MediaTek

GSM/GPRS Layer 1 Programming Guide

Preliminary Information

Revision: 2.6

Release Date: Nov, 17, 2011

Revision History

Revision	Date	Author	Comments
0.1	09/07/2002	Shine Huang	First release
0.2	05/08/2002	Shine Huang	(1) Focus on MT6119C RF module control (2) Combine sectors timing setting, serial interface, parallel interface, synthesizer N-counter evaluation, transceiver gain setting evaluation into sector RF module control. (3) Modified description of each sector.
0.3	06/08/2002	Steven Yuan	Add the brief of sleep mode and acoustic performance optimization in section 1, and revise the description of RF module connection in section 2.1
0.4	09/08/2002	Shine Huang	Modified the description of each chapter by Steven's revise
0.5	09/08/2002	Steven Yuan	Merge section 2 overview and section 3 RF control, revise Parallel interface control of RX windows create in section 2.2.3, and introduce the file for RX ADC linear range configuration and acoustic FIR configuration
0.6	09/08/2002	Anthony Chin	Add section 8 for sleep mode optimization.
0.7	11/08/2002	Phil Hsieh	Add “Acoustic Device Optimization” section
0.8	13/08/2002	Phil Hsieh	Revise “Acoustic Device Optimization” section
0.9	17/08/2002	Steven Yuan	Add section 6 AFC configuration, and section 7 RX ADC dynamic range configuration
1.0	17/09/2002	Steven Yuan	Check in DMS
1.1	17/07/2003	Shine Huang	Update RF control, BFE, Path Loss, APC part for MT6205B. Add one section to describe the limitation of BSI and BPI events.
1.2	03/08/2003	Shine Huang	Add GPRS part in this document.
1.3	12/12/2003	Adam Tseng	Update “Acoustic Device Optimization” as “AFE Configuration” Add “L1 Audio Adaptation Layer” section.
1.4	01/06/2004	Adam Tseng	Update “L1 Audio Adaptation Layer” section.
1.5	01/13/2004	Steven Yuan	Update section 1 – introduction, section 2. section 8.2.2
2.0	01/13/2004	Steven Yuan	Update for check in DMS
2.1	01/20/2006	Barry Peng	Add PT2B and different APC DC Offset features in section 2.3.5 and section 6.4
2.2	02/28/2006	MJ Yang	Add section 2.9 “RF Control Configuration of MT6139C RF Module”
2.3	06/04/2007	James Fu & MJ Yang	Add section 2.10 “RF Control Configuration of Helios2(SKY74137) RF Module” Add section 2.11 “RF Control Configuration of MT6140 RF Module” Add section 6.3 “Table of EPSK Power Ramp Profile”

Revision	Date	Author	Comments
			Add section 6.5 "EGPRS Inter-Slot Power Ramp Profile" Add section 8.3.3 "The Setpoint settings for different mode"
2.4	12/10/2009	Weining Chien	Add section 2.12 "RF Control Configuration of AD6546 RF Module"
2.5	12/21/2009	Weining Chien	Add section 2.13 "RF Control Configuration of AD6548 RF Module"
2.6	29/03/2011	Sean Yang	Add section 2.14 "RF Control Configuration of MT6162 RF Module" Add section 3.3 "TDMA Timing Limitation in GSM mode for MT6162" Add section 3.4 "TDMA Timing Limitation in GPRS mode for MT6162"

Table of contents

1 Introduction

MediaTek MT62xx series GSM/GPRS baseband chips are designed for mobile phone or mobile terminal application, and can be designed with most popular RF chips on market and versatile peripheral device. In order to saving time and efforts when customer changes his RF module design and software optimization for different acoustic device or sleep mode performance, MAUI Layer 1 programming guide is documented to shorten the period of Layer 1 porting for customer's specific product. This document describes how to program the RF related code in Layer 1, additionally, it also describes the software configuration, which is designed to optimize the performance for acoustic device.

The RF and acoustic device related source files of Layer 1 and their functions are list as follow:

Source File	Description
l1d_custom_rf.h	RF related timing and data setting of RX/TX window
m12193.c	Default path gain loss and power ramp calibration data setting.
l1d_rf.h	BB related timing and data setting of RX/TX window
l1d_cid.h	Compile option translation for Layer 1 internal using
l1d_reg.h	BB register address definition
l1d_data.h	Constant definition of Layer 1
l1d_data.c	Global variable declaration used in whole Layer 1
m12190.h	Data type definition used for all RF-related code
m12191.c	Evaluation of synthesizer setting
m12192.c	Evaluation of transceiver gain setting
m12195.c	Serial data setting for RX/TX windows
m12196.c	Power on, power off setting
m12193.h	Default AFC calibration data and DSP set point setting
m12194.c	Receive path ADC linear range configuration
Audcoeff.c	The configuration table for Audio input and output FIR coefficient
l1sm_public.h	The public functions to control sleep mode operation.

The RF radio control in Layer1 can be classified into some parts:

(1) **Hardware connection of RF module and Baseband**: MT62xx baseband chip provides some useful units to control the RF module. Most control lines can be pin-to-pin connection.

(2) **TDMA timing setting**: MT62xx baseband chip is a baseband chip with TDMA event driven architecture. All control signals are pre-programmed in the previous TDMA frames. RF module can be controlled by sending serial command and by activating control pins from MT62xx. To properly control the RF module, which time to activate which signals should be planned. The event driven units of MT62xx, including RX ADC/TX DAC active timing, BSI (Baseband Serial Interface) events, BPI (Baseband Parallel Interface) events, AFC(Automatic Frequency Control) setting, APC(Automatic Power Control) setting. All timing setting is collected into file **l1d_custom_rf.h**.

(3) **Serial interface control**: Some RF chips can be commanded by receiving serial signals. 3 wires, i.e. device select, serial clock, and serial data, are needed to control the device. In MT62xx baseband chip, this interface named **BSI** (Baseband Serial Interface) is provided. The timing to activate BSI unit sending 3-wire control signals is called **BSI event**. The serial data sent with corresponding BSI event is call **BSI data**. The initialization by BSI interface is implemented in the file **m12196.c**. The timing setting of BSI

activation is defined in **l1d_custom_rf.h**. The BSI data setting with the corresponding BSI event is implemented in **m12195.c**.

(4) **Parallel interface control**: Some control signals of RF module can be controlled by GPO (General Purpose Output) pin from MT62xx. The timing of changing the states of these signals should be matched up the flow to create a RX/TX window. MT62xx baseband chip provides a unit named **BPI** (Baseband Serial Interface) that contains event driven signals to control the RF module. The timing of changing the states of BPI bus is called **BPI event**. The states of BPI signals with corresponding BPI event is call **BPI data**.

(5) **Synthesizer N-counter evaluation**: To lock the right operation frequency of synthesizer on RF module, the subroutine to evaluate the setting value of synthesizer N-counter should be implemented. The function named **L1D_RF_GetRxPLLSetting** is used to evaluate downlink synthesizer N-counter. The function named **L1D_RF_GetTxPLLSetting** is used to evaluate uplink synthesizer N-counter. These two functions are called by Layer1 and the result is stored in variable **l1d_rf.RFN_data** and **l1d_rf.IFN_data**.

(6) **Transceiver gain setting evaluation**: To set the right received amplifier gain of transceiver on RF module, the subroutine to evaluate transceiver gain setting should be implemented. The function named **L1D_RF_GetGainSetting** is used to evaluate transceiver gain setting. The result is summed up the path loss gain and stored in variable **l1d_rf.AGC_data**.

(7) **Baseband Front End setting**: For downlink, swapping IQ data and doubling RX ADC gain of Baseband are settable. For uplink, swapping IQ data, setting IQ swing, common-mode voltage, and trim are also settable.

(8) **Path loss gain calibration**: Gain loss from antenna to Baseband is variant and depends on the operation channel. So the path loss gain is needed to be calibrated and be compensated. This compensation is taken care by Layer 1 internal, so only the path loss calibration table needs to be established. This calibration data can be stored in flash of target by using tool META.

(9) **TX power ramp profile**: Different power ramp profiles for different transmit power level may be needed. Layer 1 supports 16 ramp profiles for each PCL of each band. The detail format of ramp profile is discussed in section 6. The power ramp profile data can be stored in flash of target by using tool META.

(10) **AFC calibration data**: AFC(Automatic Frequency Control) is dedicated to control the operation frequency of VCXO, which is the oscillator generating the system operation clock (13MHz for MT6205B chip),in order to synchronize MS operation frequency with network. Therefore, VCXO characteristic should be predicted during mass production. Section 7 will introduce their related configuration in MediaTek layer 1 code.

(11) **RX ADC set-point configuration**: RX ADC set-point should be set for different RF solution, the detail is listed in section 8.

This document follows above classification to describe how to program the RF-related code of Layer 1. Additionally, the descriptions for sleep mode and Audio Front End (AFE) configuration are included. Besides, the last section illustrate L1 audio adaptation layer, which is designed to provide an abstract API interface for advanced customers to integrate a 3rd-party audio chip with the MediaTek Base Band chip. MediaTek MT62xx reference evaluation board and MT6119C reference RF module (Fountain2) are taken as examples in the following sectors.

2 RF Control Configuration

The timing settings of BPI, BSI, and window related events are introduced in this section. All timing settings of these events are defined as aliases and collected in **l1d_custom_rf.h**. The unit of setting value is QB (quarter bit). 1 QB is equal to 0.923 us.

2.1 Overview

RF radio control is an important part of Layer1. RF-related settings, including timing control, receiver amplifier gain setting, transmit power shape and power level control, VCXO voltage control... etc., effect the mobile communication performance immediately. In this sector, an overview of RF radio link control is given by taking an example of MT6119C RF module (Fountain2).

In general, A RF module is composed of several parts: Transceiver chip, TXVCO (Transmit Voltage Control Oscillator), PA (Power Amplifier), VCXO (Voltage Controlled Oscillator), and other GPIO-controlled circuit. The control scenario of these parts are described as follow:

(1) Transceiver chip is controlled by activating control pins and by receiving serial commands from Baseband. Some GPO (Generic Purpose Output) pins of MT62xx can be connected to the control pins of transceiver chip. MT62xx also provides several GPO-like signals called BPI (Baseband Parallel Interface) Bus, which are activated at the specific time, to perform this control purpose(*1). The timing of activating high/low states of signals on BPI bus is programmable, and the state of signals on BPI bus is changed on the specified timing. On the other way, commands are sent from Baseband by serial interface to order transceiver to perform some specific operation. The serial interface is usually composed of 3 signal lines. The 3 lines are serial clock, serial data, and device chip select. To perform this control purpose, MT62xx chip also provides serial interface called BSI (Baseband Serial Interface) unit to send serial command to indicated device. The timing of sending serial command on BSI bus can be programmed. And the serial command is send when the programmed timing is performed. To get more hardware information, please refer to Base-band Serial Interface, Base-band Parallel Interface, and TDMA timer sectors of MT62xx Data Sheet.

PS: (*1) MT6208 provides 16 TDMA events for 9 BPI signals.
MT6205,MT6205B provide 13 TDMA events for 8 BPI signals.
MT6218/17/19/28 provides 22 TDMA events for 10 BPI signals.
MT6226/27/27D/25 provides 26 TDMA events for 10 BPI signals.
MT6229/23/35/38 provides 42 TDMA events for 10 BPI signals

(2) In order to save the power, enabling RF VCXO is synchronous with the BB VCXOEN signal.

(3) MT62xx chip provides a unit named APC (Automatic Power Control) to control the PA. The voltage-control pin of PA should be connected to the APC DAC pin of MT62xx. The timing of control voltage on APC DAC pin can be programmed. When the programmable timing is performed, the APC DAC pin performs the planned voltage to generate the specified power shape and level. To get more hardware information, Please refer to Automatic Power Control (APC) Unit and TDMA timer sectors of MT62xx Data Sheet.

(4) The frequency of VCXO can be fine tuned by controlling voltage. MT62xx chip provides a unit named AFC (Automatic Frequency Control) to control the VCXO (*2). The timing of controlling voltage on AFC DAC pin can be programmable. When the programmed timing is performed, the AFC DAC pin performs the planned voltage. The ENABLE pin of VCXO is controlled by the VCXOEN pin of MT62xx. To get more hardware information, Please refer to Automatic Frequency Control (AFC) Unit and TDMA timer sectors of MT62xx Data Sheet.

PS: (*2) AFC DAC is 10-bit DAC in MT6208.

AFC DAC is 13-bit DAC in MT6205, MT6205B, MT6218, MT6218B, MT6219, MT6223, MT6225, MT6226, MT6227, MT6228, MT6229, MT6235 and MT6238.

(5) MT62xx chip may also control the R/TX switch of antenna, band select, or power down circuit of RF module. These control actions can be performed by using MT62xx BPI bus or GPO pins. To get more hardware information, Please refer to Base-band General Purpose Input/Output, Parallel Interface, and TDMA timer sectors of MT62xx Data Sheet.

The schematic connection between generic RF module and MT62xx chip can be listed in below diagram.

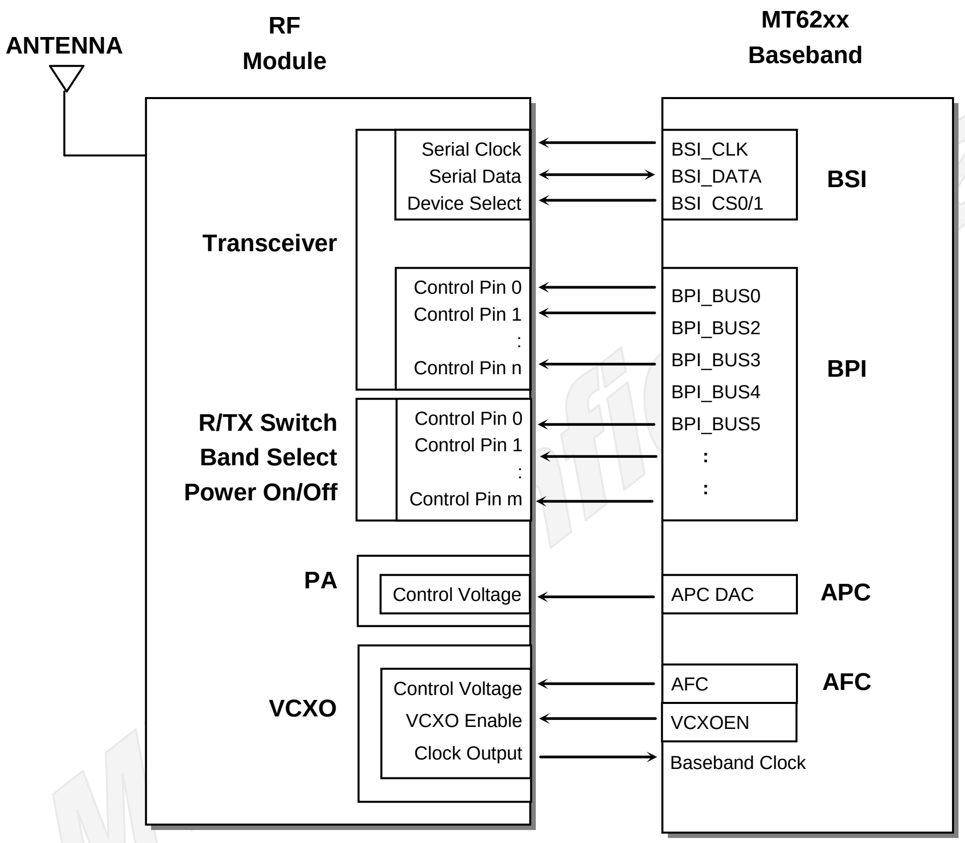


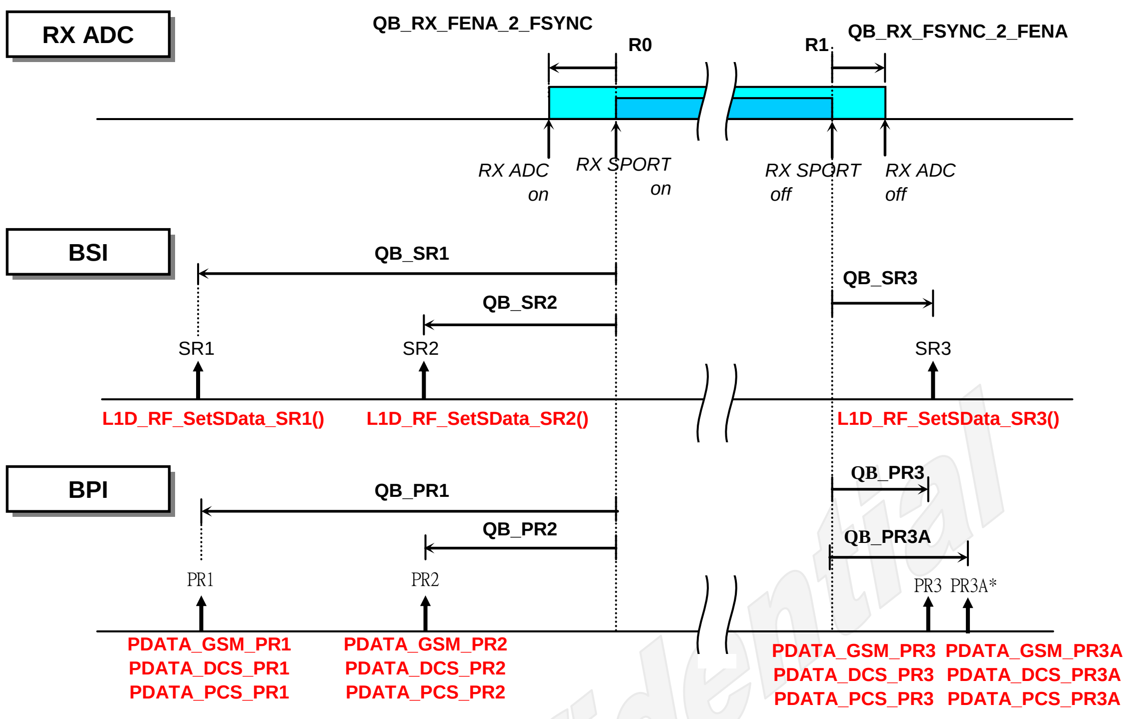
Figure 1 The connection of MT62xx and RF Module

The detail control method of RF module is introduced in next section.

2.2 Create RX Window

The TDMA timing and data setting are described in 2 parts. One part is that before turning on RX SPORT to receiving I/Q data. And the other part is after turning off RX SPORT to finish receiving I/Q data. To describe these two parts easily, the timing of turning on RX SPORT is taken as one base named **R0**. And the timing of turning off RX SPORT is taken as one base named **R1**.

Please refer to the following figure:



`* BPI event PR3A is only provided by EDGE chip MT6229, MT6235 and MT6238`

Figure 2 Event's timing of RX window

2.2.1 RX ADC Control

To setup the timing of RX ADC and SPORT, 2 timings need to be defined in **l1d_custom_rf.h**. The time from RX ADC enabling to RX SPORT turning on (**R0**) is defined as **QB_RX_FENA_2_FSYNC**.The time from RX SPORT turning on (**R1**) to RX ADC disabling is defined as **QB_RX_FSYNC_2_FENA**. The value of this two aliases should be positive or zero.

2.2.2 Serial Interface Control

BSI data and events need to be set in serial to a 3-wire base RF module. Each RX window is allocated 3 BSI events. Usually 1'st BSI event is used to warm up the synthesizer and set its N-counter to lock the operation frequency. The 2'nd BSI is used to set the receiving amplifier gain of transceiver. The 3'rd BSI is used to command transceiver entering idle mode.

The timing of 1'st BSI event is defined as **QB_SR1** in file **l1d_custom_rf.h**. With 1'st BSI event is triggered, 1 to 4 BSI data can be sent. The data count with 1'st BSI event should be claim by defining **SX1_DATA_COUNT** in file **l1d_rf.h**. To set the BSI data with 1'st BSI event, the function **L1D_RF_SetSData_SR1()** in file **m12195.c** should be implemented. That is:

(1) In **l1d_custom_rf.h**, define the timing of 1'st BSI event ahead of R0 in quarter-bit unit. If the value is negative, the timing of 1'st BSI event is behind R0. Otherwise, the timing of 1'st BSI event is ahead of R0:

```
#define   QB_SR1                QB_ahead_of_R0
```

To avoid overlapping the RF setting of other window, the limitation of SR1 setting is described detail in the chapter 3.

(2) In **custom_rf.h**, define the BSI data count with 1'st BSI event to be sent. Due to the resource limitation,

the data count with 1'st BSI event should not excess 3. I.e. the data count should be 1, 2, or 3.

```
#define  SX1_DATA_COUNT     data_count
```

(3) In **m12195.c**, implement the function **L1D_RF_SetSData_SR1** to specify the data to be sent.

(a) If BSI data count is 1,the function **L1D_RF_SetSData_SR1()** can be implemented as

```
void  L1D_RF_SetSData_SR1( void )
{
   SETUP_SR1();
   HWRITE_1_SDATA( SCTRL_WORD(device_sel, data_bit_count), data );
}
```

(b) If BSI data count is 2,the function **L1D_RF_SetSData_SR1()** can be implemented as

```
void  L1D_RF_SetSData_SR1( void )
{
   SETUP_SR1();
   HWRITE_2_SDATA( SCTRL_WORD(device_sel, data1_bit_count), data1,
                   SCTRL_WORD(device_sel, data2_bit_count), data2 );
}
```

(c) If BSI data count is 3,the function **L1D_RF_SetSData_SR1()** can be implemented as

```
void  L1D_RF_SetSData_SR1( void )
{
   SETUP_SR1();
   HWRITE_3_SDATA( SCTRL_WORD(device_sel, data1_bit_count), data1,
                   SCTRL_WORD(device_sel, data2_bit_count), data2,
                   SCTRL_WORD(device_sel, data3_bit_count), data3 );
}
```

SETUP_SR1() is the first statement in **L1D_RF_SetSData_SR1** which is a macro to hide complicated code to initialize setting 1'st BSI data. The second statement **HWRITE_1/2/3_SDATA** is specified the data to be sent by BSI bus. Two parameters are needed to specify 1 BSI data. One is control word to indicate the device selectin and the bit count of the data to be sent. The other is the data value to be sent. So 2 parameters are needed of **HWRITE_1_SDATA**, 4 parameters are needed of **HWRITE_2_SDATA**, and 6 parameters are needed of **HWRITE_3_SDATA**. Usually the 1'st BSI data may send the synthesizer N counter setting to lock the operation frequency for RX window. The variable **l1d_rf.RFN_data** which the content is the setting data evaluated by function **L1D_RF_GetRxPLLSetting** in file **m12191.c** can be used as BSI data to be sent.

The timing of 2'nd BSI event is defined as **QB_SR2** in file **l1d_custom_rf.h**. The data count with 2'nd BSI event should be claim by defining **SX2_DATA_COUNT** in file **l1d_rf.h**. Usually 2'nd BSI event is used to set the receiving amplifier gain, only 1 BSI data of gain setting is enough. So with 2'nd BSI event is triggered, only 1 BSI data can be sent by the reason of resource limitation. The value of **SX2_DATA_COUNT** should be set to 1. To set the BSI data with 2'nd BSI event, the function **L1D_RF_SetSData_SR2()** in file **m12195.c** should be implemented. That is:

(1) In **l1d_custom_rf.h**, define the timing of 2'nd BSI event ahead of R0 in quarter-bit unit. If the value is negative, the timing of 2'nd BSI event is behind R0. Otherwise, the timing of 2'nd BSI event is ahead of R0:

```
#define  QB_SR2             QB_ahead_of_R0
```

To support multi-slots receiving in GPRS mode, different LNA gain setting for each time slot is allowed. Refer to the following figure,

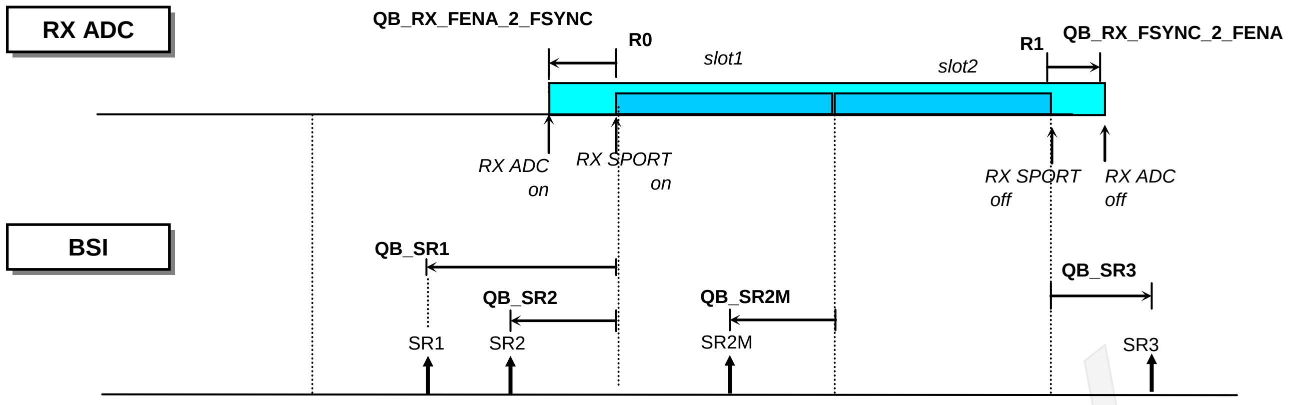


Figure 3 Event's timing of multi-slot RX window

The TDMA timing of LNA gain setting at inter slot shall be define as

```
#define  QB_SR2M             QB_ahead_of_R0
```

This timing QB_SR2M is ahead the R0 of next slot.

To avoid overlapping the RF setting of other window, the limitation of SR2 setting is described detail in the Chapter 3.

(2) In **l1d_rf.h**, define the BSI data count with 2'nd BSI event to be sent. Due to the resource limitation, the data count with 2'nd BSI event should be only 1.

```
#define  SX2_DATA_COUNT     1
```

(3) In **m12195.c**, implement the function **L1D_RF_SetSData_SR2** to specify the data to be sent.

```
void  L1D_RF_SetSData_SR2( void )
{
   SETUP_SR2();
   HWRITE_1_SDATA( SCTRL_WORD(device_sel, data_bit_count), data );
}
```

SETUP_SR2() is the first statement in **L1D_RF_SetSData_SR2** which is a macro to hide complicated code to initialize setting 2'nd BSI data. The second statement **HWRITE_1_SDATA** is specified the data to be sent by BSI bus. Two parameters are needed to specify 1 BSI data. One is control word to indicate device selection and the bit count of the data to be sent. The other is the data value to be sent. Usually the 2'nd BSI data may send the gain setting of receiving amplifier for RX window. The variable **l1d_rf.AGC_data** which the content is the setting data evaluated by function **L1D_RF_GetGainSetting** in file **m12192.c** can be used as BSI data to be sent.

The timing of 3'rd BSI event is defined as **QB_SR3** in file **l1d_custom_rf.h**. The data count with 3'rd BSI event should be claim by defining **SX3_DATA_COUNT** in file **l1d_rf.h**. Usually 3'rd BSI event is used to set the RF chip into idle or stand-by mode, only 1 BSI data of gain setting is enough. So with 3'rd BSI event is triggered, only 1 BSI data can be sent by the reason of resource limitation. The value of **SX3_DATA_COUNT** should be set to 1. To set the BSI data with 3'rd BSI event, the function **L1D_RF_SetSData_SR3()** in file **m12195.c** should be implemented. That is:

(1) In **l1d_custom_rf.h**, define the timing of 3'rd BSI event behind R1 in quarter-bit unit. If the value is negative, the timing of 3'rd BSI event is ahead of R1. Otherwise, the timing of 3'rd BSI event is behind R0:

```
#define  QB_SR3             QB_behind_R1
```

To avoid overlapping the RF setting of other window, the limitation of SR3 setting is described detail in the Chapter 3.

(2) In **l1d_rf.h**, define the BSI data count with 3'rd BSI event to be sent. Due to the resource limitation, the data count with 3'rd BSI event should be only 1.

```
#define   SX3_DATA_COUNT     1
```

(3) In **m12195.c**, implement the function **L1D_RF_SetSData_SR3** to specify the data to be sent.

```
void  L1D_RF_SetSData_SR3( void )
{
   SETUP_SR3();
   HWRITE_1_SDATA( SCTRL_WORD(device_sel, data_bit_count), data );
}
```

SETUP_SR3() is the first statement in **L1D_RF_SetSData_SR3** which is a macro to hide complicated code to initialize setting 3'rd BSI data. The using of **HWRITE_1_SDATA** is described yet. Usually Idle commend is send with 3'rd BSI event. The control word and data of BSI data of Idling command defined in **l1d_rf.h** can be used.

```
#define   SCTRL_IDLE        SCTRL_WORD(device_sel,idle_data_bit_count)
#define   SDATA_IDLE        idle_data
```

2.2.3 Parallel Interface Control

The connection of HW signals of BPI data bus and RF module is flexible and depends on customer's design. The setting timing and data setting of BPI bus are used to specify at what time and which BPI states are changed. The BPI data may be variance by the operation band, so the dedicate BPI data of each band should be defined. In MediaTek Layer 1 Design, the states transient of BPI signals are decided by the time of event, therefore the active time and the BPI states for each band shall be defined. The example of MT6119C RF module is given in Chapter 3, which could give us explicit information about the application of BPI event and data.

The 1'st BPI event is usually used to activate the RF components on RF module. The settings are:

(1) In **l1d_custom_rf.h**, define the timing of 1'st BPI event ahead of R0 in quarter-bit unit. If the value is negative, the timing of 1'st BPI event is behind R0. Otherwise, the timing of 1'st BPI event is ahead of R0:

```
#define   QB_PR1            QB_ahead_of_R0
```

To avoid overlapping the RF setting of other window, the limitation of PR1 setting is described detail in the Chapter 3.

(2) In **l1d_custom_rf.h**, define the BPI data with 1'st BPI event for each band:

```
#define   PDATA_GSM_PR1      data_for_GSM_band
#define   PDATA_DCS_PR1      data_for_DCS_band
#define   PDATA_PCS_PR1      data_for_PCS_band
```

The 2'nd BPI event is usually used to select band and switch R/TX. The settings are:

(1) In **l1d_custom.h**, define the timing of 2'nd BPI event ahead of R0 in quarter-bit unit. If the value is negative, the timing of 2'nd BPI event is behind R0. Otherwise, the timing of 2'nd BPI event is ahead of R0:

```
#define   QB_PR2            QB_ahead_of_R0
```

To avoid overlapping the RF setting of other window, the limitation of PR2 setting is described detail in the Chapter 3.

(2) In **l1d_custom.h**, define the BPI data with 2'nd BPI event for each band:

```
#define   PDATA_GSM_PR2      data_for_GSM_band
#define   PDATA_DCS_PR2      data_for_DCS_band
#define   PDATA_PCS_PR2      data_for_PCS_band
```

The 3'rd BPI event is usually used to force the RF module into idle mode. The idle state of BPI bus should be defined in **l1d_custom.h**. The settings are:

(1) In **l1d_custom.h**, define the timing of 3'rd BPI event behind R1 in quarter-bit unit. If the value is negative, the timing of 3'rd BPI event is ahead of R1. Otherwise, the timing of 3'rd BPI event is behind R0:

```
#define  QB_PR3             QB_behind_R1
```

To avoid overlapping the RF setting of other window, the limitation of PR3 setting is described detail in the Chapter 3.

(2) In **l1d_custom.h**, define the BPI data with 3'rd BPI event for each band:

```
#define  PDATA_GSM_PR3      data_for_GSM_band
#define  PDATA_DCS_PR3      data_for_DCS_band
#define  PDATA_PCS_PR3      data_for_PCS_band
```

(3) In **l1d_custom.h**, define the BPI data in idle mode:

```
#define  PDATA_IDLE         data_for_idle_mode
```

The 4'th BPI event is usually used to force the RF module into idle mode. The idle state of BPI bus should be defined in **l1d_custom.h**. The settings are:

(1) In **l1d_custom.h**, define the timing of 4'th BPI event behind R1 in quarter-bit unit. If the value is negative, the timing of 4'th BPI event is ahead of R1. Otherwise, the timing of 4'th BPI event is behind R0:

```
#define  QB_PR3A             QB_behind_R1
```

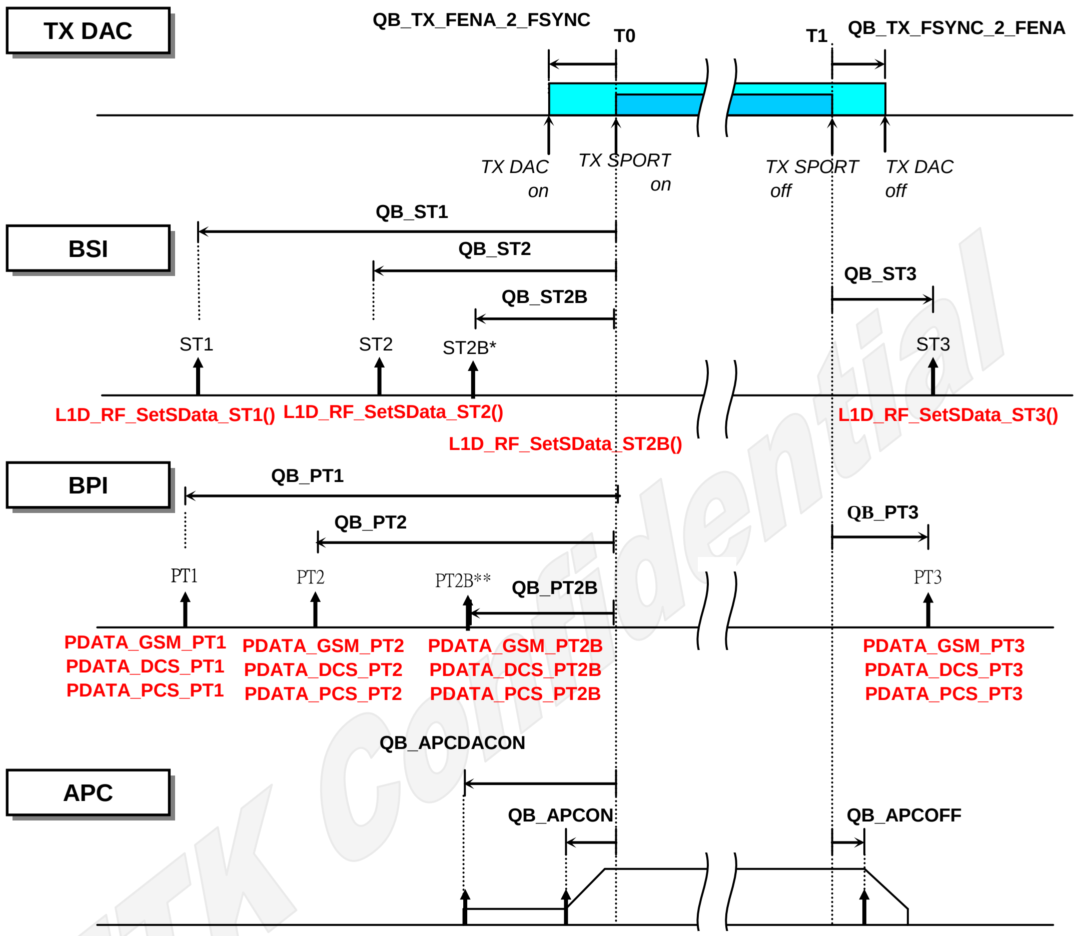
To avoid overlapping the RF setting of other window, the limitation of PR3A setting is described detail in the Chapter 3.

(2) In **l1d_custom.h**, define the BPI data with 3'rd BPI event for each band:

```
#define  PDATA_GSM_PR3A      data_for_GSM_band
#define  PDATA_DCS_PR3A      data_for_DCS_band
#define  PDATA_PCS_PR3A      data_for_PCS_band
```

2.3 Create TX Window

Please refer to the following figure:



```
* BSI event ST2B is only provided by EDGE chip MT6229, MT6235 and MT6238
** BPI event PT2B is not provided by GSM chip MT6205
```

Figure 4 Event's timing of TX window

The TDMA timing and data setting are described in 2 parts. One part is that before turning on TX SPORT to transmitting data. And the other part is after turning off TX SPORT to finish transmitting data. To describe these two parts easily, the timing of turning on TX SPORT is taken as one base named **T0**. And the timing of turning off TX SPORT is taken as one base named **T1**.

2.3.1 TX DAC Control

To setup the timing of TX DAC and SPORT, 2 timings need to be defined in **l1d_custom.h**. The time from TX DAC enabling to TX SPORT turning on (**T0**) is defined as **QB_TX_FENA_2_FSYNC**.The time from TX SPORT turning on (**T1**) to TX DAC disabling is defined as **QB_TX_FSYNC_2_FENA**. The value of this two aliases should be positive or zero.

2.3.2 Serial Interface Control

BSI data and events need to be set in order to sent serial data to 3-wire devices on RF module. Each TX window is allocated 3 BSI events. Usually 1'st BSI event is used to warm up the set synthesizer and set its N-counter to lock the operation frequency. The 2'nd BSI is used to set the transmit command and indicate the operation band. The 3'rd BSI is used to command transceiver entering idle mode.

The timing of 1'st BSI event is defined as **QB_ST1** in file **l1d_custom.h**. With 1'st BSI event is triggered, 1 to 4 BSI data can be sent. The data count with 1'st BSI event should be claim by defining **SX1_DATA_COUNT** in file **l1d_rf.h**. To set the BSI data with 1'st BSI event, the function **L1D_RF_SetSData_ST1()** in file **m12195.c** should be implemented. That is:

(1) In **l1d_custom_rf.h**, define the timing of 1'st BSI event ahead of T0 in quarter-bit unit. If the value is negative, the timing of 1'st BSI event is behind T0. Otherwise, the timing of 1'st BSI event is ahead of T0:

```
#define  QB_ST1              QB_ahead_of_T0
```

To avoid overlapping the RF setting of other window, the limitation of ST1 setting is described detail in the Chapter 3.

(2) In **l1d_rf.h**, define the BSI data count with 1'st BSI event to be sent. Due to the resource limitation, the data count with 1'st BSI event should not excess 3. I.e. the data count should be 1, 2, or 3.

```
#define  SX1_DATA_COUNT     data_count
```

(3) In **m12195.c**, implement the function **L1D_RF_SetSData_ST1** to specify the data to be sent.

(a) If BSI data count is 1,the function **L1D_RF_SetSData_ST1()** can be implemented as

```
void  L1D_RF_SetSData_ST1( void )
{
   SETUP_ST1();
   HWRITE_1_SDATA( SCTRL_WORD(device_sel, data_bit_count), data );
}
```

(b) If BSI data count is 2,the function **L1D_RF_SetSData_ST1()** can be implemented as

```
void  L1D_RF_SetSData_ST1( void )
{
   SETUP_ST1();
   HWRITE_2_SDATA( SCTRL_WORD(device_sel, data1_bit_count), data1,
                   SCTRL_WORD(device_sel, data2_bit_count), data2 );
}
```

(c) If BSI data count is 3,the function **L1D_RF_SetSData_ST1()** can be implemented as

```
void  L1D_RF_SetSData_ST1( void )
{
   SETUP_ST1();
   HWRITE_3_SDATA( SCTRL_WORD(device_sel, data1_bit_count), data1,
                   SCTRL_WORD(device_sel, data2_bit_count), data2,
                   SCTRL_WORD(device_sel, data3_bit_count), data3 );
}
```

SETUP_ST1() is the first statement in **L1D_RF_SetSData_ST1** which is a macro to hide complicated code to initialize setting 1'st BSI data. The second statement **HWRITE_1/2/3_SDATA** is specified the data to be sent by BSI bus. Two parameters are needed to specify 1 BSI data. One is control word to indicate device selection and the bit count of the data to be sent. The other is the data value to be sent. So 2 parameters are needed of **HWRITE_1_SDATA**, 4 parameters are needed of **HWRITE_2_SDATA**, and 6 parameters are needed of **HWRITE_3_SDATA**. Usually the 1'st BSI data may send the synthesizer N counter setting to lock the operation frequency for TX window. The variable **l1d_rf.RFN_data** which the content is the setting data evaluated by function **L1D_RF_GetTXPLLSetting** in file **m12191.c** can be used as BSI data to be sent.

The timing of 2'nd BSI event is defined as **QB_ST2** in file **l1d_custom_rf.h**. The data count with 2'nd BSI event should be claim by defining **SX2_DATA_COUNT** in file **l1d_rf.h**. Usually 2'nd BSI event is used to set the transmit mode and the operation band, only 1 BSI data of gain setting is enough. So with 2'nd BSI event is triggered, only 1 BSI data can be sent by the reason of resource limitation. The value of **SX2_DATA_COUNT** should be set to 1. To set the BSI data with 2'nd BSI event, the function **L1D_RF_SetSData_ST2()** in file **m12195.c** should be implemented. That is:

(1) In **l1d_custom_rf.h**, define the timing of 2'nd BSI event ahead of T0 in quarter-bit unit. If the value is negative, the timing of 2'nd BSI event is behind T0. Otherwise, the timing of 2'nd BSI event is ahead of T0:

```
#define  QB_ST2              QB_ahead_of_T0
```

To avoid overlapping the RF setting of other window, the limitation of ST2 setting is described detail in the Chapter 3.

(2) In **l1d_rf.h**, define the BSI data count with 2'nd BSI event to be sent. Due to the resource limitation, the data count with 2'nd BSI event should be only 1.

```
#define  SX2_DATA_COUNT     1
```

(3) In **m12195.c**, implement the function **L1D_RF_SetSData_ST2** to specify the data to be sent.

```
void  L1D_RF_SetSData_ST2( void )
{
   SETUP_ST2();
   HWRITE_1_SDATA( SCTRL_WORD(device_sel, data_bit_count), data );
}
```

SETUP_ST2() is the first statement in **L1D_RF_SetSData_ST2** which is a macro to hide complicated code to initialize setting 2'nd BSI data. The second statement **HWRITE_1_SDATA** is specified the data to be sent by BSI bus. Two parameters are needed to specify 1 BSI data. One is control word to indicate device selection and the bit count of the data to be sent. The other is the data value to be sent. Usually 2'nd BSI event is used to set the transmit mode and the operation band. The data with 2'nd BSI event is chosen by operation band. User can decide the data by the skill of looking-up table.

The timing of 3'rd BSI event is defined as **QB_ST3** in file **l1d_custom_rf.h**. The data count with 3'rd BSI event should be claim by defining **SX3_DATA_COUNT** in file **l1d_rf.h**. Usually 3'rd BSI event is used to set the RF chip into idle or stand-by mode, only 1 BSI data of gain setting is enough. So with 3'rd BSI event is triggered, only 1 BSI data can be sent by the reason of resource limitation. The value of **SX3_DATA_COUNT** should be set to 1. To set the BSI data with 3'rd BSI event, the function **L1D_RF_SetSData_ST3()** in file **m12195.c** should be implemented. That is:

(1) In **l1d_custom_rf.h**, define the timing of 3'rd BSI event behind R1 in quarter-bit unit. If the value is negative, the timing of 3'rd BSI event is ahead of T1. Otherwise, the timing of 3'rd BSI event is behind R0:

```
#define  QB_ST3              QB_behind_R1
```

To avoid overlapping the RF setting of other window, the limitation of ST3 setting is described detail in the Chapter 3.

(2) In **l1d_rf.h**, define the BSI data count with 3'rd BSI event to be sent. Due to the resource limitation, the data count with 3'rd BSI event should be only 1.

```
#define  SX3_DATA_COUNT     1
```

(3) In **m12195.c**, implement the function **L1D_RF_SetSData_ST3** to specify the data to be sent.

```
void  L1D_RF_SetSData_ST3( void )
{
   SETUP_ST3();
   HWRITE_1_SDATA( SCTRL_WORD(device_sel, data_bit_count), data );
}
```

SETUP_ST3() is the first statement in **L1D_RF_SetSData_ST3** which is a macro to hide complicated code to initialize setting 3'rd BSI data. The using of **HWRITE_1_SDATA** is described yet. Usually Idle commend is send with 3'rd BSI event. The control word and data of BSI data of Idling command defined in **l1d_rf.h** can be used.

```
#define  SCTRL_IDLE       SCTRL_WORD(device_sel,idle_data_bit_count)
#define  SDATA_IDLE       idle_data
```

The timing of 4'th BSI event is defined as **QB_ST2B** in file **l1d_custom_rf.h**. Usually 4'th BSI event is used to set the transmit modulation type. To set the BSI data with 4'th BSI event, the function **L1D_RF_SetSData_ST2B()** in file **m12195.c** should be implemented. That is:

(1) In **l1d_custom_rf.h**, define the timing of 4'th BSI event ahead of T0 in quarter-bit unit. If the value is negative, the timing of 4'th BSI event is behind T0. Otherwise, the timing of 4'th BSI event is ahead of T0:

```
#define  QB_ST2B              QB_ahead_of_T0
```

To avoid overlapping the RF setting of other window, the limitation of ST2B setting is described detail in the Chapter 3.

(2) In **m12195.c**, implement the function **L1D_RF_SetSData_ST2B** to specify the data to be sent.

```
void  L1D_RF_SetSData_ST2B( void )
{
   SETUP_ST2M();
   HWRITE_1_SDATA( SCTRL_WORD(device_sel, data_bit_count), data );
}
```

SETUP_ST2M() is the first statement in **L1D_RF_SetSData_ST2B** which is a macro to hide complicated code to initialize setting 4'th BSI data. The second statement **HWRITE_1_SDATA** is specified the data to be sent by BSI bus. Two parameters are needed to specify 1 BSI data. One is control word to indicate device selection and the bit count of the data to be sent. The other is the data value to be sent. Usually 4'th BSI event is used to set the transmit modulation type.

2.3.3 Parallel Interface Control

The setting of BPI bus includes timing and data setting to specify at what time what BPI states are changed. The BPI data may be variance by operation band, so the BPI data of each band should be defined. The 1'st BPI event is usually used to activate the RF components on RF module in transmit mode. The settings are:

(1) In **l1d_custom_rf.h**, define the timing of 1'st BPI event ahead of T0 in quarter-bit unit. If the value is negative, the timing of 1'st BPI event is behind T0. Otherwise, the timing of 1'st BPI event is ahead of T0:

```
#define  QB_PT1             QB_ahead_of_T0
```

To avoid overlapping the RF setting of other window, the limitation of PT1 setting is described detail in the Chapter 3.

(2) In **l1d_custom_rf.h**, define the BPI data with 1'st BPI event for each band:

```
#define  PDATA_GSM_PT1      data_for_GSM_band
#define  PDATA_DCS_PT1      data_for_DCS_band
```

```
#define   PDATA_PCS_PT1       data_for_PCS_band
```

The 2'nd BPI event is usually used to select band and switch R/TX. The settings are:

(1) In **l1d_custom_rf.h**, define the timing of 2'nd BPI event ahead of T0 in quarter-bit unit. If the value is negative, the timing of 2'nd BPI event is behind T0. Otherwise, the timing of 2'nd BPI event is ahead of T0:

```
#define   QB_PT2              QB_ahead_of_T0
```

To avoid overlapping the RF setting of other window, the limitation of PT2 setting is described detail in the Chapter 3.

(2) In **l1d_custom_rf.h**, define the BPI data with 2'nd BPI event for each band:

```
#define   PDATA_GSM_PT2       data_for_GSM_band
#define   PDATA_DCS_PT2       data_for_DCS_band
#define   PDATA_PCS_PT2       data_for_PCS_band
```

The 3'rd BPI event is usually used to force the RF module into idle mode. The idle state of BPI bus should be defined in **l1d_custom_rf.h**. The settings are:

(1) In **l1d_custom_rf.h**, define the timing of 3'rd BPI event behind T1 in quarter-bit unit. If the value is negative, the timing of 3'rd BPI event is ahead of T1. Otherwise, the timing of 3'rd BPI event is behind T0:

```
#define   QB_PT3              QB_behind_T1
```

To avoid overlapping the RF setting of other window, the limitation of PT3 setting is described detail in the Chapter 3.

(2) In **l1d_custom_rf.h**, define the BPI data with 3'rd BPI event for each band:

```
#define   PDATA_GSM_PT3       data_for_GSM_band
#define   PDATA_DCS_PT3       data_for_DCS_band
#define   PDATA_PCS_PT3       data_for_PCS_band
```

One more BPI event called PT2B has been added after Maui 05B. This will let MS to pass PvT mask and Switching ORFS easily. PT2B is located after PT1 and PT2, but before PT3.

The settings of PT2B BPI event are as following:

(1) In **l1d_custom_rf.h**, define the timing of PT2B BPI event ahead of T0 in quarter-bit unit. If the value is negative, the timing of PT2B BPI event is behind T0. Otherwise, the timing of PT2B BPI event is ahead of T0:

```
#define   QB_PT2B              QB_ahead_of_T0
```

To avoid overlapping the RF setting of other window, the limitation of PT2B setting is described detail in the Chapter 3.

(2) In **l1d_custom_rf.h**, define the BPI data with PT2B BPI event for each band:

```
#define   PDATA_GSM_PT2B       data_for_GSM_band
#define   PDATA_DCS_PT2B       data_for_DCS_band
#define   PDATA_PCS_PT2B       data_for_PCS_band
```

2.3.4 APC Control

In addition to TX DAC, TX SPORT, BSI, BPI unit needs to be set, the control of PA is important for TX window. The PA is control by the APC unit of MT62xx. Before the data transmission, APC ramps up the PA to the indicated power level. Data is transmitted at that level. After finishing transmission, APC ramps down the PA.

Before PA ramping up, A DC offset of PA is performed to let PA ramp up smoothly. The timing of triggering PA DC offset, ramp up, ramp down should be defined in **l1d_custom_rf.h**.

(1) In **l1d_custom_rf.h**, define the timing of performing DC offset of PA ahead of T0 in quarter-bit unit. If the value is negative, the timing is behind T0. Otherwise, the timing is ahead of T0:

```
#define   QB_APCDACON      QB_ahead_of_T0
```

(2) In **l1d_custom_rf.h**, define the timing of ramping up ahead of T0 in quarter-bit unit. If the value is negative, the timing is behind T0. Otherwise, the timing is ahead of T0:

```
#define   QB_APCON      QB_ahead_of_T0
```

(3) In **l1d_custom_rf.h**, define the timing of ramping down behind T1 in quarter-bit unit. If the value is negative, the timing is ahead of T1. Otherwise, the timing is behind T0:

```
#define   QB_APCOFF     QB_behind_T1
```

(4) To support multi-slot transmit in GPRS mode, The TDMA event of power ramp for inter-slot shall be defined in **l1d_custom_rf.h.** If the value is negative, the timing is behind T0 of next burst. Otherwise, the timing is ahead of T0. Refer the following figure:

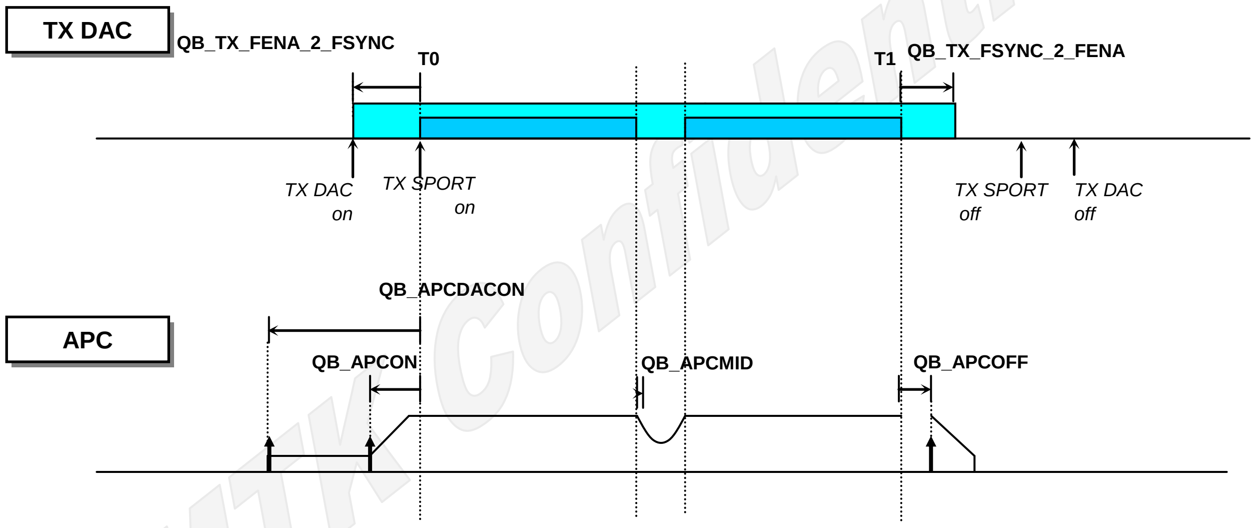


Figure 5 APC Event's timing of multi-slots TX window

```
#define   QB_APCMID  QB_ahead_of_T0
```

The propagation delay when transmit data from Baseband uplink DAC to antenna should be claimed in file **l1d_custom_rf.h**. This delay depends on different RF module. That is

```
#define   TX_PROPAGATION_DELAY     QB_of_TX_delay
```

The unit of the value *delay* is quarter bit (1QB = 0.923us).

2.3.5 Multi-slots TX window

To support multi-slots receiving in EGPRS mode, different modulation type setting for each time slot is allowed. Refer to the following figure,

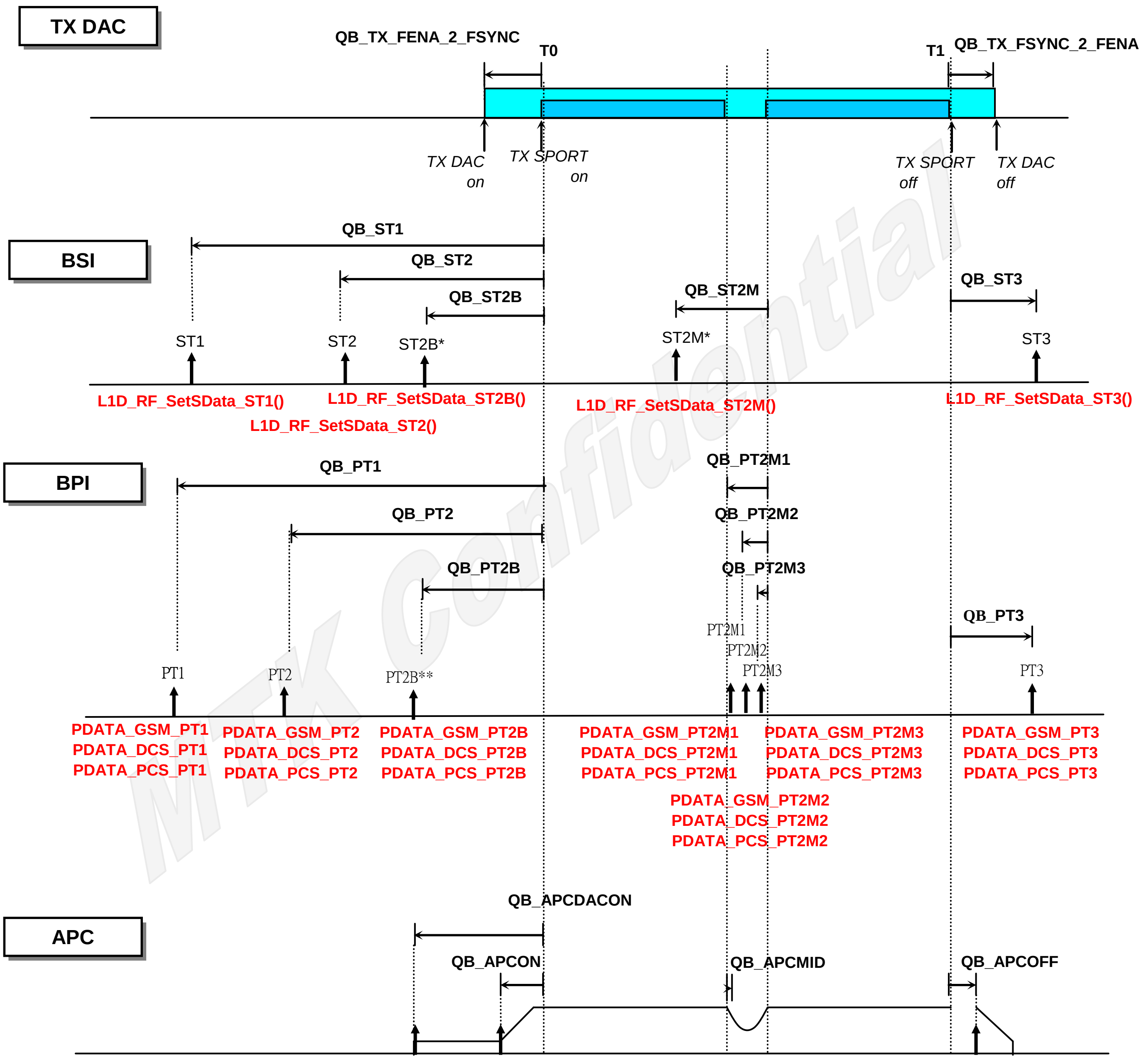


Figure 6 Event's timing of multi-slots TX window

The timing of BSI event is defined as **QB_ST2M** in file **l1d_custom_rf.h**. Usually ST2M BSI event is used to set the transmit modulation type. To set the BSI data with ST2M BSI event, the function **L1D_RF_SetSData_ST2M()** in file **m12195.c** should be implemented. That is:

(1) In **l1d_custom_rf.h**, define the timing of BSI event ahead of T0 of next slot in quarter-bit unit. If the value is negative, the timing of 2'nd BSI event is behind T0 of next slot. Otherwise, the timing of BSI event is ahead of T0 of next slot. Furthermore, there are two event timing of ST2M for mudulation type switching, QB_ST2M_8G is used for EPSK to GMSK switching, otherwise QB_ST2M_G8 is used for GMSK to EPSK switching, and the others will use QB_ST2M_8G:

```
#define  QB_ST2M_8G             QB_ahead_of_T0 of next slot .
#define  QB_ST2M_G8             QB_ahead_of_T0 of next slot .
```

(2) In **m12195.c**, implement the function **L1D_RF_SetSData_ST2M** to specify the data to be sent.

```
void  L1D_RF_SetSData_ST2M( void )
{
   SETUP_ST2M();
   HWRITE_1_SDATA( SCTRL_WORD(device_sel, data_bit_count), data );
}
```

SETUP_ST2M() is the first statement in **L1D_RF_SetSData_ST2M** which is a macro to hide complicated code to initialize setting ST2M BSI data. The second statement **HWRITE_1_SDATA** is specified the data to be sent by BSI bus. Two parameters are needed to specify 1 BSI data. One is control word to indicate device selection and the bit count of the data to be sent. The other is the data value to be sent.

Three BPI events named PT2M1, PT2M2, PT2M3 have been add for TX inter-slot control, settings are:

(1) In **l1d_custom_rf.h**, define the timing of BPI event ahead of T0 of next slot in quarter-bit unit. If the value is negative, the timing of BPI event is behind T0 of next slot. Otherwise, the timing of BPI event is ahead of T0 of next slot, there are two timing of each event depends on the relation of multi-slots mudulation type:

```
#define  QB_PT2M1_G8            QB_ahead_of_T0 of next slot
#define  QB_PT2M2_G8            QB_ahead_of_T0 of next slot
#define  QB_PT2M3_G8            QB_ahead_of_T0 of next slot
#define  QB_PT2M1_8G            QB_ahead_of_T0 of next slot
#define  QB_PT2M2_8G            QB_ahead_of_T0 of next slot
#define  QB_PT2M3_8G            QB_ahead_of_T0 of next slot
```

(2) In **l1d_custom_rf.h**, define the BPI data with BPI event for each band, each event has two BPI data depends on the relation of multi-slots mudulation type :

```
#define  PDATA_GSM_PT2M1_G8      data_for_GSM_band
#define  PDATA_DCS_PT2M1 G8     data_for_DCS_band
#define  PDATA_PCS_PT2M1 G8     data_for_PCS_band
#define  PDATA_GSM_PT2M1_8G      data_for_GSM_band
#define  PDATA_DCS_PT2M1 8G     data_for_DCS_band
#define  PDATA_PCS_PT2M1 8G     data_for_PCS_band
#define  PDATA_GSM_PT2M2_G8      data_for_GSM_band
#define  PDATA_DCS_PT2M2 G8     data_for_DCS_band
#define  PDATA_PCS_PT2M2 G8     data_for_PCS_band
#define  PDATA_GSM_PT2M2_8G      data_for_GSM_band
#define  PDATA_DCS_PT2M2 8G     data_for_DCS_band
#define  PDATA_PCS_PT2M2 8G     data_for_PCS_band
#define  PDATA_GSM_PT2M3_G8      data_for_GSM_band
#define  PDATA_DCS_PT2M3 G8     data_for_DCS_band
#define  PDATA_PCS_PT2M3 G8     data_for_PCS_band
#define  PDATA_GSM_PT2M3_8G      data_for_GSM_band
```

```
#define  PDATA_DCS_PT2M3 8G    data_for_DCS_band
#define  PDATA_PCS_PT2M3 8G    data_for_PCS_band
```

2.4 RF module Initialization

RF module initialization is implemented in **L1D_RF_PowerOn** in **m12196.c**. BPI and BSI shall be directly active without waiting for TDMA events at this time. A special mode named IMMediate mode is used for this purpose. In this mode, BPI and BSI data can immediately send by calling **IMM_XXX** functions. Some rules shall be followed by using these **IMM_XXX** functions:

(1) Before sending BPI or BSI data immediately, calling **IMM_MODE_BEGIN(** *enabled_mask* **)**
where *enabled_mask* is the mask of enabled part in immediate mode. For example,
If BPI immediate mode is used, *enabled_mask* shall include **IMMMASK_BPI**.
If BSI immediate mode is used, *enabled_mask* shall include **IMMMASK_BSI**.
If both BPI and BSI immediate mode are used, *enabled_mask* can be **IMMMASK_ALL**.

(2) BPI data can be immediately send by calling **IMM_SEND_BPI(** *bpi_data* **)**

(3) BSI data can be immediately send by calling **IMM_SEND_BSI(** `SCTRL_WORD(`*device_sel*`,` *data_bit_count*`),` *bsi_sata* **)**

(4) After sending BPI or BSI data immediately, calling **IMM_MODE_END( )** to stop immediated mode

Layer 1 calls **L1D_RF_PowerOn** when power on the phone. Immediate mode can be used in this function.

Layer 1 calls **L1D_RF_PowerOff** when power off the phone. Immediate mode can be used in this function.

Layer 1 calls **L1D_RF_WindowsOn** when no RX/TX window is created this frame but next frame needs to create.

Layer 1 calls **L1D_RF_WindowsOff** when any RX/TX window is created this frame but next frame doesn't need to create.

2.5 Global RF Variables Used for Setting BSI Data

Some useful global variables can be used for setting BSI data. These variables are only used in functions **L1D_RF_SetSData_SR1/2/3** and **L1D_RF_SetSData_ST1/2/3** in file **m12195.c**. All these useful variables are declared in the structure object **l1d_rf.** The declaration of **l1d_rf** is as the following:

```
typedef  struct
{
   char  band;
      :
   unsigned long  RFN_data;
   unsigned long  IFN_data;
   unsigned long  AGC_data;

}  sRFSETTING;

extern  sRFSETTING l1d_rf;
```

The meaning of each variable are explained as follow:

(1) **l1d_rf.band** : The band of current operation channel. The value can be the following alias declared in **m12190.h**:

FrequencyBand400	**:**	Indicate the used band is GSM 400. The value is 0.
FrequencyBand850	**:**	Indicate the used band is GSM 800. The value is 1.
FrequencyBand900	**:**	Indicate the used band is GSM 900. The value is 2.
FrequencyBand1800	**:**	Indicate the used band is DCS 1800. The value is 3.
FrequencyBand1900	**:**	Indicate the used band is PCS 1900. The value is 4.

(2) **l1d_rf.RFN_data** : The Synthesizer RF N-Counter setting evaluated by the function **L1D_RF_GetRxPLLSetting** or **L1D_RF_GetTxPLLSetting**.

(3) **l1d_rf.IFN_data** : The Synthesizer IF N-Counter setting evaluated by the function **L1D_RF_GetRxPLLSetting** or **L1D_RF_GetTxPLLSetting**.

(4) **l1d_rf.AGC_data** : The transceiver gain setting evaluated by function **L1D_RF_GetGainSetting** .

2.6 Synthesizer N-Counter Evaluation

Before receiving or transmission data, a setting value for synthesizer to lock a suitable operation frequency is sent by BSI interface. This setting value is evaluated in indicated functions. The implementation of the evaluative function is different for used RF transceiver chip. It's very important that don't waste long time to evaluate the result in this function.

2.6.1 Evaluated Synthesizer N-counter Setting for RX window

Function **L1D_RF_GetRxPLLSetting** in file **m12191.c** is called by Layer 1 to get the Synthesizer N-counter setting for RX window. The prototype of the function is:

void L1D_RF_GetRxPLLSetting(int rf_band, int arfcn, long *rfN, long *ifN)

Input parameters : (1) **rf_band** : Indicate the band of the request frequency. The value can be

FrequencyBand400	(0)
FrequencyBand850	(1)
FrequencyBand900	(2, GSM)
FrequencyBand1800	(3, DCS)
FrequencyBand1900	(4, PCS)

(2) **arfcn** : Indicate the ARFCN number of the request frequency.

Output parameters : The output parameters are called by address to return the evaluated setting value
(1) ***rfN** : the evaluated synthesizer RF-N counter setting.
(2) ***ifN** : the evaluated synthesizer IF-N counter setting.

2.6.2 Evaluated Synthesizer N-counter Setting for TX window

Function **L1D_RF_GetTxPLLSetting** in file **m12191.c** is called by Layer 1 to get the Synthesizer N-counter setting for RX window. The prototype of the function is:

void L1D_RF_GetTxPLLSetting(int rf_band, int arfcn, long *rfN, long *ifN)

Input parameters : (1) **rf_band** : Indicate the band of the request frequency. The value can be

FrequencyBand400	(0)
FrequencyBand850	(1)
FrequencyBand900	(2, GSM)

	FrequencyBand1800	(3, DCS)
	FrequencyBand1900	(4, PCS)

(2) **arfcn** : Indicate the ARFCN number of the request frequency.

Output parameters : The output parameters are called by address to return the evaluated setting value
(1) **rfN** : the evaluated synthesizer RF-N counter setting.
(2) **ifN** : the evaluated synthesizer IF-N counter setting.

2.7 Transceiver Gain Setting Evaluation

Before receiving data, a setting value of suitable receiving amplifier gain should be sent by BSI interface. This setting value is evaluated in indicated functions. The implementation of the evaluative function is different for used RF transceiver chip. It's very important that don't waste long time to evaluate the result in this function.

Function **L1D_RF_GetGainSetting** in file **m12192.c** is called by Layer 1 to get the receiving amplifier gain setting for RX window. The prototype of the function is:

```
int L1D_RF_GetGainSetting( int rf_band, int arfcn, int request_gain, long *gain_setting )
```

Input parameters : (1) **rf_band** : Indicate the operating band . The value can be

FrequencyBand400	(0)
FrequencyBand850	(1)
FrequencyBand900	(2, GSM)
FrequencyBand1800	(3, DCS)
FrequencyBand1900	(4, PCS)

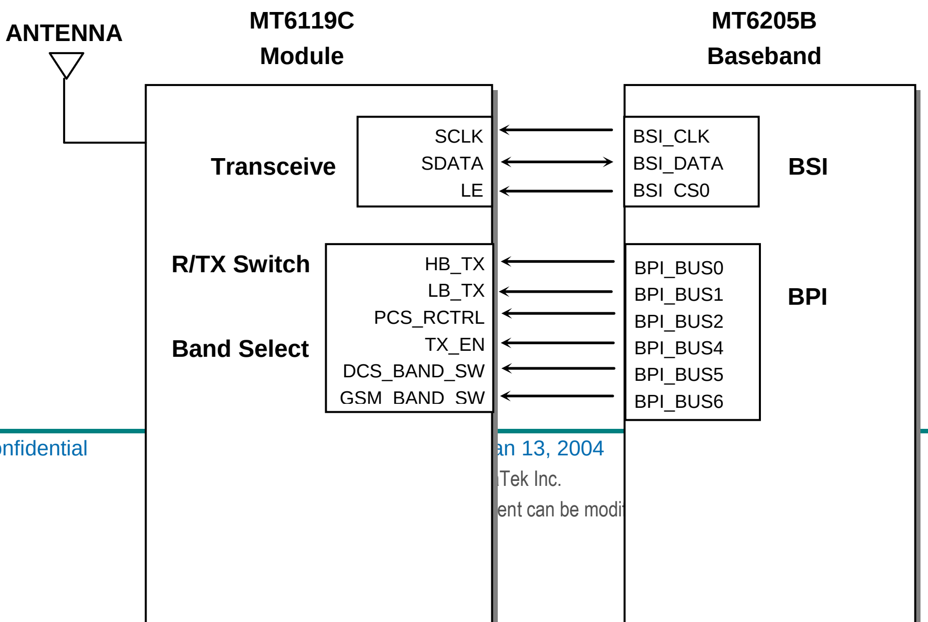
(2) **arfcn** : Indicate the operating ARFCN number
(3) **request_gain** : Indicate the request gain in the unit of 0.125 dB.
ex. **request_gain** = 103 means that request gain is 12.875 dB

Output parameters : The output parameter is called by address to return the evaluated setting value
(1) **agc_setting** : the evaluated result of transceiver gain setting.

Return value: The actual used gain with the evaluated setting value. This actual gain is in the unit of 0.125 dB.

2.8 RF Control Configuration of MT6119C RF Module

For example, the schematic circuit interface between MT6119C RF module and MT62xx chip is shown in below diagram. Please refer to MediaTek MT6119C RF Module schematic circuit to get more detail information.



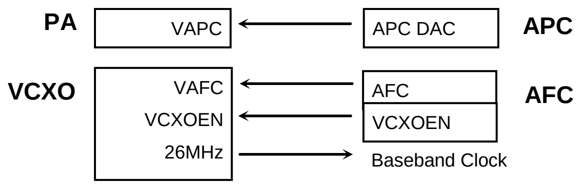


Figure 7 The connection of MT62xx and MT6119C RF Module

BSI_CLK, BSI_DATA, BSI_CS0 pins of MT62xx are directly connected to the serial interface of MT6119C transceiver. APC_DAC pin of MT62xx is connected to the VAPC pin of PA of RF module board. VCXOEN of MT62xx controls the enable state of VCXO. AFC DAC output is connected to the voltage control pin VAFC of VCXO. The generated clock of VCXO provides MT62xx system clock. Band-related pins HB_TX, LB_TX, PCS_RCTRL, TX_EN, DCS_BAND_SW, GSM_BAND_SW are controlled by the bit 0,1,2,4,5,6 of BPI bus of MT62xx. The high/low state of each pin of BPI bus and their timing of changing state can be programmed by MT62xx. The RF radio control method is introduced in the latter description.

2.8.1 Initialization

RF module can be controlled by the BSI, BPI, APC, AFC units of MT62xx chip. BSI, BPI, APC, AFC units can work in 2 modes. One mode is immediate control mode, and the other is TDMA event driven control mode.

Immediate control mode is that the output signal of BSI, BPI, APC, or AFC unit is generated as soon as setting the corresponding registers in MT62xx. When turning on the RF module, the immediate mode can be used to initialize the RF module. For example, immediate control mode is used when initializing MT6119C RF module, and the initial states are set as follow:

(1) HB_TX (BPI_BUS0) is low state
(2) LB_TX (BPI_BUS1) is low state
(3) PCS_RCTRL (BPI_BUS2) is low state
(4) TXEN (BPI_BUS4) is low state
(5) DCS_BAND_SW (BPI_BUS5) is low state
(6) GSM_BAND_SW (BPI_BUS6) is low state
(7) Serial commands (0x040000, 0x02A0A8, 0x00A0A8, 0x002366, 0x124992, 0x128992) are sending to MT6119C transceiver.

The initialization of BPI, BSI data can be implemented in **L1D_RF_PowerOn** in **m12196.c** by using immediate mode:

```
void  L1D_RF_PowerOn( void )
{
  IMM_MODE_BEGIN( IMMMASK_ALL );
  IMM_SEND_BPI( 0x0000 );
  IMM_SEND_BSI( SCTRL_IMOD(0,23), 0x040000 );
  IMM_SEND_BSI( SCTRL_IMOD(0,23), 0x02A0A8 );
  IMM_SEND_BSI( SCTRL_IMOD(0,23), 0x002366 );
  IMM_SEND_BSI( SCTRL_IMOD(0,23), 0x124992 );
  IMM_SEND_BSI( SCTRL_IMOD(0,23), 0x128992 );
  IMM_MODE_END(  );
}
```

2.8.2 Create RX window

TDMA timer is a counter unit that counts 1 per quarter-bit (0.923u). It always repeats counting from 0 to 4999. So the period of TDMA counter is 5000 QB (1 TDMA frame). If BSI, BPI, AFC, APC are programmed in TDMA event driven mode, the active time can be programmed. When the TDMA counter counts to the programmed time (count), the corresponding unit is triggered to activate.

If it is needed to receive data from air or transmit data to air, Layer1 needs to plan all RF setting and timing to create RX/TX window, control receiving gain, control VCXO, control PA... etc. For this purpose, TDMA event driven mode of BSI, BPI, APC, or AFC units should be used.

For example of creating RX window to receive data, the RF setting and timing of MT6119C RF module can be depicted in the below diagram:

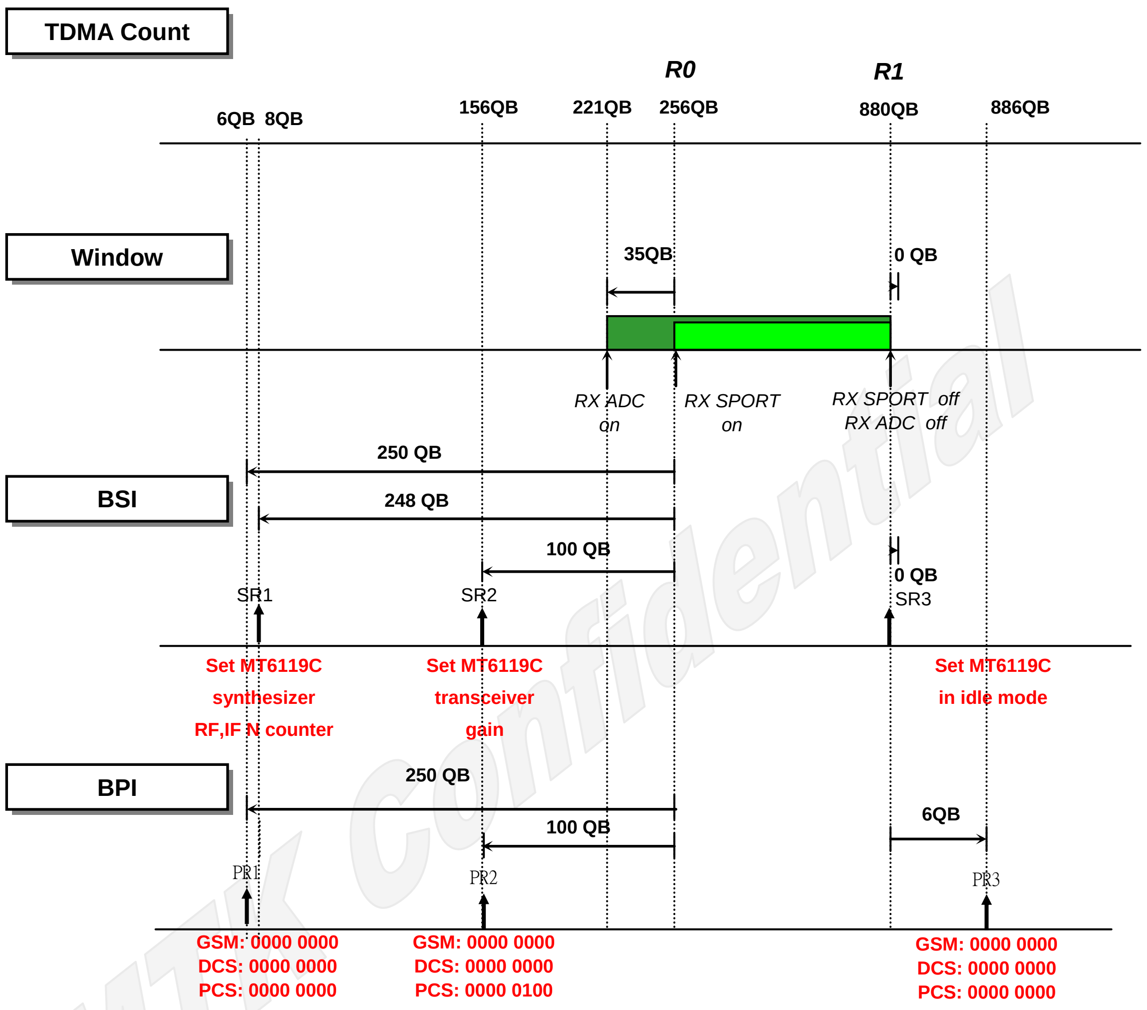


Figure 8 Event's timing of RX window

The TDMA timing and data setting are described in 2 parts. One part is that before turning on RX SPORT to receiving I/Q data from MT6119C transceiver. And the other part is after turning off RX SPORT to finish receiving I/Q data. To describe these two parts easily, the timing of turning on RX SPORT is taken as one base named **R0**. And the timing of turning off RX SPORT is taken as one base named **R1**. In this case, R0 is at TDMA counter 256 QB and R1 is at TDMA counter 880 QB.

2.8.2.1 Set Synthesizer N-Counter

250 QB ahead of R0, RF module should start to operate. MT62xx chip controls the RF module by activating control pins and sending serial command. MT62xx activates the control pins by changing BPI bus states. The high/low state of BPI bus at this time depends on operation band. After 2 QB, MT62xx also sends serial command to lock MT6119C synthesizer to lock the operation frequency by BSI bus. To perform these operations, the timing and data of BPI and BSI bus can be implemented as follow steps:

(1) Modified the timing definition in **l1d_custom_rf.h**. **QB_SR1** is the timing definition of 1'st BSI command of RX window. **QB_PR1** is the timing definition of 1'st BPI bus status changing of RX window.

```
#define   QB_SR1           250
#define   QB_PR1           248
```

(2) Define the states of BPI bus for each band. The data is shown in the below table:

BPI_BUS	7	6	5	4	3	2	1	0	Hex
Function	reserved	GSM BAND_SW	DCS BAND_SW	TX_EN	reserved	PCS_RCTRL	LB_TX	HB_TX	
GSM	X	0	0	0	X	0	0	0	00h
DCS	X	0	0	0	X	0	0	0	00h
PCS	X	0	0	0	X	0	0	0	00h

So the BPI data can be defined as follow:

```
#define  PDATA_GSM_PR1     0x00
#define  PDATA_DCS_PR1     0x00
#define  PDATA_PCS_PR1     0x00
```

(3) Specify the command of BSI to be sent. Function **L1D_RF_SetSData_SR1** in **m12195.c** be implemented to perform this action. **SETUP_SR1()** is the first statement in **L1D_RF_SetSData_SR1** which is a macro to hide complicated code to initialize setting 1'st BSI data. The second statement **HWRITE_1_SDATA** is specified the data to be sent by BSI bus. Two parameters of HWRITE_1_SDATA are control word and data word of BSI data. The control word is composed of device selection (0) and data bit count (23 bits). The data word **l1d_rf.RFN_data**, RF N-Counter setting value, is the evaluation result of **L1D_RF_GetRxPLLSetting** called by Layer 1.

```
void  L1D_RF_SetSData_SR1( void )
{
   SETUP_SR1();
   HWRITE_2_SDATA( SCTRL_WORD(0,23), l1d_rf.RFN_data,
                   SCTRL_RESERVED, 0 );
}
```

To create the receiving window with right operation frequency, evaluating the setting value of operation frequency is needed. Function **L1D_RF_GetRxPLLSetting** in file **m12191.c** needs to be implemented to evaluate the synthesizer setting. And Layer 1 calls this function when programs the RF synthesizer setting value and saves the result into variable **l1d_rf.RFN_data**. This content of this variable is the command to be sent to transceiver chip to lock the operation frequency. The evaluation of MT6119C synthesizer N counter setting is as the following code:

```
void  L1D_RF_GetRxPLLSetting( int rf_band, int arfcn, long *rfN, long *ifN )
{  int   Nfrac;

   if(rf_band==FrequencyBand900)
   {  if(arfcn<=70)
      {  if(arfcn<=5)         /*  ARFCN :   0~5    */
         {  Nfrac = 80*(arfcn-0)+4760;
            *rfN = (7L<<16) | (Nfrac<<3) | 1;
         }
         else                 /*  ARFCN :  6~70    */
         {  Nfrac = 80*(arfcn-6)+40;
            *rfN = (8L<<16) | (Nfrac<<3) | 1;
         }
      }
      else
      {  if(arfcn<=124)       /*  ARFCN : 71~124   */
         {  Nfrac = 80*(arfcn-71)+40;
            *rfN = (9L<<16) | (Nfrac<<3) | 1;
         }
         else                 /*  ARFCN : 975~1023 */
         {  Nfrac = 80*(arfcn-975)+840;
            *rfN = (7L<<16) | (Nfrac<<3) | 1;
```

```
                }
            }
            *ifN = 0;
        }
        if(rf_band==FrequencyBand1800)
        {           :
                    :
        }
        if(rf_band==FrequencyBand1900)
        {           :
                    :
        }
    }
```

2.8.2.2 Set Transceiver Gain

100 QB ahead of R0, Band selection and Rx gain setting should be performed. MT62xx activates control pins by changing BPI bus states, and also sends serial command to set transceiver gain by BSI bus. To perform these operations, the timing and data of BPI and BSI bus can be implemented as follow steps:

(1) Modified the timing definition in **l1d_custom_rf.h**. **QB_SR2** is the timing definition of 2'nd BSI command of RX window. **QB_PR2** is the timing definition of 2'nd BPI bus status changing of RX window.

```
#define  QB_SR2          100
#define  QB_PR2          100
```

(2) Define the data of BPI bus states for each band. The state of BPI of each band is shown in the table:

BPI_BUS	7	6	5	4	3	2	1	0	Hex
Function	reserved	GSM BAND_SW	DCS BAND_SW	TX_EN	reserved	PCS_RCTRL	LB_TX	HB_TX	
GSM	X	0	0	0	X	0	0	0	00h
DCS	X	0	0	0	X	0	0	0	00h
PCS	X	0	0	0	X	1	0	0	04h

So the BPI data can be defined as follow:

```
#define  PDATA_GSM_PR2    0x00
#define  PDATA_DCS_PR2    0x00
#define  PDATA_PCS_PR2    0x04
```

(3) Specify the command of BSI to be sent. Function **L1D_RF_SetSData_SR2** in **m12195.c** should be implemented to perform this action. The first statement in **L1D_RF_SetSData_SR2** is **SETUP_SR2()** which is a macro to hide complicated code to initialize setting 2'nd BSI data. The second statement **HWRITE_2_SDATA** is specified the data to be sent by BSI bus. The parameters of **HWRITE_2_SDATA** are control word and data word of BSI data. The control word is composed of device selection (0) and data bit count (23 bits). The data word **l1d_rf.AGC_data**, transceiver gain setting value, is the evaluation result of **L1D_RF_GetGainSetting** called by Layer 1.

```
void  L1D_RF_SetSData_SR2( void )
{
   SETUP_SR2();
   HWRITE_1_SDATA( SCTRL_WORD(0,23), l1d_rf.AGC_data );
}
```

To create the receiving window with suitable gain of receiving amplifier, evaluating the setting value of request gain is needed. Function **L1D_RF_GetGainSetting** in file **m12192.c** needs to be implemented to evaluate the transceiver gain setting. And Layer 1 calls this function when programs the gain setting value and saves the result into variable **l1d_rf.AGC_data**. This content of this variable is the command to be sent to transceiver chip to set the requested receive amplifier gain. The evaluation of MT6119C receiving amplifier gain setting is as the following code:

```
int   L1D_RF_GetGainSetting( int rf_band, int arfcn, int request_gain,
                             long *gain_setting )
{  int   max_gain, min_gain, real_gain;
   const int*    d32ptr;
   const sL1DAGCDATA*  agcptr;
   long  setting;
   int   bit_no;

   /* evaluate the range of available gain */
   d32ptr  = (int*)&(GAIN_RANGE_TABLE[rf_band]);
   max_gain = *d32ptr++;
   min_gain = *d32ptr;

   /* clipping the request gain to the avialable gain */
   if( request_gain>=max_gain )
   {  request_gain = max_gain;  }
   else  if( request_gain<=min_gain )
   {  request_gain = min_gain;  }

   /* evaluate the real gain setting */
   agcptr = AGC_TABLE[rf_band];
   if( request_gain < agcptr->pos_gain )
   {  agcptr++;
      if( request_gain < agcptr->pos_gain )
      {  agcptr++;  }
   }
   {  bit_no    = BIT_NO( request_gain, agcptr->A, GC_B );
      real_gain = REAL_GAIN( bit_no, agcptr->A, GC_B );
      setting   = agcptr->setting + (bit_no<<5);
   }

   *gain_setting = setting;
   return( real_gain );
}
```

2.8.2.3 Set Transceiver in Idle Mode

6 QB behind R1, RF module finishes receiving data and MT62xx set it into idle mode. MT62xx Baseband activates control pins by changing BPI bus states, and also sends serial command to set idle mode command by BSI bus. To perform these operations, the timing and data of BPI and BSI bus can be implemented as follow steps:

(1) Modified the timing definition in **l1d_rf.h**. **QB_SR3** is the timing definition of 3'rd BSI command of RX window. **QB_PR3** is the timing definition of 3'rd BPI bus status changing of RX window.

```
#define   QB_SR3           6
#define   QB_PR3           6
```

(2) Define the data of BPI bus states for each band. The state of BPI of each band is shown in the table:

BPI_BUS	7	6	5	4	3	2	1	0	Hex
Function	reserved	GSM BAND_SW	DCS BAND_SW	TX_EN	reserved	PCS_RCTRL	LB_TX	HB_TX	
GSM	X	0	0	0	X	0	0	0	00h
DCS	X	0	0	0	X	0	0	0	00h
PCS	X	0	0	0	X	0	0	0	00h

So the BPI data can be defined as follow:

```
#define   PDATA_GSM_PR3    0x00
#define   PDATA_DCS_PR3    0x00
#define   PDATA_PCS_PR3    0x00
```

(3) Specify the command of BSI to be sent. Function **L1D_RF_SetSData_SR3** in **m12195.c** should be implemented to perform this action. The first statement in **L1D_RF_SetSData_SR3** is **SETUP_SR3()** which is a macro to hide complicated code to initialize setting 3'rd BSI data. The second statement **HWRITE_1_SDATA** is specified the data to be sent by BSI bus. The parameters of **HWRITE_1_SDATA** are control word and data word of BSI data. The control word is composed of device selection (0) and data bit count (23 bits). The data word 124992h is the command to set MT6119C into Idle mode.

```
void  L1D_RF_SetSData_SR3( void )
{
   SETUP_SR3();
   HWRITE_1_SDATA( SCTRL_WORD(0,23), 0x124992 );
}
```

2.8.2.4 RX ADC Control

Besides setting the timing and data of BPI and BSI bus of MT62xx Baseband to create a RX window, the timing setting for RX frame enable and frame synchronization control of MT62xx are needed. RX frame enable is used to enable or disable RX ADC. RX frame synchronization is used to turn on/off RX SPORT. The timing of enabling RX ADC to turning on RX SPORT named **QB_RX_FENA_2_FSYNC** and the timing of turning off RX SPORT to disabling RX ADC named **QB_RX_FSYNC_2_FENA** are both also needed to be defined in **l1d_custom_rf.h**. In this case, the timing of enabling RX ADC to turning on RX SPORT is recommend to be 35 QB and the timing of turning off RX SPORT to disabling RX ADC is 0 QB. So **QB_RX_FENA_2_FSYNC** and **QB_RX_FSYNC_2_FENA** can be defined as:

```
#define  QB_RX_FENA_2_FSYNC     35
#define  QB_RX_FSYNC_2_FENA     0
```

2.8.2.5 Events Activation

If Layer1 plans to receive 156-bit data from TDMA timer count 256 QB to 880 QB, the above setting applies to create the RX window.

(1) At TDMA timer count is 6 QB (250 QB ahead of RX SPORT turn on), BPI bus changes its states to 00h .And after 2 QB (248 QB), the serial command of setting operation frequency is sent to MT6119C transceiver.

(2) At TDMA timer count is 156 QB (100 QB ahead of RX SPORT turn on), BPI bus changes its states to 00h(GSM/DCS) or 04h(PCS) and the command of setting RX gain is sent to MT6119C transceiver.

(3) At TDMA timer count is 221 QB (35 QB ahead of RX SPORT turn on), RX ADC in MT62xx is enabled.

(4) At TDMA timer count is 256 QB, RX SPORT in MT62xx is turned on. The I/Q data begin to be sampled.

(5) At TDMA timer count is 380 QB, RX SPORT in MT62xx is turned off and RX ADC is also disabled. The I/Q data stop sampling.

(6) At TDMA timer count is 880 QB (at RX SPORT turning off), the command of setting MT6119C into idle mode is sent.

(7) At TDMA timer count is 886 QB (6 QB behind RX SPORT turning off), BPI bus changes its states to 00h .

2.8.3 Create TX window

Another example is creating TX window to transmit data. The RF setting and timing of MT6119C RF module can be depicted in the below diagram:

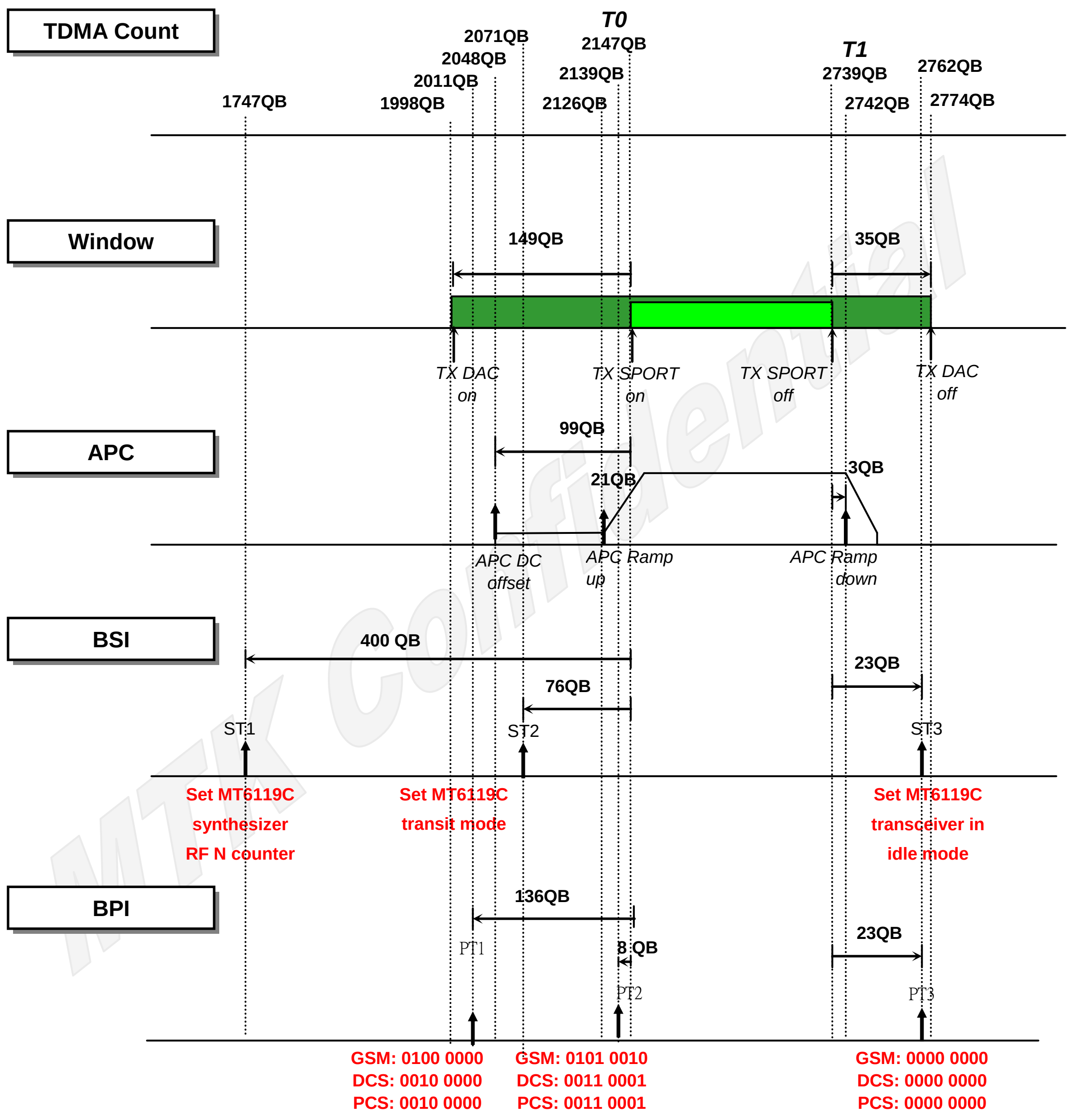


Figure 9 Event's timing of TX window

The TDMA timing and data setting are described in 2 parts. One part is that before turning on TX SPORT to receiving I/Q data from MT6119C transceiver. And the other part is after turning off TX SPORT to finish transmission I/Q data. To describe these two parts easily, the timing of turning on TX SPORT is taken as one base named **T0**. And the timing of turning off TX SPORT is taken as one base named **T1**. In this case, T0 is at TDMA counter 2147 QB and T1 is at TDNA counter 2739 QB.

2.8.3.1 Set Synthesizer N-Counter

400 QB ahead of T0, RF module starts to work. MT62xx sends serial command to lock MT6119C synthesizer to lock the operation frequency by BSI bus. To perform this operation, the timing and data of BSI bus can be implemented as follow steps:

(1) Modified the timing definition in **l1d_custom_rf.h**. **QB_ST1** is the timing definition of 1'st BSI command of TX window.

```
#define  QB_ST1           400
```

(2) Specify the command of BSI to be sent. Function **L1D_RF_SetSData_ST1** in **m12195.c** should be implemented to perform this action. **SETUP_ST1()** is the first statement in **L1D_RF_SetSData_ST1** which is a macro to hide complicated code to initialize setting 1'st BSI data. The second statement **HWRITE_2_SDATA** is specified the data to be sent by BSI bus. Two parameters of **HWRITE_2_SDATA** are control word and data word of BSI data. The control word is composed of device selection (0) and data bit count (23 bits). The data word **l1d_rf.RFN_data**, synthesizer N-counter setting value, is the evaluation result of **L1D_RF_GetTXPLLSetting** called by Layer 1. The second command is set the TX low pass filter gain, the value is 0x13D504 for GSM band and 0x13DF04 for DCS/PCS band.

```
void  L1D_RF_SetSData_ST1( void )
{
   SETUP_ST1();
   if(l1d.rf_band==FrequencyBand900)
   {   HWRITE_2_SDATA( SCTRL_WORD(0,23), l1d_rf.RFN_data,
                       SCTRL_WORD(0,23), 0x13D504 );
   }

if((l1d.rf_band==FrequencyBand1800)||(l1d.rf_band==FrequencyBand1900))
   {   HWRITE_2_SDATA( SCTRL_WORD(0,23), l1d_rf.RFN_data,
                       SCTRL_WORD(0,23), 0x13DF04 );
   }
}
```

To create the transmission window with right operation frequency, evaluating the setting value of operation frequency is needed. Function **L1D_RF_GetTxPLLSetting** in file **m12191.c** needs to be implemented to evaluate the synthesizer setting. And Layer 1 calls this function when program the RF synthesizer setting value and saves the result into variable **l1d_rf.RFN_data**. This content of this variable is the command to be sent to transceiver chip to lock the operation frequency. The function of MT6119C RF module can be implemented as follow:

```
void  L1D_RF_GetTxPLLSetting( int rf_band, int arfcn, long *rfN, long *ifN )
{  int   Nfrac;

   if( rf_band==FrequencyBand900 )
   {
      if(arfcn<=104)
      {  if(arfcn<=45)
         {  if(arfcn<=36)      /* ARFCN : 0~36       */
            {  Nfrac = 88*(arfcn-0)+1600;
               *rfN = (11L<<16) | (Nfrac<<3) | 0x400001L;
               mt6119_d_flag = 1;
            }
            else               /* ARFCN : 37~45     */
            {  Nfrac = 90*(arfcn-37)+3430;
               *rfN = (13L<<16) | (Nfrac<<3) | 0x400001L;
               mt6119_d_flag = 0;
            }
         }
         else
         {  if(arfcn<=95)      /* ARFCN : 46~95     */
            {  Nfrac = 88*(arfcn-46)+448;
               *rfN = (12L<<16) | (Nfrac<<3) | 0x400001L;
               mt6119_d_flag = 1;
            }
            else               /* ARFCN : 96~104    */
            {  Nfrac = 90*(arfcn-96)+3540;
               *rfN = (14L<<16) | (Nfrac<<3) | 0x400001L;
               mt6119_d_flag = 0;
            }
         }
      }
      else
      {  if(arfcn<=1001)
         {  if(arfcn<=124)     /* ARFCN : 105~124   */
            {  Nfrac = 88*(arfcn-105)+440;
               *rfN = (13L<<16) | (Nfrac<<3) | 0x400001L;
               mt6119_d_flag = 1;
            }
            else               /* ARFCN : 975~1001  */
            {  Nfrac = 88*(arfcn-975)+2488;
               *rfN = (10L<<16) | (Nfrac<<3) | 0x400001L;
               mt6119_d_flag = 1;
            }
         }
         else
         {  if(arfcn<=1010)    /* ARFCN : 1002~1010 */
            {  Nfrac = 90*(arfcn-1002)+3320;
               *rfN = (12L<<16) | (Nfrac<<3) | 0x400001L;
               mt6119_d_flag = 1;
            }
            else               /* ARFCN : 1011~1023 */
            {  Nfrac = 88*(arfcn-1011)+456;
               *rfN = (11L<<16) | (Nfrac<<3) | 0x400001L;
               mt6119_d_flag = 0;
            }
         }
         *ifN = 0;
      }
   }
   if( rf_band==FrequencyBand1800 )
   {           :
               :
   }
   if( rf_band==FrequencyBand1900 )
   {           :
               :
   }
}
```

2.8.3.2 Set Transceiver in Transmit Mode

136 QB ahead of T0, TX mode setting is performed. MT62xx Baseband activates control pins by changing BPI bus states, and after 60QB also sends serial command to set transmit mode by BSI bus. To perform these operations, the timing and data of BPI and BSI bus can be implemented as follow steps:

(1) Modified the timing definition in **l1d_custom_rf.h**. QB_ST2 is the timing definition of 2'nd BSI command of TX window. QB_PT1 is the timing definition of 1'st BPI bus status changing of TX window.

```
#define  QB_ST2          76
#define  QB_PT1          136
```

(2) Define the data of BPI bus states of each band. The data is shown in the below table:

BPI_BUS	7	6	5	4	3	2	1	0	Hex
Function	reserved	GSM BAND_SW	DCS BAND_SW	TX_EN	reserved	PCS_RCTRL	LB_TX	HB_TX	
GSM	X	1	0	0	X	0	0	0	40h
DCS	X	0	1	0	X	0	0	0	20h
PCS	X	0	1	0	X	0	0	0	20h

So the BPI data can be defined as follow:

```
#define  PDATA_GSM_PT1    0x40
#define  PDATA_DCS_PT1    0x20
#define  PDATA_PCS_PT1    0x20
```

(3) Specify the command of BSI to be sent. Function **L1D_RF_SetSData_ST2** in **m12195.c** should be implemented to perform this action. **SETUP_ST2()** is the first statement in **L1D_RF_SetSData_ST2** which is a macro to hide complicated code to initialize setting 2'nd BSI data. The second statement **HWRITE_1_SDATA** is specified the data to be sent by BSI bus. Two parameters of **HWRITE_1_SDATA** are control word and data word of BSI data. The control word is composed of device selection (0) and data bit count (23 bits). The data word is corresponding with the band. **l1d_rf.band** is the global variable which indicates the band of operation frequency.

```
const unsigned long SDATA_TXBAND_TABLE[5][2] =
{
   {       0L,       0L },  /* FrequencyBand400  (not support)   */
   {       0L,       0L },  /* FrequencyBand850  (not support)   */
   {0x128992L,0x138992L },  /* FrequencyBand900                  */
   {0x129192L,0x139192L },  /* FrequencyBand1800                 */
   {0x129992L,0x139992L },  /* FrequencyBand1900                 */
};
void  L1D_RF_SetSData_ST2( void )
{
   SETUP_ST2();
   HWRITE_1_SDATA( SCTRL_WORD(0,23),
                   SDATA_TXBAND_TABLE[l1d_rf.band][mt6119_d_flag] );
}
```

2.8.3.3 TX DAC Control

The timing setting for TX frame enable and frame synchronization control of MT62xx are needed. TX frame enable is used to enable or disable TX DAC. TX frame synchronization is used to turn on/off TX SPORT. The timing of enabling TX DAC to turning on TX SPORT named **QB_TX_FENA_2_FSYNC** and the timing of turning off TX SPORT to disabling TX DAC named **QB_TX_FSYNC_2_FENA** are both also needed to be defined in **l1d_custom_rf.h**. In this case, the timing of enabling TX DAC to turning on TX SPORT is 149 QB and the timing of turning off TX SPORT to disabling TX DAC is 34 QB. So **QB_TX_FENA_2_FSYNC** and **QB_TX_FSYNC_2_FENA** can be defined as:

```
#define  QB_TX_FENA_2_FSYNC    149
#define  QB_TX_FSYNC_2_FENA    35
```

2.8.3.4 Set APC

50 QB ahead of T0, APC DAC output performs a DC offset to let ramp up more smoothly. The timing of performing this DC offset is defined as **QB_APCDACON** in **l1d_custom_rf.h**. The value of APC DC offset is maintained in APC profile and it is introduced in the latter section.

```
#define  QB_APCDACON    99
```

41 QB ahead of T0, MT62xx Baseband changes BPI bus states to select operation right band.

(1) Modified the timing definition in **l1d_custom_rf.h**. **QB_PT2** is the timing definition of 2'nd BPI bus status changing of TX window.

```
#define  QB_PT2            8
```

(2) Define the data of BPI bus states of each band. The state of BPI of each band is shown in the table:

BPI_BUS	7	6	5	4	3	2	1	0	Hex
Function	reserved	GSM BAND_SW	DCS BAND_SW	TX_EN	reserved	PCS_RCTRL	LB_TX	HB_TX	
GSM	X	1	0	1	X	0	1	0	52h
DCS	X	0	1	1	X	0	0	1	31h
PCS	X	0	1	1	X	0	0	1	31h

So the BPI data can be defined as follow:

```
#define  PDATA_GSM_PT2    0x52
#define  PDATA_DCS_PT2    0x31
#define  PDATA_PCS_PT2    0x31
```

22 QB ahead of T0, APC DAC output performs ramping up. The timing of performing this ramping up is defined as **QB_APCON** in **l1d_custom_rf.h**.

```
#define  QB_APCON    21
```

3 QB behind T0 , APC DAC output performs ramping down. The timing of performing this ramping down is defined as **QB_APCOFF** in **l1d_custom_rf.h**.

```
#define  QB_APCOFF    18
```

2.8.3.5 Set Transceiver in Idle Mode

33 QB behind T1, RF module finishes transmission data and MT62xx set it into idle mode. MT62xx Baseband activates control pins by changing BPI bus states, and also sends serial command to set idle mode command by BSI bus. To perform these operations, the timing and data of BPI and BSI bus can be implemented as follow steps:

(1) Modified the timing definition in **l1d_custom_rf.h**. **QB_ST3** is the timing definition of 3'rd BSI command of TX window. **QB_PT3** is the timing definition of 3'rd BPI bus status changing of TX window.

```
#define  QB_ST3            23
#define  QB_PT3            23
```

(2) Define the data of BPI bus states of each band. The state of BPI of each band is shown in the table:

BPI_BUS	7	6	5	4	3	2	1	0	Hex
Function	reserved	GSM BAND_SW	DCS BAND_SW	TX_EN	reserved	PCS_RCTRL	LB_TX	HB_TX	
GSM	X	0	0	0	X	0	0	0	00h
DCS	X	0	0	0	X	0	0	0	00h
PCS	X	0	0	0	X	0	0	0	00h

So the BPI data can be defined as follow:

```
#define  PDATA_GSM_PT3    0x00
#define  PDATA_DCS_PT3    0x00
```

```
#define  PDATA_PCS_PT3    0x00
```

(3) Specify the command of BSI to be sent. Function **L1D_RF_SetSData_ST3** in **m12195.c** should be implemented to perform this action. The first statement in **L1D_RF_SetSData_ST3 is SETUP_ST3()** which is a macro to hide complicated code to initialize setting 3'rd BSI data. The second statement **HWRITE_1_SDATA** is specified the data to be sent by BSI bus. The parameters of **HWRITE_1_SDATA** are control word and data word of BSI data. The control word is composed of device selection (0) and data bit count (23 bits). The data word 0x124992 is the command to set MT6119C into Idle mode.

```
void  L1D_RF_SetSData_ST3( void )
{
   SETUP_ST2();
   HWRITE_1_SDATA( SCTRL_WORD(0,23), 0x124992 );
}
```

The propagation delay when transmit data from Baseband uplink DAC to antenna should be claimed in file **l1d_custom_rf.c**. This delay depends on different RF module. For MT6119C RF module, the delay is 29 QB

```
const  short  TxPropagationDelay = 36;
```

2.8.3.6 Events Activation

If Layer 1 plans to transmit 148-bit data from TDMA timer count 2147 QB to 2739 QB, the above setting applies to create the TX window.

(1) At TDMA timer count is 1747 QB (400 QB ahead of TX SPORT turn on), the command of setting operation frequency is sent.
(2) At TDMA timer count is 1998 QB (149 QB ahead of TX SPORT turn on), TX DAC is enabled.
(3) At TDMA timer count is 2011 QB (136 QB ahead of TX SPORT turn on), BPI bus changes its states to TX mode.
(4) At TDMA timer count is 2048 QB (99 QB ahead of TX SPORT turn on), APC DAC output performs a DC offset to let ramp up more smoothly.
(5) At TDMA timer count is 2071 QB (76 QB ahead of TX SPORT turn on), 3-wire command to set transmit mode is sent.
(6) At TDMA timer count is 2126 QB (21 QB ahead of TX SPORT turn on), APC DAC output performs ramping up.
(7) At TDMA timer count is 2139 QB (8 QB ahead of TX SPORT turn on), BPI bus changes its states to select operation right band.
(8) At TDMA timer count is 2147 QB, TX SPORT in MT62xx is turned on.
(9) At TDMA timer count is 2739 QB, TX SPORT in MT62xx is turned off.
(10) At TDMA timer count is 2742 QB (3 QB behind TX SPORT turning off), APC DAC output performs ramping down.
(11) At TDMA timer count is 2762 QB (23 QB behind TX SPORT turning off), BPI bus changes its states and send 3-wire command to set MT6119C into idle mode is sent.
(12) At TDMA timer count is 2774 QB (35 QB behind TX SPORT turn off), TX DAC is disabled at this time.

2.9 RF Control Configuration of MT6139C RF Module

For example, the schematic circuit interface between MT6139C RF module and MT62xx chip is shown in below diagram. Please refer to MediaTek MT6139C RF Module schematic circuit to get more detail information.

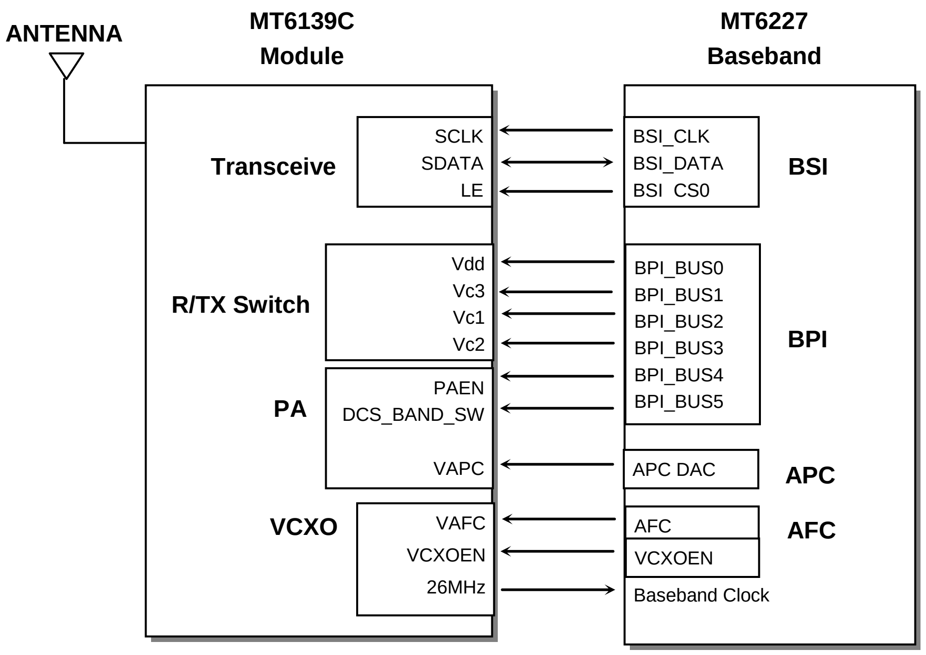


Figure 10 The connection of MT62xx and MT6139C RF Module

BSI_CLK, BSI_DATA, BSI_CS0 pins of MT62xx are directly connected to the serial interface of MT6139C transceiver. APC_DAC pin of MT62xx is connected to the VAPC pin of PA of RF module board. VCXOEN of MT62xx controls the enable state of VCXO. AFC DAC output is connected to the voltage control pin VAFC of VCXO. The generated clock of VCXO provides MT62xx system clock. Band-related pins Vdd, Vc3, Vc1, Vc2, PAEN, DCS_BAND_SW are controlled by the bit 0,1,2,3,4,5 of BPI bus of MT62xx. The high/low state of each pin of BPI bus and their timing of changing state can be programmed by MT62xx. The RF radio control method is introduced in the latter description.

2.9.1 Initialization

RF module can be controlled by the BSI, BPI, APC, AFC units of MT62xx chip. BSI, BPI, APC, AFC units can work in 2 modes. One mode is immediate control mode, and the other is TDMA event driven control mode.

Immediate control mode is that the output signal of BSI, BPI, APC, or AFC unit is generated as soon as setting the corresponding registers in MT62xx. When turning on the RF module, the immediate mode can be used to initialize the RF module. For example, immediate control mode is used when initializing MT6139C RF module, and the initial states are set as follow:

(1) Vdd (BPI_BUS0) is low state
(2) Vc3 (BPI_BUS1) is low state
(3) Vc1 (BPI_BUS2) is low state
(4) Vc2 (BPI_BUS3) is low state
(5) PAEN (BPI_BUS4) is low state

(6) DCS_BAND_SW (BPI_BUS5) is low state

(7) Serial commands (0x0024080, 0x0002802, 0x1B4E7FB, 0x01C5957, 0x0003002) are sending to MT6139C transceiver.

The initialization of BPI, BSI data can be implemented in **L1D_RF_PowerOn** in **m12196.c** by using immediate mode:

```
void  L1D_RF_PowerOn( void )
{
   IMM_MODE_BEGIN( IMMMASK_ALL );
   IMM_SEND_BPI( 0x0000 );
   IMM_SEND_BSI( SCTRL_IMOD(0,26), 0x0024080 );
   IMM_SEND_BSI( SCTRL_IMOD(0,26), 0x0002802 );
   IMM_SEND_BSI( SCTRL_IMOD(0,26), 0x1B4E7FB );
   IMM_SEND_BSI( SCTRL_IMOD(0,26), 0x01C5957 );
   IMM_SEND_BSI( SCTRL_IMOD(0,26), 0x0003002 );
   IMM_MODE_END(  );
}
```

2.9.2 Create RX window

TDMA timer is a counter unit that counts 1 per quarter-bit (0.923u). It always repeats counting from 0 to 4999. So the period of TDMA counter is 5000 QB (1 TDMA frame). If BSI, BPI, AFC, APC are programmed in TDMA event driven mode, the active time can be programmed. When the TDMA counter counts to the programmed time (count), the corresponding unit is triggered to activate.

If it is needed to receive data from air or transmit data to air, Layer1 needs to plan all RF setting and timing to create RX/TX window, control receiving gain, control VCXO, control PA... etc. For this purpose, TDMA event driven mode of BSI, BPI, APC, or AFC units should be used.

For example of creating RX window to receive data, the RF setting and timing of MT6139C RF module can be depicted in the below diagram:

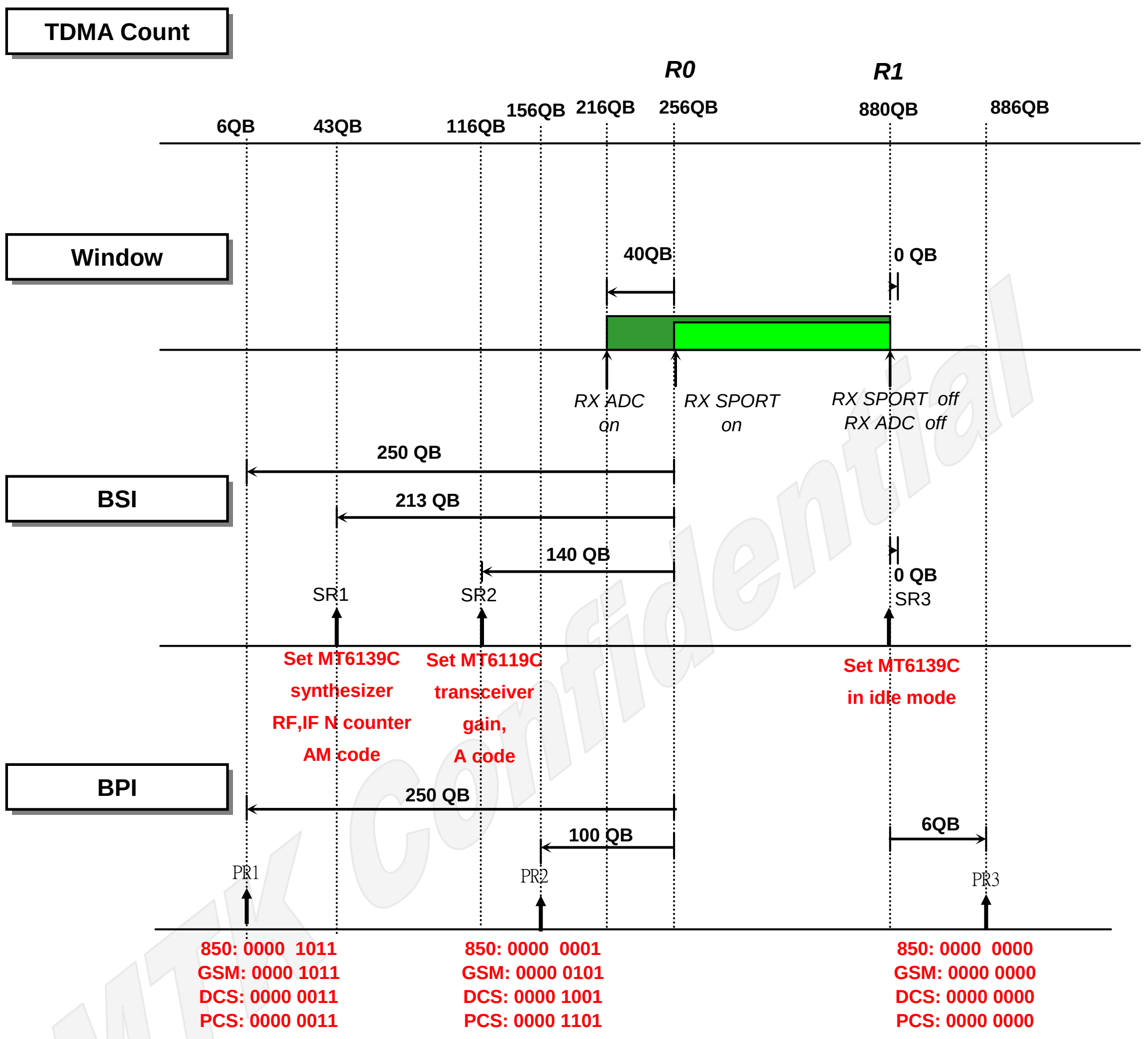


Figure 11 MT6139C Event's timing of RX window

The TDMA timing and data setting are described in 2 parts. One part is that before turning on RX SPORT to receiving I/Q data from MT6139C transceiver. And the other part is after turning off RX SPORT to finish receiving I/Q data. To describe these two parts easily, the timing of turning on RX SPORT is taken as one base named **R0**. And the timing of turning off RX SPORT is taken as one base named **R1**. In this case, R0 is at TDMA counter 256 QB and R1 is at TDMA counter 880 QB.

2.9.2.1 Set Synthesizer N-Counter

250 QB ahead of R0, RF module should start to operate. MT62xx chip controls the RF module by activating control pins and sending serial command. MT62xx activates the control pins by changing BPI bus states. The high/low state of BPI bus at this time depends on operation band. After 37 QB, MT62xx also sends serial command to lock MT6139C synthesizer to lock the operation frequency by BSI bus. To perform these operations, the timing and data of BPI and BSI bus can be implemented as follow steps:

(1) Modified the timing definition in **l1d_custom_rf.h**. **QB_SR1** is the timing definition of 1'st BSI command of RX window. **QB_PR1** is the timing definition of 1'st BPI bus status changing of RX window.

```
#define  QB_SR1          250
#define  QB_PR1          213
```

(2) Define the states of BPI bus for each band. The data is shown in the below table:

BPI_BUS	7	6	5	4	3	2	1	0	Hex
Function	reserved	reserved	DCS BAND_SW	PAEN	Vc2	Vc1	Vc3	Vdd	
850	X	X	0	0	1	0	1	1	0Bh
GSM	X	X	0	0	1	0	1	1	0Bh
DCS	X	X	0	0	0	0	1	1	03h
PCS	X	X	0	0	0	0	1	1	03h

So the BPI data can be defined as follow:

```
#define  PDATA_GSM850_PR1      0x0B
#define  PDATA_GSM_PR1         0x0B
#define  PDATA_DCS_PR1         0x03
#define  PDATA_PCS_PR1         0x03
```

(3) Specify the command of BSI to be sent. Function **L1D_RF_SetSData_SR1** in **m12195.c** be implemented to perform this action. **SETUP_SR1()** is the first statement in **L1D_RF_SetSData_SR1** which is a macro to hide complicated code to initialize setting 1'st BSI data. The second statement **HWRITE_3_SDATA** is specified the data to be sent by BSI bus. Two parameters of HWRITE_1_SDATA are control word and data word of BSI data. The control words **SCTRL_WARM**, **SCTRL_PLL** and **SCTRL_AMCODE** are composed of device selection (0) and data bit count (26 bits). The 1st data word **SDATA_WARM** is the command to set MT6139C into Warm-up mode. The 2nd data word **l1d_rf.RFN_data**, RF N-Counter setting value, is the evaluation result of **L1D_RF_GetRxPLLSetting** called by Layer 1. The 3rd data word **RFSpecialCoef.rx.mt6139ip2.data[l1d_rf.band].amcode** is MT6139C IP2 coefficient which is provided by factory mode calibration. The 3rd data word is corresponding with the band. **l1d_rf.band** is the global variable which indicates the band of operation frequency.

```
/*MT6139C*/ #define  SCTRL_WARM                    SCTRL_WORD(0,26)
/*MT6139C*/ #define  SCTRL_PLL                     SCTRL_WORD(0,26)
/*MT6139C*/ #define  SCTRL_AMCODE                  SCTRL_WORD(0,26)
/*MT6139C*/ const unsigned long SDATA_WARM   = 0x0000402L;
void  L1D_RF_SetSData_SR1( void )
{
   SETUP_SR1();
   HWRITE_3_SDATA(  SCTRL_WARM, SDATA_WARM
                    SCTRL_PLL,  l1d_rf.RFN_data,
                    SCTRL_AMCODE,
                    RFSpecialCoef.rx.mt6139ip2.data[l1d_rf.band].amcode<<4|(0x000000A)
                  );
}
```

To create the receiving window with right operation frequency, evaluating the setting value of operation frequency is needed. Function **L1D_RF_GetRxPLLSetting** in file **m12191.c** needs to be implemented to evaluate the synthesizer setting. And Layer 1 calls this function when programs the RF synthesizer setting value and saves the result into variable **l1d_rf.RFN_data**. This content of this variable is the command to be sent to transceiver chip to lock the operation frequency. The evaluation of MT6139C synthesizer N counter setting is as the following code:

```
void  L1D_RF_GetRxPLLSetting( int rf_band, int arfcn, long *rfN, long *ifN )
/*MT6139C*/{
/*MT6139C*/   switch(rf_band)
/*MT6139C*/   {
/*MT6139C*/      case  FrequencyBand850 :
/*MT6139C*/      {
/*MT6139C*/         if(arfcn<=201)
/*MT6139C*/          {  if(arfcn<=136)
/*MT6139C*/             {                                 /* ARFCN : 128~136     */
/*MT6139C*/                 *rfN = (66L<<12) | ((arfcn-72)<<5) | 0x000001L;
/*MT6139C*/             }
/*MT6139C*/             else
/*MT6139C*/             {                               /* ARFCN : 137~201     */
/*MT6139C*/                 *rfN = (67L<<12) | ((arfcn-137)<<5) | 0x000001L;
/*MT6139C*/             }
/*MT6139C*/          }
/*MT6139C*/          else
/*MT6139C*/          {                                  /* ARFCN : 202~251     */
/*MT6139C*/                 *rfN = (68L<<12) | ((arfcn-202)<<5) | 0x000001L;
/*MT6139C*/          }
/*MT6139C*/
/*MT6139C*/          break;
/*MT6139C*/       }
/*MT6139C*/       case  FrequencyBand900 :
/*MT6139C*/       {      :
/*MT6139C*/              :
/*MT6139C*/       }
/*MT6139C*/       case  FrequencyBand1800 :
/*MT6139C*/       {      :
/*MT6139C*/              :
/*MT6139C*/       }
/*MT6139C*/       case  FrequencyBand1900 :
/*MT6139C*/       {      :
/*MT6139C*/              :
/*MT6139C*/       }
/*MT6139C*/       default :
/*MT6139C*/       {
/*MT6139C*/          break;
/*MT6139C*/       }
/*MT6139C*/    }
/*MT6139C*/    *ifN = 0;
/*MT6139C*/}
```

2.9.2.2 Set Transceiver Gain

100 QB ahead of R0, Band selection and Rx gain setting should be performed. MT62xx activates control pins by changing BPI bus states, and also sends serial command to set transceiver gain by BSI bus. To perform these operations, the timing and data of BPI and BSI bus can be implemented as follow steps:

(1) Modified the timing definition in **l1d_custom_rf.h**. **QB_SR2** is the timing definition of 2'nd BSI command of RX window. **QB_PR2** is the timing definition of 2'nd BPI bus status changing of RX window.

```
#define  QB_SR2           140
#define  QB_PR2           100
```

(2) Define the data of BPI bus states for each band. The state of BPI of each band is shown in the table:

BPI_BUS	7	6	5	4	3	2	1	0	Hex
Function	reserved	reserved	DCS BAND_SW	PAEN	Vc2	Vc1	Vc3	Vdd	
850	X	X	0	0	0	0	0	1	01h
GSM	X	X	0	0	0	1	0	1	05h
DCS	X	X	0	0	1	0	0	1	09h
PCS	X	X	0	0	1	1	0	1	0Dh

So the BPI data can be defined as follow:

```
#define  PDATA_GSM850_PR2      0x01
#define  PDATA_GSM_PR2         0x05
#define  PDATA_DCS_PR2         0x09
#define  PDATA_PCS_PR2         0x0D
```

(3) Specify the command of BSI to be sent. Function **L1D_RF_SetSData_SR2** in **m12195.c** should be implemented to perform this action. The first statement in **L1D_RF_SetSData_SR2** is **SETUP_SR2()** which is a macro to hide complicated code to initialize setting 2'nd BSI data. The second statement **HWRITE_1_SDATA** is specified the data to be sent by BSI bus. The parameters of **HWRITE_1_SDATA** are control word and data word of BSI data. The control word **SCTRL_AGC** is composed of device selection (0) and data bit count (26 bits). The data word is composed of two commands. The 1st command **l1d_rf.AGC_data**, transceiver gain setting value, is the evaluation result of **L1D_RF_GetGainSetting** called by Layer 1. The 2nd command **RFSpecialCoef.rx.mt6139ip2.data[l1d_rf.band].acode** is MT6139C IP2 coefficient which is provided by factory mode calibration. The 2nd command is corresponding with the band. **l1d_rf.band** is the global variable which indicates the band of operation frequency. The **RFSpecialCoef.rx.mt6139ip2.rxamcalmode** is used for factory mode calibration to set MT6139C into calibration mode

```
/*MT6139C*/ #define  SCTRL_AGC                    SCTRL_WORD(0,26)
void  L1D_RF_SetSData_SR2( void )
{
   SETUP_SR2();
   if(RFSpecialCoef.rx.mt6139ip2.rxamcalmode == 0)
   {
      HWRITE_1_SDATA( SCTRL_AGC,
                      ((l1d_rf.AGC_data&0xFFFE3FF)|cali_mode)|auto_cal \
                      |(RFSpecialCoef.rx.mt6139ip2.data[l1d_rf.band].acode<<14) );
   }
   else if(RFSpecialCoef.rx.mt6139ip2.rxamcalmode == 1)
   {
      HWRITE_1_SDATA( SCTRL_AGC,
                      l1d_rf.AGC_data \
                      |(RFSpecialCoef.rx.mt6139ip2.data[l1d_rf.band].acode<<14) );
   }
   else
   {
      HWRITE_1_SDATA( SCTRL_AGC,
                      l1d_rf.AGC_data|auto_cal \
                      |(RFSpecialCoef.rx.mt6139ip2.data[l1d_rf.band].acode<<14) );
   }
}
```

To create the receiving window with suitable gain of receiving amplifier, evaluating the setting value of request gain is needed. Function **L1D_RF_GetGainSetting** in file **m12192.c** needs to be implemented to evaluate the transceiver gain setting. And Layer 1 calls this function when programs the gain setting value and saves the result into variable **l1d_rf.AGC_data**. This content of this variable is the command to be sent to transceiver chip to set the requested receive amplifier gain. The evaluation of MT6139C receiving amplifier gain setting is as the following code:

```
   /*MT6139C*/  int    L1D_RF_GetGainSetting( int rf_band, int arfcn, int request_gain,
long *gain_setting )
   /*MT6139C*/ {  int   max_gain, min_gain, real_gain;
   /*MT6139C*/     const int*    d32ptr;
   /*MT6139C*/     const sL1DAGCDATA*  agcptr;
   /*MT6139C*/    long  setting;
   /*MT6139C*/    int   bit_no;
   /*MT6139C*/
   /*MT6139C*/    /* evaluate the range of available gain */
   /*MT6139C*/    d32ptr  = (int*)&(GAIN_RANGE_TABLE[rf_band]);
   /*MT6139C*/    max_gain = *d32ptr++;
   /*MT6139C*/    min_gain = *d32ptr;
   /*MT6139C*/
   /*MT6139C*/    /* clipping the request gain to the avialable gain */
   /*MT6139C*/    if( request_gain>=max_gain )
   /*MT6139C*/    {  request_gain = max_gain;  }
   /*MT6139C*/    else  if( request_gain<=min_gain )
   /*MT6139C*/    {  request_gain = min_gain;  }
   /*MT6139C*/
   /*MT6139C*/    /* evaluate the real gain setting */
   /*MT6139C*/    agcptr = AGC_TABLE[rf_band];
   /*MT6139C*/    if( request_gain < agcptr->pos_gain )
   /*MT6139C*/    {  agcptr++;
   /*MT6139C*/    }
   /*MT6139C*/    {  bit_no    = BIT_NO( request_gain, agcptr->A, GC_B );
   /*MT6139C*/       real_gain = REAL_GAIN( bit_no, agcptr->A, GC_B );
   /*MT6139C*/       setting   = agcptr->setting + (bit_no<<4);
   /*MT6139C*/    }
   /*MT6139C*/
   /*MT6139C*/    *gain_setting = setting;
   /*MT6139C*/    return( real_gain );
   /*MT6139C*/ }
```

2.9.2.3 Set Transceiver in Idle Mode

6 QB behind R1, RF module finishes receiving data and MT62xx set it into idle mode. MT62xx Baseband activates control pins by changing BPI bus states, and also sends serial command to set idle mode command by BSI bus. To perform these operations, the timing and data of BPI and BSI bus can be implemented as follow steps:

(1) Modified the timing definition in **l1d_rf.h**. **QB_SR3** is the timing definition of 3'rd BSI command of RX window. **QB_PR3** is the timing definition of 3'rd BPI bus status changing of RX window.

```
#define  QB_SR3          0
#define  QB_PR3          6
```

(2) Define the data of BPI bus states for each band. The state of BPI of each band is shown in the table:

BPI_BUS	7	6	5	4	3	2	1	0	Hex
Function	reserved	reserved	DCS BAND_SW	PAEN	Vc2	Vc1	Vc3	Vdd	
850	X	X	0	0	0	0	0	0	00h
GSM	X	X	0	0	0	0	0	0	00h
DCS	X	X	0	0	0	0	0	0	00h
PCS	X	X	0	0	0	0	0	0	00h

So the BPI data can be defined as follow:

```
#define  PDATA_GSM850_PR3      0x00
#define  PDATA_GSM_PR3         0x00
#define  PDATA_DCS_PR3         0x00
#define  PDATA_PCS_PR3         0x00
```

(3) Specify the command of BSI to be sent. Function **L1D_RF_SetSData_SR3** in **m12195.c** should be implemented to perform this action. The first statement in **L1D_RF_SetSData_SR3** is **SETUP_SR3()**

which is a macro to hide complicated code to initialize setting 3'rd BSI data. The second statement **HWRITE_1_SDATA** is specified the data to be sent by BSI bus. The parameters of **HWRITE_1_SDATA** are control word and data word of BSI data. The control word SCTRL_IDLE is composed of device selection (0) and data bit count (26 bits). The data word SDATA_IDLE is the command to set MT6139C into Idle mode. The **SCTRL_IDLE** and **SDATA_IDLE** are defined in **l1d_rf.h.** The **RFSpecialCoef.rx.mt6139ip2.rxamcalmode** is used for factory mode calibration to set MT6139C not into idle mode .

```
/*MT6139C*/ #define  SCTRL_IDLE                    SCTRL_WORD(0,26)
/*MT6139C*/ #define  SDATA_IDLE                    0x0002802
/*MT6139C*/ void  L1D_RF_SetSData_SR3( void )
/*MT6139C*/ {
/*MT6139C*/    SETUP_SR3();
/*MT6139C*/    if( RFSpecialCoef.rx.mt6139ip2.rxamcalmode == 0 )
/*MT6139C*/    {
/*MT6139C*/       HWRITE_1_SDATA( SCTRL_RESERVED, 0 );
/*MT6139C*/    }
/*MT6139C*/    else
/*MT6139C*/    {
/*MT6139C*/       HWRITE_1_SDATA( SCTRL_IDLE, SDATA_IDLE );
/*MT6139C*/    }
/*MT6139C*/ }
```

2.9.2.4 RX ADC Control

Besides setting the timing and data of BPI and BSI bus of MT62xx Baseband to create a RX window, the timing setting for RX frame enable and frame synchronization control of MT62xx are needed. RX frame enable is used to enable or disable RX ADC. RX frame synchronization is used to turn on/off RX SPORT. The timing of enabling RX ADC to turning on RX SPORT named **QB_RX_FENA_2_FSYNC** and the timing of turning off RX SPORT to disabling RX ADC named **QB_RX_FSYNC_2_FENA** are both also needed to be defined in **l1d_custom_rf.h**. In this case, the timing of enabling RX ADC to turning on RX SPORT is recommend to be 40 QB and the timing of turning off RX SPORT to disabling RX ADC is 0 QB. So **QB_RX_FENA_2_FSYNC** and **QB_RX_FSYNC_2_FENA** can be defined as:

```
#define  QB_RX_FENA_2_FSYNC     40
#define  QB_RX_FSYNC_2_FENA     0
```

2.9.2.5 Events Activation

If Layer1 plans to receive 156-bit data from TDMA timer count 256 QB to 880 QB, the above setting applies to create the RX window.

(1) At TDMA timer count is 6 QB (250 QB ahead of RX SPORT turn on), BPI bus changes its states to 0Bh(GSM850/GSM) or 03h(DCS/PCS) .And after 37 QB (213 QB), the serial command of setting operation frequency is sent to MT6139C transceiver.

(2) At TDMA timer count is 116 QB (140 QB ahead of RX SPORT turn on), the serial command of setting RX gain is sent to MT6139C transceiver

(3) At TDMA timer count is 156 QB (100 QB ahead of RX SPORT turn on), BPI bus changes its states to 01h(GSM850) or 01h(GSM) or 09h(DCS) or 0Dh (PCS).

(3) At TDMA timer count is 216 QB (40 QB ahead of RX SPORT turn on), RX ADC in MT62xx is enabled.

(4) At TDMA timer count is 256 QB, RX SPORT in MT62xx is turned on. The I/Q data begin to be sampled.

(5) At TDMA timer count is 880 QB, RX SPORT in MT62xx is turned off and RX ADC is also disabled. The I/Q data stop sampling.

(6) At TDMA timer count is 880 QB (at RX SPORT turning off), the command of setting MT6139C into idle mode is sent.

(7) At TDMA timer count is 886 QB (6 QB behind RX SPORT turning off), BPI bus changes its states to 00h .

2.9.3 Create TX window

Another example is creating TX window to transmit data. The RF setting and timing of MT6139C RF module can be depicted in the below diagram:

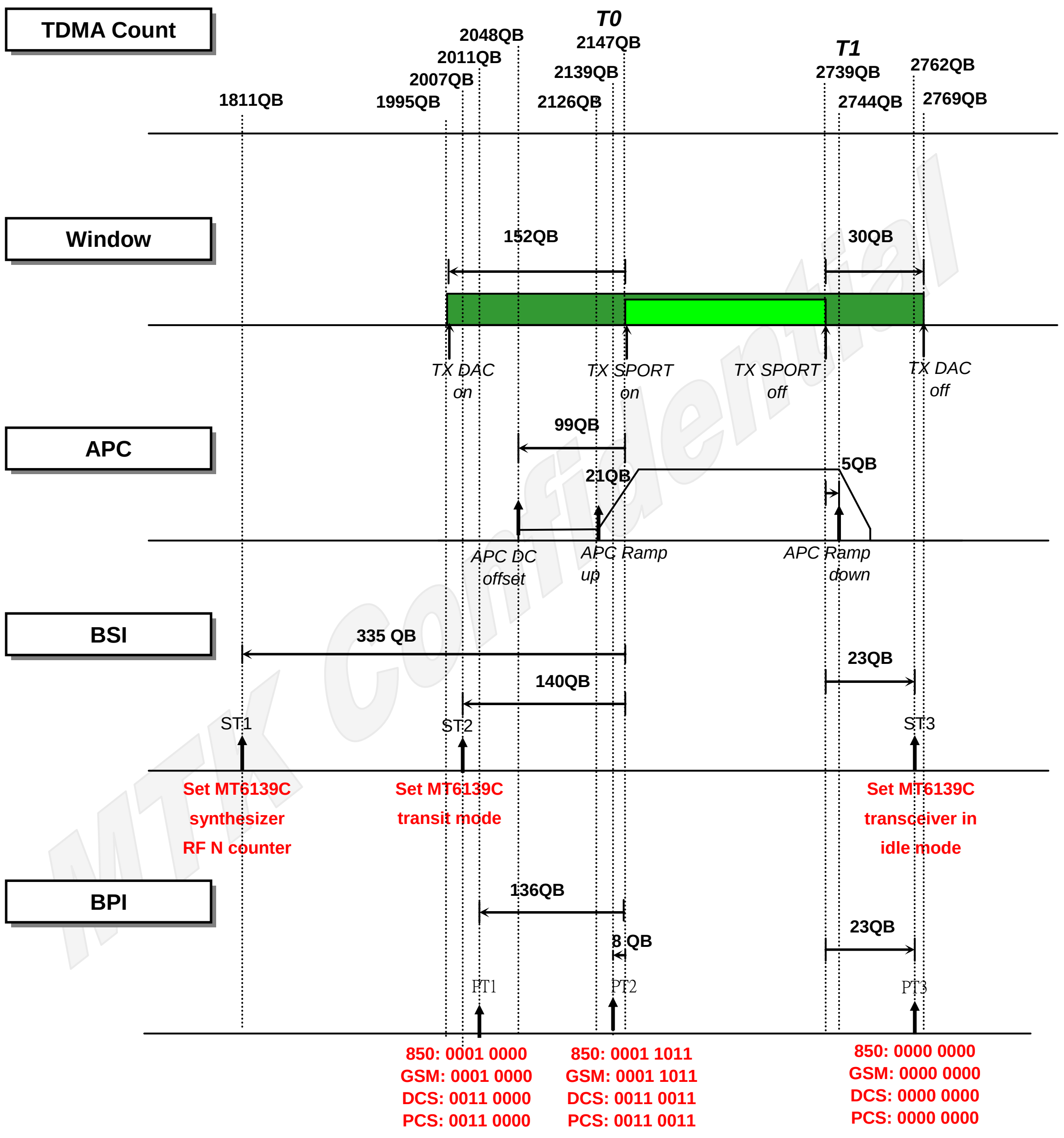


Figure 12 MT6139C Event's timing of TX window

The TDMA timing and data setting are described in 2 parts. One part is that before turning on TX SPORT to receiving I/Q data from MT6139C transceiver. And the other part is after turning off TX SPORT to finish transmission I/Q data. To describe these two parts easily, the timing of turning on TX SPORT is taken as one base named **T0**. And the timing of turning off TX SPORT is taken as one base named **T1**. In this case, T0 is at TDMA counter 2147 QB and T1 is at TDNA counter 2739 QB.

2.9.3.1 Set Synthesizer N-Counter

335 QB ahead of T0, RF module starts to work. MT62xx sends serial command to lock MT6139C synthesizer to lock the operation frequency by BSI bus. To perform this operation, the timing and data of BSI bus can be implemented as follow steps:

(1) Modified the timing definition in **l1d_custom_rf.h**. **QB_ST1** is the timing definition of 1'st BSI command of TX window.

```
#define  QB_ST1           335
```

(2) Specify the command of BSI to be sent. Function **L1D_RF_SetSData_ST1** in **m12195.c** should be implemented to perform this action. **SETUP_ST1()** is the first statement in **L1D_RF_SetSData_ST1** which is a macro to hide complicated code to initialize setting 1'st BSI data. The second statement **HWRITE_2_SDATA** is specified the data to be sent by BSI bus. Two parameters of **HWRITE_3_SDATA** are control word and data word of BSI data. The control words **SCTRL_WARM**, **SCTRL_PLL** and **SCTRL_TXDIV_GC** are composed of device selection (0) and data bit count (26 bits). The 1st data word **SDATA_WARM** is the command to set MT6139C into Warm-up mode. The 2nd data word **l1d_rf.RFN_data**, synthesizer N-counter setting value, is the evaluation result of **L1D_RF_GetTXPLLSetting** called by Layer 1. The 3rd data word **SCTRL_TXDIV_GC** is used to control the output amplitude of divider in the transmeter of MT6139C.

```
/*MT6139C*/ #define  SCTRL_WARM                   SCTRL_WORD(0,26)
/*MT6139C*/ #define  SCTRL_PLL                    SCTRL_WORD(0,26)
/*MT6139C*/ #define  SCTRL_TXDIV_GC               SCTRL_WORD(0,26)
/*MT6139C*/ const unsigned long CW5_RF[5]={ 0x0,           //GSM450
/*MT6139C*/                                 0x00103F5,     //GSM850
/*MT6139C*/                                 0x00103F5,     //GSM
/*MT6139C*/                                 0x00101F5,     //DCS
/*MT6139C*/                                 0x00101F5 };   //PCS
/*MT6139C*/ void  L1D_RF_SetSData_ST1( void )
/*MT6139C*/ {
/*MT6139C*/    SETUP_ST1();
/*MT6139C*/    HWRITE_3_SDATA(SCTRL_WARM,      SDATA_WARM,
/*MT6139C*/                   SCTRL_PLL,       l1d_rf.RFN_data,
/*MT6139C*/                   SCTRL_TXDIV_GC,  CW5_RF[l1d_rf.band]  );
/*MT6139C*/ }
```

To create the transmission window with right operation frequency, evaluating the setting value of operation frequency is needed. Function **L1D_RF_GetTxPLLSetting** in file **m12191.c** needs to be implemented to evaluate the synthesizer setting. And Layer 1 calls this function when program the RF synthesizer setting value and saves the result into variable **l1d_rf.RFN_data**. This content of this variable is the command to be sent to transceiver chip to lock the operation frequency. The function of MT6139C RF module can be implemented as follow:

```
void  L1D_RF_GetTxPLLSetting( int rf_band, int arfcn, long *rfN, long *ifN )
/*MT6139C*/{
/*MT6139C*/   switch(rf_band)
/*MT6139C*/   {
/*MT6139C*/      case  FrequencyBand850 :
/*MT6139C*/      {
/*MT6139C*/         if(arfcn<=231)
/*MT6139C*/         {  if(arfcn<=166)
/*MT6139C*/            {                              /* ARFCN : 128~166     */
/*MT6139C*/                    *rfN = (63L<<12) | ((arfcn-102)<<5) | 0x480001L;
/*MT6139C*/            }
/*MT6139C*/            else
/*MT6139C*/            {                             /* ARFCN : 167~231     */
/*MT6139C*/                    *rfN = (64L<<12) | ((arfcn-167)<<5) | 0x480001L;
/*MT6139C*/            }
/*MT6139C*/          }
/*MT6139C*/          else
/*MT6139C*/          {                                   /* ARFCN : 232~251     */
/*MT6139C*/                      *rfN = (65L<<12) | ((arfcn-232)<<5) | 0x480001L;
/*MT6139C*/          }
/*MT6139C*/
/*MT6139C*/          break;
/*MT6139C*/       }
/*MT6139C*/       case  FrequencyBand900 :
/*MT6139C*/       {      :
/*MT6139C*/              :
/*MT6139C*/       }
/*MT6139C*/       case  FrequencyBand1800 :
/*MT6139C*/       {      :
/*MT6139C*/              :
/*MT6139C*/       }
/*MT6139C*/       case  FrequencyBand1900 :
/*MT6139C*/       {      :
/*MT6139C*/              :
/*MT6139C*/       }
/*MT6139C*/       default :
/*MT6139C*/       {      :
/*MT6139C*/              :
/*MT6139C*/       }
/*MT6139C*/    }
/*MT6139C*/    *ifN = 0;
/*MT6139C*/}
```

2.9.3.2 Set Transceiver in Transmit Mode

140 QB ahead of T0, MT62xx Baseband sends serial command to set transmit mode by BSI bus and after 4QB TX mode setting is performed, MT62xx Baseband activates control pins by changing BPI bus states, To perform these operations, the timing and data of BPI and BSI bus can be implemented as follow steps:

(1) Modified the timing definition in **l1d_custom_rf.h**. QB_ST2 is the timing definition of 2'nd BSI command of TX window. QB_PT1 is the timing definition of 1'st BPI bus status changing of TX window.

```
#define  QB_ST2           140
#define  QB_PT1           136
```

(2) Define the data of BPI bus states of each band. The data is shown in the below table:

BPI_BUS	7	6	5	4	3	2	1	0	Hex
Function	reserved	reserved	DCS BAND_SW	PAEN	Vc2	Vc1	Vc3	Vdd	
850	X	X	0	1	0	0	0	0	10h
GSM	X	X	0	1	0	0	0	0	10h
DCS	X	X	1	1	0	0	0	0	30h
PCS	X	X	1	1	0	0	0	0	30h

So the BPI data can be defined as follow:

```
#define  PDATA_GSM850_PT1     0x10
#define  PDATA_GSM_PT1        0x10
#define  PDATA_DCS_PT1        0x30
#define  PDATA_PCS_PT1        0x30
```

(3) Specify the command of BSI to be sent. Function **L1D_RF_SetSData_ST2** in **m12195.c** should be implemented to perform this action. **SETUP_ST2()** is the first statement in **L1D_RF_SetSData_ST2** which is a macro to hide complicated code to initialize setting 2'nd BSI data. The second statement **HWRITE_1_SDATA** is specified the data to be sent by BSI bus. Two parameters of **HWRITE_1_SDATA** are control word and data word of BSI data. The control word **SCTRL_TXBAND** is composed of device

selection (0) and data bit count (26 bits). The data word **SDATA_TXMODE** is command to set MT6139C into TX mode..

```
/*MT6139C*/ #define  SCTRL_TXBAND                SCTRL_WORD(0,26)
/*MT6139C*/ const unsigned long SDATA_TXMODE = 0x0003002L;
/*MT6139C*/ void  L1D_RF_SetSData_ST2( void )
/*MT6139C*/ {
/*MT6139C*/    SETUP_ST2();
/*MT6139C*/    HWRITE_1_SDATA( SCTRL_TXBAND,   SDATA_TXMODE );
/*MT6139C*/ }
```

2.9.3.3 TX DAC Control

The timing setting for TX frame enable and frame synchronization control of MT62xx are needed. TX frame enable is used to enable or disable TX DAC. TX frame synchronization is used to turn on/off TX SPORT. The timing of enabling TX DAC to turning on TX SPORT named **QB_TX_FENA_2_FSYNC** and the timing of turning off TX SPORT to disabling TX DAC named **QB_TX_FSYNC_2_FENA** are both also needed to be defined in **l1d_custom_rf.h**. In this case, the timing of enabling TX DAC to turning on TX SPORT is 152 QB and the timing of turning off TX SPORT to disabling TX DAC is 30 QB. So **QB_TX_FENA_2_FSYNC** and **QB_TX_FSYNC_2_FENA** can be defined as:

```
#define  QB_TX_FENA_2_FSYNC    152
#define  QB_TX_FSYNC_2_FENA    30
```

2.9.3.4 Set APC

99 QB ahead of T0, APC DAC output performs a DC offset to let ramp up more smoothly. The timing of performing this DC offset is defined as **QB_APCDACON** in **l1d_custom_rf.h**. The value of APC DC offset is maintained in APC profile and it is introduced in the latter section.

```
#define  QB_APCDACON    99
```

8 QB ahead of T0, MT62xx Baseband changes BPI bus states to select operation right band.

(1) Modified the timing definition in **l1d_custom_rf.h**. **QB_PT2** is the timing definition of 2'nd BPI bus status changing of TX window.

```
#define  QB_PT2           8
```

(2) Define the data of BPI bus states of each band. The state of BPI of each band is shown in the table:

BPI_BUS	7	6	5	4	3	2	1	0	Hex
Function	reserved	reserved	DCS BAND_SW	PAEN	Vc2	Vc1	Vc3	Vdd	
850	X	X	0	1	1	0	1	1	1Bh
GSM	X	X	0	1	1	0	1	1	1Bh
DCS	X	X	1	1	0	0	1	1	33h
PCS	X	X	1	1	0	0	1	1	33h

So the BPI data can be defined as follow:

```
#define  PDATA_GSM850_PT2    0x1B
#define  PDATA_GSM_PT2       0x1B
#define  PDATA_DCS_PT2       0x33
#define  PDATA_PCS_PT2       0x33
```

21 QB ahead of T0, APC DAC output performs ramping up. The timing of performing this ramping up is defined as **QB_APCON** in **l1d_custom_rf.h**.

```
#define  QB_APCON    21
```

5 QB behind T1 , APC DAC output performs ramping down. The timing of performing this ramping down is defined as **QB_APCOFF** in **l1d_custom_rf.h**.

```
#define  QB_APCOFF    5
```

2.9.3.5 Set Transceiver in Idle Mode

23 QB behind T1, RF module finishes transmission data and MT62xx set it into idle mode. MT62xx Baseband activates control pins by changing BPI bus states, and also sends serial command to set idle mode command by BSI bus. To perform these operations, the timing and data of BPI and BSI bus can be implemented as follow steps:

(1) Modified the timing definition in **l1d_custom_rf.h**. **QB_ST3** is the timing definition of 3'rd BSI command of TX window. **QB_PT3** is the timing definition of 3'rd BPI bus status changing of TX window.

```
#define  QB_ST3           23
#define  QB_PT3           23
```

(2) Define the data of BPI bus states of each band. The state of BPI of each band is shown in the table:

BPI_BUS	7	6	5	4	3	2	1	0	Hex
Function	reserved	reserved	DCS BAND_SW	PAEN	Vc2	Vc1	Vc3	Vdd	
850	X	X	0	0	0	0	0	0	00h
GSM	X	X	0	0	0	0	0	0	00h
DCS	X	X	0	0	0	0	0	0	00h
PCS	X	X	0	0	0	0	0	0	00h

So the BPI data can be defined as follow:

```
#define  PDATA_GSM850_PT3     0x00
#define  PDATA_GSM_PT3        0x00
#define  PDATA_DCS_PT3        0x00
#define  PDATA_PCS_PT3        0x00
```

(3) Specify the command of BSI to be sent. Function **L1D_RF_SetSData_ST3** in **m12195.c** should be implemented to perform this action. The first statement in **L1D_RF_SetSData_ST3 is SETUP_ST3()** which is a macro to hide complicated code to initialize setting 3'rd BSI data. The second statement **HWRITE_1_SDATA** is specified the data to be sent by BSI bus. The parameters of **HWRITE_1_SDATA** are control word and data word of BSI data. The control word **SCTRL_IDLE** is composed of device selection (0) and data bit count (26 bits). The data word **SDATA_IDLE** is the command to set MT6139C into Idle mode. **SCTRL_IDLE** and **SDATA_IDLE** are defined in **l1d_rf.h.**

```
/*MT6139C*/ #define  SCTRL_IDLE                   SCTRL_WORD(0,26)
/*MT6139C*/ #define  SDATA_IDLE                   0x0002802
/*MT6139C*/ void  L1D_RF_SetSData_ST3( void )
/*MT6139C*/ {
/*MT6139C*/    SETUP_ST3();
/*MT6139C*/    HWRITE_1_SDATA( SCTRL_IDLE, SDATA_IDLE );
```

The propagation delay when transmit data from Baseband uplink DAC to antenna should be claimed in file **l1d_custom_rf.c**. This delay depends on different RF module. For MT6139C RF module, the delay is 37 QB

```
const  short  TxPropagationDelay = 37;
```

2.9.3.6 Events Activation

If Layer 1 plans to transmit 148-bit data from TDMA timer count 2147 QB to 2739 QB, the above setting applies to create the TX window.

(1) At TDMA timer count is 1811 QB (336 QB ahead of TX SPORT turn on), the serial command of setting operation frequency is sent.

(2) At TDMA timer count is 1995 QB (152 QB ahead of TX SPORT turn on), TX DAC is enabled.

(3) At TDMA timer count is 2007 QB (140 QB ahead of TX SPORT turn on), serial command to set transmit mode is sent.

(4) At TDMA timer count is 2011 QB (136 QB ahead of TX SPORT turn on), BPI bus changes its states to TX mode.

(5) At TDMA timer count is 2048 QB (99 QB ahead of TX SPORT turn on), APC DAC output performs a DC offset to let ramp up more smoothly.

(6) At TDMA timer count is 2126 QB (21 QB ahead of TX SPORT turn on), APC DAC output performs ramping up.

(7) At TDMA timer count is 2139 QB (8 QB ahead of TX SPORT turn on), BPI bus changes its states to select operation right band.

(8) At TDMA timer count is 2147 QB, TX SPORT in MT62xx is turned on.

(9) At TDMA timer count is 2739 QB, TX SPORT in MT62xx is turned off.

(10) At TDMA timer count is 2744 QB (5 QB behind TX SPORT turning off), APC DAC output performs ramping down.

(11) At TDMA timer count is 2762 QB (23 QB behind TX SPORT turning off), BPI bus changes its states and send 3-wire command to set MT6139C into idle mode is sent.

(12) At TDMA timer count is 2769 QB (30 QB behind TX SPORT turn off), TX DAC is disabled at this time.

2.10 RF Control Configuration of Helios2(SKY74137) RF Module

For example, the schematic circuit interface between Helios2 RF module and MT62xx chip is shown in below diagram. Please refer to Helios2 RF Module schematic circuit to get more detail information.

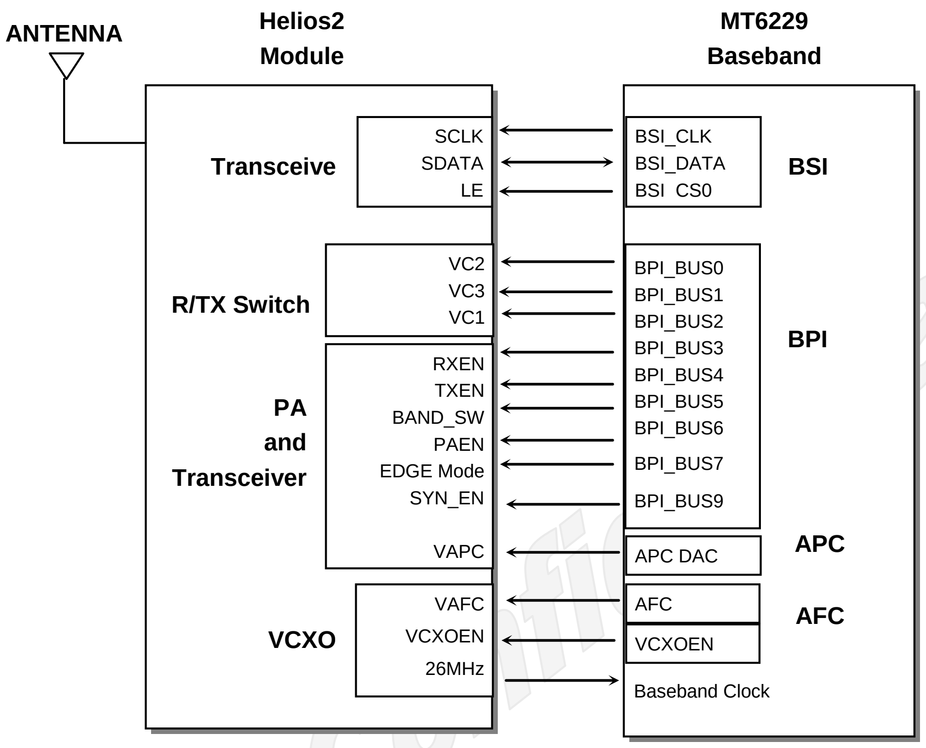


Figure 13 The connection of MT6229 and Helios2 RF Module

BSI_CLK, BSI_DATA, BSI_CS0 pins of MT62xx are directly connected to the serial interface of Helios2 transceiver. APC_DAC pin of MT62xx is connected to the VAPC pin of PA of RF module board. VCXOEN of MT62xx controls the enable state of VCXO. AFC DAC output is connected to the voltage control pin VAFC of VCXO. The generated clock of VCXO provides MT62xx system clock. Band-related pins Vdd, Vc3, Vc1, Vc2, PAEN, DCS_BAND_SW are controlled by the bit 0,1,2,3,4,5 of BPI bus of MT62xx. The high/low state of each pin of BPI bus and their timing of changing state can be programmed by MT62xx. The RF radio control method is introduced in the latter description.

2.10.1 Initialization

RF module can be controlled by the BSI, BPI, APC, AFC units of MT62xx chip. BSI, BPI, APC, AFC units can work in 2 modes. One mode is immediate control mode, and the other is TDMA event driven control mode.

Immediate control mode is that the output signal of BSI, BPI, APC, or AFC unit is generated as soon as setting the corresponding registers in MT62xx. When turning on the RF module, the immediate mode can be used to initialize the RF module. For example, immediate control mode is used when initializing Helios2 RF module, and the initial states are set as follow:

(1) Vc2 (BPI_BUS0) is low state
(2) Vc3 (BPI_BUS1) is low state
(3) Vc1 (BPI_BUS2) is low state

(4) RXEN (BPI_BUS3) is low state
(5) TXEN (BPI_BUS4) is low state
(6) BAND_SW (BPI_BUS5) is low state
(7) PAEN (BPI_BUS6) is low state
(8) EDGE_MODE (BPI_BUS7) is low state
(9) SYN_EN (BPI_BUS9) is low state
(10) Serial commands (0x44DC02, 0x03CA28, 0xFD7C38, 0x02AE46, 0x03860A, 0x007452, 0x2A5DD6, 0x00691E) are sending to Helios2 transceiver.

The initialization of BPI, BSI data can be implemented in **L1D_RF_PowerOn** in **m12196.c** by using immediate mode:

```
/*SKY74137*/ void  L1D_RF_PowerOn( void )
/*SKY74137*/ {
/*SKY74137*/    IMM_MODE_BEGIN( IMMMASK_ALL );
/*SKY74137*/    IMM_SEND_BSI( SCTRL_IMOD(0,24), 0x44DC02 );
/*SKY74137*/    IMM_SEND_BSI( SCTRL_IMOD(0,24), 0x03CA28 );
/*SKY74137*/    IMM_SEND_BSI( SCTRL_IMOD(0,24), 0xFD7C38 );
/*SKY74137*/    IMM_SEND_BSI( SCTRL_IMOD(0,24), 0x02AE46 );
/*SKY74137*/    IMM_SEND_BSI( SCTRL_IMOD(0,24), 0x03860A );
/*SKY74137*/    IMM_SEND_BSI( SCTRL_IMOD(0,24), 0x007452 );
/*SKY74137*/    IMM_SEND_BSI( SCTRL_IMOD(0,24), 0x2A5DD6 );
/*SKY74137*/    IMM_SEND_BSI( SCTRL_IMOD(0,24), 0x00691E );
/*SKY74137*/    IMM_MODE_END(  );
/*SKY74137*/ }
```

2.10.2 Create RX window

TDMA timer is a counter unit that counts 1 per quarter-bit (0.923u). It always repeats counting from 0 to 4999. So the period of TDMA counter is 5000 QB (1 TDMA frame). If BSI, BPI, AFC, APC are programmed in TDMA event driven mode, the active time can be programmed. When the TDMA counter counts to the programmed time (count), the corresponding unit is triggered to activate.

If it is needed to receive data from air or transmit data to air, Layer1 needs to plan all RF setting and timing to create RX/TX window, control receiving gain, control VCXO, control PA... etc. For this purpose, TDMA event driven mode of BSI, BPI, APC, or AFC units should be used.

For example of creating RX window to receive data, the RF setting and timing of Helios2 RF module can be depicted in the below diagram:

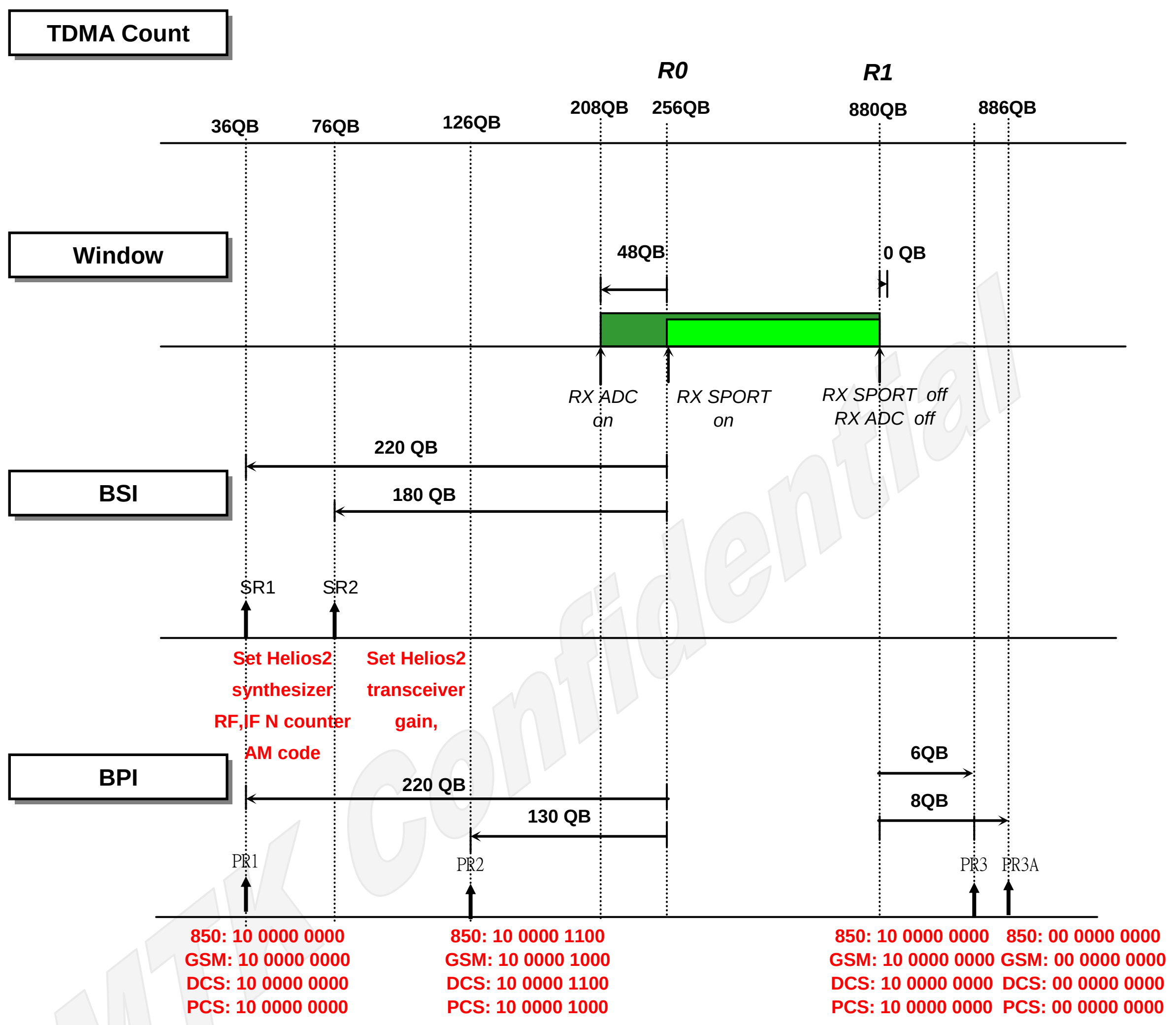


Figure 14 Helios2 Event's timing of RX window

The TDMA timing and data setting are described in 2 parts. One part is that before turning on RX SPORT to receiving I/Q data from Helios2 transceiver. And the other part is after turning off RX SPORT to finish receiving I/Q data. To describe these two parts easily, the timing of turning on RX SPORT is taken as one base named **R0**. And the timing of turning off RX SPORT is taken as one base named **R1**. In this case, R0 is at TDMA counter 256 QB and R1 is at TDMA counter 880 QB.

2.10.2.1 Set Synthesizer N-Counter

220 QB ahead of R0, RF module should start to operate. MT62xx chip controls the RF module by activating control pins and sending serial command. MT62xx activates the control pins by changing BPI bus states. The high/low state of BPI bus at this time depends on operation band. At the same time MT62xx also sends serial command to lock Helios2 synthesizer to lock the operation frequency by BSI bus. To perform these operations, the timing and data of BPI and BSI bus can be implemented as follow steps:

(1) Modified the timing definition in **l1d_custom_rf.h**. **QB_SR1** is the timing definition of 1'st BSI command of RX window. **QB_PR1** is the timing definition of 1'st BPI bus status changing of RX window.

```
#define  QB_SR1        220
#define  QB_PR1        220
```

(2) Define the states of BPI bus for each band. The data is shown in the below table:

BPI_BUS	9	8	7	6	5	4	3	2	1	0	Hex
Function	SYN_EN	Reserved	PAEN	PAEN	BANDSW	TXENA	RXENA	Vc1	Vc3	Vc2	
850	1	X	0	0	0	0	0	0	0	0	200h
GSM	1	X	0	0	0	0	0	0	0	0	200h
DCS	1	X	0	0	0	0	0	0	0	0	200h
PCS	1	X	0	0	0	0	0	0	0	0	200h

So the BPI data can be defined as follow:

```
#define  PDATA_GSM850_PR1     0x200
#define  PDATA_GSM_PR1        0x200
#define  PDATA_DCS_PR1        0x200
#define  PDATA_PCS_PR1        0x200
```

(3) Specify the command of BSI to be sent. Function **L1D_RF_SetSData_SR1** in **m12195.c** be implemented to perform this action. **SETUP_SR1()** is the first statement in **L1D_RF_SetSData_SR1** which is a macro to hide complicated code to initialize setting 1'st BSI data. The second statement **HWRITE_2_SDATA** is specified the data to be sent by BSI bus. Two parameters of HWRITE_1_SDATA are control word and data word of BSI data. The control words **SCTRL_SXR2**, **SCTRL_SXR1** are composed of device selection (0) and data bit count (24 bits). The data word **l1d_rf.IFN_data** and **l1d_rf.RFN_data**, RF N-Counter setting value, is the evaluation result of **L1D_RF_GetRxPLLSetting** called by Layer 1. The 3rd data word **SKY_AM_LOOP_GAIN** is Helios2 AM Loop coefficient which is provided by the Skyworks. The data word is corresponding with the band. **l1d_rf.band** is the global variable which indicates the band of operation frequency.

```
/*SKY74137*/ #define  SCTRL_SXR1                     SCTRL_WORD(0,24)
/*SKY74137*/ #define  SCTRL_SXR2                     SCTRL_WORD(0,24)
/*SKY74137*/ unsigned long SKY_AM_LOOP_GAIN[5] =
/*SKY74137*/ {
/*SKY74137*/           0L, /* FrequencyBand400  (not support)  */
/*SKY74137*/    0x180000L, /* FrequencyBand850                 */
/*SKY74137*/    0x180000L, /* FrequencyBand900                 */
/*SKY74137*/    0x180000L, /* FrequencyBand1800                */
/*SKY74137*/    0x180000L, /* FrequencyBand1900                */
/*SKY74137*/ };
void  L1D_RF_SetSData_SR1( void )
{
   SETUP_SR1();
   HWRITE_2_SDATA( SCTRL_SXR2, l1d_rf.IFN_data,
                   SCTRL_SXR1, (l1d_rf.RFN_data | SKY_AM_LOOP_GAIN[l1d_rf.band]) );
}
```

To create the receiving window with right operation frequency, evaluating the setting value of operation frequency is needed. Function **L1D_RF_GetRxPLLSetting** in file **m12191.c** needs to be implemented to evaluate the synthesizer setting. And Layer 1 calls this function when programs the RF synthesizer setting value and saves the result into variable **l1d_rf.RFN_data and l1d_rf.IFN_data**. This content of this variable is the command to be sent to transceiver chip to lock the operation frequency. The evaluation of Helios2 synthesizer N counter setting is as the following code:

```
void  L1D_RF_GetRxPLLSetting( int rf_band, int arfcn, long *rfN, long *ifN )
/*SKY74137*/ {  int Nfrac;
/*SKY74137*/    int f_vco;
/*SKY74137*/    switch(rf_band)
/*SKY74137*/    {
/*SKY74137*/       case  FrequencyBand850 :
```

```
/*SKY74137*/        {
/*SKY74137*/           if (arfcn <= 201)
/*SKY74137*/           {  if (arfcn <= 158)            /*  ARFCN :   128~158    */
/*SKY74137*/              {
/*SKY74137*/                Nfrac = 1226027 + 96792*(arfcn-128) - ((arfcn-128)>>2) -2 ;
/*SKY74137*/                  *rfN = 0xE20000 | (97L<<6);
/*SKY74137*/              }
/*SKY74137*/              else                               /*  ARFCN :   159~201    */
/*SKY74137*/              {   Nfrac = 32264 + 96791*(arfcn-159) + ((arfcn-159)>>1) +3 ;
/*SKY74137*/                  *rfN = 0xE20000 | (98L<<6);
/*SKY74137*/              }
/*SKY74137*/           }
/*SKY74137*/           else
/*SKY74137*/           {  if (arfcn <= 245)                /*  ARFCN :   202~245    */
/*SKY74137*/              {  Nfrac = 96792*(arfcn-202) - ((arfcn-202)>>1) +2 ;
/*SKY74137*/                 *rfN = 0xE20000 | (99L<<6);
/*SKY74137*/              }
/*SKY74137*/              else                               /*  ARFCN :   246~251    */
/*SKY74137*/              {  Nfrac = 64528 + 96792*(arfcn-246) - ((arfcn-246)>>2) ;
/*SKY74137*/                 *rfN = 0xE20000 | (100L<<6);
/*SKY74137*/              }
/*SKY74137*/           }
/*SKY74137*/           break;
/*SKY74137*/        }
/*SKY74137*/       case  FrequencyBand900 :
/*SKY74137*/       {     :
/*SKY74137*/             :
/*SKY74137*/       }
/*SKY74137*/       case  FrequencyBand1800 :
/*SKY74137*/       {     :
/*SKY74137*/             :
/*SKY74137*/       }
/*SKY74137*/       case  FrequencyBand1900 :
/*SKY74137*/       {     :
/*SKY74137*/             :
/*SKY74137*/       }
/*SKY74137*/       default :
/*SKY74137*/       {
/*SKY74137*/          break;
/*SKY74137*/       }
/*SKY74137*/    }
/*SKY74137*/    *ifN = 0x3 | (Nfrac<<2);
/*SKY74137*/}
```

2.10.2.2 Set Transceiver Gain

130 QB ahead of R0, Band selection and Rx gain setting should be performed. MT62xx activates control pins by changing BPI bus states, and also 180 QB ahead of R0, we send serial command to set transceiver gain by BSI bus. To perform these operations, the timing and data of BPI and BSI bus can be implemented as follow steps:

(1) Modified the timing definition in **l1d_custom_rf.h**. **QB_SR2** is the timing definition of 2'nd BSI command of RX window. **QB_PR2** is the timing definition of 2'nd BPI bus status changing of RX window.

```
#define  QB_SR2          180
#define  QB_PR2          130
```

(2) Define the data of BPI bus states for each band. The state of BPI of each band is shown in the table:

BPI_BUS	9	8	7	6	5	4	3	2	1	0	Hex
Function	SYN_EN	Reserved	PAEN	PAEN	BANDSW	TXENA	RXENA	Vc1	Vc3	Vc2	
850	1	X	0	0	0	0	1	1	0	0	20Ch
GSM	1	X	0	0	0	0	1	0	0	0	208h
DCS	1	X	0	0	0	0	1	1	0	0	20Ch
PCS	1	X	0	0	0	0	1	0	0	0	208h

So the BPI data can be defined as follow:

```
#define  PDATA_GSM850_PR2       0x20C
#define  PDATA_GSM_PR2          0x208
#define  PDATA_DCS_PR2          0x20C
#define  PDATA_PCS_PR2          0x208
```

(3) Specify the command of BSI to be sent. Function **L1D_RF_SetSData_SR2** in **m12195.c** should be implemented to perform this action. The first statement in **L1D_RF_SetSData_SR2** is **SETUP_SR2()** which is a macro to hide complicated code to initialize setting 2'nd BSI data. The second statement **HWRITE_1_SDATA** is specified the data to be sent by BSI bus. The parameters of **HWRITE_1_SDATA** are control word and data word of BSI data. The control word **SCTRL_RTXCTRL** is composed of device selection (0) and data bit count (24 bits). The data word is composed of two commands. The 1st command **l1d_rf.AGC_data**, transceiver gain setting value, is the evaluation result of **L1D_RF_GetGainSetting** called by Layer 1. The 2nd command **SKY_BAND_CP_CURRENT** is Helios2 charge pump current coefficient which is provided by the Skyworks. The 2nd command is corresponding with the band. **l1d_rf.band** is the global variable which indicates the band of operation frequency. The **RFSpecialCoef.rx.sky74137.fixgain_enable** is used for switch the interslot gain settings in order to avoid the out band blocking fail due to the interslot gain settings, which is a bug of the SKY74137. This variable can be switched at the engineering mode of the MS, and for the normal operations, the interslot gain settings is suggested. The control word **SCTRL_RESERVED** is used to turn off the event.

```
/*SKY74137*/ #define  SCTRL_RTXCTRL                SCTRL_WORD(0,24)
void  L1D_RF_SetSData_SR2( void )
{
   SETUP_SR2();
   if(IS_CONTINUOUS_RX_SLOT() && RFSpecialCoef.rx.sky74137.fixgain_enable == 0x1)
   {
    HWRITE_1_SDATA(SCTRL_RESERVED,(l1d_rf.AGC_data|SKY_BAND_CP_CURRENT[l1d_rf.band]) );
   }
   else
   {
    HWRITE_1_SDATA(SCTRL_RTXCTRL,(l1d_rf.AGC_data| SKY_BAND_CP_CURRENT[l1d_rf.band]) );
   }
 }
```

To create the receiving window with suitable gain of receiving amplifier, evaluating the setting value of request gain is needed. Function **L1D_RF_GetGainSetting** in file **m12192.c** needs to be implemented to evaluate the transceiver gain setting. And Layer 1 calls this function when programs the gain setting value and saves the result into variable **l1d_rf.AGC_data**. This content of this variable is the command to be sent to transceiver chip to set the requested receive amplifier gain. The evaluation of Helios2 receiving amplifier gain setting is as the following code:

```
    /*SKY74137*/ int  L1D_RF_GetGainSetting( int rf_band, int arfcn, int request_gain,
long *gain_setting )
    /*SKY74137*/ {  int   max_gain, min_gain, real_gain;
    /*SKY74137*/    const int*    d32ptr;
    /*SKY74137*/    const sL1DAGCDATA*  agcptr;
    /*SKY74137*/    long  setting;
    /*SKY74137*/    int   bit_no;
    /*SKY74137*/    int   left, right, middle;
    /*SKY74137*/
    /*SKY74137*/    /* transfer power gain to voltage gain */
    /*SKY74137*/    request_gain = request_gain + 34*8;
    /*SKY74137*/
    /*SKY74137*/    /* evaluate the range of available gain */
    /*SKY74137*/    d32ptr  = (int*)&(GAIN_RANGE_TABLE[rf_band]);
    /*SKY74137*/    max_gain = *d32ptr++;
    /*SKY74137*/    min_gain = *d32ptr;
    /*SKY74137*/
    /*SKY74137*/    /* clipping the request gain to the avialable gain */
    /*SKY74137*/    if( request_gain>=max_gain )
    /*SKY74137*/    {  request_gain = max_gain;  }
    /*SKY74137*/    else  if( request_gain<=min_gain )
    /*SKY74137*/    {  request_gain = min_gain;  }
    /*SKY74137*/
    /*SKY74137*/    /* evaluate the real gain setting */
    /*SKY74137*/    agcptr = AGC_TABLE[rf_band];
    /*SKY74137*/
    /*SKY74137*/    /* binary search */
    /*SKY74137*/    left = 0; right = AGC_TABLE_SIZE[rf_band]-1;
    /*SKY74137*/    while (left <= right)
    /*SKY74137*/    {
    /*SKY74137*/       middle = (left + right)/2;
    /*SKY74137*/       if ( request_gain > (agcptr+middle)->pos_gain )
    /*SKY74137*/          right = middle - 1;
    /*SKY74137*/       else if ( request_gain < (agcptr+middle)->pos_gain )
    /*SKY74137*/          left = middle + 1;
    /*SKY74137*/       else
    /*SKY74137*/       {
    /*SKY74137*/          left = middle;
    /*SKY74137*/          break;
    /*SKY74137*/       }
    /*SKY74137*/    }
    /*SKY74137*/
    /*SKY74137*/    agcptr = (agcptr+left);
    /*SKY74137*/
    /*SKY74137*/    {  bit_no    = BIT_NO( request_gain, agcptr->A, GC_B );
    /*SKY74137*/       real_gain = REAL_GAIN( bit_no, agcptr->A, GC_B );
    /*SKY74137*/       /* transfer voltage gain to power gain */
    /*SKY74137*/       real_gain = real_gain - 34*8;
    /*SKY74137*/       setting   = agcptr->setting + (bit_no<<6);
    /*SKY74137*/    }
    /*SKY74137*/
    /*SKY74137*/    *gain_setting = setting;
    /*SKY74137*/    return( real_gain );
    /*SKY74137*/ }
```

2.10.2.3 Set Transceiver in Idle Mode

6 QB behind R1, RF module finishes receiving data and MT62xx set it into idle mode. MT62xx Baseband activates control pins by changing BPI bus states. Note that there is a constrain for the Helios2 that the SXENA and the RX_ENA cannot be turned off simultaneously or the large current would be introduced in the idle mode of the Helios2. As the result, we have designed the new event named PR3A which is used to turn off the SXENA and the RX_ENA separately. For Helios2, BSI command is not needed to let Helios enter idle mode, but we have

to set the timing for the BSI command even the event would not happen. To perform these operations, the timing and data of BPI and BSI bus can be implemented as follow steps:

(1) Modified the timing definition in **l1d_rf.h**. **QB_SR3** is the timing definition of 3'rd BSI command of RX window. **QB_PR3** and **QB_PR3A** are the timing definition of 3'rd BPI bus status changing of RX window. Note that the PR3A is the last event of the BPI bus rather than the PR3, as the result, the timing limitations for the PR3 on the next chapter should be applied on the PR3A rather than the PR3.

```
#define  QB_SR3           0
#define  QB_PR3           6
#define  QB_PR3A          8
```

(2) Define the data of BPI bus states for each band. The state of PR3 of each band is shown in the table:

BPI_BUS	9	8	7	6	5	4	3	2	1	0	Hex
Function	SYN_EN	Reserved	PAEN	PAEN	BANDSW	TXENA	RXENA	Vc1	Vc3	Vc2	
850	1	X	0	0	0	0	0	0	0	0	20Ch
GSM	1	X	0	0	0	0	0	0	0	0	200h
DCS	1	X	0	0	0	0	0	0	0	0	200h
PCS	1	X	0	0	0	0	0	0	0	0	200h

So the PR3 data can be defined as follow:

```
#define  PDATA_GSM850_PR3      0x200
#define  PDATA_GSM_PR3         0x200
#define  PDATA_DCS_PR3         0x200
#define  PDATA_PCS_PR3         0x200
```

(2) Define the data of BPI bus states for each band. The state of PR3A of each band is shown in the table:

BPI_BUS	9	8	7	6	5	4	3	2	1	0	Hex
Function	SYN_EN	Reserved	PAEN	PAEN	BANDSW	TXENA	RXENA	Vc1	Vc3	Vc2	
850	0	X	0	0	0	0	0	0	0	0	000h
GSM	0	X	0	0	0	0	0	0	0	0	000h
DCS	0	X	0	0	0	0	0	0	0	0	000h
PCS	0	X	0	0	0	0	0	0	0	0	000h

So the PR3 data can be defined as follow:

```
#define  PDATA_GSM850_PR3A     0x000
#define  PDATA_GSM_PR3A        0x000
#define  PDATA_DCS_PR3A        0x000
#define  PDATA_PCS_PR3A        0x000
```

(3) Specify the command of BSI to be sent. Function **L1D_RF_SetSData_SR3** in **m12195.c** should be implemented to perform this action. However, the Helios2 does not need SR3 for entering idle mode, and so the L1D_RF_SetSData_SR3 is an empty function.

```
/*SKY74137*/ void  L1D_RF_SetSData_SR3( void )
/*SKY74137*/ {
/*SKY74137*/ }
```

2.10.2.4 RX ADC Control

Besides setting the timing and data of BPI and BSI bus of MT62xx Baseband to create a RX window, the timing setting for RX frame enable and frame synchronization control of MT62xx are needed. RX frame enable is used to enable or disable RX ADC. RX frame synchronization is used to turn on/off RX SPORT. The timing of enabling RX ADC to turning on RX SPORT named **QB_RX_FENA_2_FSYNC** and the timing of turning off RX SPORT to disabling RX ADC named **QB_RX_FSYNC_2_FENA** are both also needed to be defined in **l1d_custom_rf.h**. In this case, the timing of enabling RX ADC to turning on RX SPORT is recommend to be 40

QB and the timing of turning off RX SPORT to disabling RX ADC is 0 QB. So **QB_RX_FENA_2_FSYNC** and **QB_RX_FSYNC_2_FENA** can be defined as:

```
#define  QB_RX_FENA_2_FSYNC    48
#define  QB_RX_FSYNC_2_FENA    0
```

2.10.2.5 Events Activation

If Layer1 plans to receive 156-bit data from TDMA timer count 256 QB to 880 QB, the above setting applies to create the RX window.

(1) At TDMA timer count is 36 QB (220 QB ahead of RX SPORT turn on), BPI bus changes its states to 0x200h. At the same time, the serial command of setting operation frequency is sent to Helios2 transceiver.

(4) At TDMA timer count is 76 QB (180 QB ahead of RX SPORT turn on), the serial command of setting RX gain is sent to Helios2 transceiver

(5) At TDMA timer count is 126 QB (130 QB ahead of RX SPORT turn on), BPI bus changes its states to 208h(GSM850/PCS) or 20Ch(GSM/DCS).

(3) At TDMA timer count is 208 QB (48 QB ahead of RX SPORT turn on), RX ADC in MT62xx is enabled.

(4) At TDMA timer count is 256 QB, RX SPORT in MT62xx is turned on. The I/Q data begin to be sampled.

(5) At TDMA timer count is 880 QB, RX SPORT in MT62xx is turned off and RX ADC is also disabled. The I/Q data stop sampling.

(7) At TDMA timer count is 886 QB (6 QB behind RX SPORT turning off), BPI bus changes its states to 200h .

(8) At TDMA timer count is 888 QB (8 QB behind RX SPORT turning off), BPI bus changes its states to 000h .

2.10.3 Create TX window

Another example is creating TX window to transmit data. The RF setting and timing of Helios2 RF module can be depicted in the below diagram:

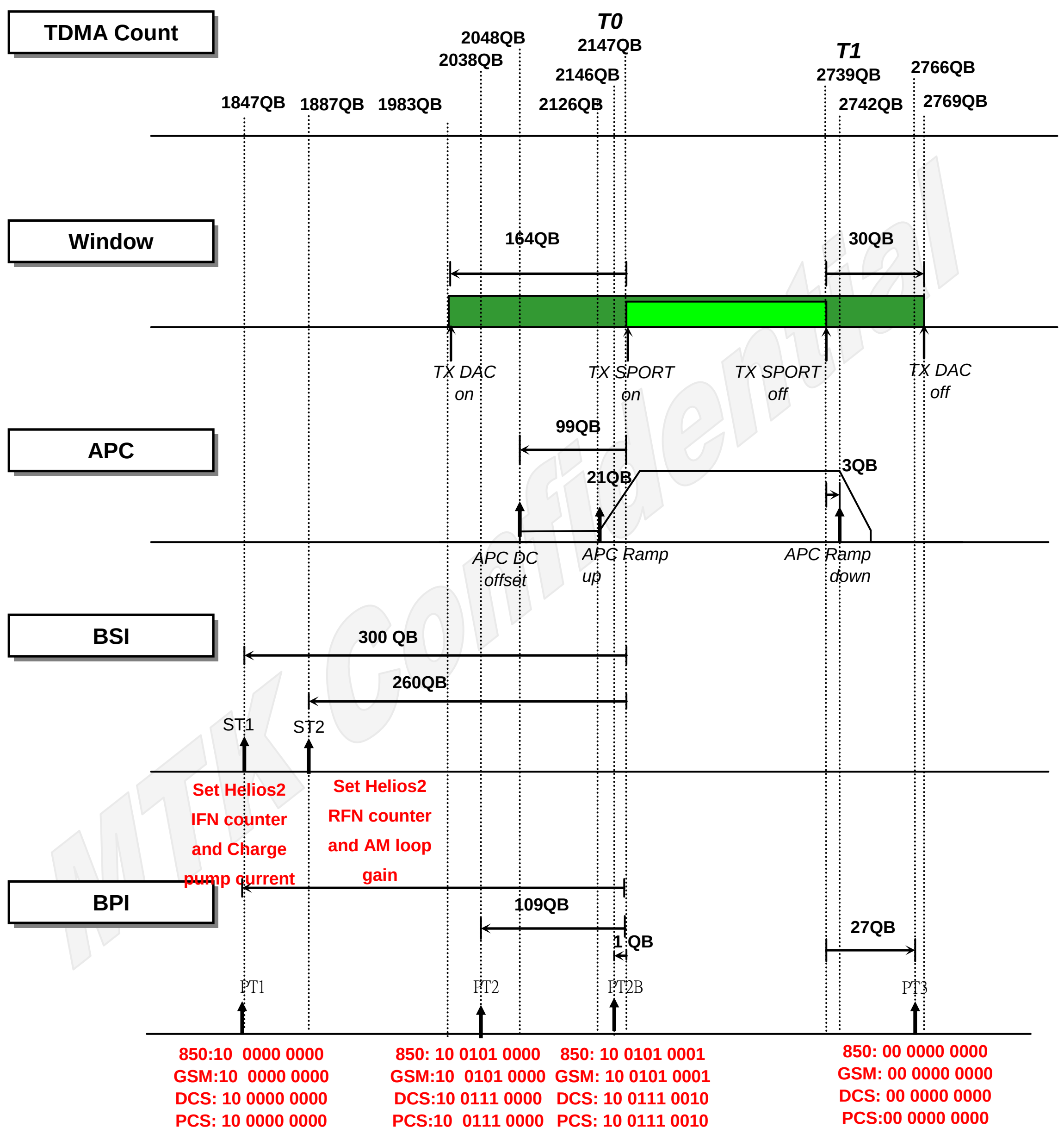


Figure 15 Helios2 Event's timing of TX window

The TDMA timing and data setting are described in 2 parts. One part is that before turning on TX SPORT to receiving I/Q data from Helios2 transceiver. And the other part is after turning off TX SPORT to finish transmission I/Q data. To describe these two parts easily, the timing of turning on TX SPORT is taken as one base named **T0**. And the timing of turning off TX SPORT is taken as one base named **T1**. In this case, T0 is at TDMA counter 2147 QB and T1 is at TDNA counter 2739 QB.

2.10.3.1 Set Synthesizer N-Counter

300 QB ahead of T0, RF module starts to work. MT62xx sends serial command to lock Helios2 synthesizer to lock the operation frequency by BSI bus and BPI bus. To perform this operation, the timing and data of BSI and BPI bus can be implemented as follow steps:

(1) Modified the timing definition in **l1d_custom_rf.h**. **QB_ST1** is the timing definition of 1'st BSI command of TX window, and **QB_ST2** is the timing definition of 2'nd BSI command of TX window. **QB_PT1** is the timing definition of 1'st BPI bus status changing of TX window.

```
#define  QB_ST1           300
#define  QB_ST2           260
#define  QB_PT1           300
```

(2) Specify the command of BSI to be sent. Function **L1D_RF_SetSData_ST1** in **m12195.c** should be implemented to perform this action. **SETUP_ST1()** is the first statement in **L1D_RF_SetSData_ST1** which is a macro to hide complicated code to initialize setting 1'st BSI data. The second statement **HWRITE_2_SDATA** is specified the data to be sent by BSI bus. Two parameters of **HWRITE_2_SDATA** are control word and data word of BSI data. The control words **SCTRL_SXR2** and **SCTRL_RTXCTRL** are composed of device selection (0) and data bit count (24 bits). The 1st data word **SKY_BAND_CP_CURRENT** is the command to set the charge pump current of Helios2. The 2nd data word **l1d_rf.IFN_data**, synthesizer N-counter setting value, is the evaluation result of **L1D_RF_GetTXPLLSetting** called by Layer 1. The data word is corresponding with the band. **l1d_rf.band** is the global variable which indicates the band of operation frequency.

(3) Another command of BSI is needed to be sent. Function **L1D_RF_SetSData_ST2** in **m12195.c** should be implemented to perform this action. **SETUP_ST2()** is the first statement in **L1D_RF_SetSData_ST2** which is a macro to hide complicated code to initialize setting 1'st BSI data. The second statement **HWRITE_1_SDATA** is specified the data to be sent by BSI bus. The parameters of **HWRITE_1_SDATA** are control word and data word of BSI data. The control words **SCTRL_SXR1** is composed of device selection (0) and data bit count (24 bits). The 1st data word **SKY_AM_LOOP_GAIN** is the command to set the AM loop gain of Helios2. The 2nd data word **l1d_rf.RFN_data**, synthesizer N-counter setting value, is the evaluation result of **L1D_RF_GetTXPLLSetting** called by Layer 1. The data word is corresponding with the band. **l1d_rf.band** is the global variable which indicates the band of operation frequency.

```
/*SKY74137*/ #define  SCTRL_SXR2                   SCTRL_WORD(0,24)
/*SKY74137*/ #define  SCTRL_RTXCTRL                SCTRL_WORD(0,24)
/*SKY74137*/ unsigned long SKY_BAND_CP_CURRENT[5] =
/*SKY74137*/ {
/*SKY74137*/           0L, /* FrequencyBand400 (not support)  */
/*SKY74137*/    0xAC0004L, /* FrequencyBand850                */
/*SKY74137*/    0xAC4004L, /* FrequencyBand900                */
/*SKY74137*/    0xB08004L, /* FrequencyBand1800               */
/*SKY74137*/    0xB0C004L, /* FrequencyBand1900               */
/*SKY74137*/ };
/*SKY74137*/ void  L1D_RF_SetSData_ST1( void )
/*SKY74137*/ {
/*SKY74137*/    extern char SKY_d_flag;
/*SKY74137*/    extern char SKY_TXCP;
/*SKY74137*/    SETUP_ST1();
/*SKY74137*/    HWRITE_2_SDATA( SCTRL_RTXCTRL,  SKY_BAND_CP_CURRENT[l1d_rf.band],
/*SKY74137*/                    SCTRL_SXR2,     l1d_rf.IFN_data  );
/*SKY74137*/ }
/*SKY74137*/ void  L1D_RF_SetSData_ST2( void )
/*SKY74137*/ {
```

```
/*SKY74137*/    SETUP_ST2();
     HWRITE_1_SDATA(SCTRL_SXR1,(l1d_rf.RFN_data|SKY_AM_LOOP_GAIN[l1d_rf.band]) );
/*SKY74137*/ }
```

(4) Define the states of BPI bus for each band. The data is shown in the below table:

BPI_BUS	9	8	7	6	5	4	3	2	1	0	Hex
Function	SYN_EN	Reserved	PAEN	PAEN	BANDSW	TXENA	RXENA	Vc1	Vc3	Vc2	
850	1	X	0	0	0	0	0	0	0	0	200h
GSM	1	X	0	0	0	0	0	0	0	0	200h
DCS	1	X	0	0	0	0	0	0	0	0	200h
PCS	1	X	0	0	0	0	0	0	0	0	200h

So the BPI data can be defined as follow:

```
#define  PDATA_GSM850_PR1     0x200
#define  PDATA_GSM_PR1        0x200
#define  PDATA_DCS_PR1        0x200
#define  PDATA_PCS_PR1        0x200
```

To create the transmission window with right operation frequency, evaluating the setting value of operation frequency is needed. Function **L1D_RF_GetTxPLLSetting** in file **m12191.c** needs to be implemented to evaluate the synthesizer setting. And Layer 1 calls this function when program the RF synthesizer setting value and saves the result into variable **l1d_rf.RFN_data** and **l1d_rf.IFN_data**. This content of this variable is the command to be sent to transceiver chip to lock the operation frequency. The function of Helios2 RF module can be implemented as follow:

```
 void  L1D_RF_GetTxPLLSetting( int rf_band, int arfcn, long *rfN, long *ifN )
/*SKY74137*/ {  int   Nfrac;
/*SKY74137*/    float f_vco;
/*SKY74137*/    switch(rf_band)
/*SKY74137*/    {
/*SKY74137*/       case  FrequencyBand850 :
/*SKY74137*/       {
/*SKY74137*/          if (arfcn <= 185)
/*SKY74137*/          {  if (arfcn <= 146)              /*  ARFCN :   128~146    */
/*SKY74137*/             {
/*SKY74137*/                Nfrac = 2147484 + 108407*(arfcn-128) - ((arfcn-128)>>2) -2 ;
/*SKY74137*/                *rfN = 0xE20000 | (103L<<6);
/*SKY74137*/             }
/*SKY74137*/             else                       /*  ARFCN :   147~185    */
/*SKY74137*/             {
/*SKY74137*/                Nfrac = 12906 + 108407*(arfcn-147) - ((arfcn-147)>>2) -3 ;
/*SKY74137*/                *rfN = 0xE20000 | (104L<<6);
/*SKY74137*/             }
/*SKY74137*/          }
/*SKY74137*/          else
/*SKY74137*/          {
/*SKY74137*/             if (arfcn <= 224)                /*  ARFCN :   186~224    */
/*SKY74137*/             {
/*SKY74137*/                Nfrac = 46460 + 108407*(arfcn-186) - ((arfcn-186)>>2) -3 ;
/*SKY74137*/                *rfN = 0xE20000 | (105L<<6);
/*SKY74137*/             }
/*SKY74137*/             else                             /*  ARFCN :   225~251    */
/*SKY74137*/             {
/*SKY74137*/                Nfrac = 80014 + 108407*(arfcn-225) - ((arfcn-225)>>2) -2 ;
/*SKY74137*/                *rfN = 0xE20000 | (106L<<6);
/*SKY74137*/             }
/*SKY74137*/          }
/*SKY74137*/
/*SKY74137*/            SKY_TXCP = 1;
/*SKY74137*/
/*SKY74137*/            break;
/*SKY74137*/         }
/*SKY74137*/        case  FrequencyBand900 :
/*SKY74137*/        {      :
/*SKY74137*/               :
/*SKY74137*/        }
/*SKY74137*/        case  FrequencyBand1800 :
/*SKY74137*/        {      :
/*SKY74137*/               :
/*SKY74137*/        }
/*SKY74137*/        case  FrequencyBand1900 :
/*SKY74137*/        {      :
/*SKY74137*/               :
/*SKY74137*/        }
/*SKY74137*/        default :
/*SKY74137*/        {      :
/*SKY74137*/               :
/*SKY74137*/        }
/*SKY74137*/    }
/*SKY74137*/    *ifN = 0x3 | (Nfrac<<2);
/*SKY74137*/}
```

2.10.3.2 Set Transceiver in Transmit Mode

109QB ahead of T0, MT62xx Baseband activates control pins by changing BPI bus states, To perform these operations, the timing and data of BPI bus can be implemented as follow steps:

(1) Modified the timing definition in **l1d_custom_rf.h**. **QB_PT2** is the timing definition of 2nd BPI bus status changing of TX window, **QB_PT2B** is the timing definition of 3rd BPI bus status changing of TX window. The PT2B is used to open the TRSW and the TXEN separately according to the performance requirement of the Helios2.

```
#define  QB_PT2          109
#define  QB_PT2B         1
```

(2) Define the states of BPI bus of PT2 for each band. The data is shown in the below table:

BPI_BUS	9	8	7	6	5	4	3	2	1	0	Hex
Function	SYN_EN	Reserved	PAEN	PAEN	BANDSW	TXENA	RXENA	Vc1	Vc3	Vc2	
850	1	X	0	1	0	1	0	0	0	0	250h
GSM	1	X	0	1	0	1	0	0	0	0	250h
DCS	1	X	0	1	1	1	0	0	0	0	270h
PCS	1	X	0	1	1	1	0	0	0	0	270h

So the BPI data can be defined as follow:

```
#define  PDATA_GSM850_PT2    0x250
#define  PDATA_GSM_PT2       0x250
#define  PDATA_DCS_PT2       0x270
#define  PDATA_PCS_PT2       0x270
```

(3) Define the states of BPI bus of PT2B for each band. The data is shown in the below table:

BPI_BUS	9	8	7	6	5	4	3	2	1	0	Hex
Function	SYN_EN	Reserved	PAEN	PAEN	BANDSW	TXENA	RXENA	Vc1	Vc3	Vc2	
850	1	X	0	1	0	1	0	0	0	1	251h
GSM	1	X	0	1	0	1	0	0	0	1	251h
DCS	1	X	0	1	1	1	0	0	1	0	272h
PCS	1	X	0	1	1	1	0	0	1	0	272h

So the BPI data can be defined as follow:

```
#define  PDATA_GSM850_PT2B    0x251
#define  PDATA_GSM_PT2B       0x251
#define  PDATA_DCS_PT2B       0x272
#define  PDATA_PCS_PT2B       0x272
```

2.10.3.3 TX DAC Control

The timing setting for TX frame enable and frame synchronization control of MT62xx are needed. TX frame enable is used to enable or disable TX DAC. TX frame synchronization is used to turn on/off TX SPORT. The timing of enabling TX DAC to turning on TX SPORT named **QB_TX_FENA_2_FSYNC** and the timing of turning off TX SPORT to disabling TX DAC named **QB_TX_FSYNC_2_FENA** are both also needed to be defined in **l1d_custom_rf.h**. In this case, the timing of enabling TX DAC to turning on TX SPORT is 164 QB and the timing of turning off TX SPORT to disabling TX DAC is 30 QB. So **QB_TX_FENA_2_FSYNC** and **QB_TX_FSYNC_2_FENA** can be defined as:

```
#define  QB_TX_FENA_2_FSYNC    164
#define  QB_TX_FSYNC_2_FENA    30
```

2.10.3.4 Set APC

99 QB ahead of T0, APC DAC output performs a DC offset to let ramp up more smoothly. The timing of performing this DC offset is defined as **QB_APCDACON** in **l1d_custom_rf.h**. The value of APC DC offset is maintained in APC profile and it is introduced in the latter section.

```
#define  QB_APCDACON    99
```

21 QB ahead of T0, APC DAC output performs ramping up. The timing of performing this ramping up is defined as **QB_APCON** in **l1d_custom_rf.h**.

```
#define  QB_APCON    21
```

3 QB behind T1 , APC DAC output performs ramping down. The timing of performing this ramping down is defined as **QB_APCOFF** in **l1d_custom_rf.h**.

```
#define  QB_APCOFF    3
```

2.10.3.5 Set Transceiver in Idle Mode

27 QB behind T1, RF module finishes transmission data and MT62xx set it into idle mode. MT62xx Baseband activates control pins by changing BPI bus states, and BSI command is not needed but the timing has to be defined. To perform these operations, the timing and data of BPI and BSI bus can be implemented as follow steps:

(1) Modified the timing definition in **l1d_custom_rf.h**. **QB_ST3** is the timing definition of 3'rd BSI command of TX window. **QB_PT3** is the timing definition of 3'rd BPI bus status changing of TX window.

```
#define  QB_ST3          0
#define  QB_PT3          27
```

(2) Define the data of BPI bus states for each band. The state of PT3 of each band is shown in the table:

BPI_BUS	9	8	7	6	5	4	3	2	1	0	Hex
Function	SYN_EN	Reserved	PAEN	PAEN	BANDSW	TXENA	RXENA	Vc1	Vc3	Vc2	
850	0	X	0	0	0	0	0	0	0	0	000h
GSM	0	X	0	0	0	0	0	0	0	0	000h
DCS	0	X	0	0	0	0	0	0	0	0	000h
PCS	0	X	0	0	0	0	0	0	0	0	000h

So the PR3 data can be defined as follow:

```
#define  PDATA_GSM850_PT3    0x000
#define  PDATA_GSM_PT3       0x000
```

```
#define  PDATA_DCS_PT3  0x000
#define  PDATA_PCS_PT3  0x000
```

(3) Specify the command of BSI to be sent. Function **L1D_RF_SetSData_ST3** in **m12195.c** should be implemented to perform this action. However, the Helios2 does not need ST3 for entering idle mode, and so the L1D_RF_SetData_ST3 is an empty function.

```
/*SKY74137*/ void  L1D_RF_SetSData_ST3( void )
/*SKY74137*/ {
/*SKY74137*/ }
```

The propagation delay when transmit data from Baseband uplink DAC to antenna should be claimed in file **l1d_custom_rf.c**. This delay depends on different RF module. For Helios2 RF module, the delay is 45 QB

```
const  short  TxPropagationDelay = 45;
```

2.10.3.6 Events Activation

If Layer 1 plans to transmit 148-bit data from TDMA timer count 2147 QB to 2739 QB, the above setting applies to create the TX window.

(1) At TDMA timer count is 1847 QB (300 QB ahead of TX SPORT turn on), the serial command of setting operation frequency and is sent, and at the meanwhile, BPI status is changed.
(2) At TDMA timer count is 1887 QB (260 QB ahead of TX SPORT turn on), the 2nd serial command of setting operation frequency and is sent
(3) At TDMA timer count is 1983 QB (164 QB ahead of TX SPORT turn on), TX DAC is enabled.
(4) At TDMA timer count is 2038 QB (109 QB ahead of TX SPORT turn on), BPI bus changes its states to TX mode.
(5) At TDMA timer count is 2048 QB (99 QB ahead of TX SPORT turn on), APC DAC output performs a DC offset to let ramp up more smoothly.
(6) At TDMA timer count is 2126 QB (21 QB ahead of TX SPORT turn on), APC DAC output performs ramping up.
(7) At TDMA timer count is 2146 QB (1 QB ahead of TX SPORT turn on), BPI bus changes its states to turn on the TRXSW.
(8) At TDMA timer count is 2147 QB, TX SPORT in MT62xx is turned on.
(9) At TDMA timer count is 2739 QB, TX SPORT in MT62xx is turned off.
(10) At TDMA timer count is 2744 QB (5 QB behind TX SPORT turning off), APC DAC output performs ramping down.
(11) At TDMA timer count is 2766 QB (27 QB behind TX SPORT turning off), BPI bus changes its states to let RF Helios2 into idle mode.
(12) At TDMA timer count is 2769 QB (30 QB behind TX SPORT turn off), TX DAC is disabled at this time.

2.10.4 TX window modulation type setup

For the TX window, two type of the modulation, GMSK/EPSK can be selected according to the EDGE mode pin of the BPI bus. For the GMSK modulation, the pin should be high, and for the EPSK modulation, the pin should be low. The correlated settings could be found in file **l1d_custom_rf.c**. The settings should be correlated to the EDGE mode pin of the BPI bus.

```
#define  PDATA_GMSK               0x080
#define  PDATA_EPSK               0x000
```

The usage of these two definitions could be found in file **m12195.c**.

```
void  L1D_RF_SetPData_PT( void )
{  APBADDR      _reg;
   const short *_d16p;

   _reg  = PDATA_REG_TABLE[ l1d_rf.cwin_idx ];
   _d16p = PDATA_TABLE[ l1d_rf.band ][ RF_TX ];
   if ( (l1d_rf.tx_mod_type) & (1<<l1d_rf.modidx) )
   {
      HW_WRITE( _reg, ((*_d16p)|(PDATA_8PSK)) );   _reg+=2;   _d16p++;
      HW_WRITE( _reg, ((*_d16p)|(PDATA_8PSK)) );   _reg+=2;   _d16p++;
      HW_WRITE( _reg, ((*_d16p)|(PDATA_8PSK)) );   _reg+=2;   _d16p++;
      HW_WRITE( _reg, (*_d16p) );//PT3 is end, doesn't need to | PDATA_8PSK
      /* PT3A */
      _d16p++;
      _reg = PDATA_REG_TABLE2[ l1d_rf.cwin_idx ];
      HW_WRITE( _reg, (*_d16p) );
   }
   else
   {
      HW_WRITE( _reg, ((*_d16p)|(PDATA_GMSK)) );   _reg+=2;   _d16p++;
      HW_WRITE( _reg, ((*_d16p)|(PDATA_GMSK)) );   _reg+=2;   _d16p++;
      HW_WRITE( _reg, ((*_d16p)|(PDATA_GMSK)) );   _reg+=2;   _d16p++;
      HW_WRITE( _reg, (*_d16p) );//PT3 is end, doesn't need to | PDATA_GMSK
      /* PT3A */
      _d16p++;
      _reg = PDATA_REG_TABLE2[ l1d_rf.cwin_idx ];
      HW_WRITE( _reg, (*_d16p) );
   }
}
```

The function **L1D_RF_SetPData_PT** is used to setup the BPI event data of the PT1, PT2, PT2B and PT3. Except the PT3, BPI event data setup in file **l1d_custom_rf.c** should append the PDATA_GMSK or the PDATA_8PSK according to the modulation type of the TX window.

As for the interslot modulation changing of the TX window, the BPI event named PT2M is used. The changing from the GMSK to the EPSK, the EPSK mode pin should be changed from the high status to the low status, as the result, the following definitions in file **l1d_custom_rf.c** according to the operating band is defined.

```
#define  PDATA_GSM850_PT2M1_G8    0x251
#define  PDATA_GSM_PT2M1_G8       0x251
#define  PDATA_DCS_PT2M1_G8       0x272
#define  PDATA_PCS_PT2M1_G8       0x272
```

Note that the team **G8** is used to identify the changing from the GMSK to the EPSK (8PSK). In fact, the definition is decided by the **PDATA_PT2B | PDATA_8PSK**.

On the other hand, the changing from the EPSK to the GPSK, the EPSK mode pin should be changed from the low status to the high status, as the result, the following definitions in file **l1d_custom_rf.c** according to the operating band is defined.

```
#define  PDATA_GSM850_PT2M1_8G    0x2D1
#define  PDATA_GSM_PT2M1_8G       0x2D1
#define  PDATA_DCS_PT2M1_8G       0x2F2
#define  PDATA_PCS_PT2M1_8G       0x2F2
```

Note that the team **8G** is used to identify the changing from the EPSK to the GPSK. In fact, the definition is decided by the **PDATA_PT2B | PDATA_GPSK**.

For the event timing of the PT2M could be found in file **l1d_custom_rf.c**. 0qB ahead of the SPORT ON of the 2nd TX window, the modulation type must be changed. The definitions are as followings.

```
#define  QB_PT2M1_G8                 0
#define  QB_PT2M1_8G                 0
```

2.11 RF Control Configuration of MT6140 RF Module

For example, the schematic circuit interface between MT6140 RF module and MT6229 chip is shown in below diagram. Please refer to MediaTek MT6140 RF Module schematic circuit to get more detail information.

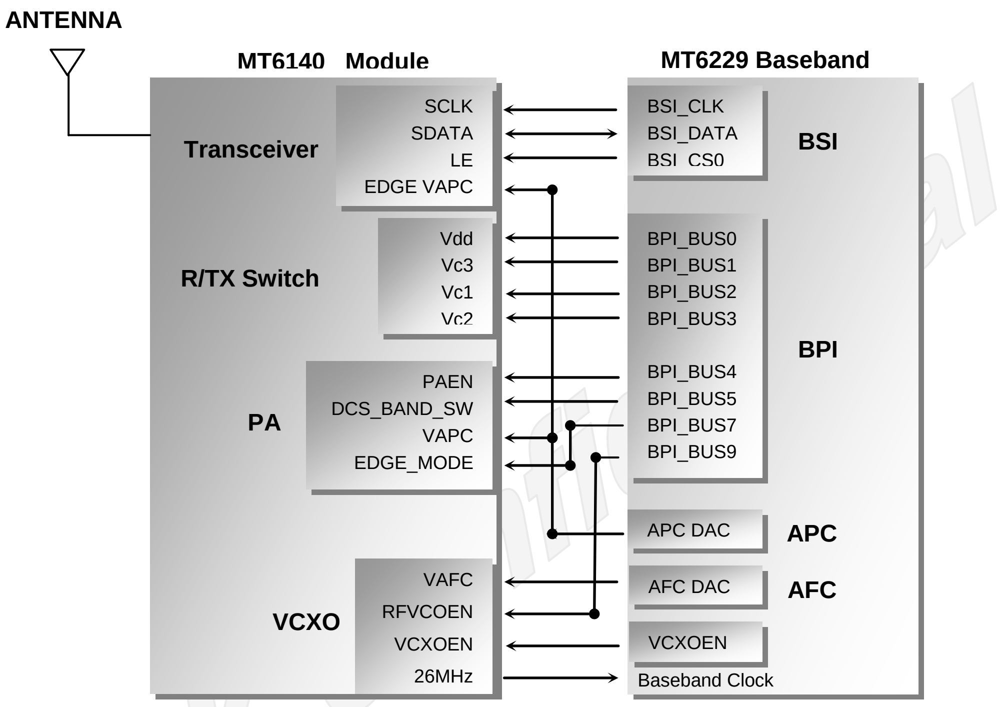


Figure 16 The connection of MT6229 and MT6140 RF Module

BSI_CLK, BSI_DATA, BSI_CS0 pins of MT6229 are directly connected to the serial interface of MT6140 transceiver. APC_DAC pin of MT6229 is connected to the VAPC pin of PA and the EDGE VAPC pin of MT6140 of RF module board. VCXOEN of MT6229 controls the enable state of VCXO. AFC DAC output is connected to the voltage control pin VAFC of VCXO. The generated clock of VCXO provides MT6229 system clock. Band-related pins Vdd, Vc3, Vc1, Vc2, PAEN, DCS_BAND_SW are controlled by the bit 0,1,2,3,4,5 of BPI bus of MT6229. PA EDGE_MODEis controled by the bit 7 of BPI bus of MT6229. RFVCOEN of MT6140 is controled by the bit 9 of BPI bus of MT6229. The high/low state of each pin of BPI bus and their timing of changing state can be programmed by MT6229. The RF radio control method is introduced in the latter description.

2.11.1 Initialization

RF module can be controlled by the BSI, BPI, APC, AFC units of MT6229 chip. BSI, BPI, APC, AFC units can work in 2 modes. One mode is immediate control mode, and the other is TDMA event driven control mode.

Immediate control mode is that the output signal of BSI, BPI, APC, or AFC unit is generated as soon as setting the corresponding registers in MT6229. When turning on the RF module, the immediate mode can be used to initialize the RF module. For example, immediate control mode is used when initializing MT6140 RF module, and the initial states are set as follow:

(1) Vdd (BPI_BUS0) is low state
(2) Vc3 (BPI_BUS1) is low state
(3) Vc1 (BPI_BUS2) is low state
(4) Vc2 (BPI_BUS3) is low state
(5) PAEN (BPI_BUS4) is low state
(6) DCS_BAND_SW (BPI_BUS5) is low state
(5) EDGE_MODE (BPI_BUS7) is low state
(6) RFVCOEN (BPI_BUS9) is low state
(7) Serial commands SDATA_TABLE[0:13] are sending to MT6140 transceiver.

The initialization of BPI, BSI data can be implemented in **L1D_RF_PowerOn** in **m12196.c** by using immediate mode:

```
/*MT6140D*/  unsigned long SDATA_TABLE[14]=
/*MT6140D*/ {
/*MT6140D*/   0x0004090,// 0:CW0
/*MT6140D*/   0x013F001,// 1:CW1
/*MT6140D*/   0x20823E2,// 2:CW2
/*MT6140D*/   0x07F3FF3,// 3:CW3
/*MT6140D*/   0x00A8714,// 4:CW4
/*MT6140D*/   0x00041F5,// 5:CW5
/*MT6140D*/   0x001D016,// 6:CW6
/*MT6140D*/   0x0001957,// 7:CW7
/*MT6140D*/   0x0FA0408,// 8:CW8
/*MT6140D*/   0x00E0409,// 9:CW9
/*MT6140D*/   0x002040A,//10:CW10
/*MT6140D*/   0x200070B,//11:CW11
/*MT6140D*/   0x201370C,//12:CW12
/*MT6140D*/   0x000000F //13:CW15
/*MT6140D*/ };
/*MT6140D*/ void  L1D_RF_PowerOn( void )
/*MT6140D*/ {
/*MT6140D*/    IMM_MODE_BEGIN( IMMMASK_ALL );
/*MT6140D*/    IMM_SEND_BPI( 0x0000 );
/*MT6140D*/    IMM_SEND_BSI( SCTRL_IMOD(0,26), 0x0044090 ); //CW0
/*MT6140D*/    #ifdef RFVCO_SW_CONTROL
/*MT6140D*/             IMM_SEND_BSI(  SCTRL_IMOD(0,26),  0x0008090|(XO_CapID<<9)  );
/*MT6140D*/    #else
/*MT6140D*/             IMM_SEND_BSI(  SCTRL_IMOD(0,26),  0x0000090|(XO_CapID<<9)  );
/*MT6140D*/    #endif
/*MT6140D*/    /* POR Start */
/*MT6140D*/    IMM_SEND_BSI( SCTRL_IMOD(0,26), SDATA_TABLE[1] ); // 1:CW1
/*MT6140D*/    IMM_SEND_BSI( SCTRL_IMOD(0,26), SDATA_TABLE[2] ); // 2:CW2
/*MT6140D*/    IMM_SEND_BSI( SCTRL_IMOD(0,26), SDATA_TABLE[3] ); // 3:CW3
/*MT6140D*/    IMM_SEND_BSI( SCTRL_IMOD(0,26), SDATA_TABLE[4] ); // 4:CW4
/*MT6140D*/    IMM_SEND_BSI( SCTRL_IMOD(0,26), SDATA_TABLE[5] ); // 5:CW5
/*MT6140D*/    IMM_SEND_BSI( SCTRL_IMOD(0,26), SDATA_TABLE[6] ); // 6:CW6
/*MT6140D*/    IMM_SEND_BSI( SCTRL_IMOD(0,26), SDATA_TABLE[7] ); // 7:CW7
/*MT6140D*/    IMM_SEND_BSI( SCTRL_IMOD(0,26), SDATA_TABLE[8] ); // 8:CW8
/*MT6140D*/    IMM_SEND_BSI( SCTRL_IMOD(0,26), SDATA_TABLE[9] ); // 9:CW9
/*MT6140D*/    IMM_SEND_BSI( SCTRL_IMOD(0,26), SDATA_TABLE[10] );//10:CW10
/*MT6140D*/    IMM_SEND_BSI( SCTRL_IMOD(0,26), SDATA_TABLE[11] );//11:CW11
/*MT6140D*/    IMM_SEND_BSI( SCTRL_IMOD(0,26), SDATA_TABLE[12] );//12:CW12
/*MT6140D*/    IMM_SEND_BSI( SCTRL_IMOD(0,26), SDATA_TABLE[13] );//13:CW15
/*MT6140D*/    /* POR End */
/*MT6140D*/    #ifndef L1D_SIM
/*MT6140D*/    if(l1d_rf.is_init)
/*MT6140D*/    {  /* for power on tx calibration, we program mt6140 to GSM TX mode */
/*MT6140D*/       IMM_SEND_BSI( SCTRL_IMOD(0,26), 0x0003002L );
/*MT6140D*/    }
/*MT6140D*/    #endif
/*MT6140D*/    IMM_MODE_END(  );
/*MT6140D*/}
```

2.11.2 Create RX window

TDMA timer is a counter unit that counts 1 per quarter-bit (0.923u). It always repeats counting from 0 to 4999. So the period of TDMA counter is 5000 QB (1 TDMA frame). If BSI, BPI, AFC, APC are programmed in TDMA event driven mode, the active time can be programmed. When the TDMA counter counts to the programmed time (count), the corresponding unit is triggered to activate.

If it is needed to receive data from air or transmit data to air, Layer1 needs to plan all RF setting and timing to create RX/TX window, control receiving gain, control VCXO, control PA... etc. For this purpose, TDMA event driven mode of BSI, BPI, APC, or AFC units should be used.

For example of creating RX window to receive data, the RF setting and timing of MT6140 RF module can be depicted in the below diagram:

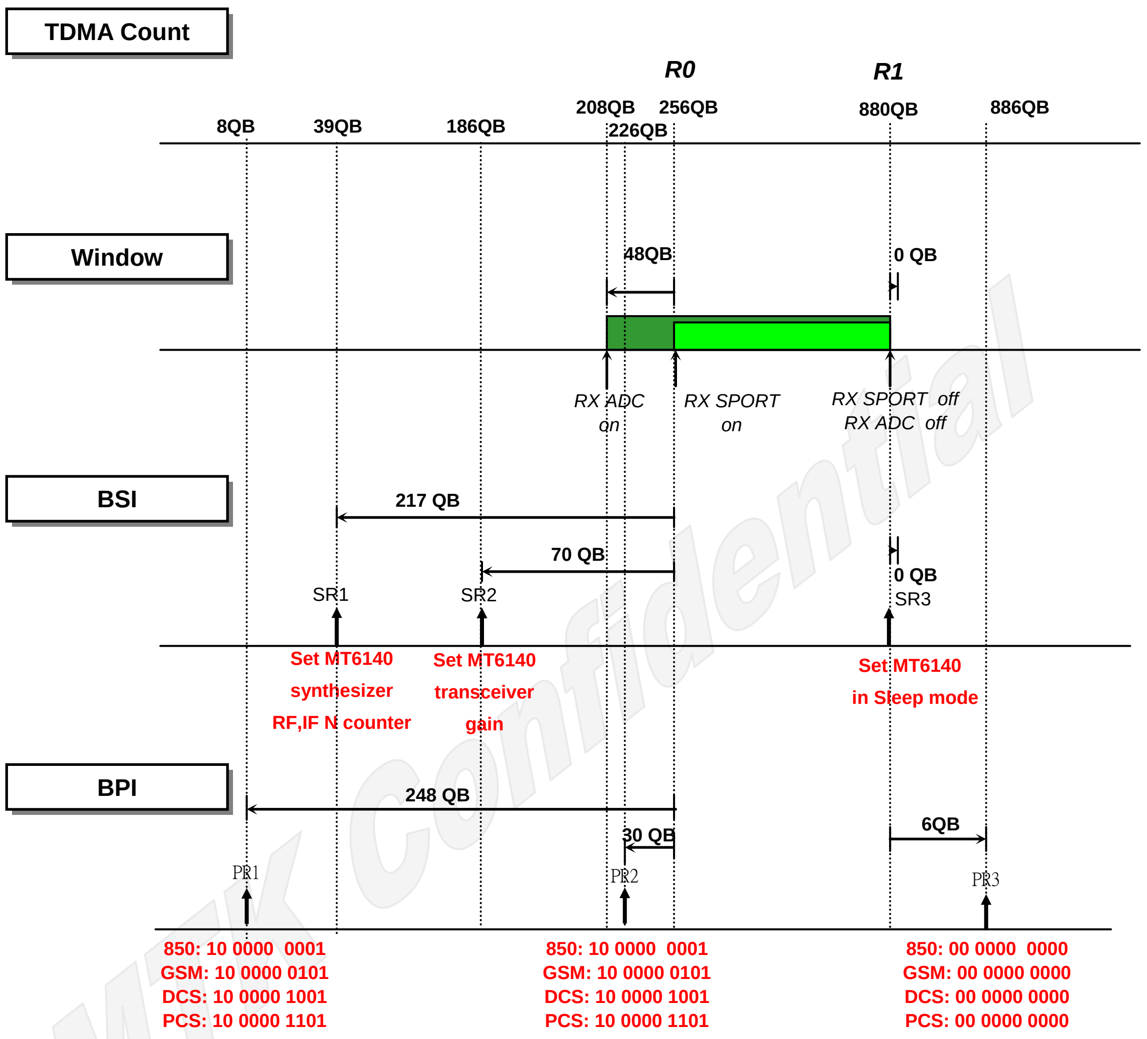


Figure 17 MT6140 Event's timing of RX window

The TDMA timing and data setting are described in 2 parts. One part is that before turning on RX SPORT to receiving I/Q data from MT6140 transceiver. And the other part is after turning off RX SPORT to finish receiving I/Q data. To describe these two parts easily, the timing of turning on RX SPORT is taken as one base named **R0**. And the timing of turning off RX SPORT is taken as one base named **R1**. In this case, R0 is at TDMA counter 256 QB and R1 is at TDMA counter 880 QB.

2.11.2.1 Set Synthesizer N-Counter

248 QB ahead of R0, RF module should start to operate. MT6229 chip controls the RF module by activating control pins and sending serial command. MT6229 activates the control pins by changing BPI bus states. The high/low state of BPI bus at this time depends on operation band. After 31 QB, MT6229 also sends serial command to lock MT6140 synthesizer to lock the operation frequency by BSI bus. To perform these operations, the timing and data of BPI and BSI bus can be implemented as follow steps:

(1) Modified the timing definition in **l1d_custom_rf.h**. **QB_SR1** is the timing definition of 1'st BSI command of RX window. **QB_PR1** is the timing definition of 1'st BPI bus status changing of RX window.

```
#define   QB_SR1           250
#define   QB_PR1           217
```

(2) Define the states of BPI bus for each band. The data is shown in the below table:

BPI_BUS	9	8	7	6	5	4	3	2	1	0	Hex
Function	RFVCOEN	Not Used	EDGE Mode	Not Used	DCS BAND_SW	PAEN	Vc2	Vc1	Vc3	Vdd	
850	1	X	X	X	0	0	0	0	0	1	201h
GSM	1	X	X	X	0	0	0	1	0	1	205h
DCS	1	X	X	X	0	0	1	0	0	1	209h
PCS	1	X	X	X	0	0	1	1	0	1	20Dh

So the BPI data can be defined as follow:

```
#define  PDATA_RFDOO   0x200
#define  PDATA_GSM850_PR1      ( 0x001 | PDATA_RFDOO )
#define  PDATA_GSM_PR1         ( 0x005 | PDATA_RFDOO )
#define  PDATA_DCS_PR1         ( 0x009 | PDATA_RFDOO )
#define  PDATA_PCS_PR1         ( 0x00D | PDATA_RFDOO )
```

(3) Specify the command of BSI to be sent. Function **L1D_RF_SetSData_SR1** in **m12195.c** be implemented to perform this action. **SETUP_SR1()** is the first statement in **L1D_RF_SetSData_SR1** which is a macro to hide complicated code to initialize setting 1'st BSI data. The second statement **HWRITE_3_SDATA** is specified the data to be sent by BSI bus. Two parameters of HWRITE_3_SDATA are control word and data word of BSI data. The control words **SCTRL_WARM**, and **SCTRL_PLL** are composed of device selection (0) and data bit count (26 bits). The 1st data word **SDATA_WARM** is the command to set MT6140 into Warm-up mode. The 2nd data word **l1d_rf.RFN_data**, RF N-Counter setting value, is the evaluation result of **L1D_RF_GetRxPLLSetting** called by Layer 1. The 3rd data word is reserved.

```
/*MT6140D*/ #define  SCTRL_WARM                    SCTRL_WORD(0,26)
/*MT6140D*/ #define  SCTRL_PLL                     SCTRL_WORD(0,26)
/*MT6140D*/ const unsigned long SDATA_WARM  = 0x20827E2L; // Warm up mode
/*MT6140D*/ void  L1D_RF_SetSData_SR1( void )
/*MT6140D*/ {
/*MT6140D*/    SETUP_SR1();
/*MT6140D*/    HWRITE_3_SDATA(SCTRL_WARM,    SDATA_WARM,
/*MT6140D*/                   SCTRL_PLL,     l1d_rf.RFN_data,
/*MT6140D*/                   SCTRL_RESERVED, 0 );
/*MT6140D*/ }
```

To create the receiving window with right operation frequency, evaluating the setting value of operation frequency is needed. Function **L1D_RF_GetRxPLLSetting** in file **m12191.c** needs to be implemented to evaluate the synthesizer setting. And Layer 1 calls this function when programs the RF synthesizer setting value and saves the result into variable **l1d_rf.RFN_data**. This content of this variable is the command to be sent to transceiver chip to lock the operation frequency. The evaluation of MT6140 synthesizer N counter setting is as the following code:

```
void  L1D_RF_GetRxPLLSetting( int rf_band, int arfcn, long *rfN, long *ifN )
/*MT6140D*/{
/*MT6140D*/   switch(rf_band)
/*MT6140D*/   {
/*MT6140D*/      case  FrequencyBand850 :
/*MT6140D*/      {
/*MT6140D*/         if(arfcn<=201)
/*MT6140D*/         {  if(arfcn<=136)
/*MT6140D*/            {                                 /* ARFCN : 128~136     */
/*MT6140D*/               *rfN = (66L<<12) | ((arfcn-72)<<5) | 0x000001L;
/*MT6140D*/            }
/*MT6140D*/            else
/*MT6140D*/              {                              /* ARFCN : 137~201    */
/*MT6140D*/                  *rfN = (67L<<12) | ((arfcn-137)<<5) | 0x000001L;
/*MT6140D*/              }
/*MT6140D*/           }
/*MT6140D*/           else
/*MT6140D*/           {                                 /* ARFCN : 202~251    */
/*MT6140D*/                  *rfN = (68L<<12) | ((arfcn-202)<<5) | 0x000001L;
/*MT6140D*/           }
/*MT6140D*/
/*MT6140D*/           break;
/*MT6140D*/        }
/*MT6140D*/        case  FrequencyBand900 :
/*MT6140D*/        { :
/*MT6140D*/          :
/*MT6140D*/           break;
/*MT6140D*/        }
/*MT6140D*/        case  FrequencyBand1800 :
/*MT6140D*/        { :
/*MT6140D*/          :
/*MT6140D*/           break;
/*MT6140D*/        }
/*MT6140D*/        case  FrequencyBand1900 :
/*MT6140D*/        { :
/*MT6140D*/          :
/*MT6140D*/           break;
/*MT6140D*/        }
/*MT6140D*/        default :
/*MT6140D*/        {
/*MT6140D*/           break;
/*MT6140D*/        }
/*MT6140D*/     }
/*MT6140D*/     *ifN = 0;
/*MT6140D*/}
```

2.11.2.2 Set Transceiver Gain

70 QB ahead of R0, Band selection and Rx gain setting should be performed. MT6229 activates control pins by changing BPI bus states, and also sends serial command to set transceiver gain by BSI bus. To perform these operations, the timing and data of BPI and BSI bus can be implemented as follow steps:

(1) Modified the timing definition in **l1d_custom_rf.h**. **QB_SR2** is the timing definition of 2'nd BSI command of RX window. **QB_PR2** is the timing definition of 2'nd BPI bus status changing of RX window.

```
#define  QB_SR2            70
#define  QB_PR2            30
```

(2) Define the data of BPI bus states for each band. The state of BPI of each band is shown in the table:

BPI_BUS	9	8	7	6	5	4	3	2	1	0	Hex
Function	RFVCOEN	Not Used	EDGE Mode	Not Used	DCS BAND_SW	PAEN	Vc2	Vc1	Vc3	Vdd	
850	1	X	X	X	0	0	0	0	0	1	201h
GSM	1	X	X	X	0	0	0	1	0	1	205h
DCS	1	X	X	X	0	0	1	0	0	1	209h
PCS	1	X	X	X	0	0	1	1	0	1	20Dh

So the BPI data can be defined as follow:

```
#define  PDATA_GSM850_PR2      ( 0x001 | PDATA_RFDOO )
#define  PDATA_GSM_PR2         ( 0x005 | PDATA_RFDOO )
#define  PDATA_DCS_PR2         ( 0x009 | PDATA_RFDOO )
#define  PDATA_PCS_PR2         ( 0x00D | PDATA_RFDOO )
```

(3) Specify the command of BSI to be sent. Function **L1D_RF_SetSData_SR2** in **m12195.c** should be implemented to perform this action. The first statement in **L1D_RF_SetSData_SR2** is **SETUP_SR2()** which is a macro to hide complicated code to initialize setting 2'nd BSI data. The second statement **HWRITE_1_SDATA** is specified the data to be sent by BSI bus. The parameters of **HWRITE_1_SDATA** are control word and data word of BSI data. The control word **SCTRL_AGC** is composed of device selection (0) and data bit count (26 bits). The data word is composed of two commands. The 1st command **l1d_rf.AGC_data**, transceiver gain setting value, is the evaluation result of **L1D_RF_GetGainSetting** called by Layer 1.

```
/*MT6140D*/ #define  SCTRL_AGC                          SCTRL_WORD(0,26)
/*MT6140D*/ void  L1D_RF_SetSData_SR2( void )
/*MT6140D*/ {
/*MT6140D*/    SETUP_SR2();
/*MT6140D*/
/*MT6140D*/ HWRITE_1_SDATA( SCTRL_AGC,     SDATA_RXMODE|l1d_rf.AGC_data|auto_cal);
/*MT6140D*/
/*MT6140D*/ }
```

To create the receiving window with suitable gain of receiving amplifier, evaluating the setting value of request gain is needed. Function **L1D_RF_GetGainSetting** in file **m12192.c** needs to be implemented to evaluate the transceiver gain setting. And Layer 1 calls this function when programs the gain setting value and saves the result into variable **l1d_rf.AGC_data**. This content of this variable is the command to be sent to transceiver chip to set the requested receive amplifier gain. The evaluation of MT6140 receiving amplifier gain setting is as the following code:

```
    /*MT6140D*/ int    L1D_RF_GetGainSetting( int rf_band, int arfcn, int request_gain,
long *gain_setting )
    /*MT6140D*/ {  int   max_gain, min_gain, real_gain;
    /*MT6140D*/    const int*    d32ptr;
    /*MT6140D*/    const sL1DAGCDATA*  agcptr;
    /*MT6140D*/    long  setting;
    /*MT6140D*/    int   bit_no;
    /*MT6140D*/
    /*MT6140D*/    /* evaluate the range of available gain */
    /*MT6140D*/    d32ptr  = (int*)&(GAIN_RANGE_TABLE[rf_band]);
    /*MT6140D*/    max_gain = *d32ptr++;
    /*MT6140D*/    min_gain = *d32ptr;
    /*MT6140D*/
    /*MT6140D*/    /* clipping the request gain to the avialable gain */
    /*MT6140D*/    if( request_gain>=max_gain )
    /*MT6140D*/    {  request_gain = max_gain;  }
    /*MT6140D*/    else  if( request_gain<=min_gain )
    /*MT6140D*/    {  request_gain = min_gain;  }
    /*MT6140D*/
    /*MT6140D*/    /* evaluate the real gain setting */
    /*MT6140D*/    agcptr = AGC_TABLE[rf_band];
    /*MT6140D*/    if( request_gain < agcptr->pos_gain )
    /*MT6140D*/    {  agcptr++;
    /*MT6140D*/    }
    /*MT6140D*/    {  bit_no    = BIT_NO( request_gain, agcptr->A, GC_B );
    /*MT6140D*/       real_gain = REAL_GAIN( bit_no, agcptr->A, GC_B );
    /*MT6140D*/        setting   = agcptr->setting + (bit_no<<4);
    /*MT6140D*/     }
    /*MT6140D*/
    /*MT6140D*/     *gain_setting = setting;
    /*MT6140D*/     return( real_gain );
    /*MT6140D*/ }
```

2.11.2.3 Set Transceiver in Idle Mode

6 QB behind R1, RF module finishes receiving data and MT6229 set it into idle mode. MT6229 Baseband activates control pins by changing BPI bus states, and also sends serial command to set idle mode command by BSI bus. To perform these operations, the timing and data of BPI and BSI bus can be implemented as follow steps:

(1) Modified the timing definition in **l1d_rf.h**. **QB_SR3** is the timing definition of 3'rd BSI command of RX window. **QB_PR3** is the timing definition of 3'rd BPI bus status changing of RX window.

```
#define  QB_SR3             0
#define  QB_PR3             6
```

(2) Define the data of BPI bus states for each band. The state of BPI of each band is shown in the table:

BPI_BUS	9	8	7	6	5	4	3	2	1	0	Hex
Function	RFVCOEN	Not Used	EDGE Mode	Not Used	DCS BAND_SW	PAEN	Vc2	Vc1	Vc3	Vdd	
850	0	X	X	X	0	0	0	0	0	0	000h
GSM	0	X	X	X	0	0	0	0	0	0	000h
DCS	0	X	X	X	0	0	0	0	0	0	000h
PCS	0	X	X	X	0	0	0	0	0	0	000h

So the BPI data can be defined as follow:

```
#define  PDATA_GSM850_PR3     0x000
#define  PDATA_GSM_PR3        0x000
#define  PDATA_DCS_PR3        0x000
#define  PDATA_PCS_PR3        0x000
```

(3) Specify the command of BSI to be sent. Function **L1D_RF_SetSData_SR3** in **m12195.c** should be implemented to perform this action. The first statement in **L1D_RF_SetSData_SR3** is **SETUP_SR3()** which is a macro to hide complicated code to initialize setting 3'rd BSI data. The second statement **HWRITE_1_SDATA** is specified the data to be sent by BSI bus. The parameters of **HWRITE_1_SDATA** are control word and data word of BSI data. The control word SCTRL_IDLE is composed of device selection (0) and data bit count (26 bits). The data word SDATA_SLEEP is the command to set MT6140 into Sleep mode.

```
/*MT6140D*/ #define  SCTRL_IDLE                   SCTRL_WORD(0,26)
/*MT6140D*/ const unsigned long SDATA_SLEEP   = 0x0000002L; // Sleep mode
/*MT6140D*/ void  L1D_RF_SetSData_SR3( void )
/*MT6140D*/ {
/*MT6140D*/     SETUP_SR3();
/*MT6140D*/
/*MT6140D*/     HWRITE_1_SDATA( SCTRL_IDLE, SDATA_SLEEP);
/*MT6140D*/ }
```

2.11.2.4 RX ADC Control

Besides setting the timing and data of BPI and BSI bus of MT6229 Baseband to create a RX window, the timing setting for RX frame enable and frame synchronization control of MT6229 are needed. RX frame enable is used to enable or disable RX ADC. RX frame synchronization is used to turn on/off RX SPORT. The timing of enabling RX ADC to turning on RX SPORT named **QB_RX_FENA_2_FSYNC** and the timing of turning off RX SPORT to disabling RX ADC named **QB_RX_FSYNC_2_FENA** are both also needed to be defined in **l1d_custom_rf.h**. In this case, the timing of enabling RX ADC to turning on RX SPORT is recommend to be 40 QB and the timing of turning off RX SPORT to disabling RX ADC is 0 QB. So **QB_RX_FENA_2_FSYNC** and **QB_RX_FSYNC_2_FENA** can be defined as:

```
#define  QB_RX_FENA_2_FSYNC     48
#define  QB_RX_FSYNC_2_FENA     0
```

2.11.2.5 Events Activation

If Layer1 plans to receive 156-bit data from TDMA timer count 256 QB to 880 QB, the above setting applies to create the RX window.

(1) At TDMA timer count is 8 QB (247 QB ahead of RX SPORT turn on), BPI bus changes its states to 201h(GSM850) or 201h(GSM) or 209h(DCS) or 20Dh (PCS) .And after 31 QB (217 QB), the serial command of setting operation frequency is sent to MT6140 transceiver.

(2) At TDMA timer count is 186 QB (70 QB ahead of RX SPORT turn on), the serial command of setting RX gain is sent to MT6140 transceiver

(3) At TDMA timer count is 208 QB (48 QB ahead of RX SPORT turn on), RX ADC in MT6229 is enabled.

(4) At TDMA timer count is 226 QB (30 QB ahead of RX SPORT turn on), BPI bus changes its states to 201h(GSM850) or 201h(GSM) or 209h(DCS) or 20Dh (PCS).

(5) At TDMA timer count is 256 QB, RX SPORT in MT6229 is turned on. The I/Q data begin to be sampled.

(6) At TDMA timer count is 880 QB, RX SPORT in MT6229 is turned off and RX ADC is also disabled. The I/Q data stop sampling.

(7) At TDMA timer count is 880 QB (at RX SPORT turning off), the command of setting MT6140 into idle mode is sent.

(8) At TDMA timer count is 886 QB (6 QB behind RX SPORT turning off), BPI bus changes its states to 000h .

2.11.3 Create TX window

Another example is creating TX window to transmit data. The RF setting and timing of MT6139C RF module can be depicted in the below diagram:

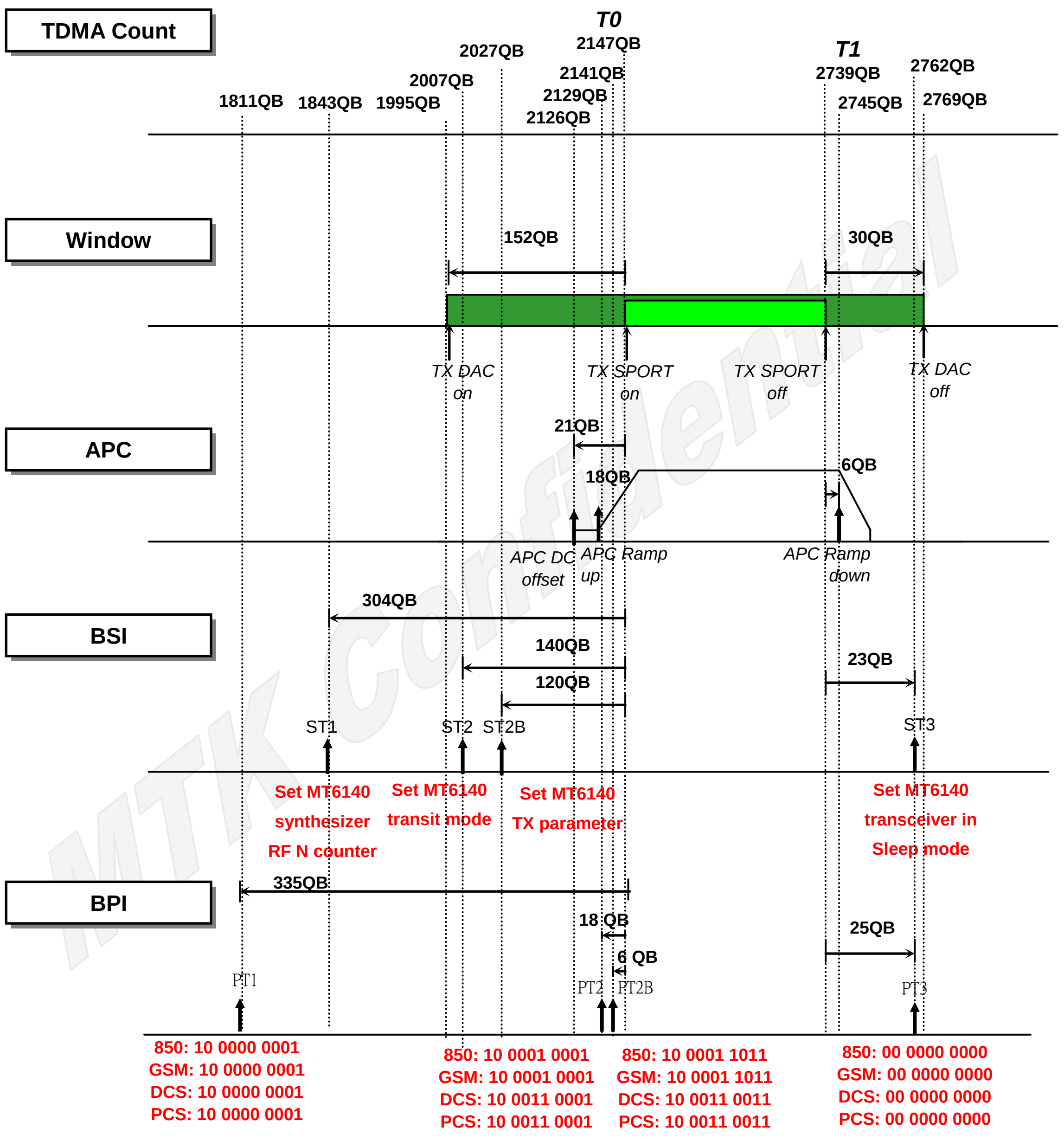


Figure 18 MT6140 Event's timing of TX window

The TDMA timing and data setting are described in 2 parts. One part is that before turning on TX SPORT to receiving I/Q data from MT6140 transceiver. And the other part is after turning off TX SPORT to finish transmission I/Q data. To describe these two parts easily, the timing of turning on TX SPORT is taken as one base named **T0**. And the timing of turning off TX SPORT is taken as one base named **T1**. In this case, T0 is at TDMA counter 2147 QB and T1 is at TDNA counter 2739 QB.

2.11.3.1 Set Synthesizer N-Counter

335 QB ahead of T0, RF module starts to work. MT6229 Baseband activates control pins by changing BPI bus states, Then MT6229 sends serial command to lock MT6140 synthesizer to lock the operation frequency by BSI bus. To perform this operation, the timing and data of BSI bus can be implemented as follow steps:

(1) Modified the timing definition in l1d_custom_rf.h. QB_ST1 is the timing definition of 1'st BSI command ofTX window. QB_PT1 is the timing definition of 1'st BPI bus status changing of TX window.

```
#define  QB_PT1             335
#define  QB_ST1             304
```

(2) Define the data of BPI bus states of each band. The data is shown in the below table:

BPI_BUS	9	8	7	6	5	4	3	2	1	0	Hex
Function	RFVCOEN	Not Used	EDGE Mode	Not Used	DCS BAND_SW	PAEN	Vc2	Vc1	Vc3	Vdd	
850	1	X	0	X	0	0	0	0	0	1	201h
GSM	1	X	0	X	0	0	0	0	0	1	201h
DCS	1	X	0	X	0	0	0	0	0	1	201h
PCS	1	X	0	X	0	0	0	0	0	1	201h

So the BPI data can be defined as follow:

```
#define  PDATA_GSM850_PT1     ( 0x001 | PDATA_RFDOO )
#define  PDATA_GSM_PT1        ( 0x001 | PDATA_RFDOO )
#define  PDATA_DCS_PT1        ( 0x001 | PDATA_RFDOO )
#define  PDATA_PCS_PT1        ( 0x001 | PDATA_RFDOO )
```

(3) Specify the command of BSI to be sent. Function **L1D_RF_SetSData_ST1** in **m12195.c** should be implemented to perform this action. **SETUP_ST1()** is the first statement in **L1D_RF_SetSData_ST1** which is a macro to hide complicated code to initialize setting 1'st BSI data. The second statement **HWRITE_3_SDATA** is specified the data to be sent by BSI bus. Two parameters of **HWRITE_3_SDATA** are control word and data word of BSI data. The control words **SCTRL_WARM** and **SCTRL_PLL** are composed of device selection (0) and data bit count (26 bits). The 1st data word **SDATA_WARM** is the command to set MT6140 into Warm-up mode. The 2nd data word **l1d_rf.RFN_data**, synthesizer N-counter setting value, is the evaluation result of **L1D_RF_GetTXPLLSetting** called by Layer 1. The 3nd data word is reserved.

```
/*MT6140D*/ #define  SCTRL_WARM                   SCTRL_WORD(0,26)
/*MT6140D*/ #define  SCTRL_PLL                    SCTRL_WORD(0,26)
/*MT6140D*/ void  L1D_RF_SetSData_ST1( void )
/*MT6140D*/ {
/*MT6140D*/    SETUP_ST1();
/*MT6140D*/    HWRITE_3_SDATA(
/*MT6140D*/                   SCTRL_WARM,          SDATA_WARM,
/*MT6140D*/                   SCTRL_PLL,           l1d_rf.RFN_data,
/*MT6140D*/                   SCTRL_RESERVED,         0);
/*MT6140D*/
/*MT6140D*/ }
```

To create the transmission window with right operation frequency, evaluating the setting value of operation frequency is needed. Function **L1D_RF_GetTxPLLSetting** in file **m12191.c** needs to be implemented to evaluate the synthesizer setting. And Layer 1 calls this function when program the RF synthesizer setting value and saves the result into variable **l1d_rf.RFN_data**. This content of this variable is the command to be sent to transceiver chip to lock the operation frequency. The function of MT6140 RF module can be implemented as follow:

```
void  L1D_RF_GetTxPLLSetting( int rf_band, int arfcn, long *rfN, long *ifN )
/*MT6140D*/{
```

```
/*MT6140D*/    switch(rf_band)
/*MT6140D*/    {
/*MT6140D*/       case  FrequencyBand850 :
/*MT6140D*/       {
/*MT6140D*/          if(arfcn<=231)
/*MT6140D*/          {  if(arfcn<=166)
/*MT6140D*/             {                                /* ARFCN : 128~166     */
/*MT6140D*/                     *rfN = (63L<<12) | ((arfcn-102)<<5) | 0x480001L;
/*MT6140D*/             }
/*MT6140D*/             else
/*MT6140D*/             {                               /* ARFCN : 167~231     */
/*MT6140D*/                     *rfN = (64L<<12) | ((arfcn-167)<<5) | 0x480001L;
/*MT6139C*/             }
/*MT6140D*/          }
/*MT6140D*/          else
/*MT6140D*/          {                                  /* ARFCN : 232~251     */
/*MT6140D*/                     *rfN = (65L<<12) | ((arfcn-232)<<5) | 0x480001L;
/*MT6140D*/          }
/*MT6140D*/
/*MT6140D*/          break;
/*MT6140D*/       }
/*MT6140D*/       case  FrequencyBand900 :
/*MT6140D*/       {      :
/*MT6140D*/              :
/*MT6140D*/       }
/*MT6140D*/       case  FrequencyBand1800 :
/*MT6140D*/       {      :
/*MT6140D*/              :
/*MT6140D*/       }
/*MT6140D*/       case  FrequencyBand1900 :
/*MT6140D*/       {      :
/*MT6140D*/              :
/*MT6140D*/       }
/*MT6140D*/       default :
/*MT6140D*/       {      :
/*MT6140D*/              :
/*MT6140D*/       }
/*MT6140D*/    }
/*MT6140D*/    *ifN = 0;
/*MT6140D*/}
```

2.11.3.2 Set Transceiver in Transmit Mode

140 QB ahead of T0, MT6229 Baseband sends serial command to set transmit mode by BSI bus and after 20QB TX mode setting is performed, MT6229 Baseband sends serial command to set transmiter parameter by BSI bus and after 102QB, MT6229 Baseband activates control pins by changing BPI bus states, To perform these operations, the timing and data of BPI and BSI bus can be implemented as follow steps:

(1) Modified the timing definition in **l1d_custom_rf.h**. QB_ST2 is the timing definition of 2'nd BSI command of TX window. QB_ST2B is the timing definition of 3'rd BSI command of TX window. QB_PT2 is the timing definition of 2'nd BPI bus status changing of TX window.

```
#define  QB_ST2            140
#define  QB_ST2B           120
#define  QB_PT2            18
```

(2) Define the data of BPI bus states of each band. The data is shown in the below table:

BPI_BUS	9	8	7	6	5	4	3	2	1	0	Hex
Function	RFVCOEN	Not Used	EDGE Mode	Not Used	DCS BAND_SW	PAEN	Vc2	Vc1	Vc3	Vdd	

850	1	X	0	X	0	1	0	0	0	1	211h
GSM	1	X	0	X	0	1	0	0	0	1	211h
DCS	1	X	0	X	1	1	0	0	0	1	231h
PCS	1	X	0	X	1	1	0	0	0	1	231h

So the BPI data can be defined as follow:

```
#define  PDATA_GSM850_PT2      ( 0x011 | PDATA_RFDOO )
#define  PDATA_GSM_PT2         ( 0x011 | PDATA_RFDOO )
#define  PDATA_DCS_PT2         ( 0x031 | PDATA_RFDOO )
#define  PDATA_PCS_PT2         ( 0x031 | PDATA_RFDOO )
```

(3) Specify the command of BSI to be sent. Function **L1D_RF_SetSData_ST2** in **m12195.c** should be implemented to perform this action. **SETUP_ST2()** is the first statement in **L1D_RF_SetSData_ST2** which is a macro to hide complicated code to initialize setting 2'nd BSI data. The second statement **HWRITE_1_SDATA** is specified the data to be sent by BSI bus. Two parameters of **HWRITE_1_SDATA** are control word and data word of BSI data. The control word **SCTRL_TXBAND** is composed of device selection (0) and data bit count (26 bits). The data word **SDATA_TXMODE** is command to set MT6140 into TX mode..

```
/*MT6140D*/ #define  SCTRL_TXBAND                SCTRL_WORD(0,26)
/*MT6140D*/ void  L1D_RF_SetSData_ST2( void )
/*MT6140D*/ {
/*MT6140D*/    SETUP_ST2();
/*MT6140D*/    HWRITE_1_SDATA( SCTRL_TXBAND,   SDATA_TXMODE );
/*MT6140D*/ }
```

(4) Specify the command of BSI to be sent. Function **L1D_RF_SetSData_ST2B** in **m12195.c** should be implemented to perform this action. **SETUP_ST2M()** is the first statement in **L1D_RF_SetSData_ST2B** which is a macro to hide complicated code to initialize setting 3'rd BSI data. The second statement **HWRITE_1_SDATA** is specified the data to be sent by BSI bus. Two parameters of **HWRITE_1_SDATA** are control word and data word of BSI data. The control word **SCTRL_TXBAND** is composed of device selection (0) and data bit count (26 bits). The data word is command to set TX parameter of MT6140 .

```
/*MT6140D*/ #define  SCTRL_TXBAND                SCTRL_WORD(0,26)
/*MT6140D*/ void L1D_RF_SetSData_ST2B( void )
/*MT6140D*/ {
/*MT6140D*/    SETUP_ST2B_ST2M();
/*MT6140D*/    {
/*MT6140D*/ #if defined(__EPSK_TX__)
/*MT6140D*/       if((l1d_rf.tx_mod_type2>>l1d_rf.cur_slot)&0x1)
/*MT6140D*/       {
/*MT6140D*/       unsigned short   cw_idx;
/*MT6140D*/  cw_idx=L1D_RF_GetTxPAVBias_TxITC(LB_CW_High_Idx, HB_CW_High_Idx);
/*MT6140D*/HWRITE_1_SDATA(SCTRL_TXBAND,
 TXCW[(cw_idx>>4)&0x1][1][l1d_rf.band]|((cw_idx&0x7)<<8));
/*MT6140D*/       }
/*MT6140D*/       else
/*MT6140D*/ #endif
/*MT6140D*/       {
/*MT6140D*/          HWRITE_1_SDATA( SCTRL_TXBAND,   TXCW[0][0][l1d_rf.band]);
/*MT6140D*/       }
/*MT6140D*/    }
/*MT6140D*/ }
```

2.11.3.3 TX DAC Control

The timing setting for TX frame enable and frame synchronization control of MT6229 are needed. TX frame enable is used to enable or disable TX DAC. TX frame synchronization is used to turn on/off TX SPORT. The timing of enabling TX DAC to turning on TX SPORT named **QB_TX_FENA_2_FSYNC** and the timing of turning off TX SPORT to disabling TX DAC named **QB_TX_FSYNC_2_FENA** are both also needed to be defined in **l1d_custom_rf.h**. In this case, the timing of enabling TX DAC to turning on TX SPORT is 152 QB and the timing of turning off TX SPORT to disabling TX DAC is 30 QB. So **QB_TX_FENA_2_FSYNC** and **QB_TX_FSYNC_2_FENA** can be defined as:

```
#define  QB_TX_FENA_2_FSYNC     152
#define  QB_TX_FSYNC_2_FENA     30
```

2.11.3.4 Set APC

21 QB ahead of T0, APC DAC output performs a DC offset to let ramp up more smoothly. The timing of performing this DC offset is defined as **QB_APCDACON** in **l1d_custom_rf.h**. The value of APC DC offset is maintained in APC profile and it is introduced in the latter section.

```
#define  QB_APCDACON     21
```

8 QB ahead of T0, MT6229 Baseband changes BPI bus states to select operation right band.

(1) Modified the timing definition in **l1d_custom_rf.h**. **QB_PT2B** is the timing definition of 3'rd BPI bus status changing of TX window.

```
#define  QB_PT2B             6
```

(2) Define the data of BPI bus states of each band. The state of BPI of each band is shown in the table:

BPI_BUS	9	8	7	6	5	4	3	2	1	0	Hex
Function	RFVCOEN	Not Used	EDGE Mode	Not Used	DCS BAND_SW	PAEN	Vc2	Vc1	Vc3	Vdd	
850	1	X	X	X	0	1	1	0	1	1	21Bh
GSM	1	X	X	X	0	1	1	0	1	1	21Bh
DCS	1	X	X	X	1	1	0	0	1	1	233h
PCS	1	X	X	X	1	1	0	0	1	1	233h

So the BPI data can be defined as follow:

```
#define  PDATA_GSM850_PT2B     ( 0x01B | PDATA_RFDOO )
#define  PDATA_GSM_PT2B        ( 0x01B | PDATA_RFDOO )
#define  PDATA_DCS_PT2B        ( 0x033 | PDATA_RFDOO )
#define  PDATA_PCS_PT2B        ( 0x033 | PDATA_RFDOO )
```

21 QB ahead of T0, APC DAC output performs ramping up. The timing of performing this ramping up is defined as **QB_APCON** in **l1d_custom_rf.h**.

```
#define  QB_APCON     18
```

5 QB behind T1 , APC DAC output performs ramping down. The timing of performing this ramping down is defined as **QB_APCOFF** in **l1d_custom_rf.h**.

```
#define  QB_APCOFF     6
```

2.11.3.5 Set Transceiver in Idle Mode

23 QB behind T1, RF module finishes transmission data and MT6229 set it into idle mode. MT6229 Baseband activates control pins by changing BPI bus states, and also sends serial command to set idle mode command by BSI bus. To perform these operations, the timing and data of BPI and BSI bus can be implemented as follow steps:

(1) Modified the timing definition in **l1d_custom_rf.h**. **QB_ST3** is the timing definition of 4'th BSI command of TX window. **QB_PT3** is the timing definition of 4'th BPI bus status changing of TX window.

```
#define  QB_ST3            23
#define  QB_PT3            25
```

(2) Define the data of BPI bus states of each band. The state of BPI of each band is shown in the table:

BPI_BUS	9	8	7	6	5	4	3	2	1	0	Hex
Function	RFVCOEN	Not Used	EDGE Mode	Not Used	DCS BAND_SW	PAEN	Vc2	Vc1	Vc3	Vdd	
850	0	X	X	X	0	0	0	0	0	0	000h
GSM	0	X	X	X	0	0	0	0	0	0	000h
DCS	0	X	X	X	0	0	0	0	0	0	000h
PCS	0	X	X	X	0	0	0	0	0	0	000h

So the BPI data can be defined as follow:

```
#define  PDATA_GSM850_PT3    0x000
#define  PDATA_GSM_PT3       0x000
#define  PDATA_DCS_PT3       0x000
#define  PDATA_PCS_PT3       0x000
```

(3) Specify the command of BSI to be sent. Function **L1D_RF_SetSData_ST3** in **m12195.c** should be implemented to perform this action. The first statement in **L1D_RF_SetSData_ST3 is SETUP_ST3()** which is a macro to hide complicated code to initialize setting 3'rd BSI data. The second statement **HWRITE_1_SDATA** is specified the data to be sent by BSI bus. The parameters of **HWRITE_1_SDATA** are control word and data word of BSI data. The control word **SCTRL_IDLE** is composed of device selection (0) and data bit count (26 bits). The data word **SDATA_IDLE** is the command to set MT6139C into Idle mode. **SCTRL_IDLE** and **SDATA_IDLE** are defined in **l1d_rf.h.**

```
/*MT6140D*/ #define  SCTRL_IDLE                    SCTRL_WORD(0,26)
/*MT6140D*/ const unsigned long SDATA_SLEEP   = 0x0000002L; // Sleep mode
/*MT6140D*/ void  L1D_RF_SetSData_ST3( void )
/*MT6140D*/ {
/*MT6140D*/    SETUP_ST3();
/*MT6140D*/    HWRITE_1_SDATA( SCTRL_IDLE, SDATA_SLEEP );
/*MT6140D*/ }
```

The propagation delay when transmit data from Baseband uplink DAC to antenna should be claimed in file **l1d_custom_rf.c**. This delay depends on different RF module. For MT6140 RF module, the delay is 45 QB

```
const  short  TxPropagationDelay = 45;
```

2.11.3.6 Events Activation

If Layer 1 plans to transmit 148-bit data from TDMA timer count 2147 QB to 2739 QB, the above setting applies to create the TX window.

(1) At TDMA timer count is 1811 QB (336 QB ahead of TX SPORT turn on), the serial command of setting operation frequency is sent.

(2) At TDMA timer count is 1995 QB (152 QB ahead of TX SPORT turn on), TX DAC is enabled.

(3) At TDMA timer count is 2007 QB (140 QB ahead of TX SPORT turn on), serial command to set transmit mode is sent.

(4) At TDMA timer count is 2011 QB (136 QB ahead of TX SPORT turn on), BPI bus changes its states to TX mode.

(5) At TDMA timer count is 2048 QB (99 QB ahead of TX SPORT turn on), APC DAC output performs a DC offset to let ramp up more smoothly.

(6) At TDMA timer count is 2126 QB (21 QB ahead of TX SPORT turn on), APC DAC output performs ramping up.

(7) At TDMA timer count is 2139 QB (8 QB ahead of TX SPORT turn on), BPI bus changes its states to select operation right band.
(8) At TDMA timer count is 2147 QB, TX SPORT in MT6229 is turned on.
(9) At TDMA timer count is 2739 QB, TX SPORT in MT6229 is turned off.
(10) At TDMA timer count is 2744 QB (5 QB behind TX SPORT turning off), APC DAC output performs ramping down.
(11) At TDMA timer count is 2762 QB (23 QB behind TX SPORT turning off), BPI bus changes its states and send 3-wire command to set MT6139C into idle mode is sent.
(12) At TDMA timer count is 2769 QB (30 QB behind TX SPORT turn off), TX DAC is disabled at this time.

2.11.4 TX window modulation type setup

For the TX window, two type of the modulation, GMSK/EPSK can be selected according to the EDGE mode pin of the BPI bus. For the GMSK modulation, the pin should be low, and for the EPSK modulation, the pin should be high. The correlated settings could be found in file **l1d_custom_rf.c**. The settings should be correlated to the EDGE mode pin of the BPI bus.

```
#define  PDATA_GMSK              0x000
#define  PDATA_EPSK              0x080
```

The usage of these two definitions could be found in file **m12195.c**.

```
void  L1D_RF_SetPData_PT( void )
{  APBADDR      _reg;
   const short *_d16p;

   _reg  = PDATA_REG_TABLE[ l1d_rf.cwin_idx ];
   _d16p = PDATA_TABLE[ l1d_rf.band ][ RF_TX ];
   if ( (l1d_rf.tx_mod_type) & (1<<l1d_rf.modidx) )
   {
      HW_WRITE( _reg, ((*_d16p)|(PDATA_8PSK)) );   _reg+=2;   _d16p++;
      HW_WRITE( _reg, ((*_d16p)|(PDATA_8PSK)) );   _reg+=2;   _d16p++;
      HW_WRITE( _reg, ((*_d16p)|(PDATA_8PSK)) );   _reg+=2;   _d16p++;
      HW_WRITE( _reg, (*_d16p) );//PT3 is end, doesn't need to | PDATA_8PSK
      /* PT3A */
      _d16p++;
      _reg = PDATA_REG_TABLE2[ l1d_rf.cwin_idx ];
      HW_WRITE( _reg, (*_d16p) );
   }
   else
   {
      HW_WRITE( _reg, ((*_d16p)|(PDATA_GMSK)) );   _reg+=2;   _d16p++;
      HW_WRITE( _reg, ((*_d16p)|(PDATA_GMSK)) );   _reg+=2;   _d16p++;
      HW_WRITE( _reg, ((*_d16p)|(PDATA_GMSK)) );   _reg+=2;   _d16p++;
      HW_WRITE( _reg, (*_d16p) );//PT3 is end, doesn't need to | PDATA_GMSK
      /* PT3A */
      _d16p++;
      _reg = PDATA_REG_TABLE2[ l1d_rf.cwin_idx ];
      HW_WRITE( _reg, (*_d16p) );
   }
}
```

The function **L1D_RF_SetPData_PT** is used to setup the BPI event data of the PT1, PT2, PT2B and PT3. Except the PT3, BPI event data setup in file **l1d_custom_rf.c** should append the PDATA_GMSK or the PDATA_8PSK according to the modulation type of the TX window.

As for the interslot modulation changing of the TX window, the BPI event named PT2M is used. The changing from the GMSK to the EPSK, the EPSK mode pin should be changed from the high status to the low status, as the result, the following definitions in file **l1d_custom_rf.c** according to the operating band is defined.

```
#define  PDATA_GSM850_PT2M1_G8   ( 0x08B | PDATA_RFDOO )
#define  PDATA_GSM850_PT2M2_G8   ( 0x09B | PDATA_RFDOO )
#define  PDATA_GSM850_PT2M3_G8   ( 0x09B | PDATA_RFDOO )
#define  PDATA_GSM_PT2M1_G8      ( 0x08B | PDATA_RFDOO )
#define  PDATA_GSM_PT2M2_G8      ( 0x09B | PDATA_RFDOO )
#define  PDATA_GSM_PT2M3_G8      ( 0x09B | PDATA_RFDOO )
#define  PDATA_DCS_PT2M1_G8      ( 0x0A3 | PDATA_RFDOO )
#define  PDATA_DCS_PT2M2_G8      ( 0x0B3 | PDATA_RFDOO )
#define  PDATA_DCS_PT2M3_G8      ( 0x0B3 | PDATA_RFDOO )
#define  PDATA_PCS_PT2M1_G8      ( 0x0A3 | PDATA_RFDOO )
#define  PDATA_PCS_PT2M2_G8      ( 0x0B3 | PDATA_RFDOO )
#define  PDATA_PCS_PT2M3_G8      ( 0x0B3 | PDATA_RFDOO )
```

Note that the team **G8** is used to identify the changing from the GMSK to the EPSK (8PSK). In fact, the definition is decided by the **PDATA_PT2B | PDATA_8PSK**.

On the other hand, the changing from the EPSK to the GPSK, the EPSK mode pin should be changed from the low status to the high status, as the result, the following definitions in file **l1d_custom_rf.c** according to the operating band is defined.

```
#define  PDATA_GSM850_PT2M1_8G   ( 0x0DB | PDATA_RFDOO )
#define  PDATA_GSM850_PT2M2_8G   ( 0x05B | PDATA_RFDOO )
#define  PDATA_GSM850_PT2M3_8G   ( 0x05B | PDATA_RFDOO )
#define  PDATA_GSM_PT2M1_8G      ( 0x0DB | PDATA_RFDOO )
#define  PDATA_GSM_PT2M2_8G      ( 0x05B | PDATA_RFDOO )
#define  PDATA_GSM_PT2M3_8G      ( 0x05B | PDATA_RFDOO )
#define  PDATA_DCS_PT2M1_8G      ( 0x0F3 | PDATA_RFDOO )
#define  PDATA_DCS_PT2M2_8G      ( 0x073 | PDATA_RFDOO )
#define  PDATA_DCS_PT2M3_8G      ( 0x073 | PDATA_RFDOO )
#define  PDATA_PCS_PT2M1_8G      ( 0x0F3 | PDATA_RFDOO )
#define  PDATA_PCS_PT2M2_8G      ( 0x073 | PDATA_RFDOO )
#define  PDATA_PCS_PT2M3_8G      ( 0x073 | PDATA_RFDOO )
```

Note that the team **8G** is used to identify the changing from the EPSK to the GPSK. In fact, the definition is decided by the **PDATA_PT2B | PDATA_GPSK**.

For the event timing of the PT2M could be found in file **l1d_custom_rf.c**. 0qB ahead of the SPORT ON of the 2nd TX window, the modulation type must be changed. The definitions are as followings.

```
#define  QB_PT2M1_G8                  4
#define  QB_PT2M2_G8                  3
#define  QB_PT2M3_G8                  2
#define  QB_PT2M1_8G                 11
#define  QB_PT2M2_8G                 10
#define  QB_PT2M3_8G                  9
```

2.12 RF Control Configuration of AD6546 RF Module

For example, the schematic circuit interface between AD6546 RF module and MT6235 chip is shown in below diagram. Please refer to MediaTek AD6546 RF Module schematic circuit to get more detail information.

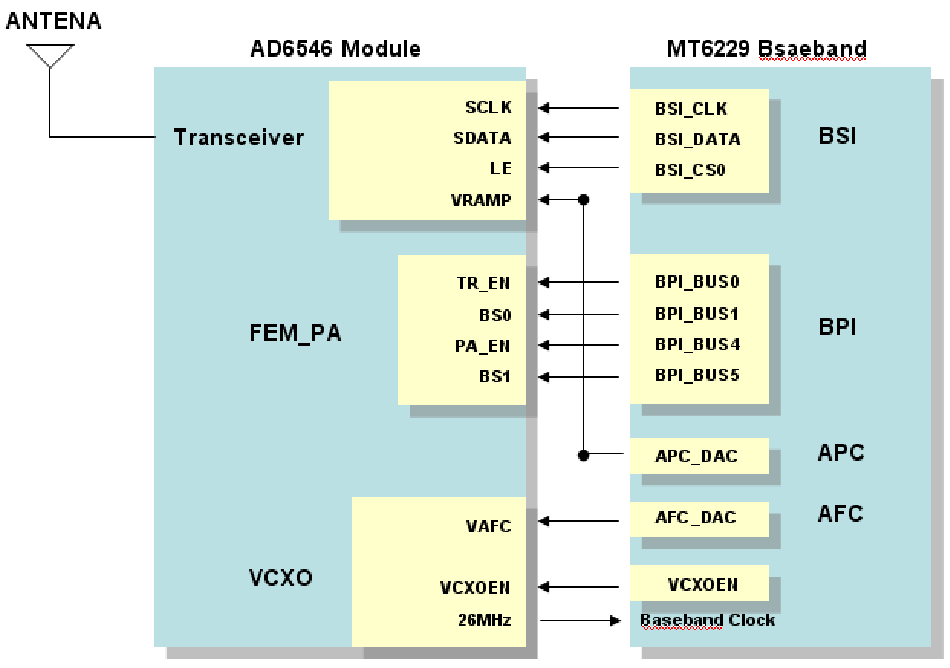


Figure 19 The connection of MT6235 and AD6546 RF Module

BSI_CLK, BSI_DATA, BSI_CS0 pins of MT6235 are directly connected to the serial interface of AD6546 transceiver. APC_DAC pin of MT6235 is connected to the VRAM pin of AD6546 of RF module board. VCXOEN of MT6235 controls the enable state of VCXO. AFC DAC output is connected to the voltage control pin VAFC of VCXO. The generated clock of VCXO provides MT6235 system clock. Band-related pins TR_EN, BS0, BS1, PA_EN are controlled by the bit 0,1,4,5 of BPI bus of MT6235. The high/low state of each pin of BPI bus and their timing of changing state can be programmed by MT6235. The RF radio control method is introduced in the latter description.

2.12.1 Initialization

RF module can be controlled by the BSI, BPI, APC, AFC units of MT6235 chips. BSI, BPI, APC, AFC units can work in 2 modes. One mode is immediate mode, and the other is TDMA event control mode.

Immediate control mode is that the output signal of BSI, BPI, APC, or AFC unit is generated as soon as setting the corresponding registers in MT6235. When turning on the RF module, the immediate mode can be used to initialize the RF module. For example, immediate control mode is used when initializing AD6546 RF module, and the initial states are set as follow:

(1) TR_EN (BPI_BUS0) is low state
(2) BS0 (BPI_BUS1) is low state
(3) PA_EN (BPI_BUS4) is low state

(4) BS1 (BPI_BUS5) is low state
(5) Serial commands (0x00000000, 0x00000184, 0x00000105, 0x00291686, 0x00081F9E, 0x0003901f, 0x00006384, 0x00022F80, 0x00002401, 0x00000184) are sending to AD6546 transceiver.

The initialization of BPI, BSI data can be implemented in **L1D_RF_PowerOn** in **m12196.c** by using immediate mode.

```
/*AD6546*/ #define INITIAL_REG0_RESET              0x00000000  /*Performs chip reset */
/*AD6546*/ #define INITIAL_REG0_NORST              0x00000080  /*No Reset performed */
/*AD6546*/ #define INITIAL_REG1                    0x00000001|(XO_CapID<<7)
/*AD6546*/ #define INITIAL_REG5                    0x00000105
/*AD6546*/ #define INITIAL_REG6                    0x00291686
/*AD6546*/ #define INITIAL_REG30                   0x00081F9E
/*AD6546*/ #define INITIAL_REG31                   0x0003901f
/*AD6546*/ #define PROG_POWER_CAL                  0x00006384
/*AD6546*/ #define PROG_POWER_ON                   0x00000184
/*AD6546*/ #define PROG_POWER_OFF                  0x00000004
/*AD6546*/ #define PROG_CAL_BAND         0x00002F80|(HIGH_BAND_PATH<<16)|((1-LOW_BAND_PATH)<<17)
/*AD6546*/
/*AD6546*/ void  L1D_RF_PowerOn( void )
/*AD6546*/ {
/*AD6546*/    IMM_MODE_BEGIN( IMMMASK_ALL );
/*AD6546*/    IMM_SEND_BPI( PDATA_INIT );
/*AD6546*/
/*AD6546*/ #ifndef L1D_SIM
/*AD6546*/    if(l1d_rf.is_init)
/*AD6546*/    {
/*AD6546*/       IMM_SEND_BSI( SCTRL_IMOD(0,24), INITIAL_REG0_RESET );/*Performs chip reset */
/*AD6546*/       IMM_SEND_BSI( SCTRL_IMOD(0,24), PROG_POWER_ON );    /*Power on LDO1 & LDO2 */
/*AD6546*/       IMM_SEND_BSI( SCTRL_IMOD(0,24), INITIAL_REG5 );              /*BB gain 3dB */
/*AD6546*/       IMM_SEND_BSI( SCTRL_IMOD(0,24), INITIAL_REG6 ); /*Initial PLL Word ARFCN 62*/
/*AD6546*/       /*Test Register,initialize only*/
/*AD6546*/       IMM_SEND_BSI( SCTRL_IMOD(0,24), INITIAL_REG30 );
/*AD6546*/       IMM_SEND_BSI( SCTRL_IMOD(0,24), INITIAL_REG31 );
/*AD6546*/       WAIT_TIME_QB(27);                      /* Wait 25 us after LDO1 and LDO2 on */
/*AD6546*/       IMM_SEND_BSI( SCTRL_IMOD(0,24), PROG_POWER_CAL );  /*Start auto calibration*/
/*AD6546*/       WAIT_TIME_QB(27);                   /* Wait 25+25 us after LDO1 and LDO2 on */
/*AD6546*/       IMM_SEND_BSI( SCTRL_IMOD(0,24), PROG_CAL_BAND );
/*AD6546*/       WAIT_TIME_QB(4660);          /* Wait 4.3ms for RF internal calibration done */
/*AD6546*/     }
/*AD6546*/ #endif
/*AD6546*/       IMM_SEND_BSI( SCTRL_IMOD(0,24), INITIAL_REG1 );                  /*AFC_CAP */
/*AD6546*/       IMM_SEND_BSI( SCTRL_IMOD(0,24), PROG_POWER_ON );
/*AD6546*/       IMM_MODE_END(  );                               /*Turn Immediate Mode off*/
/*AD6546*/  }
/*AD6546*/
```

2.12.2 Create RX window

TDMA timer is a counter unit that counts 1 per quarter-bit (0.923u). It always repeats counting from 0 to 4999. So the period of TDMA counter is 5000 QB (1 TDMA frame). If BSI, BPI, AFC, APC are programmed in TDMA event driven mode, the active time can be programmed. When the TDMA counter counts to the programmed time(count), the corresponding unit is triggered to activate.

If it is needed to receive data from air or transmit data to air, Layer1 needs to plan all RF setting and timing to create RX/TX window, control receiving gain, control VCXO, control PA…etc. For this purpose, TDMA event driven mode of BSI, BPI, APC, or AFC units should be used.

For example of creating RX window to receive data, the RF setting and timing of AD6546 RF module can be depicted in the below diagram:

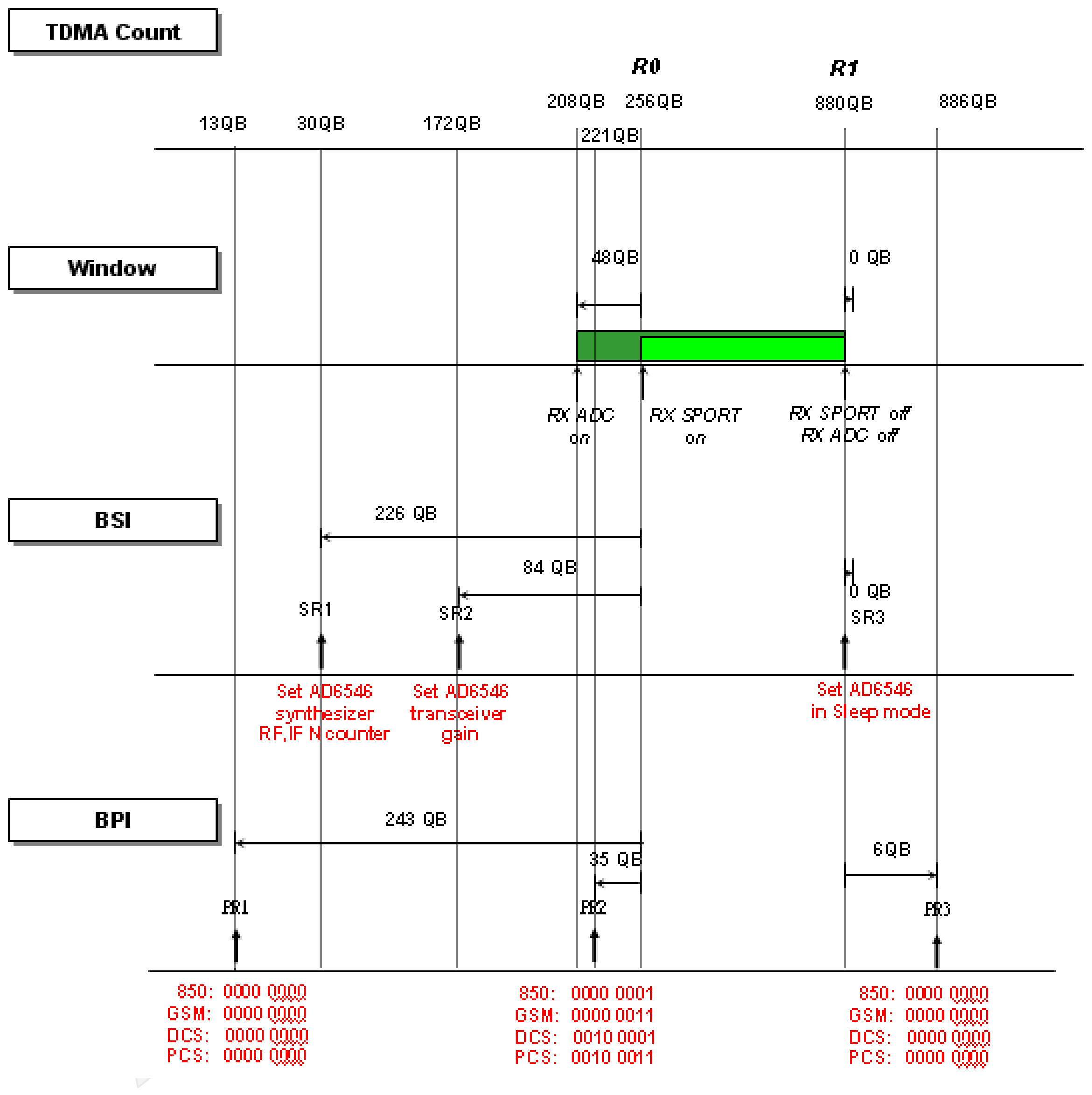


Figure 20 AD6546 Event's timing of RX window

The TDMA timing and data setting are described in 2 parts. One part is that before turning on RX SPORT to receive I/Q data from AD6546 transceiver. And the other part is after turning off RX SPORT to finish receiving I/Q data. To describe these two parts easily, the timing of turning on RX SPORT is taken as one base named **R0**. And the timing of turning off RX SPORT is taken as one base named **R1**. In this case, R0 is at TDMA counter 256 QB and R1 is at TDMA counter 880 QB.

2.12.2.1 Set Synthesizer N-Counter

243 QB ahead of R0, RF module should start to operate. MT6235 chip controls the RF module by activating control pins and sending serial command. MT6235 activates the control pins by BPI bus status. The high/low state of BPI bus at this time depends on operation band. After 17QB, MT6235 also sends serial command to lock

AD6546 synthesizer to lock the operation frequency by BSI bus. To perform these operations, the timing and data of BPI and BSI bus can be implemented as follow steps:

(1) Modified the timing definition in **l1d_custom_rf.h**. **QB_SR1** is the timing definition of 1'st BSI command of RX window. **QB_PR1** is the timing definition of 1'st BPI bus status changing of RX window.

```
#define  QB_SR1                     226
#define  QB_PR1                     243
```

(2) Define the states of BPI bus for each band. The data is shown in the below table:

BPI_BUS	5	4	3	2	1	0	Hex
Function	BS1	PA_EN	reserved	reserved	BS0	TR_EN	
850	0	0	X	X	0	0	00h
GSM	0	0	X	X	0	0	00h
DCS	0	0	X	X	0	0	00h
PCS	0	0	X	X	0	0	00h

So the BPI data can be defined as follow:

```
#define  PDATA_GSM850_PR1         0x00
#define  PDATA_GSM_PR1            0x00
#define  PDATA_DCS_PR1            0x00
#define  PDATA_PCS_PR1            0x00
```

(3) Specify the command of BSI to be sent. Function **L1D_RF_SetSData_SR1** in **m12195.c** be implemented to perform this action. **SETUP_SR1()** is the first statement in **L1D_RF_SetSData_SR1** which is a macro to hide complicated code to initialize setting 1'st BSI data. The second statement **HWRITE_2_SDATA** is specified the data to be sent by BSI bus. Two parameters of **HWRITE_2_SDATA** are control word and data word of BSI data. The control words **SCTRL_WARM** and **SCTRL_PLL** are composed of device selection (0) and data bit count (24 bits). The 1st data word **SDATA_WARM_BSEL[l1d_rf.band]** is the command to set AD6546 into Warm-up mode which is corresponding with the band. The 2nd data word **l1d_rf.RFN_data**, RF N-Counter setting value, is the evaluation result of **L1D_RF_GetRxPLLSetting** called by Layer 1.

```
/*AD6546*/ #define  SCTRL_WARM              SCTRL_WORD(0,24)
/*AD6546*/ #define  SCTRL_PLL               SCTRL_WORD(0,24)
/*AD6546*/ void  L1D_RF_SetSData_SR1( void )
/*AD6546*/ {
/*AD6546*/    SETUP_SR1();
/*AD6546*/    HWRITE_2_SDATA( SCTRL_WARM,    SDATA_WARM_BSEL[l1d_rf.band],
/*AD6546*/                    SCTRL_PLL,     l1d_rf.RFN_data);
/*AD6546*/ }
```

To create the receiving window with right operation frequency, evaluating the setting value of operation frequency is needed. Function **L1D_RF_GetRxPLLSetting** in file **m12191.c** needs to be implemented to evaluate the synthesizer setting. And Layer 1 calls this function when programs the RF synthesizer setting value and saves the result into variable **l1d_rf.RFN_data**. This content of this variable is the command to be sent to transceiver chip to lock the operation frequency. The evaluation of AD6546 synthesizer N counter setting is as the following code.

```
/*AD6546*/ void  L1D_RF_GetRxPLLSetting( int rf_band, int arfcn, long *rfN, long *ifN )
/*AD6546*/ {
/*AD6546*/    switch(rf_band)
/*AD6546*/    {
/*AD6546*/       case  FrequencyBand850 :
/*AD6546*/       {
/*AD6546*/          if(arfcn<=201)
/*AD6546*/          {   if(arfcn<=158)
/*AD6546*/              {                                /* ARFCN : 128~158      */
/*AD6546*/                  *rfN = (((arfcn-128)*24 + 304)<<13) | 0x1206L /*(36L<<7|0x06L)*/;
/*AD6546*/              }
/*AD6546*/              else
/*AD6546*/              {                              /* ARFCN : 159~201      */
/*AD6546*/                  *rfN = (((arfcn-159)*24 + 8)<<13) | 0x1286L /*(37L<<7|0x06L)*/;
/*AD6546*/              }
/*AD6546*/          }
/*AD6546*/          else
/*AD6546*/          {   if(arfcn<=245)
/*AD6546*/              {                                /* ARFCN : 202~245      */
/*AD6546*/                  *rfN = (((arfcn-202)*24 )<<13) | 0x1306L /*(38L<<7|0x06L)*/;
/*AD6546*/              }
/*AD6546*/              else
/*AD6546*/              {                              /* ARFCN : 246~251      */
/*AD6546*/                  *rfN = (((arfcn-246)*24+16)<<13) | 0x1386L /*(39L<<7|0x06L)*/;
/*AD6546*/              }
/*AD6546*/          }
/*AD6546*/          break;
/*AD6546*/       }
/*AD6546*/       case  FrequencyBand900 :
/*AD6546*/       {   :
/*AD6546*/          :
/*AD6546*/          break;
/*AD6546*/       }
/*AD6546*/       case  FrequencyBand1800 :
/*AD6546*/       {   :
/*AD6546*/          :
/*AD6546*/          break;
/*AD6546*/       }
/*AD6546*/       case  FrequencyBand1900 :
/*AD6546*/       {   :
/*AD6546*/          :
/*AD6546*/          break;
/*AD6546*/       }
/*AD6546*/       default :
/*AD6546*/       {
/*AD6546*/          break;
/*AD6546*/       }
/*AD6546*/    }
/*AD6546*/    *ifN = 0;
/*AD6546*/ }
```

2.12.2.2 Set Transceiver Gain

84 QB ahead of R0, Band selection and Rx gain setting should be performed. MT6235 activates the control pins by BPI bus status, and also sends serial command to set transceiver gain by BSI bus. To perform these operations, the timing and data of BPI and BSI bus can be implemented as follow steps:

(1) Modified the timing definition in **l1d_custom_rf.h**. **QB_SR2** is the timing definition of 2'nd BSI command of RX window. **QB_PR2** is the timing definition of 2'nd BPI bus status changing of RX window.

```
#define  QB_SR2                        84
#define  QB_PR2                        35
```

To support multi-slots receiving in GPRS mode, different LNA gain setting for each time slot is allowed. The TDMA timing of LNA gain setting at inter slot shall be defined as

```
          #define   QB_SR2M                         36
```

This timing QB_SR2M is ahead the R0 of the next slot.

(2) Define the states of BPI bus for each band. The data is shown in the below table:

BPI_BUS	5	4	3	2	1	0	Hex
Function	BS1	PA_EN	reserved	reserved	BS0	TR_EN	
850	0	0	X	X	0	1	01h
GSM	0	0	X	X	1	1	03h
DCS	1	0	X	X	0	1	21h
PCS	1	0	X	X	1	1	23h

So the BPI data can be defined as follow:

```
          #define   PDATA_GSM850_PR2          0x01
          #define   PDATA_GSM_PR2             0x03
          #define   PDATA_DCS_PR2             0x21
          #define   PDATA_PCS_PR2             0x23
```

(3) Specify the command of BSI to be sent. Function **L1D_RF_SetSData_SR2** in **m12195.c** should be implemented to perform this action. The first statement in **L1D_RF_SetSData_SR2** is **SETUP_SR2()** which is a macro to hide complicated code to initialize setting 2'nd BSI data. The second statement **HWRITE_2_SDATA** is specified the data to be sent by BSI bus. The parameters of **HWRITE_2_SDATA** are control word and data word of BSI data. If this is the first slot of continuous RX slot or this is the single slot, the control words **SCTRL_AGC** and **SCTRL_RXCON** which are composed of device selection (0) and data bit count (24 bits) and two kinds of BSI data are sent. The 1st command **l1d_rf.AGC_data**, transceiver gain setting value, is the evaluation result of **L1D_RF_GetGainSetting** called by Layer 1. The data word **SDATA_WARM_BSEL[l1d_rf.band]| SDATA_RXON_MASK** is command to set AD6546 into RX mode. If this is not the first slot of continuous RX slot, the control word **SCTRL_AGC** and command **l1d_rf.AGC_data** are sent.

```
/*AD6546*/ #define  SCTRL_AGC               SCTRL_WORD(0,24)
/*AD6546*/ #define  SCTRL_RXON              SCTRL_WORD(0,24)
/*AD6546*/ #define  SDATA_RXON_MASK         0x000200L
/*AD6546*/ void  L1D_RF_SetSData_SR2( void )
/*AD6546*/ {
/*AD6546*/    SETUP_SR2();
/*AD6546*/
/*AD6546*/    if(IS_CONTINUOUS_RX_SLOT())
/*AD6546*/    {
/*AD6546*/      HWRITE_2_SDATA( SCTRL_AGC,    l1d_rf.AGC_data,
/*AD6546*/                      SCTRL_RESERVED,    0 );
/*AD6546*/    }
/*AD6546*/    else
/*AD6546*/    {
/*AD6546*/      HWRITE_2_SDATA( SCTRL_AGC,    l1d_rf.AGC_data,
/*AD6546*/                      SCTRL_RXON,    SDATA_WARM_BSEL[l1d_rf.band]|SDATA_RXON_MASK );
/*AD6546*/    }
/*AD6546*/ }
```

To create the receiving window with suitable gain of receiving amplifier, evaluating the setting value of request gain is needed. Function **L1D_RF_GetGainSetting** in file **m12192.c** needs to be implemented to evaluate the transceiver gain setting. And Layer 1 calls this function when programs the gain setting value and saves the result into variable **l1d_rf.AGC_data**. This content of this variable is the command sent to transceiver chip is to set the requested receive amplifier gain. The evaluation of AD6546 receiving amplifier gain setting is as the following code:

```
int   L1D_RF_GetGainSetting( int rf_band, int arfcn, int request_gain, long *gain_setting )
/*AD6546*/ {
/*AD6546*/   int   max_gain, min_gain, real_gain;
/*AD6546*/   const int*    d32ptr;
/*AD6546*/   const sL1DAGCDATA*  agcptr;
/*AD6546*/   long  setting;
/*AD6546*/   int   bit_no;
/*AD6546*/
/*AD6546*/   /* evaluate the range of available gain */
/*AD6546*/   d32ptr  = (int*)&(GAIN_RANGE_TABLE[rf_band]);
/*AD6546*/   max_gain = *d32ptr++;
/*AD6546*/   min_gain = *d32ptr;
/*AD6546*/
/*AD6546*/   /* clipping the request gain to the avialable gain */
/*AD6546*/   if( request_gain>=max_gain )
/*AD6546*/   {  request_gain = max_gain;  }
/*AD6546*/   else  if( request_gain<=min_gain )
/*AD6546*/   {  request_gain = min_gain;  }
/*AD6546*/
/*AD6546*/   /* evaluate the real gain setting */
/*AD6546*/   agcptr = AGC_TABLE[rf_band];
/*AD6546*/   if( request_gain < agcptr->pos_gain )
/*AD6546*/   {  agcptr++;
/*AD6546*/   }
/*AD6546*/   {  bit_no    = BIT_NO( request_gain, agcptr->A, GC_B );
/*AD6546*/      real_gain = REAL_GAIN( bit_no, agcptr->A, GC_B );
/*AD6546*/      setting   = agcptr->setting | (bit_no<<7);
/*AD6546*/   }
/*AD6546*/
/*AD6546*/   *gain_setting = setting;
/*AD6546*/   return( real_gain );
/*AD6546*/ }
```

2.12.2.3 Set Transceiver in idle Mode

6 QB behind R1, RF module finishes receiving data and MT6235 set into idle mode. MT6235 Baseband activates control pins by changing BPI bus states, and also sends serial command to set idle mode command by BSI bus. To perform these operations, the timing and data of BPI and BSI bus can be implemented as follow steps:

(1) Modified the timing definition in **l1d_custom_rf.h**. **QB_SR3** is the timing definition of 3'rd BSI command of RX window. **QB_PR3** is the timing definition of 3'rd BPI bus status changing of RX window.

```
#define  QB_SR3                         0
#define  QB_PR3                         6
```

(2) Define the data of BPI bus states for each band. The data is shown in the below table:

BPI_BUS	5	4	3	2	1	0	Hex
Function	BS1	PA_EN	reserved	reserved	BS0	TR_EN	
850	0	0	X	X	0	0	00h
GSM	0	0	X	X	0	0	00h
DCS	0	0	X	X	0	0	00h
PCS	0	0	X	X	0	0	00h

So the BPI data can be defined as follow:

```
        #define  PDATA_GSM850_PR3         0x00
        #define  PDATA_GSM_PR3            0x00
        #define  PDATA_DCS_PR3            0x00
        #define  PDATA_PCS_PR3            0x00
```

(3) Specify the command of BSI to be sent. Function **L1D_RF_SetSData_SR3** in **m12195.c** should be implemented to perform this action. The first statement in **L1D_RF_SetSData_SR3** is **SETUP_SR3()** which is a macro to hide complicated code to initialize setting 3'rd BSI data. The second statement HWRITE_1_SDATA is specified the data to be sent by BSI bus. The parameters of **HWRITE_1_SDATA** are control word and data word of BSI data. The control word **SCTRL_IDLE** is composed of device selection (0) and data bit count (18 bits). The data word **SDATA_IDLE** is the command to set AD6546 into idle mode. **SCTRL_IDLE** and **SDATA_IDLE** are defined in **l1d_rf.h**.

```
/*AD6546*/ #define  SCTRL_IDLE                        SCTRL_WORD(0,18)
/*AD6546*/ #define  SDATA_IDLE                        0x000184
/*AD6546*/ void  L1D_RF_SetSData_SR3( void )
/*AD6546*/ {
/*AD6546*/   SETUP_SR3();
/*AD6546*/   HWRITE_1_SDATA( SCTRL_IDLE, SDATA_IDLE );
/*AD6546*/ }
```

2.12.2.4 RX ADC CONTROL

Besides setting the timing and data of BPI and BSI bus of MT6235 Baseband to create a RX window, the timing setting for RX frame enable and frame synchronization control of MT6235 are needed. RX frame enable is used to enable or disable RX DAC. RX frame synchronization is used to turn on/off RX SPORT. The timing of enabling TX DAC to turning on RX SPORT named **QB_RX_FENA_2_FSYNC** and the timing of turning off RX SPORT to disabling RX DAC named **QB_RX_FSYNC_2_FENA** are both also needed to be defined in **l1d_custom_rf.h**. In this case, the timing of enabling RX DAC to turning on RX SPORT is recommend to be 48 QB and the timing of turning off RX SPORT to disabling TX DAC is 0 QB. So **QB_RX_FENA_2_FSYNC** and **QB_RX_FSYNC_2_FENA** can be defined as:

```
     #define  QB_RX_FENA_2_FSYNC                 48
     #define  QB_RX_FSYNC_2_FENA                 0
```

2.12.2.5 Event Activation

If Layer1 plans to receive 156-bit data from TDMA timer count 256 QB to 880 QB, the above setting applies to create the RX window.

(1) At TDMA timer count is 13QB (243QB ahead of RX SPORT turn on), BPI bus changes its states to 00h (GSM850/GSM/DCS/PCS). And after 17 QB (226 QB), the serial command of setting operation frequency is sent to AD6546 transceiver.

(2) At TDMA timer count is 172 QB (84 QB ahead of RX SPORT turn on), the serial command of setting RX gain is sent to AD6546 transceiver.

(3) At TDMA timer count is 208 QB (48 QB ahead of RX SPORT turn on), RX ADC in MT6235 is enable.

(4) At TDMA timer count is 221 QB (35 QB ahead of RX SPORT turn on), BPI bus changes its states to 01h(GSM850) or 03h(GSM) or 21h(DCS) or 23h(PCS).

(5) At TDMA timer count is 256 QB, RX SPORT in MT6235 is turned on. The I/Q data begin to be sampled.

(6) At TDMA timer count is 880 QB, RX SPORT in MT6235 is turned off and RX ADC is also disabled. The I/Q data stop sampling.

(7) At TDMA timer count is 880 QB (at RX SPORT turning off), the command of setting AD6546 into idle mode is sent.

(8) At TDMA timer count is 886 QB(6 QB behind RX SPORT turning off), BPI bus changes its states to 00h.

2.12.3 Create TX window

Another example is creating TX window to transmit data. The RF setting and timing of AD6546 RF module can be depicted in the below diagram:

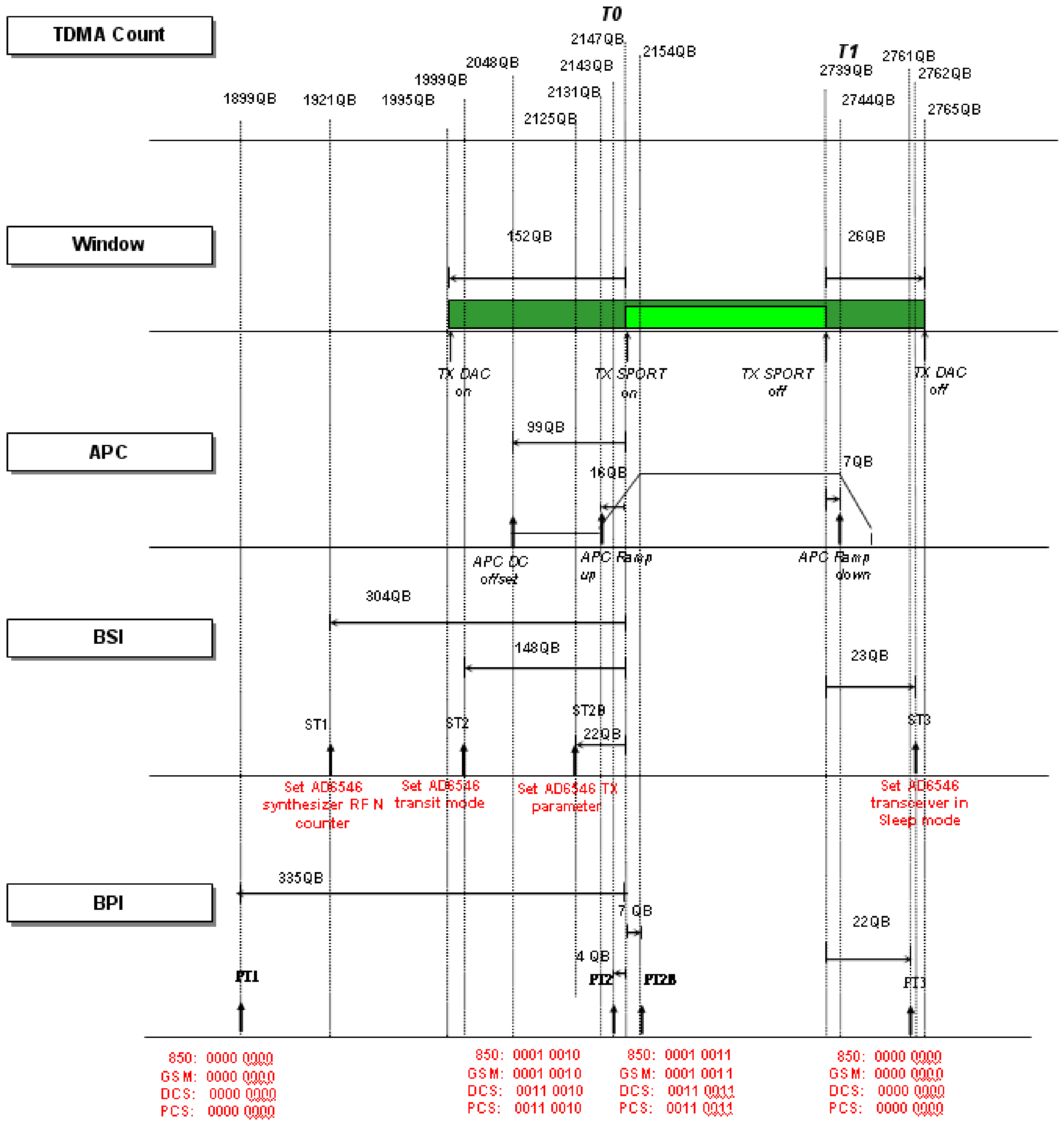


Figure 21 AD6546 Event's timing of TX window

The TDMA timing and data setting are described in 2 parts. One part is that before turning on TX SPORT to receive I/Q data from AD6546 transceiver. And the other part is after turning off TX SPORT to finish receiving I/Q data. To describe these two parts easily, the timing of turning on TX SPORT is taken as one base named **T0**. And the timing of turning off TX SPORT is taken as one base named **T1**. In this case, T0 is at TDMA counter 2147 QB and T1 is at TDMA counter 2739 QB.

2.12.3.1 Set synthesizer N-Counter

248 QB ahead of R0, RF module start to work. MT6235 Baseband activates control pins by changing BPI bus states. Then MT6235 sends serial command to lock AD6546 synthesizer to lock the operation frequency by BSI bus. To perform this operation, the timing and data of BSI bus can be implemented as follow steps:

(1) Modified the timing definition in **l1d_custom_rf.h**. **QB_ST1** is the timing definition of 1'st BSI command of TX window. **QB_PT1** is the timing definition of 1'st BPI bus status changing of TX window.

```
#define  QB_PT1                       248
#define  QB_ST1                       226
```

(2) Define the states of BPI bus for each band. The data is shown in the below table:

BPI_BUS	5	4	3	2	1	0	Hex
Function	BS1	PA_EN	reserved	reserved	BS0	TR_EN	
850	0	0	X	X	0	0	00h
GSM	0	0	X	X	0	0	00h
DCS	0	0	X	X	0	0	00h
PCS	0	0	X	X	0	0	00h

So the BPI data can be defined as follow:

```
#define  PDATA_GSM850_PT1         0x00
#define  PDATA_GSM_PT1            0x00
#define  PDATA_DCS_PT1            0x00
#define  PDATA_PCS_PT1            0x00
```

(3) Specify the command of BSI to be sent. Function **L1D_RF_SetSData_ST1** in **m12195.c** be implemented to perform this action. **SETUP_ST1()** is the first statement in **L1D_RF_SetSData_ST1** which is a macro to hide complicated code to initialize setting 1'st BSI data. Two parameters of **HWRITE_2_SDATA** are control word and data word of BSI data. The control words SCTRL_WARM and SCTRL_PLL are composed of device selection (0) and data bit count (24 bits). The 1st data word **SDATA_WARM_BSEL[l1d_rf.band]|SDATA_PLLTX_MASK** is the command to set AD6546 into Warm-up mode which is corresponding with the band and indicate the TX PLL word. The 2nd data word **l1d_rf.RFN_data**, RF N-Counter setting value, is the evaluation result of **L1D_RF_GetTxPLLSetting** called by Layer 1. The global variable **bfetxset.field.iqswap** is used to set I/Q swap for low band.

```
/*AD6546*/ #define  SCTRL_WARM              SCTRL_WORD(0,24)
/*AD6546*/ #define  SCTRL_PLL               SCTRL_WORD(0,24)
/*AD6546*/ #define  SDATA_PLLTX_MASK        0x008000L
/*AD6546*/const unsigned long  SDATA_WARM_BSEL[5]=
/*AD6546*/ { /*Warm*/ /*Band sel*/
/*AD6546*/   0x000184,             /*GSM 450 no support*/
/*AD6546*/   0x006184 | 0x001800,    /*GSM 850*/
/*AD6546*/   0x006184 | 0x000000,    /*GSM 900*/
/*AD6546*/   0x006184 | 0x000800,    /*DCS 1800*/
/*AD6546*/   0x006184 | 0x001000,    /*PCS 1900*/
/*AD6546*/ };
/*AD6546*/ void  L1D_RF_SetSData_ST1( void )
/*AD6546*/ {
/*AD6546*/   SETUP_ST1();
/*AD6546*/   HWRITE_2_SDATA( SCTRL_WARM,    SDATA_WARM_BSEL[l1d_rf.band]|SDATA_PLLTX_MASK,
/*AD6546*/                   SCTRL_PLL,     l1d_rf.RFN_data);
/*AD6546*/
/*AD6546*/   if( l1d_rf.d2c_txiqswap==0 )
/*AD6546*/   {     BFETXSET  bfetxset;
/*AD6546*/         HW_READ_BFETXSET( bfetxset );
/*AD6546*/         bfetxset.field.iqswap = (l1d_rf.band >2 ? 0 : 1)& 0x1;
/*AD6546*/         HW_WRITE_BFETXSET( bfetxset );
/*AD6546*/   }
/*AD6546*/ }
```

2.12.3.2 Set Transceiver in Transmit Mode

148 QB ahead of R0, MT6235 Baseband sends serial command to set transmit mode by BSI bus and after 144 QB, MT6235 Baseband activates control pins by changing BPI bus status. To perform these operations, the timing and data of BPI and BSI bus can be implemented as follow steps:

(1) Modified the timing definition in **l1d_custom_rf.h**. **QB_ST2** is the timing definition of 2'nd BSI command of TX window. **QB_PT2** is the timing definition of 2'nd BPI bus status changing of TX window.

```
#define  QB_ST2                            148
#define  QB_PT2                            4
```

(2) Define the states of BPI bus for each band. The data is shown in the below table:

BPI_BUS	5	4	3	2	1	0	Hex
Function	BS1	PA_EN	reserved	reserved	BS0	TR_EN	
850	0	1	X	X	1	0	12h
GSM	0	1	X	X	1	0	12h
DCS	1	1	X	X	1	0	32h
PCS	1	1	X	X	1	0	32h

So the BPI data can be defined as follow:

```
#define  PDATA_GSM850_PT2        0x12
#define  PDATA_GSM_PT2           0x12
#define  PDATA_DCS_PT2           0x32
#define  PDATA_PCS_PT2           0x32
```

(3) Specify the command of BSI to be sent. Function **L1D_RF_SetSData_ST2** in **m12195.c** should be implemented to perform this action. The first statement in **L1D_RF_SetSData_ST2** is **SETUP_ST2()** which is a macro to hide complicated code to initialize setting 2'nd BSI data. The second statement **HWRITE_2_SDATA** is specified the data to be sent by BSI bus. The parameters of **HWRITE_2_SDATA** are control word and data word of BSI data. The control word **SCTRL_TXON** is composed of device selection (0) and data bit count (24 bits). The data word **SDATA_WARM_BSEL[l1d_rf.band]|SDATA_TXON_MASK** is command to set AD6546 into TX mode.

```
/*AD6546*/ #define  SCTRL_TXON               SCTRL_WORD(0,24)
/*AD6546*/ #define  SDATA_TXON_MASK          0x008400L
/*AD6546*/ void  L1D_RF_SetSData_ST2( void )
/*AD6546*/ {
/*AD6546*/   SETUP_ST2();
/*AD6546*/   HWRITE_2_SDATA( SCTRL_TXON,     SDATA_WARM_BSEL[l1d_rf.band]|SDATA_TXON_MASK,
/*AD6546*/                   SCTRL_RESERVED,    0 );
/*AD6546*/ }
```

(4) 7 QB behind T0, MT6235 Baseband changes BPI bus states to select operation right band. Modified the timing definition in **l1d_custom_rf.h**. **QB_PT2B** is the timing definition of 3rd BPI bus status changing of TX window.

```
#define  QB_PT2B                 -7
```

(5) Define the states of BPI bus for each band. The data is shown in the below table:

BPI_BUS	5	4	3	2	1	0	Hex
Function	BS1	PA_EN	reserved	reserved	BS0	TR_EN	
850	0	1	X	X	1	1	13h
GSM	0	1	X	X	1	1	13h
DCS	1	1	X	X	1	1	33h
PCS	1	1	X	X	1	1	33h

So the BPI data can be defined as follow:

```
          #define  PDATA_GSM850_PT2B         0x13
          #define  PDATA_GSM_PT2B            0x13
          #define  PDATA_DCS_PT2B            0x33
          #define  PDATA_PCS_PT2B            0x33
```

(6) Modified the timing definition in **l1d_custom_rf.h**. **QB_ST2B** is the timing definition of 3'rd BSI command of TX window.

```
          #define  QB_ST2B                   22
```

(7) Specify the command of BSI to be sent. Function **L1D_RF_SetSData_ST2B** in **m12195.c** should be implemented to perform this action. **SETUP_ST2B_ST2M()** is the first statement in **L1D_RF_SetSData_ST2B** which is a macro to hide complicated code to initialize setting 3'rd BSI data. The second statement **HWRITE_3_SDATA** is specified the data to be sent by BSI bus. The parameters of **HWRITE_3_SDATA** are control word and data word of BSI data. The control words **SCTRL_TESTREG**, **SCTRL_AMLOOP** and **SCTRL_START_PCL_CAL** are composed of device selection (0) and data bit count (24 bits). The 1st data word **SDATA_TESTREG1[3]** is phase loop coefficient which is provided by the RFWS. The 2nd data word is got form **AM_Loop_Calculate()** function which is for AD6546 Register8 calculation. The 3rd data word **SDATA_START_PCL_CAL** is used to trigger State-Machine sequence for initialization of the AM loop and is also used to select modulation mode between GMSK and EPSK.

```
/*AD6546*/ unsigned long  SDATA_TESTREG1[3]=
/*AD6546*/ { /*Can be used to improve phase loop TX blocker immunity at low PCL*/
/*AD6546*/   0x080F9E,        /*Used in CS service*/
/*AD6546*/   0x0C1F9E,        /*Used in PS service*/
/*AD6546*/   0x081F9E         /*Used in PS service*/
/*AD6546*/ };
/*AD6546*/ #define  SCTRL_TESTREG            SCTRL_WORD(0,24)
/*AD6546*/ #define  SCTRL_AMLOOP             SCTRL_WORD(0,24)
/*AD6546*/ #define  SCTRL_START_PCL_CAL      SCTRL_WORD(0,24)
/*AD6546*/ #define SDATA_START_PCL_CAL
           (0x020024L|(((l1d_rf.tx_mod_type2>>l1d_rf.cur_slot)&0x1)<<16))
/*AD6546*/ void L1D_RF_SetSData_ST2B( void )
/*AD6546*/ {
/*AD6546*/   SETUP_ST2B_ST2M();
/*AD6546*/
/*AD6546*/   if((l1d_rf.tx_mod_type2>>l1d_rf.cur_slot)&0x1)  // 8PSK
/*AD6546*/   {
/*AD6546*/      HWRITE_3_SDATA(      SCTRL_TESTREG, SDATA_TESTREG1[2],
/*AD6546*/                           SCTRL_AMLOOP, AM_Loop_Calculate(),
/*AD6546*/                           SCTRL_START_PCL_CAL,  SDATA_START_PCL_CAL );
/*AD6546*/   }
/*AD6546*/   else  // GMSK
/*AD6546*/   {
/*AD6546*/     HWRITE_3_SDATA(       SCTRL_TESTREG, SDATA_TESTREG1[1],
/*AD6546*/                                SCTRL_AMLOOP, AM_Loop_Calculate(),
/*AD6546*/                                SCTRL_START_PCL_CAL,  SDATA_START_PCL_CAL );
/*AD6546*/    }
/*AD6546*/ }
```

(8) The transmitter of AD6546 implements a large signal Polar Loop. The ST2B event triggers the PCL auto calibration in AD6546. This again triggers a VAPC (OE to PA) enable after approx 5 us and after another 2us TX PLL switches the feedback path to the external path through the PA. At point of VAPC enable the system is ready for closing the AM loop around the PA. PA_EN must be enabled just prior to this event and also the VRAMP voltage must be high enough for the loop to lock or there is a spike on PvT.

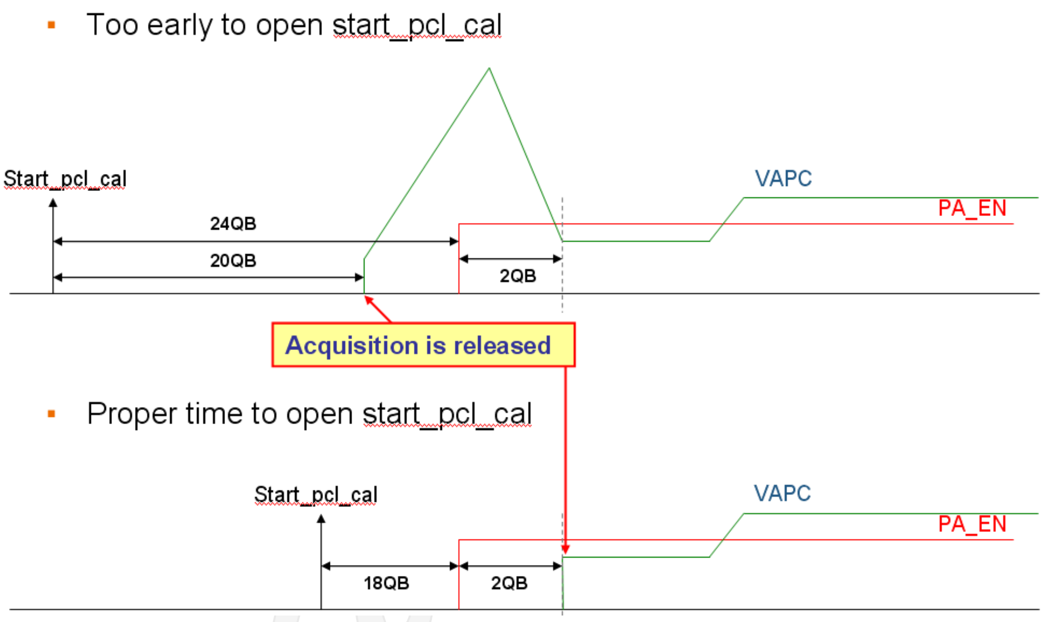


Figure 22 Zoom on ramp up for timing details about ST2B and PA_EN

(9) Because the DCM (dynamic clock management) would affect the timing taken for searching BSI data, the timing shift caused by DCM would change the point of PCL auto calibration and could affect the RF performance. In order to reduce the timing shift, we change the BSI data table and replace the front words with the ST2B/ST2M data words for OE. This can reduce the search time. Even though the DCM is applied, the effect has been reduced (<2us). The mechanism has a switch defined in **l1d_rf.h** to decide if apply this method or not. The default is turned on.

```
#define  IS_BSI_DATA_TABLE_Y_SHIFT            1
```

2.12.3.3 TX DAC Control

The timing setting for TX frame enable and frame synchronization control of MT6235 are needed. TX frame enable is used to enable or disable TX DAC. TX frame synchronization is used to turn on/off TX SPORT. The timing of enabling TX DAC to turning on TX SPORT named **QB_TX_FENA_2_FSYNC** and the timing of turning off TX SPORT to disabling TX DAC named **QB_TX_FSYNC_2_FENA** are both also needed to be defined in **l1d_custom_rf.h**. In this case, the timing of enabling TX DAC to turning on TX SPORT is 152 QB and the timing of turning off TX SPORT to disabling TX DAC is 20 QB. So **QB_TX_FENA_2_FSYNC** and **QB_TX_FSYNC_2_FENA** can be defined as:

```
#define  QB_TX_FENA_2_FSYNC                   152
#define  QB_TX_FSYNC_2_FENA                   26
```

2.12.3.4 Set APC

99 QB ahead of T0, APC DAC output performs a DC offset to let ramp up more smoothly. The timing of performing this DC offset is defined as **QB_APCDACON** in **l1d_custom_rf.h**. The value of APC DC offset is maintained in APC profile and it is introduced in the latter section.

```
#define  QB_APCDACON                        99
```

22 QB ahead of T0, APC DAC output performs ramping up. The timing of performing this ramp up is defined as **QB_APCON** in **l1d_custom_rf.h**.

```
#define  QB_APCON                           16
```

8 QB behind of T1, APC DAC output performs ramping down. The timing of performing this ramp down is defined as **QB_APCOFF** in **l1d_custom_rf.h**.

```
#define  QB_APCOFF                          7
```

2.12.3.5 Set Transceiver in idle Mode

23 QB behind T1, RF module finishes transmission data and MT6235 set it into idle mode. MT6235 Baseband activates control pins by changing BPI bus states, and also sends serial command to set idle mode command by BSI bus. To perform these operations, the timing and data of BPI and BSI bus can be implemented as follow steps:

(1) Modified the timing definition in **l1d_custom_rf.h**. **QB_ST3** is the timing definition of 3'rd BSI command of RX window. QB_PT3 is the timing definition of 3'rd BPI bus status changing of RX window.

```
#define  QB_ST3                             23
#define  QB_PT3                             22
```

(2) Define the data of BPI bus states for each band. The data is shown in the below table:

BPI_BUS	5	4	3	2	1	0	Hex
Function	BS1	PA_EN	reserved	reserved	BS0	TR_EN	
850	0	0	X	X	0	0	00h
GSM	0	0	X	X	0	0	00h
DCS	0	0	X	X	0	0	00h
PCS	0	0	X	X	0	0	00h

So the BPI data can be defined as follow:

```
#define  PDATA_GSM850_PT3          0x00
#define  PDATA_GSM_PT3             0x00
#define  PDATA_DCS_PT3             0x00
#define  PDATA_PCS_PT3             0x00
```

(3) Specify the command of BSI to be sent. Function **L1D_RF_SetSData_ST3** in **m12195.c** should be implemented to perform this action. The first statement in **L1D_RF_SetSData_ST3** is **SETUP_ST3()** which is a macro to hide complicated code to initialize setting 3'rd BSI data. The second statement **HWRITE_1_SDATA** is specified the data to be sent by BSI bus. The parameters of **HWRITE_1_SDATA** are control word and data word of BSI data. The control word **SCTRL_IDLE** is composed of device selection (0) and data bit count (18 bits). The data word **SDATA_IDLE** is the command to set AD6546 into idle mode. **SCTRL_IDLE** and **SDATA_IDLE** are defined in **l1d_rf.h**.

```
/*AD6546*/ #define  SCTRL_IDLE                     SCTRL_WORD(0,18)
/*AD6546*/ #define  SDATA_IDLE                     0x000184
/*AD6546*/ void  L1D_RF_SetSData_ST3( void )
/*AD6546*/ {
/*AD6546*/   SETUP_ST3();
/*AD6546*/   HWRITE_1_SDATA( SCTRL_IDLE, SDATA_IDLE );
/*AD6546*/ }
```

The propagation delay when transmit data from Baseband uplink DAC to antenna should be claimed in file **l1d_custom_rf.c**. This delay depends on different RF module. For AD6546 RF module, the delay is 42 QB.

```
          #define  TX_PROPAGATION_DELAY         42
```

2.12.3.6 Event Activation

If Layer 1 plans to transmit 148-bit data from TDMA timer count 2147 QB to 2739 QB, the above setting applies to create the TX window.

(1) At TDMA timer count is 1899 QB (248 QB ahead of TX SPORT turn on), BPI bus changes its states to 00h (GSM850/GSM/DCS/PCS). And after 22 QB (226 QB ahead of TX SPORT turn on), the serial command of setting operation frequency is sent to AD6546 transceiver.
(2) At TDMA timer count is 1995 QB (152 QB ahead of TX SPORT turn on), TX ADC in MT6235 is enable.
(3) At TDMA timer count is 1999 QB (148 QB ahead of TX SPORT turn on), the serial command to set transmit mode is sent.
(4) At TDMA timer count is 2048 QB (99 QB ahead of TX SPORT turn on), APC DAC output performs a DC offset to let ramp up more smoothly.
(5) At TDMA timer count is 2125 QB (22 QB ahead of TX SPORT turn on), the 3'rd serial command to set phase loop coefficient and AM loop parameter is sent.
(6) At TDMA timer count is 2131 QB (16 QB ahead of TX SPORT turn on), APC DAC output performs ramping up.
(7) At TDMA timer count is 2143 QB (4 QB ahead of TX SPORT turn on), BPI bus changes its states to 12h (GSM850/GSM) or 32h (DCS/PCS) to enable PA.
(8) At TDMA timer count is 2147 QB, TX SPORT in MT6235 is turned on.
(9) At TDMA timer count is 2154 QB (7 QB behind TX SPORT turn on), BPI bus changes its states to 13h (GSM850/GSM) or 33h (DCS/PCS) to turn on TRSW.
(10) At TDMA timer count is 2739 QB, TX SPORT in MT6235 is turned off.
(11) At TDMA timer count is 2744 QB (7 QB behind TX SPORT turning off), APC DAC output performs ramping down.
(12) At TDMA timer count is 2761 QB (22 QB behind TX SPORT turning off), BPI bus changes its states to 00h.
(13) At TDMA timer count is 2762 QB (23QB behind TX SPORT turning off), the 3-wire command of setting AD6546 into idle mode is sent.
(14) At TDMA timer count is 2765 QB (26 QB behind TX SPORT turning off), TX DAC is disabled at this time.

2.12.4 TX window modulation type setup

For the TX window, two type of the modulation, GMSK/EPSK can be selected by sending serial command by BSI bus. To perform these operations, the timing and data of BSI and BPI bus can be implemented as follow steps:

(1) As for the interslot modulation changing of the TX window, the BPI event named PT2M is used. For the event timing of the PT2M could be found in file **l1d_custom_rf.c**. The definitions are as followings.

```
  #define  QB_PT2M1_G8              6      /*TRSW Off when modulation mode is changing*/
  #define  QB_PT2M2_G8              2      /*TRSW On when modulation mode change done*/
```

```
#define  QB_PT2M1_8G              6       /*TRSW Off when modulation mode is changing*/
#define  QB_PT2M2_8G              2       /*TRSW On when modulation mode change done*/
```

(2) The following definition in file **l1d_custom_rf.h** according to the operating band is defined.

```
#define  PDATA_GSM850_PT2M1_G8                  0x12
#define  PDATA_GSM850_PT2M2_G8                  0x13
#define  PDATA_GSM850_PT2M1_8G                  0x12
#define  PDATA_GSM850_PT2M2_8G                  0x13
#define  PDATA_GSM_PT2M1_G8                     0x12
#define  PDATA_GSM_PT2M2_G8                     0x13
#define  PDATA_GSM_PT2M1_8G                     0x12
#define  PDATA_GSM_PT2M2_8G                     0x13
#define  PDATA_DCS_PT2M1_G8                     0x32
#define  PDATA_DCS_PT2M2_G8                     0x33
#define  PDATA_DCS_PT2M1_8G                     0x32
#define  PDATA_DCS_PT2M2_8G                     0x33
#define  PDATA_PCS_PT2M1_G8                     0x32
#define  PDATA_PCS_PT2M2_G8                     0x33
#define  PDATA_PCS_PT2M1_8G                     0x32
#define  PDATA_PCS_PT2M2_8G                     0x33
```

Note that the team **G8** is used to identify the changing from the GMSK to the EPSK (8PSK) and the team **8G** is used to identify the changing from the EPSK to the GMSK.

(3) Specify the command of BSI to be sent. Function **L1D_RF_SetSData_ST2M** in **m12195.c** should be implemented to perform this action. **SETUP_ST2B_ST2M()** is the first statement in **L1D_RF_SetSData_ST2B** which is a macro to hide complicated code to initialize setting BSI data. The second statement **HWRITE_3_SDATA** is specified the data to be sent by BSI bus. The parameters of **HWRITE_3_SDATA** are control word and data word of BSI data. The control words **SCTRL_TESTREG**, **SCTRL_AMLOOP** and **SCTRL_MOD_CHANGE** are composed of device selection (0) and data bit count (24 bits). The 1st data word **SDATA_TESTREG1[3]** is phase loop coefficient which is provided by the RFWS. The 2nd data word is got form **AM_Loop_Calculate()** function which is for AD6546 Register8 calculation. According to modulation type, the 3rd data word **SDATA_MOD8** is used to select EPSK modulation mode or **SDATA_MODG** is used to select GMSK modulation mode.

```
/*AD6546*/     #define  SCTRL_TESTREG               SCTRL_WORD(0,24)
/*AD6546*/     #define  SCTRL_AMLOOP                SCTRL_WORD(0,24)
/*AD6546*/     #define  SCTRL_MOD_CHANGE            SCTRL_WORD(0,24)
/*AD6546*/     #define  SDATA_MOD8                  0x010024L
/*AD6546*/     #define  SDATA_MODG                  0x010044L
/*AD6546*/ unsigned long  SDATA_TESTREG1[3]=
/*AD6546*/ { /*Can be used to improve phase loop TX blocker immunity at low PCL*/
/*AD6546*/   0x080F9E,/*Used in CS service*/
/*AD6546*/   0x0C1F9E,/*Used in PS service*/
/*AD6546*/   0x081F9E /*Used in PS service*/
/*AD6546*/ };
/*AD6546*/ void L1D_RF_SetSData_ST2M( void )
/*AD6546*/ {
/*AD6546*/   SETUP_ST2B_ST2M();
/*AD6546*/
/*AD6546*/   if((l1d_rf.tx_mod_type2>>l1d_rf.cur_slot)&0x1)
/*AD6546*/   {
/*AD6546*/       HWRITE_3_SDATA(     SCTRL_TESTREG,     SDATA_TESTREG1[2],
/*AD6546*/                           SCTRL_AMLOOP, AM_Loop_Calculate(),
/*AD6546*/                           SCTRL_MOD_CHANGE,    SDATA_MOD8      )
/*AD6546*/   }
/*AD6546*/   else
/*AD6546*/   {
/*AD6546*/      HWRITE_3_SDATA(      SCTRL_TESTREG,      SDATA_TESTREG1[1],
/*AD6546*/                           SCTRL_AMLOOP, AM_Loop_Calculate(),
/*AD6546*/                           SCTRL_MOD_CHANGE,    SDATA_MODG      )
/*AD6546*/   }
/*AD6546*/ }
```

2.12.5 Switch CAPID for 26MHz settle time

26MHz settle time on AD6546 will depends on CAPID. When the CAPID is bigger the 26MHz settle time is longer. Using CAPID 63, the 26MHz may not be stable before the time the MCU start to work with 26MHz instead of 32KHz after waking up. It would cause the whole MCU would work abnormally. Before entering sleep mode, switch CAPID to 0. It can guarantee the 26MHz settle time would be stable before MCU start to work. After waking up, switch the CAPID to the calibrated result. The corresponding functions are **L1D_RF_CAP_Set()** and **L1D_RF_CAP_Clear()** in **m12196.c**.

2.13 RF Control Configuration of AD6548 RF Module

For example, the schematic circuit interface between AD6548 RF module and MT6235 chip is shown in below diagram. Please refer to MediaTek AD6548 RF Module schematic circuit to get more detail information.

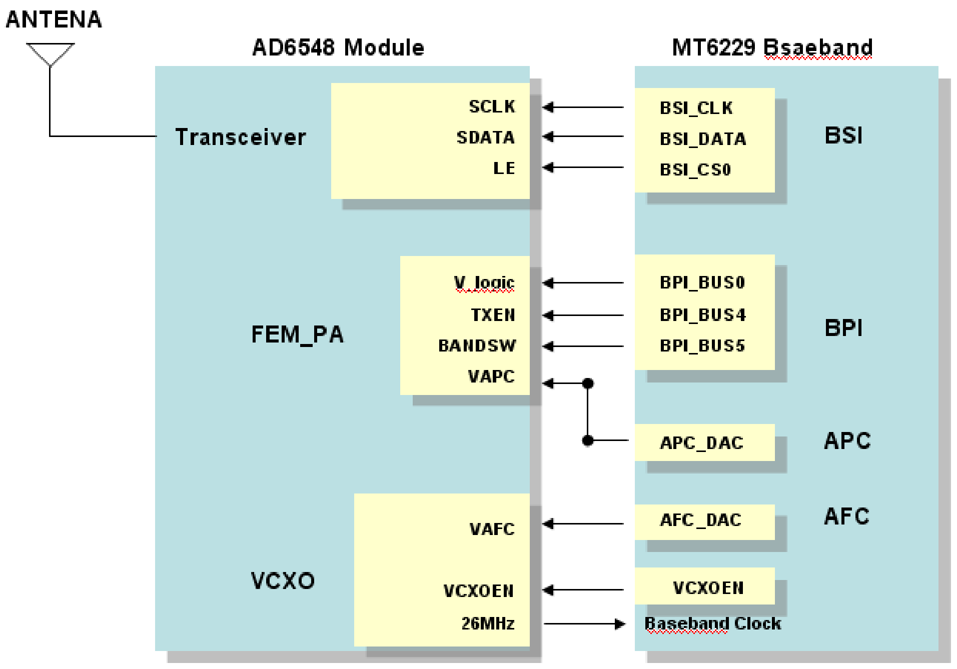


Figure 23 The connection of MT6235 and AD6548 RF Module

BSI_CLK, BSI_DATA, BSI_CS0 pins of MT6235 are directly connected to the serial interface of AD6548 transceiver. APC_DAC pin of MT6235 is connected to the VAPC pin of PA of RF module board. VCXOEN of MT6235 controls the enable state of VCXO. AFC DAC output is connected to the voltage control pin VAFC of VCXO. The generated clock of VCXO provides MT6235 system clock. The LDOs of AD6548 transceiver are independently controlled vai the 3 wire serial bus. Band-related pins V_logic, TXEN, and BANDSW are controlled by the bit 0,4,5 of BPI bus of MT6235. The high/low state of each pin of BPI bus and their timing of changing state can be programmed by MT6235. The RF radio control method is introduced in the latter description.

2.13.1 Initialization

RF module can be controlled by the BSI, BPI, APC, AFC units of MT6235 chips. BSI, BPI, APC, AFC units can work in 2 modes. One mode is immediate mode, and the other is TDMA event control mode.

Immediate control mode is that the output signal of BSI, BPI, APC, or AFC unit is generated as soon as setting the corresponding registers in MT6235. When turning on the RF module, the immediate mode can be used to initialize the RF module. For example, immediate control mode is used when initializing AD6548 RF module, and the initial states are set as follow:

(1) V_logic (BPI_BUS0) is low state
(2) TXEN (BPI_BUS4) is low state
(3) BANDSW (BPI_BUS5) is low state

(4) Serial commands (0x00000000, 0x00000184, 0x00000105, 0x00291686, 0x0000001D, 0x0000001F, 0x00006384, 0x00022F80, 0x00002101, 0x00000184) are sending to AD6548 transceiver.

The initialization of BPI, BSI data can be implemented in **L1D_RF_PowerOn** in **m12196.c** by using immediate mode.

```
/*AD6548*/  #define INITIAL_REG0_RESET      0x00000000  /* Performs chip reset */
/*AD6548*/  #define INITIAL_REG0_NORST      0x00000080  /* No Reset performed */
/*AD6548*/  #define INITIAL_REG1            0x00000001|(XO_CapID<<8)
/*AD6548*/  #define INITIAL_REG5            0x00000105
/*AD6548*/  #define INITIAL_REG6            0x00291686
/*AD6548*/  #define INITIAL_REG29           0x0000001D
/*AD6548*/  #define INITIAL_REG31           0x0000001F
/*AD6548*/  #define PROG_POWER_CAL          0x00006384
/*AD6548*/  #define PROG_POWER_ON           0x00000184
/*AD6548*/  #define PROG_POWER_OFF          0x00000004
/*AD6548*/  #define PROG_CAL_BAND           0x00022F80
/*AD6548*/
/*AD6548*/  void  L1D_RF_PowerOn( void )
/*AD6548*/  {
/*AD6548*/     IMM_MODE_BEGIN( IMMMASK_ALL );
/*AD6548*/     IMM_SEND_BPI( PDATA_INIT );
/*AD6548*/  #ifndef L1D_SIM
/*AD6548*/     if(l1d_rf.is_init)
/*AD6548*/     {
/*AD6548*/     IMM_SEND_BSI( SCTRL_IMOD(0,24), INITIAL_REG0_RESET ); /*Performs chip reset*/
/*AD6548*/     IMM_SEND_BSI( SCTRL_IMOD(0,24), PROG_POWER_ON );      /*Power on LDO1 & LDO2*/
/*AD6548*/     IMM_SEND_BSI( SCTRL_IMOD(0,24), INITIAL_REG5 );       /*BB gain 3dB*/
/*AD6548*/     IMM_SEND_BSI( SCTRL_IMOD(0,24), INITIAL_REG6 );       /*PLL Word*/
               /*Test Register, initialize only*/
/*AD6548*/     IMM_SEND_BSI( SCTRL_IMOD(0,24), INITIAL_REG29 );
/*AD6548*/     IMM_SEND_BSI( SCTRL_IMOD(0,24), INITIAL_REG31 );
/*AD6548*/     WAIT_TIME_QB(27);             /* Wait 25 us after LDO1 and LDO2 on */
/*AD6548*/     IMM_SEND_BSI( SCTRL_IMOD(0,24), PROG_POWER_CAL );
/*AD6548*/     WAIT_TIME_QB(27);             /* Wait 25+25 us after LDO1 and LDO2 on */
/*AD6548*/     IMM_SEND_BSI( SCTRL_IMOD(0,24), PROG_CAL_BAND );
/*AD6548*/     WAIT_TIME_QB(4660);           /* Wait 4.3ms for RF internal calibration done */
/*AD6548*/     }
/*AD6548*/  #endif
/*AD6548*/     IMM_SEND_BSI( SCTRL_IMOD(0,24), INITIAL_REG1 );       /* AFC_CAP */
/*AD6548*/     IMM_SEND_BSI( SCTRL_IMOD(0,24), PROG_POWER_ON );
/*AD6548*/     IMM_MODE_END(  );/*Turn Immediate Mode off*/
/*AD6548*/  }
```

2.13.2 Create RX window

TDMA timer is a counter unit that counts 1 per quarter-bit (0.923u). It always repeats counting from 0 to 4999. So the period of TDMA counter is 5000 QB (1 TDMA frame). If BSI, BPI, AFC, APC are programmed in TDMA event driven mode, the active time can be programmed. When the TDMA counter counts to the programmed time (count), the corresponding unit is triggered to activate.

If it is needed to receive data from air or transmit data to air, Layer1 needs to plan all RF setting and timing to create RX/TX window, control receiving gain, control VCXO, control PA…etc. For this purpose, TDMA event driven mode of BSI, BPI, APC, or AFC units should be used.

For example of creating RX window to receive data, the RF setting and timing of AD6548 RF module can be depicted in the below diagram:

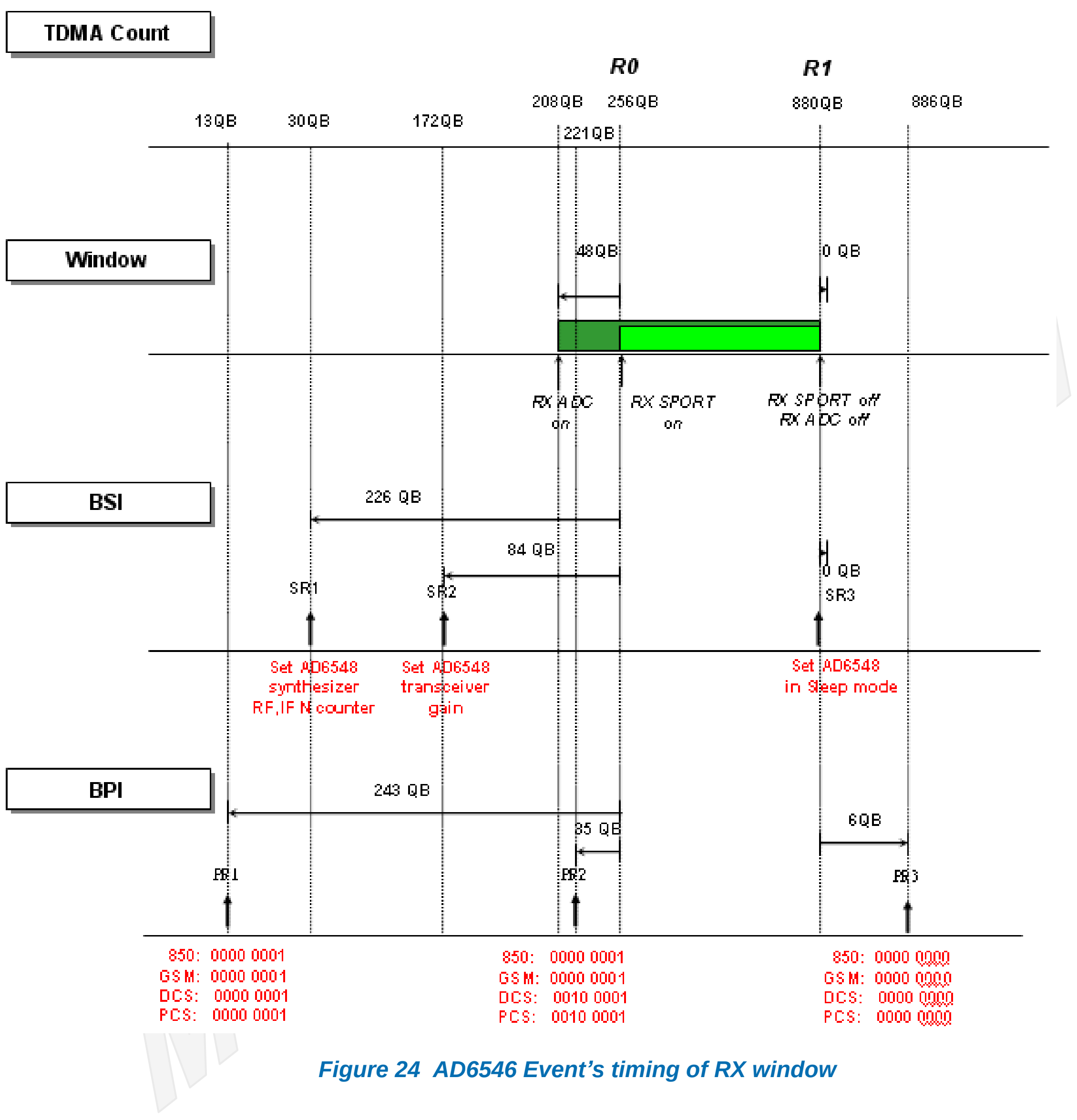


Figure 24 AD6546 Event's timing of RX window

The TDMA timing and data setting are described in 2 parts. One part is that before turning on RX SPORT to receive I/Q data from AD6548 transceiver. And the other part is after turning off RX SPORT to finish receiving I/Q data. To describe these two parts easily, the timing of turning on RX SPORT is taken as one base named **R0**. And the timing of turning off RX SPORT is taken as one base named **R1**. In this case, R0 is at TDMA counter 256 QB and R1 is at TDMA counter 880 QB.

2.13.2.1 Set Synthesizer N-Counter

243 QB ahead of R0, RF module should start to operate. MT62xx chip controls the RF module by activating control pins and sending serial command. MT62xx activates the control pins by BPI bus status. The high/low state of BPI bus at this time depends on operation band. After 17QB, MT62xx also sends serial command to lock AD6548 synthesizer to lock the operation frequency by BSI bus. To perform these operations, the timing and data of BPI and BSI bus can be implemented as follow steps:

(1) Modified the timing definition in **l1d_custom_rf.h**. **QB_SR1** is the timing definition of 1'st BSI command of RX window. **QB_PR1** is the timing definition of 1'st BPI bus status changing of RX window.

```
          #define  QB_SR1                    226
          #define  QB_PR1                    243
```

(2) Define the states of BPI bus for each band. The data is shown in the below table:

BPI_BUS	5	4	3	2	1	0	Hex
Function	BANDSW	TXEN	reserved	reserved	reserved	V_logic	
850	0	0	X	X	X	1	01h
GSM	0	0	X	X	X	1	01h
DCS	0	0	X	X	X	1	01h
PCS	0	0	X	X	X	1	01h

So the BPI data can be defined as follow:

```
          #define  PDATA_GSM850_PR1          0x01
          #define  PDATA_GSM_PR1             0x01
          #define  PDATA_DCS_PR1             0x01
          #define  PDATA_PCS_PR1             0x01
```

(3) Specify the command of BSI to be sent. Function **L1D_RF_SetSData_SR1** in **m12195.c** be implemented to perform this action. **LNA_SWAP_INDEX[LNA_SET_IDX][l1d_rf.band]** is the table which is used for swap function. **LNA_SET_IDX** is index and used to tell which band has been swap. **SETUP_SR1()** is the first statement in **L1D_RF_SetSData_SR1** which is a macro to hide complicated code to initialize setting 1'st BSI data. The second statement **HWRITE_2_SDATA** is specified the data to be sent by BSI bus. Two parameters of **HWRITE_2_SDATA** are control word and data word of BSI data. The control words **SCTRL_WARM** and **SCTRL_PLL** are composed of device selection (0) and data bit count (24 bits). The 1st data word **SDATA_WARM_BSEL[CW_WARM_IDX]** is the command to set AD6548 into Warm-up mode which is corresponding with the band. **CW_WARM_IDX** is the local variable which indicates the band of operation frequency. The 2nd data word **l1d_rf.RFN_data**, RF N-Counter setting value, is the evaluation result of **L1D_RF_GetRxPLLSetting** called by Layer 1.

```
/*AD6548*/#define  SCTRL_WARM        SCTRL_WORD(0,24)
/*AD6548*/#define  SCTRL_PLL         SCTRL_WORD(0,24)
/*AD6548*/#define  LNA_SET_IDX       (((BBTXParameters.bbrx_dcs1800_pcs1900_swap&0x1)<<1)|
                                     (BBTXParameters.bbrx_gsm850_gsm900_swap&0x1))
/*AD6548*/
/*AD6548*/ const unsigned long  SDATA_WARM_BSEL[5]=
/*AD6548*/ { /*Warm*/  /*Band sel*/
/*AD6548*/   0x000184,              /*GSM 450 no support*/
/*AD6548*/   0x006184 | 0x001800,    /*GSM 850*/
/*AD6548*/   0x006184 | 0x000000,    /*GSM 900*/
/*AD6548*/   0x006184 | 0x000800,    /*DCS 1800*/
/*AD6548*/   0x006184 | 0x001000,    /*PCS 1900*/
/*AD6548*/ };
/*AD6548*/ const unsigned short  LNA_SWAP_INDEX[4][5]=
/*AD6548*/ {
/*AD6548*/   { 0, 1, 2, 3, 4 },      /*No Swap*/
/*AD6548*/   { 0, 2, 1, 3, 4 },      /*Low Band Swap*/
/*AD6548*/   { 0, 1, 2, 4, 3 },      /*High Band Swap*/
/*AD6548*/   { 0, 2, 1, 4, 3 },      /*High&Low Band Swap*/
/*AD6548*/ };
/*AD6548*/ void  L1D_RF_SetSData_SR1( void )
/*AD6548*/ {
/*AD6548*/    unsigned short CW_WARM_IDX = LNA_SWAP_INDEX[LNA_SET_IDX][l1d_rf.band];
/*AD6548*/    SETUP_SR1();
/*AD6548*/    HWRITE_2_SDATA(       SCTRL_WARM,    SDATA_WARM_BSEL[ CW_WARM_IDX ],
/*AD6548*/                           SCTRL_PLL,     l1d_rf.RFN_data);
/*AD6548*/
/*AD6548*/ }
```

To create the receiving window with right operation frequency, evaluating the setting value of operation frequency is needed. Function **L1D_RF_GetRxPLLSetting** in file **m12191.c** needs to be implemented to evaluate the synthesizer setting. And Layer 1 calls this function when programs the RF synthesizer setting value and saves the result into variable **l1d_rf.RFN_data**. This content of this variable is the command to be sent to transceiver chip to lock the operation frequency. The evaluation of AD6548 synthesizer N counter setting is as the following code.

```
/*AD6548*/void  L1D_RF_GetRxPLLSetting( int rf_band, int arfcn, long *rfN, long *ifN )
/*AD6548*/{
/*AD6548*/   switch(rf_band)
/*AD6548*/   {
/*AD6548*/      case  FrequencyBand850 :
/*AD6548*/      {
/*AD6548*/         if(arfcn<=201)
/*AD6548*/         {   if(arfcn<=158)
/*AD6548*/             {                                /* ARFCN : 128~158      */
/*AD6548*/                 *rfN = (((arfcn-128)*24 + 304)<<13) | 0x1206L /*(36L<<7|0x06L)*/;
/*AD6548*/             }
/*AD6548*/             else
/*AD6548*/             {                              /* ARFCN : 159~201      */
/*AD6548*/                 *rfN = (((arfcn-159)*24 + 8)<<13) | 0x1286L /*(37L<<7|0x06L)*/;
/*AD6548*/             }
/*AD6548*/         }
/*AD6548*/         else
/*AD6548*/         {   if(arfcn<=245)
/*AD6548*/             {                                /* ARFCN : 202~245      */
/*AD6548*/                 *rfN = (((arfcn-202)*24 )<<13) | 0x1306L /*(38L<<7|0x06L)*/;
/*AD6548*/             }
/*AD6548*/             else
/*AD6548*/             {                              /* ARFCN : 246~251      */
/*AD6548*/                 *rfN = (((arfcn-246)*24+16)<<13) | 0x1386L /*(39L<<7|0x06L)*/;
/*AD6548*/             }
/*AD6548*/         }
/*AD6548*/         break;
/*AD6548*/      }
/*AD6548*/      case  FrequencyBand900 :
/*AD6548*/      {      :
/*AD6548*/             :
/*AD6548*/         break;
/*AD6548*/      }
/*AD6548*/      case  FrequencyBand1800 :
/*AD6548*/      {      :
/*AD6548*/             :
/*AD6548*/         break;
/*AD6548*/      }
/*AD6548*/      case  FrequencyBand1900 :
/*AD6548*/      {      :
/*AD6548*/             :
/*AD6548*/         break;
/*AD6548*/      }
/*AD6548*/      default :
/*AD6548*/      {
/*AD6548*/         break;
/*AD6548*/      }
/*AD6548*/    }
/*AD6548*/    *ifN = 0;
/*AD6548*/}
```

2.13.2.2 Set Transceiver Gain

84 QB ahead of R0, Band selection and Rx gain setting should be performed. MT6235 activates the control pins by BPI bus status, and also sends serial command to set transceiver gain by BSI bus. To perform these operations, the timing and data of BPI and BSI bus can be implemented as follow steps:

(1) Modified the timing definition in **l1d_custom_rf.h**. **QB_SR2** is the timing definition of 2'nd BSI command of RX window. **QB_PR2** is the timing definition of 2'nd BPI bus status changing of RX window.

```
#define  QB_SR2                         84
#define  QB_PR2                         35
```

To support multi-slots receiving in GPRS mode, different LNA gain setting for each time slot is allowed. The TDMA timing of LNA gain setting at inter slot shall be defined as

```
#define  QB_SR2M                        36
```

This timing QB_SR2M is ahead the R0 of the next slot.

(2) Define the states of BPI bus for each band. The data is shown in the below table:

BPI_BUS	5	4	3	2	1	0	Hex
Function	BANDSW	TXEN	reserved	reserved	reserved	V_logic	
850	0	0	X	X	X	1	01h
GSM	0	0	X	X	X	1	01h
DCS	1	0	X	X	X	1	21h
PCS	1	0	X	X	X	1	21h

So the BPI data can be defined as follow:

```
#define  PDATA_GSM850_PR2        0x01
#define  PDATA_GSM_PR2           0x01
#define  PDATA_DCS_PR2           0x21
#define  PDATA_PCS_PR2           0x21
```

(3) Specify the command of BSI to be sent. Function **L1D_RF_SetSData_SR2** in **m12195.c** should be implemented to perform this action. The first statement in **L1D_RF_SetSData_SR2** is **SETUP_SR2()** which is a macro to hide complicated code to initialize setting 2'nd BSI data. The second statement HWRITE_2_SDATA is specified the data to be sent by BSI bus. The parameters of **HWRITE_2_SDATA** are control word and data word of BSI data. If this is the first slot of continuous RX slot or this is the single slot, the control words **SCTRL_AGC** and **SCTRL_RXCON** which are composed of device selection (0) and data bit count (24 bits) and two kinds of BSI data are sent. The 1st command **l1d_rf.AGC_data**, transceiver gain setting value, is the evaluation result of **L1D_RF_GetGainSetting** called by Layer 1. The data word **SDATA_WARM_BSEL[CW_WARM_IDX]| SDATA_RXON_MASK** is command to set AD6548 into RX mode. If this is not the first slot of continuous RX slot, the control word **SCTRL_AGC** and command **l1d_rf.AGC_data** are just sent.

```
/*AD6548*/  #define    SCTRL_AGC              SCTRL_WORD(0,24)
/*AD6548*/  #define    SCTRL_RXON             SCTRL_WORD(0,24)
/*AD6548*/  #define    SDATA_RXON_MASK        0x000200L
/*AD6548*/ void  L1D_RF_SetSData_SR2( void )
/*AD6548*/ {
```

```
/*AD6548*/    SETUP_SR2();
/*AD6548*/
/*AD6548*/    if(IS_CONTINUOUS_RX_SLOT())
/*AD6548*/    {
/*AD6548*/       HWRITE_2_SDATA(     SCTRL_AGC,     l1d_rf.AGC_data,
/*AD6548*/                           SCTRL_RESERVED,    0 );
/*AD6548*/    }
/*AD6548*/    else
/*AD6548*/    {
/*AD6548*/       unsigned short CW_WARM_IDX = LNA_SWAP_INDEX[LNA_SET_IDX][l1d_rf.band];
/*AD6548*/       HWRITE_2_SDATA( SCTRL_AGC,     l1d_rf.AGC_data,
/*AD6548*/                    SCTRL_RXON,    SDATA_WARM_BSEL[CW_WARM_IDX]|SDATA_RXON_MASK );
/*AD6548*/    }
/*AD6548*/ }
```

To create the receiving window with suitable gain of receiving amplifier, evaluating the setting value of request gain is needed. Function **L1D_RF_GetGainSetting** in file **m12192.c** needs to be implemented to evaluate the transceiver gain setting. And Layer 1 calls this function when programs the gain setting value and saves the result into variable **l1d_rf.AGC_data**. This content of this variable is the command to be sent to transceiver chip to set the requested receive amplifier gain. The evaluation of AD6548 receiving amplifier gain setting is as the following code:

```
int L1D_RF_GetGainSetting( int rf_band, int arfcn, int request_gain, long *gain_setting )
/*AD6548*/ {  int   max_gain, min_gain, real_gain;
/*AD6548*/    const int*    d32ptr;
/*AD6548*/    const sL1DAGCDATA*  agcptr;
/*AD6548*/    long  setting;
/*AD6548*/    int   bit_no;
/*AD6548*/
/*AD6548*/    /* evaluate the range of available gain */
/*AD6548*/    d32ptr  = (int*)&(GAIN_RANGE_TABLE[rf_band]);
/*AD6548*/    max_gain = *d32ptr++;
/*AD6548*/    min_gain = *d32ptr;
/*AD6548*/
/*AD6548*/    /* clipping the request gain to the avialable gain */
/*AD6548*/    if( request_gain>=max_gain )
/*AD6548*/    {  request_gain = max_gain;  }
/*AD6548*/    else  if( request_gain<=min_gain )
/*AD6548*/    {  request_gain = min_gain;  }
/*AD6548*/
/*AD6548*/    /* evaluate the real gain setting */
/*AD6548*/    agcptr = AGC_TABLE[rf_band];
/*AD6548*/    if( request_gain < agcptr->pos_gain )
/*AD6548*/    {  agcptr++;
/*AD6548*/    }
/*AD6548*/    {  bit_no    = BIT_NO( request_gain, agcptr->A, GC_B );
/*AD6548*/       real_gain = REAL_GAIN( bit_no, agcptr->A, GC_B );
/*AD6548*/       setting   = agcptr->setting | (bit_no<<7);
/*AD6548*/    }
/*AD6548*/
/*AD6548*/    *gain_setting = setting;
/*AD6548*/    return( real_gain );
/*AD6548*/ }
```

2.13.2.3 Set Transceiver in idle Mode

6 QB behind R1, RF module finishes receiving data and MT6235 set into idle mode. MT6235 Baseband activates control pins by changing BPI bus states, and also sends serial command to set idle mode command by BSI bus. To perform these operations, the timing and data of BPI and BSI bus can be implemented as follow steps:

(1) Modified the timing definition in **l1d_custom_rf.h**. **QB_SR3** is the timing definition of 3'rd BSI command of RX window. **QB_PR3** is the timing definition of 3'rd BPI bus status changing of RX window.

```
#define   QB_SR3                    0
#define   QB_PR3                    6
```

(2) Define the data of BPI bus states for each band. The data is shown in the below table:

BPI_BUS	5	4	3	2	1	0	Hex
Function	BANDSW	TXEN	reserved	reserved	reserved	V_logic	
850	0	0	X	X	X	0	00h
GSM	0	0	X	X	X	0	00h
DCS	0	0	X	X	X	0	00h
PCS	0	0	X	X	X	0	00h

So the BPI data can be defined as follow:

```
#define   PDATA_GSM850_PR3          0x00
#define   PDATA_GSM_PR3             0x00
#define   PDATA_DCS_PR3             0x00
#define   PDATA_PCS_PR3             0x00
```

(3) Specify the command of BSI to be sent. Function **L1D_RF_SetSData_SR3** in **m12195.c** should be implemented to perform this action. The first statement in **L1D_RF_SetSData_SR3** is **SETUP_SR3()** which is a macro to hide complicated code to initialize setting 3'rd BSI data. The second statement **HWRITE_1_SDATA** is specified the data to be sent by BSI bus. The parameters of **HWRITE_1_SDATA** are control word and data word of BSI data. The control word **SCTRL_IDLE** is composed of device selection (0) and data bit count (18 bits). The data word **SDATA_IDLE** is the command to set AD6548 into idle mode. **SCTRL_IDLE** and **SDATA_IDLE** are defined in **l1d_rf.h**.

```
/*AD6548*/  #define  SCTRL_IDLE                  SCTRL_WORD(0,18)
/*AD6548*/  #define  SDATA_IDLE                  0x000184
/*AD6548*/  void  L1D_RF_SetSData_SR3( void )
/*AD6548*/  {
/*AD6548*/     SETUP_SR3();
/*AD6548*/     HWRITE_1_SDATA( SCTRL_IDLE, SDATA_IDLE );
/*AD6548*/  }
```

2.13.2.4 RX ADC CONTROL

Besides setting the timing and data of BPI and BSI bus of MT6235 Baseband to create a RX window, the timing setting for RX frame enable and frame synchronization control of MT6235 are needed. RX frame enable is used to enable or disable RX DAC. RX frame synchronization is used to turn on/off RX SPORT. The timing of enabling TX DAC to turning on RX SPORT named **QB_RX_FENA_2_FSYNC** and the timing of turning off RX SPORT to disabling RX DAC named **QB_RX_FSYNC_2_FENA** are both also needed to be defined in **l1d_custom_rf.h**. In this case, the timing of enabling RX DAC to turning on RX SPORT is recommend to be 48 QB and the timing of turning off RX SPORT to disabling TX DAC is 0 QB. So **QB_RX_FENA_2_FSYNC** and **QB_RX_FSYNC_2_FENA** can be defined as:

```
#define  QB_RX_FENA_2_FSYNC                    48
#define  QB_RX_FSYNC_2_FENA                    0
```

2.13.2.5 Event Activation

If Layer1 plans to receive 156-bit data from TDMA timer count 256 QB to 880 QB, the above setting applies to create the RX window.

(1) At TDMA timer count is 13QB (243QB ahead of RX SPORT turn on), BPI bus changes its states to 01h (GSM850/GSM/DCS/PCS). And after 17 QB (226 QB), the serial command of setting operation frequency is sent to AD6548 transceiver.
(2) At TDMA timer count is 172 QB (84 QB ahead of RX SPORT turn on), the serial command of setting RX gain is sent to AD6548 transceiver.
(3) At TDMA timer count is 208 QB (48 QB ahead of RX SPORT turn on), RX ADC in MT6235 is enable.
(4) At TDMA timer count is 221 QB (35 QB ahead of RX SPORT turn on), BPI bus changes its states to 01h (GSM850/GSM) or 21h (DCS/PCS).
(5) At TDMA timer count is 256 QB, RX SPORT in MT6235 is turned on. The I/Q data begin to be sampled.
(6) At TDMA timer count is 880 QB, RX SPORT in MT6235 is turned off and RX ADC is also disabled. The I/Q data stop sampling.
(7) At TDMA timer count is 880 QB (at RX SPORT turning off), the command of setting AD6548 into idle mode is sent.
(8) At TDMA timer count is 886 QB(6 QB behind RX SPORT turning off), BPI bus changes its states to 00h.

2.13.3 Create TX window

Another example is creating TX window to transmit data. The RF setting and timing of AD6548 RF module can be depicted in the below diagram:

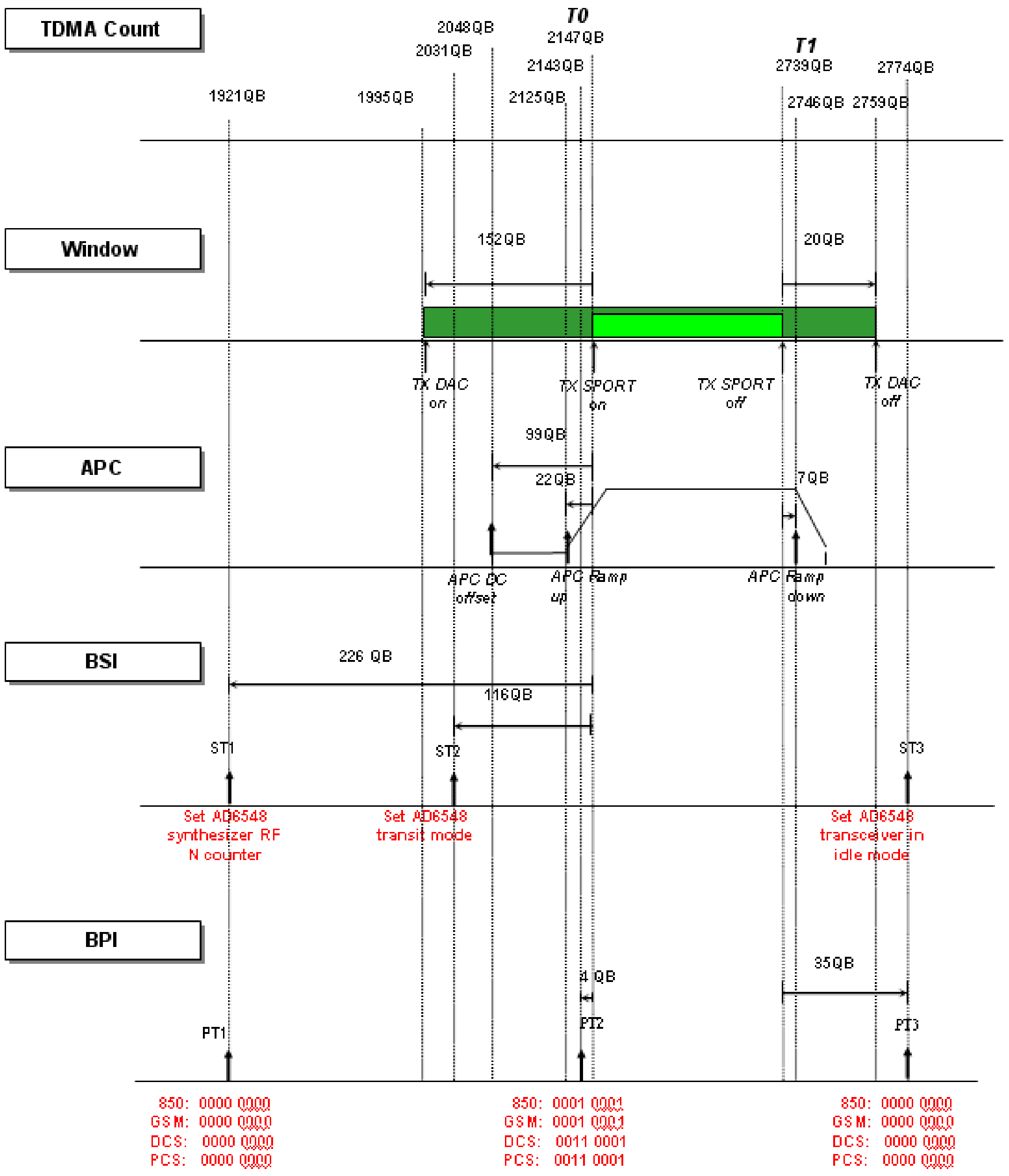


Figure 25 AD6546 Event's timing of TX window

The TDMA timing and data setting are described in 2 parts. One part is that before turning on TX SPORT to receive I/Q data from AD6548 transceiver. And the other part is after turning off TX SPORT to finish receiving I/Q data. To describe these two parts easily, the timing of turning on TX SPORT is taken as one base named **T0**. And the timing of turning off TX SPORT is taken as one base named **T1**. In this case, T0 is at TDMA counter 2147 QB and T1 is at TDMA counter 2739 QB.

2.13.3.1 Set synthesizer N-Counter

226 QB ahead of R0, RF module start to work. MT6235 Baseband activates control pins by changing BPI bus states. Then MT6235 sends serial command to lock AD6548 synthesizer to lock the operation frequency by BSI bus. To perform this operation, the timing and data of BSI bus can be implemented as follow steps:

(1) Modified the timing definition in **l1d_custom_rf.h**. **QB_ST1** is the timing definition of 1'st BSI command of TX window. **QB_PT1** is the timing definition of 1'st BPI bus status changing of TX window.

```
#define  QB_PT1                     226
#define  QB_ST1                     226
```

(2) Define the states of BPI bus for each band. The data is shown in the below table:

BPI_BUS	5	4	3	2	1	0	Hex
Function	BANDSW	TXEN	reserved	reserved	reserved	V_logic	
850	0	0	X	X	X	1	01h
GSM	0	0	X	X	X	1	01h
DCS	0	0	X	X	X	1	01h
PCS	0	0	X	X	X	1	01h

So the BPI data can be defined as follow:

```
#define  PDATA_GSM850_PT1          0x01
#define  PDATA_GSM_PT1             0x01
#define  PDATA_DCS_PT1             0x01
#define  PDATA_PCS_PT1             0x01
```

(3) Specify the command of BSI to be sent. Function **L1D_RF_SetSData_ST1** in **m12195.c** be implemented to perform this action. **LNA_SWAP_INDEX[LNA_SET_IDX][l1d_rf.band]** is the table which is used for swap function. **LNA_SET_IDX** is index and used to tell which band has been swap. **SETUP_ST1()** is the first statement in **L1D_RF_SetSData_ST1** which is a macro to hide complicated code to initialize setting 1'st BSI data. Two parameters of **HWRITE_2_SDATA** are control word and data word of BSI data. The control words SCTRL_WARM and SCTRL_PLL are composed of device selection (0) and data bit count (24 bits). The 1st data word **SDATA_WARM_BSEL[CW_WARM_IDX]|SDATA_PLLTX_MASK** is the command to set AD6548 into Warm-up mode which is corresponding with the band and indicate the TX PLL word. **CW_WARM_IDX** is the local variable which indicates the band of operation frequency. The 2nd data word **l1d_rf.RFN_data**, RF N-Counter setting value, is the evaluation result of **L1D_RF_GetTxPLLSetting** called by Layer 1. The global variable **bfetxset.field.iqswap** is used to set I/Q swap for low band.

```
/*AD6548*/ #define  SCTRL_WARM              SCTRL_WORD(0,24)
/*AD6548*/ #define  SCTRL_PLL               SCTRL_WORD(0,24)
/*AD6548*/ #define  SDATA_PLLTX_MASK        0x008000L
/*AD6548*/ #define  LNA_SET_IDX          (((BBTXParameters.bbrx_dcs1800_pcs1900_swap&0x1)<<1)|
                                         (BBTXParameters.bbrx_gsm850_gsm900_swap&0x1))
/*AD6548*/
/*AD6548*/ const unsigned long  SDATA_WARM_BSEL[5]=
/*AD6548*/ { /*Warm*/  /*Band sel*/
/*AD6548*/    0x000184,              /*GSM 450 no support*/
/*AD6548*/    0x006184 | 0x001800,   /*GSM 850*/
/*AD6548*/    0x006184 | 0x000000,   /*GSM 900*/
/*AD6548*/    0x006184 | 0x000800,   /*DCS 1800*/
/*AD6548*/    0x006184 | 0x001000,   /*PCS 1900*/
/*AD6548*/ };
/*AD6548*/ const unsigned short  LNA_SWAP_INDEX[4][5]=
/*AD6548*/ {
/*AD6548*/    { 0, 1, 2, 3, 4 },     /*No Swap*/
/*AD6548*/    { 0, 2, 1, 3, 4 },     /*Low Band Swap*/
/*AD6548*/    { 0, 1, 2, 4, 3 },     /*High Band Swap*/
/*AD6548*/    { 0, 2, 1, 4, 3 },     /*High&Low Band Swap*/
/*AD6548*/ };
/*AD6548*/ void  L1D_RF_SetSData_ST1( void )
/*AD6548*/ {
/*AD6548*/   unsigned short CW_WARM_IDX = LNA_SWAP_INDEX[LNA_SET_IDX][l1d_rf.band];
/*AD6548*/   SETUP_ST1();
/*AD6548*/
/*AD6548*/   HWRITE_2_SDATA(SCTRL_WARM, SDATA_WARM_BSEL[CW_WARM_IDX]|SDATA_PLLTX_MASK,
/*AD6548*/                  SCTRL_PLL,       l1d_rf.RFN_data);
/*AD6548*/
/*AD6548*/   if( l1d_rf.d2c_txiqswap==0 )
/*AD6548*/   {    BFETXSET  bfetxset;
/*AD6548*/        HW_READ_BFETXSET( bfetxset );
/*AD6548*/        bfetxset.field.iqswap = (l1d_rf.band >2 ? 0 : 1)& 0x1;
/*AD6548*/        HW_WRITE_BFETXSET( bfetxset );
/*AD6548*/   }
/*AD6548*/ }
```

To create the transmission window with right operation frequency, evaluating the setting value of operation frequency is needed. Function **L1D_RF_GetTxPLLSetting** in file **m12191.c** needs to be implemented to evaluate the synthesizer setting. And Layer 1 calls this function when programs the RF synthesizer setting value and saves the result into variable **l1d_rf.RFN_data**. This content of this variable is the command to be sent to transceiver chip to lock the operation frequency. The evaluation of AD6548 synthesizer N counter setting is as the following code.

```
/*AD6548*/ void  L1D_RF_GetTxPLLSetting( int rf_band, int arfcn, long *rfN, long *ifN )
/*AD6548*/ {
/*AD6548*/    switch(rf_band)
/*AD6548*/    {
/*AD6548*/       case  FrequencyBand850 :
/*AD6548*/       {
/*AD6548*/          if(arfcn<=185)
/*AD6548*/          {  if(arfcn<=143)
/*AD6548*/             {                                /* ARFCN : 128~143     */
/*AD6548*/                 *rfN = (((arfcn-128)*28+728)<<13) | 0x1106L /*(34L<<7|0x06L)*/;
/*AD6548*/             }
/*AD6548*/             else
/*AD6548*/             {                                /* ARFCN : 144~185     */
/*AD6548*/                 *rfN = (((arfcn-144)*28+6)<<13) | 0x1186L /*(35L<<7|0x06L)*/;
/*AD6548*/             }
/*AD6548*/          }
/*AD6548*/          else
/*AD6548*/          {  if(arfcn<=227)
/*AD6548*/             {                                /* ARFCN : 186~227     */
/*AD6548*/                 *rfN = (((arfcn-186)*28+12)<<13) | 0x1206L /*(36L<<7|0x06L)*/;
/*AD6548*/             }
/*AD6548*/             else
/*AD6548*/             {                              /* ARFCN : 228~251     */
/*AD6548*/                 *rfN = (((arfcn-228)*28+18)<<13) | 0x1286L /*(37L<<7|0x06L)*/;
/*AD6548*/             }
/*AD6548*/          }
/*AD6548*/          break;
/*AD6548*/       }
/*AD6548*/      case  FrequencyBand900 :
/*AD6548*/      {   :
/*AD6548*/          :
/*AD6548*/          break;
/*AD6548*/      }
/*AD6548*/      case  FrequencyBand1800 :
/*AD6548*/      {   :
/*AD6548*/          :
/*AD6548*/          break;
/*AD6548*/      }
/*AD6548*/      case  FrequencyBand1900 :
/*AD6548*/      {       :
/*AD6548*/          :
/*AD6548*/          break;
/*AD6548*/      }
/*AD6548*/      default :
/*AD6548*/      {   :
/*AD6548*/          break;
/*AD6548*/      }
/*AD6548*/   }
/*AD6548*/   *ifN = 0;
/*AD6548*/}
```

2.13.3.2 Set Transceiver in Transmit Mode

116 QB ahead of R0, MT6235 Baseband sends serial command to set transmit mode by BSI bus and after 112 QB, MT6235 Baseband activates control pins by changing BPI bus status. To perform these operations, the timing and data of BPI and BSI bus can be implemented as follow steps:

(1) Modified the timing definition in **l1d_custom_rf.h**. **QB_ST2** is the timing definition of 2'nd BSI command of TX window. **QB_PT2** is the timing definition of 2'nd BPI bus status changing of TX window.

```
#define  QB_ST2                  116
#define  QB_PT2                  4
```

(2) Define the states of BPI bus for each band. The data is shown in the below table:

BPI_BUS	5	4	3	2	1	0	Hex
Function	BANDSW	TXEN	reserved	reserved	reserved	V_logic	
850	0	1	X	X	X	1	11h
GSM	0	1	X	X	X	1	11h
DCS	1	1	X	X	X	1	31h
PCS	1	1	X	X	X	1	31h

So the BPI data can be defined as follow:

```
#define  PDATA_GSM850_PT2         0x11
#define  PDATA_GSM_PT2            0x11
#define  PDATA_DCS_PT2            0x31
#define  PDATA_PCS_PT2            0x31
```

(3) Specify the command of BSI to be sent. Function **L1D_RF_SetSData_ST2** in **m12195.c** should be implemented to perform this action. **LNA_SWAP_INDEX[LNA_SET_IDX][l1d_rf.band]** is the table which is used for swap function. The first statement in **L1D_RF_SetSData_ST2** is **SETUP_ST2()** which is a macro to hide

complicated code to initialize setting 2'nd BSI data. The second statement **HWRITE_2_SDATA** is specified the data to be sent by BSI bus. The parameters of **HWRITE_2_SDATA** are control word and data word of BSI data. The control word **SCTRL_TXON** is composed of device selection (0) and data bit count (24 bits). The data word **SDATA_TXON_MASK** is command to set AD6548 into TX mode.

```
/*AD6548*/ #define  SCTRL_TXON        SCTRL_WORD(0,24)
/*AD6548*/ void  L1D_RF_SetSData_ST2( void )
/*AD6548*/ {
/*AD6548*/   unsigned short CW_WARM_IDX = LNA_SWAP_INDEX[LNA_SET_IDX][l1d_rf.band];
/*AD6548*/   SETUP_ST2();
/*AD6548*/
/*AD6548*/   HWRITE_2_SDATA( SCTRL_TXON, SDATA_WARM_BSEL[CW_WARM_IDX]|SDATA_TXON_MASK,
/*AD6548*/                   SCTRL_RESERVED,    0 );
/*AD6548*/ }
```

2.13.3.3 TX DAC Control

The timing setting for TX frame enable and frame synchronization control of MT6235 are needed. TX frame enable is used to enable or disable TX DAC. TX frame synchronization is used to turn on/off TX SPORT. The timing of enabling TX DAC to turning on TX SPORT named **QB_TX_FENA_2_FSYNC** and the timing of turning off TX SPORT to disabling TX DAC named **QB_TX_FSYNC_2_FENA** are both also needed to be defined in **l1d_custom_rf.h**. In this case, the timing of enabling TX DAC to turning on TX SPORT is 152 QB and the timing of turning off TX SPORT to disabling TX DAC is 20 QB. So **QB_TX_FENA_2_FSYNC** and **QB_TX_FSYNC_2_FENA** can be defined as:

```
#define  QB_TX_FENA_2_FSYNC                 152
#define  QB_TX_FSYNC_2_FENA                 20
```

2.13.3.4 Set APC

99 QB ahead of T0, APC DAC output performs a DC offset to let ramp up more smoothly. The timing of performing this DC offset is defined as **QB_APCDACON** in **l1d_custom_rf.h**. The value of APC DC offset is maintained in APC profile and it is introduced in the latter section.

```
#define  QB_APCDACON                        99
```

22 QB ahead of T0, APC DAC output performs ramping up. The timing of performing this ramp up is defined as **QB_APCON** in **l1d_custom_rf.h**.

```
#define  QB_APCON                           22
```

7 QB behind of T1, APC DAC output performs ramping down. The timing of performing this ramp down is defined as **QB_APCOFF** in **l1d_custom_rf.h**.

```
#define  QB_APCOFF                          7
```

2.13.3.5 Set Transceiver in idle Mode

6 QB behind R1, RF module finishes receiving data and MT6235 set into idle mode. MT6235 Baseband activates control pins by changing BPI bus states, and also sends serial command to set idle mode command by

BSI bus. To perform these operations, the timing and data of BPI and BSI bus can be implemented as follow steps:

(1) Modified the timing definition in **l1d_custom_rf.h**. **QB_SR3** is the timing definition of 3'rd BSI command of RX window. **QB_PR3** is the timing definition of 3'rd BPI bus status changing of RX window.

```
        #define  QB_ST3                           35
        #define  QB_PT3                           35
```

(2) Define the data of BPI bus states for each band. The data is shown in the below table:

BPI_BUS	5	4	3	2	1	0	Hex
Function	BANDSW	TXEN	reserved	reserved	reserved	V_logic	
850	0	0	X	X	X	0	00h
GSM	0	0	X	X	X	0	00h
DCS	0	0	X	X	X	0	00h
PCS	0	0	X	X	X	0	00h

So the BPI data can be defined as follow:

```
        #define  PDATA_GSM850_PT3           0x00
        #define  PDATA_GSM_PT3              0x00
        #define  PDATA_DCS_PT3              0x00
        #define  PDATA_PCS_PT3              0x00
```

(3) Specify the command of BSI to be sent. Function **L1D_RF_SetSData_ST3** in **m12195.c** should be implemented to perform this action. The first statement in **L1D_RF_SetSData_ST3** is **SETUP_ST3()** which is a macro to hide complicated code to initialize setting 3'rd BSI data. The second statement HWRITE_1_SDATA is specified the data to be sent by BSI bus. The parameters of **HWRITE_1_SDATA** are control word and data word of BSI data. The control word **SCTRL_IDLE** is composed of device selection (0) and data bit count (18 bits). The data word **SDATA_IDLE** is the command to set AD6548 into idle mode. **SCTRL_IDLE** and **SDATA_IDLE** are defined in **l1d_rf.h**.

```
/*AD6548*/ #define  SCTRL_IDLE                  SCTRL_WORD(0,18)
/*AD6548*/ #define  SDATA_IDLE                  0x000184
/*AD6548*/ void  L1D_RF_SetSData_ST3( void )
/*AD6548*/ {
/*AD6548*/    SETUP_ST3();
/*AD6548*/    HWRITE_1_SDATA( SCTRL_IDLE, SDATA_IDLE );
/*AD6548*/ }
```

The propagation delay when transmit data from Baseband uplink DAC to antenna should be claimed in file **l1d_custom_rf.c**. This delay depends on different RF module. For AD6548 RF module, the delay is 42 QB.

```
          #define  TX_PROPAGATION_DELAY         42
```

2.13.3.6 Event Activation

If Layer 1 plans to transmit 148-bit data from TDMA timer count 2147 QB to 2739 QB, the above setting applies to create the TX window.

(1) At TDMA timer count is 1921 QB (226 QB ahead of TX SPORT turn on), BPI bus changes its states to 00h (GSM850/GSM/DCS/PCS). At the same time, the serial command of setting operation frequency is sent to AD6548 transceiver.
(2) At TDMA timer count is 1995 QB (152 QB ahead of TX SPORT turn on), TX ADC in MT6235 is enable.

(3) At TDMA timer count is 2031 QB (116 QB ahead of TX SPORT turn on), the serial command to set transmit mode is sent.
(4) At TDMA timer count is 2048 QB (99 QB ahead of TX SPORT turn on), APC DAC output performs a DC offset to let ramp up more smoothly.
(5) At TDMA timer count is 2125 QB (22 QB ahead of TX SPORT turn on), APC DAC output performs ramping up.
(6) At TDMA timer count is 2143 QB (4 QB ahead of TX SPORT turn on), BPI bus changes its states to 11h (GSM850/GSM) or 31h (DCS/PCS).
(7) At TDMA timer count is 2147 QB, TX SPORT in MT6235 is turned on.
(8) At TDMA timer count is 2739 QB, TX SPORT in MT6235 is turned off.
(9) At TDMA timer count is 2746 QB (7 QB behind TX SPORT turning off), APC DAC output performs ramping down.
(10) At TDMA timer count is 2759 QB (20 QB behind TX SPORT turning off), TX DAC is disabled at this time.
(11) At TDMA timer count is 2774 QB (35 QB behind TX SPORT turning off), BPI bus changes its states to 00h and send 3-wire command to set AD6548 into idle mode.

2.14 RF Control Configuration of MT6162 RF Module

For example, the schematic circuit interface between MT6162 RF module and MT6276 chip is shown in the below diagram. Please refer to MediaTek MT6162 RF Module schematic circuit to get more detail information.

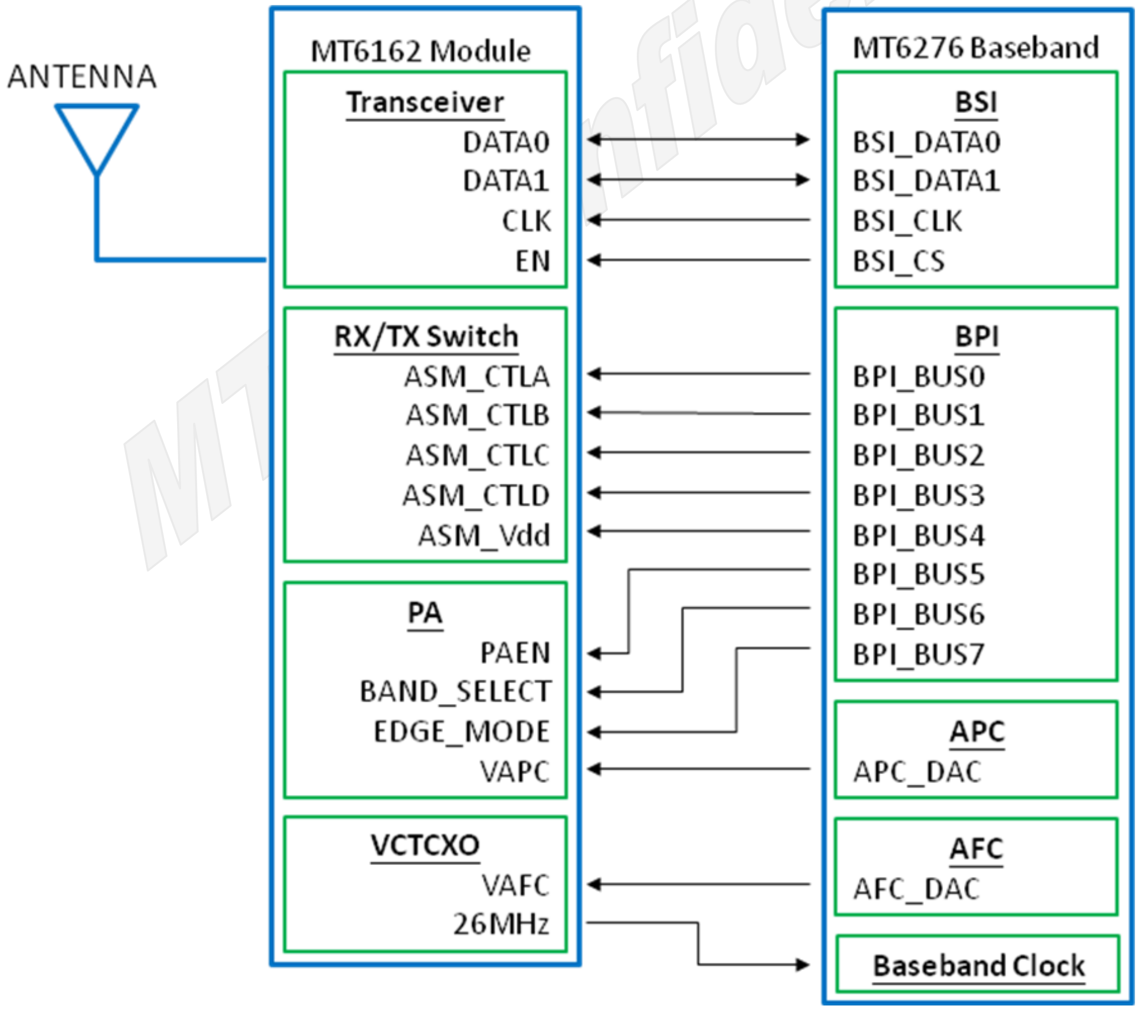


Figure 26 The connection of MT6276 and MT6162 RF Module

BSI_CLK, BSI_DATA0, BSI_DATA1, and BSI_CS pins of MT6276 are directly connected to the serial interface of the MT6162 transceiver. APC_DAC pin of MT6276 is connected to the V_{APC} pin of PA and the EDGE V_{APC} pin of MT6162 of RF module board. The AFC_DAC output is connected to the voltage control pin V_{AFC} of VCTCXO. The generated clock of VCTCXO provides the system clock of MT6276. Logical control pins ASM_CTLA, ASM_CTLB, ASM_CTLC, ASM_CTLD, ASM_Vdd , PAEN, BAND_SELECT, and EDGE_MODE are controlled by the bit 0, 1, 2, 3, 4, 5, 6, and 7 of BPI bus of MT6276. The high/low state of each pin of BPI bus and their timing of changing state can be programmed by MT6276. The RF radio control method is introduced in the next.

2.14.1 Initialization

The RF module can be controlled by the BSI, BPI, APC, AFC units of MT6276 chip. BSI, BPI, APC, AFC units can work in 2 modes. One mode is immediate control mode, and the other is TDMA event driven control mode.

Immediate control mode is that the output signal of BSI, BPI, APC, or AFC unit is generated as soon as setting the corresponding registers in MT6276. After the MT6162 RF module is turned on, the immediate mode can be used to initialize the RF module. The initial states are set as follow:

(1) CTLA (BPI_BUS0) is in the low state
(2) CTLB (BPI_BUS1) is in the low state
(3) CTLC (BPI_BUS2) is in the low state
(4) CTLD (BPI_BUS3) is in the low state
(5) Vdd (BPI_BUS4) is in the low state
(6) PA_EN (BPI_BUS5) is in the low state
(7) PA_BAND_SELECT (BPI_BUS6) is in the low state
(8) PA_EDGE_MODE (BPI_BUS7) is in the low state
(9) Serial commands SDATA_TABLE[] are sending to the MT6162 transceiver one by one.

The initialization of BPI, BSI data can be implemented in **L1D_RF_PowerOn** in **m12196.c** by using the immediate mode:

```
/* for power up */
#define INIT_DDC_RESET_SWPOR      0x0D000001L
#define INIT_DDC_LDOCTR_DIGEN     0x0D120001L
#define INIT_DDC_RESET_RELEASE    0x0D00003EL
#define INIT_TX_LO_L              0x0B42C89FL
#define INIT_TX_LO_H              0x0B532723L
#define INIT_PDCADC_CONFIG        0x0B10EAA8L
#define INIT_CW10                 0x00A41150L
#define INIT_CW44                 0x02CFFFFFL
#define INIT_RX_ABB               0x08504100L
#define INIT_RX_GAIN_TEST         0x09031000L
#define INIT_GMSK_DATA            0x0A704D85L
#define INIT_TTG_TEST             0x09707000L
#define INIT_TX_GAINWRITE         0x0A40FFFCL
#define INIT_TX_BB_FILT_2G        0x0AC20948L
#define INIT_TX_GAINRF_2G         0x0C04A864L
#define INIT_SRX_LF_ADJUST        0x06747028L
#define INIT_SRX_CP_ADJUST        0x06800019L
#define INIT_CW14                 0x00EFFD53L
#define INIT_CW15                 0x00F00D57L
#define INIT_CW16                 0x010FEF11L
#define INIT_CW17                 0x011FE017L
#define INIT_CW18                 0x01205BCFL
#define INIT_CW19                 0x01304E49L
#define INIT_CW20                 0x014F6C59L
#define INIT_CW21                 0x015F8CAFL
#define INIT_CW22                 0x016070C8L
```

```
#define INIT_CW23                   0x0170490EL
#define INIT_CW24                   0x018FC84CL
#define INIT_CW25                   0x019FF2AAL
#define INIT_CW26                   0x01A01232L
#define INIT_CW27                   0x01BFFC6EL
#define INIT_CW28                   0x01CFFFA2L
#define INIT_CW29                   0x01D00000L
#define INIT_CW43                   0x02B9C00CL
#define INIT_CW58                   0x03A39878L
#define INIT_DDC_CLK_CTRL           0x0E20060FL
#define INIT_CW00_E2                0x000402A8L
#define INIT_SRX_VCOCAL_E2          0x06560000L
#define INIT_DCXO_CDAC_MASK         0x0E000000L

#define INIT_REGISTER_NUM_E2     35
unsigned long rfid_readback;
const unsigned long SDATA_INIT_REGISTER_E2[INIT_REGISTER_NUM_E2]=
{
   INIT_TX_LO_L,          INIT_TX_LO_H,          INIT_PDCADC_CONFIG,
   INIT_CW10,             INIT_CW44,             INIT_RX_ABB,
   INIT_SRX_LF_ADJUST,    INIT_SRX_CP_ADJUST,    INIT_TX_BB_FILT_2G,
   INIT_TX_GAINRF_2G,     INIT_SRX_VCOCAL_E2,    INIT_RX_GAIN_TEST,
   INIT_GMSK_DATA,        INIT_TTG_TEST,         INIT_TX_GAINWRITE,
   INIT_CW00_E2,          INIT_CW14,             INIT_CW15,
   INIT_CW16,             INIT_CW17,             INIT_CW18,
   INIT_CW19,             INIT_CW20,             INIT_CW21,
   INIT_CW22,             INIT_CW23,             INIT_CW24,
   INIT_CW25,             INIT_CW26,             INIT_CW27,
   INIT_CW28,             INIT_CW29,             INIT_CW43,
   INIT_CW58,             INIT_DDC_CLK_CTRL,
};

void  L1D_RF_PowerOn( void )
{
…
   /* Turn Immediate Mode on */
   IMM_MODE_BEGIN( IMMMASK_ALL );

   if( l1d_rf.is_init || l1d_rf2.is_wakeup )
   {
      if( l1d_rf.is_init )
      {
         L1D_PMU_VRF18Setting( 1 );
         L1D_PMU_VRF28Setting( 1 );
         IMM_SEND_BSI( SCTRL_IMOD(0,30), INIT_DDC_RESET_SWPOR );      /* DDC_RESET_Reset */
         WAIT_TIME_QB( 300 );                                         /* wait for SW POR */
         IMM_SEND_BSI( SCTRL_IMOD(0,30), INIT_DDC_LDOCTR_DIGEN );     /* DDC_LDOCTR_DigEn */
         WAIT_TIME_QB(55);                                            /* 50us, Wait T_LDO */
         IMM_SEND_BSI( SCTRL_IMOD(0,30), INIT_DDC_RESET_RELEASE );    /* DDC_RESET_Release */
         IMM_RECEIVE_BSI( SCTRL_IMOD(0,10), 0xEF, SCTRL_IMOD(0,20), &(rfid_readback) ); /* read
back RFID */
         l1d_rf2.mt6162_rfid = (char) ( rfid_readback&0xFF );
         if( L1D_RF_Get6162Version() == 1 )
         {  /* Set values into initialization registers for OH E1 */
            for( i=0; i<INIT_REGISTER_NUM_E1; i++ )
            {  IMM_SEND_BSI( SCTRL_IMOD(0,30), SDATA_INIT_REGISTER_E1[i] );  }
         }
         else
         {  /* Set values into initialization registers for OH E2 */
            for( i=0; i<INIT_REGISTER_NUM_E2; i++ )
            {  IMM_SEND_BSI( SCTRL_IMOD(0,30), SDATA_INIT_REGISTER_E2[i] );  }
         }
      }
      else
      {…}
      if( l1d_rf.is_init )
      {
         /* Initial Calibration */
         L1D_RF_BASEBAND_FILTER_CAL();
         L1D_RF_TX_GAINSTEP_CAL();
      }
      else
      {
         /* Write back the LPFCAP[4:0] value to the RX_RCCAL(0x84) register */
         IMM_SEND_BSI( SCTRL_IMOD(0,30), ((l1d_rf2.lpfcap&0x1F)<<10)|RX_RCCAL_WRITEBACK_MASK );
      }
   }
   else
   {…}
   /* Turn Immediate Mode off */
   IMM_MODE_END(  );
   L1D_RF_TXPC_CL_AUXADC_POWERON();
}
```

2.14.2 Create RX Window

TDMA timer is a counter unit that counts 1 per quarter-bit (QB, 0.923us). It always repeats counting from 0 to 4999, so the period of TDMA counter is 5000 QB (1 TDMA frame). If BSI, BPI, AFC, APC are programmed in TDMA event driven mode, the active time can be programmed. When the TDMA counter counts to the programmed time (count), the corresponding unit is triggered to activate.

If it is needed to receive data from air or transmit data to air, Layer1 needs to plan all RF setting and timing to create RX/TX window, control receiving gain, control VCXO, control PA... etc. For this purpose, TDMA event driven mode of BSI, BPI, APC, or AFC units should be used.

For example of creating RX window to receive data, the RF setting and timing of MT6162 RF module can be depicted in the below diagram:

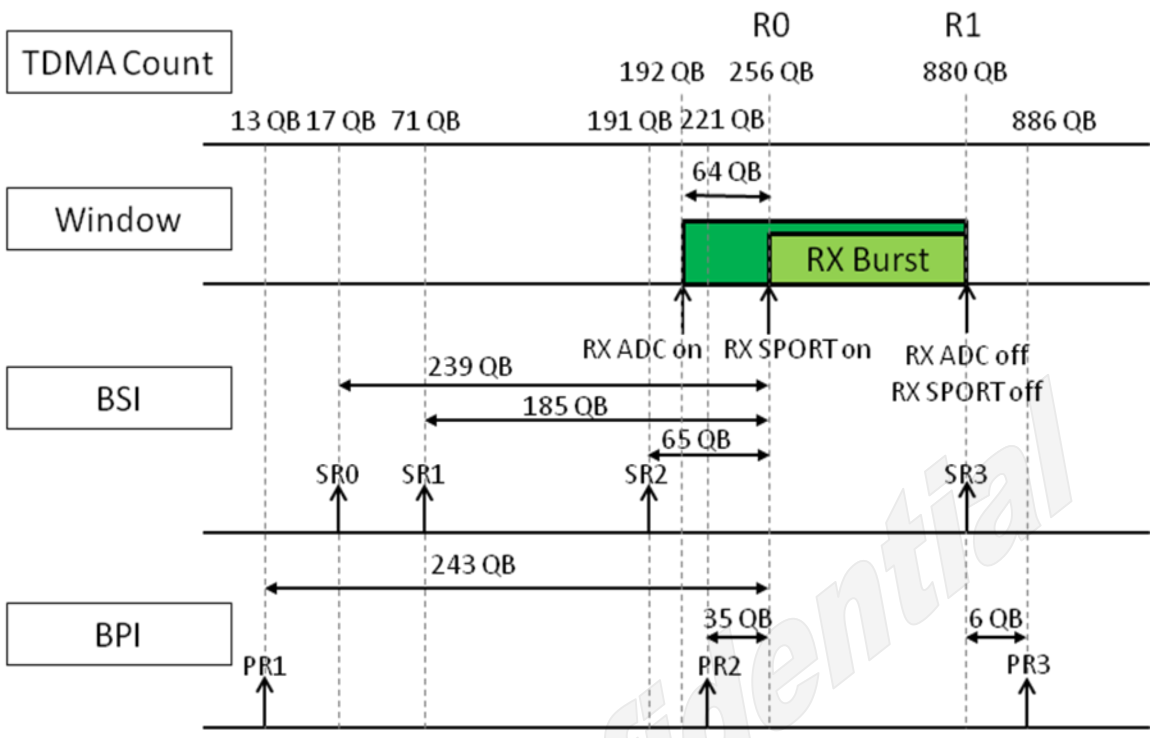


Figure 27 MT6162 Event's timing of RX window

The TDMA timing and data setting are described in 2 parts. One part is before RX_SPORT_on when I/Q data start to be received from the MT6162 transceiver. The other part is after RX_SPORT_off when the data reception is finished. To describe these two parts easily, the timing of turning on RX SPORT is named as **R0**, and the timing of turning off RX SPORT is named as **R1**. In this case, **R0** is at TDMA counter 256 QB, and **R1** is at TDMA counter 880 QB.

The 1st BSI event (SR0) is to turn on the RX LDOs. The happened timing of this event is 239QB before R0. The 2nd BSI event (SR1) is to select the RX band, enable and set the RX synthesizer to the operation frequency and the happened timing of this event is 185QB before R0. The 3rd BSI event (SR2) is to enable the RX mode and set the RX amplifier gain and the RX DC offset compensation value and the happened timing of this event is 65QB before R0. The 4th BSI event (SR3) is to command the transceiver to enter the idle mode and the happened timing of this event is 0QB after R1. The 1st BPI event (PR1) is to control the antenna switch depending on the operation band. The happened timing of this event is 243QB before R0. The 2nd BPI event (PR2) is to keep the antenna switch setting the same and the happened timing of this event is 35QB before R0. The 3rd BPI event is to command the antenna switch to enter the idle mode and the happened timing of this event is 6QB after R1.

2.14.2.1 Turn on RX LDOs

239 QB ahead of R0, the MT6162 transceiver should start to operate, and MT6276 chip sends the serial command to control the MT6162 transceiver. To perform the operation, the timing and data of BSI bus can be implemented as follow steps:

(1) Modify the timing definition in **l1d_custom_rf.h**. **QB_SR0** is the timing definition of the 1st BSI command of RX window.

```
#define  QB_SR0          239
```

(2) Specify the command of BSI to be sent. Function **L1D_RF_SetSData_SR0** in **m12195.c** be implemented to perform this action. **SETUP_SR0()** is the first statement in **L1D_RF_SetSData_SR0** which is a macro to hide complicated code to initialize setting 1'st BSI data. The second statement **HWRITE_4_SDATA** is specified the data to be sent by BSI bus. Two parameters of **HWRITE_4_SDATA** are control word and data word of BSI data. The control words **SCTRL_DDC_RXEN** and **SCTRL_TX_BB_FILT_2G** are composed of device selection (0) and data bit count (30 bits) (**SCTRL_WORD(0,30)**). The 1st data word

data_ddc_en is to turn on the MT6162 RX LDOs. The 2nd data word **SDATA_TX_BB_FILT_2G** is set the MT6162 internal filter for the 2G mode. The 3rd and 4th data words are reserved. Note that if the DXCO is used (IS_MT6162_DCXO_SUPPORT), one additional command **SCTRL_DCXO_CAFC_CTRL** should be sent to set the 13-bit AFC control data **l1d_rf.AFC_data**.

```
#define  SCTRL_DDC_RXEN       SCTRL_WORD(0,30)
#define  SCTRL_TX_BB_FILT_2G  SCTRL_WORD(0,30)
void  L1D_RF_SetSData_SR0( void )
{
   unsigned long data_ddc_en;
   SETUP_SR0();
   data_ddc_en = SDATA_DDC_RXEN;
#if IS_MT6162_DCXO_SUPPORT
   HWRITE_5_SDATA( SCTRL_DDC_RXEN, data_ddc_en,
                   SCTRL_TX_BB_FILT_2G, SDATA_TX_BB_FILT_2G,
                   SCTRL_DCXO_CAFC_CTRL, (l1d_rf.AFC_data&0x1FFF)|SDATA_DCXO_CAFC_CTL_MASK,
                   SCTRL_RESERVED, 0,                           /* reserved */
                   SCTRL_RESERVED, 0 );                         /* reserved */
#else
   HWRITE_4_SDATA( SCTRL_DDC_RXEN, data_ddc_en,
                   SCTRL_TX_BB_FILT_2G, SDATA_TX_BB_FILT_2G,
                   SCTRL_RESERVED, 0,                           /* reserved */
                   SCTRL_RESERVED, 0 );                         /* reserved */
#endif
}
```

2.14.2.2 Set the Synthesizer N-Counter

185 QB ahead of R0, MT6276 sends the BSI command to select the MT6162 RX band and set the RX synthesizer to the operation frequency. To perform these operations, the timing and data of BSI bus can be implemented as follow steps:

(1) Modified the timing definition in **l1d_custom_rf.h**. **QB_SR1** is the timing definition of 2nd BSI command of RX window.

```
#define  QB_SR1           185
```

(2) Specify the command of BSI to be sent. Function **L1D_RF_SetSData_SR1** in **m12195.c** be implemented to perform this action. **SETUP_SR1()** is the first statement in **L1D_RF_SetSData_SR1** which is a macro to hide complicated code to initialize the 1st BSI data. The second statement **HWRITE_3_SDATA** specifies the data to be sent by the BSI bus. Two parameters of HWRITE_3_SDATA are control word and data word of BSI data. The control words **SCTRL_RX_BAND**, **SCTRL_RX_MODE_SRX**, and **SCTRL_SRX_FREQ** are composed of device selection (0) and data bit count (30 bits) (**SCTRL_WORD(0,30)**). The 1st data word **SDATA_RX_BAND** is to set the MT6162 RX band. The 2nd data word **data_rx_mode** is to enable the RX synthesizer. The 3rd data word **rfn_data**, RF N-Counter setting value, is the evaluation result of **L1D_RF_GetRxPLLSetting** called by Layer 1.

```
#define  SCTRL_RX_BAND      SCTRL_WORD(0,30)
#define  SCTRL_RX_MODE_SRX  SCTRL_WORD(0,30)
#define  SCTRL_SRX_FREQ     SCTRL_WORD(0,30)
void  L1D_RF_SetSData_SR1( void )
{
   unsigned long data_rx_mode = SDATA_RX_MODE_SRX;
   unsigned long rfn_data;

   SETUP_SR1();
   rfn_data = l1d_rf.RFN_data;

   HWRITE_3_SDATA( SCTRL_RX_BAND, SDATA_RX_BAND[l1d_rf.band],
                   SCTRL_RX_MODE_SRX, data_rx_mode,
                   SCTRL_SRX_FREQ, rfn_data|SDATA_SRX_FREQ_MASK );
}
```

To create the receiving window with the right operation frequency, evaluating the setting value of operation frequency is needed. Function **L1D_RF_GetRxPLLSetting** in **m12191.c** needs to be implemented to evaluate the synthesizer setting. Layer 1 calls this function when programs the RF synthesizer setting value and saves the result into variable **l1d_rf.RFN_data**. This content of this variable is the command to be sent to transceiver chip to lock the operation frequency. The evaluation of MT6162 synthesizer N counter setting is as the following code:

```
void  L1D_RF_GetRxPLLSetting( int rf_band, int arfcn, long *rfN, long *ifN )
{
   int channelFrequency = 0;
   int synthesizerFrequency = 0;
   int N_INT;
   int N_FRAC;

   switch(rf_band)
   {
      case  FrequencyBand850 :
      {
         channelFrequency = 8242+2*(arfcn-128)+450; /* 824.2+0.2*(arfcn-128)+45 */
         synthesizerFrequency = 4*channelFrequency;
         break;
      }
      case  FrequencyBand900 :
      {
         if(arfcn<=124)
         {  channelFrequency = 8900+2*(arfcn)+450; /* 890+0.2*(arfcn)+45 */ }
         else
         {  channelFrequency = 8900+2*(arfcn-1024)+450; /* 890+0.2*(arfcn-1024)+45 */   }
         synthesizerFrequency = 4*channelFrequency;
         break;
      }
      case  FrequencyBand1800 :
      {
         channelFrequency = 17102+2*(arfcn-512)+950; /* 1710.2+0.2*(arfcn-512)+95 */
         synthesizerFrequency = 2*channelFrequency;
         break;
      }
      case  FrequencyBand1900 :
      {
         channelFrequency = 18502+2*(arfcn-512)+800; /* 1850.2+0.2*(arfcn-512)+80 */
         synthesizerFrequency = 2*channelFrequency;
         break;
      }
   }
   N_INT = synthesizerFrequency/520;
   N_FRAC = ((synthesizerFrequency-520*N_INT)*2080) / 520;
   *rfN = (N_FRAC&0xFFF) | ((N_INT&0x7F)<<12);
}
```

2.14.2.3 Set the Transceiver Gain

65 QB ahead of R0, MT6276 sends the BSI command to enable the RX mode and set the RX amplifier gain and the RX DC offset compensation value. To perform these operations, the timing and data of BSI bus can be implemented as follow steps

(1) Modified the timing definition in **l1d_custom_rf.h**. **QB_SR2** is the timing definition of the 3rd BSI command of RX window.

```
        #define  QB_SR2             65
```

(2) Specify the command of BSI to be sent. Function **L1D_RF_SetSData_SR2** in **m12195.c** should be implemented to perform this action. The first statement in **L1D_RF_SetSData_SR2** is **SETUP_SR2()** which is a macro to hide complicated code to initialize setting 2'nd BSI data. The second statement

HWRITE_5_SDATA is specified the data to be sent by BSI bus. The parameters of **HWRITE_4_SDATA** are control word and data word of BSI data. The control word **SCTRL_RX_MODE_RXEN**, **ctrl_rx_dc_offset**, and **ctrl_rx_agc** are composed of device selection (0) and data bit count (30 bits) (**SCTRL_WORD(0,30)**). The 1st data word **data_rx_mode** is to enable the RX mode. The 2nd data word **data_rx_dc_offset** is to set the RX DC offset compensation value. The 3rd data word **data_rx_agc** is to set the RX gain, which is the evaluation result of **L1D_RF_GetGainSetting** called by Layer 1. The 4th data word is reserved. Note that if there is a continuous RX slot (**IS_CONTINUOUS_RX_SLOT()**), we only need to set **ctrl_rx_dc_offset**, and **ctrl_rx_agc** for the 2nd RX slot.

```
#define  SCTRL_RX_MODE_RXEN  SCTRL_WORD(0,30)
#define  SCTRL_RX_OFFSET     SCTRL_WORD(0,30)
#define  SCTRL_RX_GAIN       SCTRL_WORD(0,30)
void  L1D_RF_SetSData_SR2( void )
{
   unsigned long ctrl_rx_dc_offset = SCTRL_RX_OFFSET;
   unsigned long ctrl_rx_agc = SCTRL_RX_GAIN;
   unsigned long data_rx_mode = SDATA_RX_MODE_RXEN;
   unsigned long data_rx_dc_offset;
   unsigned long data_rx_agc;

   SETUP_SR2();
   /* normal RX */
   data_rx_dc_offset = l1d_rf2.rx_dc_offset|SDATA_RX_Offset_MASK;
   data_rx_agc = l1d_rf.AGC_data|SDATA_RX_Gain_MASK;
   if( IS_CONTINUOUS_RX_SLOT() )
   {
      HWRITE_4_SDATA( ctrl_rx_dc_offset, data_rx_dc_offset, /* set RX DC offset */
                      ctrl_rx_agc, data_rx_agc,             /* set RX gain */
                      SCTRL_RESERVED, 0,                    /* reserved */
                      SCTRL_RESERVED, 0 );                  /* reserved */
   }
   else
   {
      HWRITE_4_SDATA( SCTRL_RX_MODE_RXEN, data_rx_mode,     /* enable RX circuits */
                      ctrl_rx_dc_offset, data_rx_dc_offset, /* set RX DC offset */
                      ctrl_rx_agc, data_rx_agc,             /* set RX gain */
                      SCTRL_RESERVED, 0 );                  /* reserved */
   }
}
```

To create the receiving window with suitable gain of receiving amplifier, evaluating the setting value of request gain is needed. Function **L1D_RF_GetGainSetting** in file **m12192.c** needs to be implemented to evaluate the transceiver gain setting. Layer 1 calls this function when programs the gain setting value and saves the result into variable **l1d_rf.AGC_data**. This content of this variable is the command to be sent to transceiver chip to set the requested received amplifier gain. The evaluation of MT6162 receiving amplifier gain setting is as the following code:

```
int   L1D_RF_GetGainSetting( int rf_band, int arfcn, int request_gain, long *gain_setting )
{
   int   max_gain, min_gain, real_gain;
   const int*    d32ptr;
   const sL1DAGCDATA*  agcptr;
   long  setting;
   int   bit_no;

   /* evaluate the range of available gain */
   d32ptr  = (int*)&(GAIN_RANGE_TABLE[rf_band]);
   max_gain = *d32ptr++;
   min_gain = *d32ptr;
   /* clipping the request gain to the avialable gain */
   if( request_gain>=max_gain )
   {  request_gain = max_gain;  }
   else  if( request_gain<=min_gain )
   {  request_gain = min_gain;  }

   /* evaluate the real gain setting */
   agcptr = AGC_TABLE[rf_band];
   while( request_gain < agcptr->pos_gain )
   {  agcptr++;
   }
   {  bit_no    = BIT_NO( request_gain, agcptr->A, GC_B );
      real_gain = REAL_GAIN( bit_no, agcptr->A, GC_B );
      setting   = agcptr->setting | (bit_no&0xF);
   }
   *gain_setting = setting;
   l1d_rf2.rx_dc_offset = 0;
   return( real_gain );
}
```

2.14.2.4 Set the Transceiver in the Idle Mode

0 QB behind R1, RF module finishes receiving data and MT6276 sets it into the idle mode. To perform this operation, the timing and data of BSI bus can be implemented as follow steps:

(1) Modified the timing definition in **l1d_rf.h**. **QB_SR3** is the timing definition of the 4th BSI command of RX window.

```
#define  QB_SR3          0
```

(2) Specify the command of BSI to be sent. Function **L1D_RF_SetSData_SR3** in **m12195.c** should be implemented to perform this action. The first statement in **L1D_RF_SetSData_SR3** is **SETUP_SR3()** which is a macro to hide complicated code to initialize setting 3'rd BSI data. The second statement **HWRITE_3_SDATA** is specified the data to be sent by BSI bus. The parameters of **HWRITE_3_SDATA** are control word and data word of BSI data. The control word **SCTRL_RX_MODE_OFF** and **SCTRL_DDC_RXOFF** are composed of device selection (0) and data bit count (30 bits) (**SCTRL_WORD(0,30)**). The 1st data word **SDATA_RX_MODE_OFF** is to disable the MT6162 RX mode. The 2nd data word is to turn off the RX LDOs. The 3rd data word is reserved

```
#define  SCTRL_RX_MODE_OFF  SCTRL_WORD(0,30)
#define  SCTRL_DDC_RXOFF    SCTRL_WORD(0,30)
void  L1D_RF_SetSData_SR3( void )
{
   unsigned long data_ddc_off = SDATA_DDC_RXOff;         /* set LDOs to the idle+dig state */

   SETUP_SR3();
   HWRITE_3_SDATA( SCTRL_RX_MODE_OFF, SDATA_RX_MODE_OFF, /* turn off RX circuits */
                   SCTRL_DDC_RXOFF, data_ddc_off,        /* set LDOs */
                   SCTRL_RESERVED, 0 );                  /* reserved */
}
```

2.14.2.5 The BPI Control

MT6276 activates the control pins by changing BPI bus states to control the RF module. The 1st BPI event (PR1) is to control the antenna switch depending on the operation band. The 2nd BPI event (PR2) is to keep the antenna switch setting the same. The 3rd BPI event (PR3) is to command the antenna switch to enter the idle mode. To perform these operations, the timing and data of BPI bus can be implemented as follow steps:

(1) Modified the timing definition in **l1d_custom_rf.h**. **QB_PR1**, **QB_PR2**, and **QB_PR3** are the timing definition of the 1st, 2nd, and 3rd BPI bus status changing of RX window respectively.

```
#define  QB_PR1          243
#define  QB_PR2          35
```

```
        #define   QB_PR3            6
```

(2) Define the states of BPI bus for each band. The data is shown in the below table:

PR1:

BPI_BUS	9	8	7	6	5	4	3	2	1	0	Hex
Function	Error detection pin	Not used	EDGE mode	Band select	PA enable	Vdd	CTLD	CTLC	CTLB	CTLA	
GSM850	1	X	0	0	0	1	0	0	1	0	212h
GSM	1	X	0	0	0	1	0	1	0	1	215h
DCS	1	X	0	0	0	1	0	1	1	0	216h
PCS	1	X	0	0	0	1	0	1	0	0	214h

PR2:

BPI_BUS	9	8	7	6	5	4	3	2	1	0	Hex
Function	Error detection pin	Not used	EDGE mode	Band select	PA enable	Vdd	CTLD	CTLC	CTLB	CTLA	
GSM850	1	X	0	0	0	1	0	0	1	0	212h
GSM	1	X	0	0	0	1	0	1	0	1	215h
DCS	1	X	0	0	0	1	0	1	1	0	216h
PCS	1	X	0	0	0	1	0	1	0	0	214h

PR3:

BPI_BUS	9	8	7	6	5	4	3	2	1	0	Hex
Function	Error detection pin	Not used	EDGE mode	Band select	PA enable	Vdd	CTLD	CTLC	CTLB	CTLA	
GSM850	0	X	0	0	0	0	0	0	0	0	000h
GSM	0	X	0	0	0	0	0	0	0	0	000h
DCS	0	X	0	0	0	0	0	0	0	0	000h
PCS	0	X	0	0	0	0	0	0	0	0	000h

So the BPI data can be defined as follow:

```
        #define   PDATA_GSM850_PR1   0x212
        #define   PDATA_GSM850_PR2   0x212
        #define   PDATA_GSM850_PR3   0x000
        #define   PDATA_GSM_PR1      0x215
        #define   PDATA_GSM_PR2      0x215
        #define   PDATA_GSM_PR3      0x000
        #define   PDATA_DCS_PR1      0x216
        #define   PDATA_DCS_PR2      0x216
        #define   PDATA_DCS_PR3      0x000
        #define   PDATA_PCS_PR1      0x214
        #define   PDATA_PCS_PR2      0x214
        #define   PDATA_PCS_PR3      0x000
```

2.14.2.6 RX ADC Control

In addition to set the timing and data of BPI and BSI bus of the MT6276 baseband to create a RX window, the timing setting for RX frame enable and frame synchronization control of MT6276 are needed. RX frame enable is used to enable or disable RX ADC. RX frame synchronization is used to turn on/off RX SPORT. The timing of enabling RX ADC to turning on RX SPORT named **QB_RX_FENA_2_FSYNC** and the timing of turning off RX SPORT to disabling RX ADC named **QB_RX_FSYNC_2_FENA** are both also needed to be defined in **l1d_custom_rf.h**. In this case, the timing of enabling RX ADC to turning on RX SPORT is recommend to be 64 QB and the timing of turning off RX SPORT to disabling RX ADC is 0 QB. So **QB_RX_FENA_2_FSYNC** and **QB_RX_FSYNC_2_FENA** can be defined as:

```
        #define   QB_RX_FENA_2_FSYNC     64
        #define   QB_RX_FSYNC_2_FENA     0
```

2.14.2.7 Events Activation

If Layer1 plans to receive 156-bit data from TDMA timer count 256 QB to 880 QB, the following setting is to create the RX window.

(1) At TDMA timer count 13 QB (243 QB before RX SPORT is turned on), BPI bus changes its states to 212h(GSM850), 215h(GSM), 216h(DCS), or 214h (PCS).

(2) At TDMA timer count 17 QB (239 QB before RX SPORT is turned on), the 1st serial command is sent to the MT6162 transceiver.

(3) At TDMA timer count 71 QB (185 QB before RX SPORT is turned on), the 2nd serial command is sent to the MT6162 transceiver.

(4) At TDMA timer count 191 QB (65 QB before RX SPORT is turned on), the 3rd serial command is sent to the MT6162 transceiver

(5) At TDMA timer count 191 QB (64 QB before RX SPORT is turned on), RX ADC in MT6276 is enabled.

(6) At TDMA timer count 221 QB (35 QB before RX SPORT is turned on), BPI bus keeps its states on 212h(GSM850), 215h(GSM), 216h(DCS), or 214h (PCS).

(7) At TDMA timer count 256 QB, RX SPORT in MT6276 is turned on, and I and Q data begin to be sampled.

(8) At TDMA timer count 880 QB, RX SPORT in MT6229 is turned off and RX ADC is also disabled. I and Q data stop sampling.

(9) At TDMA timer count 880 QB (at RX SPORT turning off), the 3rd serial command is sent to the MT6162 transceiver.

(10) At TDMA timer count 886 QB (6 QB after RX SPORT is turned on), BPI bus changes its states to 000h.

2.14.3 Create TX Window

Another example is creating TX window to transmit data. The RF setting and timing of the MT6162 RF module can be depicted in the below diagram:

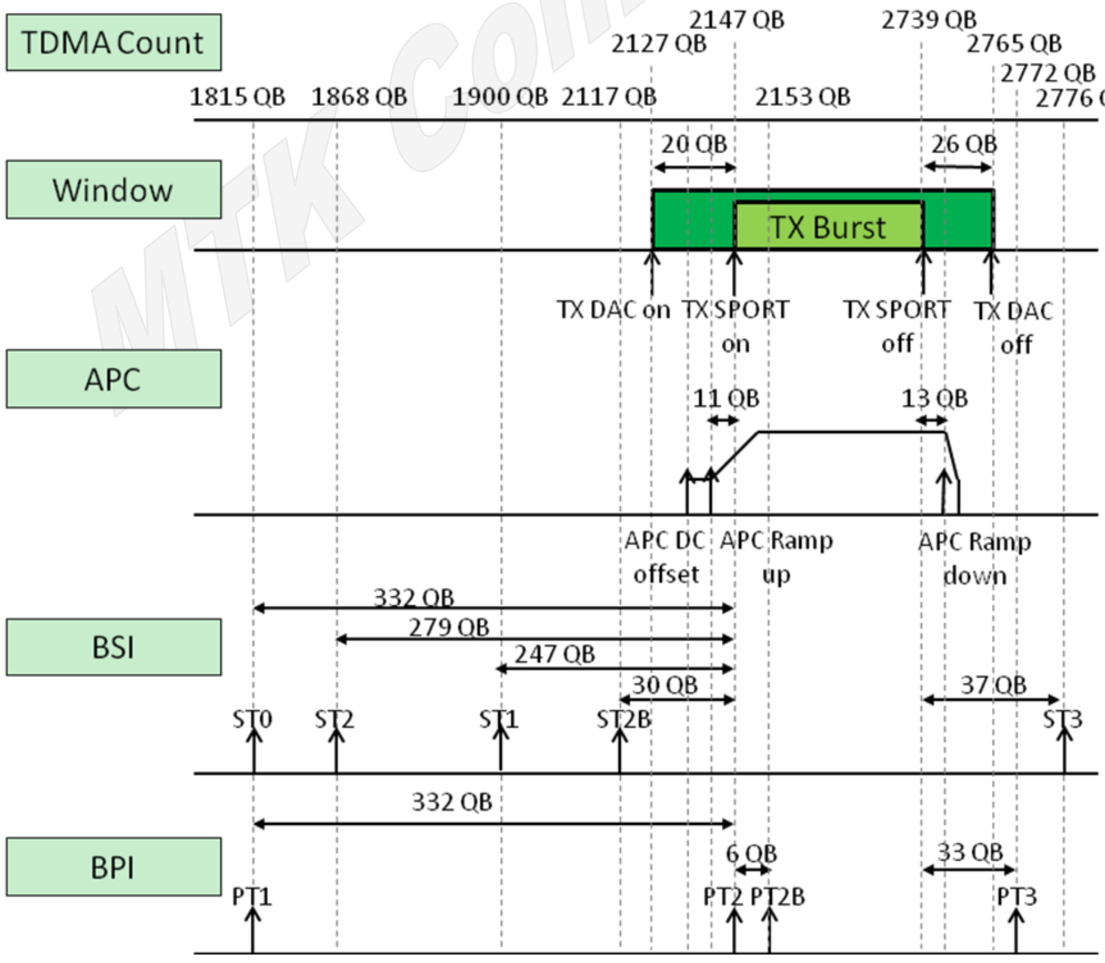


Figure 28 MT6162 Event's timing of TX window

The TDMA timing and data setting are described in 2 parts. One part is that before turning on TX SPORT to transmit I and Q data to the MT6162 transceiver. The other part is after turning off TX SPORT to finish transmit I and Q data. To describe these two parts easily, the timing of turning on TX SPORT is taken as one base named **T0**. And the timing of turning off TX SPORT is taken as one base named **T1**. In this case, T0 is at TDMA counter 2147 QB and T1 is at TDMA counter 2739 QB.

The 1st BSI event (ST0) is to turn on the TX LDOs. The happened timing of this event is 332QB before T0. The 2nd BSI event (ST2) is to turn on the internal TXVCO LDOs and the happened timing of this event is 279QB before T0. The 3rd BSI event (ST1) is to select the TX band, enable and set the TX synthesizer to the operation frequency and the happened timing of this event is 247QB before T0. The 4th BSI event (ST2B) is to select the GMSK/EPSK mode and to set the RF TX gain, and the happened timing of this event is 30QB before T0. The 5th BSI event (ST3) is to command the transceiver to enter the idle mode and the happened timing of this event is 37QB after T1. The 1st BPI event (PT1) is to start the error detection function for baseband and the happened timing of this event is 332QB before T0. The 2nd BPI event (PT2) is to enable PA and select the PA band and the happened timing of this event is 0QB before T0. The 3rd BPI event (PT2B) is to control the antenna switch depending on the operation band and the happened timing of this event is 6QB after T0. The 4th BPI event (PT3) is to command the antenna switch and PA to enter the idle mode and the happened timing of this event is 33QB after T1.

2.14.3.1 Turn on TX LDOs

332 QB ahead of T0, the MT6162 transceiver should start to operate, and MT6276 chip sends the serial command to control the MT6162 transceiver. To perform the operation, the timing and data of BSI bus can be implemented as follow steps:

(1) Modify the timing definition in **l1d_custom_rf.h**. **QB_ST0** is the timing definition of the 1st BSI command of TX window.

```
#define  QB_ST0          332
```

(2) Specify the command of BSI to be sent. Function **L1D_RF_SetSData_ST0** in **m12195.c** be implemented to perform this action. **SETUP_ST0()** is the first statement in **L1D_RF_SetSData_ST0** which is a macro to hide complicated code to initialize setting the 1st BSI data. The second statement **HWRITE_4_SDATA** is specified the data to be sent by BSI bus. The parameters of **HWRITE_4_SDATA** are control word and data word of BSI data. The control words **SCTRL_TX_CONFIG** and **SCTRL_DDC_TXEN** are composed of device selection (0) and data bit count (30 bits) (**SCTRL_WORD(0,30)**). The 1st data word **data_tx_config** is set the MT6162 internal TX configuration depending on the low band (Band850 and Band900) or the high band (Band1800 and Band1900). The 2nd data word **data_ddc_en** is to turn on the MT6162 TX LDOs. The 3rd data word **data_cw00** is to set the DFM default value for the 2G mode. The 4th data word **data_cw10** is to set the DFM default value for the 2G mode. Note that if the DXCO is used (IS_MT6162_DCXO_SUPPORT), one additional command **SCTRL_DCXO_CAFC_CTRL** should be sent to set the 13-bit AFC control data **l1d_rf.AFC_data**.

```
#define  SCTRL_TX_CONFIG  SCTRL_WORD(0,30)
#define  SCTRL_DDC_TXEN   SCTRL_WORD(0,30)
#define  SCTRL_CW00       SCTRL_WORD(0,30)
#define  SCTRL_CW10       SCTRL_WORD(0,30)
void  L1D_RF_SetSData_ST0( void )
{
   unsigned long data_tx_config = (l1d_rf.band<=FrequencyBand900) ? (SDATA_TX_CONFIG_LB) :
(SDATA_TX_CONFIG_HB);
   unsigned long data_ddc_en = SDATA_DDC_TXEN;
   unsigned long ctrl_cw00 = SCTRL_RESERVED;
   unsigned long ctrl_cw10 = SCTRL_RESERVED;
   unsigned long data_cw00 = 0;
   unsigned long data_cw10 = 0;

   SETUP_ST0();
   if( L1D_RF_Get6162Version()==1 )
   {  /* for OH E1, do nothing */  }
   else
   {  /* for OH E2 */
      ctrl_cw00 = SCTRL_CW00;
      ctrl_cw10 = SCTRL_CW00;
      data_cw00 = SDATA_CW00_2G;
      data_cw10 = (l1d_rf2.is_isotpol==1) ? SDATA_CW10_ISOTPOL_1 : SDATA_CW10_ISOTPOL_0;
      l1d_rf2.is_isotpol = 0;
   }

#if IS_MT6162_DCXO_SUPPORT
   HWRITE_5_SDATA( SCTRL_TX_CONFIG, data_tx_config,
                   SCTRL_DDC_TXEN, data_ddc_en,
                   SCTRL_DCXO_CAFC_CTRL, (l1d_rf.AFC_data&0x1FFF)|SDATA_DCXO_CAFC_CTL_MASK,
                   ctrl_cw00, data_cw00,
                   ctrl_cw00, data_cw10 );
#else
   HWRITE_4_SDATA( SCTRL_TX_CONFIG, data_tx_config,
                   SCTRL_DDC_TXEN, data_ddc_en,
                   ctrl_cw00, data_cw00,
                   ctrl_cw00, data_cw10 );
#endif
}
```

2.14.3.2 Set the Synthesizer N-Counter

279 QB ahead of T0, MT6276 sends the BSI command to turn on the internal TXVCO LDOs. 247QB ahead of T0, MT6276 sends another BSI command to select the MT6162 TX band and set the TX synthesizer to the operation frequency. To perform these operations, the timing and data of BSI bus can be implemented as follow steps:

(1) Modified the timing definition in **l1d_custom_rf.h**. **QB_ST2 / QB_ST1** are the timing definition of the 2nd / 3rd BSI command of TX window.

```
#define  QB_ST2          279
#define  QB_ST1          241
```

(2) Specify the command of BSI to be sent. Function **L1D_RF_SetSData_ST2** in **m12195.c** be implemented to perform this action. **SETUP_ST2()** is the first statement in **L1D_RF_SetSData_ST2** which is a macro to hide complicated code to initialize the 2nd BSI data. The second statement **HWRITE_4_SDATA** specifies the data to be sent by the BSI bus. The parameters of **HWRITE_4_SDATA** are control word and data word of BSI data. The control words **SCTRL_TXMODE_CTRL**, **SCTRL_CW02_DFM_PLL**, **SCTRL_INIT_TIME**, and **SCTRL_INIT_GAINRF** are composed of device selection (0) and data bit count (30 bits) (**SCTRL_WORD(0,30)**). The 1st data word **SDATA_TXMODE_CTRL_2G** is to set TX to the 2G mode. The 2nd data word **data_cw02** is to turn on the internal TXVCO LDOs, select the TX band, and set the synthesizer (**IFN_data**, RF N-Counter setting value, is the evaluation result of **L1D_RF_GetTxPLLSetting** called by Layer 1, which will be described in details later). The 3rd data word **SDATA_INIT_TIME** is to set the power detector measurement time for the 2G mode. The 4th data word **data_gainrf** is to set the TX gain default parameter for the 2G mode.

```
#define  SCTRL_TXMODE_CTRL   SCTRL_WORD(0,30)
#define  SCTRL_CW02_DFM_PLL  SCTRL_WORD(0,30)
#define  SCTRL_INIT_TIME     SCTRL_WORD(0,30)
#define  SCTRL_INIT_GAINRF   SCTRL_WORD(0,30)
void  L1D_RF_SetSData_ST2( void )
{
   unsigned long ctrl_cw02 = SCTRL_CW02_DFM_PLL;
   unsigned long data_cw02;
   unsigned long data_gainrf;
   SETUP_ST2();
   data_cw02                                                                     =
(l1d_rf.IFN_data&0x1FFF)|SDATA_CW02_DFM_PLL_MASK[l1d_rf.band]|SDATA_CW02_Standby_MASK;
   data_gainrf = SDATA_INIT_GAINRF;

   HWRITE_4_SDATA( SCTRL_TXMODE_CTRL, SDATA_TXMODE_CTRL_2G,
                   ctrl_cw02, data_cw02,
                   SCTRL_INIT_TIME, SDATA_INIT_TIME,
                   SCTRL_INIT_GAINRF, data_gainrf );
}
```

(3) Specify the command of BSI to be sent. Function **L1D_RF_SetSData_ST1** in **m12195.c** be implemented to perform this action. **SETUP_ST1()** is the first statement in **L1D_RF_SetSData_ST1** which is a macro to hide complicated code to initialize the 3rd BSI data. The second statement **HWRITE_3_SDATA** specifies the data to be sent by the BSI bus. The parameters of **HWRITE_3_SDATA** are control word and data word of BSI data. The control words **SCTRL_TX_BAND_BSEL**, **SCTRL_CW01_DFM_PLL**, and **SCTRL_INIT_DET_SETUP** are composed of device selection (0) and data bit count (30 bits) (**SCTRL_WORD(0,30)**). The 1st data word **tx_band** is select the TX band. The 2nd data word **rfn_data**, RF N-Counter setting value, is the evaluation result of **L1D_RF_GetTxPLLSetting** called by Layer 1, which will be described in details later. The 3rd data word **SDATA_INIT_DET_SETUP** is to set the power detector default parameter for the 2G mode.

```
#define  SCTRL_TX_BAND_BSEL    SCTRL_WORD(0,30)
#define  SCTRL_CW01_DFM_PLL    SCTRL_WORD(0,30)
#define  SCTRL_INIT_DET_SETUP  SCTRL_WORD(0,30)
void  L1D_RF_SetSData_ST1( void )
{
   unsigned long tx_band;
   unsigned long rfn_data;

   SETUP_ST1();
   tx_band  = SDATA_TX_BAND_BSEL[l1d_rf.band];
   rfn_data = l1d_rf.RFN_data;

   HWRITE_3_SDATA( SCTRL_TX_BAND_BSEL, tx_band,
                   SCTRL_CW01_DFM_PLL, rfn_data|SDATA_CW01_DFM_PLL_MASK,
                   SCTRL_INIT_DET_SETUP, SDATA_INIT_DET_SETUP );
}
```

To create the transmitting window with the right operation frequency, evaluating the setting value of operation frequency is needed. Function **L1D_RF_GetTxPLLSetting** in **m12191.c** needs to be implemented to evaluate the synthesizer setting. Layer 1 calls this function when programs the RF synthesizer setting value and saves the result into variable **l1d_rf.RFN_data** and **l1d_rf.IFN_data**. The content of these variables are the command to be sent to transceiver chip to lock the operation frequency. The evaluation of MT6162 synthesizer N counter setting is as the following code:

```
void  L1D_RF_GetTxPLLSetting( int rf_band, int arfcn, long *rfN, long *ifN )
{
   int channelFrequency = 0;
   int synthesizerFrequency = 0;
   int N_INT;
   int N_FRAC;

   l1d_rf2.arfcn = arfcn;  //for L1D_RF_SetTxGainWrite()

   switch(rf_band)
   {
      case  FrequencyBand850 :
      {
         channelFrequency = 8242+2*(arfcn-128); /* 824.2+0.2*(arfcn-128) */
         synthesizerFrequency = 4*channelFrequency;
         break;
      }
      case  FrequencyBand900 :
      {  if(arfcn<=124)
         {  channelFrequency = 8900+2*(arfcn); /* 890+0.2*(arfcn) */  }
         else
         {  channelFrequency = 8900+2*(arfcn-1024); /* 890+0.2*(arfcn-1024) */  }
         synthesizerFrequency = 4*channelFrequency;
         break;
      }
      case  FrequencyBand1800 :
      {
         channelFrequency = 17102+2*(arfcn-512); /* 1710.2+0.2*(arfcn-512) */
         synthesizerFrequency = 2*channelFrequency;
         break;
      }
      case  FrequencyBand1900 :
      {
         channelFrequency = 18502+2*(arfcn-512); /* 1850.2+0.2*(arfcn-512) */
         synthesizerFrequency = 2*channelFrequency;
         break;
      }
   }
   N_INT = synthesizerFrequency/260;
   N_FRAC = ((synthesizerFrequency-260*N_INT)<<21) / 65;
   *rfN = (N_FRAC&0x3FF) | ((N_INT&0xFF)<<10);
   *ifN = (N_FRAC>>10)&0x1FFF;
```

2.14.3.3 Set the Transceiver Gain

30 QB ahead of T0, MT6276 sends the BSI command to select the GMSK/EPSK mode and to set the RF TX gain. To perform these operations, the timing and data of BSI bus can be implemented as follow steps

(1) Modified the timing definition in **l1d_custom_rf.h**. **QB_ST2B** is the timing definition of the 4th BSI command of TX window.

```
#define  QB_ST2B          30
```

(2) Specify the command of BSI to be sent. Function **L1D_RF_SetSData_ST2B** in **m12195.c** should be implemented to perform this action. The first statement in **L1D_RF_SetSData_ST2B** is **SETUP_ST2B_ST2M()** which is a macro to hide complicated code to initialize setting the 4th BSI data. The second statement **HWRITE_4_SDATA** is specified the data to be sent by BSI bus. The parameters of **HWRITE_4_SDATA** are control word and data word of BSI data. The control word **SCTRL_CW43_DFM**, **SCTRL_TXMODE_CTRL**, and **SCTRL_TX_GAINWRITE** are composed of device selection (0) and data bit count (30 bits) (**SCTRL_WORD(0,30)**). The 1st data word **data_cw43_dfm_edge** or **data_cw43_dfm_gsm** is to select the EPSK or GMSK mode. The 2nd data word **data_txmode_ctrl_edge** or **data_txmode_ctrl_gsm** is to enable the TX mode depending on EPSK or GMSK. The 3rd data word **data_tx_gain_set** is to trigger the power detector to do the measurement and set the TX gain, which is the evaluation result of **L1D_RF_SetTxGainWrite** called by Layer 1.

```
#define  SCTRL_CW43_DFM      SCTRL_WORD(0,30)
#define  SCTRL_TXMODE_CTRL   SCTRL_WORD(0,30)
#define  SCTRL_TX_GAINWRITE  SCTRL_WORD(0,30)
#define  SCTRL_CW02_DFM_PLL  SCTRL_WORD(0,30)
void  L1D_RF_SetSData_ST2B( void )
{
   unsigned long  ctrl_cw02             = (l1d_rf2.is_integer==1) ? SCTRL_CW02_DFM_PLL :
SCTRL_RESERVED;
   unsigned long  data_tx_gain_set;
   unsigned long  data_txmode_ctrl_gsm  = SDATA_TXMODE_CTRL_GSM;
   unsigned long  data_txmode_ctrl_edge = SDATA_TXMODE_CTRL_EDGE;
   unsigned long  data_cw43_dfm_edge    = (L1D_RF_Get6162Version() == 1) ?
(SDATA_CW43_DFM_EDGE_E1) : (SDATA_CW43_DFM_EDGE_E2);
   unsigned long  data_cw43_dfm_gsm     = (L1D_RF_Get6162Version() == 1) ?
(SDATA_CW43_DFM_GSM_E1)  : (SDATA_CW43_DFM_GSM_E2);
   unsigned long  data_cw02             = (l1d_rf2.is_integer==1) ?
((l1d_rf.IFN_data&0x1FFF)|SDATA_CW02_DFM_PLL_MASK[l1d_rf.band]|SDATA_CW02_TXMODE_MASK) : 0;

   SETUP_ST2B_ST2M();
   data_tx_gain_set  = L1D_RF_SetTxGainWrite( l1d_rf.tx_mod_type2, l1d_rf.cur_slot,
l1d_rf.tx_power[l1d_rf.cur_slot], (FrequencyBand)l1d_rf.band );
   L1D_RF_SetEPSKTxIQ( (signed short)l1d_rf.cur_slot, data_tx_gain_set );
   data_tx_gain_set |= SDATA_TX_MEASURE_TRIGGER;

   if( (l1d_rf.tx_mod_type2>>l1d_rf.cur_slot)&0x1 )
   {  /* 8PSK */
      HWRITE_4_SDATA( SCTRL_CW43_DFM, data_cw43_dfm_edge,
                      SCTRL_TXMODE_CTRL, data_txmode_ctrl_edge,
                      SCTRL_TX_GAINWRITE, data_tx_gain_set,
                      ctrl_cw02, data_cw02 );
   }
   else
   {  /* GMSK */
      HWRITE_4_SDATA( SCTRL_CW43_DFM, data_cw43_dfm_gsm,
                      SCTRL_TXMODE_CTRL, data_txmode_ctrl_gsm,
                      SCTRL_TX_GAINWRITE, data_tx_gain_set,
                      ctrl_cw02, data_cw02 );
   }
}
```

To create the transmitting window with the suitable TX gain, evaluating the setting value of request gain is needed. Function **L1D_RF_SetTxGainWrite** in file **m12190.c** needs to be implemented to evaluate the transceiver gain setting. This content of this variable is the command to be sent to transceiver chip to set the suitable TX gain. Note that there are total three gain values to be evaluated, gain_rf, grain_bb, and gain_det. The gain_rf and gain_bb values are for the TX amplifier gain in the EPSK mode, and the gain_det value is for the detector gain in both GMSK and EPSK modes. The evaluation of MT6162 TX gain setting is as the following code:

```
unsigned long L1D_RF_SetTxGainWrite( unsigned short mod_type, char slot_idx, signed short
tx_power, FrequencyBand rf_band )
{
   signed short     gain_det;
   signed short     gain_rf = 0;
   signed short     gain_bb = 0;
   signed short     gain_cpl;
   signed short     power_dbm = tx_power;
   signed short     power_dbm_oh = tx_power;
   signed short     lowest_dbm = LOWEST_TX_POWER[(int)rf_band];
   signed short     power_idx, rollback_gain;
   unsigned short   lb_or_hb = (rf_band<=FrequencyBand900) ? 0 : 1;
   unsigned short   weighted_PA_gain;
   unsigned short   n;
   unsigned long    tx_gain_write;
   const sRAMPDATA *band;
   static   const   sARFCN_SECTION*  weight;

   if( power_dbm > (lowest_dbm+2*15) )
   {  power_dbm = lowest_dbm+2*15;  }
   else if( power_dbm < lowest_dbm )
   {  power_dbm = lowest_dbm;  }
   #if IS_EPSK_TX_SUPPORT
   if( (mod_type>>slot_idx)&0x1 )
   {  /* 8psk */
      band = RampData_EPSK[(int)rf_band];
      if( lb_or_hb )
         power_dbm_oh = ( power_dbm > 26 ) ? 26 : power_dbm; //High band
      else
         power_dbm_oh = ( power_dbm > 27 ) ? 27 : power_dbm; //Low band
      weight    =    WeightARFCN_BinarySearch((l1d_rf2.arfcn&0x3FF),    l1d_rf.band    ,    band-
>arfcn_weight, 1);
      if( weight )
      {  n = (power_dbm > weight->mid_level) ? weight->hi_weight : weight->low_weight;  }
      else // no entry, not found
      {  n = WEIGHT(1.0);  }
      power_idx = (power_dbm-lowest_dbm)/2;
      weighted_PA_gain = (unsigned short)( ( band->power[power_idx]*n )>>14 );

     /*TX power rollback start*/
      rollback_gain = L1D_RF_PowerRollback( power_dbm, l1d_rf.band, n, 1);
      weighted_PA_gain -= rollback_gain;
      /*TX power rollback end*/

      gain_cpl = (rf_band<=FrequencyBand900) ? GAIN_CPL_LB : GAIN_CPL_HB;
      gain_rf    =    (signed    short)    (RFSpecialCoef.rx.mt6162_gain_rf.gain_rf_table[rf_band-
1].map[power_idx]);
      gain_bb = (signed short)( 1023-weighted_PA_gain );
      /* the range of gain_bb is 0~1023 (10bits) */
      if( gain_bb < 0 ) gain_bb = 0;
      else if( gain_bb > 1023 ) gain_bb = 1023;
      gain_det = ( (GAIN_PBBOFS)-(gain_cpl)-((power_dbm_oh<<8)-POWER_BB) )>>8;
   }
   else
   #endif
   {  /* GMSK: Only for calculate gain_det */
      power_dbm_oh = power_dbm;
      gain_cpl = (rf_band<=FrequencyBand900) ? GAIN_CPL_LB : GAIN_CPL_HB;
      gain_rf   = 0xF;
      gain_bb   = 0x3FF;
      gain_det = ( (GAIN_PBBOFS)+(POWER_BB)-(power_dbm_oh<<8)+(-gain_cpl) )>>8;
   }

   gain_det = ( gain_det < 3 ) ? (0) : (gain_det/3-1);
   gain_det = ( gain_det >= 10 ) ? (10) : (gain_det);

   tx_gain_write = ( (gain_det&0xF)<<16 ) | ( (gain_rf&0xF)<<12 ) | ( (gain_bb&0x3FF)<<2 );
   return tx_gain_write;
}
```

2.14.3.4 Set the Transceiver in the Idle Mode

37 QB behind T1, RF module finishes transmitting data and MT6276 sets it into the idle mode. To perform this operation, the timing and data of BSI bus can be implemented as follow steps:

(1) Modified the timing definition in **l1d_rf.h**. **QB_ST3** is the timing definition of the 5th BSI command of TX window.

```
        #define  QB_ST3          37
```

(2) Specify the command of BSI to be sent. Function **L1D_RF_SetSData_ST3** in **m12195.c** should be implemented to perform this action. The first statement in **L1D_RF_SetSData_ST3** is **SETUP_ST3()** which is a macro to hide complicated code to initialize setting the 5th BSI data. The second statement **HWRITE_3_SDATA** is specified the data to be sent by BSI bus. The parameters of **HWRITE_3_SDATA**

are control word and data word of BSI data. The control word **SCTRL_CW02_DFM_PLL_OFF**, **SCTRL_TXMODE_CTRL_OFF**, and **SCTRL_DDC_TXOff** are composed of device selection (0) and data bit count (30 bits) (**SCTRL_WORD(0,30)**). The 1st data word **data_cw02** is to power down DFM. The 2nd data word SDATA_TXMODE_CTRL_OFF is to disable the MT6162 RX mode. The 3rd data word **data_ddc_off** is to turn off the TX LDOs.

```
#define  SCTRL_CW02_DFM_PLL_OFF  SCTRL_WORD(0,30)
#define  SCTRL_TXMODE_CTRL_OFF   SCTRL_WORD(0,30)
#define  SCTRL_DDC_TXOff         SCTRL_WORD(0,30)
void  L1D_RF_SetSData_ST3( void )
{
   unsigned long  data_ddc_off;
   unsigned long  data_cw02;

   SETUP_ST3();
   data_cw02 = (l1d_rf.IFN_data&0x1FFF)|SDATA_CW02_DFM_PLL_OFF[l1d_rf.band];
   data_ddc_off = SDATA_DDC_TXOff;

   HWRITE_3_SDATA( SCTRL_CW02_DFM_PLL_OFF, data_cw02,
                   SCTRL_TXMODE_CTRL_OFF, SDATA_TXMODE_CTRL_OFF,
                   SCTRL_DDC_TXOff, data_ddc_off );
}
```

2.14.3.5 The BPI Control

MT6276 activates the control pins by changing BPI bus states to control the RF module. The 1st BPI event (PT1) is to start the error detection function for baseband. The 2nd BPI event (PT2) is to enable PA and select the PA band. The 3rd BPI event (PT2B) is to control the antenna switch depending on the operation band. The 4th BPI event (PT3) is to command the antenna switch and PA to enter the idle mode. To perform these operations, the timing and data of BPI bus can be implemented as follow steps:

(1) Modified the timing definition in **l1d_custom_rf.h**. **QB_PT1**, **QB_PT2**, **QB_PT2B**, and **QB_PT3** are the timing definition of the 1st, 2nd, 3rd and 4th BPI bus status changing of TX window respectively.

```
#define  QB_PT1          314
#define  QB_PT2          0
#define  QB_PT2B         -6
#define  QB_PT3          33
```

(2) Define the states of BPI bus for each band. The data is shown in the below table:

PT1:

BPI_BUS	9	8	7	6	5	4	3	2	1	0	Hex
Function	Error detection pin	Not used	EDGE mode	Band select	PA enable	Vdd	CTLD	CTLC	CTLB	CTLA	
GSM850	1	X	0	0	0	1	0	0	0	0	210h
GSM	1	X	0	0	0	1	0	0	0	0	210h
DCS	1	X	0	0	0	1	0	0	0	0	210h
PCS	1	X	0	0	0	1	0	0	0	0	210h

PT2:

BPI_BUS	9	8	7	6	5	4	3	2	1	0	Hex
Function	Error detection pin	Not used	EDGE mode	Band select	PA enable	Vdd	CTLD	CTLC	CTLB	CTLA	
GSM850	1	X	0	0	1	1	0	0	0	0	230h
GSM	1	X	0	0	1	1	0	0	0	0	230h
DCS	1	X	0	1	1	1	0	0	0	0	270h
PCS	1	X	0	1	1	1	0	0	0	0	270h

PT2B:

BPI_BUS	9	8	7	6	5	4	3	2	1	0	Hex

Function	Error detection pin	Not used	EDGE mode	Band select	PA enable	Vdd	CTLD	CTLC	CTLB	CTLA	
GSM850	1	X	0	0	1	1	0	0	1	1	233h
GSM	1	X	0	0	1	1	0	0	1	1	233h
DCS	1	X	0	1	1	1	0	0	0	1	271h
PCS	1	X	0	1	1	1	0	0	0	1	271h

PT3:

BPI_BUS	9	8	7	6	5	4	3	2	1	0	Hex
Function	Error detection pin	Not used	EDGE mode	Band select	PA enable	Vdd	CTLD	CTLC	CTLB	CTLA	
GSM850	0	X	0	0	0	0	0	0	0	0	000h
GSM	0	X	0	0	0	0	0	0	0	0	000h
DCS	0	X	0	0	0	0	0	0	0	0	000h
PCS	0	X	0	0	0	0	0	0	0	0	000h

So the BPI data can be defined as follow:

```
#define  PDATA_GSM850_PT1    0x210
#define  PDATA_GSM850_PT2    0x230
#define  PDATA_GSM850_PT2B   0x233
#define  PDATA_GSM850_PT3    0x000
#define  PDATA_GSM_PT1       0x210
#define  PDATA_GSM_PT2       0x230
#define  PDATA_GSM_PT2B      0x233
#define  PDATA_GSM_PT3       0x000
#define  PDATA_DCS_PT1       0x210
#define  PDATA_DCS_PT2       0x270
#define  PDATA_DCS_PT2B      0x271
#define  PDATA_DCS_PT3       0x000
#define  PDATA_PCS_PT1       0x210
#define  PDATA_PCS_PT2       0x270
#define  PDATA_PCS_PT2B      0x271
#define  PDATA_PCS_PT3       0x000
```

2.14.3.6 TX DAC Control

In addition to set the timing and data of BPI and BSI bus of the MT6276 baseband to create a TX window, the timing setting for TX frame enable and frame synchronization control of MT6276 are needed. TX frame enable is used to enable or disable TX DAC. TX frame synchronization is used to turn on/off TX SPORT. The timing of enabling TX DAC to turning on TX SPORT named **QB_TX_FENA_2_FSYNC** and the timing of turning off TX SPORT to disabling TX DAC named **QB_TX_FSYNC_2_FENA** are both also needed to be defined in **l1d_custom_rf.h**. In this case, the timing of enabling TX DAC to turning on TX SPORT is recommend to be 20 QB and the timing of turning off TX SPORT to disabling TX DAC is 26 QB. So **QB_TX_FENA_2_FSYNC** and **QB_TX_FSYNC_2_FENA** can be defined as:

```
#define  QB_TX_FENA_2_FSYNC    20
#define  QB_TX_FSYNC_2_FENA    26
```

2.14.3.7 Set APC

0 QB ahead of T0, APC DAC output performs a DC offset to let ramp up more smoothly. The timing of performing this DC offset is defined as **QB_APCDACON** in **l1d_custom_rf.h**. The value of APC DC offset is maintained in APC profile and it will be described in details in the latter section.

```
#define  QB_APCDACON   0
```

11 QB ahead of T0, APC DAC output performs ramping up. The timing of performing this ramping up is defined as **QB_APCON** in **l1d_custom_rf.h**.

```
#define  QB_APCON      11
```

13 QB behind T1, APC DAC output performs ramping down. The timing of performing this ramping down is defined as **QB_APCOFF** in **l1d_custom_rf.h**.

```
#define  QB_APCOFF    13
```

2.14.3.8 Events Activation

If Layer 1 plans to transmit 148-bit data from TDMA timer count 2147 QB to 2739 QB, the following setting is to create the TX window.

(1) At TDMA timer count 1815 QB (332 QB before TX SPORT is turned on), the 1st serial command is sent.
(2) At TDMA timer count 1815 QB (332 QB before TX SPORT is turned on), BPI bus changes its states to 200h(GSM850), 200h(GSM), 200h(DCS), or 200h (PCS).
(3) At TDMA timer count 1868 QB (279 QB before TX SPORT is turned on), the 2nd serial command is sent.
(4) At TDMA timer count 1900 QB (247 QB before TX SPORT is turned on), the 3rd serial command is sent.
(5) At TDMA timer count 2117 QB (30 QB before TX SPORT is turned on), the 4th serial command is sent.
(6) At TDMA timer count 2127 QB (20 QB before TX SPORT is turned on), TX DAC is enabled.
(7) At TDMA timer count 2136 QB (11 QB before TX SPORT is turned on), APC DAC output performs ramping up.
(8) At TDMA timer count 2147 QB (0 QB before TX SPORT is turned on), BPI bus changes its states to 230h(GSM850), 230h(GSM), 270h(DCS), or 270h (PCS).
(9) At TDMA timer count 2147 QB, TX SPORT in MT6276 is turned on.
(10) At TDMA timer count 2153 QB (6 QB after TX SPORT is turned on), BPI bus changes its states to 233h(GSM850), 233h(GSM), 271h(DCS), or 271h (PCS).
(11) At TDMA timer count 2739 QB, TX SPORT in MT6276 is turned off.
(12) At TDMA timer count 2752 QB (13 QB after TX SPORT is turned off), APC DAC output performs ramping down.
(13) At TDMA timer count 2765 QB (26 QB after TX SPORT is turned off), TX DAC is disabled at this time.
(14) At TDMA timer count 2772 QB (33 QB after TX SPORT is turned off), BPI bus changes its states to 000h.
(15) At TDMA timer count 2776 QB (37 QB after TX SPORT is turned on), the 5th serial command is sent.

2.14.4 TX window modulation type setup

For the TX window, two type of the modulation, GMSK/EPSK can be selected according to the EDGE mode pin of the BPI bus. For the GMSK modulation, the pin should be low, and for the EPSK modulation, the pin should be high. The correlated settings could be found in file **l1d_custom_rf.h**. The settings should be correlated to the EDGE mode pin of the BPI bus.

```
#define  PDATA_GMSK              0x000
#define  PDATA_EPSK              0x080
```

The usage of these two definitions could be found in file **m12195.c**.

```
void  L1D_RF_SetPData_PT( void )
{  APBADDR      _reg;
   const short *_d16p;

   _reg  = PDATA_REG_TABLE[ l1d_rf.cwin_idx ];
   _d16p = PDATA_TABLE[ l1d_rf.band ][ RF_TX ];
   if ( (l1d_rf.tx_mod_type) & (1<<l1d_rf.modidx) )
   {
      HW_WRITE( _reg, ((*_d16p)|(PDATA_8PSK)) );   _reg+=2;   _d16p++;
      HW_WRITE( _reg, ((*_d16p)|(PDATA_8PSK)) );   _reg+=2;   _d16p++;
      HW_WRITE( _reg, ((*_d16p)|(PDATA_8PSK)) );   _reg+=2;   _d16p++;
      HW_WRITE( _reg, (*_d16p) );//PT3 is end, doesn't need to | PDATA_8PSK
      /* PT3A */
      _d16p++;
      _reg = PDATA_REG_TABLE2[ l1d_rf.cwin_idx ];
      HW_WRITE( _reg, (*_d16p) );
   }
   else
   {
      HW_WRITE( _reg, ((*_d16p)|(PDATA_GMSK)) );   _reg+=2;   _d16p++;
      HW_WRITE( _reg, ((*_d16p)|(PDATA_GMSK)) );   _reg+=2;   _d16p++;
      HW_WRITE( _reg, ((*_d16p)|(PDATA_GMSK)) );   _reg+=2;   _d16p++;
      HW_WRITE( _reg, (*_d16p) );//PT3 is end, doesn't need to | PDATA_GMSK
      /* PT3A */
      _d16p++;
      _reg = PDATA_REG_TABLE2[ l1d_rf.cwin_idx ];
      HW_WRITE( _reg, (*_d16p) );
   }
}
```

The function **L1D_RF_SetPData_PT** is used to setup the BPI event data of the PT1, PT2, PT2B and PT3. Except for PT3, BPI event data setup in file **l1d_custom_rf.h** should append the PDATA_GMSK or the PDATA_8PSK according to the modulation type of the TX window.

For the inter-slot modulation changing of the TX window, the BPI event named PT2M is used. If the change is from the GMSK mode to the EPSK mode, the EPSK mode pin should be changed from the low status to the high status, as the result, the following definitions in file **l1d_custom_rf.h** according to the operating band is defined.

```
#define  PDATA_GSM850_PT2M1_G8  0x233
#define  PDATA_GSM850_PT2M2_G8  0x2B3
#define  PDATA_GSM_PT2M1_G8     0x233
#define  PDATA_GSM_PT2M2_G8     0x2B3
#define  PDATA_DCS_PT2M1_G8     0x271
#define  PDATA_DCS_PT2M2_G8     0x2F1
#define  PDATA_PCS_PT2M1_G8     0x271
#define  PDATA_PCS_PT2M2_G8     0x2F1
```

Note that the team **G8** is used to identify the change is from GMSK to EPSK (8PSK). In fact, the definition is decided by the **PDATA_PT2B | PDATA_8PSK**.

On the other hand, if the change is from EPSK to GPSK, the EPSK mode pin should be changed from the high status to the low status, as the result, the following definitions in file **l1d_custom_rf.h** according to the operating band is defined.

```
#define  PDATA_GSM850_PT2M1_8G  0x200
#define  PDATA_GSM850_PT2M2_8G  0x233
#define  PDATA_GSM_PT2M1_8G     0x200
#define  PDATA_GSM_PT2M2_8G     0x233
#define  PDATA_DCS_PT2M1_8G     0x240
#define  PDATA_DCS_PT2M2_8G     0x271
#define  PDATA_PCS_PT2M1_8G     0x240
#define  PDATA_PCS_PT2M2_8G     0x271
```

Note that the team **8G** is used to identify the change is from the EPSK to the GPSK. In fact, the definition is decided by the **PDATA_PT2B | PDATA_GPSK**.

The event timing of PT2M could be found in file **l1d_custom_rf.h**. 0QB ahead of the SPORT ON of the 2nd TX window, the modulation type must be changed. The definitions are as followings.

```
#define  QB_PT2M1_G8         20
#define  QB_PT2M2_G8         -2
#define  QB_PT2M1_8G         15
#define  QB_PT2M2_8G         2
```

In addition to the BPI setting, 3QB ahead of the SPORT ON of the 2nd TX window, MT6276 sends the BSI command to set the mode again if the operation mode is changed between GMSK and EPSK. To perform the operation, the timing and data of BSI bus can be implemented as follow steps:

(1) Modify the timing definition in **l1d_custom_rf.h**. **QB_ST2M_G8** and **QB_ST2M_8G** are the timing definition of the BSI command of changing modes.

```
#define  QB_ST2M_G8       3
#define  QB_ST2M_8G       3
```

(2) Specify the command of BSI to be sent. Function **L1D_RF_SetSData_ST2M** in **m12195.c** should be implemented to perform this action. The first statement in **L1D_RF_SetSData_ST2M** is **SETUP_ST2B_ST2M()** which is a macro to hide complicated code to initialize setting the BSI data. The second statement **HWRITE_4_SDATA** is specified the data to be sent by BSI bus. The parameters of **HWRITE_4_SDATA** are control word and data word of BSI data. The control word **SCTRL_CW43_DFM**, **SCTRL_TXMODE_CTRL**, and **SCTRL_TX_GAINWRITE** are composed of device selection (0) and data bit count (30 bits) (**SCTRL_WORD(0,30)**). If the change is from GMSK to EPSK, the 1st data word **data_cw43_dfm_edge** is to select the EPSK mode. The 2nd data word **data_txmode_ctrl_edge** is to enable the TX EPSK mode. The 3rd data word **data_tx_gain_set** is to trigger the power detector to do the measurement and set the TX gain, which is the evaluation result of **L1D_RF_SetTxGainWrite** called by Layer 1. The 4th data word is reserved. If the change is from EPSK to GMSK, the 1st data word **data_cw43_dfm_gsm** is to select the GMSK mode. The 2nd data word **data_txmode_ctrl_gsm** is to enable the TX GMSK mode. The 3rd data word **data_tx_gain_set** is to trigger the power detector to do the measurement. The 4th data word is reserved. Note that if there is no change between the 2 continuous TX slots, ONLY one data word **data_tx_gain_set** should be sent to trigger the power detector to do the measurement and set the TX gain.

```
#define  SCTRL_CW43_DFM      SCTRL_WORD(0,30)
#define  SCTRL_TXMODE_CTRL   SCTRL_WORD(0,30)
#define  SCTRL_TX_GAINWRITE  SCTRL_WORD(0,30)
void  L1D_RF_SetSData_ST2M( void )
{
   unsigned long  data_tx_gain_set;
   unsigned long  data_txmode_ctrl_gsm  = SDATA_TXMODE_CTRL_GSM;
   unsigned long  data_txmode_ctrl_edge = SDATA_TXMODE_CTRL_EDGE;
   unsigned long  data_cw43_dfm_edge    = (L1D_RF_Get6162Version() == 1) ?
(SDATA_CW43_DFM_EDGE_E1) : (SDATA_CW43_DFM_EDGE_E2);;
   unsigned long  data_cw43_dfm_gsm     = (L1D_RF_Get6162Version() == 1) ?
(SDATA_CW43_DFM_GSM_E1)  : (SDATA_CW43_DFM_GSM_E2);

   SETUP_ST2B_ST2M();
   data_tx_gain_set = L1D_RF_SetTxGainWrite( l1d_rf.tx_mod_type2, l1d_rf.cur_slot,
l1d_rf.tx_power[l1d_rf.cur_slot], (FrequencyBand)l1d_rf.band );
   L1D_RF_SetEPSKTxIQ( (signed short)l1d_rf.cur_slot, data_tx_gain_set );
   data_tx_gain_set |= SDATA_TX_MEASURE_TRIGGER;

   /* now change */
   if( (l1d_rf.change)&(1<<l1d_rf.cur_slot) )
   {  /* change g->8 */
      if( (l1d_rf.tx_mod_type2>>l1d_rf.cur_slot)&0x1 )
      {
         HWRITE_4_SDATA( SCTRL_CW43_DFM, data_cw43_dfm_edge,
                         SCTRL_TXMODE_CTRL, data_txmode_ctrl_edge,
                         SCTRL_TX_GAINWRITE, data_tx_gain_set,
                         SCTRL_RESERVED, 0 );
      }
      /* change 8->g */
      else
      {
         HWRITE_4_SDATA( SCTRL_CW43_DFM, data_cw43_dfm_gsm,
                           SCTRL_TXMODE_CTRL, data_txmode_ctrl_gsm,
                           SCTRL_TX_GAINWRITE, data_tx_gain_set,
                           SCTRL_RESERVED, 0 );
      }
   }
   else
   {
      HWRITE_4_SDATA( SCTRL_TX_GAINWRITE, data_tx_gain_set,
                      SCTRL_RESERVED, 0,
                      SCTRL_RESERVED, 0,
                      SCTRL_RESERVED, 0 );
   }
}
```

3 The Limitation of TDMA Timing Setting

3.1 TDMA Timing Limitation in GSM mode

This section describes the TDMA timing limitation in GSM mode. The limitation in GPRS mode is described in the next section. The timing setting cannot exceed the limitation described in this section if the MS supports GSM mode.

3.1.1 BSI TDMA Timing Limitation in GSM mode

3.1.1.1 Limitation of SR1

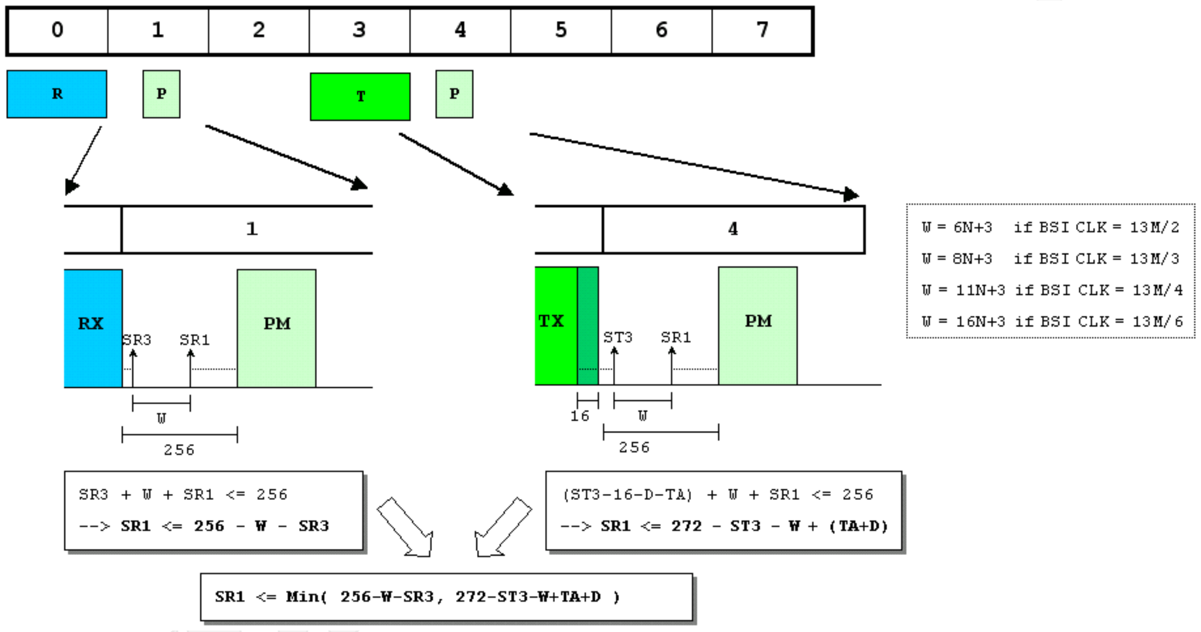

0
1
2
3
4
5
6
7
R
P
T
P
1
4
RX
SR3
SR1
PM
TX
ST3
SR1
PM
W
256
16
W
256
W = 6N+3 if BSI CLK = 13M/2
W = 8N+3 if BSI CLK = 13M/3
W = 11N+3 if BSI CLK = 13M/4
W = 16N+3 if BSI CLK = 13M/6
SR3 + W + SR1 <= 256
--> SR1 <= 256 - W - SR3
(ST3-16-D-TA) + W + SR1 <= 256
--> SR1 <= 272 - ST3 - W + (TA+D)
SR1 <= Min(256-W-SR3, 272-ST3-W+TA+D)

Figure 29 The TDMA timing limitation of SRi in GSM mode

Consider the above figure. The figure depicts the worse case of the SR1 limitation in GSM mode. Layer 1 may arrange one PM after NB RX window or TX window.

If PM behinds NB RX window, the timing equation can be written as

SR3 + W + SR1 <= 256 .... (3.1.1.1A)

Where W is the timing of 3-wire command sent from BB chip to RF transceiver. MT62xx can be set 4 3-wire clock rate: 13M/2, 13M/3, 13M/4 Hz, 13M/6 Hz. If N is the count of commands sent at SR3, the W can be evaluate as

W = 6N+3	if BSI_CLK is 13M/2 Hz
W = 8N+3	if BSI_CLK is 13M/3 Hz
W = 11N+3	if BSI_CLK is 13M/4 Hz
W = 16N+3	if BSI_CLK is 13M/6 Hz

By the equation (3.1.1.1A), SR1 can be gotten as

SR1 <= 256 - W - SR3 .... (3.1.1.1B)

If PM behinds NB TX window, the timing equation can be written as

(ST3 -16 - D - TA) + W + SR1 <= 256... (3.1.1.1C)

Where D is the TX propagation delay. Half guard period is 16 QB. TA is the timing advance and the range is 0 to 256.

If TA is 0, SR1 can be gotten by the equation (3.1.1.1C) :

SR1 <= 272 - ST3 - W + D .... (3.1.1.1D)

The limitation of SR1 is the minimum value of (3.1.1.1B) and (3.1.1.1D).

SR1 <= min(256 - W - SR3, 272 - ST3 - W + D) .... (3.1.1.1E)

3.1.1.2 Limitation of SR2

SR2 shall be set after the commands of SR1 are sent.

SR1 -SR2 >= W ....(3.1.1.2A)

If N is the count of commands sent at SR1, the W can be evaluate as

W = 6N+3	if BSI_CLK is 13M/2 Hz
W = 8N+3	if BSI_CLK is 13M/3 Hz
W = 11N+3	if BSI_CLK is 13M/4 Hz
W = 16N+3	if BSI_CLK is 13M/6 Hz

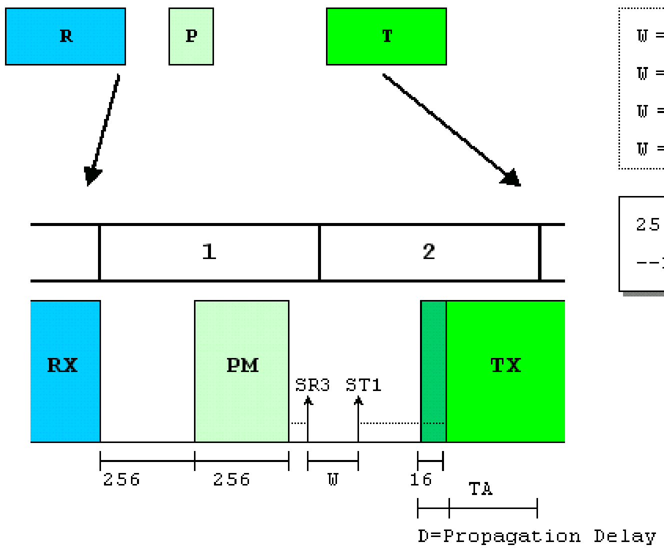
And SR2 can be gotten as

SR2 <= SR1 - W..(3.1.1.2B)

3.1.1.3 Limitation of SR3

SR3 does not have the limitation but follows the equation (3.1.1.1A).

3.1.1.4 Limitation of ST1



```
W = 6N+3   if BSI CLK = 13M/2
W = 8N+3   if BSI CLK = 13M/3
W = 11N+3  if BSI CLK = 13M/4
W = 16N+3  if BSI CLK = 13M/6
```

```
256 + 256 + SR3 + W + (ST1-16)+(D+TA) <= 625+625
--> ST1  <=  754  -  SR3  -  W  -  (D+TA)
```

Figure 30 The TDMA timing limitation of STi in GSM mode

Consider the above figure. The figure depicts the worse case of the ST1 limitation in GSM mode. Layer 1 may arrange one PM between NB RX and TX window. The timing equation can be written as

256 + 256 + SR3 + W + (ST1-16) + (D+TA) <= 625 + 625...................................... (3.1.1.4A)

If N is the count of commands sent at SR3, the W can be evaluate as

W = 6N+3	if BSI_CLK is 13M/2 Hz
W = 8N+3	if BSI_CLK is 13M/3 Hz
W = 11N+3	if BSI_CLK is 13M/4 Hz
W = 16N+3	if BSI_CLK is 13M/6 Hz

D is TX propagation delay. TA is the timing advance and the value can be 0 to 256. If TA is 256, ST1 can be gotten by the equation (3.1.1.4A) :

ST1 <= 498 - SR3 - W - D ....(3.1.1.4B)

3.1.1.5 Limitation of ST2

ST2 shall be set after the commands of ST1 are sent.

ST1 -ST2 >= W ....(3.1.1.5A)

If N is the count of commands sent at ST1, the W can be evaluate as

W = 6N+3	if BSI_CLK is 13M/2 Hz
W = 8N+3	if BSI_CLK is 13M/3 Hz
W = 11N+3	if BSI_CLK is 13M/4 Hz
W = 16N+3	if BSI_CLK is 13M/6 Hz

And ST2 can be gotten as

ST2 <= ST1 - W...(3.1.1.5B)

3.1.1.6 Limitation of ST3

ST3 does not have the limitation but follows the equation (3.1.1.4A).

3.1.2 BPI TDMA Timing Limitation in GSM mode

3.1.2.1 Limitation of PR1

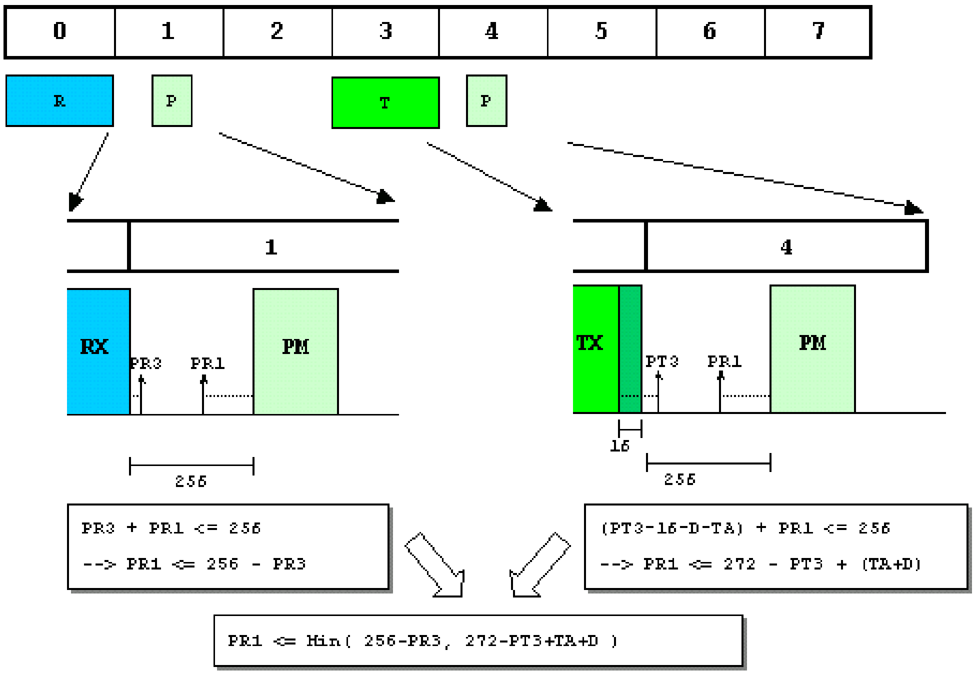

0
1
2
3
4
5
6
7
R
P
T
P
1
4
RX
PR3
PR1
PM
TX
PT3
PR1
PM
16
256
256
PR3 + PR1 <= 256
--> PR1 <= 256 - PR3
(PT3-16-D-TA) + PR1 <= 256
--> PR1 <= 272 - PT3 + (TA+D)
PR1 <= Min(256-PR3, 272-PT3+TA+D)

Figure 31 The TDMA timing limitation of PRi in GSM mode

Consider the above figure. The figure depicts the worse case of the PR1 limitation in GSM mode. Layer 1 may arrange one PM after NB RX window or TX window.

If PM behinds NB RX window, the timing equation can be written as

PR3 + PR1 <= 256 .... (3.1.2.1A)

PR1 can be gotten as

PR1 <= 256 - PR3 .... (3.1.2.1B)

If PM behinds NB TX window, the timing equation can be written as

(PT3 -16 - D - TA) + PR1 <= 256... (3.1.2.1C)

Where D is the TX propagation delay. TA is the timing advance and the value can be 0 to 256.

If TA is 0, PR1 can be gotten as

PR1 <= 272 - PT3 + D .... (3.1.2.1D)

The limitation of PR1 is the minimum value of (3.1.2.1B) and (3.1.2.1D).

PR1 <= min(256 - PR3, 272 - PT3 + D) .... (3.1.2.1E)

3.1.2.2 Limitation of PR2

PR2 shall be set after the commands of PR1 are sent.

PR1 -PR2 > 0 ....(3.1.2.2A)

And PR2 can be gotten as

PR2 < PR1..(3.1.2.2B)

3.1.2.3 Limitation of PR3

PR3 does not have the limitation but follows the equation (3.1.2.1A).

3.1.2.4 Limitation of PT1

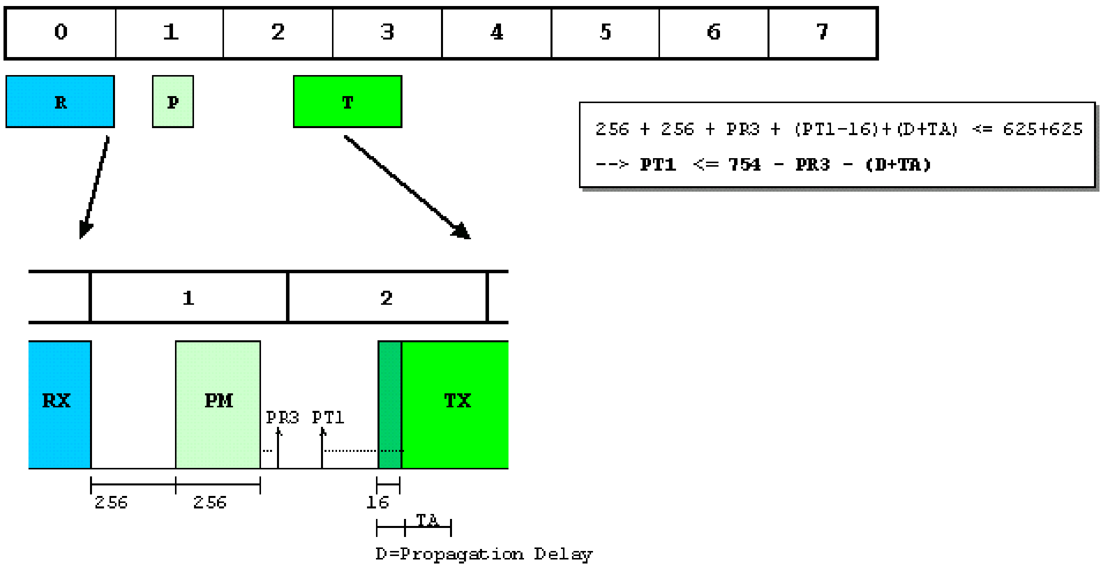

0
1
2
3
4
5
6
7
R
P
T
256 + 256 + PR3 + (PT1-16)+(D+TA) <= 625+625
--> PT1 <= 754 - PR3 - (D+TA)
1
2
RX
PM
PR3
PT1
TX
256
256
16
TA
D=Propagation Delay

Figure 32 The TDMA timing limitation of PTi in GSM mode

Consider the above figure. The figure depicts the worse case of the PT1 limitation in GSM mode. Layer 1 may arrange one PM between NB RX and TX window.

256 + 256 + PR3 + (PT1-16) + (D+TA) <= 625 + 625... (3.1.2.4A)

D is TX propagation delay. TA is the timing advance and the value can be 0 to 256. If TA is 256, PT1 can be gotten as

PT1 <= 498 - PR3 - D ....(3.1.2.4B)

3.1.2.5 Limitation of PT2

PT2 shall be set after the commands of PT1 are sent.

PT1 -PT2 > 0 ....(3.1.2.5A)

And PT2 can be gotten as

PT2 < PT1..(3.1.2.5B)

3.1.2.6 Limitation of PT3

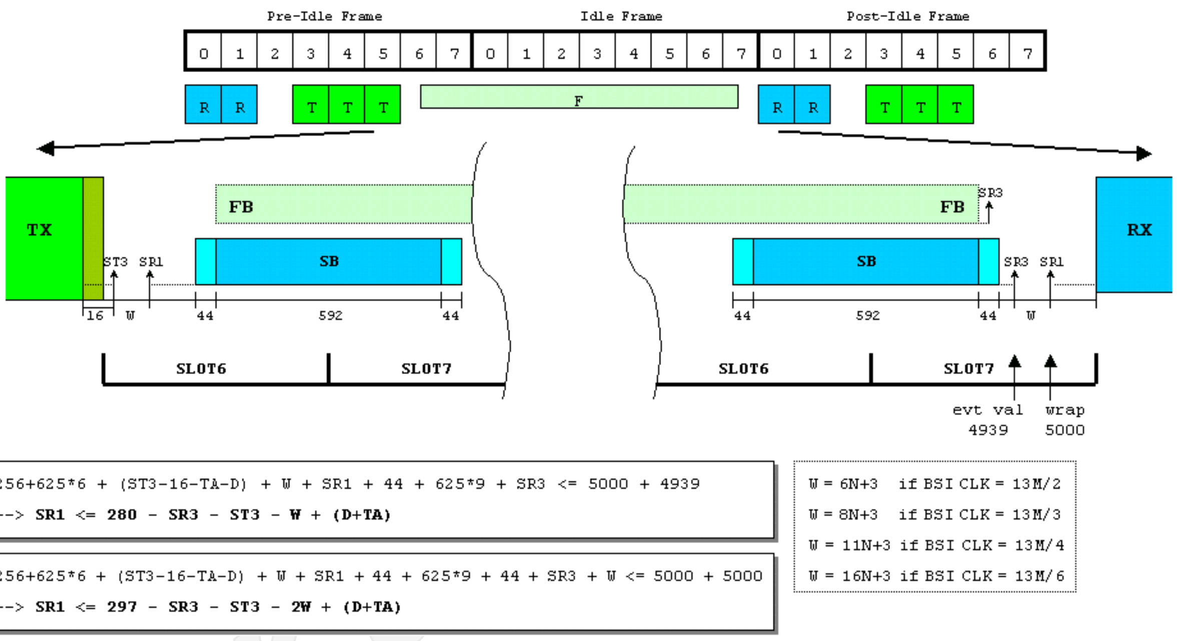
PT3 does not have the limitation but follows the equation (3.1.2.4A).

3.2 TDMA Timing Limitation in GPRS mode

This section describes the TDMA timing limitation in GPRS mode. If the mobile only supports GSM application, the TDMA timing limitation only needs to follow the rules described in last section. Otherwise, the timing setting cannot exceed the limitation described in this section.

3.2.1 BSI TDMA Timing Limitation in GPRS mode

3.2.1.1 Limitation of SR1



```
256+625*6 + (ST3-16-TA-D) + W + SR1 + 44 + 625*9 + SR3 <= 5000 + 4939
--> SR1 <= 280 - SR3 - ST3 - W + (D+TA)

256+625*6 + (ST3-16-TA-D) + W + SR1 + 44 + 625*9 + 44 + SR3 + W <= 5000 + 5000
--> SR1 <= 297 - SR3 - ST3 - 2W + (D+TA)
```

```
W = 6N+3  if BSI CLK = 13M/2
W = 8N+3  if BSI CLK = 13M/3
W = 11N+3 if BSI CLK = 13M/4
W = 16N+3 if BSI CLK = 13M/6
```

Figure 33 The TDMA timing limitation of SRi in GPRS mode

The above figure is the critical constellation of SR1 in GPRS mode. The figure depicts the worse case of the SR1 limitation in GPRS mode. Layer 1 may arrange the 2R3T constellation and one 9 slots length FCCh window. 2 equations can be written as

256+625*6 + (ST3-16-TA-D) + W + SR1 + 44 + 625*9 + SR3 <= 5000 + 4939........................ (3.2.1.1A)
256+625*6 + (ST3-16-TA-D) + W + SR1 + 44 + 625*9 + 44 + SR3 + W <= 5000 + 5000.......... (3.2.1.1B)

Where (1) 256 is the TQ count of slot0.
(2) 625*6 is slot0 to slot6 of pre-idle frame.
(3)16 is the half guard of NB.
(4) 44 is the half extended bits of SB.
(5) 625*9 is the FB search window length.
(6) 5000 is one frame length.
(7) 4939 is the TQ count of Event Validate of idle frame.
(8) W is the timing of 3-wire command sent from BB chip to RF transceiver.

MT62xx can be set 4 3-wire clock rate: 13M/2, 13M/3, 13M/4 Hz, 13M/6 Hz. If N commands need to be sent at SR3, the W can be evaluate as

W = 6N+3	if BSI_CLK is 13M/2 Hz
W = 8N+3	if BSI_CLK is 13M/3 Hz
W = 11N+3	if BSI_CLK is 13M/4 Hz

W = 16N+3 if BSI_CLK is 13M/6 Hz

By the equation (3.2.1.1A) and (3.2.1.1B), SR1 can be gotten as

SR1 <= 280 - SR3 - ST3 - W + (D+TA) .... (3.2.1.1C)
SR1 <= 297 - SR3 - ST3 - 2W + (D+TA) .... (3.2.1.1D)

Where D is the TX propagation delay. TA is the timing advance and the value can be 0 to 256.
If TA is 0, SR1 can be gotten by the equation (3.2.1.1C) and (3.2.1.1D):

SR1 <= 280 - SR3 - ST3 - W + D .... (3.2.1.1E)
SR1 <= 297 - SR3 - ST3 - 2W + D .... (3.2.1.1F)

The limitation of SR1 is the minimum value of (3.2.1.1E) and (3.2.1.1F).

SR1 <= min(280 - SR3 - ST3 - W + D, 297 - SR3 - ST3 - 2W + D) .... (3.2.1.1G)

3.2.1.2 Limitation of SR2

SR2 shall be set after the commands of SR1 are sent.

SR1 - SR2 >= W ....(3.2.1.2A)

If N commands need to be sent at SR1, the W can be evaluate as

W = 6N+3	if BSI_CLK is 13M/2 Hz
W = 8N+3	if BSI_CLK is 13M/3 Hz
W = 11N+3	if BSI_CLK is 13M/4 Hz
W = 16N+3	if BSI_CLK is 13M/6 Hz

And SR2 can be gotten as

SR2 <= SR1 - W..(3.2.1.2B)

3.2.1.3 Limitation of SR3

SR3 does not have the limitation but follows the equation (3.2.1.1A) and (3.2.1.1B).

3.2.1.4 Limitation of ST1

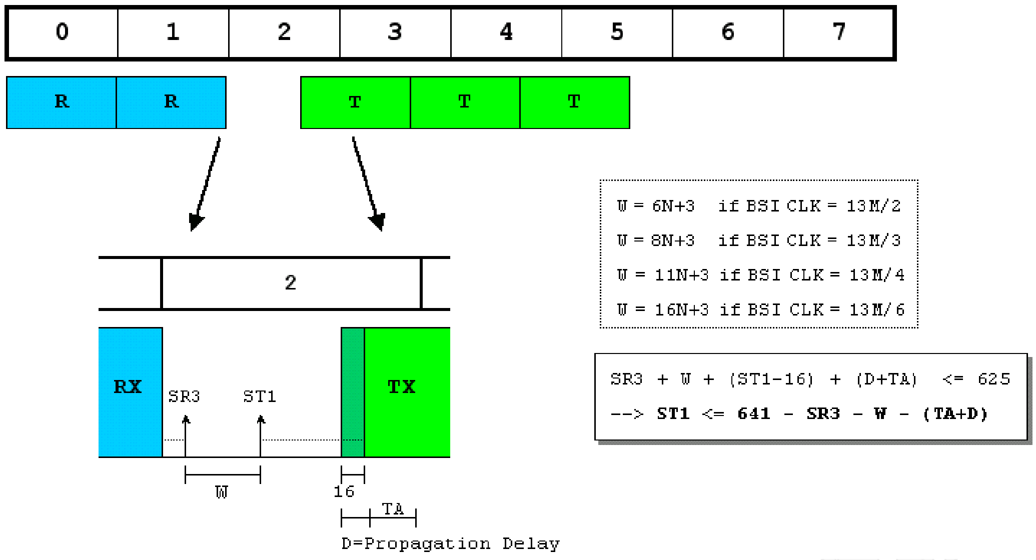

0
1
2
3
4
5
6
7
R
R
T
T
T
2
RX
SR3
ST1
TX
W
16
TA
D=Propagation Delay
W = 6N+3 if BSI CLK = 13M/2
W = 8N+3 if BSI CLK = 13M/3
W = 11N+3 if BSI CLK = 13M/4
W = 16N+3 if BSI CLK = 13M/6
SR3 + W + (ST1-16) + (D+TA) <= 625
--> ST1 <= 641 - SR3 - W - (TA+D)

Figure 34 The TDMA timing limitation of STi in GPRS mode

The above figure is the critical constellation of SR1 in GPRS mode. The figure depicts the worse case of the ST1 limitation in GPRS mode. Layer 1 may arrange the constellation 2R3T where only one slot between RX and TX slot.

SR3 + W + (ST1-16) + (D+TA) <= 625.. (3.2.1.4A)

If N commands need to be sent at SR3, the W can be evaluate as

W = 6N+3	if BSI_CLK is 13M/2 Hz
W = 8N+3	if BSI_CLK is 13M/3 Hz
W = 11N+3	if BSI_CLK is 13M/4 Hz
W = 16N+3	if BSI_CLK is 13M/6 Hz

D is TX propagation delay. TA is the timing advance and the value can be 0 to 256. If TA is 256, ST1 can be gotten by the equation (3.2.4A) :

ST1 <= 385 - SR3 - W - D...(3.2.1.4B)

3.2.1.5 Limitation of ST2

ST2 shall be set after the commands of ST1 are sent.

ST1 -ST2 >= W ....(3.2.1.5A)

Where W is the same definition of section 3.2.1.
And ST2 can be gotten as

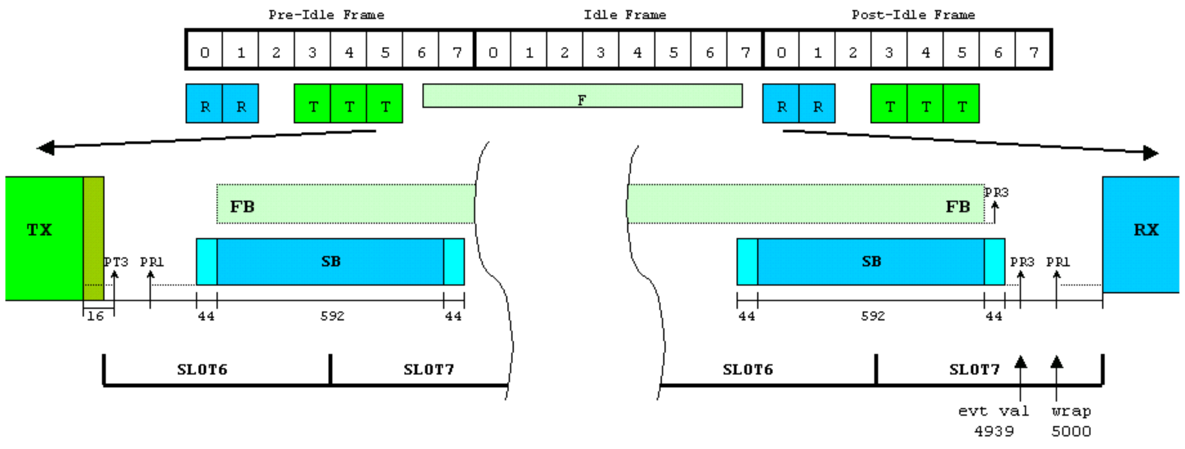
ST2 <= ST1 - W...(3.2.1.5B)

3.2.1.6 Limitation of ST3

ST3 does not have the limitation but follows the equation (3.2.1.4A).

3.2.2 BPI TDMA Timing Limitation in GPRS mode

3.2.2.1 Limitation of PR1



```
256+625*6 + (PT3-16-TA-D) + PR1 + 44 + 625*9 + PR3 <= 5000 + 4939
--> PR1 <= 280 - PR3 - PT3 + (D+TA)
```

Figure 35 The TDMA timing limitation of PRi in GPRS mode

The above figure is the critical constellation of SR1 in GPRS mode. The figure depicts the worse case of the SP1 limitation in GPRS mode. Layer 1 may arrange the 2R3T constellation and one 9 slots length FCCh window.

256+625*6 + (PT3-16-TA-D) + PR1 + 44 + 625*9 + PR3 <= 5000 + 4939..................... (3.2.2.1A)

PR1 can be gotten as

`PR1 <= 280 - PR3 - PT3 + (D+TA)` .... (3.2.2.1B)

Where D is the TX propagation delay. TA is the timing advance and the value can be 0 to 256.
If TA is 0, PR1 can be gotten as

PR1 <= 280 - PR3 - PT3 + D .... (3.2.2.1C)

3.2.2.2 Limitation of PR2

PR2 shall be set after the commands of PR1 are sent.

PR1 -PR2 > 0 ....(3.2.2.2A)

And PR2 can be gotten as

PR2 < PR1..(3.2.2.2B)

3.2.2.3 Limitation of PR3

PR3 does not have the limitation but follows the equation (3.2.2.1A).

3.2.2.4 Limitation of PT1

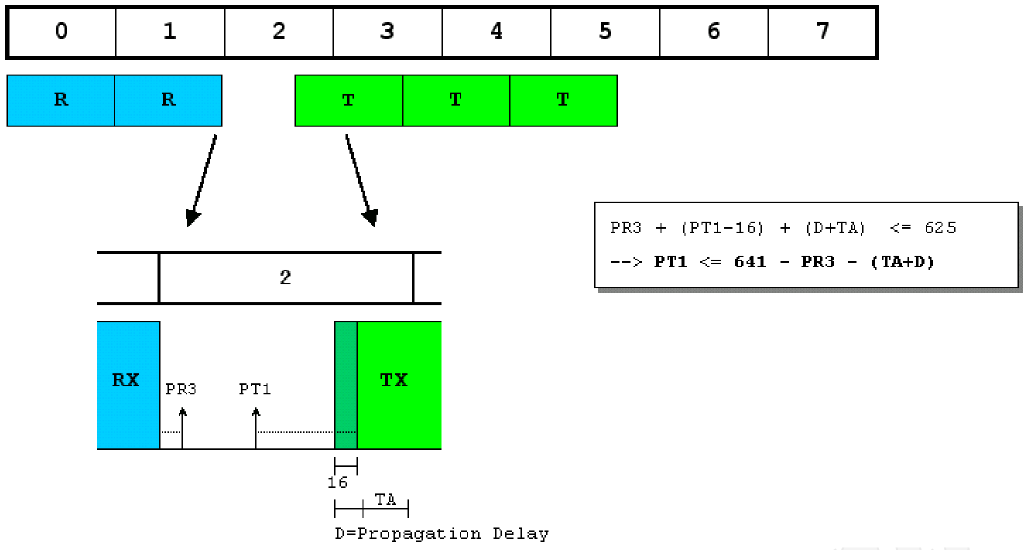


Figure 36 The TDMA timing limitation of PTi in GPRS mode

The above figure is the critical constellation of SR1 in GPRS mode. The figure depicts the worse case of the PT1 limitation in GPRS mode. Layer 1 may arrange the constellation 2R3T where only one slot between RX and TX slot.

PR3 + (PT1-16) + (D+TA) <= 625 .... (3.2.2.4A)

D is TX propagation delay. TA is the timing advance and the value can be 0 to 256. If TA is 256, PT1 can be gotten as

PT1 <= 385 - PR3 - D ....(3.2.2.4B)

3.2.2.5 Limitation of PT2

PT2 shall be set after the commands of PT1 are sent.

PT1 -PT2 > 0 ....(3.2.2.5A)

And PT2 can be gotten as

PT2 < PT1...(3.2.2.5B)

3.2.2.6 Limitation of PT3

PT3 does not have the limitation but follows the equation (3.2.2.4A).

3.3 TDMA Timing Limitation in GSM Mode for MT6162

This section describes the TDMA timing limitation in GSM mode for the MT6162 transceiver. The limitation in GPRS mode for the MT6162 transceiver is described in the next section. The timing setting cannot exceed the limitation described in this section if the MS supports GSM mode.

3.3.1 BSI TDMA Timing Limitation in GSM Mode

3.3.1.1 Limitation of SR0

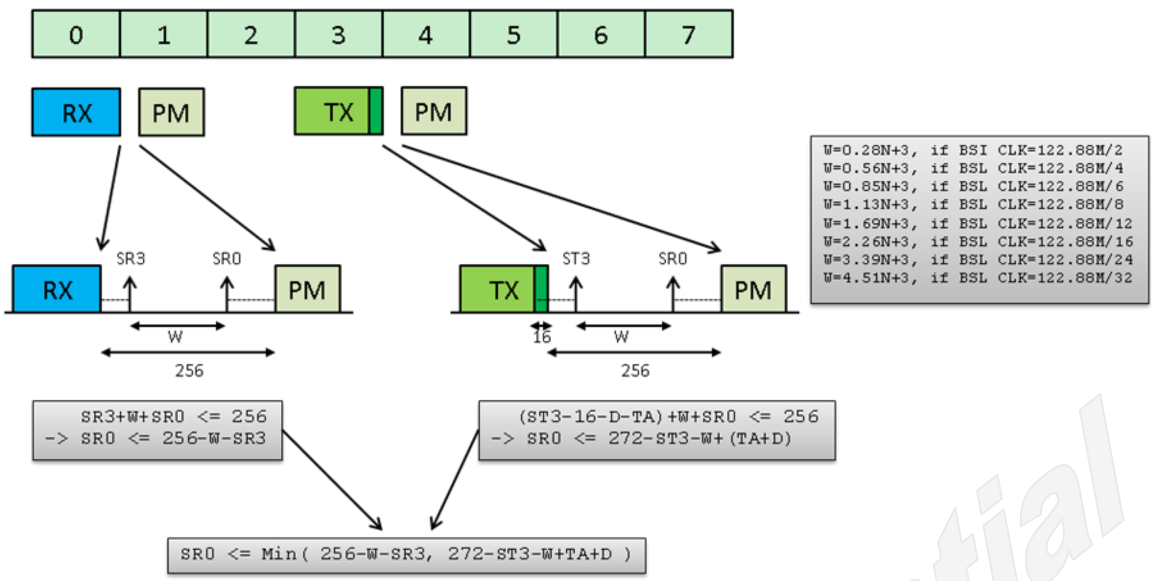

0
1
2
3
4
5
6
7
RX
PM
TX
PM
SR3
SR0
RX
PM
W
256
ST3
SR0
TX
PM
16
W
256
W=0.28N+3, if BSI CLK=122.88M/2
W=0.56N+3, if BSL CLK=122.88M/4
W=0.85N+3, if BSL CLK=122.88M/6
W=1.13N+3, if BSL CLK=122.88M/8
W=1.69N+3, if BSL CLK=122.88M/12
W=2.26N+3, if BSL CLK=122.88M/16
W=3.39N+3, if BSL CLK=122.88M/24
W=4.51N+3, if BSL CLK=122.88M/32
SR3+W+SR0 <= 256
-> SR0 <= 256-W-SR3
(ST3-16-D-TA)+W+SR0 <= 256
-> SR0 <= 272-ST3-W+(TA+D)
SR0 <= Min(256-W-SR3, 272-ST3-W+TA+D)

Figure 37 The TDMA timing limitation of SRi in GSM mode for MT6162

Consider the above figure. The figure depicts the worse case of the SR0 limitation in GSM mode for MT6162. Layer 1 may arrange one PM after NB RX window or TX window.

If PM is behind the NB RX window, the timing equation can be written as

SR3 + W + SR0 <= 256 .... (3.3.1.1A)

Where W is the timing of 4-wire command sent from BB chip to RF transceiver. Total 8 4-wire clock rates can be set in MT6276, 122.88M/2, 122.88M/4, 122.88M/6, 122.88M/8, 122.88M/12, 122.88M/16, 122.88M/24, and 122.88M/32 Hz. Note that for MT6162, the clock rate should be set to 122.88M/2 Hz. If N is the count of commands sent at SR3, the W can be evaluate as

W = 0.28N+3	if BSI_CLK is 122.88M/2 Hz
W = 0.56N+3	if BSI_CLK is 122.88M/4 Hz
W = 0.85N+3	if BSI_CLK is 122.88M/6 Hz
W = 1.13N+3	if BSI_CLK is 122.88M/8 Hz
W = 1.69N+3	if BSI_CLK is 122.88M/12 Hz
W = 2.26N+3	if BSI_CLK is 122.88M/16 Hz
W = 3.39N+3	if BSI_CLK is 122.88M/24 Hz
W = 4.51N+3	if BSI_CLK is 122.88M/32 Hz

By the equation (3.3.1.1A), SR0 can be gotten as

SR0 <= 256 - W - SR3 .... (3.3.1.1B)

If PM is behind the NB TX window, the timing equation can be written as

(ST3 -16 - D - TA) + W + SR0 <= 256.. (3.3.1.1C)

Where D is the TX propagation delay. Half guard period is 16 QB. TA is the timing advance and the range is 0 to 256.

If TA is 0, SR0 can be gotten by the equation (3.3.1.1C) :

SR0 <= 272 - ST3 - W + D .... (3.3.1.1D)

The limitation of SR1 is the minimum value of (3.3.1.1B) and (3.3.1.1D).

SR0 <= min(256 - W - SR3, 272 - ST3 - W + D) .... (3.3.1.1E)

3.3.1.2 Limitation of SR1

SR1 shall be set after the commands of SR0 are sent.

SR0 - SR1 >= W ....(3.3.1.2A)

If N is the count of commands sent at SR0, the W can be evaluate as

W = 0.28N+3	if BSI_CLK is 122.88M/2 Hz
W = 0.56N+3	if BSI_CLK is 122.88M/4 Hz
W = 0.85N+3	if BSI_CLK is 122.88M/6 Hz
W = 1.13N+3	if BSI_CLK is 122.88M/8 Hz
W = 1.69N+3	if BSI_CLK is 122.88M/12 Hz
W = 2.26N+3	if BSI_CLK is 122.88M/16 Hz
W = 3.39N+3	if BSI_CLK is 122.88M/24 Hz
W = 4.51N+3	if BSI_CLK is 122.88M/32 Hz

And SR1 can be gotten as

SR1 <= SR0 - W ....(3.3.1.2B)

3.3.1.3 Limitation of SR2

SR2 shall be set after the commands of SR1 are sent.

SR1 - SR2 >= W ....(3.3.1.3A)

If N is the count of commands sent at SR0, the W can be evaluate as

W = 0.28N+3	if BSI_CLK is 122.88M/4 Hz
W = 0.56N+3	if BSI_CLK is 122.88M/4 Hz
W = 0.85N+3	if BSI_CLK is 122.88M/6 Hz
W = 1.13N+3	if BSI_CLK is 122.88M/8 Hz
W = 1.69N+3	if BSI_CLK is 122.88M/12 Hz
W = 2.26N+3	if BSI_CLK is 122.88M/16 Hz
W = 3.39N+3	if BSI_CLK is 122.88M/24 Hz
W = 4.51N+3	if BSI_CLK is 122.88M/32 Hz

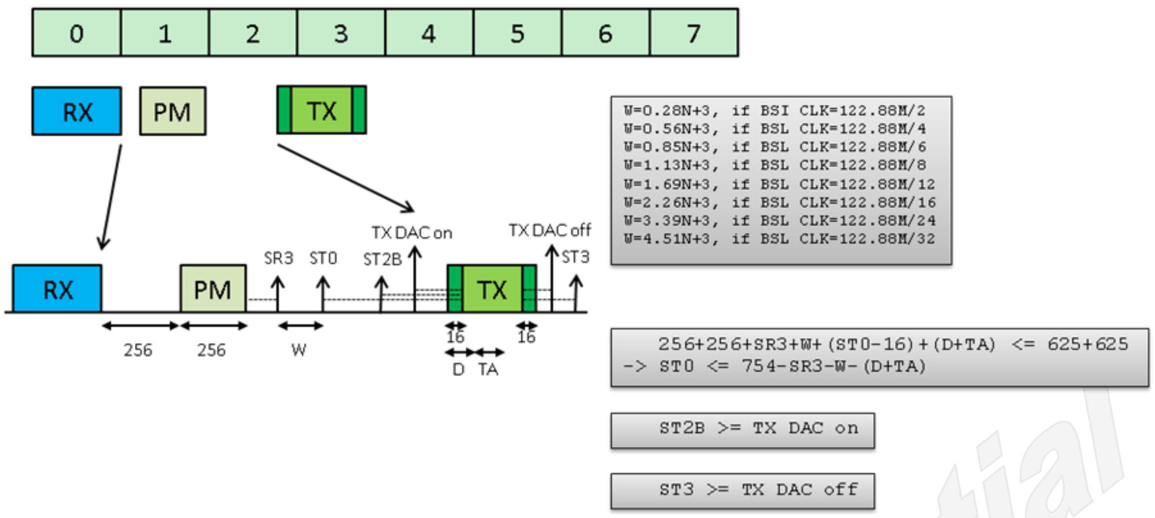
And SR2 can be gotten as

SR2 <= SR1 - W ....(3.3.1.3B)

3.3.1.4 Limitation of SR3

SR3 does not have the limitation but follows the equation (3.3.1.1A).

3.3.1.5 Limitation of ST0



```
W=0.28N+3, if BSI CLK=122.88M/2
W=0.56N+3, if BSL CLK=122.88M/4
W=0.85N+3, if BSL CLK=122.88M/6
W=1.13N+3, if BSL CLK=122.88M/8
W=1.69N+3, if BSL CLK=122.88M/12
W=2.26N+3, if BSL CLK=122.88M/16
W=3.39N+3, if BSL CLK=122.88M/24
W=4.51N+3, if BSL CLK=122.88M/32
```

```
   256+256+SR3+W+(ST0-16)+(D+TA)  <= 625+625
-> ST0 <= 754-SR3-W-(D+TA)
```

```
   ST2B >= TX DAC on
```

```
   ST3 >= TX DAC off
```

Figure 38 The TDMA timing limitation of STi in GSM mode for MT6162

Consider the above figure. The figure depicts the worse case of the ST0 limitation in GSM mode for MT6162. Layer 1 may arrange one PM between NB RX and TX window. The timing equation can be written as

256 + 256 + SR3 + W + (ST0-16) + (D+TA) <= 625 + 625...................................... (3.3.1.5A)

If N is the count of commands sent at SR3, the W can be evaluate as

W = 0.28N+3	if BSI_CLK is 122.88M/4 Hz
W = 0.56N+3	if BSI_CLK is 122.88M/4 Hz
W = 0.85N+3	if BSI_CLK is 122.88M/6 Hz
W = 1.13N+3	if BSI_CLK is 122.88M/8 Hz
W = 1.69N+3	if BSI_CLK is 122.88M/12 Hz
W = 2.26N+3	if BSI_CLK is 122.88M/16 Hz
W = 3.39N+3	if BSI_CLK is 122.88M/24 Hz
W = 4.51N+3	if BSI_CLK is 122.88M/32 Hz

D is TX propagation delay. TA is the timing advance and the value can be 0 to 256. If TA is 256, ST0 can be gotten by the equation (3.3.1.5A) :

ST0 <= 498 - SR3 - W - D ....(3.3.1.5B)

3.3.1.6 Limitation of ST2

ST2 shall be set after the commands of ST0 are sent.

ST0 -ST2 >= W ....(3.3.1.6A)

If N is the count of commands sent at ST0, the W can be evaluate as

W = 0.28N+3	if BSI_CLK is 122.88M/4 Hz
W = 0.56N+3	if BSI_CLK is 122.88M/4 Hz
W = 0.85N+3	if BSI_CLK is 122.88M/6 Hz
W = 1.13N+3	if BSI_CLK is 122.88M/8 Hz
W = 1.69N+3	if BSI_CLK is 122.88M/12 Hz
W = 2.26N+3	if BSI_CLK is 122.88M/16 Hz

W = 3.39N+3	if BSI_CLK is 122.88M/24 Hz
W = 4.51N+3	if BSI_CLK is 122.88M/32 Hz

And ST2 can be gotten as

ST2 <= ST0 - W ....(3.3.1.6B)

3.3.1.7 Limitation of ST1

ST1 shall be set after the commands of ST2 are sent.

ST2 - ST1 >= W ....(3.3.1.7A)

If N is the count of commands sent at ST0, the W can be evaluate as

W = 0.28N+3	if BSI_CLK is 122.88M/4 Hz
W = 0.56N+3	if BSI_CLK is 122.88M/4 Hz
W = 0.85N+3	if BSI_CLK is 122.88M/6 Hz
W = 1.13N+3	if BSI_CLK is 122.88M/8 Hz
W = 1.69N+3	if BSI_CLK is 122.88M/12 Hz
W = 2.26N+3	if BSI_CLK is 122.88M/16 Hz
W = 3.39N+3	if BSI_CLK is 122.88M/24 Hz
W = 4.51N+3	if BSI_CLK is 122.88M/32 Hz

And ST1 can be gotten as

ST1 <= ST2 - W ....(3.1.1.7B)

3.3.1.8 Limitation of ST2B

ST2B shall be set after the commands of ST1 are sent.

ST1 - ST2B >= W ....(3.3.1.8A)

If N is the count of commands sent at ST0, the W can be evaluate as

W = 0.28N+3	if BSI_CLK is 122.88M/4 Hz
W = 0.56N+3	if BSI_CLK is 122.88M/4 Hz
W = 0.85N+3	if BSI_CLK is 122.88M/6 Hz
W = 1.13N+3	if BSI_CLK is 122.88M/8 Hz
W = 1.69N+3	if BSI_CLK is 122.88M/12 Hz
W = 2.26N+3	if BSI_CLK is 122.88M/16 Hz
W = 3.39N+3	if BSI_CLK is 122.88M/24 Hz
W = 4.51N+3	if BSI_CLK is 122.88M/32 Hz

In addition, ST2B shall be set before TX DAC on, as shown in Figure 31.

ST2B >= TX DAC on ....(3.3.1.8B)

And ST2B can be gotten as

TX DAC on <= ST2B <= ST1 - W ....(3.3.1.8C)

3.3.1.9 Limitation of ST3

ST3 shall be set after TX DAC off, as shown in Figure 31.

ST3 >= TX DAC off ....(3.3.1.9A)

3.3.2 BPI TDMA Timing Limitation in GSM Mode

3.3.2.1 Limitation of PR1

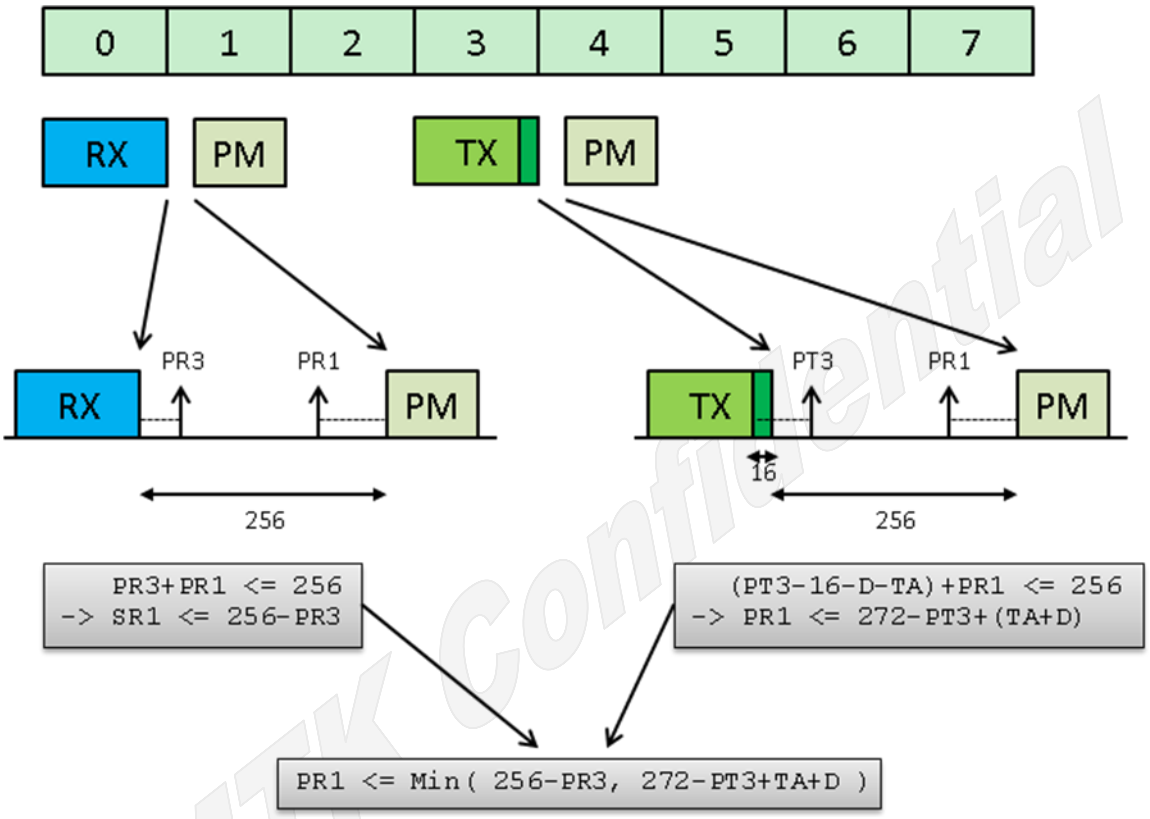


Figure 39 The TDMA timing limitation of PRi in GSM mode for MT6162

Consider the above figure. The figure depicts the worse case of the PR1 limitation in the GSM mode for MT6162. Layer 1 may arrange one PM after NB RX window or TX window.

If PM behinds NB RX window, the timing equation can be written as

PR3 + PR1 <= 256 .... (3.3.2.1A)

PR1 can be gotten as

PR1 <= 256 - PR3 .... (3.3.2.1B)

If PM behinds NB TX window, the timing equation can be written as

(PT3 -16 - D - TA) + PR1 <= 256.. (3.3.2.1C)

Where D is the TX propagation delay. TA is the timing advance and the value can be 0 to 256.

If TA is 0, PR1 can be gotten as

PR1 <= 272 - PT3 + D .... (3.3.2.1D)

The limitation of PR1 is the minimum value of (3.3.2.1B) and (3.3.2.1D).

PR1 <= min(256 - PR3, 272 - PT3 + D) .... (3.3.2.1E)

3.3.2.2 Limitation of PR2

PR2 shall be set after the commands of PR1 are sent.

PR1 -PR2 > 0 ....(3.3.2.2A)

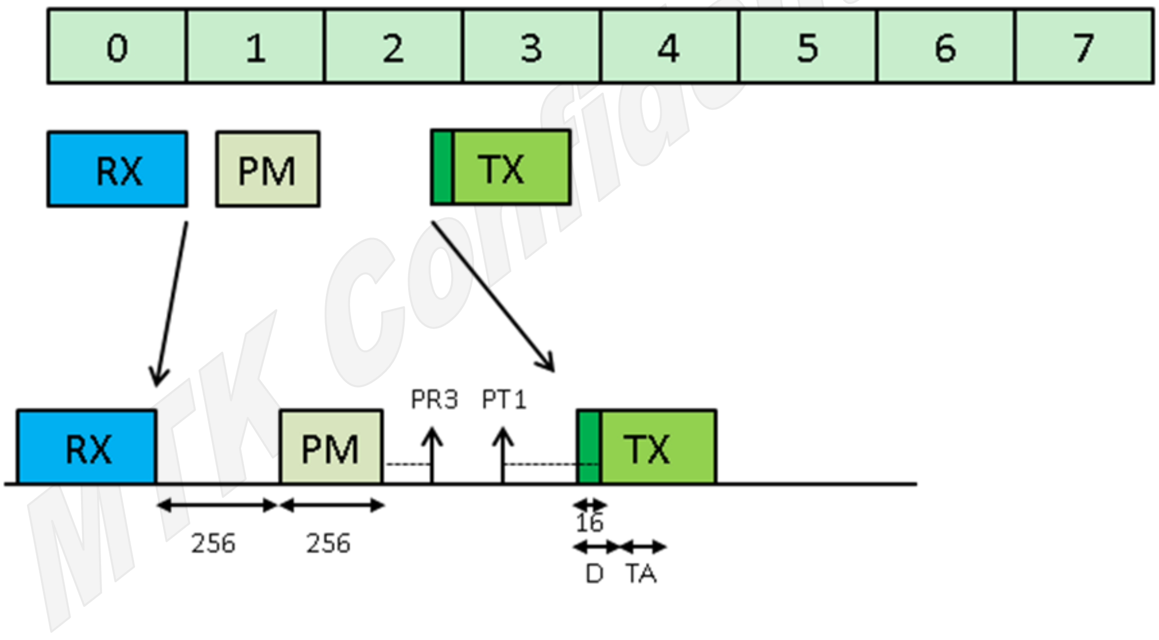
And PR2 can be gotten as

PR2 < PR1..(3.3.2.2B)

3.3.2.3 Limitation of PR3

PR3 does not have the limitation but follows the equation (3.3.2.1A).

3.3.2.4 Limitation of PT1



```
   256+256+PR3+(PT1-16)+(D+TA)  <=  625+625
-> PT1  <=  754-PR3-(D+TA)
```

Figure 40 The TDMA timing limitation of PTi in GSM mode for MT6162

Consider the above figure. The figure depicts the worse case of the PT1 limitation in GSM mode for MT6162. Layer 1 may arrange one PM between NB RX and TX window.

256 + 256 + PR3 + (PT1-16) + (D+TA) <= 625 + 625... (3.3.2.4A)

D is TX propagation delay. TA is the timing advance and the value can be 0 to 256. If TA is 256, PT1 can be gotten as

PT1 <= 498 - PR3 - D ....(3.3.2.4B)

3.3.2.5 Limitation of PT2

PT2 shall be set after the commands of PT1 are sent.

PT1 -PT2 > 0 ....(3.3.2.5A)

And PT2 can be gotten as

PT2 < PT1..(3.3.2.5B)

3.3.2.6 Limitation of PT2B

PT2B shall be set after the commands of PT2 are sent.

PT2 -PT2B > 0 ....(3.3.2.6A)

And PT2B can be gotten as

PT2B < PT2..(3.3.2.6B)

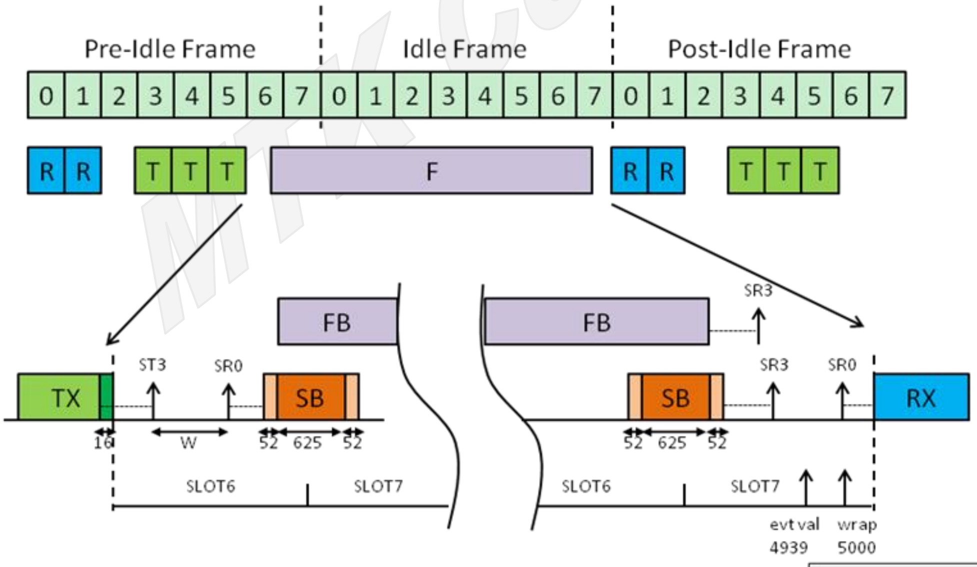
3.3.2.7 Limitation of PT3

PT3 does not have the limitation but follows the equation (3.3.2.1C).

3.4 TDMA Timing Limitation in GPRS Mode for MT6162

3.4.1 BSI TDMA Timing Limitation in GPRS Mode

3.4.1.1 Limitation of SR0



```
   256+625*6+(ST3-16-TA-D)+W+SR0+52+625*9+SR3 <= 5000+4939
-> SR0 <= 272-SR3-ST3-W+(D+TA)
```

```
   256+625*6+(ST3-16-TA-D)+W+SR0+52+625*9+52+SR3+W <= 5000+5000
-> SR0 <= 281-ST3-W+(D+TA)
```

```
W=0.28N+3, if BSI CLK=122.88M/2
W=0.56N+3, if BSL CLK=122.88M/4
W=0.85N+3, if BSL CLK=122.88M/6
W=1.13N+3, if BSL CLK=122.88M/8
W=1.69N+3, if BSL CLK=122.88M/12
W=2.26N+3, if BSL CLK=122.88M/16
W=3.39N+3, if BSL CLK=122.88M/24
W=4.51N+3, if BSL CLK=122.88M/32
```

Figure 41 The TDMA timing limitation of SRi in GPRS mode for MT6162

The above figure is the critical constellation of SR1 in GPRS mode. The figure depicts the worse case of the SR1 limitation in GPRS mode. Layer 1 may arrange the 2R3T constellation and one 9 slots length FCCh window. 2 equations can be written as

256+625*6 + (ST3-16-TA-D) + W + SR0 + 52 + 625*9 + SR3 <= 5000 + 4939........................ (3.4.1.1A)
256+625*6 + (ST3-16-TA-D) + W + SR0 + 52 + 625*9 + 52 + SR3 + W <= 5000 + 5000.......... (3.4.1.1B)

Where	(1) 256 is the TQ count of slot0.
	(2) 625*6 is slot0 to slot6 of pre-idle frame.
	(3) 16 is the half guard of NB.
	(4) 52 is the half extended bit of SB.
	(5) 625*9 is the FB search window length.
	(6) 5000 is one frame length.
	(7) 4939 is the TQ count of Event Validate of idle frame.
	(8) W is the timing of 4-wire command sent from BB chip to RF transceiver.

Total 8 4-wire clock rates can be set in MT6276, 122.88M/2, 122.88M/4, 122.88M/6, 122.88M/8, 122.88M/12, 122.88M/16, 122.88M/24, and 122.88M/32 Hz. Note that for MT6162, the clock rate should be set to 122.88M/2 Hz. If N is the count of commands sent at SR3, the W can be evaluate as

W = 0.28N+3	if BSI_CLK is 122.88M/2 Hz
W = 0.56N+3	if BSI_CLK is 122.88M/4 Hz
W = 0.85N+3	if BSI_CLK is 122.88M/6 Hz
W = 1.13N+3	if BSI_CLK is 122.88M/8 Hz
W = 1.69N+3	if BSI_CLK is 122.88M/12 Hz
W = 2.26N+3	if BSI_CLK is 122.88M/16 Hz
W = 3.39N+3	if BSI_CLK is 122.88M/24 Hz
W = 4.51N+3	if BSI_CLK is 122.88M/32 Hz

By the equation (3.4.1.1A) and (3.4.1.1B), SR1 can be gotten as

SR0 <= 272 - SR3 - ST3 - W + (D+TA) .... (3.4.1.1C)
SR0 <= 281 - SR3 - ST3 - 2W + (D+TA) .... (3.4.1.1D)

Where D is the TX propagation delay. TA is the timing advance and the value can be 0 to 256.
If TA is 0, SR0 can be gotten by the equation (3.4.1.1C) and (3.4.1.1D):

SR0 <= 272 - SR3 - ST3 - W + D .... (3.4.1.1E)
SR0 <= 281 - SR3 - ST3 - 2W + D .... (3.4.1.1F)

The limitation of SR0 is the minimum value of (3.4.1.1E) and (3.4.1.1F).

SR0 <= min(272 - SR3 - ST3 - W + D, 281 - SR3 - ST3 - 2W + D) .... (3.4.1.1G)

3.4.1.2 Limitation of SR1

SR1 shall be set after the commands of SR0 are sent.

SR0 - SR1 >= W ....(3.4.1.2A)

If N is the count of commands sent at SR0, the W can be evaluate as

W = 0.28N+3	if BSI_CLK is 122.88M/2 Hz
W = 0.56N+3	if BSI_CLK is 122.88M/4 Hz

W = 0.85N+3	if BSI_CLK is 122.88M/6 Hz
W = 1.13N+3	if BSI_CLK is 122.88M/8 Hz
W = 1.69N+3	if BSI_CLK is 122.88M/12 Hz
W = 2.26N+3	if BSI_CLK is 122.88M/16 Hz
W = 3.39N+3	if BSI_CLK is 122.88M/24 Hz
W = 4.51N+3	if BSI_CLK is 122.88M/32 Hz

And SR1 can be gotten as

SR1 <= SR0 - W ....(3.4.1.2B)

3.4.1.3 Limitation of SR2

SR2 shall be set after the commands of SR1 are sent.

SR1 - SR2 >= W ....(3.4.1.3A)

If N is the count of commands sent at SR0, the W can be evaluate as

W = 0.28N+3	if BSI_CLK is 122.88M/4 Hz
W = 0.56N+3	if BSI_CLK is 122.88M/4 Hz
W = 0.85N+3	if BSI_CLK is 122.88M/6 Hz
W = 1.13N+3	if BSI_CLK is 122.88M/8 Hz
W = 1.69N+3	if BSI_CLK is 122.88M/12 Hz
W = 2.26N+3	if BSI_CLK is 122.88M/16 Hz
W = 3.39N+3	if BSI_CLK is 122.88M/24 Hz
W = 4.51N+3	if BSI_CLK is 122.88M/32 Hz

And SR2 can be gotten as

SR2 <= SR1 - W ....(3.4.1.3B)

3.4.1.4 Limitation of SR3

SR3 does not have the limitation but follows the equation (3.4.1.1A) and (3.4.1.1B).

3.4.1.5 Limitation of ST0

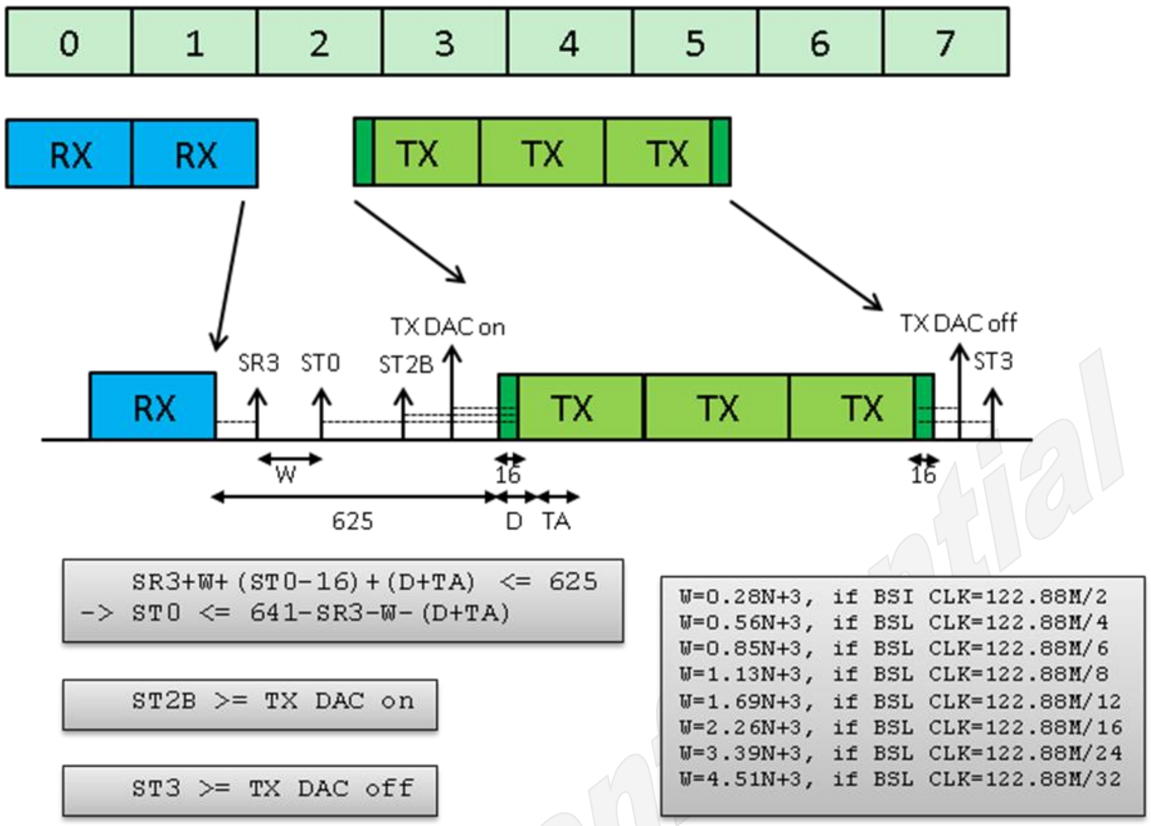

0
1
2
3
4
5
6
7
RX
RX
TX
TX
TX
TX DAC on
TX DAC off
SR3
ST0
ST2B
ST3
RX
TX
TX
TX
W
16
16
625
D
TA
SR3+W+(ST0-16)+(D+TA) <= 625
-> ST0 <= 641-SR3-W-(D+TA)
ST2B >= TX DAC on
ST3 >= TX DAC off
W=0.28N+3, if BSI CLK=122.88M/2
W=0.56N+3, if BSL CLK=122.88M/4
W=0.85N+3, if BSL CLK=122.88M/6
W=1.13N+3, if BSL CLK=122.88M/8
W=1.69N+3, if BSL CLK=122.88M/12
W=2.26N+3, if BSL CLK=122.88M/16
W=3.39N+3, if BSL CLK=122.88M/24
W=4.51N+3, if BSL CLK=122.88M/32

Figure 42 The TDMA timing limitation of STi in GPRS mode for MT6162

The above figure is the critical constellation of SR0 in the GPRS mode for MT6162. The figure depicts the worse case of the ST0 limitation in the GPRS mode. Layer 1 may arrange the constellation 2R3T where only one slot between the RX and TX slots.

SR3 + W + (ST0-16) + (D+TA) <= 625.. (3.4.1.5A)

If N commands need to be sent at SR3, the W can be evaluate as

W = 0.28N+3	if BSI_CLK is 122.88M/2 Hz
W = 0.56N+3	if BSI_CLK is 122.88M/4 Hz
W = 0.85N+3	if BSI_CLK is 122.88M/6 Hz
W = 1.13N+3	if BSI_CLK is 122.88M/8 Hz
W = 1.69N+3	if BSI_CLK is 122.88M/12 Hz
W = 2.26N+3	if BSI_CLK is 122.88M/16 Hz
W = 3.39N+3	if BSI_CLK is 122.88M/24 Hz
W = 4.51N+3	if BSI_CLK is 122.88M/32 Hz

D is TX propagation delay. TA is the timing advance and the value can be 0 to 256. If TA is 256, ST1 can be gotten by the equation (3.4.1.5A) :

ST0 <= 385 - SR3 - W - D...(3.4.1.5B)

3.4.1.6 Limitation of ST2

ST2 shall be set after the commands of ST0 are sent.

ST0 -ST2 >= W ....(3.4.1.6A)

If N is the count of commands sent at ST0, the W can be evaluate as

W = 0.28N+3	if BSI_CLK is 122.88M/4 Hz
W = 0.56N+3	if BSI_CLK is 122.88M/4 Hz
W = 0.85N+3	if BSI_CLK is 122.88M/6 Hz
W = 1.13N+3	if BSI_CLK is 122.88M/8 Hz
W = 1.69N+3	if BSI_CLK is 122.88M/12 Hz
W = 2.26N+3	if BSI_CLK is 122.88M/16 Hz
W = 3.39N+3	if BSI_CLK is 122.88M/24 Hz
W = 4.51N+3	if BSI_CLK is 122.88M/32 Hz

And ST2 can be gotten as

ST2 <= ST0 - W ....(3.4.1.6B)

3.4.1.7 Limitation of ST1

ST1 shall be set after the commands of ST2 are sent.

ST2 - ST1 >= W ....(3.4.1.7A)

If N is the count of commands sent at ST0, the W can be evaluate as

W = 0.28N+3	if BSI_CLK is 122.88M/4 Hz
W = 0.56N+3	if BSI_CLK is 122.88M/4 Hz
W = 0.85N+3	if BSI_CLK is 122.88M/6 Hz
W = 1.13N+3	if BSI_CLK is 122.88M/8 Hz
W = 1.69N+3	if BSI_CLK is 122.88M/12 Hz
W = 2.26N+3	if BSI_CLK is 122.88M/16 Hz
W = 3.39N+3	if BSI_CLK is 122.88M/24 Hz
W = 4.51N+3	if BSI_CLK is 122.88M/32 Hz

And ST1 can be gotten as

ST1 <= ST2 - W ....(4.1.1.7B)

3.4.1.8 Limitation of ST2B

ST2B shall be set after the commands of ST1 are sent.

ST1 - ST2B >= W ....(3.4.1.8A)

If N is the count of commands sent at ST0, the W can be evaluate as

W = 0.28N+3	if BSI_CLK is 122.88M/4 Hz
W = 0.56N+3	if BSI_CLK is 122.88M/4 Hz
W = 0.85N+3	if BSI_CLK is 122.88M/6 Hz
W = 1.13N+3	if BSI_CLK is 122.88M/8 Hz
W = 1.69N+3	if BSI_CLK is 122.88M/12 Hz
W = 2.26N+3	if BSI_CLK is 122.88M/16 Hz
W = 3.39N+3	if BSI_CLK is 122.88M/24 Hz
W = 4.51N+3	if BSI_CLK is 122.88M/32 Hz

In addition, ST2B shall be set before TX DAC on, as shown in Figure 33.

ST2B >= TX DAC on ....(3.4.1.8B)

And ST2B can be gotten as

TX DAC on <= ST2B <= ST1 - W ....(3.4.1.8C)

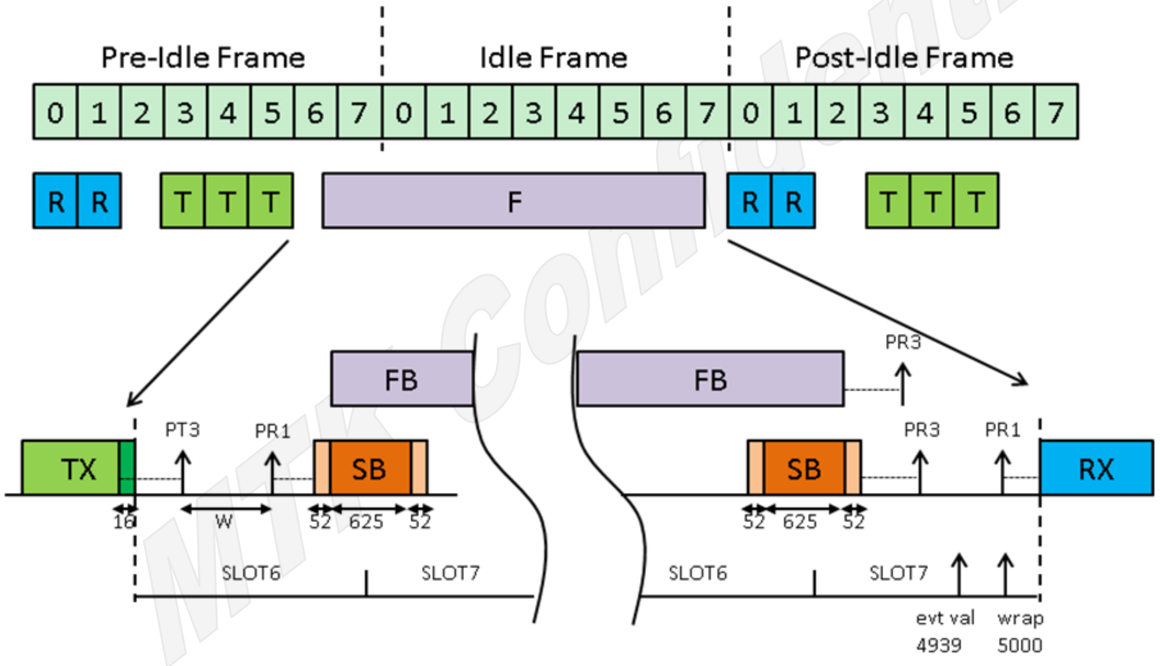
3.4.1.9 Limitation of ST3

ST3 shall be set after TX DAC off, as shown in Figure 33.

ST3 >= TX DAC off ....(3.4.1.9A)

3.4.2 BPI TDMA Timing Limitation in GPRS Mode

3.4.2.1 Limitation of PR1



```
   256+625*6+(PT3-16-TA-D)+PR1+52+625*9+PR3  <=  5000+4939
-> SR0  <=  272-PR3-PT3+(D+TA)
```

Figure 43 The TDMA timing limitation of PRi in GPRS mode for MT6162

The above figure is the critical constellation of SR1 in the GPRS mode for MT6162. The figure depicts the worse case of the BP1 limitation in GPRS mode. Layer 1 may arrange the 2R3T constellation and one 9-slot length FCCh window.

256+625*6 + (PT3-16-TA-D) + PR1 + 52 + 625*9 + PR3 <= 5000 + 4939.................... (3.4.2.1A)

PR1 can be gotten as

PR1 <= 280 - PR3 - PT3 + (D+TA) .... (3.4.2.1B)

Where D is the TX propagation delay. TA is the timing advance and the value can be 0 to 256.
If TA is 0, PR1 can be gotten as

PR1 <= 280 - PR3 - PT3 + D .... (3.4.2.1C)

3.4.2.2 Limitation of PR2

PR2 shall be set after the commands of PR1 are sent.

PR1 -PR2 > 0 ....(3.4.2.2A)

And PR2 can be gotten as

PR2 < PR1..(3.4.2.2B)

3.4.2.3 Limitation of PR3

PR3 does not have the limitation but follows the equation (3.4.2.1A).

3.4.2.4 Limitation of PT1

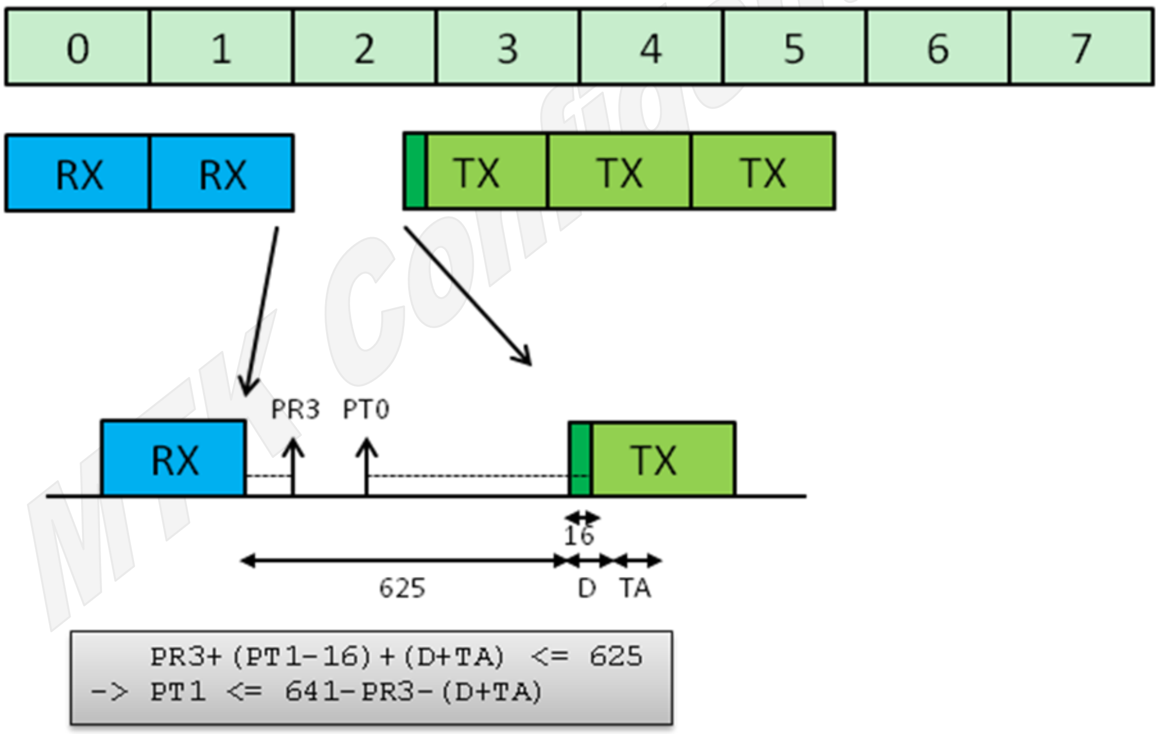


Figure 44 The TDMA timing limitation of PTi in GPRS mode for MT6162

The above figure is the critical constellation of PT1 in the GPRS mode for MT6162. The figure depicts the worse case of the PT1 limitation in GPRS mode. Layer 1 may arrange the constellation 2R3T where only one slot between RX and TX slot.

PR3 + (PT1-16) + (D+TA) <= 625 .... (3.4.2.4A)

D is TX propagation delay. TA is the timing advance and the value can be 0 to 256. If TA is 256, PT1 can be gotten as

PT1 <= 385 - PR3 - D ....(3.2.2.4B)

3.4.2.5 Limitation of PT2

PT2 shall be set after the commands of PT1 are sent.

PT1 -PT2 > 0 ....(3.4.2.5A)

And PT2 can be gotten as

PT2 < PT1..(3.4.2.5B)

3.4.2.6 Limitation of PT2B

PT2B shall be set after the commands of PT2 are sent.

PT2 -PT2B > 0 ....(3.4.2.6A)

And PT2B can be gotten as

PT2B < PT2..(3.4.2.6B)

3.4.2.7 Limitation of PT3

PT3 does not have the limitation but follows the equation (3.4.2.1C).

4 Baseband Front End Setting

The setting of MT62xx Baseband front end is defined as aliases in **l1d_custom_rf.h**.

4.1 RX Path

4.1.1 IQ Swap

The I/Q data of receiving can be swapped by setting the alias **BBRX_IQ_SWAP.** To enable the swapping, set the value to 1. Or to bypass the swapping, set the value to 0:

```
#define   BBRX_IQ_SWAP        on_off
```

4.2 TX Path

4.2.1 IQ Swap

The I/Q data of transmission can be swapped by setting the alias **BBTX_IQ_SWAP.** To enable the swapping, set the value to 1. Or to bypass the swapping, set the value to 0:

```
#define   BBTX_IQ_SWAP        on_off
```

4.2.2 I/Q Swing

The amplitude of TX I/Q data from baseband to TX modulator in transceiver may be different depend on transceiver. The I and Q swing can be set by the alias **BBTX_GAIN. BBTX_TRIM_I** and **BBTX_TRIM_Q** are set to individually compensate gain mismatch.

```
#define   BBTX_GAIN      level
```

The value of level is mapping to the following table:

level	voltage	level	voltage
3	AVDD*0.900	7	AVDD*0.360
2	AVDD*0.720	6	AVDD*0.288
1	AVDD*0.576	5	AVDD*0.225
0	AVDD*0.450	4	AVDD*0.180

The AVDD is the supply voltage for baseband transmission section. Please refer to Power Description sector of MT62xx Data Sheet.

```
#define   BBTX_TRIM_I     i_gain
#define   BBTX_TRIM_Q     q_gain
```

The value of gain is mapping to the following table:

value	gain	value	gain
7	1.18 dB	15	-0.16 dB
6	1.00 dB	14	-0.31 dB
5	0.83 dB	13	-0.46 dB
4	0.66 dB	12	-0.61 dB

3	0.49 dB	11	-0.75 dB
2	0.32 dB	10	-0.90 dB
1	0.16 dB	9	-1.04 dB
0	0.00 dB	8	-1.18 dB

4.2.3 I/Q DC Offset

The bias of TX I/Q data from baseband to TX modulator in transceiver may be different depend on transceiver. To control the bias voltage of TX I/Q data from baseband, TX common-mode voltage should be set by the alias **BBTX_COMMON_MODE_VOLTAGE.** The individual I/Q dc offset fine tuned are set by **BBTX_OFFSET_I** and **BBTX_OFFSET_Q.**

```
#define   BBTX_COMMON_MODE_VOLTAGE      level
```

The value of level is mapping to the following table:

level	voltage	level	voltage
3	AVDD*0.62	7	AVDD*0.46
2	AVDD*0.58	6	AVDD*0.42
1	AVDD*0.54	5	AVDD*0.38
0	AVDD*0.50	4	AVDD*0.34

The AVDD is the supply voltage for baseband transmission section. Please refer to Power Description sector of MT62xx Data Sheet.

```
#define   BBTX_OFFSET_I        i_setting
#define   BBTX_OFFSET_Q        q_setting
```

The offset value can be evaluated by the following equation:

Offset = setting / 1023 * AVDD

4.2.4 Bias Current

The bias current of TX path can be set by the alias **BBTX_CALBIAS.**

```
#define   BBTX_CALBIAS      setting
```

.

The value of setting is mapping to the following table:

setting	output current	setting	output current
15	1.750	31	0.958
14	1.686	30	0.918
13	1.624	29	0.879
12	1.564	28	0.842
11	1.507	27	0.807
10	1.452	26	0.773
9	1.399	25	0.741
8	1.348	24	0.709
7	1.298	23	0.680
6	1.251	22	0.651
5	1.205	21	0.624

4	1.161	20	0.598
3	1.118	19	0.572
2	1.077	18	0.548
1	1.038	17	0.525
0	1.000	16	0.503

Higher bias current can speed the TX part circuit of BFE and improve the TX SNR, but the extra power will be wasted.

4.2.5 Cut-off Frequency of Smooth Filter

The cut-off frequency of smooth filter of TX path can be set by the alias **BBTX_CALRCSEL.**

```
#define  BBTX_CALRCSEL     setting
```

.

The value of setting is mapping to the following table:

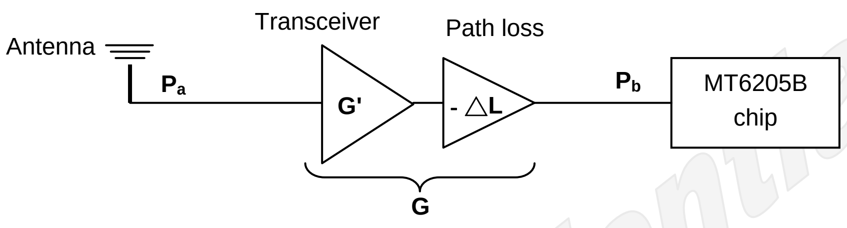
setting	Band Width	setting	Band Width
3	213 KHz	7	394 KHz
2	245 KHz	6	450 KHz
1	289 KHz	5	520 KHz
0	350 KHz	4	620 KHz

5 Path Loss and Gain Configuration

The gain setting of transceiver evaluated by function **L1D_RF_GetGainSetting** may have some inaccuracy depending on radio channel. A configuration table to compensate this loss of gain is needed.

5.1 Radio Gain and Path Loss

Refer to the below diagram. The signal power received by antenna is $\mathbf{P_a}$. The signal power received by MT62xx chip is $\mathbf{P_b}$. The setting gain of transceiver is **G'**. The path loss gain is **-△L**. The real gain from antenna to MT62xx chip is **G**.



$$P_b = P_a + G$$
$$= P_a + G' - \triangle L$$

Figure 45 The path loss of gain control

In ideal case, this path loss gain **-△L** is zero. In real case, the path loss gain is none zero and depends on the radio channel. It means that the gain loss is exist if **△L** is a positive value. And the transceiver gain is above the default value if **△L** is negative. In order to evaluate a suitable gain setting by the request gain, the path loss of RF received path should be compensated at first. That's

$$\textbf{Request Gain} = \textbf{Setting Gain} - \textbf{Path Loss Gain}$$
$$= G' - \triangle L$$
$$\Rightarrow \textbf{Setting Gain} = \textbf{Request Gain} + \triangle L$$

5.2 Table of Path Loss

Path loss gain is variant with different radio channel. But the value of path loss gain should be the same in a continuous ARFCN group. So a table describing the path loss gain is constructed by ARFCN groups. The path loss tables are located in two places. One is in file **m12193.c** and the other one is in the flash on target. The path loss tables defined in **m12193.c** could be considered as the default value for all MS with the image code built in. However, it will be overwrite by unique path loss value for the specific MS, if FDM task is enabled. That means FDM keeps the calibrated data, and set by META tool during calibration test.

The general format of the path loss gain table is as following.

```
sAGCGAINOFFSET  AGC_PATHLOSS_bandname[ PLTABLE_SIZE ] =
{
    {   arfcn1,  GAINLOSS( loss_gain1 ) },
    {   arfcn2,  GAINLOSS( loss_gain2 ) },
    {   arfcn3,  GAINLOSS( loss_gain3 ) },
    {      :  ,        :              } },
    {   arfcnN,  GAINLOSS( loss_gainN) },
    /*--------------------------------*/
```

```
                   { TABLE_END }
                };
```

The rules of filling the table are listed as follow:

(1) If the table is used for GSM band , the *bandname* is **GSM900** and the base *arfcn0* is 0.
If the table is used for DCS band , the *bandname* is **DCS1800** and the base *arfcn0* is 512.
If the table is used for PCS band , the *bandname* is **PCS1900** and the base *arfcn0* is 512.

(2) The table means that
The path loss gain of ARFCN group [*arfcn0 ~arfcn1*] is *loss_gain1*.
The path loss gain of ARFCN group [*arfcn1+1~arfcn2*] is *loss_gain2*.
The path loss gain of ARFCN group [*arfcn2+1~arfcn3*] is *loss_gain3*.
:
The path loss gain of ARFCN group [*arfcn(N-1)+1~arfcnN*] is *loss_gainN*.

(3) The relation of ARFCN is *arfcn1 < arfcn2 < arfcn3 < .... < arfcnN*

(4) The band can be separated into ARFCN groups as many as user prefer. But the maximum group count is 12. This means the number of ARFCN group count *N* should not excess 12.

(5) At the end of the table, the value **TABLE_END** (-1) is needed to be filled.

For example, the path loss gain table of reference MT6119C RF module is constructed as below.

(1) In GSM band:
The path loss gain of arfcn 0~ 43 is 0.0 dB
The path loss gain of arfcn 44~ 124 is 1.0 dB
The path loss gain of arfcn 975~1024 is 1.3 dB
(2) In DCS band:
The path loss gain of arfcn 512~ 601 is 0.2 dB
The path loss gain of arfcn 602~ 666 is 1.3 dB
The path loss gain of arfcn 667~ 723 is 1.0 dB
The path loss gain of arfcn 724~ 885 is 1.5 dB
(3) In PCS band:
The path loss gain of arfcn 512~ 810 is 1.0 dB

The path loss table is established as follow:
(1) For GSM band:

```
sAGCGAINOFFSET  AGC_PATHLOSS_GSM900[ PLTABLE_SIZE ] =
{
   {    43,  GAINLOSS( 0.0 ) },
   {  124,  GAINLOSS( 1.0 )  },
   { 1024,  GAINLOSS( 1.3 )  },
   /*------------------------*/
   { TABLE_END }
};
```

(2) For DCS band:

```
sAGCGAINOFFSET  AGC_PATHLOSS_DCS1800[ PLTABLE_SIZE ] =
{
   {   601,  GAINLOSS( 0.2 ) },
  {   666,  GAINLOSS( 1.3 )   },
  {   723,  GAINLOSS( 1.0 )   },
  {   885,  GAINLOSS( 1.5 )   },
  /*------------------------*/
  { TABLE_END }
};
```

(3) For PCS band:

```
sAGCGAINOFFSET  AGC_PATHLOSS_PCS1900[ PLTABLE_SIZE ] =
{
  {    810,  GAINLOSS( 1.0 ) },
  /*------------------------*/
  { TABLE_END }
};
```

6 TX Power Ramp Profile

This section describes the transmission power ramp profile. The profile needs to be modified to fit the ramp shape and transmit level of the RF module.

6.1 Power Ramp

A set of 16 APC DAC data is used for ramp up or ramp down profile. When APC event occurs, MT62xx chip generates 32 interpolated data based on the 16 data and scaled them by the power level. Then Baseband APC DAC output generates specified voltage QB by QB. So the 32 APC Data are sent within 32 QB.

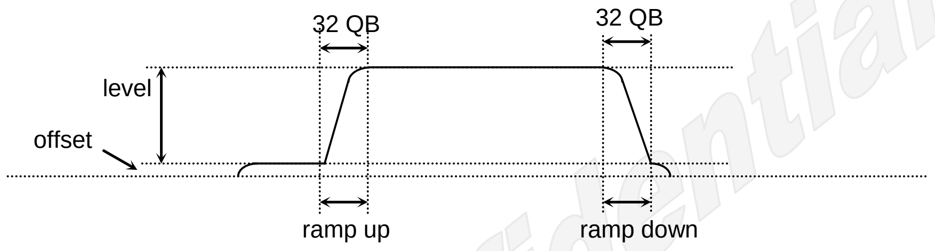


Figure 46 Power Ramp Shape Diagram

An APC DC offset before the power ramp is generated to let ramp up more smoothly. The power ramp profile is scaled by the power level and summation with this DC offset. The APC bus of MT62xx chip is 10-bit width. But the setting value of APC data is 8-bit width. The relationship between the DC offset, Power Level, 16 APC data and 32 generated APC DAC output is as following:

$$\text{APC\_Scale} = \text{APC\_POWER\_LEVEL}/1024$$
$$\text{APC\_DAC\_OUTPUT}_0 = \text{APC\_DC\_Offset} + \text{APC\_Scale}*(\text{APC\_DATA}_1 *4)$$
$$\text{APC\_DAC\_OUTPUT}_{2k+1} = \text{APC\_DC\_Offset} + \text{APC\_Scale}*(\text{APC\_DATA}_k *4)$$
$$\text{APC\_DAC\_OUTPUT}_{2k} = \text{APC\_DC\_Offset} + \text{APC\_Scale}*(\text{APC\_DATA}_{k-1} *4+\text{APC\_DATA}_k *4)/2$$
where k = 0 ~ 15

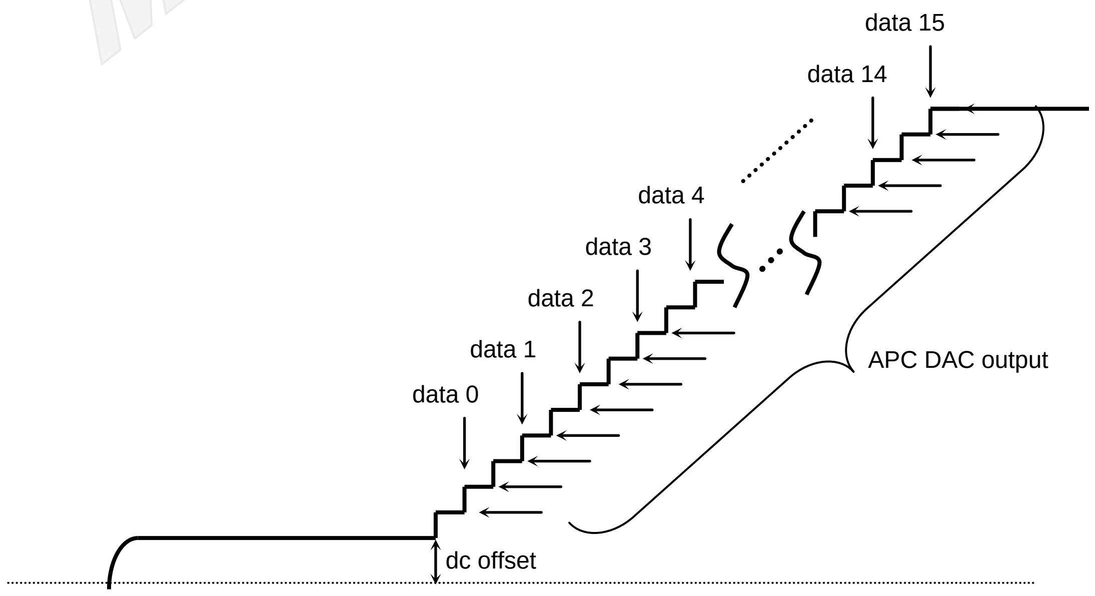


Figure 47 Ramp Up Shape

The concept of setting ramp profile is depicted by the following figure.

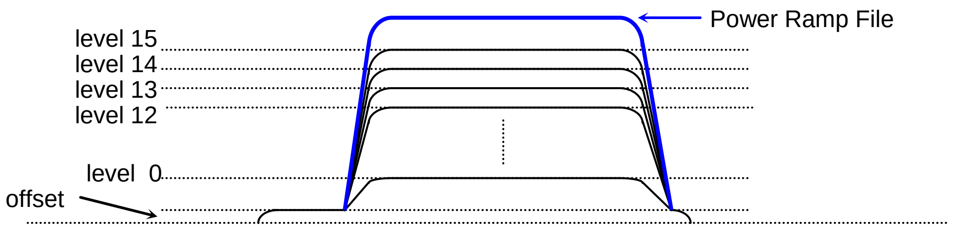


Figure 48 Power Ramp Scaled by Power Level

6.2 Table of Power Ramp Profile

In our design, ramp profile is selectable based on RF band and TX power level. 16 ramp profiles of each PCL are supported for each band. A power ramp profile should indicate the value of each power level and ramp shape.

The general format of the table is as following.

```
sRAMPDATA  band_RampData =
{
   /*-----------------------------------------------------------------------------------------*/
   /* lowest power */
   ((APC_DC_OFFSET)<<8) | lowest_power_level,
   /*-----------------------------------------------------------------------------------------*/
   /* power level  */
   /*  5,  7,  9, 11, 13, 15, 17, 19, 21, 23,  25,  27,  29,  31,  33,  35  dBm  for GSM band     */
   /*  0,  2,  4,  6,  8, 10, 12, 14, 16, 18,  20,  22,  24,  26,  28,  30  dBm  for DCS/PCS band */
   { pl0,pl1,pl2,pl3,pl4,pl5,pl6,pl7,pl8,pl9,pl10,pl11,pl12,pl13,pl14,pl15 },
   /*-----------------------------------------------------------------------------------------*/
   {  /* profile 0   : GSM(5dBm)   DCS/PCS(0dBm)     */
      {  /* ramp up   */ {{ pu0,pu1,pu2,pu3,pu4,pu5,pu6,pu7,pu8,pu9,pu10,pu11,pu12,pu13,pu14,pu15 },
         /* ramp down */  { pd0,pd1,pd2,pd3,pd4,pd5,pd6,pd7,pd8,pd9,pd10,pd11,pd12,pd13,pd14,pd15 }}
      }, /*-----------------------------------------------------------------------------------*/
      /* profile 1   : GSM(7dBm)   DCS/PCS(2dBm)     */
      {  /* ramp up   */ {{ pu0,pu1,pu2,pu3,pu4,pu5,pu6,pu7,pu8,pu9,pu10,pu11,pu12,pu13,pu14,pu15 },
         /* ramp down */  { pd0,pd1,pd2,pd3,pd4,pd5,pd6,pd7,pd8,pd9,pd10,pd11,pd12,pd13,pd14,pd15 }}
      }, /*-----------------------------------------------------------------------------------*/
      /* profile 2   : GSM(9dBm)   DCS/PCS(4dBm)     */
      {  /* ramp up   */ {{ pu0,pu1,pu2,pu3,pu4,pu5,pu6,pu7,pu8,pu9,pu10,pu11,pu12,pu13,pu14,pu15 },
         /* ramp down */  { pd0,pd1,pd2,pd3,pd4,pd5,pd6,pd7,pd8,pd9,pd10,pd11,pd12,pd13,pd14,pd15 }}
      }, /*-----------------------------------------------------------------------------------*/
      /* profile 3   : GSM(11dBm)   DCS/PCS(6dBm)     */
      {  /* ramp up   */ {{ pu0,pu1,pu2,pu3,pu4,pu5,pu6,pu7,pu8,pu9,pu10,pu11,pu12,pu13,pu14,pu15 },
         /* ramp down */  { pd0,pd1,pd2,pd3,pd4,pd5,pd6,pd7,pd8,pd9,pd10,pd11,pd12,pd13,pd14,pd15 }}
      }, /*-----------------------------------------------------------------------------------*/
      /* profile 4   : GSM(13dBm)   DCS/PCS(8dBm)     */
      {  /* ramp up   */ {{ pu0,pu1,pu2,pu3,pu4,pu5,pu6,pu7,pu8,pu9,pu10,pu11,pu12,pu13,pu14,pu15 },
         /* ramp down */  { pd0,pd1,pd2,pd3,pd4,pd5,pd6,pd7,pd8,pd9,pd10,pd11,pd12,pd13,pd14,pd15 }}
      }, /*-----------------------------------------------------------------------------------*/
      /* profile 5   : GSM(15dBm)   DCS/PCS(10dBm)     */
      {  /* ramp up   */ {{ pu0,pu1,pu2,pu3,pu4,pu5,pu6,pu7,pu8,pu9,pu10,pu11,pu12,pu13,pu14,pu15 },
         /* ramp down */  { pd0,pd1,pd2,pd3,pd4,pd5,pd6,pd7,pd8,pd9,pd10,pd11,pd12,pd13,pd14,pd15 }}
      }, /*-----------------------------------------------------------------------------------*/
      /* profile 6   : GSM(17dBm)   DCS/PCS(12dBm)     */
      {  /* ramp up   */ {{ pu0,pu1,pu2,pu3,pu4,pu5,pu6,pu7,pu8,pu9,pu10,pu11,pu12,pu13,pu14,pu15 },
         /* ramp down */  { pd0,pd1,pd2,pd3,pd4,pd5,pd6,pd7,pd8,pd9,pd10,pd11,pd12,pd13,pd14,pd15 }}
      }, /*-----------------------------------------------------------------------------------*/
      /* profile 7   : GSM(19dBm)   DCS/PCS(14dBm)      */
      {  /* ramp up   */ {{ pu0,pu1,pu2,pu3,pu4,pu5,pu6,pu7,pu8,pu9,pu10,pu11,pu12,pu13,pu14,pu15 },
         /* ramp down */  { pd0,pd1,pd2,pd3,pd4,pd5,pd6,pd7,pd8,pd9,pd10,pd11,pd12,pd13,pd14,pd15 }}
      }, /*-------------------------------------------------------------------------------------*/
      /* profile 8   : GSM(21dBm)   DCS/PCS(16dBm)      */
      {  /* ramp up   */ {{ pu0,pu1,pu2,pu3,pu4,pu5,pu6,pu7,pu8,pu9,pu10,pu11,pu12,pu13,pu14,pu15 },
         /* ramp down */  { pd0,pd1,pd2,pd3,pd4,pd5,pd6,pd7,pd8,pd9,pd10,pd11,pd12,pd13,pd14,pd15 }}
      }, /*-------------------------------------------------------------------------------------*/
      /* profile 9   : GSM(23dBm)   DCS/PCS(18dBm)      */
      {  /* ramp up   */ {{ pu0,pu1,pu2,pu3,pu4,pu5,pu6,pu7,pu8,pu9,pu10,pu11,pu12,pu13,pu14,pu15 },
         /* ramp down */  { pd0,pd1,pd2,pd3,pd4,pd5,pd6,pd7,pd8,pd9,pd10,pd11,pd12,pd13,pd14,pd15 }}
      }, /*-------------------------------------------------------------------------------------*/
      /* profile 10   : GSM(25dBm)   DCS/PCS(20dBm)      */
      {  /* ramp up   */ {{ pu0,pu1,pu2,pu3,pu4,pu5,pu6,pu7,pu8,pu9,pu10,pu11,pu12,pu13,pu14,pu15 },
         /* ramp down */  { pd0,pd1,pd2,pd3,pd4,pd5,pd6,pd7,pd8,pd9,pd10,pd11,pd12,pd13,pd14,pd15 }}
      }, /*-------------------------------------------------------------------------------------*/
      /* profile 11   : GSM(27dBm)   DCS/PCS(22dBm)      */
      {  /* ramp up   */ {{ pu0,pu1,pu2,pu3,pu4,pu5,pu6,pu7,pu8,pu9,pu10,pu11,pu12,pu13,pu14,pu15 },
         /* ramp down */  { pd0,pd1,pd2,pd3,pd4,pd5,pd6,pd7,pd8,pd9,pd10,pd11,pd12,pd13,pd14,pd15 }}
      }, /*-------------------------------------------------------------------------------------*/
      /* profile 12   : GSM(29dBm)   DCS/PCS(24dBm)      */
      {  /* ramp up   */ {{ pu0,pu1,pu2,pu3,pu4,pu5,pu6,pu7,pu8,pu9,pu10,pu11,pu12,pu13,pu14,pu15 },
         /* ramp down */  { pd0,pd1,pd2,pd3,pd4,pd5,pd6,pd7,pd8,pd9,pd10,pd11,pd12,pd13,pd14,pd15 }}
      }, /*-------------------------------------------------------------------------------------*/
      /* profile 13   : GSM(31dBm)   DCS/PCS(26dBm)      */
      {  /* ramp up   */ {{ pu0,pu1,pu2,pu3,pu4,pu5,pu6,pu7,pu8,pu9,pu10,pu11,pu12,pu13,pu14,pu15 },
         /* ramp down */  { pd0,pd1,pd2,pd3,pd4,pd5,pd6,pd7,pd8,pd9,pd10,pd11,pd12,pd13,pd14,pd15 }}
      }, /*-------------------------------------------------------------------------------------*/
      /* profile 14   : GSM(33dBm)   DCS/PCS(28dBm)      */
      {  /* ramp up   */ {{ pu0,pu1,pu2,pu3,pu4,pu5,pu6,pu7,pu8,pu9,pu10,pu11,pu12,pu13,pu14,pu15 },
         /* ramp down */  { pd0,pd1,pd2,pd3,pd4,pd5,pd6,pd7,pd8,pd9,pd10,pd11,pd12,pd13,pd14,pd15 }}
      }, /*-------------------------------------------------------------------------------------*/
      /* profile 15   : GSM(35dBm)   DCS/PCS(30dBm)      */
      {  /* ramp up   */ {{ pu0,pu1,pu2,pu3,pu4,pu5,pu6,pu7,pu8,pu9,pu10,pu11,pu12,pu13,pu14,pu15 },
         /* ramp down */  { pd0,pd1,pd2,pd3,pd4,pd5,pd6,pd7,pd8,pd9,pd10,pd11,pd12,pd13,pd14,pd15 }}
      }, /*-------------------------------------------------------------------------------------*/
   }
   /* PCL weighting table   */
  {  /*  max arfcn   , mid_level ,  hi_weight              ,  lo_weight        */
      {    arfcn1    ,  pcl1     , WEIGHT(hi_weight1)    , WEIGHT(lo_weight1)  },
      {    arfcn2    ,  pcl2     , WEIGHT(hi_weight2)    , WEIGHT(lo_weight2)  },
      {    arfcn3    ,  pcl3     , WEIGHT(hi_weight3)    , WEIGHT(lo_weight3)  },
      {       :      ,    :      ,      :          ,      :        },
      /*-------------------------------------------------------*/
      { TABLE_END }
  },
};
```

The rules of filling the table are as follow:

(1) The name *power level* used here is the transmit power level in unit dBm, not the level value specified in GSM Spec. 05.05 .

(2) If the table is used for GSM band , the `bandname` is **GSM** and the `lowest_power_level` is 5 dBm.
If the table is used for DCS band , the `bandname` is **DCS** and the `lowest_power_level` is 0 dBm.
If the table is used for PCS band , the `bandname` is **PCS** and the `lowest_power_level` is 0 dBm.

(3) The `APC_DC_OFFSET` is the APC dc offset used in all profile.

(4) The power level pl0~pl15 are power level in 10-bit numeric value.
In GSM band, the value of {`pl0,pl1,pl2,...,pl15`} are mapping to [5Bm,7dBm,9dBm,...,35 dBm]
In DCS band, the value of {`pl0,pl1,pl2,...,pl15`} are mapping to [0Bm,2dBm,4dBm,...,30 dBm]
In PCS band, the value of {`pl0,pl1,pl2,...,pl15`} are mapping to [0Bm,2dBm,4dBm,...,30 dBm]

(5) The 16 ramp up data set [`pu0,pu1,...,pu15`] is the ramp up profile with 8-bit numeric value.
The 16 ramp down data set [`pd0,pd1,...,pd15`] is the ramp down profile with 8-bit numeric value.

(6) PCL weighting table is used to set different PCL value by sub-band. If TX arfcn is in `arfcn(i-1)+1` ~ `arfcn(i)` and TX PCL is equal or greater than `pcl(i)` , the PCL setting is times the `hi_weight(i)`. If TX arfcn is in `arfcn(i-1)+1` ~ `arfcn(i)` and TX PCL is less than `pcl(i)` , the PCL setting is times the `hi_weight(i)`.

(7) 16 ramp up data set [`pu0,pu1,...,pu15`] is the ramp up profile with 8-bit numeric value. The 16 ramp down data set [`pd0,pd1,...,pd15`] is the ramp down profile with 8-bit numeric value.

So if the transmit power level is equal or lower than `power_level(i)`, use profile i. The APC bus out is

APC_Scale = `power_level(i)`

APC_DAC_OUTPUT_0 = `dc_offset(i)` + APC_Scale*`p(0)`/256

$\text{APC\_DAC\_OUTPUT}_{2k+1}$ = `dc_offset(i)` + APC_Scale*`p(k)`/256

$\text{APC\_DAC\_OUTPUT}_{2k}$ = `dc_offset(i)` + APC_Scale*(`p(k-1)` +`p(k)`)/512

where k = 0 ~ 15
i = 0 ~ 3
p(n) is pun or pdn in profile i

6.3 Table of EPSK Power Ramp Profile

In our design, EPSK ramp profile is selectable based on RF band and TX power level. 16 ramp profiles of each PCL are supported for each band. A power ramp profile should indicate the value of each power level and ramp shape.

The general format of the table is as following.

```
sRAMPDATA  band_RampData_EPSK =
{
   /*------------------------------------------------------------------------------*/
   /* lowest power */
     ((Band_INTERSLOT_LOWEST_DAC)<<18) | ((APC_DC_OFFSET)<<8) | 5,
   /*------------------------------------------------------------------------------*/
   /* power level  */
   /*  5,  7,  9, 11, 13, 15, 17, 19, 21, 23,  25,  27,  29,  31,  33,  35  dBm  for GSM band     */
   /*  0,  2,  4,  6,  8, 10, 12, 14, 16, 18,  20,  22,  24,  26,  28,  30  dBm  for DCS/PCS band */
   { pl0,pl1,pl2,pl3,pl4,pl5,pl6,pl7,pl8,pl9,pl10,pl11,pl12,pl13,pl14,pl15 },
   /*------------------------------------------------------------------------------*/
   {  /* profile 0   : GSM(5dBm)   DCS/PCS(0dBm)      */
      {  /* ramp up   */ {{ pu0,pu1,pu2,pu3,pu4,pu5,pu6,pu7,pu8,pu9,pu10,pu11,pu12,pu13,pu14,pu15 },
         /* ramp down */  { pd0,pd1,pd2,pd3,pd4,pd5,pd6,pd7,pd8,pd9,pd10,pd11,pd12,pd13,pd14,pd15 }}
      }, /*------------------------------------------------------------------------*/
      /* profile 1   : GSM(7dBm)   DCS/PCS(2dBm)      */
      {  /* ramp up   */ {{ pu0,pu1,pu2,pu3,pu4,pu5,pu6,pu7,pu8,pu9,pu10,pu11,pu12,pu13,pu14,pu15 },
         /* ramp down */  { pd0,pd1,pd2,pd3,pd4,pd5,pd6,pd7,pd8,pd9,pd10,pd11,pd12,pd13,pd14,pd15 }}
      }, /*------------------------------------------------------------------------*/
      /* profile 2   : GSM(9dBm)   DCS/PCS(4dBm)      */
      {  /* ramp up   */ {{ pu0,pu1,pu2,pu3,pu4,pu5,pu6,pu7,pu8,pu9,pu10,pu11,pu12,pu13,pu14,pu15 },
         /* ramp down */  { pd0,pd1,pd2,pd3,pd4,pd5,pd6,pd7,pd8,pd9,pd10,pd11,pd12,pd13,pd14,pd15 }}
      }, /*------------------------------------------------------------------------*/
      /* profile 3   : GSM(11dBm)   DCS/PCS(6dBm)      */
      {  /* ramp up   */ {{ pu0,pu1,pu2,pu3,pu4,pu5,pu6,pu7,pu8,pu9,pu10,pu11,pu12,pu13,pu14,pu15 },
         /* ramp down */  { pd0,pd1,pd2,pd3,pd4,pd5,pd6,pd7,pd8,pd9,pd10,pd11,pd12,pd13,pd14,pd15 }}
      }, /*------------------------------------------------------------------------*/
      /* profile 4   : GSM(13dBm)   DCS/PCS(8dBm)      */
      {  /* ramp up   */ {{ pu0,pu1,pu2,pu3,pu4,pu5,pu6,pu7,pu8,pu9,pu10,pu11,pu12,pu13,pu14,pu15 },
         /* ramp down */  { pd0,pd1,pd2,pd3,pd4,pd5,pd6,pd7,pd8,pd9,pd10,pd11,pd12,pd13,pd14,pd15 }}
      }, /*------------------------------------------------------------------------*/
      /* profile 5   : GSM(15dBm)   DCS/PCS(10dBm)      */
      {  /* ramp up   */ {{ pu0,pu1,pu2,pu3,pu4,pu5,pu6,pu7,pu8,pu9,pu10,pu11,pu12,pu13,pu14,pu15 },
         /* ramp down */  { pd0,pd1,pd2,pd3,pd4,pd5,pd6,pd7,pd8,pd9,pd10,pd11,pd12,pd13,pd14,pd15 }}
      }, /*------------------------------------------------------------------------*/
      /* profile 6   : GSM(17dBm)   DCS/PCS(12dBm)      */
      {  /* ramp up    */ {{ pu0,pu1,pu2,pu3,pu4,pu5,pu6,pu7,pu8,pu9,pu10,pu11,pu12,pu13,pu14,pu15 },
         /* ramp down */  { pd0,pd1,pd2,pd3,pd4,pd5,pd6,pd7,pd8,pd9,pd10,pd11,pd12,pd13,pd14,pd15 }}
      }, /*-------------------------------------------------------------------------------------*/
      /* profile 7   : GSM(19dBm)   DCS/PCS(14dBm)      */
      {  /* ramp up    */ {{ pu0,pu1,pu2,pu3,pu4,pu5,pu6,pu7,pu8,pu9,pu10,pu11,pu12,pu13,pu14,pu15 },
         /* ramp down */  { pd0,pd1,pd2,pd3,pd4,pd5,pd6,pd7,pd8,pd9,pd10,pd11,pd12,pd13,pd14,pd15 }}
      }, /*-------------------------------------------------------------------------------------*/
      /* profile 8   : GSM(21dBm)   DCS/PCS(16dBm)      */
      {  /* ramp up    */ {{ pu0,pu1,pu2,pu3,pu4,pu5,pu6,pu7,pu8,pu9,pu10,pu11,pu12,pu13,pu14,pu15 },
         /* ramp down */  { pd0,pd1,pd2,pd3,pd4,pd5,pd6,pd7,pd8,pd9,pd10,pd11,pd12,pd13,pd14,pd15 }}
      }, /*-------------------------------------------------------------------------------------*/
      /* profile 9   : GSM(23dBm)   DCS/PCS(18dBm)      */
      {  /* ramp up    */ {{ pu0,pu1,pu2,pu3,pu4,pu5,pu6,pu7,pu8,pu9,pu10,pu11,pu12,pu13,pu14,pu15 },
         /* ramp down */  { pd0,pd1,pd2,pd3,pd4,pd5,pd6,pd7,pd8,pd9,pd10,pd11,pd12,pd13,pd14,pd15 }}
      }, /*-------------------------------------------------------------------------------------*/
      /* profile 10   : GSM(25dBm)   DCS/PCS(20dBm)      */
      {  /* ramp up    */ {{ pu0,pu1,pu2,pu3,pu4,pu5,pu6,pu7,pu8,pu9,pu10,pu11,pu12,pu13,pu14,pu15 },
         /* ramp down */  { pd0,pd1,pd2,pd3,pd4,pd5,pd6,pd7,pd8,pd9,pd10,pd11,pd12,pd13,pd14,pd15 }}
      }, /*-------------------------------------------------------------------------------------*/
      /* profile 11   : GSM(27dBm)   DCS/PCS(22dBm)      */
      {  /* ramp up    */ {{ pu0,pu1,pu2,pu3,pu4,pu5,pu6,pu7,pu8,pu9,pu10,pu11,pu12,pu13,pu14,pu15 },
         /* ramp down */  { pd0,pd1,pd2,pd3,pd4,pd5,pd6,pd7,pd8,pd9,pd10,pd11,pd12,pd13,pd14,pd15 }}
      }, /*-------------------------------------------------------------------------------------*/
      /* profile 12   : GSM(29dBm)   DCS/PCS(24dBm)      */
      {  /* ramp up    */ {{ pu0,pu1,pu2,pu3,pu4,pu5,pu6,pu7,pu8,pu9,pu10,pu11,pu12,pu13,pu14,pu15 },
         /* ramp down */  { pd0,pd1,pd2,pd3,pd4,pd5,pd6,pd7,pd8,pd9,pd10,pd11,pd12,pd13,pd14,pd15 }}
      }, /*-------------------------------------------------------------------------------------*/
      /* profile 13   : GSM(31dBm)   DCS/PCS(26dBm)      */
      {  /* ramp up    */ {{ pu0,pu1,pu2,pu3,pu4,pu5,pu6,pu7,pu8,pu9,pu10,pu11,pu12,pu13,pu14,pu15 },
         /* ramp down */  { pd0,pd1,pd2,pd3,pd4,pd5,pd6,pd7,pd8,pd9,pd10,pd11,pd12,pd13,pd14,pd15 }}
      }, /*-------------------------------------------------------------------------------------*/
      /* profile 14   : GSM(33dBm)   DCS/PCS(28dBm)      */
      {  /* ramp up    */ {{ pu0,pu1,pu2,pu3,pu4,pu5,pu6,pu7,pu8,pu9,pu10,pu11,pu12,pu13,pu14,pu15 },
         /* ramp down */  { pd0,pd1,pd2,pd3,pd4,pd5,pd6,pd7,pd8,pd9,pd10,pd11,pd12,pd13,pd14,pd15 }}
      }, /*-------------------------------------------------------------------------------------*/
      /* profile 15   : GSM(35dBm)   DCS/PCS(30dBm)      */
      {  /* ramp up    */ {{ pu0,pu1,pu2,pu3,pu4,pu5,pu6,pu7,pu8,pu9,pu10,pu11,pu12,pu13,pu14,pu15 },
         /* ramp down */  { pd0,pd1,pd2,pd3,pd4,pd5,pd6,pd7,pd8,pd9,pd10,pd11,pd12,pd13,pd14,pd15 }}
      }, /*-------------------------------------------------------------------------------------*/
   }
   /* PCL weighting table     */
  {  /*  max arfcn   , mid_level  ,  hi_weight             ,  lo_weight        */
      {    arfcn1    ,  pcl1      , WEIGHT(hi_weight1)    , WEIGHT(lo_weight1) },
      {    arfcn2    ,  pcl2      , WEIGHT(hi_weight2)    , WEIGHT(lo_weight2) },
      {    arfcn3    ,  pcl3      , WEIGHT(hi_weight3)    , WEIGHT(lo_weight3) },
      {       :      ,    :       ,       :              ,      :         },
      /*-----------------------------------------------------------*/
      { TABLE_END }
  },
};
```

The rules of filling the table are as follow:

(1) The name *power level* used here is the transmit power level in unit dBm, not the level value specified in GSM Spec. 05.05 .

(2) If the table is used for GSM band , the `bandname` is **GSM** and the ***GSM_INTERSLOT_LOWEST_DAC*** is defined the interslot lowest DAC.
If the table is used for DCS band , the `bandname` is **DCS** and the ***DCS_INTERSLOT_LOWEST_DAC*** is defined the interslot lowest DAC.
If the table is used for PCS band , the `bandname` is **PCS** and the ***PCS_INTERSLOT_LOWEST_DAC*** is defined the interslot lowest DAC.

(3) The *APC_DC_OFFSET* is the APC dc offset used in all profile.

(4) The power level pl0~pl15 are power level in 10-bit numeric value.
In GSM band, the value of {`pl0,pl1,pl2,...,pl15`} are mapping to [5Bm,7dBm,9dBm,...,35 dBm]
In DCS band, the value of {`pl0,pl1,pl2,...,pl15`} are mapping to [0Bm,2dBm,4dBm,...,30 dBm]
In PCS band, the value of {`pl0,pl1,pl2,...,pl15`} are mapping to [0Bm,2dBm,4dBm,...,30 dBm]

(5) The 16 ramp up data set [`pu0,pu1,...,pu15`] is the ramp up profile with 8-bit numeric value.
The 16 ramp down data set [`pd0,pd1,...,pd15`] is the ramp down profile with 8-bit numeric value.

(8) PCL weighting table is used to set different PCL value by sub-band. If TX arfcn is in `arfcn(i-1)+1` ~ `arfcn(i)` and TX PCL is equal or greater than `pcl(i)` , the PCL setting is times the `hi_weight(i)`. If TX arfcn is in `arfcn(i-1)+1` ~ `arfcn(i)` and TX PCL is less than `pcl(i)` , the PCL setting is times the `hi_weight(i)`.

(9) 16 ramp up data set [`pu0,pu1,...,pu15`] is the ramp up profile with 8-bit numeric value.
The 16 ramp down data set [`pd0,pd1,...,pd15`] is the ramp down profile with 8-bit numeric value.

So if the transmit power level is equal or lower than `power_level(i)`, use profile i. The APC bus out is

APC_Scale = `power_level(i)`
$APC\_DAC\_OUTPUT_0$ = `dc_offset(i)` + APC_Scale*`p(0)`/256
$APC\_DAC\_OUTPUT_{2k+1}$ = `dc_offset(i)` + APC_Scale*`p(k)`/256
$APC\_DAC\_OUTPUT_{2k}$ = `dc_offset(i)` + APC_Scale*(`p(k-1)` +`p(k)`)/512
where k = 0 ~ 15
i = 0 ~ 3
p(n) is pun or pdn in profile i

The power ramp profile setting in the EPSK mode is different from the setting in the GMSK mode when the MT6162 transceiver is used. We only need to provide a constant V_{APC} value to the PA, and it will work as a linear amplifier. The ramp shape is decided from the MT6276 baseband output, and we will use different gain settings on MT6162 to provide different output power levels. The structure is shown in the following figure.

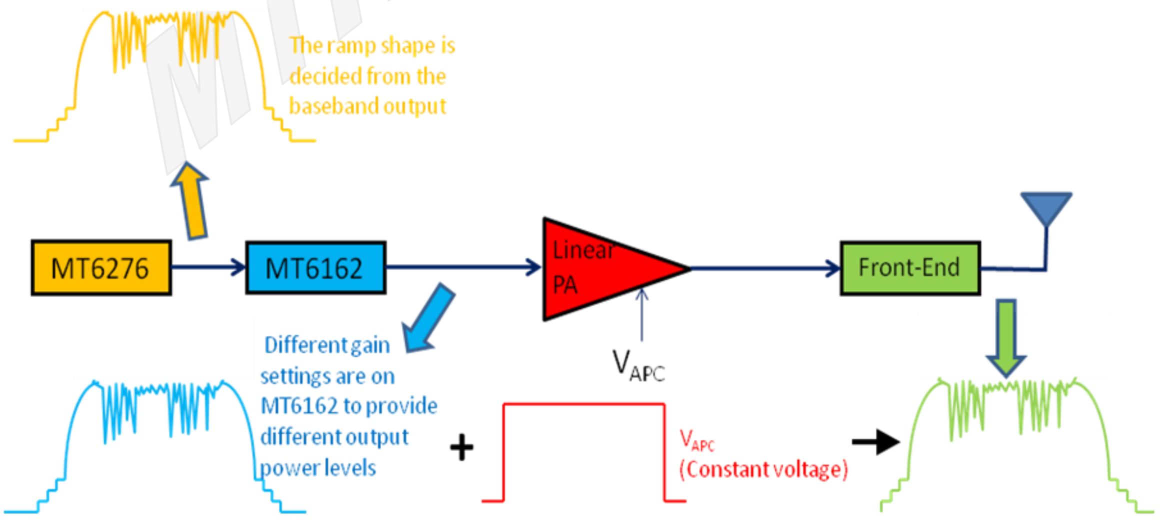

The ramp shape is decided from the baseband output
MT6276
MT6162
Linear PA
Front-End
V APC
Different gain settings are on MT6162 to provide different output power levels
+
V APC (Constant voltage)

Figure 49 The Illustration of the EPSK Signal on MT6162

The difference is the setting of 16 ramp-up [`pu0,pu1,...,pu15`] and ramp-down [`pd0,pd1,...,pd15`] data set. We will set these profiles to 255 to give a constant APC output for PA in the EPSK mode.

6.4 GPRS Inter-Slot Power Ramp Profile

The concept of GPRS Inter-slot ramp connection is depicted by the following figure:

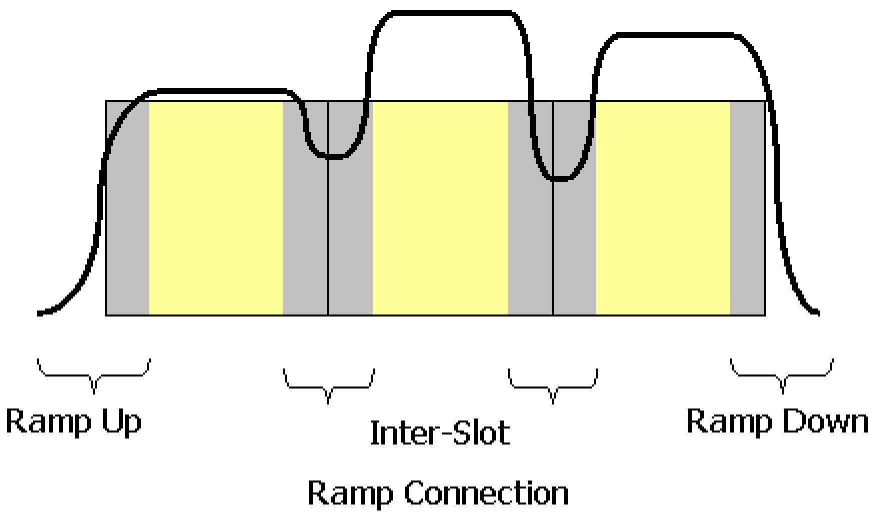
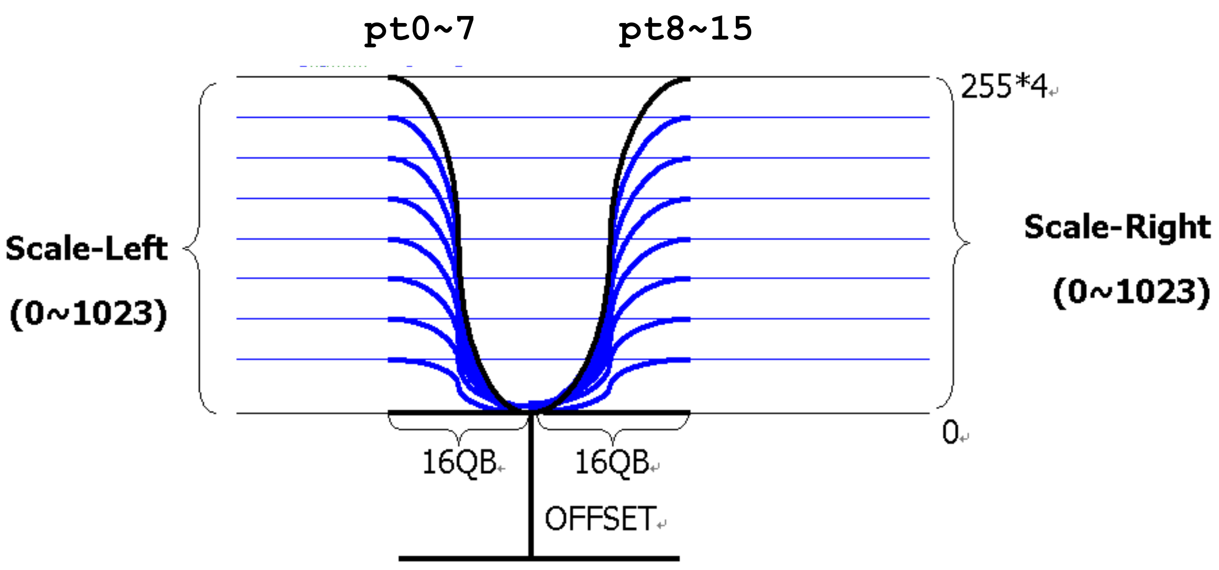


Figure 50 APC ramp shape of inter-slot TX bursts

In Layer1, a table is used to describe the inter-slot profile of each band in **m12193.c**:

```
sMIDRAMPDATA  bandname_InterRampData =
               {pt0,pt1,pt2,pt3,pt4,pt5,pt6,pt7,pt8,pt9,pt10,pt11,pt12,pt13,pt14,pt15};
```

To let ramp shape can continuously connect the1'st and 2'nd TX burst APC level, 2 scales value are use for one connection in L1. Scale 1 is used to scale pt0~pt7 to ensure the 1'st burst PCL and the inter-slot ramp shape is continuous. Scale 2 is used to scale pt0~pt7 to ensure the 2'nd burst PCL and the inter-slot ramp shape is continuous. please refer the following figure:



The inter-slot ramp profile table is fixed at RD phase, and is not necessary to adjust it during mass production.

6.5 EGPRS Inter-Slot Power Ramp Profile

The concept of EGPRS Inter-slot ramp connection is depicted by the following figure:

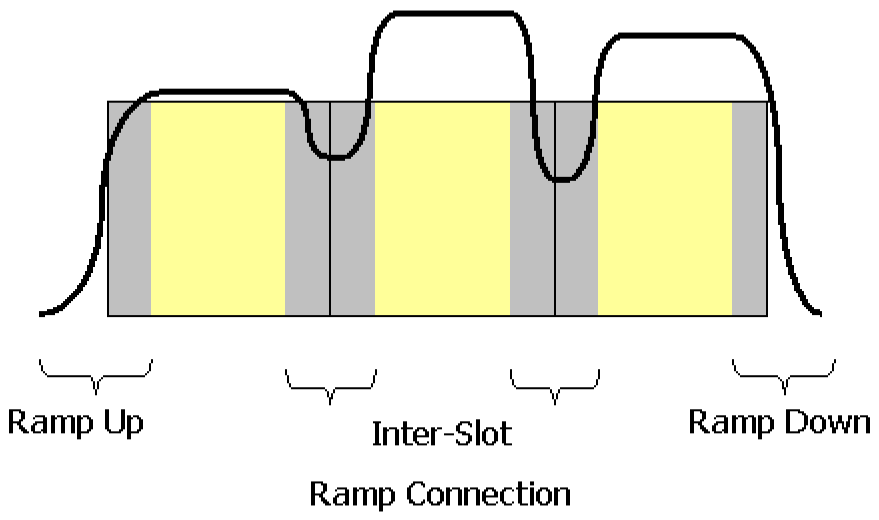


Figure 51 APC ramp shape of inter-slot TX bursts

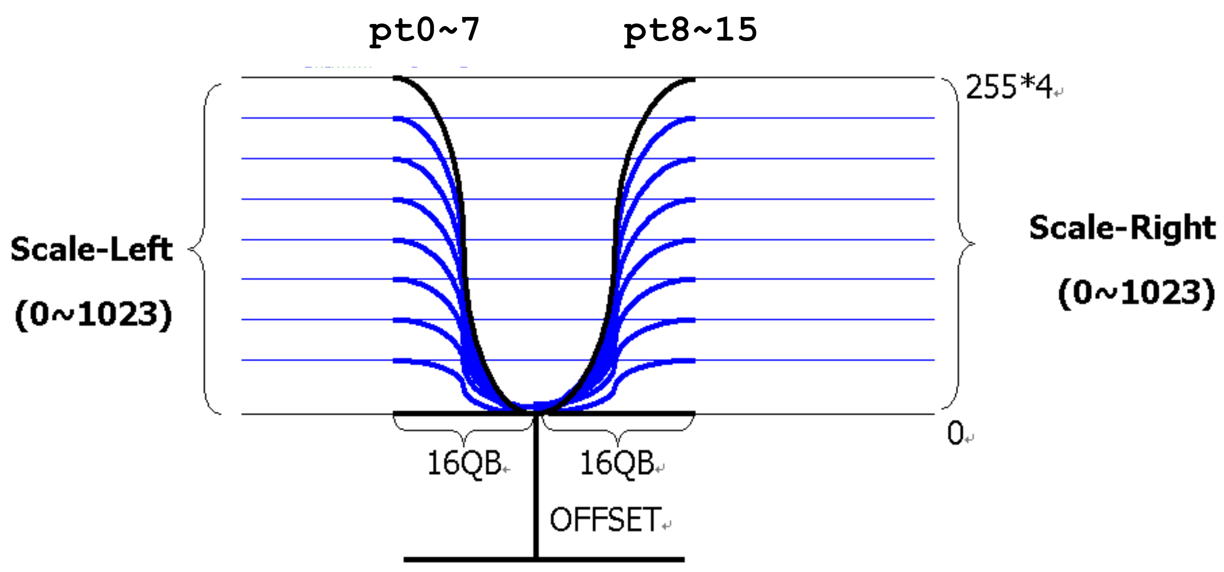
In Layer1, a table is used to describe the inter-slot profile of each band in **m12193.c**, four interslot ramp data will be used depend on the releation of multi-slots modulation type :

```
sMIDRAMPDATA  bandname_InterRampDataG_2_G =
               {pt0,pt1,pt2,pt3,pt4,pt5,pt6,pt7,pt8,pt9,pt10,pt11,pt12,pt13,pt14,pt15};
sMIDRAMPDATA  bandname_InterRampDataE_2_E =
               {pt0,pt1,pt2,pt3,pt4,pt5,pt6,pt7,pt8,pt9,pt10,pt11,pt12,pt13,pt14,pt15};
sMIDRAMPDATA  bandname_InterRampDataG_2_E =
               {pt0,pt1,pt2,pt3,pt4,pt5,pt6,pt7,pt8,pt9,pt10,pt11,pt12,pt13,pt14,pt15};
sMIDRAMPDATA  bandname_InterRampDataE_2_G =
               {pt0,pt1,pt2,pt3,pt4,pt5,pt6,pt7,pt8,pt9,pt10,pt11,pt12,pt13,pt14,pt15};
```

To let interslot ramp can pass PvT easily, interslot lowest dac can be set:

```
    #define   band_INTERSLOT_LOWEST_DAC   band_INTERSLOT_LOWEST_DAC
```

To let ramp shape can continuously connect the1'st and 2'nd TX burst APC level, 2 scales value are use for one connection in L1. Scale 1 is used to scale pt0~pt7 to ensure the 1'st burst PCL and the inter-slot ramp shape is continuous. Scale 2 is used to scale pt0~pt7 to ensure the 2'nd burst PCL and the inter-slot ramp shape is continuous. please refer the following figure:



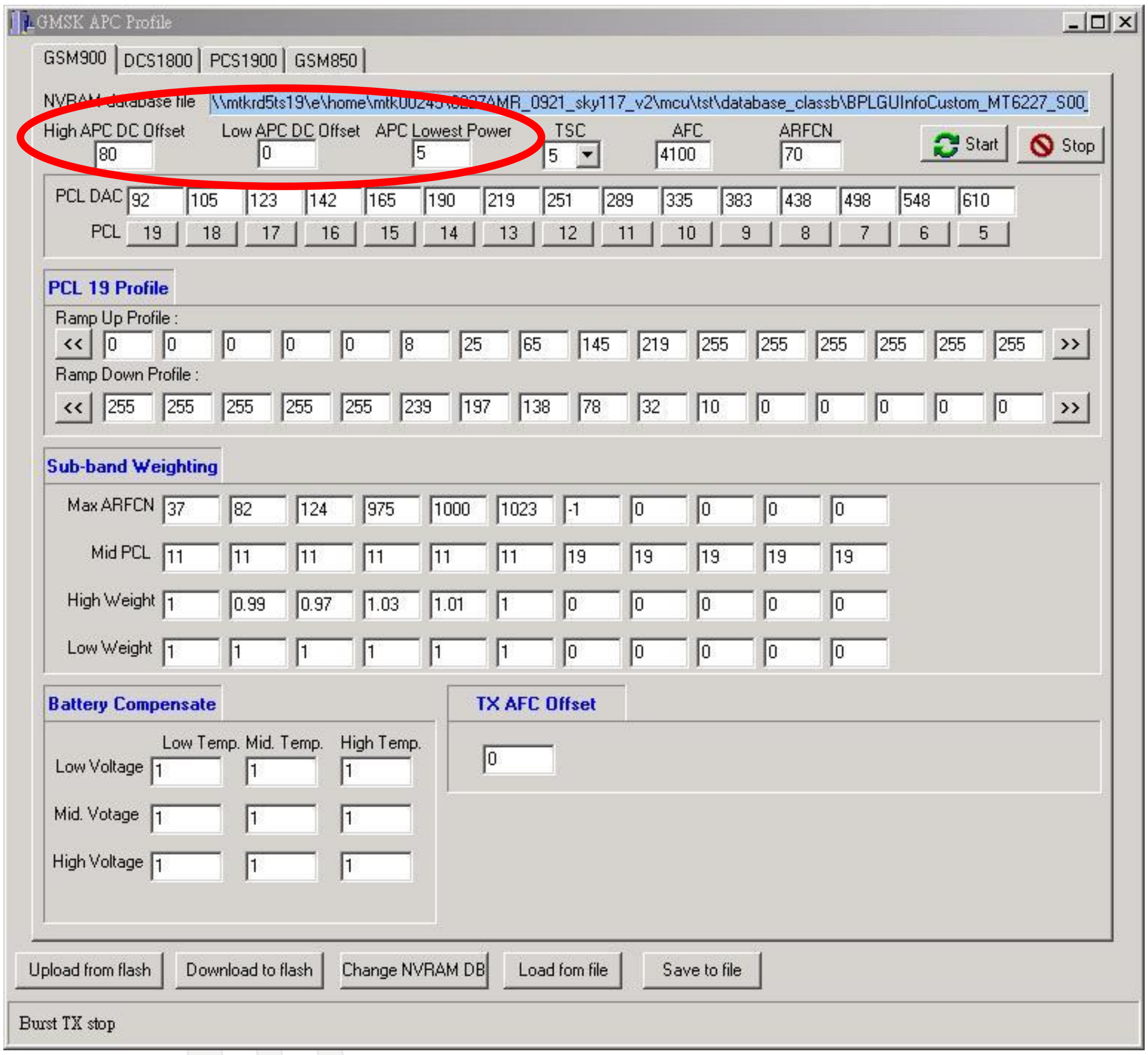
The inter-slot ramp profile table is fixed at RD phase, and is not necessary to adjust it during mass production.

6.6 Different APC DC offset turn on voltage for different TX power level

From Maui 05B SW Release, We start to support different APC DC offset turn on voltage. This is because some current sense PA ex: SKYWORKS 77328 need this feature. But due to the practicability and implementation, we only support Two different APC DC offset turn on voltage. This will meet current sense PA's requirement and does not impact our current architecture too much. If the transmit power is larger than or equal to *lowest_power_level*, *APC_DC_OFFSET_HIGH* will be applied. In the other hand, if the transmit power is smaller than *lowest_power_level*, *APC_DC_OFFSET_LOW* will be applied. In other words, *lowest_power_level* is the threshold to control when to change APC DC Offset.

```
sRAMPDATA  band_RampData =
{
   /*------------------------------------------------------------------------------------------*/
   /* lowest power */
((APC_DC_OFFSET_LOW)<<18)|(APC_DC_OFFSET_HIGH)<<8) | lowest_power_level,
}
```

You may use Meta (After version 3.7) to setup these fields. Please see following figure in the Red Circle column.



GMSK APC Profile

GSM900 | DCS1800 | PCS1900 | GSM850

NVRAM database file: \\mtkrd5ts19\e\home\mtk0024?\0227AMR_0921_sky117_v2\mcu\tst\database_classb\BPLGUInfoCustom_MT6227_S00_

High APC DC Offset	Low APC DC Offset	APC Lowest Power	TSC	AFC	ARFCN
80	0	5	5	4100	70

Start | Stop

PCL DAC	92	105	123	142	165	190	219	251	289	335	383	438	498	548	610
PCL	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5

PCL 19 Profile

Ramp Up Profile :	0	0	0	0	0	8	25	65	145	219	255	255	255	255	255	255
Ramp Down Profile :	255	255	255	255	255	239	197	138	78	32	10	0	0	0	0	0

Sub-band Weighting

Max ARFCN	37	82	124	975	1000	1023	-1	0	0	0	0
Mid PCL	11	11	11	11	11	11	19	19	19	19	19
High Weight	1	0.99	0.97	1.03	1.01	1	0	0	0	0	0
Low Weight	1	1	1	1	1	1	0	0	0	0	0

Battery Compensate

	Low Temp.	Mid. Temp.	High Temp.
Low Voltage	1	1	1
Mid. Votage	1	1	1
High Voltage	1	1	1

TX AFC Offset

0

Upload from flash | Download to flash | Change NVRAM DB | Load fom file | Save to file

Burst TX stop

7 AFC Configuration

7.1 AFC operation

In GSM, DCS and PCS mobile system, GSM05.10 specify the MS carrier frequency should be accurate to within 0.1 ppm, or accurate to within 0.1 ppm compared to signals received from the BTS. In order to compensate the frequency error, which caused by BTS frequency error or Doppler shift effect, exists between MS and BTS, AFC (automatic frequency control) resides in MS should take the responsibility to adjust MS reference clock, 13MHz or 26MHz voltage controlled crystal oscillator (VCXO), in order to track up the frequency base of the signal from BTS.

MT62XX provide a dedicated 13 bit DAC for AFC operation, please refer to MT62XX Data Sheet[1] for detail information.

7.2 TCVCXO characteristic

For AFC operation, crystal characteristic should be well known while MS is in operation. Typically, temperature compensated VCXO (TCVCXO) is used popularly in MS phone design, the following diagram illustrates the typical characteristic curve of TCVCXO in GSM900 and used on radio daughter board of MT62XX EVB. Therefore, by MT62XX EVB design, the tuning range 0~8191 of MT62XX 13bit DAC is mapping to the frequency error –8535~ 10208 Hz gotten from MT62XX report, and the slope is quite linear within 15% deviation in the full range of TCVCXO used for MT62XX DVB. By calculation, the slope is 2.367 Hz/step, which the important parameter used in AFC operation. Actually, the slope is only valid for GSM900 application, layer 1 will convert it for DCS1800 or PCS1900, automatically.

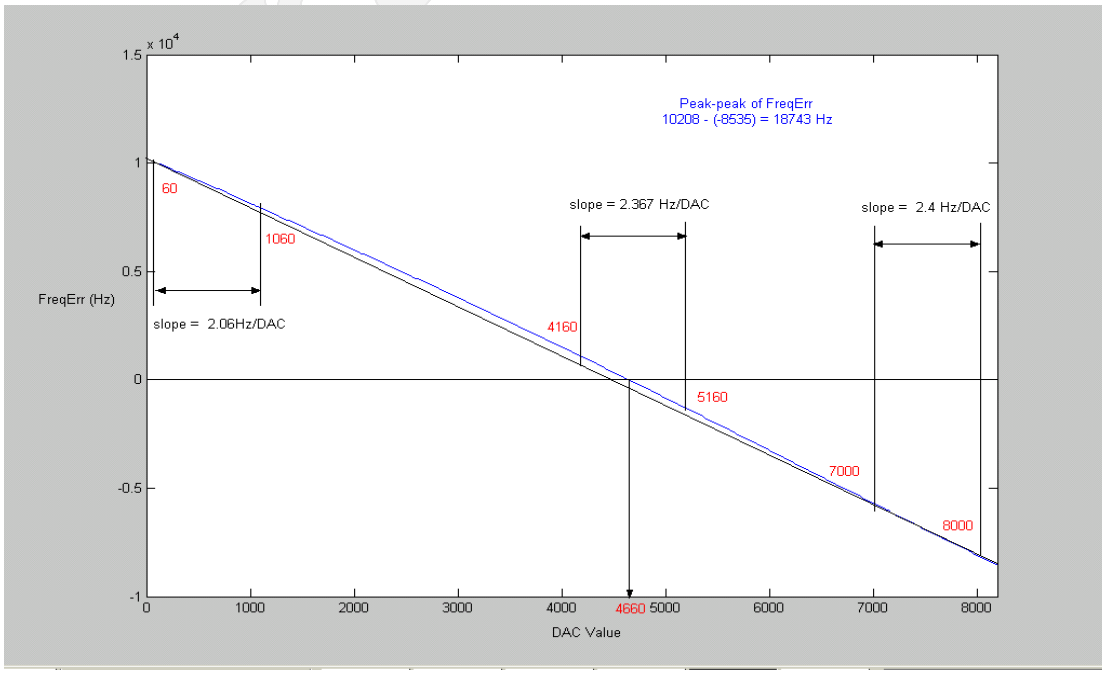


Figure 52 TXVCXO characteristic

7.3 Exported interface and constant

In this subsection, the important parameters of AFC algorithm are listed in the following.

```
  #define AFC_DP_MIN          0
  #define AFC_DP_MAX          8191
   #define  C_PSI_STA         1730
   #define  PSI_EE            4100
```

AFC_DP_MIN and AFC_DP_MAX present the minimum and maximum control value of AFC DAC, for MT62XX, 13bit DAC is supported with the tuning range of 0~8191. C_PSI_STA indicates the inverted slope of TCVCXO, their relation is $C\_PSI\_STA = 4096/slope$; therefore, 1730 is selected. PSI_EE is the default DAC value used once MS is power on, to make it easy to camp on SS or real network. The above definition is defined in the file **m12193.h**.

For each MS, the unique C_PSI_ST and PSI_EE will be acquired during calibration in mass production, and stored on FDM. Once calibrated AFC parameter C_PSI_ST and PSI_EE exists on FDM, these parameter, defined in **m12193.h**, will be overwritten by the one from FDM via the function **L1I_SetAFcData**, which is implemented in **m12194.c**.

Define

void L1I_SetAFcData(int16 calibrated_dac_default, intx calibrated_int_slope)

Description:

This function is designed to update the AFC parameter – DAC default value and inverted slope.

Parameter:

Parameter		Description
Int16 calibrated_dac_default	IN	AFC DAC default value
intx calibrated_int_slope	IN	VCXO inverted slope

8 RX ADC Dynamic Range Configuration

8.1 AGC operation

To help with receive block decoding, cell selection, sell reselection, and handover processes, RSSI measurement is necessary and important. Received signal goes through radio circuits and gets presented at the input of the ADC to be digitized and read by DSP. DSP reads the digitized receive samples, in windows controlled by MCU, and performs power calculation of the signal. This RSSI is reported to Layer 1 for each windows every frame to get accumulated and averaged. The averaged value in Layer 1 will be used to perform Automatic Gain Control (AGC) of the radio gain stages, to make sure that receive power level is in good range next time so that proper power measurement can be done and GSM protocol messages can be received properly. This averaged value is also reported to upper layer and gets reported back to the base station to help with handover decision.

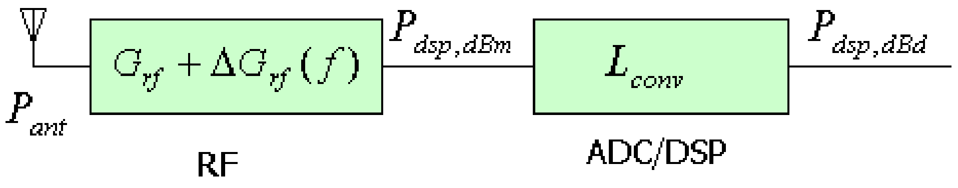


Figure 53 Downlink Receive Path

In order to depict AGC operation, the downlink signal receive path is illustrated in Figure 20, and AGC take the responsibility of tracking received signal via next received level prediction and radio gain G_{rf} adjustment, to make sure the input signal level of RX ADC located in the dynamic range of RX ADC as well as the linear range for RSSI reporting.

8.2 RX ADC characteristic

8.2.1 10+1 bit RX ADC characteristic

RX ADC of MT6208, MT6205, MT6205B are 10+1bits and 1 bit of sign bit; therefore, ideally 60 dB of dynamic range is provided for AGC tracking. However, due to noise flow exists intrinsically in receiver, the linear range of RSSI reporting is between 8~60dB. Figure 21 illustrate the ADC dynamic range for AGC and RSSI reporting, respectively. The configuration is fixed for MT62XX, not recommend to revise the definition defined in **m12194.c**.

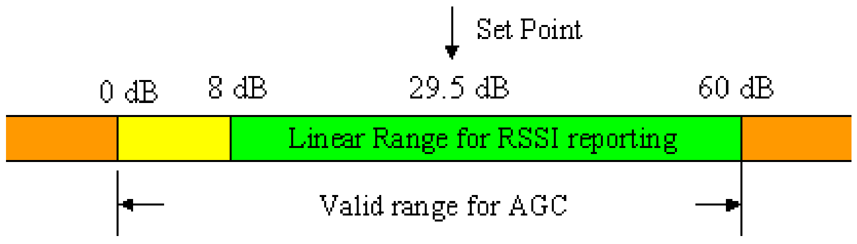


Figure 54 Dynamic range for AGC and RSSI reporting (10+1 bit RX ADC)

8.2.2 13+1 bit RX ADC characteristic

RX ADC of MT6218, MT6218B and Mt6219 are 13+1bits and 1 bit of sign bit; therefore, ideally 78 dB of dynamic range is provided for AGC tracking. If we integrate with MT6119C, the linear range of RSSI reporting is between 34~77dB due to noise flow exists intrinsically in receiver. If we integrate with RF receiver, the noise flower could be less than 8dB. Figure 22 illustrate the ADC dynamic range for AGC and RSSI reporting, respectively. The configuration is fixed for MT62XX, not recommend to revise the definition defined in **m12194.c**.

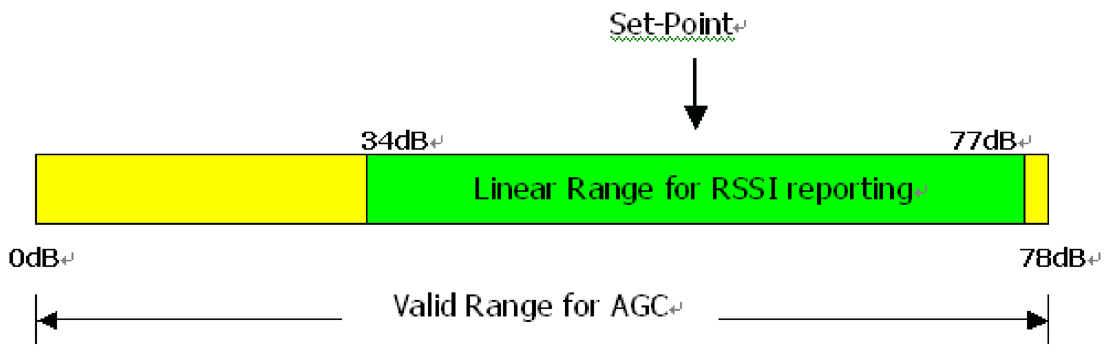


Figure 55 Dynamic range for AGC and RSSI reporting (1 v+1 bit RX ADC)

8.3 Set Point Calculation

8.3.1 Set Point Calculation with 10+1 bit RX ADC

In addition to AGC tracking and RSSI reporting, the set point should be chosen carefully. The optimum set point is dependent of radio design and DSP operation range. The recommend set point should be located in the range of 28 ~ 45 dB, and finally decided by customer's radio design. The set point DSP_SETPOINT is defined in m12193.h, and set as 45 for Hitachi Bright 2 radio and 29.5 for Bright 4 plus radio.

The bellow equation converts the signal level in mV_{p-p} at I/Q signal end output of radio to DSP_SETPOINT value defined in layer 1.

$$20\log(x) - 1.378 = DSP\_SETPOINT$$

For example, $x = 35\,mV_{p-p}$ and DSP_SETPOINT=29.5.

8.3.2 Set Point Calculation with 13+1 bit RX ADC

In addition to AGC tracking and RSSI reporting, the set point should be chosen carefully. The optimum set point is dependent of radio design and DSP operation range. The recommend set point should be located in the range of 60 dBd with MT6119C, and finally decided by customer's radio design. The set point DSP_SETPOINT is defined in m12193.h.

The bellow equation converts the signal level in mV_{p-p} at I/Q signal end output of radio to DSP_SETPOINT value defined in layer 1.

$$20\log(x) + 17.0 = DSP\_SETPOINT$$

For example, $x = 140\ mV_{p\text{-}p}$ if choose DSP_SETPOINT=60.

8.3.3 The Setpoint settings for different mode

For some RF module, the setpoint need to adjust according to the input signal lever or the operation mode in order to achieve the best modem performance. As the result, in the **m12193.h** there are two definitions named SETPOINT_GAIN_OFFSET and EDGE_SETPOINT_GAIN_OFFSET which is used to adjust the setpoint. SETPOINT_GAIN_OFFSET is used to lower the setpoint if the signal power is larger than 86dBm. For the most RF, to lower the setpoint is not needed, and as the result, we could just define the SETPOINT_GAIN_OFFSET equals 0. On the other hand, the definition EDGE_SETPOINT_GAIN_OFFSET is used to raise the setpoint for the EGPRS TBF in order to provide the sufficient SNR for the MCS9. Take the example for the SKY74137, the EDGE_SETPOINT_GAIN_OFFSET is 5 means that while the EGPRS TBF is established, the setpoint is 5dB higher than the DSP_SETPOINT definition.

9 Sleep Mode optimization

9.1 Sleep mode operation

The sleep mode operation is controlled by Layer 1. Before Layer 1 can write commands to the hardware to enter sleep mode, some sleep mode conditions need to be checked individually. The sleep mode prerequisites of RTOS and Layer 1 activities are checked in Layer 1. Additionally Layer 1 provides a interface for other software module or device driver that wants to inhibit the power saving function and keep the handset running in normal mode. The interface functions are provided in **l1sm_public.h**. The interface uses a sleep disable flag. When this flag contains a non-zero value, the system cannot enter sleep mode until it is cleared. For any software module that needs inhibiting the power saving mode shall firstly register and get a sleep mode handler from Layer 1. The API for this functionality is uint8 **L1SM_GetHandle**(void). Each sleep mode handler independently controls a bit of the Sleep Disable Flag. If a software module has already got a sm_handle from Layer 1 and wants to inhibit the handset from low power mode, it shall set the corresponding bit in sleep disable flag to disable low power mode. The API for this functionality is void **SM_SleepDisable**(sm_handle). Whenever a software module can enter low power mode, it shall clear the corresponding bit in sleep disable flag to enable the handset for low power operation again. The API for this functionality is void **SM_SleepEnaable**(sm_handle). A prerequisite to enter low power operation is that all sm_handles are enabled, not in disabled state. If there are one or more software modules in sm_disable state, the handset will not enter low power mode.

Regarding Layer 1 software control for sleep mode operation, it follows the control scenario of MT6205B slow clock unit, and MT6205B data sheet [1] can be referenced for detail information. In **l1d_data.h**, 2 customer-defined constants are declared to optimize sleep mode performance. The first constant is “FM_DURATION_DEFAULT”. Due to the inaccuracy of 32KHz oscillator, compared with the 13MHz oscillator, we need to calibrate the frequency error before entering sleep mode. “**FM_DURATION_DEFAULT**” is about the time interval to perform frequency measurement operation. The actual time interval to perform frequency measurement is 2*(FM_DURATION_DEFAULT +1) 32KHz cycles. If “FM_DURATION_DEFAULT” gets larger, the frequency measurement result will be more reliable and the power consumption will increase because of the frequency calibration process. The second constant to optimize sleep mode performance is the “**CLK_SETTLE_DEFAULT**”, the default off-chip VCXO settling duration. We need to program it to wakeup 13MHz clock and make sure that 13MHz clock has reached a stable amplitude and frequency before the time of expected wakeup point. The settle time is related to the chip and board circuits design. A typical value is from 3-6ms.

9.2 Exported interface and constant

Definition:

uint8 L1SM_GetHandle(void)

Description:

This function provides to get a sleep mode disable handler.

GLOBALS AFFECTED:

GLOBALS AFFECTED	Description

Return Value:

Return Value	Description
Uint8	A sleep mode disable handler

Parameter:

Parameter		Description

Definition:

void L1SM_SleepEnable(uint8 handle)

Description:

This function provides to enable sleep mode operation.

GLOBALS AFFECTED:

GLOBALS AFFECTED	Description
sm.sleepDisable	The sleep mode disable flag. Layer 1 will not enter sleep operation if this flag is non-zero.

Return Value:

Return Value	Description

Parameter:

Parameter		Description
uint8 handle	IN	A sleep mode disable handler that shall be gotten from L1SM_GetHandle().

Definition:

void L1SM_SleepDisable(uint8 handle)

Description:

This function provides to disable sleep mode operation.

GLOBALS AFFECTED:

GLOBALS AFFECTED	Description
sm.sleepDisable	The sleep mode disable flag. Layer 1 will not enter sleep operation if this flag is non-zero.

Return Value:

Return Value	Description

Parameter:

Parameter		Description
uint8 handle	IN	A sleep mode disable handler that shall be gotten from L1SM_GetHandle().

Definition:

FM_DURATION_DEFAULT

Description:

The default time interval used to perform frequency measurement operation. The actual time interval is 2*(FM_DURATION_DEFAULT +1) 32KHz cycles.

Definition:

CLK_SETTLE_DEFAULT

Description:

The default off-chip VCXO settling duration.

Bibliography references

[1] MT6205B GSM Baseband Processor Data
[2] JS Huang, “Acoustic Application Notes for FTA”
[3] HW Nien, and John Chuang, “DSP Acoustic Application Notes“

Figures Index

Tables index