MIPI RFFE Interface for Wi-Fi® / Bluetooth® Technology eFEMs

White Paper

Version 1.0

12 September 2016

MIPI Board Approved for Public Distribution 29-Sep-2016

Contents

1 Technical Content

This White Paper summarizes a method for using the MIPI RFFE serial communication interface to control external/peripheral RF components in Wi-Fi® (up to 2x2 dual-band) and Bluetooth® enabled products. Due to the fact that serial communication adds an overhead time for each configuration, the suggested Master-Slave mapping helps minimize the number of telegrams sent for SISO, MIMO, and Concurrent Dual Band (CDB) cases, while complying with the MIPI RFFE Specification ***[MIPI01]***.

1.1 Introduction

The pursuit of high data rates and increased performance for Wi-Fi® forces strict requirements on both TX and RX, in order to compete in the high-end market segment. The current trend is to suggest proliferations that incorporate external Front End (eFEM) components, in order to achieve those high-end requirements.

The eFEM controls use discrete logic GPIOs, and with the increase in external units, multi-band support, and functionality, we see resulting increases in package pinout and routing complexity. These in turn affect cost and Form Factor (FF). For example, current 2x2 Wi-Fi®/Bluetooth® technology solutions implement 13 discrete control pins and 4 analog pins (17 pins in total), for supporting up to 4 eFEMs: two for Wi-Fi® Low Band (LB) and Bluetooth® technology, and two for Wi-Fi® High Band (HB) (see ***Figure 1***).

This White Paper summarizes a method for using the MIPI RFFE serial communication interface to control external/peripheral RF components for Wi-Fi® (up to 2x2 dual-band) and Bluetooth® enabled products. Due to the fact that serial communication adds an overhead time for each configuration, the suggested Master-Slave mapping helps minimize the number of telegrams for SISO, MIMO and Concurrent Dual Band (CDB) cases, while complying with the MIPI RFFE Specification.

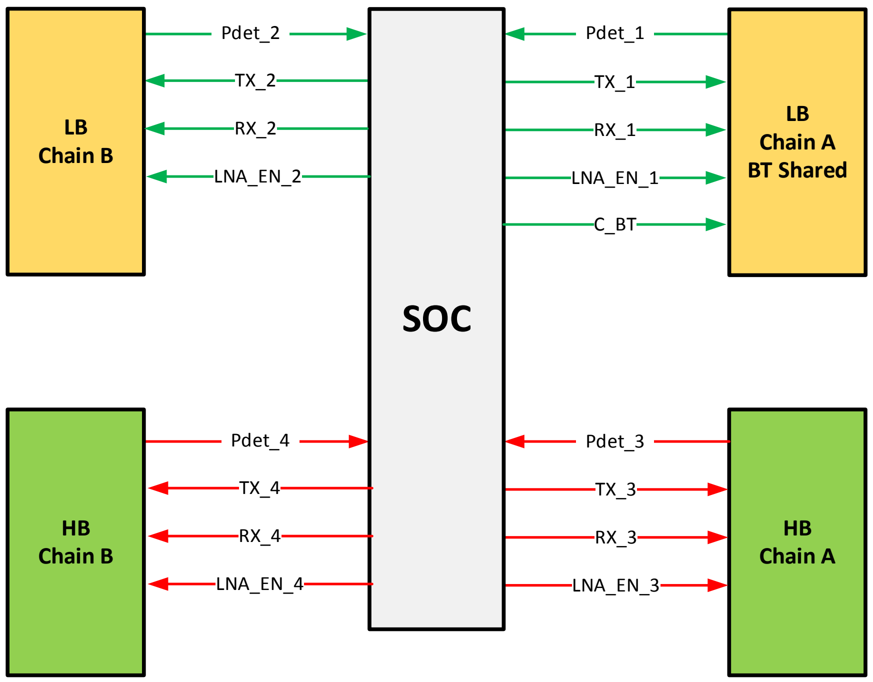


Figure 1 SOC Connected via Discrete Logic to External 2x2 Wi-Fi®/Bluetooth® Front-End Components (17 Pins)

1.2 Adopting the MIPI RFFE Specification

Adopting MIPI RFFE serial communication allows connection of up to 15 external components to the SOC by sharing just three control pins: VIO, SDATA, and SCLK. All the eFEMs are connected in parallel to these three pins, reducing the number of pins needed to control them. For example, MIPI RFFE reduces the number of SOC pins required to support 2x2 Wi-Fi®/Bluetooth® technology from as many as 17 pins with current discrete logic methods (see ***Figure 1***), to as few as 3 pins (see ***Figure 2***).

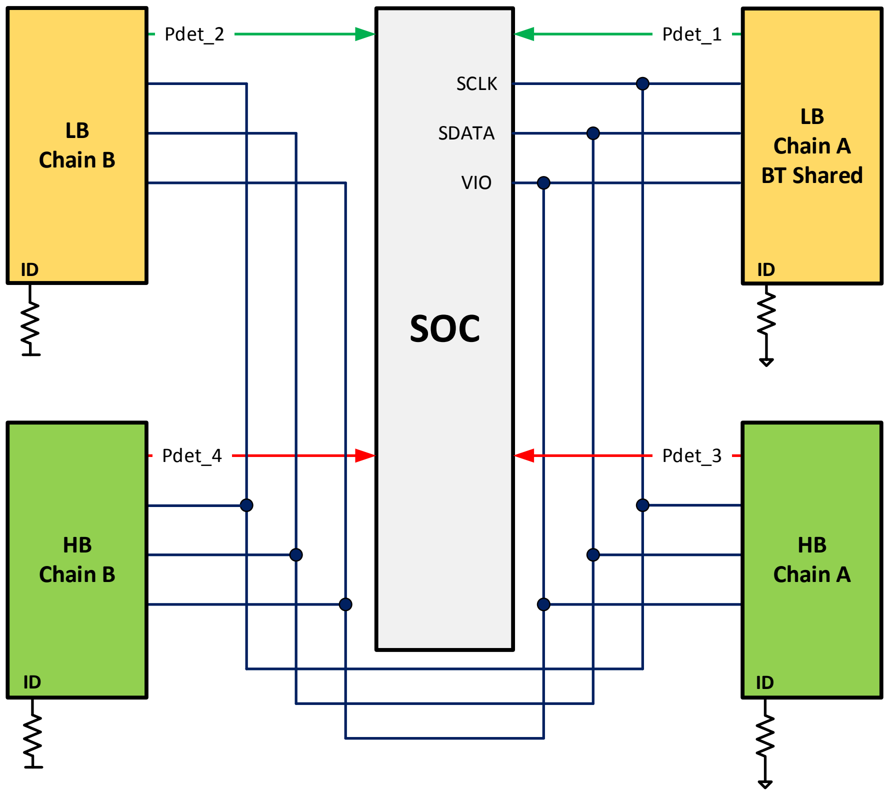


Figure 2 SOC Connected via MIPI RFFE Serial Communication to External 2x2 Wi-Fi®/Bluetooth® Front-End Components (3 to 7 Pins)

Note:

ID Port is an optional solution for identical Unique Slave Identification (USID) between two FEMs having the same P/N.

RFFE is an emerging Specification ***[MIPI01]***, developed by the MIPI Alliance to offer a common, widespread method for controlling RF Front-End designs. The MIPI Alliance determined that the majority of today's RF communication standards are proprietary or de-facto and are not utilized industry-wide, and developed the MIPI RFFE Specification in order to establish a single standard for eFEM communication, from the PHY level to the protocol and software levels.

Today, eFEMs targeting the Wi-Fi®/Bluetooth® technology segment do not yet support MIPI RFFE serial communication; instead, the control scheme is dominated by discrete control signaling. However we expect the drive for an SOC solution optimized for both form factor, pin count, and routing complexity will push eFEM vendors to introduce MIPI RFFE in this segment as well.

The MIPI RFFE Specification defines Read/Write command sequences with five address bits, for a space of thirty-two addressable Registers. All Registers contain one byte of data. The MIPI RFFE Specification divides the Register address space into Reserved Registers (which are used to store special modes, group triggers, and Slave/manufacture IDs), and User-Defined Registers (which enable Slave functionality). User-

Defined Register 0 can be accessed (in addition to normal Register read/write commands), with a special Register 0 Write command, allowing faster write telegrams to that address.

1.3 When Telegrams Pile Up…

As stated above, today the eFEMs are controlled by discrete signaling. eFEM vendors have consolidated common configurations for different eFEM system modes. ***Figure 3*** illustrates example eFEM modules for LB and HB. ***Table 1*** details the control configuration commonly used on LB and HB eFEMs, showing all system modes for LB and for HB.

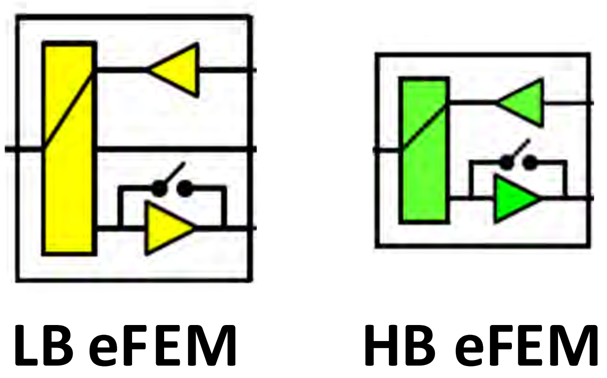


Figure 3 – Example LB and HB eFEMs

Table 1 LB and HB Slave Discrete Configurations and Their Operating Modes

LB FEMs

#	System Mode	Discrete Configuration			
		TX	RX	LNA_EN	C_BT
1	STBY	0	0	0	0
2	TX	1	0	0	0
3	RX Gain	0	1	1	0
4	RX Bypass	0	1	0	0
5	BT	0	0	0	1

HB FEMs

#	System Mode	Discrete Configuration		
		TX	RX	LNA_EN
1	STBY	0	0	0
2	TX	1	0	0
3	RX Gain	0	1	1
4	RX Bypass	0	1	0

Before adopting MIPI RFFE, we first need to agree on address mapping for the eFEMs.

The immediate method for controlling the eFEM using the MIPI RFFE protocol will be by issuing a direct Register Write to each eFEM with the desired System Mode code. For example, in order to move HB Chain A eFEM to TX, we will write 0x08 to a pre-defined Register, and then the eFEM switching will be done according to the transmitted System Mode code. Unfortunately, using this method is inefficient since it requires a dedicated transaction for each eFEM module. Moreover, only three or four bits (out of eight) are used in assigning the actual configuration on each Write transaction.

Another mapping scheme would suggest that Register 0 bits [6:0] would hold two coded words: a three-bit word for LB (encoding the five LB System Modes), and a two-bit word for HB (encoding the four HB System Modes). Using this allocation in broadcast mode, we can separately configure LB and HB eFEMs in a single telegram. However if we would like to configure two eFEM on the same band (i.e., two LB eFEMs, or two HB eFEMs), then we would need to issue two or more telegrams, which would in turn increase our turnaround overhead accordingly.

For example, let's assume 2x2 Wi-Fi®/Bluetooth® technology (i.e., a single chain is shared between Wi-Fi® and Bluetooth® devices), with both chains configured to RX Gain Mode, and that we would like to transition on one chain to Bluetooth® technology, and to transition the other chain to RX Bypass Mode. In this scenario it will be necessary to issue two consecutive telegrams, and also to resolve the priority conflicts between the two telegrams.

A more severe example scenario would be to change the state of both chains in MIMO mode upon AGC command, for example from RX Gain Mode to RX Bypass Mode for both chains. In this scenario, the serial communication overhead cost for two consecutive telegrams would be unacceptable.

In order to avoid such issues, we suggest using a different mapping that increases flexibility for issued telegrams, and ensures single-telegram transitions between all System Modes without imposing the need for a complex triggering-mechanisms implementation on the eFEM side. In the suggested mapping, it is only necessary to support the Register 0 (Reg0) Write, Register Write, and Broadcast capabilities.

1.4 The Proposed Solution

1.4.1 High Level Description

The proposed solution uses two Registers as control registers, and uses part of the MIPI RFFE Register address space for System Mode discrete configuration (which is uploaded upon initialization to the eFEM):

1. Register 0 is defined as the pointer for different system modes, i.e., Register 0 points to the relevant eFEM configuration previously stored to the eFEM (at initialization time). Seven Register 0 bits are divided into two pointers: four bits for LB pointer, and three bits for the HB pointer. This allows the controller to independently set different System Modes for the LB eFEMs and for the HB eFEMs (see ***Table 4***). In real time, we issue broadcast telegrams configuring all eFEMs to a given System Mode.
2. Register 1 enables or disables each eFEM, bringing the flexibility to enable any combination of the connected eFEM. Disabling Register 1 is required to also reset Register 0, in order to ensure that when the eFEM is enabled, it will point to a valid pre-defined state. ***Table 5*** shows a suggestion for mapping the Register 1 Enable/Disable bits in a manner that allows for single-telegram configuration, by using broadcast.
3. The rest of the Registers span the LB and HB system modes, taking into account combinations of any subset of the eFEMs, and all the possible Wi-Fi®– Bluetooth® technology coexistence modes. Assuming we keep the same notation used for discrete control signaling, ***Table 6*** shows a detailed mapping of the four Slaves.

   At first glance we can see that some configurations are identical between the LB eFEMs and the HB eFEMs, but a closer look reveals all the combinations of the two LB eFEMs and HB eFEMs. Note that the number of Registers needed for storing System Modes is different for LB eFEMs than it is for HB eFEMs, due to LB's need to support simultaneous modes for Bluetooth®. Moreover, since we store System Mode combinations, the number of bits also depends on the number of chains to be supported. For example, the case of 2x2 Wi-Fi®/Bluetooth® technology LB & HB requires 36 LB bits plus 18 HB bits, in order to represent all the System Modes.

In order to configure the eFEMs, we have to define the Operation Mode (Enable/Disable) in Register 1, and the System Mode (TX/RX Gain/RX Bypass/BT) in Register 0. Transitioning between chains, and/or transitioning between bands, requires two consecutive telegrams: the first one to set the Operation Mode

(Enable or Disable) for the appropriate eFEMs, and the second one to set its System Mode. But if we only need to change the eFEM System Mode, then we only need to send a single telegram (to transition to the next System Mode). Continuing the example used in the previous section, in order to transition the LB eFEMs to Bluetooth® mode and RX Bypass, we can send the appropriate configuration bits via a single Register 0 broadcast telegram.

Table 2 Register 0 Pointer Functionality Enabling Separate System Mode Assignment for HB vs. LB

Reg0	7	6	5	4	3	2	1	0
	Unused	HB Pointer			LB Pointer			

Table 3 Register 1 Enable/Disable Per Slave

Reg1	7	6	5	4	3	2	1	0
					HB ID1	HB ID0	LB ID1	LB ID0
					0 / 1	0 / 1	0 / 1	0 / 1

Note:

Identical Slaves Enable/Disable on different Register bits, as selected by the ID external pin

Table 4 – Slave Address Mapping with Different System Modes

Reg	Register Label	Slave 1 LB Chain A (BT Shared)	Slave 2 LB Chain B	Slave 3 HB Chain A	Slave 4 HB Chain B
		Assignment	Assignment	Assignment	Assignment
0	Pointer	LB Pointer	LB Pointer	HB Pointer	HB Pointer
1	Disable / Enable	Disable / Enable	Disable / Enable	Disable / Enable	Disable / Enable
2	System mode 0	STBY	STBY	STBY	STBY
3	System mode 1	Tx	Tx	Tx	Tx
4	System mode 2	Rx Gain	Rx Gain	Rx Gain	Rx Gain
5	System mode 3	Rx Gain	Rx Bypass	Rx Gain	Rx Bypass
6	System mode 4	Rx Bypass	Rx Gain	Rx Bypass	Rx Gain
7	System mode 5	Rx Bypass	Rx Bypass	Rx Bypass	Rx Bypass
8	System mode 6	BT	Rx Gain		
9	System mode 7	BT	Rx Bypass		
10	System mode 8	BT	Tx		

Note:

System Modes are loaded into the eFEM by the SOC upon initialization

1.4.2 MIPI RFFE Master/Slave Recommendations

The MIPI RFFE Specification defines Read/Write command sequences with five-bit addresses, thus spanning 32 addressable registers. All the registers hold one byte of data. The MIPI RFFE Specification divides the register address space into two parts:

- **Reserved Registers:** Used to store special modes, group triggers and Slave/manufacture IDs
- **User-Defined Registers:** For enabling the Slave's functionality. User-Defined Register 0 supports an additional (i.e., in addition to the ordinary register read/write commands) special 'Register 0 Write' command featuring faster write performance.

The MIPI RFFE core will support the MIPI RFFE v2.0 Specification baseline, with the following modifications/waivers:

- Single Master only. As a result, there is no need to support Multi-Master capabilities, Master hand-overs, or Master Write/Read telegrams.
- Support both Register 0 Write and normal Write, both with unicast and broadcast capabilities, and normal Read and Synchronous Read. Other extended Read/Write functions are not needed with the suggested mapping method.
- Read command is required to also support half clock rate
- The MIPI Core real time interface is required to transmit only single-telegram in RX Gain step mode, and in Wi-Fi®/Bluetooth® mode simultaneous eFEM transitions.

1.4.3 System Flows Outlined

This Section describes high level, general System Modes and the corresponding eFEM states.

1.4.3.1 SOC Initialization

Upon SOC initialization, the SOC firmware will trigger the MIPI RFFE Master side interface to program each iSlave with a different eFEM configuration. After configuring all the eFEMs, the eFEMs will remain in STBY mode.

The initialization flow is:

1. Initiate Register 1 broadcast, setting all Slaves to STBY mode
2. Initiate Write telegrams to Registers 2 through 10 (inclusive) of Slaves 1 through 4 (inclusive), with the pre-defined System Modes

Table 5 Slave Address Mapping After SOC Initialization

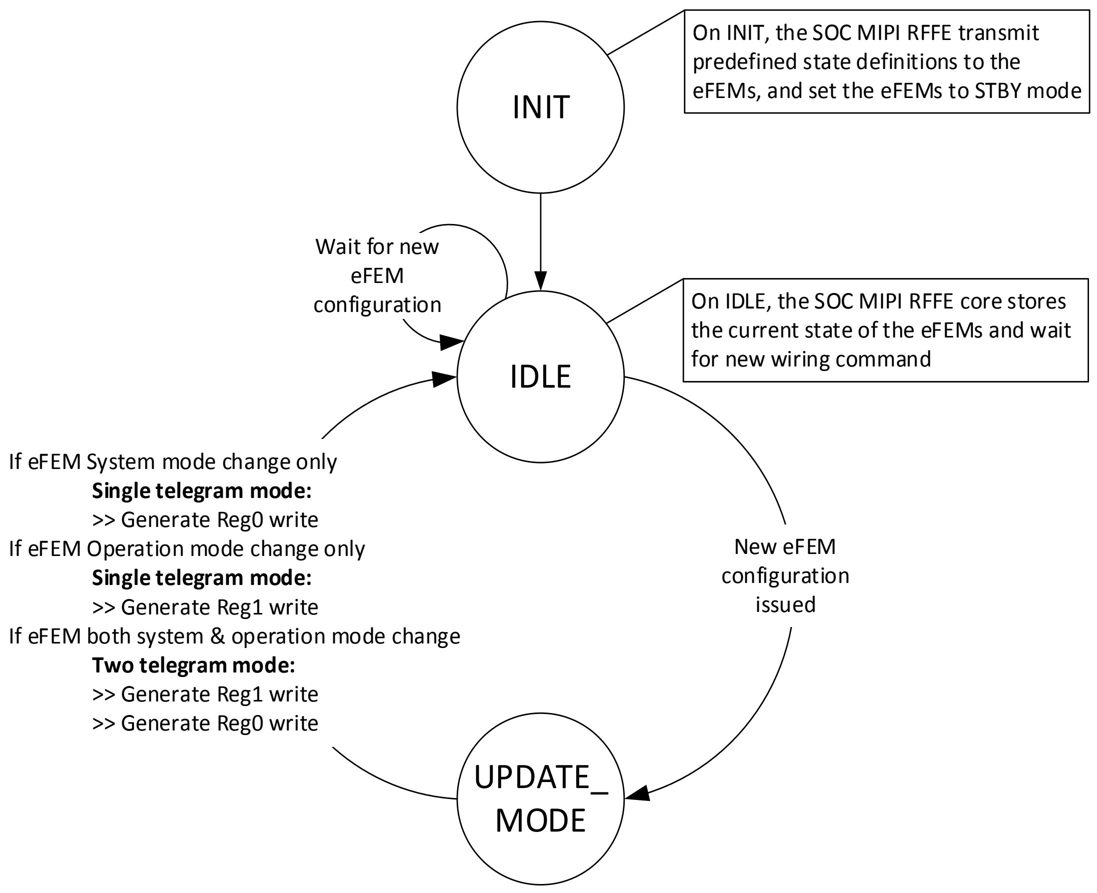
Reg	Register Label	Slave 1 LB Chain A (BT Shared)	Slave 2 LB Chain B	Slave 3 HB Chain A	Slave 4 HB Chain B
		Assignment	Assignment	Assignment	Assignment
0	Pointer	Sys. Mode 0	Sys. Mode 0	Sys. Mode 0	Sys. Mode 0
1	Disable / Enable	Disable	Disable	Disable	Disable
2	System mode 0	STBY	STBY	STBY	STBY
3	System mode 1	Tx	Tx	Tx	Tx
4	System mode 2	Rx Gain	Rx Gain	Rx Gain	Rx Gain
5	System mode 3	Rx Gain	Rx Bypass	Rx Gain	Rx Bypass
6	System mode 4	Rx Bypass	Rx Gain	Rx Bypass	Rx Gain
7	System mode 5	Rx Bypass	Rx Bypass	Rx Bypass	Rx Bypass
8	System mode 6	BT	Rx Gain		
9	System mode 7	BT	Rx Bypass		
10	System mode 8	BT	Tx		

1.4.3.2 Real Time

In real time, we generate MIPI RFFE telegrams according to the eFEM configuration command required for the desired system state: Wi-Fi® TX mode, Wi-Fi® RX mode, Wi-Fi®/Bluetooth® Simultaneous mode, or Bluetooth® Standalone mode. On the SOC side, we first store the current eFEM status, the Operation Modes, and the System Modes. After that, the SOC will trigger generation of one or more MIPI RFFE telegrams (as needed) whenever a new eFEM configuration command is issued.

A change triggering such RFFE telegrams can be either operational (i.e. caused by Enabling or Disabling any subset of the connected eFEMs), or it can be a change to the eFEM System Mode, or it can be a combination of the two change types:

1. If the eFEM changes only the System Mode and does not require a change of the eFEM Operation Mode (e.g., moving between RX Gain and RX Bypass), then the SOC will generate a single Register 0 write telegram in order to transition the eFEM to the new System Mode.
2. If the eFEM is requested to change only its Operation Mode (e.g., an eFEM transition from Active Mode to Bypass), then the SOC will generate a single telegram to either Enable or Disable the eFEM Operation Mode Register (i.e., a single Register 1 Write).
3. If the eFEM is required to transition both its System Mode and its Operation Mode (e.g., when transitioning between chains, or to transition from SISO operation to MIMO operation), then the SOC will generate two telegrams. The first telegram will either Enable or Disable the eFEM's Operation Mode Register (i.e., a Register 1 Write), and the second telegram will transition the eFEM to the desired System Mode.


INIT
On INIT, the SOC MIPI RFFE transmit predefined state definitions to the eFEMs, and set the eFEMs to STBY mode
Wait for new eFEM configuration
IDLE
On IDLE, the SOC MIPI RFFE core stores the current state of the eFEMs and wait for new wiring command
If eFEM System mode change only
Single telegram mode:
>> Generate Reg0 write
If eFEM Operation mode change only
Single telegram mode:
>> Generate Reg1 write
If eFEM both system & operation mode change
Two telegram mode:
>> Generate Reg1 write
>> Generate Reg0 write
New eFEM configuration issued
UPDATE_ MODE

161

Figure 4 System Flows High Level State Diagram

1.5 eFEM Slave Recommendations

This section describes the eFEM recommendations for supporting the suggested Master-Slave mapping. In this mapping, the eFEMs are RFFE bus Slaves and the SOC is the RFFE bus Master.

1.5.1 High Level Recommendations

As stated earlier, the Slave should support the MIPI RFFE v2.0 Specification, and may optionally take advantage of the waivers detailed in ***Section 1.4.2*** which permit the eFEM-side Slave interface to be considerably simplified.

The Slave must support:

1. SCLK rate up to 40MHz
2. Use of VIO to retain (as a memory) the Slave's pre-defined System Modes
3. Register Write/Read telegram – **Must also support broadcast**
4. Register 0 Write telegram – **Must also support broadcast**

   Note:

   Triggers and extended memory address span need not be supported
5. Register Mapping:
   - Register 0 will point to the System Mode that the SOC uploaded to the eFEM address range upon initialization
   - Register 1 will Enable (1) or Disable (0) the eFEM

     Note:

     When Disabled, Register 0 will overwrite and retain its value, to point to System Mode 0
     - Use of Register 1 bits S0 through S3 (inclusive) will be determined by the state of the ID external pin:
       - If pull-down: S0 is LB Enable/Disable, and S2 is HB Enable/Disable
       - If pull-up: S1 is LB Enable/Disable, and S3 is HB Enable/Disable
   - Extra bits to configure the System Modes upon SOC initialization:
     - 36 bits for LB: 4 bits per eFEM (on different Register address)
     - 18 bits for HB: 3 bits per eFEM (on different Register address)

Examples

The total number of Slave side bits needed to support a 2x2 product is:

- For LB with Bluetooth® technology, 42 bits:

  4 bits for Register 0 pointer + 2 bits for Register 1 Enable/Disable + 36 bits for System Mode configuration
- For HB, 23 bits:

  3 bits for Register 0 pointer + 2 bits for Register 1 Enable/Disable + 18 bits for System Mode configuration

1.5.2 Slave Switching Time Recommendations

MIPI RFFE serial communication adds latency in configuring the eFEMs, and this latency must be taken into account. As stated above, the suggested solution differentiates between a change to System Mode only, versus a simultaneous change to both System Mode and Operation Mode. Changing System Mode alone has stricter timing requirements, especially for RX Gain Step (i.e., changing from LNA Enable to LNA Bypass).

One more parameter added to the timing budget when using MIPI RFFE is the decoding time that the Slave needs to extract the desired configuration bits from the telegram. Since the decoder will most likely be implemented by combinational logic, decoding time should be negligible compared to the eFEM settling time.

The overall latency can be calculated as:

$$MIPI\ RFFE\ added\ latency = RFFE\ overhead + RFFE\ decoding + eFEM\ settling\ time$$

1.5.3 Slave Register Map Recommendations

Table 6 and Table 7 show the suggested Slave mapping for LB and HB eFEMs, respectively.

After the initialization procedure, the SOC writes the System Modes to the eFEM as detailed in these Tables. Therefore Register 2 through 10 (inclusive) on LB eFEMs, and Registers 2 through 7 (inclusive) on HB eFEMs, should be configurable.

This suggests two possible optimizations for eFEM devices intended to use the suggested RFFE Slave mapping:

- For faster-on devices, Registers 2 through 10 (inclusive) on LB eFEMs, and Registers 2 through 7 (inclusive) on HB eFEMs, could be designed to reset to the values shown in the Tables.
- For lower cost devices, Registers 2 through 10 (inclusive) on LB eFEMs, and Registers 2 through 7 (inclusive) on HB eFEMs, could be hard-coded to the values shown in the Tables.

Table 6 LB Detailed Slave Register Mapping After Initialization Procedure

Reg	LB eFEM								LB eFEM							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0					P3	P2	P1	P0					P3	P2	P1	P0
1							S1	S0							S1	S0
Label					CBT	TX	RX	LNAen					CBT	TX	RX	LNAen
2					0	0	0	0					0	0	0	0
3					0	1	0	0					0	1	0	0
4					0	0	1	1					0	0	1	1
5					0	0	1	1					0	0	1	0
6					0	0	1	0					0	0	1	1
7					0	0	1	0					0	0	1	0
8					1	0	0	0					0	0	1	1
9					1	0	0	0					0	0	1	0
10					1	0	0	0					0	1	0	0

Table 7 HB Detailed Slave Register Mapping After Initialization Procedure

Reg	HB eFEM								HB eFEM							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0		P2	P1	P0						P2	P1	P0				
1					S3	S2							S3	S2		
Label						TX	RX	LNAen						TX	RX	LNAen
2						0	0	0						0	0	0
3						1	0	0						1	0	0
4						0	1	1						0	1	1
5						0	1	1						0	1	0
6						0	1	0						0	1	1
7						0	1	0						0	1	0
8																
9																
10																

Table 8 Register 0 Pointer Value Configuration Mapping

Register 0 Value	System Mode	eFEM Configuration
0x0	0	Register 2
0x1	1	Register 3
0x2	2	Register 4
0x3	3	Register 5
0x4	4	Register 6
0x5	6	Register 7
0x6	7	Register 8
0x7	8	Register 9
0x8	9	Register 10

1.5.4 Slave Configuration Flow Recommendations

In order to ensure safe system operation, the Slave side should support the Slave flows described in this Section.

1.5.4.1 eFEM MIPI RFFE Power-Up Recommendations

Power-up is starting when VIO rises to the RFFE bus' operating voltage. After sensing VIO power up, the MIPI RFFE bus will initialize to its default state, i.e., all Slaves will be in STBY mode and disabled. At that time the address span used for the System Modes will be all zeroes, i.e. Register 0 will point to System Mode 0. The Slave side should support this flow.

1.5.4.2 SOC Initialization Recommendations

The eFEM will be accessed multiple times while the SOC configures the System Modes to the eFEMs address map. After completing initialization, the SOC will keep the eFEMs in STBY mode until either Bluetooth® mode or Wi-Fi® mode require access to the eFEM for TX/RX. The Slave side should support this flow.

1.5.4.3 Real Time Recommendations

The SOC will initiate telegram transmission, and will configure the Slaves to the desired System Modes and Operation Modes (see ***Section 1.4.3.2*** for more details). The Slave side should support this flow.

References

[MIPI01] *MIPI Alliance Specification for RF Front-End Control Interface (RFFE^{SM})*, Version 2.0, MIPI Alliance, Inc., 25 September 2014.