame individually. Data provided from address 0x0007 and 0x0008 shall be consistent with any other Slave-supported Read Command Sequences accessing these register locations. The important point to note is that the process for fetching the data from address 0x0009 begins after the Command Frame and the Address Frame are processed by the Slave.

343 The Extended Register Read Command Sequence method is illustrated in Figure 43.

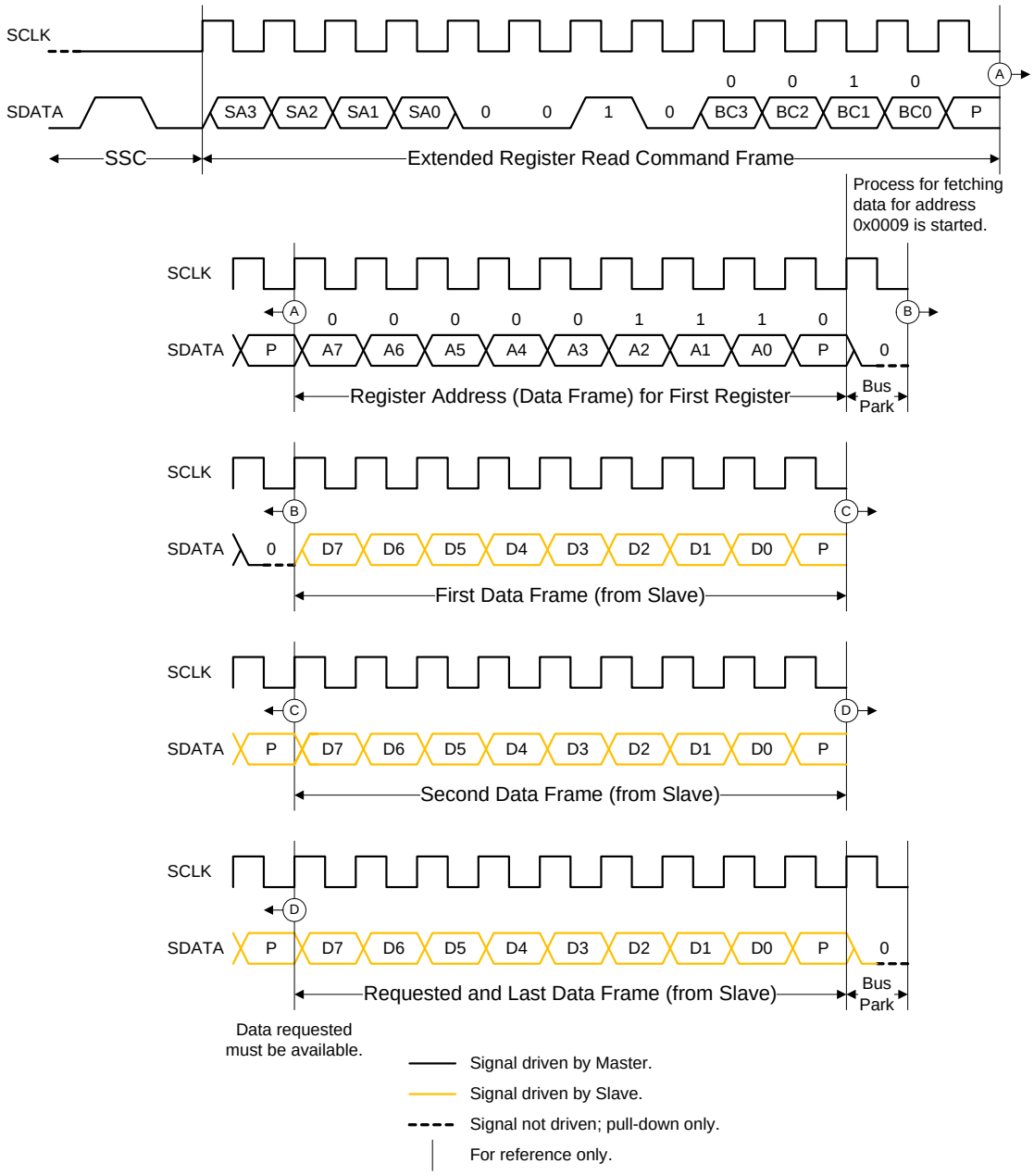

Figure 43 Using an Extended Register Read Command Sequence

7 Applications

344 This section defines features related to the usage of the RFFE specification in real world implementations and configurations. Some features are mandatory and defined in detail whereas others are vendor choices and thus only mentioned and briefly explained. These may be regarded as options as well, but usually do not require any additional functionality in the Master or Slave, so it is more of an explanation of how to use existing mandatory features to accomplish an optional desired functionality.

345 The RFFE bus is intended to be used in conjunction with a multitude of different air interface standards. Ideally all possible use cases would be covered in this section; however, an exhaustive analysis and definition of all possible RFFE use cases is not possible within the scope of this Specification. Instead this section highlights a number of features that are necessary to ensure successful operation.

7.1 Programming Model

346 The intention of the programming model is to provide insight into how the interface is utilized in applications, which should aid in the development of the control SW. Command Sequence lengths vary as a function of access type and clock rate. Scheduling of the Command Sequences need to take this into account to ensure time critical control is issued at the correct time. A timing accurate access requires a certain procedure to be taken into account (see Section 7.1.2.1) and devices relying on the RFFE SCLK signal for clocking need to be considered appropriately (see Section 7.1.2.2) while designing the control scheme.

347 Register content, except for the specific RFFE registers defined in Section 6.9.1, is vendor-specific. Also, the allocation of the bits into the various registers is vendor-specific. Thus, it is not possible for the programming model to cover all possibilities for the actual control queues and underlying Command Sequences. Therefore, the definition and specification work is left to the RF control system designer.

348 The amount and timing criticality of the control is very case-specific. Proper bus load analysis must be made by the RF control system designer. This document provides only a toolset to help identify and solve potential issues.

349 The programming model in general assumes the use of complete Command Sequences in a sequential manner. The Command Sequences are various types of register writes and reads (see Section 6.4). A device supplier may also give the payload bits a higher meaning (i.e. program code), but that is entirely vendor-specific. RFFE merely provides a means to transfer those bits. Scheduling and issuance of the Command Sequences is left to the Master design, and thus this is also device-specific. RFFE bus compliance is nevertheless achieved by ensuring that Master and Slaves are implemented according to this Specification, while a great deal of freedom and flexibility is left to the implementer, particularly on the RF system level.

7.1.1 Basic Control Related Items

350 This section describes some basic information necessary for successful control implementation.

7.1.1.1 Command Sequence Building

351 Table 20 illustrates the Command Sequence length in SCLK cycles including the SSC (length is two SCLK cycles in these examples as well as in Table 21). Register 0 Write is only available for the register with address 0x0h. There is no corresponding Read.

Table 20 Command Sequence Length In SCLK Cycles as Function of Access Type

Access Type	Write	Read
Register 0 Write	16	N/A
Register	25	26

Table 20 Command Sequence Length In SCLK Cycles as Function of Access Type

Access Type	Write	Read
Extended Register	34 to 169	35 to 170
Extended Register Long	43 to 106	44 to 107

352 Typical Write Command Sequence lengths with a single byte of data are:

353 • 16 SCLK cycles in Register 0 Write (only seven LSBs are written)

354 • 25 SCLK cycles in Register Write

355 • 34 SCLK cycles in Extended Register Write

356 • 43 SCLK cycles in Extended Register Write Long

357 The Register Write supports a 5-bit address space, Extended Register Write and Extended Register Write Long 8-bit and 16-bit respectively. Same applies to corresponding Read Command Sequences.

Table 21 Command Sequence Length Versus Command Sequence Type & Clock Rate

Clock Rate	Register 0 Write	Register Write	Register Read	Extended Register Write	Extended Register Read	Extended Register Write Long	Extended Register Read Long
26 MHz	0.62 µs	0.96 µs	1.00 µs	1.31 µs	1.35 µs	1.65 µs	1.69 µs
19.2 MHz	0.83 µs	1.30 µs	1.35 µs	1.77 µs	1.82 µs	2.24 µs	2.29 µs
13 MHz	1.33 µs	1.92 µs	2.00 µs	2.62 µs	2.69 µs	3.31 µs	3.38 µs

358 Table 21 summarizes Command Sequence lengths in µs for various accesses types as a function of a few typical bus clock rates. Note that only the shortest Command Sequences for Extended Register and Extended Register Long accesses have been included in the table.

359 Table 22 illustrates the impact of the Half Speed read for Command Sequence lengths corresponding to the respective Read Command Sequences in Table 21. The assumption here is that the readback is performed with a clock rate being half of the clock rate used during the Command Frame. The lengths of the Command Sequences in bits do not change but switching to a half frequency clock causes the readback Data Frames to consume double the time than in the equivalent normal read.

Table 22 Command Sequence Lengths Using Half-Speed Read

Command Frame Clock Rate	Reduced Readback Clock Rate	Register Read	Extended Register Read	Extended Register Read Long
26 MHz	13 MHz	1.42 µs	1.93 µs	2.85 µs
19.2 MHz	9.6 MHz	2.16 µs	2.86 µs	4.23 µs
13 MHz	6.5 MHz	2.81 µs	3.80 µs	5.62 µs

7.1.1.2 Control Scheduling and Timing Considerations

360 The time budgeted for control varies as a function of case and actual implementation. A similar device by two different vendors might require a different amount of control to perform the same task. There is additionally a settling time (in this context the time required before the controlled feature has become active, usually caused by the time analog blocks need to stabilize to the new settings) that vary significantly from one implementation to another.The RFFE specification cannot take vendor-specific issues like settling times into account so definition of the control budget is left to the RF system designer.

361 In addition to the typical length of bus access Command Sequences summarized in Table 21, a minimum of 10 ns has to be reserved for the Bus Park Cycle which separates two subsequent messages (see Section 6.2.4). Knowledge of the number of control Command Sequences, their types, and the time available for control allows the system designer to assess whether a control sequence may be accomplished. Note that the access time is a function of the actual clock rate (see Table 21) – the slower the clock that is used, the more time the Command Sequences consume.

362 Pre-loading of registers and usage of the triggering mechanism may be used to circumvent a potential timing bottleneck (see Section 6.9.1.4.2). A well designed triggering approach could be capable of performing all control with a precisely timed trigger Command Sequence. Defining such system is out of the scope of the RFFE specification and left to the RF system designer.

7.1.2 Procedures

363 There are two RFFE procedures to be mentioned in more detail. Precise access timing is fundamental for the applications utilizing the RFFE bus and additional clock provisioning is necessary to ensure that devices connected to the bus with no other clock source obtain clocking in a controlled manner.

7.1.2.1 Precise Access Timing

364 Very precise timing is sometimes desired. The RFFE specification cannot constrain the implementations by defining precise latency values. One reason is that although the processing time and Command Sequence triggering point could be defined, the settling time is still solution-dependent.

365 Device suppliers are expected to define latencies and provide that information to the system designer. The same applies to the trigger latency. An optimal solution is to include an additional timer for every trigger in the Slave to allow alignment or staggering of the triggering points. The lack of an always existing clock ruins this approach. A system designer might want to circumvent this issue by providing additional clocks (see Section 7.1.2.2), but the design of such solution is outside the scope of this document.

7.1.2.1.1 Write Access

366 Typically, write access activates the control immediately after receiving the last Data Frame. Special cases are those situations where additional clocks are needed for activation. The Master needs to know in advance the number of clock cycles required for each special case in order to provision the additional clock cycles.

367 It might be possible to constrain the latency variation by using the triggering mechanism (see Section 6.9.1.4.2).

7.1.2.1.2 Read Access

368 A timing-precise read access might be harder to obtain than a corresponding write access. This applies especially if some of the means to provide the Slave additional clock cycles for processing (see Section 7.1.2.2) are utilized. Only the Slave vendor may specify the exact timing for Extended Read or Read Command Sequences using multiple Command Sequences. For ordinary read accesses, the fetching point is, by definition, the bus turnaround unless fetching is bit-based instead of byte-based. Such a special case is outside the scope of this document.

369 A very precise timing might be required regardless of the read access used to obtain the result. The (group) trigger function could be used in this case.

370 A triggered read access requires that one of the triggers (see Section 6.9.1.4.2) is used for storing a particular register content for readback at a later time. Implementing such a read latch gives a timing precision of the read access corresponding to the timing precision in issuing the trigger Command Sequence. A triggered read access implemented in several devices could even use the group trigger functionality to enable a snapshot at multiple devices simultaneously.

371 The trigger would initiate the fetching of data, although the challenge here is to provide a sufficient number of clock cycles such that the data fetching may be completed during the clock cycles of the trigger Command Sequence (in principle this means during the remaining clock cycles of the access when trigger identification is done; in practice this likely requires only a cycle or two). Readback data may then be acquired with a read access at some later time. This approach might require a specific register to temporarily store the read data, but that is left to the implementer.

372 Note that use of extended Command Sequences allows read access timing to be deterministic. The clock cycle in which the read access occurs may be known and thus, the Master may adjust the issuance of the Command Sequence such that the read clock cycle occurs at the desired moment in time.

7.1.2.2 Provisioning of Additional Clocks for Processing

373 An RFFE device, in general, does not have any other clock input than SCLK provided by the interface. The device might nevertheless require a clock to perform various actions. SCLK is used in such cases. Typically, the SCLK provided during a Command Sequence is sufficient for a Slave. However, there are situations when additional clock cycles are necessary.

374 There are three ways to provide these additional clocks:

375 • A Command Sequence to another device

376 • A dummy Command Sequence and Frame

377 • Use of extended Command Sequences

378 A Command Sequence to another device is equal to any normal traffic on the bus. If there is traffic on the bus and it is timed suitably then there is no need for further actions to provide a clock. This is the preferred way especially at high bus load as other solutions further increase the load.

379 A dummy Command Sequence (examples of Command Sequences with desired dummy functionality are given in [MIPI06]) shall be transmitted if there is no ordinary traffic at the required time. The dummy Command Sequence does not cause any action by a Slave, and just provides the clock cycles needed.

380 The additional clock cycles might be required in conjunction with a Command Sequence. Extended Command Sequences shall be used in this case. Extended Command Sequences shall be constructed such that there are either unused address bytes, data bytes, or both. The actual frame construction thus becomes case-dependent. Slave implementers shall be notified that usage of all address or data bytes in extended Command Sequences prevents provisioning of the requested additional clock cycles because all available clock cycles are used for ordinary Frame decoding and there are no spare clock cycles left for performing the actions. In this case, a Slave implementer needs to find another vendor-specific solution.

381 It is the responsibility of the entity controlling the Master to determine if, and when, additional clock cycles are needed as well as selection of the method to do so. A Master shall support all three methods.

7.2 Command Sequence Timing

382 RFFE applications include cases where accurate timing of control Command Sequences is of utmost importance. Time accurate scheduling of control Command Sequences is managed by the entity controlling the Master; the RFFE bus cannot handle time conflicting Command Sequences. Such a case might be resolved either by shifting in time one of the conflicting Command Sequences, or if that cannot be done, by using the triggering mechanism (see Section 6.9.1.4.2).

383 Deterministic time critical control is most easily implemented by use of a single Master configuration. . This is one of the major motives for restricting the RFFE bus to single Master and non-RCS operations. In the case of systems with multiple RFFE buses, and thus involving one or more Masters, it is beyond the scope of the RFFE Specification to define the means for resolving any timing conflicts that may exist between the RFFE

buses. It is up to the implementers of the Master devices or the systems designers to provide for this capability.

384 Aborting an ongoing Command Sequence may be performed in some circumstances. However, this is primarily reserved for malfunction condition on the bus, and thus the use of an SSC to terminate a Command Sequence in process should not be used to resolve timing conflicts.

385 A wait function may be incorporated by stalling the bus (Master retains the signal state on SDATA and SCLK until ready to continue). Although the wait function might not be typically needed in a single Master system, it might be useful in cases where scheduling necessitates stalling the bus.

7.3 Additional Configurations

7.3.1 Multiple Logical Slaves in one Device

386 A single device may have more than one logical Slave. Each logical Slave needs its own USID. Thus, using multiple, logical Slaves in one device should be considered carefully.

387 Figure 44 illustrates an example case of two logical Slaves in one device. Device N incorporates both Slave N and N+1. Accessing device N would thus be done either through Slave N, N+1 or both. The device could consist of two physical dies (modules) molded into one package. In this particular case, each Slave could be considered an ordinary single device Slave with the difference that in the logical Slave case the VIO, SCLK and SDATA pins could be shared.

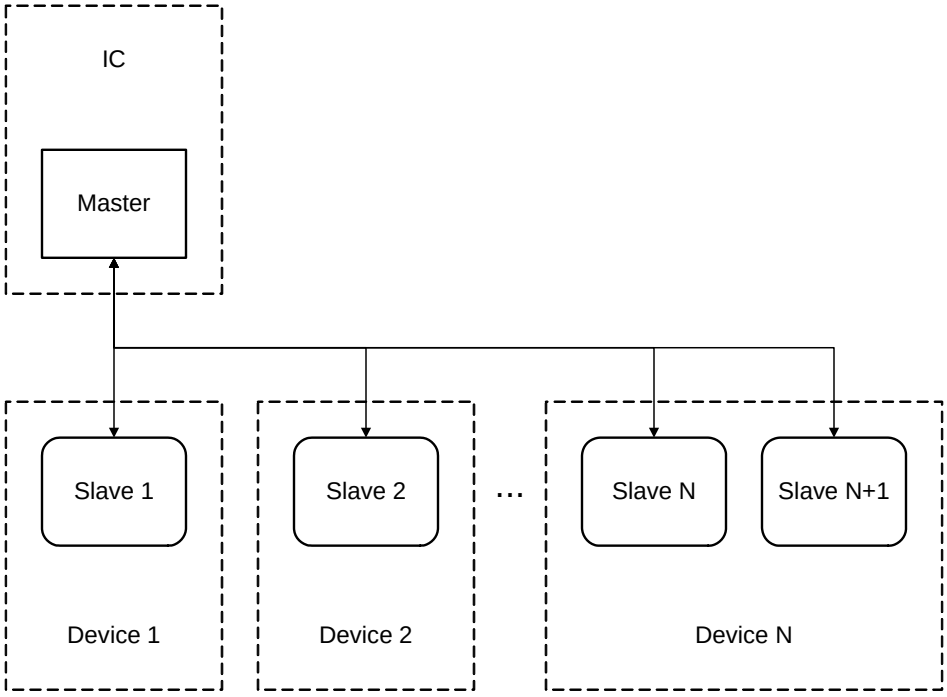

Figure 44 Example of Multiple Logical Slaves in Single Device Configuration

7.4 DDB in RFFE

388 *MIPI Alliance Specification for Device Descriptor Block (DDB)* [MIPI05] is defined for transfer of description and configuration data between devices on an interconnect. An RFFE bus is a interconnect with the Master and Slaves representing devices. An RFFE device shall thus be DDB compliant.

389 An RFFE device is a simple device, therefore only needing compliance with DDB Level 1. The response to a DDB Level 1 request is an 80-bit field, which needs many gates to support. Thus, a DDB request is not passed to Slaves.

390 In an RFFE implementation, the Master shall construct the full DDB Level 1 field for every requested device. In order to do this, the Master shall first fetch the MANUFACTURER_ID and PRODUCT_ID from the requested Slave using ordinary RFFE Read Command Sequences (see Section 6.7). Using the MANUFACTURER_ID and PRODUCT_ID the Master shall build the corresponding complete DDB block as defined in Table 23.

Table 23 Building DDB Level 1 Block

DDB Field	Size (bits)	RFFE Master Outputs
Revision	8	0b0001 0000 (DDB version 1.0)
Level	8	0b0000 0001 (DDB Level 1 only)
DeviceClass	16	0b0000 0000 0000 0000 (Unless Device ID is specified by MIPI Alliance and known by the Master, in which case that value shall be used)
ManufacturerID	16	0b0000 00mm mmmm mmmm (mm mmmm mmmm being the 10-bit MANUFACTURER_ID read from the Slave)
ProductID	16	0b0000 0000 pppp pppp (pppp pppp being the 8-bit PRODUCT_ID read from the Slave)
Length	16	0b0000 0000 0000 0000 (Not used)

391 The implementation of the DDB block builder in the Master is vendor-specific and may be accomplished in hardware, software, or some combination of the two.

392 Note that interfacing of the Master to the next hierarchal level (either in HW or SW) is out of the scope of the RFFE Specification. Thus it is assumed that such a connection exists, and that the Master has the capability of providing DDB Level 1 data over that connection by whatever means are specified by device implementers.

Annex A Additional Information for RFFE Reference (Informative)

A.1 Optional Signals

393 Discrete signals may be added by the system integrator to provide optional functionality outside the scope of the RFFE Specification.

A.1.1 Optional Reset Signal

394 A Reset signal may be implemented to allow sharing VIO with components not utilizing the RFFE bus. The Reset signal overriding VIO in this case thus provides the same reset functionality as VIO (see Section 5.2.2). Reset by VIO still applies for RFFE components independent of the state of this optional Reset signal.

A.2 Estimation of RFFE Bus Load Ranges for Various Configurations

395 It is useful to consider estimated bus loads for various RFFE bus configurations in order to analyze the trade-offs of different architectures. It is also useful for IC device manufacturers to consider the environments where their devices might be utilized, so as to provide devices well-suited for the maximum range of applications.

396 This section presents a method to calculate the bus load for various RFFE configurations, which may be used to achieve a first-order estimate. For this estimate, only capacitance is considered as a factor in the impedances that devices on the bus will experience. The load capacitance is a major factor in determining the timing characteristics that may be realized in the final configuration. Note that more exact values should be used where they are available, in order to achieve the most accurate assessment of RFFE bus timing, as well as other electrical requirements of the RFFE Specification.

397 The first step is to calculate an estimate of the total load due to the package capacitance of all devices on the bus, $C_{L(PKG)}$. In these calculations each Slave is assumed to contribute 2 pF of total pin capacitance (package plus device). The load presented by the driving device (Master) should also be included in the calculation, and here is also assumed to be 2 pF. Since an RFFE bus is a single-Master system, this estimate is given by the following general equation:

398 $C_{L(PKG)} = (n + 1) \times C(est)$, where n is the # of Slaves and C(est) is per device

399 A four-Slave system configuration with package capacitance of 2 pF per Slave device (and Master) gives the following result:

400 $C_{L(PKG)} = 5 \text{ devices} \times 2 \text{ pF / device} = 10 \text{ pF}$

401 An eight-Slave system configuration with package pin capacitances of 2 pF per Slave device and 2 pF for the Master then gives the following result:

402 $C_{L(PKG)} = 9 \text{ devices} \times 2 \text{ pF / device} = 18 \text{ pF}$

403 A maximally-configured RFFE system, with fifteen Slaves and one Master, each with a package capacitance of 2 pF per device gives the following result:

404 $C_{L(PKG)} = 16 \text{ devices} \times 2 \text{ pF / device} = 32 \text{ pF}$

405 Note that where differing values are known for specific devices that are to be used in an RFFE system, the pin capacitance values of the Master and the pin capacitance for each of the Slave devices may be individually summed to achieve a more accurate estimate for the total pin capacitance value of the proposed system.

406 The next step is to calculate an estimate of the capacitive load due to PCB interconnect, $C_{L(INT)}$. The estimate used here is that the PCB interconnect contributes 4 pF for up to four Slaves connected to a single Master, with an additional 1 pF for each additional Slave. $C_{L(INT)}$ is expressed with the following equation:

407 $$C_{L(INT)} = 4\ pF + (n - 4) \times 1\ pF$$

408 Thus, for a four-Slave system, the capacitive load due to the estimated interconnect is 4 pF.

409 For an eight-Slave system, the estimation for interconnect load is given by the following equation:

410 $$C_{L(INT)} = 4\ pF + (8 - 4) \times 1\ pF = 8\ pF$$

411 Similarly, for a fifteen-Slave system, the estimation for interconnect load is given by the following equation:

412 $$C_{L(INT)} = 4\ pF + (15 - 4) \times 1\ pF = 15\ pF$$

413 As with pin capacitance, more accurate values for the interconnect may be substituted if they are known. More accurate estimates may be achieved in cases where factors such PCB material characteristics, layer stack-up, layout style, and other variables specific to a given situation are available.

414 The final step is to calculate the total load, $C_{L(TOT)}$, which is the sum of all of the package loads and the estimated interconnect load, as is shown in the following equation:

415 $$C_{L(TOT)} = C_{L(PKG)} + C_{L(INT)}$$

416 Thus, for the eight-Slave example, the total load is given by the following equation:

417 $$C_{L(TOT)} = 18\ pF + 8\ pF = 26\ pF$$

418 A fully-configured RFFE bus, with fifteen Slaves, using the estimated values assumed here, would result in a total load as shown by the following equation:

419 $$C_{L(TOT)} = 32\ pF + 15\ pF = 47\ pF$$

420 Device manufacturers and system integrators are encouraged to utilize more accurate values for these parameters if/when they are available. Nonetheless, it is useful to be able to utilize this type of quick estimation in order to be able to make trade-offs in potential RFFE configurations, particularly with regards to the decision to implement single or multiple RFFE buses, when the number of Slaves to be controlled becomes larger. Segmentation of various devices to separate buses might allow for easier realization of certain characteristics to be achieved for selected devices in an RFFE system. It might also be useful in systems with a large number of devices to provide separate buses to help ensure that all devices achieve the highest speed and timing reliability.

421 As may be seen from these calculations, the actual values used for device pin capacitances, and the additional interconnect load contribution from the board environment utilized, can have a substantial impact on the total RFFE bus load. These contributions will typically have to be minimized for maximally-configured RFFE bus implementations. Although the RFFE Specification provides for buses encompassing a large number of components, there are performance constraints that might require additional considerations. IC providers and systems integrators thus need to perform trade-offs to maximize bus performance, while at the same time minimizing additional pins and device interconnections. Such factors may actually result in a choice to deploy multiple RFFE buses, where a single bus might otherwise be possible. However, bus loading is only one reason why such a decision might be made, since other performance factors could also influence this assessment.

Annex B Differences between SPMI and RFFE Specifications

422 *MIPI Alliance Specification for RF Front-End Control Interface* is based on *MIPI Alliance Specification for System Power Management (SPMI)* [MIPI03]. While the intention of the RFFE and SPMI Working Groups has been to retain as much commonality and interoperability between the SPMI and RFFE bus Specifications as possible, there are notable differences. This Annex endeavors to summarize the differences that exist between them in a simplified fashion.

B.1 Application Environment

423 The RFFE bus is intended for connecting a Radio-Frequency IC (RFIC) to the Front-End Modules (FEMs) associated with it in a system. This interface has to support all existing 3GPP standards as well as other radio standards. This means that very tightly controlled and predictable real-time control is required. At the same time, semiconductor processes used to implement an FEM device as well as realistic assumptions about cost factors mean that an RFFE Slave has extremely tight implementation logic gate count restrictions, surpassing those for any other MIPI Alliance bus application environment.

424 For these reasons the RFFE specification is a feature-limited version of the SPMI specification retaining only those functions and features that are absolutely necessary in this application.

B.2 Detailed Differences between SPMI and RFFE Specifications

B.2.1 Single Master Operation with Simple Slave Devices.

425 An RFFE bus has only one Master, and does not allow the use of Request Capable Slaves (RCS) on the bus. This means that the sole Master is the only device which may provide the clock on the bus, and that all Command Sequences on the bus are initiated by the Master. The Master is assumed to be the RFIC (see Section 4.1).

B.2.2 No Bus Arbitration

426 Single Master operation means that no bus arbitration is needed for RFFE. Master starts all communications sequences with the Sequence Start Condition, and once the sequence is over the Master may initiate another sequence without any intermediate bus arbitration (see Section 4.2)

B.2.3 No Bus Arbitration Requests by Slave Devices

427 Since no RCS devices are allowed on the RFFE bus, there are no bus arbitration requests from RCS devices (see Section 4.1).

B.2.4 Limited Command Sequence Set

428 The RFFE Command Sequence set is limited to Register 0 Write, Register Read, Register Write, Extended Register Read, Extended Register Write, Extended Register Read Long and Extended Write Long Command Sequences. The SPMI specification defines additional Command Sequences, but these are not used by RFFE devices (see Table 12).

429 RFFE supports a limited number of SPMI Command Sequences, as many of the SPMI Command Sequences are not applicable for a configuration in which only one Bus Owner Master may exist on the bus.

430 Table 24 lists the Master and Slave supported SPMI Command Sequences.

Table 24 RFFE Supported Command Sequences

RFFE[1]		SPMI Command	
Master	**Slave**	**Hex**	**Description**
Y	O	00 to 0F	Extended Register Write
N	N	10	Reset
N	N	11	Sleep
N	N	12	Shutdown
N	N	13	Wakeup
N	N	14	Authenticate
N/A	N/A	15	Master Read
N/A	N/A	16	Master Write
		17	Reserved
		18	Reserved
		19	Reserved
N/A	N/A	1A	Transfer Bus Ownership
N/A	N/A	1B	Device Descriptor Block Master Read
N	N	1C	Device Descriptor Block Slave Read
		1D	Reserved
		1E	Reserved
		1F	Reserved
Y	O	20 to 2F	Extended Register Read
O	O	30 to 37	Extended Register Write Long
O	O	38 to 3F	Extended Register Read Long
Y	O	40 to 5F	Register Write
Y	O	60 to 7F	Register Read
Y	O	80 to FF	Register 0 Write

1. Y = Supported, N = Not Supported, O = Optional Support, N/A = Not Applicable

B.2.5 Limited Support for Device Descriptor Block Data

431 The RFFE specification does not include SPMI Command Sequences supporting DDB data. Instead the RFFE specification specifies which registers RFFE devices hold the equivalent data and assumes any DDB data requests are handled by reading this data using normal Read Command Sequences and collecting the DDB data in the Master for further use (see Section 6.9 and Section 7.4).

B.2.6 Unified Slave Register Space

432 Slave register space is uniform and linear with all Register Read and Register Write Command Sequences pointing to a single linear register address space. SPMI base register address space and extended register address space are separate (see Section 6.6 and Section B.4).

B.2.7 Pull-Down Function in Master Device

433 In RFFE applications SDATA and SCLK pull-down functionality is implemented in the Master, or in a Master-associated external device, while in SPMI the bus pull-down is implemented in Slave devices, or in a Slave-associated external device. The bus pull-down realization in RFFE and SPMI is functionally equivalent; however, it is possible that in RFFE applications utilizing multiple SPMI Slaves that pull-down currents could exceed normal expectations in extreme cases. This situation may be avoided if SPMI Slaves used in RFFE applications possess pull-down devices which may be disabled. See Section 5.1 for more information about bus pull-down functionality in RFFE.

B.2.8 I/O Drive Strength Classes

434 Both SPMI and RFFE specifications specify bus timing for maximum 50 pF load per signal line. Additionally, the RFFE specification allows bus drive capability to be vendor-specific.

B.2.9 High Speed Devices Only

435 An RFFE device supports bus operation up to 26 MHz. There is no low speed device class as in SPMI (see Section 4.2)

B.2.10 Explicit Support for Half Speed Read

436 While also allowed within the SPMI specification, the RFFE specification explicitly explains and specifies Half Speed read operation timing for a Slave that cannot respond at Full Speed to register read operations (see Section 4.2).

B.2.11 EMI

437 The RFFE specification places additional requirements on the signaling waveforms for EMI reduction reasons; the rise and fall times in the SPMI specification are defined differently. A Master designed for SPMI might not necessarily meet the slew rate requirements of the RFFE specification.

B.2.12 Slave Identifier Allocation

438 The RFFE specification recommends allocation of Slave identifier number for different types of FEM devices in Table 13. The SPMI specification does not specify or recommend such values (see Table 13 and Section 6.8.1).

B.2.13 Register Allocation

439 Registers 0x001C to 0x001F have predefined functions. Register 0x001C controls trigger control and power modes, while registers 0x001D to 0x001F are related to DDB Level 1 support as well as programmable USID values for Slaves.

440 The SPMI specification does not specify device register usage. The specification does not explicitly describe a method to implement a programmable USID, although the same method as described in the RFFE specification (see Section 6.8.1) could be used as it falls outside the scope of the SPMI specification.

B.2.14 VIO Sensing for State Control

441 Instead of using external ENABLE, RESETN and VDDOK signals for power up control, sequencing and reporting like SPMI, RFFE controls Slave power-up sequencing and reset with the IO voltage pin. This alters the state control for STARTUP, ACTIVE, POWER DOWN and SHUTDOWN states.

B.3 SPMI Baseline and Compatibility to RFFE

442 The objective of this section is to compare in detail all features from SPMI and RFFE specifications summarized in Table 25. An all-encompassing mapping between SPMI and RFFE specification and features is listed with proper references to the sections in the respective specifications and comments pointing out which modifications i.e. reduction of the SPMI feature set have been applied.

Table 25 SPMI Feature Compatibility Matrix

SPMI Feature	SPMI Section	Comment	RFFE Section
Physical Interface	5		5
I/O Structures	5.1		5.1
I/O Configuration with Multiple Slaves and Masters	5.1.1		5.1.2
I/O Voltage and Logic Levels	5.2		5.2
Signaling Voltages	5.2.1		5.1.1
SPMI I/O Voltage Supply	5.2.2	Superseded by Section 5.2	5.2
Device Classes	5.3	RFFE does not implement device speed classes. Only high speed devices exist.	4.2
SPMI Clock (SCLK)	5.4		4.2.1
Electrical Specifications for the Master SCLK Driver	5.4.1		4.2.1.1
Electrical Specifications for SCLK Input	5.4.2		4.2.1.2
SPMI Data (SDATA)	5.5		4.2.2
Electrical Specifications for the SDATA Driver	5.5.1		4.2.2.1
Electrical Specifications for the SDATA Receiver	5.5.2		4.2.2.2
Device Characterization	5.6		5.3
Electromagnetic Interference	5.7		5.4
SPMI Constructs	6		6
Bit Ordering	6.1		6.1
Command Sequences	6.2		6.2
Bus Arbitration	6.2.1	RFFE is single Master only. There is no bus arbitration.	-
Sequence Start Condition	6.2.2		6.2.1
Frames	6.2.3		6.2.2
Parity Bit	6.3		6.2.3
Bus Park Cycle	6.4		6.2.4
Device Enumeration	7		6.8
Unique Slave Identifier	7.2		6.8.2
Group Slave Identifier	7.3		6.8.3
Master Connecting on the Bus	9.2	RFFE Master can turn on and off as necessary.	-

Table 25 SPMI Feature Compatibility Matrix (continued)

SPMI Feature	SPMI Section	Comment	RFFE Section
Connecting by Detecting SSC	9.2.1		-
Connecting by Detecting Bus Idle	9.2.2		-
Connecting by Detecting Bus Arbitration	9.2.3		-
Bus Initialization	9.3	Not needed in RFFE, single Master	-
Disconnecting from the Bus	9.4	Not needed in RFFE, single Master	-
Bus Monitoring by Disconnected Master	9.4.1	Not needed in RFFE, single Master	-
Slave Communication Request	10	Not used in RFFE	-
Alert bit on an Initialized Bus	10.1	Not used in RFFE	-
Slave Request Bit on an Initialized Bus	10.2	Not used in RFFE	-
Slave Communication Request on Uninitialized Bus	10.3	Not used in RFFE	-
Master Operating States	11.1	Not defined in RFFE	-
SPMI Slave Requirements	12		4.3
Register Spaces	12.1	RFFE uses linear, unified register space	6.6
Slave External Control Signals	12.2		A.1
RESETN	12.2.1	Optional in RFFE	A.1.1
ENABLE	12.2.2	Not defined in RFFE	-
PWROK	12.2.3	Not defined in RFFE	-
Exceptional Control Inputs	12.2.4		4.3.5
Other Control Inputs and Outputs	12.2.5		A.1
Multiple Logical Slaves on a Single Physical Device	12.2.6		7.3.1
Slave Operating States	12.3		4.3
STARTUP	12.3.1		4.3.1
ACTIVE	12.3.2		4.3.2
SLEEP	12.3.3	SLEEP is called LOW POWER in RFFE.	4.3.3
SHUTDOWN	12.3.4		4.3.4
Exceptional State Transitions	12.3.5		4.3.5
Optional Command Sequence Support	12.4	Not needed in RFFE due to single Master operation	-
Command Sequences	13		6.7
Command Sequence Summary	13.1		6.7.1
Command Sequence Descriptions	13.2		6.7.2
Extended Register Write Command Sequence	13.2.1		6.7.2.1
Extended Register Read Command Sequence	13.2.2		6.7.2.2

Table 25 SPMI Feature Compatibility Matrix (continued)

SPMI Feature	SPMI Section	Comment	RFFE Section
Reset, Sleep, Shutdown, Wakeup Command Sequences	13.2.3	Not used in RFFE.	-
Authentication Command Sequence	13.2.4	Not used in RFFE.	-
Device Descriptor Block Slave Read Command Sequence	13.2.8	Different DDB support in RFFE.	7.4
Device Descriptor Block Master Read Command Sequence	13.2.9	Different DDB support in RFFE.	7.4
Extended Register Write Long Command Sequence	13.2.10		6.7.2.3
Extended Register Read Long Command Sequence	13.2.11		6.7.2.4
Register Write Command Sequence	13.2.12		6.7.2.5
Register Read Command Sequence	13.2.13		6.7.2.6
Register 0 Write Command Sequence	13.2.14		6.7.2.7
Error Handling	13.3		6.2.3.1
Parity Error in the Command Frame	13.3.1	RFFE Table 11	6.2.3.1
Unsupported Command Sequences	13.3.2	RFFE Table 11	6.2.3.1
Parity Error in the Address Frame	13.3.3	RFFE Table 11	6.2.3.1
Parity Error in the Data Frame	13.3.4	RFFE Table 11	6.2.3.1
Unsupported Address	13.3.5	RFFE Table 11	6.2.3.1
SPMI Reference (informative)	Annex A		–
High Speed (HS) Device Class	Annex A.2.1	No device classes in RFFE.	–
Low Speed (LS) Device Class	Annex A.2.2	No device classes in RFFE.	–
Other Signaling Electrical Specifications	Annex A.2.3		–
Command Sequence Reference (informative)	Annex A.3		6.4

B.4 SPMI Register Space

443 The SPMI register space is divided into two register spaces, Base and Extended, as illustrated in Figure 45. The RFFE register space merges the two SPMI register spaces into a single register space. See Section 6.6 for further details regarding the RFFE register space.

444 The SPMI Base register space consists of up to thirty-two 8-bit registers. These registers may be accessed one byte at a time using only Register Write, Register Read and Register 0 Write Command Sequences.

445 The SPMI Extended and Extended Long register space consists of up to 65536 8-bit registers. These registers may be accessed using only Extended Register Write, Extended Register Read, Extended Register Write Long and Extended Register Read Long Command Sequences with single-byte or multi-byte access. Extended Command Sequences may be used for accessing registers 0 to 255, while Extended Long Command Sequences may be used for accessing registers 0 to 65535. Extended and Extended Long registers overlap the same registers.

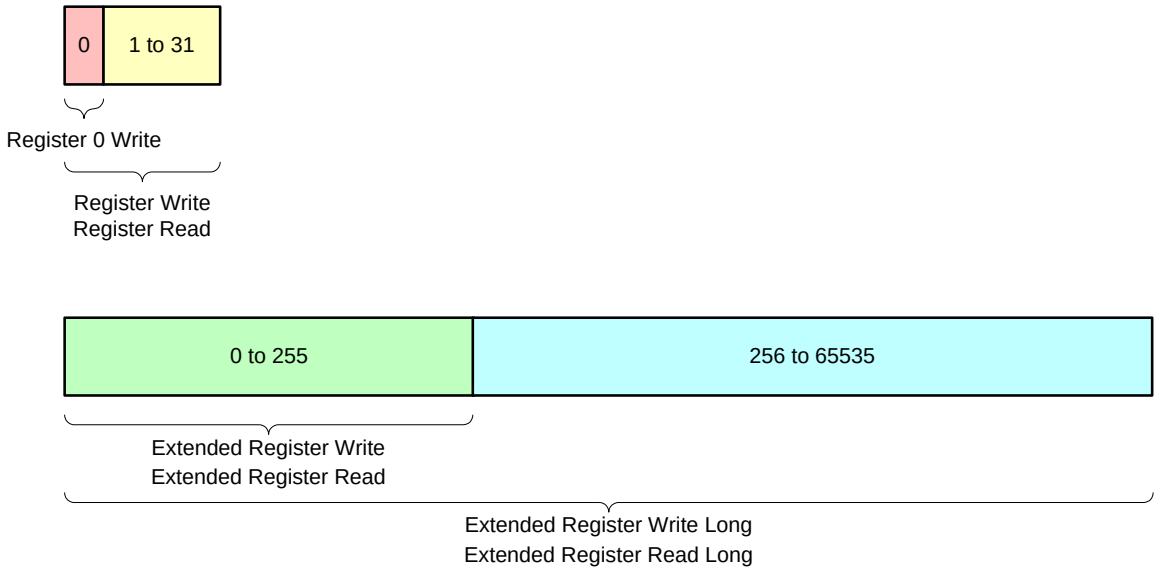

Figure 45 Slave Register Spaces, SPMI

Participants

446 The following list of individuals contributed to the development of this document, and consented to appear on this list. For a listing of all individuals who contributed to the original RFFE v1.00.00 release, see *MIPI Alliance Specification for RF Front-End Control Interface - Version 1.00.00 – 3 May 2010.*

Anhofer, Robert - IEEE-ISTO (staff)
Arrakoski, Jori – Nokia Corporation
Calvo, Carlos – Analog Devices Inc.
Chlopek, Dan – Fujitsu Limited
Gruber, George – Qualcomm Incorporated
Hallberg, Peter – ST-Ericsson
Hueber, Gernot – NXP Semiconductors
Keese, Liam – National Semiconductor
Ngo, Chris – RF Micro Devices
Osman, Saleh – Panasonic Corporation
Pennanen, Juha – National Semiconductor
Podsiadlo, Dave – Peregrine Semiconductor
Ross, Jim – Skyworks Solutions, Inc.
Seth, Marten – WiSpry Inc.
Soni, Samir – Fujitsu Limited
Wilkerson, Victor – MIPI Alliance (staff)