Packet Interface of CT3258

Version:1.0

Date: 2015/01/14

Change History

Version	Date	Change Descriptions	Author
1.0	2015/01/14	Initial version	Hao Ye

Contents:

1 Packet Interface Descriptions

Packet interface is defined above the physical interface (HPI, Serial Port) for communications between CT3258 and the host MCU. The packet format is identical regardless of physical interfaces.

As HPI in CT3258 is preconfigured to use 16 bit format, the total number of bytes in a packet should be even. If the total number of byte is odd, a zero is appended at the end. This appended zero will not be counted in the LENGTH field.

1.1 Packet Format

The packet format is as shown in the table below. A packet consists of a packet header and number of packet fields and optional parity bytes. A packet header starts with 3 bytes of START_BYTES. The next two bytes contain the packet LENGTH and the next byte contains the TYPE. Each packet contains one or more fields which are shown as FIELD (0) through FILELD (N-1). Optionally, at the end of the packet, there are two parity bytes.

Packet Header			Field			Parity	
START_BYTES	LENGTH	TYPE	FIELD(0)	...	FIELD(N-1)	FBYTE	PARITY_BYTE
3 byte	2 byte	1 byte	Variable		Variable	1 byte	1 byte

Table 1-1 General Packet Format

Packet details are described in the sections below.

1.1.1 START_BYTES

The three bytes of START_BYTES always have fixed values of 0x84, 0xA9, 0x61.

1.1.2 LENGTH

The PACKET LENGTH occupies two bytes of the packet. The MS byte of the packet length is the fourth byte of the packet and the LS byte of the packet length is the fifth byte of the packet. The PACKET LENGTH is the sum of each FIELD. Note that PACKET LENGTH excludes the first 6 bytes taken up by the Packet Header. If parity check is enabled, the length field includes the parity byte as well.

As HPI in CT3258 is preconfigured to use 16 bit format, the total number of bytes in a packet should be even. If the total number of byte is odd, a zero is appended at the end. This appended zero will not be counted in the LENGTH field.

1.1.3 TYPE

TYPE field specifies the destination, read/write property and the types of the packet. It includes

multiple fields, described in the table below:

Bit Position	Field	Descriptions
7	AMBE bypass	External Vocoder message bypass 0: CT3258 Control 1: AMBE control
6	Codec bypass	Codec message bypass 0: CT3258 Control 1: Codec control, any bytes after packet type are send to codec directly
5	Read/Write	Read Write Control 0: Write. The MCU write fields to CT3258 1: Read. CT3258 report fields to the MCU. Not all parameters are readable.
4	Near/Far	Near end / Far end control 0: Near end 1: Far End
3-0	Packet Type	See Table Below

Table 1-2 Fields in TYPE Byte

Internally, the call related information is stored in registers in CT3258. External MCU can write to and read from these registers through packet interface. The write/read control is through bit 5 of TYPE field. Not all registers are readable. Column 6 of Table 1-5 describes whether a register is readable or not.

Some fields, such as Own ID, Called ID, have two sets of internal registers, for far end and near end of calls. When writing a packet field, the content is always stored in the near end registers. When reading field information, the MCU can request the near end information or the far end information. The near end information is the information that is to be sent or already sent to the far end. The far end information is the information that is received from the far end. The near/far end control is through bit 4 of the TYPE byte.

For example, station A has an Own ID 1234567, station B has an Own ID of 1234563. The user (MCU) can set the Own ID for the two stations with field DPMR_OWN_ID_BCD, and with write flag. The user can then read back the Own ID of each station with read flag, and near end flag at the two stations. After a call is made from A to B, the user can read the Own ID of station A by issuing read command to station B with far end flag. Column 6 of Table 1-5 also describes whether far end register is available for reading.

The details of Packet Type are described in Table 1-3 below.

Type Value	Packet Name	Descriptions
0	Control Packets	Used to set up chip operation mode, configure hardware, set up or end a call. When a control packet is received, the chip returns a control packet with response fields that contain response data or indication of errors in the control packet.
3	DPMR Packet	Used for DPMR protocol specific information. User can use these messages to change the elements of DPMR protocol.
4	Program packet	Used for host MCU to program CT3258.
5	DMR	Used for DMR protocol specific information. User can use these messages to change the elements of DMR protocol.
Others	Reserved	Reserved

Table 1-3 Packet Types

Note that the boot loader for DPMR mode and DMR mode is different. Depending on the boot loader types, CT3258 enters DPMR mode or DMR mode. In DPMR mode, commands for DMR only (type 5) are not recognized. In DMR mode, command for DPMR only (type 3) are not recognized.

Also note that for program packet only (type 4), the packet header and parity bytes can be omitted. In that case, the boot loader searches for a match of the field and command code.

1.1.4 Packet Fields

The packet fields contain useful packet information. Various different packet fields each with their own format are defined in the next section, however, the general format of a field is shown in the table below.

A field consists of a field identifier followed by field data. The length of field data is dependent upon the field identifier. Many fields have fixed lengths. Some fields are variable in length; and in such cases the length of the field data is embedded inside field data explicitly or implicitly.

Field Identifier	Field Data
1 byte	L(n)-1 bytes

Table 1-4 General Field Format

1.1.5 Parity Field

The parity field is a 2-byte field at the end of a packet. The first byte of the parity field is the parity field identifier and is always equal to 0x2f. The second byte of the parity field is the parity byte. It is obtained by “Exclusive-oring” every byte in the packet, except for the START_BYTES and the parity byte (last byte), together. CT3258 checks the parity byte for all received packets, informs the MCU of any parity error, and discards any packet that has an incorrect parity byte. When

parity check fails, the receiver replies with a packet (with field 0x17) indicating parity failure. Parity check is disabled at reset. It can be enabled when CT3258 receives a control packet with PARITY_ENABLE field.

1.1.6 Time Out

If a valid message is received, CT3258 responds to MCU in 40 milliseconds (except some especial commands in the table). If a response packet is not received within 40 milliseconds, the MCU can consider the message lost and resend the packet if necessary.
When a response packet is received, the MCU or CT3258 does not send an ACK to the response packet.

Commands	Response time
CHAN_D	80ms
VOCOER_SEL	1-3s
CODEC_SELECT	1 s
DIGC_DATA_FRAME	80ms. The response time of the first data is header frame(80ms) + Preamble + Silence
ACK_MESSAGE	The response time of receiving 17 0A after finishing sending is about 400ms. We have to wait the 17 0A to do the subsequent operation.

1.2Field Formats

A packet must contain one or more fields. The field formats are different for different types of packets. Each packet requires a response packet.

Control packets can be used to configure the chip prior to operation and also to query for information from the chip. The response packet for must fields just echoes back the control field identifier followed by a 0x00 byte to indicate the control field was received successfully. If multiple fields are present in the packets, the response packet only needs to echo the last field of the packet.

DPMR / DMR packets contain the call information. They can be originated either from host or from CT3258. The response packets to DPMR / DMR packet echoes back the control field identifier followed by a 0x00 byte to indicate the field was received successfully. If multiple fields are present in the packets, the response packet only needs to echo the last field of the packet.

The Program Packet is used by the host to down load application program to CT3258. The response packet to Program Packet echoes back the control field identifier followed by a 0x00 byte to indicate the field was received successfully.

Either the MCU or CT3258 can initiate a packet exchange. The packet can be used to write information to the other end (CT3258 or MCU), or it can be used to request information from the

other end. The Read/Write control is by bit 5 of the TYPE byte.

When requesting field information, the MCU can request near the end information or the far end information. The near end information is the information that is to be sent or already sent to the far end. The far end information is the information that is received from the far end. The near/far end control is through bit 4 of the TYPE byte.

The table below summarizes various fields used in CT3258.

Field Name	Field Identifier	Packet Type	Control Field Length (Bytes)	Response Field Length (Bytes)	Note	Descriptions
CHAN_D	0x01	0	Varies	2	YP	Encoded voice data
VOCODER_CMODE	0x02	0	2	2	N	Vocoder mode flags for current vocoder
AFC_CONFIG	0x05	0	3	2	YR	
COMPANDER_CONFIG	0x06	0	8	2	N	Compander Configuration
VOCODE_TONE	0x08	0	Varies	2	Y	Force current encoder/decoder to generate tone frames
ANALOG_FEATURE	0x0A	0	3	2	N	Analog Feature configuration
VOCOER_SEL	0x10	0	2	2	N	Vocoder Selection
VOCODER_INI	0x12	0	2	2	N	Vocoder initialization
CHIP_RESET	0x14	0	1	2	N	Chip reset
CHIP_LOWPWR	0x15	0	2	2	N	Set chip to low power mode
PARITY_ENABLE	0x16	0	2	2	N	Enable parity check
ACK_MESSAGE	0x17	0	None	2	N	Message error report
WORK_MODE	0x18	0	4	2	Y	Set up CT3258 work mode
CARRIER_READY	0x19	0	2	2	N	Carrier Ready Message
PROCESS_MODE	0x1A	0	2	2	N	Process Mode
NEW_EVENT	0x1B	0	None	2	YX	New event indication
DEMOD_GAIN	0x1C	0	3	2	Y	Demodulator Gain
REPORT_FIELD	0x1D	0	2	2	NX	Fields to report when receiving calls.
BER_REPORT	0x1E	0	2	10	Y	Bit error rate test report
ANALOG_TONE	0x1F	0	18	2	N	Generate Local Tone
CALL_START	0x20	0	2	2	N	Start call send
CALL_STOP	0x21	0	1	2	N	Stop call send
CONNECT	0x22	0	2	2	NP	Send connection
DISCONNECT	0x23	0	2	2	NP	Send disconnection
DPMR_ACK	0x24	0	2	2	NP	Send ack
SYSTEM_INFO	0x25	0	2	2	NP	Send system or status

						information
ANALOG_SUBAUDIO	0x26	0	Varies	2	N	analog subaudio
CALL_MATCH	0x27	0	None	2	N	Call match report
ADDRESSING_MODE	0x28	3	2	2	NP	Dialing mask length
GROUP_ID	0x29	3	5	2	YP	Group ID number
SCRAMBLER_SEED	0x2A	0	Varies	2	NP	Scrambler seed
MOD_GAIN	0x2C	0	3	2	N	Modulator Gain
CODEC_SELECT	0x2D	0	2	2	N	Codec Selection
SET_I2C_ADDRESS	0x2E	0	2	2	N	Set the I2C address for the codec
SYNC_MISS_CNT	0x2F	0	2	2	N	Set the number of Sync miss before reporting
Q_PRO_STR	0x30	0	1	Varies	Y	Query for product code
Q_HW_VER	0x31	0	1	Varies	Y	Query for hardware version
Q_SW_VER	0x32	0	1	Varies	Y	Query for software version
Q_CID_SN	0x33	0	1	9	Y	Query for customer ID
DMR_GROUP_ID	0x36	5	6	2	YD	Group ID number
DC_OFFSET	0x39	0	13	2	N	DC Offset
ANALOG_CONFIG	0x3c	0	12	2	N	Analog Configuration
VOCODER_IO_SET	0x3e	0	2	2	NP	Vocoder IO settings
CTC_PARAM	0x3d	0	23	2	N	CTC detection parameters
MODOUT_CONFIG	0x41	0	6	2	N	Two point modulation delay
IQCOMP_COEF	0x42	0	10	2	YR	IQ compensation configuration
DIGC_DATA_FRAME	0x43	0,3,5	Varies	2	YFX	Data frames
DPMR_PREAMBLE_LENGTH	0x46	0,3	3	N	N	Preamble and silence length
DPMR_CALL_OPTION	0x47	3	6	2	N	DPMR Call Option
DPMR_END_UE	0x4B	3	4	2	YF	17-bit un-encoded END
DIGC_FS1	0x4C	3	7	2	YF	48-bit of FS1
DIGC_CALLED_ID_BIN	0x50	3, 5	4	2	YF	24-bit called ID in binary
DIGC_OWN_ID_BIN	0x51	3, 5	4	2	YF	24-bit own ID in binary
DPMR_CALLED_ID_BCD	0x52	3	5	2	YFP	7 digit called ID in BCD
DPMR_OWN_ID_BCD	0x53	3	5	2	YFP	7 digit own ID in BCD
DPMR_M_V_F_E	0x54	3	2	2	YFP	3-bit communication mode and 4-bit communication format
DPMR_PROTOCOL_OPTION	0x55	3	2	2	NP	DPMR protocol option

DPMR_SLD	0x56	3	4	2	YFP	18-bit slow data
DPMR_HT_CI_PM	0x57	3	3	2	YFP	4-bit header type and 11-bit call information
DPMR_CI	0x5A	3	3	2	FP	11-bit call information
DPMR_CC	0x5C	3	4	2	YFP	24-bit of color code (Di-bit)
DPMR_SEND_SF	0x5F	3	1	2	NP	Send super frame command. Upon receiving this command, CT3258 assemble a super frame and send to the far end
DPMR_SEND_HEADER	0x60	3	1	2	NP	Send header frame command. Upon receiving this command, CT3258 assemble a header frame and send to the far end
DPMR_SEND_END	0x61	3	1	2	NP	Send end frame command. Upon receiving this command, CT3258 assemble a end frame and send to the far end
DPMR_SEND_AD	0x62	3	1	2	NP	Send appended data
DIGI_MIC_GAIN	0x68	0	3	2	N	Digital Microphone Gain
DIGI_SPEAKER_GAIN	0x69	0	3	2	N	Digital Speaker Gain
I2C_OPERATION	0x6A	0	Varies	5	Y	I2C Read and Write
MISC_GAIN	0x6B	0	13	2	N	Misc. Gains
SQ_LEVEL	0x6C	0	13	2	YR	RSSI and OOBE levels
SPI_OPERATION	0x6D	0	3	2	YR	SPI Write Operation
DMR_SLC	0x6E	5	5	2	ND	DMR SLC report
DMR_CALL_SLOT	0x6F	5	2	2	ND	DMR call slot information
EQUALIZER_FILTER	0x70	0	12	2	N	Set coefficients for audio equalizer
DMR_FLC	0x76	5	4	2	YD	DMR Full LC
DMR_CC	0x77	5	2	2	YD	DMR Color Code
DMR_CALL_START	0x78	5	4	2	ND	DMR Call Start
DMR_CALL_OPTION	0x79	5	6	2	ND	DMR Call option
DMR_OFFSET	0x7A	5	9	2	ND	DMR Slot Timing Offset
DMR_SLOT_TYPE	0x7B	5	2	2	ND	DMR slot type
DMR_EMB	0x7C	5	2	2	ND	DMR EMB field
DMR_CALLED_ID_BCD	0x7D	5	5	2	YFD	8 digit called ID in BCD
DMR_OWN_ID_BCD	0x7E	5	5	2	YFD	8 digit called ID in BCD
DMR_SLOT_FOUND	0x7F	5	2	2	ND	DMR_slot found

Table 1-5 Overview of Packet Fields

In the note field in the above table, Y means the command can be used as both WRITE and READ

command; N means the command can only be used as a WRITE command. F means the command can be used to read both Far End information and Near End information. R means the command is used only when paired with SCT3700. D means command can be used for DMR only. P means command can be used for DPMR only. X means command is interpreted differently for DMR and DPMR.

As most fields are used as commands to CT3258 for controlling and configuration purposes, we sometimes call field as command. As a result we use the terms field and command in-discriminatively throughout the document.

The detailed descriptions of each filed is described in the following sections. Only field format in Write Packet is shown. Read Packets usually only have the field ID byte.

1.2.1 CHAN_D

This field can be used to pass encoded voice data between the MCU and CT3258. This command is only used in DPMR mode.

Field Identifier	Data Length	Vocoder Configuration field
1 Byte	1 Byte	1 Bytes
0x01	The length of the data in bytes	Encoded voice data. 36 bytes per 80 milliseconds.
Default:	36	

Table 1-6 CHAN_D Field Format

1.2.2 VOCODER_CMODE

This field can be used to change the mode of the vocoder.

Field Identifier	Vocoder Configuration field
1 Byte	1 Bytes
0x02	{ AMBE_COMP, 0, 0, 0, NS, 0, TD, AGC}
Default:	0x08

Table 1-7 VOCODER_CMODE Field Format

The meaning of each bit is given below:

Bit Name	Descriptions	Default
AMBE_ COMP	AMBE vocoder compatibility bit	0
NS	Noise Suppression Enable 0: disable	1

	1: enable	
Reserved	Reserved	0
TD	Tone Detection Enable 0: disable 1: enable	0
AGC	AGC Enable 0: disable 1: enable	0

Table 1-8 VOCODER_CMODE Field Options

1.2.3 AFC_CONFIG

This command is used to configure AFC (automatic frequency control), when used with SCT3700.

If the receiver carrier frequency is not matched to the transmitter carrier frequency, the performance degrade.

The use the AFC, the user first use this command to read the frequency offset between the transmitter and the receiver. To compensate for the frequency offset, the user has two options:

1. Change SCT3700 receiver LO, using SCT3700 frequency configuration command.
2. Change the internal mixer inside CT3258, using this command.

The AFC_CONFIG write command has the following format:

Field Identifier	Internal frequency offset
1 Byte	1 Bytes
0x05	Internal frequency offset applied at the CT3258 mixer. The unit is Hz
Default:	0

Table 1-9 AFC_CONFIG Field Format

AFC_CONFIG command is also used to read the frequency offset between the receiver and the transmitter.

To read the frequency offset, AFC_CONFIG command should be sent with “Read/Write” bit in the TYPE field set to 1. The AFC_CONFIG command used in “read” mode has the following format:

Field Identifier	DC Time Constant
1 Byte	1 Byte
0x05	Time constant for averaging AFC frequency offset
Default	3

Time constant of 0 corresponds to average time of 31.25 ms, which is the time it takes to reach within 10% of the steady state energy value. If the constant is n, the average time is given by the following formula:

T = 31.25 * 2^n.

The Response to the read AFC_CONFIG command has the following format:

Field Identifier	Frequency offset	Reserved
1 Byte	1 Byte	1 Bytes
0x05	Frequency offset between the transmitter and the receiver, in Hz	Reserved
Default	-	-

1.2.4 COMPANDER_CONFIG

This command is used to configure the compander in analog call mode. The configurable parameters include compander reference level for the transmitter and the receiver, the energy estimation timing constant, and the compander gain update interval.

The compander energy level is calculated as the running average of the absolute values of received audio signal, given by the equation below:

E(n) = (1-alpha) * E(n-1) + alpha * |x(n)|,

where |x(n)| is the absolute value of the received audio signal, alpha is the time constant. The smaller is the alpha, the larger is the time constant. Roughly, T = 4 / alpha, where T is the time it takes to reach within 10% of the steady state energy value. For example, with the default alpha = 68, the time constant is T = 4 / 68 = 60 ms.

The TX and RX compander reference levels are the signal levels that the input signal are compared against. At the TX side, if the input signal energy level is greater than the TX compander reference level, the input signal are reduced; if it is less than the TX compander reference level, it is enlarged. At the RX side, if the input signal energy level is greater than the RX compander reference level, the input signal are enlarged; if it is less than the RX compander reference level, it is reduced.

The compander gain update interval is the interval that the compander gains are re-calculated.

Within the interval, the compander gain is unchanged. The compander gain update interval is in terms of 24 kHz samples.

Field Identifier	Compander Update interval	Compander TX reference	Comander RX reference	Compander Energy Constant
1 Byte	1 Byte	2 Bytes	2 Bytes	2 Bytes
0x06	Compander update interveral in terms of 24 samples	Compander TX reference, 0-32768	Compander RX reference, 0-32768	Compander Energy update constant, 0-16384
Default	60	2048	2048	68

Table 1-10 COMPANDER_CONFIG Field Format

1.2.5 VOCODE_TONE

This field is used to force the encoder to transmit a tone frame. The frequency (or frequencies) and amplitude of the tone are specified by this field. It is also used to report the detected DTMF tone or single tone to the MCU.

Field Identifier	Tone Configuration	Tone Data	Amplitude Data	16 bit Frequency
1 Byte	1 Byte	1 Byte	1 Byte	2 Bytes (Optional)
0x08	See Table 1-10 for details	See Table 1-12 for details	Amplitude scale in dB +3dB to -90 dB	Single tone Frequency in 16 bit decimal
Default	0	0x80	-10 dB (0xF6)	0

Table 1-11 VOCODE_TONE Field Format

The details of Tone Configuration byte is as follows:

Bit Position	Descriptions	Default
Bit 7-3	Reserved	0
2	Optional 16 bit frequency field flag 0: No 16 bit frequency field 1: 16 bit frequency field present	0
1	Tone Direction 0: Send or detect near end Vocoder tone 1: Send or detect far end Vocoder tone	0

0	Tone on / off flag 0: Tone Off 1: Tone On	0

Table 1-12 Tone Configuration Details

DTMF Tone Data	Tone Type	Frequency 1 (Hz)	Frequency 2 (Hz)
0-4	Invalid Tone	NA	NA
0x05-0x7A	Single Tone	Tone Data * 31.25 Hz	NA
0x7B-0x7F	Invalid tone	NA	NA
0x80	DTMF “0”	1336	941
0x81	DTMF “1”	1209	697
0x82	DTMF “2”	1336	697
0x83	DTMF “3”	1477	697
0x84	DTMF “4”	1209	770
0x85	DTMF “5”	1336	770
0x86	DTMF “6”	1477	770
0x87	DTMF “7”	1209	852
0x88	DTMF “8”	1336	852
0x89	DTMF “9”	1477	852
0x8a	DTMF “A”	1633	697
0x8b	DTMF “B”	1633	770
0x8c	DTMF “C”	1633	852
0x8d	DTMF “D”	1633	951
0x8e	DTMF “*”	1209	941
0x8f	DTMF “#”	1477	941
0x90	KNOX “0”	1162	820
0x91	KNOX “1”	1052	606
0x92	KNOX “2”	1162	606
0x93	KNOX “3”	1279	606
0x94	KNOX “4”	1052	672
0x95	KNOX “5”	1162	672
0x96	KNOX “6”	1279	672
0x97	KNOX “7”	1052	743
0x98	KNOX “8”	1162	743
0x99	KNOX “9”	1279	743
0x9a	KNOX “A”	1430	606
0x9b	KNOX “B”	1430	672
0x9c	KNOX “C”	1430	743
0x9d	KNOX “D”	1430	820
0x9e	KNOX “*”	1052	820
0x9f	KNOX “#”	1279	820
0xa0	Dial Tone	440	350
0xa1	Ring tone	480	440

0xa2	Busy Tone	620	480
0xa3	Call Progress Tone	490	350
0xa4-0xff	Inactive	NA	NA

Table 1-13 VOCODE_TONE Field Options

Note that the bytes for 16 bit Frequency is optional. It is present only if bit 2 of Tone Configuration Byte is 1.

1.2.6 ANALOG_FEATURE

This field configures for special feature. Note that these special feature can only be used when the analog vocoder (ID = 16) is loaded.

Field Identifier	ANALOG_FEATURE
1 Byte	2 Bytes
0x010	Bit 0: DTMF enable flag 0: DTMF detection is disabled 1: DTMF detection is enabled Bit 1: DTMF detection position 0: DTMF detection is done before the de-emphasis filter 1: DTMF detection is done after the de-emphasis filter Bit 2: MSK enable flag 0: MSK transmission and detection disabled 1: MSK transmission and detection is enabled. Bit 3: Compander enable flag 0: Compander is disabled 1: Compander is enabled. Bit 4-15: reserved.
Default	3

Table 1-14 ANALOG_FEATURE Field Format

1.2.7 VOCOER_SEL

This field forces vocoder type.

CT3258 supports AMBE+2 vocoder from DVSI.

Field Identifier	Vocoder Type
1 Byte	1 Byte
0x010	4: AMBE+2 16: Analog calls Others, reserved
Default	4

Table 1-15 VOCODER_SEL Field Format

1.2.8 VOCODER_INI

This command initializes the vocoder.

Field Identifier	Initialization type
1 Byte	1 Byte
0x012	N/A
Default	0

Table 1-16 VOCODER_INI Field Format

1.2.9 CHIP_RESET

This command resets CT3258. This command has the effect as NMI signal (non-maskable interrupt), which restarting the firmware in CT3258 without reloading the firmware.

Field Identifier
1 Byte
0x14

Table 1-17 CHIP_RESET Field Format

1.2.10 CHIP_LOWPWR

This command puts CT3258 in low power mode.

Field Identifier	Low Power Mode
1 Byte	1 byte
0x015	Bit 7:6, Reserved Bit 5: Power down external vocoder Bit 4: Power down external codec Bit 3:0: Power mode of CT3258 0: Normal 1: Stop TX and RX processing 2: Reserved

	3: Power down certain peripheral of CT3258 including serial port, DMA and timers. 4: CT3258 enters IDLE mode 5: CT3258 enters Sleep mode 6: CT3258 enters Halt mode
Default	0

Table 1-18 CHIP_LOWPWR Field Format

1.2.11 PARITY_ENABLE

This command enables parity checks on all packets:

Field Identifier	Parity Enable/Disable
1 Byte	1 Byte
0x016	0: Disable parity checks 1: Enable parity checks
Default	0

Table 1-19 PARITY_ENABLE Field Format

Note that unlike other fields, PARITY_ENABLE field can only be the only field in a packet.

1.2.12 ACK_MESSAGE

This is a response packet to a received packet, when packet is not received correctly, and the packet field is not recognizable or unreliable.

Note that when received field is recognizable reliably even though there is error in the packet, CT3258 respond with the command field instead of ACK_MESSAGE field.

Note that 0x17 0x0A does not indicate an error condition. This is sent when CT3258 has completely sent the message to be sent to the far end.

Field Identifier	Failure Code
1 Byte	1 Byte
0x017	0: No error 1: Response to system reset. CT3258 is ready to receive command from the host. 2: Field length exceeds maximum allowed 3: Parity error 4: Unknown command error 5: Packet length does not match with field Length 6: Error in processing field

	7: I2C write error 8: SLD request time out error 9: TCH request time out error 10. Message to far end completion indication 11. Error in processing called ID or own ID 12. Chip not activated 13. Unsupported feature 14. Function not loaded 15. Odd byte error Others: reserved
Default	0

Table 1-20 PACKET_FAILURE Field Format

1.2.13 WORK_MODE

This command set the work mode for CT3258.

Field Identifier	Work Mode	Modem Loop Mode	Audio Loop Back	Debug Mode
1 Byte	1 Byte	1/2 Byte [7:4]	1/2 Byte [3:0]	1 Byte
0x018	See Table 1-19 for details.	0: No Loop 1: Codec Loop back 2: 4 FSK Loop back 3: Packet Loop back 4: FEC Loop back 5: Vocoder Loop back 6: Decimator Loop back Others: reserved	0: No Loop 1: Codec Loop back 2: Decimator Loop back 3: Vocoder Loop back 4: FEC Loop back 5: Packet Loop back 6: 4 FSK Loop back Others: reserved	See Table 1-20 for details
Default	0	0	0	0

Table 1-21 WORK MODE Field Format

Bit Position	Descriptions
7	Two Point Modulation Disable Flag 0: Enable two point modulation in TX mode 1: Disable two point modulation in TX mode
6	Special SCT3700 calibration loop enable 0: Normal mode 1: Enable the special SCT3700 calibration loop
5-4	Reserved
3-0	Work Mode

	0: Idle 1: RX 2: TX 3: Full Duplex

Table 1-22 Details of Work Mode Byte

Byte Value	Descriptions
0	No debug
1	Mod port send fixed sine wave of 1000 Hz
2	Mod port send 4 FSK signal with FS1 pattern
3	Reserved
4	Reserved
5	BER Test mode
6	Reserved
7	Reserved
8	Reserved
9	TX sends 1031 Hz test Tone instead Voice from microphone
10	TX sends silence instead Voice from microphone
11	DMR BER test mode (STD ID 511)
14	DPMR/DMR BER test mode (STD IB CALL), see note 1

Table 1-23 Details of Debug Mode Byte

Note 1: The STD IB CAL pattern changes the STB IB 511 pattern with every 100th bit inverted. This gives this pattern a 1% bit error rate

1.2.14 CARRIER_READY

This command informs CT3258 that carrier is detected on the RX side. Upon receiving this message, CT3258 start to look for preamble and frame SYNCs. This command is also used to inform CT3258 whether to mute the audio if the called ID (for DPMR calls) or CTCSS/DCS code does match.

If carrier detection (SQ) is to be done by CT3258 (enabled by CALLL_OPTION command), this command starts the carrier detection on CT3258.

Field Identifier	Carrier Ready Indication
1 Byte	1 Byte
0x019	Bit 7:3, Reserved Bit 2: Mute Flag for muting audio if called ID or CTCSS/DCS code does not match. 0: Un-mute the audio even if the called ID (for DPMR calls) or CTCSS/DCS code (for a analog calls) does not match

	1: Mute the audio if the called ID (for DPMR call) or CTCSS/DCS code (for analog calls) does not match, un-mute other wise. Bit 1:0, Carrier ready flag 0: Carrier Lost 1: Carrier Ready 2: Carrier Ready, but the phase of the demodulated signal is reversed. Others: reserved Note： The reverse of demodulated signal polarity does not work in analog mode. Please configure the polarity with ANALOG_SUBAUDIO.
Default	0

Table 1-24 CARRIER_READY Field Format

1.2.15 PROCESS_MODE

This command set the process mode of CT3258. The process mode includes DPMR / DMR physical layer only (1), physical layer plus data link layer (2) and physical layer plus data link layer plus call control layer (3). This command is also used to enable automatic detection of analog or digital calls.

Field Identifier	Process Mode
1 Byte	1 Byte
0x01A	2: Layer 2, DPMR / DMR Layer 2 Mode 3: Layer 3, Easy Mode 128: Analog Mode 131: Automatic detection of analog call or DPMR calls (layer 3) Others: Reserved
Default	2

Table 1-25 PROCESS_MOD Field Format

With layer two processing mode, CT3258 is responsible to perform layer one and layer two processing of the DPMR / DMR protocol, including the FEC, interleaving and message interpretation.

Layer 2 processing complies with ETSI TS 102 490 or TS 102 658 in DPMR mode, and ETSI TS 102 361 in DMR mode,

With layer three processing mode, in addition to layer two processing, the CT3258 also performs call control of the DPMR / DMR processing. ANNEX A Standard User Interface for CSF radio in DPMR mode, and ANNEX C DMR Numbering and Dialing plan are implemented in CT3258. At the transmitter, CT3258 supports wild character dialing for group and all calls for DPMR and DMR, abbreviated dialing and masked dialing for DPMR. At the receiver, when a call is received,

CT3258 does matching of individual call number and group number and informs the MCU of matching results.

1.2.16 NEW_EVENT

This message is sent when a new event occurs and requires MCU attention. Note that some of the fields different in DPMR and DMR mode.

Field Identifier	Event
1 Byte	1 Byte
0x01B	0: No event 1: DPMR: New Header frame received, with FEC error DMR: Data Sync received with FEC error 2: DPMR: New End frame received 3: DPMR: New CCH received 4: DPMR New Header frame received with no FEC error DMR: Data SYNC received 5: DPMR: Super Frame received, with no FEC error DMR: Voice SYNC received 6: DPMR: FS3 received 7: DPMR: FS4 received 8: Lost synchronization 9. DPMR: FS1 found pre-alert 10. DPMR: FS2 found pre-alert 11. Reserved 12. Carrier detected 13. Carrier lost 14. Unsupported call received 15. Wrong vocoder 16-255 reserved
Default	0

Table 1-26 NEW_EVENT Field Format

If a digital call is received and correctly decoded, CT3258 reports NEW_EVENT 4 OR 5. If a digital call is received but has FEC error, CT3258 reports NEW_EVENT 1. These event reports can also be disabled with REPORT_FIELD command.

In DPMR mode, if mixed call detection is enabled, and a digital call is received, NEW_EVENT 9 or 10 will be reported first, before complete FEC checking process, to alert MCU that a digital call is received. These events can be used to terminate analog call detection.

If CT3258 is to do carrier detection (SQ), NEW_EVENT 12 and 13 are used to report carrier status.

During the course of a digital call, if synchronization pattern is not detected for a period of time (configured through command SYNC_MISS_COUNT), CT3258 reports NEW_EVENT 8.

1.2.17 DEMOD_GAIN

This message set the demodulation gain for the demodulator. If the value is zero, auto calibration is used.

Field Identifier	Event
1 Byte	2 Byte
0x01C	0: Auto calibration Others: linear gain values in Q16.11 (16 bits total, 11 bits fraction, MSB first), with 2048 as 0 dB
Default	0

Table 1-27 DEMOD_GAIN Field Format

1.2.18 REPORT_FIELD

This message informs CT3258 what field to report when receiving a call.

Field Identifier	Mask Byte
1 Byte	1 Byte
0x01D	Mask to indicate the field to report when receiving a new call. Each individual bit select one field. Multiple fields can be selected.
Default	0x70

Table 1-28 REPORT_FIELD Field Format

The details of the bits are different depending on DPMR or DMR mode.

Bit Position	Field to report
7	NEW_EVENT (FS1 or FS2)
6	DPMR_M_V_F_E
5	DPMR_HT_CI_PM
4	DIGC_CALLED_ID_BIN
3	DPMR_CALLED_ID_BCD
2	DIGC_OWN_ID_BIN
1	DPMR_OWN_ID_BCD
0	DPMR_CC

Table 1-29 Bit definition for Mask Byte in DPMR Mode

Bit Position	Field to report
7	NEW_EVENT (FS1 or FS2)
6	DMR_FLC
5	DMR_CSBK
4-1	Reserved
0	DMR SLOTTYPE OR EMB

Table 1-30 Bit definition for Mask Byte in DMR Mode

1.2.19 BER_REPORT

This command enquires for the BER test report. The BER includes test success flag, test duration, total number of errors, and error rate in ppm (pieces per million). A duration parameter in the command control the duration used in ppm calculation

Field Identifier	BER Duration
1 Byte	1 Byte
0x01E	BER statistics duration 0: All errors in the test are used in BER ppm calculation 1-30, number of seconds used for ppm calculation. Error beyond the duration are discarded in BER ppm calculation
Default	0

Table 1-31 BER_REPORT Field Format

The response packet returns the BER results.

Field Identifier	Valid Indication	Test length	Errors	Error in PPM
1 Byte	1 Byte	2 Bytes	4 Bytes	4 Bytes
0x01E	1: BER test successful 0: BER test unsuccessful. Result is not meaningful	Word (MSB first) to indicate the number of seconds in BER test	Long word (MSB first) to indicate the number of errors found.	Long word (MSB first) to indicate the error percentage (in ppm)
Default	0	0	0	0

Table 1-32 BER_REPORT response Field Format

1.2.20 ANALOG_TONE

This command enables the CT3258 to generate or stop a tone signals to be played out from the line out port or the MOD port. The tone signal is a repetition of a two-tone pattern, in the form as Tone1-Tone2-Tone1-Tone2….-Tone1-Tone2. Each tone is made up of two frequencies. The

frequencies and amplitudes are programmable. If silence is desired for the second tone, the user should set the two frequencies or the two amplitudes of the second tone to zero.

Field Identifier	TONE_CTRL	Frequency 11	Frequency 12	Frequency 21	Frequency 22
1 Byte	1 Byte	2 Bytes	2 Bytes	2 Bytes	2 Bytes
0x01F	See Table 1-32 for Details	First frequency of the first tone, in Hz If Bit 7 of the MSB is 1, siren is generated	Second frequency of the first tone, in Hz	First frequency of the Second tone, in Hz	Second frequency of the second tone, in Hz
Default	0	NA	NA	NA	NA

Amplitude11	Amplitude 12	Amplitude 21	Amplitude 22	Length 1	Length 2
1 Byte	1 Byte	1 Bytes	1 Bytes	2 Bytes	2 Bytes
Amplitude of the first frequency of the first tone, in 8 bit linear scale with 6 bit fractions	Amplitude of the second frequency of the first tone, in 8 bit linear scale with 6 bit fractions	Amplitude of the first frequency of the second tone, in 8 bit linear scale with 6 bit fractions	Amplitude of the second frequency of the second tone, in 8 bit linear scale with 6 bit fractions	Length of the first tone in milliseconds.	Length of the second tone in milliseconds.
NA	NA	NA	NA	NA	NA

Table 1-33 LOCAL_TONE Field Format

Bit Position	Field to report
7	Remote Tone Flag 0: Local Tone to the line out port 1: Remote Tone to the MOD port
6-0	Repeat Number 0: Stop Tone 127: Infinite Others: Repetition Number

Table 1-34 Definition for TONE_CTRL

1.2.21 CALL_START

Call Start command. Upon receiving this command, CT3258 begins to send call to the far end.

Field Identifier	Call Details
1 Byte	1 Byte
0x20	DATA[0]={M[2:0], V[1:0], F[1:0], EP} Call Mode, Version, Call Format, Emergency Call flag
Default	0

Table 1-35 CALL_START Field Format

With CALL_START command, the header type, HT is set to 0x0.

The M, F, and P fields are described in the following tables

M	Communication Modes
000	Voice communication (no user data in SLD field)
001	Voice + slow data (user data in SLD field)
010	Data communication type 1 (Payload is user data without FEC)
011	Data communication type 2 (Payload is user data with FEC)
100	Data communication type 3 (Packet data, ARQ method)
101	Voice and appended data (type 2)
110	Appended data
Others	Reserved

Table 1-36 Communication Mode Field Details

V	Version number
00	DPMR standard traffic
01	Reserved
10	Reserved
11	Manufacturer specific

Table 1-37 Version Field Details

F	Communication Format
00	Call ALL (Broadcast)
01	Peer-to-peer communication
10	BS uplink
11	BS downlink

Table 1-38 Communication Format Field Details

EP	Emergency Priority Flag
0	Normal call
1	Emergency call

Table 1-39 Emergency Priority Flag Field Details

1.2.22 CALL_STOP

Call Stop command. Upon receiving this command, CT3258 sends an END frame to the far end and then stop the call to the far end.

Field Identifier
1 Byte
0x21

Table 1-40 CALL_STOP Field Format

This command is also used to end an analog call.

1.2.23 CONNECT

Call Connect command. Upon receiving this command, CT3258 send connection command to the far end. It is also used send a call alert to the far end in the case of voice calls.

Field Identifier	Call Details
1 Byte	1 Byte
0x22	DATA[0]={M[2:0], V[1:0], F[1:0], EP} Call Mode, Version, Call Format, Emergency Call flag
Default	0

Table 1-41 CONNECT Field Format

With CONNECT command, the header type, HT is set to 0x01.

The meaning of M, V, F and EP fields are described in Table 1-32 to Table 1-35.

If there is no frame after sending this command, we must wait CT3258 report 17 0A then make the subsequent operation (The response time is 150ms+Preamble +Silence+PowerSaveHeader).

1.2.24 DISCONNECT

Call Disconnection command. Upon receiving this command, CT3258 sends disconnect command to the far end.

Field Identifier	Call Details
1 Byte	1 Byte
0x23	DATA[0]={M[2:0], V[1:0], F[1:0], EP} Call Mode, Version, Call Format, Emergency Call flag

Default	0

Table 1-42 DISCONNECT Field Format

With DISCONNECT command, the header type, HT is set to 0x02.

The meaning of M, V, F and EP fields are described in section CALL_START.

If there is no frame after sending this command, we must wait CT3258 report 17 0A then make the subsequent operation (The response time is 300ms+Preamble +Silence +PowerSaveHeader).

1.2.25 DPMR_ACK

ACK command: Upon receiving this command, CT3258 send acknowledgment to the far end.

Field Identifier	Call Details
1 Byte	1 Byte
0x24	DATA[0]={M[2:0], V[1:0], F[1:0], EP} Call Mode, Version, Call Format, Emergency Call flag
Default	0

Table 1-43 DPMR_ACK field Format

With DPMR_ACK command, the header type, HT is set to 0x03.

The meaning of M, V, F and EP fields are described in section CALL_START.

If there is no frame after sending this command, we must wait CT3258 report 17 0A then make the subsequent operation (The response time is 100ms+Preamble +Silence+PowerSaveHeader).

1.2.26 SYSTEM_INFO

System Info command, upon receiving this command, CT3258 sends system or status request or response to the far end.

Field Identifier	Call Details
1 Byte	1 Byte
0x25	DATA[0]={M[2:0], F[3:0], EP} Call Mode, Version, Call Format, Emergency Call flag
Default	0

Table 1-44 SYSTEM_INFO field Format

SYSTEM_INFO command does not set header type, HT, field. The header type can be set with DPMR_HT_CI_PM or DPMR_HT command before sending SYSTEM_INFO.

The meaning of M, V, F and EP fields are described in in section CALL_START.

If there is no frame after sending this command, we must wait CT3258 report 17 0A then make the subsequent operation (The response time is 150ms+Preamble +Silence+PowerSaveHeader).

1.2.27 ANALOG_SUBAUDIO

This field contains analog sub_audio message. Upon receiving this message, CT3258 configure the sub-audio generation and detection setting of CT3258 in analog mode. It is also used to report detection of specific CTCSS/DCS tones and tail detection results.

The number of bytes for this field varies depending on the whether arbitrary CTCSS/DCS generation and detection is required. If no arbitrary CTCSS/DCS is required, the field length is 3. If arbitrary DCS code is required, the field length is 5. The added two bytes are for the arbitrary DCS code in hex form. If arbitrary CTCSS code is required, the field length is 9. Two of the added six bytes are for the CTCSS frequencies, another four bytes are for the two coefficients for the narrow band filter for that particular frequency.

The CTCSS coefficients are calculated from the arbitrary CTCSS frequency to be detected. The formula is:

a1 = round (2048 * sqrt (1.9990 * (1-cos(2*w0))))

a2 = round (-2047*2^9 * cos (w0)) + 2^20

where w0 is calculated from the desired arbitrary CTCSS frequency, f0, by

w0 = f0/1909.9

For example for CTCSS frequency 67 Hz, the coefficients are calculated as follows:

a1 = round (2048*sqrt (1.9990 * (1-cos (2*67/1909.9)))) = 144
a2 = round ((-2047*2^9 * cos (67/1909.9))) + 2^20 = 1157

The MCU is responsible to calculate the coefficients and pass them to CT3258.

Field Identifier	Sub-audio Configuration / Status	CTCSS/DCS number	Arbitrary CTCSS frequency or DCS code	Coefficient A1	Coefficient a2

1 Byte	1 Byte	1 Byte	2 Bytes	2 Bytes	2 Bytes
0x26	See Table 1-44 for details	0: No CTCSS /DCS 1-108: CTCSS/DCS code 255: Arbitrary CTCSS/DCS code Others: reserved	For CTCSS: CTCSS frequency in steps of 0.1 Hz (MSB first) For DCS, 0-0x1ff, corresponding to the DCS code, in HEX format , MSB first	Narrow band filter coefficient a1 for arbitrary CTCSS frequency	Narrow band filter coefficient a1 for arbitrary CTCSS frequency
Default	0x40	0	NA	NA	NA

Table 1-45 SUBAUDIO Field Format

Bit Position	Bit Name	Descriptions
7	CD_SEL	DCS and CTCSS select 0: CTCSS 1: DCS
6	EXP_SEL	Expanded CTCSS / DCS code Select 0: Standard CTCSS/DCS set (38 for CTCSS, 83 for DCS) 1: Expanded CTCSS DCS Set (51 for CTCSS, 107 for DCS)
5	Polarity	DCS polarity 0: Positive DCS polarity 1: Negative DCS polarity
4	Blind	Blind sub-audio detection enable 0: No blind sub-audio detection 1: Blind sub-audio detection enabled.
3	Auto Polarity Detection	DCS Auto polarity detection enable 0: DCS Auto polarity detection disabled 1: DCS Audio polarity detection enable
2-1	Tail Detection Configuration or Report	In a configuration packet, it is used to enabling tail generation and detection: 00: No Tail detection 01: 120 degree tail 10: 180 degree tail for CTCSS. Enable tail detection for DCS. 11: 240 degree tail In a report packet, it is used to report tail detection

		results: 00: No Tail found 01: 120 degree tail found 10: 180 degree tail found 11: 240 degree tail found
0	FOUND	Sub audio find flag 0: No CTCSS/DCS found 1: CTCSS/DCS found

Table 1-46 Sub-Audio Configuration Details

Serial Number：	DCS code in Octal format	DCS code in Hex format
1	017	00F
2	023	013
3	025	015
4	026	016
5	031	019
6	032	01A
7	036	01E
8	043	023
9	047	027
10	050	028
11	051	029
12	053	02B
13	054	02C
14	065	035
15	071	039
16	072	03A
17	073	03B
18	074	03C
19	114	04C
20	115	04D
21	116	04E
22	122	052
23	125	055
24	131	059
25	132	05A
26	134	05C
27	143	063
28	145	065
29	152	06A
30	155	06D
31	156	06E

32	162	072
33	165	075
34	172	07A
35	174	07C
36	205	085
37	212	08A
38	223	093
39	225	095
40	226	096
41	243	0A3
42	244	0A4
43	245	0A5
44	246	0A6
45	251	0A9
46	252	0AA
47	255	0AD
48	261	0B1
49	263	0B3
50	265	0B5
51	266	0B6
52	271	0B9
53	274	0BC
54	306	0C6
55	311	0C9
56	315	0CD
57	325	0D5
58	331	0D9
59	332	0DA
60	343	0E3
61	346	0E6
62	351	0E9
63	356	0EE
64	364	0F4
65	365	0F5
66	371	0F9
67	411	109
68	412	10A
69	413	10B
70	423	113
71	431	119
72	432	11A
73	445	125
74	446	126

75	452	12A
76	454	12C
77	455	12D
78	462	132
79	464	134
80	465	135
81	466	136
82	503	143
83	506	146
84	516	14E
85	523	153
86	526	156
87	532	15A
88	546	166
89	565	175
90	606	186
91	612	18A
92	624	194
93	627	197
94	631	199
95	632	19A
96	646	1A6
97	654	1AC
98	662	1B2
99	664	1B4
100	703	1C3
101	712	1CA
102	723	1D3
103	731	1D9
104	732	1DA
105	734	1DC
106	743	1E3
107	754	1EC
108	0	000

Table 1-47 DCS Extend Table

Serial Number：	DCS code in Octal format	DCS code in Hex Format
1	023	013
2	025	015
3	026	016
4	031	019
5	032	01A
6	043	023
7	047	027

8	051	029
9	054	02C
10	065	035
11	071	039
12	072	03A
13	073	03B
14	074	03C
15	114	04C
16	115	04D
17	116	04E
18	125	055
19	131	059
20	132	05A
21	134	05C
22	143	063
23	152	06A
24	155	06D
25	156	06E
26	162	072
27	165	075
28	172	07A
29	174	07C
30	205	085
31	223	093
32	226	096
33	243	0A3
34	244	0A4
35	245	0A5
36	251	0A9
37	261	0B1
38	263	0B3
39	265	0B5
40	271	0B9
41	306	0C6
42	311	0C9
43	315	0CD
44	331	0D9
45	343	0E3
46	346	0E6
47	351	0E9
48	364	0F4
49	365	0F5
50	371	0F9

51	411	109
52	412	10A
53	413	10B
54	423	113
55	431	119
56	432	11A
57	445	125
58	464	134
59	465	135
60	466	136
61	503	143
62	506	146
63	516	14E
64	532	15A
65	546	166
66	565	175
67	606	186
68	612	18A
69	624	194
70	627	197
71	631	199
72	632	19A
73	654	1AC
74	662	1B2
75	664	1B4
76	703	1C3
77	712	1CA
78	723	1D3
79	731	1D9
80	732	1DA
81	734	1DC
82	743	1E3
83	754	1EC
84	0	000

Table 1-48 DCS Standard Table

Serial Number：	CTCSS frequency
1	67
2	71.9
3	74.4
4	77
5	79.7

6	82.5
7	85.4
8	88.5
9	91.5
10	94.8
11	97.4
12	100
13	103.5
14	107.2
15	110.9
16	114.8
17	118.8
18	123
19	127.3
20	131.8
21	136.5
22	141.3
23	146.2
24	151.4
25	156.7
26	162.2
27	167.9
28	173.8
29	179.9
30	186.2
31	192.8
32	203.5
33	210.7
34	218.1
35	225.7
36	233.6
37	241.8
38	250.3
39	134.4

Table 1-49 CTCSS Standard Table

Serial Number：	ctcss 52 digit decimal
1	63
2	67
3	69.3
4	71.9
5	74.4

6	77
7	79.7
8	82.5
9	85.4
10	88.5
11	91.5
12	94.8
13	97.4
14	100
15	103.5
16	107.2
17	110.9
18	114.8
19	118.8
20	123
21	127.3
22	131.8
23	136.5
24	141.3
25	146.2
26	151.4
27	156.7
28	159.8
29	162.2
30	165.5
31	167.9
32	171.3
33	173.8
34	177.3
35	179.9
36	183.5
37	186.2
38	189.9
39	192.8
40	196.6
41	199.5
42	203.5
43	206.5
44	210.7
45	218.1
46	225.7
47	229.1
48	233.6

49	241.8
50	250.3
51	254.1
52	134.4

Table 1-50 CTCSS Extend Table

1.2.28 CALL_MATCH

This field is used in a CT3258 generated report message when a call is received, as a result from a CARRIER_READY command from the MCU. It is generated if a DPMR or a DMR call is received and that the receiver can start to SYNC with the transmitter.

Field Identifier	Call Details
1 Byte	1 Byte
0x27	See Table 1-48 for details
Default	0

Table 1-51 CALL_MATCH Field Format

Bit Position	Bit Name	Descriptions
7-4	Group ID	For group calls, the matched group ID
3	CC Match	Color code match status 0: Color code matched 1: Color code not matched
2	Un-match Reason	The reason for call not matching 0: Valid Called ID not received 1: Called ID not matched Others: reserved
1	Group Call	Group or individual call flag when a call is matched 0: Individual call 1: Group call
0	Called ID Match Flag	Called ID Match Flag 0: Called ID matched with receiver Own ID, or one of the receiver group ID 1: Called ID is not matched

Table 1-52 Call Match Details

1.2.29 ADDRESSING_MODE

This field is used to configure the addressing mode of CT3258 for DPMR. As described in the DPMR spec (TS 102 490 or 658).), with initial addressing mode, the user uses 254 binary ID's. With configured addressing mode, the user can use a dial pad to dial 7 digital numbers.

The field format is described below.

Field Identifier	Addressing Mode Details
1 Byte	1 Byte
0x28	See Table 1-50 for details
Default	0

Table 1-53 ADDRESSING_MODE Field Format

Bit Position	Bit Name	Descriptions
7-4	Reserved	Reserved
3-1	Dial mask Length	0-6: The length of the Dial mask for Configured Addressing mode Others: invalid.
0	Mode Select	Address mode select 0: Initial Addressing mode 1: Configured Address mode (Standard User Interface)

Table 1-54 Addressing Mode Details

The number of digital the user is allowed to dial is (7- Dial_Mask_Length). For example, 3 mean that 3 numbers are masked. The user is able to dial a maximum of 4 numbers. By default, Dial_Mask_Length = 0, meaning no mask is used and the user is allowed to dial all 7 digits.

1.2.30 GROUP_ID

The command is for DPMR mode only. A separate command DMR_GROUP_ID is used for DMR mode.

Each DPMR terminal can be assigned an individual ID (own ID) and one or several group ID's. When an incoming called ID matched with the receiver's individual ID or group ID, the receiver responds. The GROUP_ID field is used to pass the group ID to CT3258.

This field contains the group ID in 7 BCD codes and one group number.

Field Identifier	Group ID and group number
1 Byte	4 Bytes
0x29	DATA[0]= {K1, K2} DATA[1]= {K3, K4} DATA[2]= {K5, K6} DATA[3]= {K7, Group Number}, where group number can be 1-15
Default	0,0,0,0

Table 1-55 GROUP_ID Field Format

K1, … , K7 are 0-9 for BCD numbers, or 10 for '*'.

The command fields support up to 15 group numbers.

1.2.31 SCRAMBLER_SEED

This command enables/disables the voice scrambler for encryptions and selects the 32 bit scrambler seed.

Field Identifier	Scrambler Type	Scrambler seeds
1 Byte	1 Bytes	0/4 bytes
0x2A	0: Disable the Scrambler 1: Enable 32 bit scrambler (DPMR Mode only) 4: Enabled 16 bit scrambler	0 byte if scrambler type = 0; 4 bytes if scrambler type = 1; DATA[0] = SEED[31:24] DATA[1] = SEED[24:16] DATA[2] = SEED[15:8] DATA[3] = SEED[7:0] 2 bytes if scrambler type = 4; DATA[0] = SEED[15:8] DATA[1] = SEED[7:0]
Default	0	0,0,0,0

Table 1-56 SCRAMBLER_SEED Field Format

Note: The 32 bit scrambler is indeed a 16 bit scrambler. The first 16-bit is used as scrambler seed which the end user can program. The second 16-bit is an XOR mask, which can be used by radio manufacturers to purposely make their radio to be incompatible with radio made by other manufactures. The second 16-bit should have at least 5 “1” to be effective.

For the 16 bit scrambler, when this command is set in TX, please set the PM in HT_CI_PM command to enable the encryption function.

1.2.32 MOD_GAIN

This message set the modulation gain for the modulator.

Field Identifier	Event
1 Byte	2 Byte
0x02C	Linear gain applied to the modulator in Q16.11 (16 bits total, 11 bits fraction, MSB first), with 2048 as 0 dB

Table 1-57 MOD_GAIN Field Format

1.2.33 CODEC_SELECT

This command selects the codec that works with CT3258. Current supported codecs include TLC320AIC3204 from TI and WM8758B from Wolfson.

Field Identifier	Codec Selection
1 Byte	1 Bytes
0x02D	1: TLC320AIC3204 2: WM8758B 3: ES8338 Others: reserved.
Default	2

Table 1-58 CODEC_SELECT Field Format

1.2.34 SET_I2C_ADDRESS

CT3258 also supports other types of codecs, as long as it has an I2C interface for control and a serial interface for data transfer which conforms with CT3258 format and timing. The I2C address of the codec can be set the command SET_I2C_ADDRESS

Field Identifier	I2C Address
1 Byte	1 Bytes
0x02E	I2C address to pass to codec
Default	0x34

Table 1-59 SET_I2C_ADDRESS Field Format

1.2.35 SYNC_MISS_COUNT

Once a call is set up, CT3258 monitor the link conditions and continuously check the presence of synchronization word, (FS2 in the case DPMR and Voice SYNC pattern in DMR). It reports SYNC miss if the number of sync miss is greater than a threshold. This command set the threshold.

Field Identifier	Sync Miss Count
1 Byte	1 Bytes
0x02F	Threshold for sync miss counter before reporting to MCU. 0-254: threshold of sync miss to exceed (greater than) before report sync miss 255: never report sync miss
Default	255

Table 1-60 SYNC_MISS_COUNT Field Format

The message to report Sync miss is NEW_EVENT with field value of 0x08.

1.2.36 Q_PRO_STR

This command enquires for product code.

Field Identifier
1 Byte
0x030

Table 1-61 Q_PRO_STR Field Format

The response packet returns a string: “CT3258” or other product code.

Field Identifier	Number of Bytes	Data
1 Byte	1 Byte	Variable Number of bytes
0x30	Number of bytes	“CT3258F”

Table 1-62 Q_PRO_STR Field Options

1.2.37 Q_HW_VER

This command enquires for hardware version.

Field Identifier
1 Byte
0x031

Table 1-63 Q_HW_VER Field Format

The response packet returns a string: “V0.01.01” or other hardware version.

Field Identifier	Number of Bytes	Data
1 Byte	1 Byte	Variable Number of bytes
0x31	Number of Bytes	“V0.02.02”

Table 1-64 Q_HW_VER Field Options

1.2.38 Q_SW_VER

This command enquires for software version:

Field Identifier
1 Byte
0x032

Table 1-65 Q_SW_VER Field Format

The response packet returns a string: “V0.01.01” or other software version.

Field Identifier	Number of Bytes	Data
1 Byte	1 Byte	Variable Number of bytes
0x32	Number of Bytes	“V1.00.04”

Table 1-66 Q_SW_VER Field Options

1.2.39 Q_CID_SN

This command enquires for customer ID and serial number.

Field Identifier
1 Byte
0x033

Table 1-67 Q_CID_SN Field Format

The response packet returns customer ID in 6 BCD numbers and serial number in 10 BCD numbers.

Field Identifier	Customer ID	Serial Number
1 Byte	3 Bytes	5 Bytes
0x33	Customer ID in 6 BCD numbers	Serial Number in 10 BCD numbers

Table 1-68 Q_CIN_SN Field Options

1.2.40 DMR_GROUP_ID

Each DMR terminal can be assigned an individual ID (own ID) and one or several group ID’s. When an incoming called ID matched with the receiver’s individual ID or group ID, the receiver responds. The DMR_GROUP_ID field is used to pass the group ID to CT3258.

This field contains the group ID in 8 BCD codes and one group number.

Field Identifier	Group ID and group number
1 Byte	5 Bytes
0x29	DATA[0]= {K1, K2} DATA[1]= {K3, K4} DATA[2]= {K5, K6} DATA[3]= {K7, K8} DATA[4] = Group Number where group number can be 1-15
Default	0,0,0,0

Table 1-69 GROUP_ID Field Format

K1, … , K8 are 0-9 for BCD numbers.

The command fields support up to 15 group numbers.

1.2.41 DC_OFFSET

This command applies a DC offset to the input signals from the ADC and to the output signal to the DAC. It also applies a digital gain to the signals to the DAC before DC offset is added.

The value of the DC offset is in 2's complements, with 0x7fff corresponds to the maximum positive DC offset, and 0x8000 corresponds to the maximum negative DC offsets. The electrical voltage of the DC offset depends on the codec types and the analog gain that is applied to the codec.

With zero analog gain, the maximum offset is capped by the supply voltage of the codec, and the minimum is capped by zero.

In the case of DAC output, a gain is applied to the output signal before DC offset is added. The gain is in linear scale, with 0x0800 corresponding to zero gain.

Field Identifier	DC offset for ADC Left Channel	DC offset for ADC Right Channel	DC offset for DAC Left Channel	DC offset for DAC right Channel	Digital Gain for DAC left channel	Digital Gain for DAC right Chanel
1 Byte	2 Byte	2 Bytes	2 Bytes	2 Bytes	2 Bytes	2 Bytes
0x039	DC offset Value, MSB followed by LSB, 0x8000-0x7fff	DC offset Value, MSB followed by LSB, 0x8000-0x7fff	DC offset Value, MSB followed by LSB, 0x8000-0x7fff	DC offset Value, MSB followed by LSB, 0x8000-0x7fff	Digital Gain Value, MSB followed by LSB, In linear Scale, Q16.11	Digital Gain Value, MSB followed by LSB, In linear Scale, Q16.11
Default	0	0	0	0	2048	2048

Table 1-70 DC_OFFSET Field Format

When paired with SCT3700, DC_OFFSET command is also used to read the DC levels and amplitudes of the signal in the RX path. In addtion, the DC levels and amplitudes of TX signal can be read by looping the TX to the RX and then read the RX DC levels and amplitudes.

To read DC levels and amplitudes, DC_OFFSET command should be sent with “Read/Write” bit in the TYPE field set to 1. The DC_OFFSET command used in “read” mode has the following format:

Field Identifier	DC Time Constant
1 Byte	1 Byte
0x39	Time constant for averaging DC and amplitude.
Default	0

Default time constant of 0 corresponds to average time of 31.25 ms, which is the time it takes to reach within 10% of the steady state energy value. If the constant is n, the average time is given by the following formula:

T = 31.25 * 2^n.

The Response to the read DC_OFFSET read command has the following format:

Field Identifier	DC offset for the Left Channel	DC offset for the Right Channel	Amplitude for the Left Channel	Amplitude for the Right Channel	Total energy of IQ channel	Division out of AMP_I/AMP_Q
1 Byte	2 Byte	2 Bytes	2 Bytes	2 Bytes	4 Bytes	1 Byte
0x039	DC offset Value, MSB followed by LSB, in Q16.15, 0x8000-0x7fff	DC offset Value, MSB followed by LSB, in Q16.15, 0x8000-0x7fff	Amplitude Value, MSB followed by LSB, in Q16.15 0x0-0x7fff	Amplitude Value, MSB followed by LSB, in Q16.15 0x0-0x7fff	I^2 +Q^2, used to do TX DC calibration, only for SCT3700,	Used for IQ calibration only for SCT3700
Default	-	-	-			-

1.2.42 ANALOG_CONFIG

This command changes the CT3258 configuration in analog mode. The parameter that can be configured in analog mode include LPF selection for 12.5 kHz or 25 kHz channel spacing, pre-emphasis and de-emphasis filter settings, and various gain values at different stages in the signal chain. All gain values are linear gain with 16 bit resolution.

Field Identifier	Mode Configure	TX gain Before Limiter	TX Gain after Limiter	CTCSS Gain	DCS Gain	RX Volume Gain
1 Byte	1 Byte	2 Bytes	2 Bytes	2 Bytes	2 bytes	2 Bytes
0x03C	See Table	16 bit linear	16 bit linear	16 bit linear	16 bit	16 bit linear

	1-67 for details	gain, MSB first, in Q16.11	gain, MSB first, in Q16.12	gain, MSB first, in Q16.12	linear gain, MSB first, in Q16.12	gain, MSB first, in Q16.11
Default	2	2048	4096	4096	4096	2048

Table 1-71 ANALOG_CONFIG Field Format

Bit Position	Bit Name	Descriptions
7	ALT_LPF	Additional attenuation at 3 kHz when BB_CH_SEL = 1
6	DCS_HPF	DCS high pass filter enable flag, for removing DC 0: DCS high pass filter is disabled 1: DCS high pass filter is enabled
5	COMPANDER	Compander enable 0: Compander is disabled 1: Compander is enabled.
4	ANA_TONE_CFG	Analog Tone configure 0: No CTCSS or DCS when sending analog tones 1: Send CTCSS or DCS when sending analog tones
3	IF_CH_SEL	IF Channel Select 0: 12.5 kHz channel 1: 25 kHz channel
2	SKIP_EMP	Skip pre-emphasis and de-emphasis filters 0: No skip pre-emphasis and de-emphasis filters 1: Skip pre-emphasis and de-emphasis filters
1	BB_CH_SEL	Base band channel filter select 0: LPF corner at 2.55k Hz 1: LPF corner at 3 kHz
0	FILTER_BYPASS	Filter by pass. All analog signal processing in the TX path and the RX path are bypassed if this bit is set.

Table 1-72 Mode Configure Details

Note that IF_CH_SEL and BB_CH_SEL can be different. IF_CH_SEL selects 12.5KHz narrow band channel selection filter or 25 kHz wide band channel selection filter. BB_CH_SEL selects the base band filter LPF corner. The following table gives a guide line for IF_CH_SEL or BB_CH_SEL settings.

IF_CH_SEL	BB_CH_SEL	Standard
0	0	12.5kHz narrow band channel, compliant with Europe CE standards (ETSI EN 300 296)
0	1	12.5kHz narrow band channel, compliant with US FCC standards (TIA 603)
1	0	N/A
1	1	25 kHz wide band channel, compliant with Europe CE standards

		(ETSI EN 300 296) and US FCC standards (TIA 603).

When BB_CH_SEL = 1, the user can configure ALT_LPF, to add an 1.5 dB attenuation at 3kHz. This is useful for two point modulation when the VCO boosts the frequency response at 3kHz, which may result in the total frequency response out of the spec.

1.2.43 CTC_PARAM

This command is used to configure CTCSS detection parameters. The CTCSS detection is done with two conditions, frequency and energy. The frequency has to be close to the designated CTCSS frequency, and the CTCSS energy has to above a certain threshold. To reduce the probability of false detection, both conditions have to be satisfied for a number of times before a CTCSS detection is declared. If one of the conditions is not satisfied for a number of times, CTCSS lost is declared. This command sets those thresholds.

Instead of using the standard phase reversal as CTCSS tail, SCT3258 can also send 55 Hz tone at the end of call as an indication for call end. “55 Hz” is used here as a general term for the special tone for call end indication. Its frequency is a programmable. CTC_PARAM command is also used to configure the mode and frequency of the 55 Hz tone. Also, as in the CTCSS tone case, a narrow band filter (NBF) is used to capture the 55 Hz tone. The coefficients for the NBF need to be set in this command if the 55 Hz tone detection is enabled.

Field Identifier	Energy_Hi	Energy_Lo	Freq_Miss	Eng_Miss	Freq_norm	Eng_Norm
1 Byte	2 Byte	2 Bytes	1 Bytes	1 Bytes	1 bytes	1 Bytes
0x3D	Energy high threshold for CTCSS found detection	Energy low Threshold for CTCSS lost detection	Number of times frequency conditions missed before declaring CTCSS lost	Number of times energy conditions missed before declaring CTCSS lost	Number of times frequency conditions satisfied before declaring CTCSS found under good condition	Number of times energy conditions satisfied before declaring CTCSS found under good condition
Default	0x0384	0x0258	6	4	2	1

Freq_Tough	Eng_Tough	Tail_180	Tail_120	CTC_Tail_DI	55 Hz Tail Configuration
1 Byte	1 Bytes	1 Bytes	1 Bytes	1 Bytes	1 Byte
Number of times	Number of times energy	180 degree CTCSS tail	120 degree CTCSS tail	CTCSS tail detection	55 Hz Tail Configuration,

frequency conditions satisfied before declaring CTCSS found under tough condition	conditions satisfied before declaring CTCSS found under tough condition	detection threshold. Higher threshold make it more difficult to detect tail.	detection threshold. Higher threshold make it more difficult to detect tail.	index, larger index value makes detection easier. A value of 0 is the hardest.	See Table below for details
7	6	0x80	0xd0	0	0

Tail Frequency	Tail NBF Coefficients	Signal to Total Energy Ratio	Reserved
2 Bytes	4 Bytes	1 Byte	2 bytes
Special Tail frequency, in steps of 0.1 Hz (MSB first). 55 Hz as default.	Narrow band filter coefficients for the special tail frequency. See Section 1.2.27 for details of the coefficients	Signal to Total Energy Ratio Threshold for CTCSS detection. The larger the threshold, the more difficult to detect	
2, 38	0, 118, 3, 179	0xb0	0

Table 1-73 CTC_PARAM Field Format

Bit Position	Bit Name	Descriptions
7	ANA_END_SEND	Flag for sending end of transmission 0: No 17 0A sent after CTCSS/DCS tail is sent 1: 17 0A sent after CTCSS/DCS tail is sent
6	CTCSS_TOL	CTCSS detection tolerance 0: No tolerance if received CTCSS frequency is off spec 1: More tolerance if received CTCSS frequency is off spec.
5	TAIL_LENGTH	Flag for analog tail length for CTCSS and DCS 0: tail length 180 ms 1: tail length 250 ms
4	HPF_CORNER	Flag for analog High pass filter corner 0: HPF corner at 300 Hz 1: HPF corner at 450 Hz
3	CTC_TAIL_RX	55 Hz tail detection enable in CTCSS call:

		0: 55 Hz Tone detection enabled 1: 55 Hz Tone detection enabled
2	VOICE_TAIL_RX	55 Hz tail detection enable in call with no sub-audio: 0: 55 Hz Tone detection enabled 1: 55 Hz Tone detection enabled
1	CTC_TAIL_TX	55 Hz tail enable in CTCSS call: 0: No 55 Hz Tone are sent at the end of call 1: 55 Hz Tone are sent at the end of call
0	VOICE_TAIL_TX	55 Hz tail enable in voice call with no sub-audio: 0: No 55 Hz Tone are sent at the end of call 1: 55 Hz Tone are sent at the end of call

Table 1-74 55 Hz Tail Configuration Details

1.2.44 VOCOER_IO_SET

This field selects the input and output settings for vocoders. It is most used for voice recording and play back. Currently, this command is only used in DPMR mode only.

Field Identifier	Recording/Play Back Setting
1 Byte	1 Byte
0x03e	Vocoder IO configuration, See table below
Default	0

Table 1-75 VOCODER_IO_SET Field Format

Bit Position	Bit Name	Descriptions
7	Reserved	Reserved
6	MOD_IN	Modulator input select: 0: from vocoder 1: from MCU (through CHAN_D commanad)
5:4	ENC_OUT	Voice encoder output select: 00: Voice encoder output to modem 01: Voice encoder output to MCU 10: Voice encoder output to modem and MCU
3	DEC_IN	Voice decoder input select: 0: from demodulator 1: from MCU (through CHAN_D commanad)
2	DEMOD_OUT	Demodulator output select 0: to voice decoder 1: to MCU (through CHAN_D commanad) and to voice decoder
1	Reserved	Reserved
0	Reserved	Reserved

Table 1-76 Vocoder IO Configuration Details

With VOCODER_IO_SET command, the user can implement functions like local/remote recording and local/remote play back. The following table gives example of the settings.

Function	Vocoder IO Settings
Local Recording	0x20
Remote Recording	0x04
Local Play back	0x08
Remote Play back	0x40

Table 1-77 Vocoder IO Settings for Different Functions

1.2.45 MODOUT_CONFIG

This command is used to configure the phase delay between the two channels in a two point modulation scheme. Adding a phase delay between the channels is useful to achieve a flat frequency response in two point modulation. The minimum step size is 10.4 us. And maximum delay is plus or minus 416 us. For example, if the phase delay time is set to 0x08,the signal at LOR is 83.2 us after the signal at LOL .If phase delay time is set to 0xf8, then the signal at LOR is 83.2 us a head of with the signal at LOL.

Field Identifier	Phase delay time configure	Reserved
1 Byte	1 Byte	4 Bytes
0x041	Minimum to -40 max to 40, with a step size of 10.4 us	Reserved
Default	0	0,0,0,0

Table 1-78 MODOUT_CONFIG Field Format

1.2.46 IQCOMP_COEF

This command is used when paired with SCT700 only.

This command is used to configure the IQ compensation filter of for the RX path and the TX path. If CT3258 works in IQ input or IQ output mode, the I path and the Q path of the RF front en should be perfectly matched, or the performance suffers. However, the I path and the Q path are never perfectly matched in the analog world. IQ compensation filter are used to re-aligned the IQ signals so that they are matched. The figure below shows the control data and control signals of IQ compensation filter.

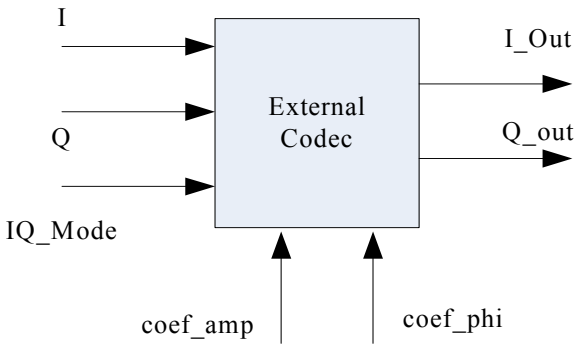

Figure 1-1 IQ Compensation Filter

The IQ compensation filter has two operation modes: the normal mode and the calibration mode. In normal mode (IQ_Mode = 0), the IQ compensation filter is enabled. In calibration mode (IQ_Mode = 1), the IQ compensation filter is disabled, allowing the algorithm to evaluate the amount of IQ imbalance, and calculate the filter coefficients coef_amp and coef_phase accordingly.

The default values of coef_amp and coef_phi are 0, which should be the value if I and Q path are perfect aligned and need to compensation.

In calibration mode, the value coef_phase has no effect, while coef_amp has an effect to the output. To complete disable IQ compensation filter, coef_amp should be set to zero, while IQ_Mode is set to 1.

CT3258 has internal algorithm to aid the calculation of the IQ compensation filter coefficients. The user can use the DC_OFFSET read command to see the effect of the calibration.

The structure of the TX IQ compensation filter and the RX IQ compensation filter are identical. Both compensation filters are configured with IQCOMP_COEF command.

The contents of the IQCOMP_COEF command are described in the table below.

Field Identifier	Mode Configure	rx_coef_amp	rx_coef_phi	tx_coef_amp	tx_coef_phi
1 Byte	1 Byte	2 Bytes	2 Bytes	2 Bytes	2 bytes
0x042	See Table 1-80 for details	RX amplitude coefficient, 16 bit linear value, MSB first, in Q16.14	RX phase coefficient, 16 bit linear value, MSB first, in Q16.14	TX amplitude coefficient, 16 bit linear value, MSB first, in Q16.14	TX phase coefficient, 16 bit linear value, MSB first, in Q16.14
Default	2	0	0	0	0

Table 1-79 IQCOMP_COEF Field Format

Bit Position	Bit Name	Descriptions
7	Reserved	Reserved
6	RX_IQ_MODE	RX IQ compensation Filter work mode 0: Enable RX IQ compensation Filter, with coefficients described in the subsequent field parameters 1: RX IQ compensation Filter in calibration mode (only rx_coef_amp is effective)
5-3	Reserved	Reserved
2	TX_IQ_MODE	TX IQ compensation Filter work mode 0: Enable TX IQ compensation Filter, with coefficients described in the subsequent field parameters 1: TX IQ compensation Filter in calibration mode (only tx_coef_amp is effective)
1-0	Reserved	Reserved

Table 1-80Mode Configure Details

If Bit 6 (RX_IQ_MODE) is set to 1, the RX phase coefficients in the command do not take effect. Likewise, the TX phase coefficients in the command does not take effect unless Bit 2 (TX_IQ_MODE) of the mode byte is 0.

IQCOMP_COEF command is also used to read the IQ compensation filter coefficients currently used by CT3258, either calculated by CT3258 or set previously by the MCU. The response packet to the read command is identical to configuration packet.

1.2.47 DIGC_DATA_FRAME

This field contains type 1, 2, 3 data, voice and appended data, transparent data and short appended data.

Field Identifier	Frame Property	Data Length	Data Bytes
1 Byte	1 Byte	1 Byte	Up to 180 bytes
0x43	See the following tables for detail	0-180: Indicate the length of the Data Others: reserved	DATA[0] DATA[1] … DATA[DataLength-1] Contain data, start from DATA[0], MSB first
Default	NA	NA	NA

Table 1-81 DIGC_DATA_FRAME Field Format

The frame property field for DPMR and for DMR are different.

Bit Position	Bit Name	Descriptions
7	DATA_VALID	Flag to indicate whether current frame has valid data 0: Current frame has no valid data. Far end can discard current frame 1: Current frame has valid data.
6	CONT_FLAG	Continuation flag 0: Data Continues after this frame 1: Data finish at this frame CT3258 does not process this field. It just passes this bit to the far end MCU. The MCU should not send more DIGC_DATA_FRAME commands after CONT_FLAG is set to one.
5	FEC_ERROR	FEC error flag 0: No FEC error in the received frame 1: FEC error in the received frame
4	Reserved	Reserved
3-0	Frame Type	Frame types: 0: Type 1 Data, Valid Data Length is 0 to 36 1: Type 2 Data, Valid Data Length is 0 to 20 2: Type 3 Data, Not implemented yet 3: Voice and Appended Data, Valid Data Length is 0 to 20 4: Transparent Data, Valid Data Length is 0 to 48 5. Short Appended Data, Valid Data Length is 9, 18, 27, 36 depending on the number of appended data frames. Others, reserved

Table 1-82 Frame Property Details in DPMR Mode

Bit Position	Bit Name	Descriptions
7-5	Reserved	
4	Data/Voice Burst	Data or voice burst selection 0: Data Burst 1: Voice Burst
3-0	Data Type	DMR data type (for data burst only) 0: PI header 1: Voice LC header 2: Terminator with LC 3: CSBK 4: MBC Header 5: MBC Continuation 6: Data Header 7: Rate 1/2 Data 8: Rate 3/4 Data

		9: Idle 10: Rate 1 Data 11-15: Reserved

Table 1-83 Frame Property Details in DMR Mode

At the transmitter, the MCU passes the contents of data burst to CT3258 through this field. CT3258 adds FEC according to the data type and puts into a TDMA burst. At the receiver, CT3258 applies FEC decoding to the received data burst and reports the contents to the MCU using the same field.

The content and the length of the frame is different according to the data type in DMR_DATA_FRAME. The details is describe in the below text.

When Data Type is PI header , Voice LC Header ,Terminator with LC, or the bit 4 is Voice Burst, the content of the frame is FULL LC PDU(without CRC). The length is 9 bytes.

Information element	Length	Remark
Protect Flag (PF)	1	
Reserved	1	
Full Link Control Opcode (FLCO)	6	
Feature set ID (FID)	8	The FID shall be either SFID or MFID
Error Full LC Data	56	(see note 1)
NOTE 1: The data information element is defined by the feature protocol document TS 102 361-2 [5].		

Table 1-84 FULL LC PDU

When Data Type is CSBK or MBC Header, the content of the frame is Control Signalling Block (CSBK) PDU (without CRC). The length is 10 bytes.

Information element	Length	Remark
Last Block	1	This bit shall be set to 1
Protect Flag	1	
CSBK Opcode(CSBKO)	6	
FID	8	The FID shall be either SFID or MFID
CSBK Data	64	
NOTE : The data information element is defined by TS 102 361-2[5]		

Table 1-85 CSBK PDU

When Data Type is MBC Continuation, the content of the frame can be MBC Continuation Block or MBC Last Block.The length is 12 bytes.

Information element	Length	Remark
Last Block	1	

MBC data	95	

Table 1-86 MBC Continuation Block

Information element	Length	Remark
Last Block	1	
MBC Data	79	
MBC CRC	16	Filled with 0, calculated by the chip automatically

Table 1-87 MBC Last Block

When Data Type is DATA_HEADER, the content of the frame is DATA_HEAD_PDU. We introduce the C_HEAD_PDU only (without CRC, the other HEAD_PDU is in the TS 102 361-1 Protocol). The length is 10 bytes.

Information element	Length	Remark
Group or Individual	1	This bit is set to indicate that the destination LLID is for a group
Response Requested (A)	1	
Header Compression (HC)	1	
Pad Octet Count (POC)	1	
Format	4	Data packet identification
SAP Identifier	4	
Pad Octet Count (POC)	4	
Logical Link ID (LLID)	24	Destination
Logical Link ID (LLID)	24	Source
Full Message Flag (FMF)	1	
Blocks to Follow (BF)	7	
Re-Synchronize flag (S)	1	
Send sequence Number (N(S))	3	
Fragment Sequence Number (FSN)	4	

Table 1-88 C_HEADER_PDU

When Data Type is Rate 1/2 Data, the contents of the frame can be Confirm_R_1_2_DATA_CONTINU_PDU, UNConfirm_R_1_2_DATA_CONTINU_PDU, Confirm_R_1_2_DATA_LAST_PDU or UNConfirm_R_1_2_DATA_LAST_PDU.The length is 12 bytes. See details from table 1-65 to 1-68.

Information element	Length	Remark

Data Block Serial Number (DBSN)	7	
C-DATA CRC	9	Filled with 0, calculated by the chip automatically
User Data	80	The user data field may contain pad octets

Table 1-89 Confirm_R_1_2_DATA_CONTINU_PDU

Information element	Length	Remark
User Data	96	The user data field may contain pad octets

Table 1-90 UNConfirm_R_1_2_DATA_CONTINU_PDU

Information element	Length	Remark
Data Block Serial Number (DBSN)	7	
C-DATA CRC	9	Filled with 0, calculated by the chip automatically
User Data	48	The user data field may contain up to 6 pad octets
Message CRC	32	Filled with 0, calculated by the chip automatically

Table 1-91 Confirm_R_1_2_DATA_LAST_PDU

Information element	Length	Remark
User Data	64	(see note)
Message CRC	32	Filled with 0, calculated by the chip automatically
NOTE:The user data field may contain up to 8 pad octets.		

Table 1-92 UNConfirm_R_1_2_DATA_LAST_PDU

When Data Type is Rate 3/4 Data, the contents of the frame can be Confirm_R_3_4_DATA_CONTINU_PDU, UNConfirm_R_3_4_DATA_CONTINU_PDU, Confirm_R_3_4_DATA_LAST_PDU or UNConfirm_R_3_4_DATA_LAST_PDU.The length is 12 bytes. See details from table 1-69 to 1-72.

Information element	Length	Remark
Data Block Serial Number (DBSN)	7	
C-DATA CRC	9	Filled with 0, calculated by the chip automatically
User Data	128	The user data field may contain pad octets

Table 1-93 Confirm_R_3_4_DATA_CONTINU_PDU

Information element	Length	Remark
User Data	144	The user data field may contain pad octets

Table 1-94 UNConfirm_R_3_4_DATA_CONTINU_PDU

Information element	Length	Remark
Data Block Serial Number (DBSN)	7	
C-DATA CRC	9	Filled with 0, calculated by the chip automatically
User Data	96	The user data field may contain up to 12 pad octets
Message CRC	32	Filled with 0, calculated by the chip automatically

Table 1-95 Confirm_R_3_4_DATA_LAST_PDU

Information element	Length	Remark
User Data	112	The user data field may contain up to 14 pad octets
Message CRC	32	Filled with 0, calculated by the chip automatically

Table 1-96 UNConfirm_R_3_4_DATA_LAST_PDU

When Data Type is IDLE, the content of the frame is IDLE_PDU. The length is 12 bytes.See details in table 1-73.

Information element	Length	Remark
User Data	96	

Table 1-97 IDLE_PDU

When Data Type is Rate 1 Data, the contents of the frame can be Confirm_R_1_DATA_CONTINU_PDU, UNConfirm_R_1_DATA_CONTINU_PDU, Confirm_R_1_DATA_LAST_PDU or UNConfirm_R_1_DATA_LAST_PDU.The length is 24 bytes. See details from table 1-74 to 1-77.

Information element	Length	Remark
Data Block Serial Number (DBSN)	7	
C-DATA CRC	9	Filled with 0, calculated by the chip automatically
User Data	176	The user data field may contain pad octets

Table 1-98 Confirm_R_1_DATA__CONTINU_PDU

Information element	Length	Remark
User Data	192	The user data field may contain pad octets

Table 1-99 UNConfirm_R_1_DATA__CONTINU_PDU

Information element	Length	Remark
Data Block Serial Number (DBSN)	7	
C-DATA CRC	9	Filled with 0, calculated by the chip automatically
User Data	144	The user data field may contain up to 18 pad octets
Message CRC	32	Filled with 0, calculated by the chip automatically

Table 1-100 Confirm_R_1_DATA__LAST_PDU

Information element	Length	Remark
User Data	160	(see note)
Message CRC	32	Filled with 0, calculated by the chip automatically
NOTE:The user data field may contain up to 20 pad octets.		

Table 1-101 UNConfirm_R_1_DATA__LAST_PDU

1.2.48 DPMR_PREAMBLE_LENGTH

DPMR standard allows the length of the preamble to be more than the default 72 bits (15 milliseconds). This command sets the length of additional preambles and length of silence (before preamble) in increment of 20 milliseconds

Field Identifier	Preamble Length	Silence Length
1 Byte	1 Bytes	1 Byte
0x046	Length additional Preambles in 20 millisecond unit	Silence length before preamble in 20 millisecond unit
Default	0	0

Table 1-102 DPMR_PREAMBLE_LENGTH Field Format

1.2.49 DPMR_CALL_OPTION

This field is used to set the DPMR specific call options

Field Identifier	DPMR Call Options
1 Byte	5 Bytes
0x47	Data[0]: Bit 7: Slot timing flag 0: The message sent to far end can be sent freely 1: The message sent to far end has to be sent at slot boundary. The first bit of the preamble will be sent 30 milliseconds from the last bit of the last received message Bit 6: SQ detection enable 0: SQ detection is not done by CT3258 1: SQ detection is done by CT3258 Bit 5: IQ receive enable 0: CT3258 in IF IQ mode 1: CT3258 in base band mode Bit 4: RF control enable 0: CT3258 does not control RF chip (SCT3700) 1: CT3258 controls RF chip.

	Bit 3-0: Number of the power saving headers to precede the next message. Zero to disable power saving headers Data[1]: voice delay constant and transmit control Bit 7: IQ transmit enable 0: CT3258 transmitting in IQ mode 1: CT3258 transmitting in base band mode Bit 6: Enable Wrong Vocoder report Bit 5:0: voice delay constant 0-40: voice delay constant in steps of 40 milliseconds Data[2]: Report options Bit 7: 1B 09, 1B 0A report 0: No report of 1B 09 and 1B 0A 1: Report 1B 09 and 1B 0A Others: reserved
Default	0

Table 1-103 DPMR_CALL_OPTION Field Format

Note: In order to send extended power saving headers, DPMR_CALL_OPTION must be called each time when a communication is to start

1.2.50 DPMR_END_UE

This field contains the 17 bits of unencoded END data.

Field Identifier	17 bit of unencoded END
1 Byte	3 Bytes
0x4B	DATA[0]={ET[1:0], ARQ[1:0], TX_WAIT[3:0]} DATA[1]={STATUS[4:0], 3b000} DATA[2]=0
Default	0,0,0

Table 1-104 DPMR_END_UE Field Format

The details of ET, ARQ, TX_WAIT and status are given in the following tables.

00	Normal end frame
01	End frame with status message
10	Reserved
11	Reserved

Table 1-105 Details ET Field

00	No ACK request to called station
01	ACK request to called station

10	Reserved
11	Reserved

Table 1-106 ARQ Field Details

0000	No specified time
0001	40 ms (half a frame)
0010	80 ms (one frame)
0011	160 ms (two frames)
0100	320 ms (one superframe)
Other	Other Reserved

Table 1-107 Details of Tx WAIT Field

1.2.51 DIGC_FS1

This field contains the 48 bits of FS1 data for DPMR. It is also used to generate a fixed 48 bit test pattern for both DPMR and DMR mode.

Field Identifier	48 bit of encoded FS1
1 Byte	6 Bytes
0x4C	DATA[0]=FS1[47:40] DATA[1]=FS1[39:32] … DATA[5]=FS1[7:0]
Default	0x57,0xff,0x5f,0x75,0xd5,0x77

Table 1-108 DIGC_FS1 Field Format

1.2.52 DIGC_CALLED_ID_BIN

This field contains the called ID in 24 bit binary form.

Field Identifier	24 bit binary called ID
1 Byte	3 Bytes
0x50	DATA[0]=CALL_ID_BIN[23:16] DATA[1]= CALL_ID_BIN [15:8] DATA[2]= CALL_ID_BIN [7:0]
Default	0,0,0

Table 1-109 DIGC_CALLED_ID_BIN Field Format

1.2.53 DIGC_OWN_ID_BIN

This field contains the own ID in 24 bit binary form.

Field Identifier	24 bit binary own ID
1 Byte	3 Bytes
0x51	DATA[0]=OWN_ID_BIN[23:16] DATA[1]= OWN_ID_BIN [15:8] DATA[2]= OWN_ID_BIN [7:0]
Default	0,0,0

Table 1-110 DIGC_OWN_ID_BIN Field Format

1.2.54 DPMR_CALLED_ID_BCD

This field contains the called ID in 7 BCD codes.

Field Identifier	Called ID in 7 BCD codes
1 Byte	4 Bytes
0x52	DATA[0]= {K1, K2} DATA[1]= {K3, K4} DATA[2]= {K5, K6} DATA[3]= {K7, 0x4b0000}
Default	0xaa,0xaa,0xaa,0xa0

Table 1-111 DPMR_CALLED_ID_BCD Field Format

If the user dials less than 7 digits, the user should use 0x0F to fill the place for un-dialed digits. Upon receiving the message with less than 7 digits, CT3258 will fill the rest of the digits to complete 7 digits if layer 3 process mode is enabled.

1.2.55 DPMR_OWN_ID_BCD

This field contains the own ID in 7 BCD codes.

Field Identifier	Own ID in 7 BCD codes
1 Byte	4 Bytes
0x53	DATA[0]= {K1, K2} DATA[1]= {K3, K4} DATA[2]= {K5, K6} DATA[3]= {K7, 0x4b0000}
Default	0x00,0x00,0x00,0x00

Table 1-112 DPMR_OWN_ID_BCD Field Format

1.2.56 DPMR_M_V_F_E

This field contains communication mode (M), version (V), and communication format (F) fields.

Field Identifier	M and F fields
1 Byte	1 Byte

0x54	DATA[0]={M[2:0], V[1:0], F[1:0], EP}
Default	0

Table 1-113 DPMR_M_V_F Field Format

The meaning of M, V, F and EP are described in section CALL_START.

1.2.57 DPMR_PROTOCOL_OPTION

This field is used to set the DPMR specific protocol options

Field Identifier	DPMR Protocol Options
1 Byte	1 Byte
0x55	Bit 7: Protocol Support 0: Support TS 102 490 1: Support TS 102 490 and TS 102 658 Bit 6: NDR Support 0: No Support for NDR 1: Support for NDR Bit 5 : TS 102 658 Mode 3 Support 0: No Support for Mode 3 1: Support for Mode 3 Bit 4: Maintenance Message Report 0: Repeated Maintenance Message only reported only once. 1: Every reported messages are reported Bit 3: Reserve (Should set to zero) Bit 2: Reserve Bit 1 : All call mapping 0: Old way of all call mapping: TS 102 658 (V2.1.1 or earlier) 1: New way of all call mapping Bit 0 : 12.5 kHz DPMR support. 0: 6.25 kHz DPMR 1: 12.5 kHz DPMR
Default	0

Table 1-114 DPMR_PROTOCOL_OPTION Field Format

1.2.58 DPMR_SLD

This field contains 18 bits of slow data.

Field Identifier	M and F fields
1 Byte	4 Bytes
0x56	DATA[0]=SLD[17:10]

	DATA[1]=SLD[9:2] DATA[2]={SLD[1:0], 0x000000}
Default	0x80,0x40,0x00

Table 1-115 DPMR_SLD Field Format

The meaning of SLD field is defined in the TS 102 658 or TS 102 490:

Bit position	Meaning
17	Continuation Flag 1 0: User data continuous after the following byte 1: User data is terminated by the following byte
16-9	User data byte 1
8	Continuation Flag 2 0: User data continuous after the following byte 1: User data is terminated by the following byte
7-0	User data byte 2

Table 1-116 Details SLD Field

Note that SLD filed can also be modified by WORK_MODE command. Upon receiving command WORK_MODE with the mode set to TX or duplex, the SLD field is reset to default 0x80, 0x40, 0x00. In a typical voice call with slow data, the MCU always send WORK_MODE command before sending the first DPMR_SLD command.

1.2.59 DPMR_HT_CI_PM

This field contains header type (HT) and call information (CI) fields.

Field Identifier	HT and CI fields
1 Byte	2 Bytes
0x57	DATA[0]={HT[3:0], CI[10:7]} DATA[1]={CI[6:0], PM}
Default	0,0

Table 1-117 DPMR_HT_CI_PM Field Format

Details of HT, CI and PM can be found in the following tables.

HT indicates header type.

0000	Communication start header (a superframe follows)
0001	Connection request header (an END frame follows)
0010	Unconnect request header (an END frame follows)
0011	ACK (this a single frame, ACK or NACK is differentiated by the CI bits setting)
0100	System request header (an END frame follows) or Maintenance Message header

0101	ACK header reply to a system request (a superframe follows)
0110	System delivery header (a superframe follows)
0111	Status response header (an END frame follows)
1000	Status request header
Other	Reserved

Table 1-118 Header Type Details

PM is the channel preservation flag used for base station to send out channel preservation messages. When sending a voice call, this bit is used to enable the encryption.

PM	Channel Preservation Flag
0	Channel is not preserved
1	Channel is preserved

Table 1-119 Channel Preservation Flag Field Details

CI contains 3 bits of CI type and 8 bits of CI information.

CI Type	CI Information
3 bits	8 bits

Table 1-120 CI Format

The meaning of CI depends on the header types.

Use	Purpose
Power save	Indicate normal or extended header type
T1 or T2 Data	Indicate the type of data (supplementary service)
T3 Data (Packet)	Indicate data frame size and number of frames
Acknowledgements	Indicate ACK or NACK and reason
System request System response Delivery Header	CI Type defines the purpose CI Information is not used and set to 0000 0000

Table 1-121 CI Usuage

The details of CI for different header types are described in the following sub-sections,

1.2.59.1 Call Information for Power Save

CI Type (3 bits):

CI Type	Definition
111	Extended wake-up Header
Other value	Normal Header type

Table 1-122 CI Type for Power Save

If the extended wake-up Header is used then the last 4 information bits will show how many

Headers frames follow the current one (i.e. counting down to zero).

0000 0000	Normal Header frame
0000 0001 ---------- 0000 1111	Ext Header frame 1 ---------- Ext Header frame 15
Other	Reserved

Table 1-123 CI Information for Power Save

1.2.59.2 Call Information for Types 1 and 2 data

Data communications (types 1 and 2):

CI Type	**CI Information**	
001	**Format**	**Reserved**
	4 bits	4 bits

Table 1-124 CI Type for Type 1 and Type 2 Data

Reserved bits are set to 0000.
Format:

0000	Status message
0001	Precoded message
0010	Free text message (radio generated data)
0011	Short file transfer
0100	User defined data 1
0101	User defined data 2
0110	User defined data 3
0111	User defined data 4
Other	Reserved

Table 1-125 CI Information for Type 1 and Type 2 Data

1.2.59.3 Call Information for Type 3 (packet) data

Information bits for Packet data format (Type 3):

CI Type	**CI Information**	
011	**pdS** Frame Size	**pdM** Data Size
	4 bits	4 bits

Table 1-126 CI for Type 3 Data

Details of pdS (packet data size) and pdM (number of transmitted frames) are details by the tables below.

pdS	Frame time (ms)	Data size bits
0	80	288
1	160	672
2	240	1056
3	320	1440
Other	Reserved	Reserved

Table 1-127 Packet Data Frame Sizes (pdS)

pdM	Number of Data frames
0	1 frame
1	2 frames
2	3 frames
3	4 frames
4	5 frames
5	6 frames
6	7 frames
7	8 frames
Other	Reserved

Table 1-128 Number of Transmitted Frames (pdM)

1.2.59.4 Call Information for System Transactions

System request/answer/delivery header:

CI	Type Definition
000	Reserved
001	Dynamic group request/answer/delivery
010	Reserved
011	Reserved
100	ESN request/reply
101	MFID request/reply
110	Contact station address(via Interconnect, IP)
111	Reserved

Table 1-129 CI Type for System Transactions

CI Info	Definition
0000 0000	All bits set to zero (the data size is indicated in the CCH SLD field)

Table 1-130 CI Information for System Transactions

1.2.59.5 Call Information for Acknowledgments

Acknowledgment:

CI Type	Definition
000	Reserved
001	ACK (Rx OK)
010	NACK (data error, resend request)
011	NACK (request denied)
Other	Reserved

Table 1-131 CI Type for Acknowledgements

CI Info	Definition
0	
1 to 255	ACK/NACK status (rejection reason defined by user)

Table 1-132 CI Information for Acknowledgements

1.2.59.6 Call Information for appended data

CI type:

CI	Type Definition	Meaning
000	Short Data	Service Requested is Short Data
011	Call Diversion	Call Diversion Service

Table 1-133 CI type for Appended data

Short data CI Info:

CI Info	length	Meaning
UAD(0 - 3)	[7:6],2 bits	Appended Short Data Number of appended UDTs required to transport short data
SYMB	[5:0],6 bits	Number of symbols in the short data ,

Table 1-134 CI Information for Short data

Call Diversion CI Info:

CI Info	Length	Meaning
UAD(00)	[7:6],2 bits	Number of Appended Data = 1
SYMB	[5:0],6 bits	N/A for call diversion

Table 1-135 CI Information for Call Diversion

1.2.59.7 Call Information for Maintenance Message

CI type	Type Definition	CI Info
000	IDLE Message	N/A
001	Guard Message	See table below
	Preservation message	See note

Table 1-136 Call Information for Maintenance Message

CI Info	length	Meaning		
Reserved	[7:4],4bits	Reserved		
Guard Kind	[3:0],4 bits	0000	Reserved	
		0001	DIS_PTT	Disable Target MS or Talkgroup PTT
		0010	EN_PTT	Enable Target MS or Talkgroup PTT
		0011	ILLEGALLY PARKED	Clear down from the payload channel MS whose address does not match,Source or Target Address
		0100 to 1111	Reserved	

Table 1-137 Call Information for Guard Message

Note: Preservation message is not identified by CI. It is identified with PM as 1. In preservation message, CI keeps value of previous traffic channel down link messages, along with CALLED ID, OWN ID, M, V, F, and EP fields.

1.2.60 DPMR_CI

This field contains 11 bits call information. It is used for reporting CI information by CT3258 to the MCU upon receiving an extended header.

Field Identifier	CI field
1 Byte	2 Byte
0x5A	DATA[0]={CI[10:3]} DATA[1]={CI[2:0], 0x5b00000}
Default	0,0

Table 1-138 DPMR_CI Field Format

1.2.61 DPMR_CC

This field contains the 24 bits of CC data for the transmitter and the receiver.

Field Identifier	24 bit of encoded CC
1 Byte	3 Bytes
0x5C	DATA[0]=CC[23:16] DATA[1]=CC[15:8] DATA[2]=CC[7:0]
Default	0x57, 0x75,0x77

Table 1-139 DPMR_CC Field Format

1.2.62 DPMR_SEND_SF

Send super frame command. Upon receiving this command, CT3258 assemble a super frame and send to the far end.

Field Identifier
1 Byte
0x5F

Table 1-140 DPMR_SEND_SF Field Format

1.2.63 DPMR_SEND_HEADER

Send header frame command. Upon receiving this command, CT3258 assemble a header frame and send to the far end.

Field Identifier
1 Byte
0x60

Table 1-141 DPMR_SEND_HEADER Field Format

1.2.64 DPMR_SEND_END

Send end frame command. Upon receiving this command, CT3258 assemble an end frame and send to the far end.

Field Identifier
1 Byte
0x61

Table 1-142 DPMR_SEND_END Field Format

1.2.65 DPMR_SEND_AD

Send DPMR_SEND_AD command. Upon receiving this command,CT3258 assemble a series of appended data frmae，include HEADER FRAME ,APPENDED DATA FRAME (according as appended data number),END FRAME.

Field Identifier	Call Details
1 Byte	1 Byte
0x62	DATA[0]={M[2:0], V[1:0], F[1:0], EP} Call Mode, Version, Call Format, Emergency Priority

Default	0

Table 1-143 DPMR_SEND_AD Field Format

DPMR_SEND_AD can only be used in TS 102 658 PROTOCOL.

The meaning of M, V, F and EP fields are described in the section CALL_START.

Note: Please set the content of appended data frame with DIGC_DATA_FRAME command before sending appended data frame with this command.

1.2.66 DIGI_MIC_GAIN

This message set the modulation gain for the microphone used in the digital mode only.

Field Identifier	Event
1 Byte	2 Byte
0x068	Linear gain applied to the microphone in Q16.11 (16 bits total, 11 bits fraction, MSB first), with 2048 as 0 dB. This gain is used in digital mode only

Table 1-144 DIGI_MIC_GAIN Field Format

1.2.67 DIGI_SPEAKER_GAIN

This message set the modulation gain for the speaker used in the digital mode only.

Field Identifier	Event
1 Byte	2 Byte
0x069	Linear gain applied to the speaker in Q16.11 (16 bits total, 11 bits fraction, MSB first), with 2048 as 0 dB. This gain is used in digital mode only

Table 1-145 DIGI_SPEAKER_GAIN Field Format

1.2.68 I2C_OPERATION

This command is used to read and write raw I2C command to the I2C devices that are connected to CT3258. Examples of the connected devices include codec, security keys and RF transceiver.

This command differs from the codec bypass packet described in 1.1.3 in that codec bypass packet can only be used for certain codecs, while I2C_OPERATION can be used for any connected I2C devices.

Field Identifier	I2C device Address	Device register address	I2C Data Length	I2C data
1 Byte	1 Byte	1 Byte	1 Byte	I2C Data Length
0x06A	I2C device address	Device register address	Number of Bytes to Write.	Data[0] Data[1] … Data[I2C Data Length]
Default	0	0	0	N/A

Table 1-146 I2C_OPERATION Write Field Format

To read from I2C device, the user should set TYPE field to 0x20, and the I2C_OPERATION syntax is as follows:

Field Identifier	I2C device Address	Device register address	I2C Data Length
1 Byte	1 Byte	1 Byte	1 Byte
0x06A	I2C device address	Device register address	Number of Bytes to Read
Default	0	0	0

Table 1-147 I2C_OPERATION Read Field Format

The response packet returns the I2C data

Field Identifier	Data read from I2C
1 Byte	I2C Data Length
0x6A	Data[0] Data[1] … Data[I2C Data Length-1]
Default	0

Table 1-148 I2C_OPERATION Read Response Field Format

1.2.69 MISC_GAIN

This command set the various gain levels, including the input and output gain for IQ signal, and CTCSS/DCS receive gains, and frequency of the signal used for calibration, and DC filter settings.

Field Identifier	IQ TX Gain	IQ RX Gain	CTCSS/DCS Receive Gain	Calibration Signal Frequency	DC Filter Configuration	Reserved
1 Byte	2 Bytes	2 Bytes	2 Bytes	2 Bytes	1 Byte	3 Bytes
0x06B	16 bit linear	16 bit linear	16 bit linear	16 bit linear	See table	

	gain, MSB first, in Q16.11	gain, MSB first, in Q16.11	gain, MSB first, in Q16.11	gain, MSB first, in Q16.11	below for detail.	
Default	2048	2048	2048	2048	0	

Table 1-149 MISC_GAINS Field Format

Bit Position	Bit Name	Descriptions
7-2	Reserved	
1	DC Filter enable	0: DC filter is disabled 1: DC filter is enabled.
0	Manual DC Mode	0: Auto DC Mode, DC filter is enabled at low input level and disabled at high input level for digital calls, and always enabled for analog call. 1: Manual DC Mode. See bit 1 for details.

Table 1-150 DC Configuration Detail

The IQ RX gain is applied to the input IQ signals (when IQ receive is enabled) after it is received by CT3258. The IQ TX gain is applied to the output IQ signals (when IQ transmit is enabled) before it is sent to the codec.

The CTCSS/DCS receive gain is used in analog mode, and is applied to the CTCSS/DCS signals after the 300 HF LPF. It is used to adjust the level of DCS subaudio signals without affecting the level of audio signals.

The calibration signals are I / Q sinusoidal signals generated by CT3258 TX path, which can be looped back by SCT3700 to CT3258 RX for I Q calibration. The frequency of the signals can be programmable with this command. A default value of 2048 corresponds to a frequency of 350 Hz.

1.2.70 SQ_LEVEL

This command is used for calibration of SQ detection. It is also used to read the RSSI and OOBE levels.

Field Identifier	GAIN_ADJUST	EXT LNA_GAINL	Forced RF Gain	OOBE_INTV	OOBE_HI_D	OOBE_LO_D	OOBE_HI_A	OOBE_LO_A
1 Byte	1 Byte	1Byte	1 Byte	1 Byte	2 Bytes	2 Bytes	2 Bytes	2 Bytes

0x06C	Gain Adjustment, from -128 dB to +127 dB	External LNA value. Refer to table below.	Forced RF Gain Setting. See table below for Details	OOBE detection interval in units of 1.333 ms.	Digital OOBE high threshold, 16 bit linear value	Digital OOBE low threshold, 16 bit linear value	Analog OOBE high threshold, 16 bit linear value	Analog OOBE low threshold, 16 bit linear value
Default	0	0	0	90	0x4000	0x0400	0x4000	0x0400

Table 1-151 SQ_LEVEL Field Format

Bit Position	Bit Name	Descriptions
7	Forced RF Gain Enabled	0: Forced RF Gain Disable. RF gain controlled by AGC algorithm 1: Forced RF Gain Enabled. RF gain are setting by bit 6-0.
6-0	Gain Details	RF Gain Details: It is different for RDA1847 or SCT3700 For RDA1847, 0-15, Gain Index Others, reserved For SCT3700 Bit 6-4: LNA gain setting 001: highest gain 010: highest gain -6 dB 011: highest gain -12 dB 100: highest gain -24 dB 101: highest gain -36 dB 110: highest gain -48 dB Others: reserved Bit 3-0: PGA settings: 0000: lowest gain 0001-1100, 2-24 dB, 2 dB a step Others, reserved

Table 1-152 Force RF Gain Details

Bit Position	Bit Name	Descriptions
7:0	External LNA gain Setting	Only For SCT3700 Bit 7:6: Reduction of AGC target level. 00: Maximum AGC target level used 01: Maximum AGC target level – 6 dB

		10: Maximum AGC target level – 12 dB 11: Maximum AGC target level – 18 dB Bit 5:0: External LNA gain in db if it is used. If the external LNA is not used, the value is always zero. If the external LNA is used, it is the difference of external gain value when external LNA is enabled and when it is disabled. Please refer to Chapter 2.6 for its usage.

Table 1-153 External LNA setting Details

GAIN ADJUST is used to account for all unspecified gains in the analog path. It may be different with different analog implementations. It may be even different from radio to radio. The RSSI is correct only if the value is set correctly. If external LNA is used, it need to be disabled when calibrating GAIN_ADJUST.

If external LNA is not used, "EXT LNA GAIN" should be assigned to its default value of zero.

If external LNA is used, the user should set EXT LNA GAIN appropriately to reflect the characteristic of the external LNA. It involves two part. Bit 5-0 is used to set the value of LNA gain in dB. It reflect the receive gain increase when the external LNA is enabled from when it is disabled. Also, if the external LNA is used, the user should lower the maximum SCT3700 AGC gain to avoid saturating the RX signals. The amount of maximum AGC target level decrease are set by bit 7-6 of "EXT LNA GAIN" fields. The following is a guide line:

00: if the external LNA gain is less than 6 dB.
01: if the external LNA gain range in 6-12db
10: if the external LNA gain range in 12-18db
11: if the external LNA gain range in 18-24db

OOBE indicates the energy of FM demodulated signals at frequency band above 3 kHz. OOBE value is small when RF carrier signal exists, and large when no carrier signals present. It is a good indication for SQ (or carrier) detection. Typically, when carrier signals present, OOBE level is below 8.

OOBE level has a high threshold and a low threshold for SQ detection, with the high threshold larger than the low threshold. To detect the presence of carrier, the OOBE level is compared with the low threshold. Carrier detection is reported if the OOBE level is above the low threshold. Once carrier is found, the OOBE level is compared with the high threshold. If the OOBE level is lower than the high threshold, carrier lost is reported.

The OOBE levels for analog and digital calls are different. As a result, CT3258 has two sets of high and low OOBE threshold for digital and analog separately.

SQ_LEVEL command is also used to read narrow, wide band RSSI and OOBE levels. The response packet has the following format.

Field Identifier	NB_RSSI	WB_RSSI	OOBE Level	Inband Level
1 Byte	1 Byte	1 Byte	2 Byte	2 Byte
0x06C	Narrow band RSSI + 137	Wide band RSSI + 137	OOBE level	Inband signal energy level
Default				

There are two RSSI estimators in the CT3258. One of them is narrow band RSSI, used to estimate the energy of signals in the channel of interests. The other is wide band RSSI, used to estimate the energy of the signal in the channel of interests and surrounding channels. All RSSI values in this command are of dB scale.

All RSSI values are expressed as unsigned 8 bit integer. The real RSSI is this value minus 137. For example, for WB_RSSI value of 17, the real wide band RSSI is 17 – 137 = -120 dBm.

1.2.71 SPI_OPERATION

This command is used to write raw SPI command to the SPI devices that are connected to CT3258. Examples of the connected devices include SCT3700.

Field Identifier	SPI Register Address	SPI register Contents
1 Byte	1 Byte	1 Byte
0x06D	SPI Register Address	SPI register Contents
Default	0	0

Table 1-154 SPI_OPERATION Write Field Format

The SPI register table for SCT3700 can be found in SCT3700 data sheet.

1.2.72 DMR_SLC

This command is used to report the short LC data from the base station.

Field Identifier	Offset
1 Byte	4 Bytes
0x06E	Contains 28 bit of SLC data
Default	0

Table 1-155 DMR_SLC Field Format

1.2.73 DMR_CALL_SLOT

This field is used to set the slot number for the transmitting or receiving calls.

DMR_CALL_SLOT	Call	Timing Update	Direct TDMA	Rev	Slot mode
1 BYTE	BIT[7]	BIT[6]	BIT[3]	BIT[4]	BIT[3:0]
0x6F	0:Receive Call 1:Transmit Call	0: keep current TX timng 1: Update TX timing according RX timing	0: Direct TDMA Mode disabled 1: Direct TDMA mode enabled	Reserved	0:random 1:slot1 2:slot2 3:both(only Rx)

Table 1-156 DMR_CALL_SLOT Field Format

Note that the slot number is in terms of mobile station slot number, or downlink slot number.

1.2.74 EQULIZER_FILTER

This command configures the coefficients for the audio equalizer filters. A total of up to three equalizer filters can be enabled. Each filter is a second order IIR filter implemented in straight form. The transfer function of the filters are shown in the following equation:

H(z) = (b0 + b1 * z^(-1) + b2*z^(-2)) / (1 + a1 * z^(-1) + a2 * z^(-2))
The sampling rate of the filter is 24 kHz.

This command gives the filter coefficients. All coefficients are in Q16.14 format (16 bit total, with 14 bit fraction), with MSB first

By default, no equalizer filter is enabled.

Field Identifier	Filter Selection Byte	a1	a2	b0	b1	b2
1 Byte	1 Byte	2 Bytes	2 Bytes	2 Bytes	2 Bytes	2 Bytes
0x70	Bit 7-6: reserved Bit 5: TX or RX selection 0: Filter at RX side 1: Filter at TX side Bit 4: Enable flag 1: Selected filter enabled 0: Selected filter not enabled Bit 3-0: Filter selection	a1 in Q16.14 MSB first	a2 in Q16.14 MSB First	b0 in Q16.14 MSB First	b1 in Q16.14 MSB first	b2 in Q16.14 MSB First

	1: Filter 1 selected 2: Filter 2 selected 3: Filter 3 selected Others: reserved					
Default	0	-	-	-	-	-

Table 1-157 EQUALIZER FILTER Field Format

1.2.75 DMR_FLC

This message set the information for the FLC in a DMR control packet.

Field Identifier	FLC
1 Byte	3 Byte
0x076	Data[0] = {PF, REV, FLCO[5:0]} Data[1] = {FID[7:0]} Data[2] = {SeversOption[7:0]}

Table 1-158 DMR_FLC Field Format

Details of the elements are defined in the DMR standard (TS 102 361)

PF field is defined by：

PF	**Protect Flag**
0	Reserved

Table 1-159 PF Settings

FLCO field is defined by:

FLCO	**Full Link Control Opcode**
000000	Group Voice Call（Grp_V_Ch_Usr）
000011	Unit to Unit Voice Call（UU_V_Ch_Usr）

Table 1-160 FLCO Settings

FID field is defined by:

FID	**Feature set ID**
0000 0000	Standardized feature set ID for the services
0000 0001	Reserved for future standardization
0000 0010	Reserved for future standardization
0000 0011	Reserved for future standardization
0000 0100	Manufacture’s specific feature set ID(MFID)
Etc.	Etc

0111 1111	Manufacture's specific feature set ID(MFID)
1xxx xxxx	Reserved for future MFID's allocation(MFID)

Table 1-161 FID Settings

Service Option field is defined by:

Service option element	Bit position	Value remark
Emergency	Bit[7]	0:Non-Emergency service 1:Emergency service
Privacy	Bit[6]	0:privacy is not defined
Reserved	Bit[5:4]	00:Reserved for future
Broadcast	Bit[3]	0:Non-broadcast service
Open voice call mode(OVCM)	Bit[2]	0:Non_OVCM call 1:OVCM call
Priority level	Bit[1:0]	00: NO Priority 01: Priority 1 10: Priority 2 11: Priority 3 Note: Priority 3 is the highest Priority

Table 1-162 Service Option Settings

1.2.76 DMR_CC

This message set the color code index for DMR calls

Field Identifier	CC
1 Byte	1 Byte
0x077	0: CC0 is used 1: CC1 is used … 15: CC15 is used

Table 1-163 DMR_CC Field Format

1.2.77 DMR_CALL_START

This command is used to start a call. Upon receiving this command, CT3258 begins to send call to the far end.

Field Identifier
1 Byte
0x078

Table 1-164 DMR_CALL_START Field Format

1.2.78 DMR_CALL_OPTION

This field is used to set the DMR specific call options

Field Identifier	DMR Call Options
1 Byte	5 Bytes
0x79	Data[0]: Bit 7: unused slot fill pattern 0: unused slot filled with 0 1: unused slot filled with random bits. Bit 6: SQ detection enable 0: SQ detection is not done by CT3258 1: SQ detection is done by CT3258 Bit 5: IQ receive enable 1: CT3258 in IF IQ mode 0: CT3258 in base band mode Bit 4: RF control enable 0: CT3258 does not control RF chip (SCT3700 or RDA 1847) 1: CT3258 controls RF chip. Bit 3: RF slot timing control (SCT3700 Only) 0: No control of SCT3700 TX signal during transmission 1: Turn on and off SCT3700 TX signal aligned with the base band slot timing. Bit 2: RF_TIMING RX mode enable 0: RF_TIMING port outputs slot timing information in TX mode only 1: RF_TIMING port outputs slot timing information in TX mode and RX mode Bit 1-0: Traffic mode 0: DMR continuous transmission mode 1: DMR direct mode with no reverse channel 2: DMR direct mode with reverse channel Data[1]: voice delay constant and IQ transmit mode Bit 7: IQ transmit enable 1: CT3258 transmitting in IQ mode 0: CT3258 transmitting in base band mode Bit 6: Reserved Bit 5:0: voice delay constant 0-40: voice delay constant in steps of 40 milliseconds Others: reserved

	Data[2]: Bit 7-4: Number of good CACH received before reporting DMR_SLOT_FOUND with the slot verified flag set. Bit 3-0: Number of repeated LC header Data[3]: Bit 7-6: Reserved Bit 5-0: Bit mask for reporting repeated identical messages. Bit 4: 0, report identical SLC once 1, report every SLC Bit 3: 0: Report identical IDLE data frames once 1: Report all IDLE data frames Bit 2: 0: Report identical CSBK data frames once 1: Report all CSBK data frames Bit 1: 0: Report identical LC headers once 1: Report all LC headers Bit 0: 0: Report identical LC terminators once 1: Report all LC terminators Data[4]: Reserved
Default	0

Table 1-165 DMR_CALL_OPTION Field Format

1.2.79 DMR_OFFSET

This command set the timing offset of the RF_TIMING signal to the internal slot timing signal, which is aligned with base station timing. With default parameters, the RF_TIMING signal is identical to the internal slot timing signal, which has a 60 millisecond period and 50% duty cycle, with the rising edge aligned with the end of a BS slot. With DMR_OFFSET command, the RF_TIMING signal can be ahead or lag of the internal slot timing signals.

Field Identifier	Offset	Length
1 Byte	2 Bytes	2 Bytes
0x07A	16 bit signed integer, MSB first. The unit is 41.7 microseconds. The range is -720 to 720, or -30 milliseconds to +30	16 bit unsigned integer, MSB first. The unit is 41.7 microseconds. The range is 240 to 1200, or 10

	milliseconds	milliseconds to 50 milliseconds
Default	0	720

Table 1-166 DMR_OFFSET Field Format

1.2.80 DMR_SLOT_TYPE

This field is used report the slot type when a call is received.

Slot Type	Colour Code	Data Type
1 BYTE	BIT[7:4]	BIT[3:0]
0x7B	Value:0-15	0000:PI header 0001:Voice LC header 0010:Terminatior with LC 0011:CSBK 0100:MBC header 0101:MBC continuation 0110:data header 0111:rate 1/2 data 1000:rate 3/4 data 1001:idle 1010:Rate1 Data Others:Reserved for future use

Table 1-167 DMR_SLOT_TYPE Field Format

1.2.81 DMR_EMB

This field is used report the EMB field when a call is received.

DMR_EMB	Colour Code	PI	LCSS	Rev
1 BYTE	BIT[7:4]	BIT[3]	BIT[2:1]	BIT[0]
0x7C	Value:0-15	Reserved	00:Single fragment LC or first fragment CSBK signalling 01:First fragment of LC signalling 10:Last fragment of LC or CSBK signaling 11:Continuation fragment of LC or CSBK	Reserved

Table 1-168 DMR_EMB Field Format

1.2.82 DMR_CALLED_ID_BCD

This field contains the called ID in 8 BCD codes.

Field Identifier	Called ID in 7 BCD codes
1 Byte	4 Bytes
0x7D	DATA[0]= {K1, K2} DATA[1]= {K3, K4} DATA[2]= {K5, K6} DATA[3]= {K7, K8}
Default	0xaa,0xaa,0xaa,0xaa

Table 1-169 DMR_CALLED_ID_BCD Field Format

K1 should be either 0 or 1. An 8 digit BCD should not exceed “16777016”. If the user wishes to dial a 7 digit BCD number “1234567”, he/she should fill the K1-K8 as “01234567”.

1.2.83 DMR_OWN_ID_BCD

This field contains the own ID in 8 BCD codes.

Field Identifier	Own ID in 7 BCD codes
1 Byte	4 Bytes
0x7E	DATA[0]= {K1, K2} DATA[1]= {K3, K4} DATA[2]= {K5, K6} DATA[3]= {K7, K8}
Default	0x00,0x00,0x00,0x00

Table 1-170 DMR_OWN_ID_BCD Field Format

1.2.84 DMR_SLOT_FOUND

This field is used by CT3258 to inform the MCU that synchronization with the far end BS or MS is achieved and CACH is decoded.

DMR_SLOT FOUND	Slot Found Details
1 BYTE	1 Byte
0x7F	Details in the table below

Table 1-171 DMR_SLOT_FOUND Field Format

Bit Position	Bit Name	Bit Descriptions
7	Sync Type	MS Sync or BS Sync

		0: MS Sync 1: BS Sync
6	Slot Verify	Slot verify flag 0: Slot not verified 1: Slot Verified
5	Slot invert	Slot invert flag 0: slot not inverted from last report 1: slot inverted from last report
4	Reserved	Reversed
3	AT	Access Type 0:inbound channel is idle 1:inbound channel is busy note:in continuous transmission mode, for both voice and data,the AT bit is set to 1;
2	TC	TDMA Channel selection 0:Following outbound burst is channel 1 1:Following outbound burst is channel 2
1:0	LCSS/TDMA SLOT	When BS SYNC is found (bit 7 = 1), These two bits indicate LCSS 00:Single fragment LC or first fragment CSBK signalling see note 01:First fragment of LC signalling 10:Last fragment of LC or CSBK signaling 11:Continuation fragment of LC or CSBK Signaling When MS SYNC is found (bit 7 = 0), this two bit indicates the types of MS SYNC found 0: no TDMA direct SYNC 1: TDMA SYNC 1 found 2: TDMA SYNC 2 found

Table 1-172 Slot Found Details

This field is used only by CT3258 to report the CACH information when BS SYNC is found, or types of MS SYNC if MS SYNC is found. Identical CACH are not reported to reduce the traffic between CT3258 and the MCU (after CACH is verified). A CACH verify scheme is used in reporting DMR_SLOT_FOUND. Bit 7-4 of Byte 2 of DMR_CALL_OPTION command specifies the number of good CACH received before reporting DMR_SLOT_FOUND with the slot verified flag set. Before the number of good CACH has reached the specified limit, CT3258 reports every CACH that is found, without setting the slot verified flag. When the specified limit is reached, CT3258 reports the last CACH with the slot verified flag set. If the slot number of the last reported CACH matches the previously reported CACH, it indicates that no slot inversion is needed. If the slot number of the last reported CACH does not match the previously reported CACH, it indicates that slot inversion is needed. In this case, the user should treat all previous report CACH as if the slot number is inverted (1 as 2, and 2 as 1).

2 Application Guides

2.1 Code Downloading

After CT3258 reset, it first down loads a boot loader from the MCU through the HPI and then execute the boot loader. The boot loader then loads the rest of the application from the embedded flash and start execution.

2.1.1 Boot loader Downloading

The boot loader is a program with fixed 1024 words. It is downloaded to CT3258 after CT3258 reset.

After the initial reset, CT3258 enters into boot loading mode. In boot loading mode, CT3258 is automatically configured after reset to load a 1024-word block of instructions from the HPI port. CT3258 then begins executing that instruction block immediately.

The HPI port is configured as 16-bit mode, low-byte first, and either Intel mode or Motorola mode. PIO1 is sampled and used to select either Intel (LOW input) or Motorola (HIGH input).

The host must download exact 1024 words for this boot process to work.

Note that the boot loader for DPMR mode and DMR mode is different. Depending on the boot loader types, CT3258 enters DPMR mode or DMR mode. In DPMR mode, commands for DMR only (type 5) are not recognized. In DMR mode, command for DPMR only (type 3) are not recognized.

2.1.2 Application Downloading

After the boot loader is downloaded to CT3258 and executed, CT3258 reads the application firmware in the embedded flash and start execution.

2.2 DPMR Call Processing

CT3258 support two ways of processing digital calls: Easy Mode (layer 3), DPMR Layer 2 Mode (layer 2).

2.2.1 Easy Mode

The Easy Mode is for users who don't have the in-depth understanding the DPMR protocol. MCU uses simple CALL_START, CALL_STOP, CONNECT, DISCONNECT messages to start, stop a voice calls, or initiate or disconnect a data calls. Internally, CT3258 handles all call processing according the DPMR Protocol .

With Easy Mode, in addition to layer two processing, the CT3258 also performs call control of the DPMR processing. "ANNEX A" Standard User Interface for CSF radio are implemented in CT3258. At the transmitter, CT3258 supports wild character dialing, abbreviated dialing and masked dialing. At the receiver, when a call is received, CT3258 matches incoming called number with its own individual call number and group number and informs the MCU of matching results.

With Easy Mode, CT3258 strictly complies with the DPMR protocol (ETSI TS 102 490 or 658). The ability to change any aspect of the protocol is disabled.

The process mode can be set or changed with PROCESS_MODE command. For Easy Mode, Mode 3 is selected with PROCESS_MODE command.

2.2.1.1 Transmitting

In Easy Mode, the MCU passes called ID and own ID to CT3258. CT3258 saves this information in the internal registers. Upon receiving a CALL_START packet, it formatted the information according to the DPMR protocol and send to the far end.

For example, a voice can be started with the following packet.

Packet Header	START_BYTES
	LENGTH
	TYPE
Fields	DPMR_CALLED_ID
	DPMR_OWN_ID
	CALL_START
Parity	FBYTE
	PARITY_BYTE

Table 2-1 Example Packet For Starting a Voice Call in Easy Mode

If a field has been sent from the MCU to CT3258 before, and does not change in the next packet, this field can be omitted from the packet. For example, to redial a previous dial number, the MCU only needs to send the following packet to CT3258

Packet Header	START_BYTES
	LENGTH
	TYPE
Fields	CALL_START
Parity	FBYTE
	PARITY_BYTE

Table 2-2 Example of Redial Packet

The call can be ended with a packet with CALL_STOP field.

Packet Header	START_BYTES
	LENGTH
	TYPE
Fields	CALL_STOP
Parity	FBYTE
	PARITY_BYTE

Table 2-3 Example of Call Stop Packet

2.2.1.2 Receiving

Upon receiving a call from the far end, CT3258 extracts useful information according to the DPMR protocol and reports to the MCU.

The example below shows the packet sent to the MCU after receiving a new header frame.

Packet Header	START_BYTES
	LENGTH
	TYPE
Fields	CALL_HT_CI
	DPMR_CALLED_ID
	DPMR_OWN_ID
	DPMR_M_V_F_E
	DPMR_CC
	CALL_MATCH
Parity	FBYTE
	PARITY_BYTE

Table 2-4 Example of Packet from DSP after Receiving a Call

The type of FIELD to report can be changed with REPORT_FIELD command.

The fields can be sent in one packet or in multiple packets.

2.2.2 DPMR Layer 2 Mode

With DPMR Layer 2 Mode, CT3258 support DPMR up to layer 2 the does not handle the call processing layer of the DPMR protocol. All call processing supports are provided by external MCU. In this case, CT3258 just performs the tasks of transporting DPMR related messages between the MCU's of the two terminals. As a result, the following process will be performed by the MCU:

Air Interface Call Control Layer (Layer 3)

1. Establishing, maintaining and termination of calls
2. Individual or group call transmission and receptions
3. Destination addressing
4. Automatic matching of Called ID of incoming call to own ID and group ID

As comparison, these processes are completed by CT3258 with Easy Mode.

With DPMR Layer 2 Mode, the MCU can access primitive elements of the DPMR architecture. For example, to start a call, the MCU send individual elements in the protocol to CT3258, with DPMR_SEND_HEADER and DPMR_SEND_SF, whereas in easy mode, the user sends a simple CALL_START to CT3258.

With easy mode, the user also has the ability to change synchronization pattern with DPMR_FS1, DPMR_FS2, DPMR_FS3 and DPMR_FS4.

2.2.2.1 Transmitting

In DPMR Layer 2 Mode, the MCU pass un-encoded HT, END and CCH frames, or individual information elements such as Called_ID or Own_ID to CT3258. CT3258 saves this information in the internal registers. Upon receiving a DPMR Packet with DPMR_SEND_HEADER, DPMR_SEND_END or DPMR_SEND_SF, the MCU performs channel coding and interleaving, forms the complete HI, END and Super Frame structure, and send to the far end.

For example, upon receiving the DPMR Packet below, CT3258 builds a CCH0 to CCH3 structure.

Packet Header	START_BYTES
	LENGTH
	TYPE
Fields	DPMR_CCH_UE (0)
	DPMR_CCH_UE (1)
	DPMR_CCH_UE (2)
	DPMR_CCH_UE (3)
Parity	FBYTE
	PARITY_BYTE

Table 2-5 Example 1 of a Packet from MCU in DPMR Layer 2 Mode

Then if the MCU send a packet with DPMR_SEND_SF, it forms a super frame with the saved information, and sends it to the far end.

Similarly the host message to build a HI frame and END frame can be like the one below:

Packet Header	START_BYTES
	LENGTH
	TYPE
Fields	DPMR_HI_UE
Parity	FBYTE
	PARITY_BYTE

Table 2-6 Example of a HI Packet from MCU

Packet Header	START_BYTES
	LENGTH
	TYPE
Fields	DPMR_END_UE
Parity	FBYTE
	PARITY_BYTE

Table 2-7 Example of END Packet from MCU

2.2.2.2 Receiving

Upon receiving a Super Frame, Header Frame or an End Frame, CT3258 extracts the CCH0 – CCH3, HI or END fields, performs de-interleaving and channel decoding on the frames. CT3258 extracts individual elements to the MCU.

The example below shows the packet sent to the MCU after receiving a new header frame.

Packet Header	START_BYTES
	LENGTH
	TYPE
Fields	DPMR_CALLED_ID
	DPMR_OWN_ID
	DPMR_M_F
	DPMR_HT_CI_PM
Parity	FBYTE
	PARITY_BYTE

Table 2-8 Example of Packet from DSP after Receiving a Header Frame

2.3 DMR Call Processing

CT3258 support two ways of processing digital calls: Easy Mode (Layer 3 Mode), DMR Layer 2 Mode.

In Easy Mode (Layer 3 Mode), CT3258 digital mode support layer 1-3 processing of the DMR protocol complied with ETSI TS 102 361. The MCU uses simple DMR_CALL_START, CALL_STOP messages to start, stop a voice calls.

In Layer 2 Mode, CT3258 implements Layer 1 and Layer 2 of the DMR protocol, complied with ETSI TS 102 361. Layer 3 and above are implemented on the MCU. The interface between CT3258 and the MCU are based on Layer 2 and Layer 3 interface.

Note that if SCT3700 supported is enabled, only Layer 2 Mode is supported.

2.3.1 DMR Easy Mode

In Easy Mode (Layer 3 Mode), the MCU uses simple DMR_CALL_START, CALL_STOP messages to start, stop a voice calls. Internally, CT3258 handles all call processing according the DMR Protocol. “ANNEX C of TS 102 361” numbering plan is implemented in CT3258. At the transmitter, CT3258 supports individual call and group call dialing. At the receiver, when a call is received, CT3258 matches incoming called number with its own individual call number and group number and informs the MCU of matching results.

2.3.2 DMR Layer 2 Mode

In DMR Layer 2 Mode, CT3258 interacts with the control processor (the MCU) to complete the DMR voice and data calls. The layer 1 and layer 2 of the DMR are implemented in CT3258, while layer 3 and above are implemented in the MCU. The information exchange between the MCU and CT3258 is through DIGC_DATA_FRAME commands or indications. The MCU passes the data

types and the payload information the CT3258 through DIGC_DATA_FRAME to CT3258. CT3258 performs FEC according to the data types, CT3258 adds FEC, formulates them into 30 ms data burst and adds 4FSK modulation before sending to the far end. At the receiver, CT3258 decode the received data burst, and report to the MCU through DIGC_DATA_FRAME indications.

In DMR Layer 2 mode, care must be taken to make sure that the DIGC_DATA_FRAME messages are sent no more and no less than 60 millisecond at a time. Examples of layer 2-3 message exchange are described in the example session below.

Two types of commands are used by the MCU to control CT3258. The first type is hardware related for configuring CT3258 in certain mode ready for communication. The second type is protocol related, where commands are used to pass layer 2 and layer 3 information. In particular DIGC_DATA_FRAME command is used to start or stop a voice or data call.

2.3.3 DMR MS Call Flow

The diagram below shows at typical DMR voice call from one mobile station to another mobile station without a repeater.

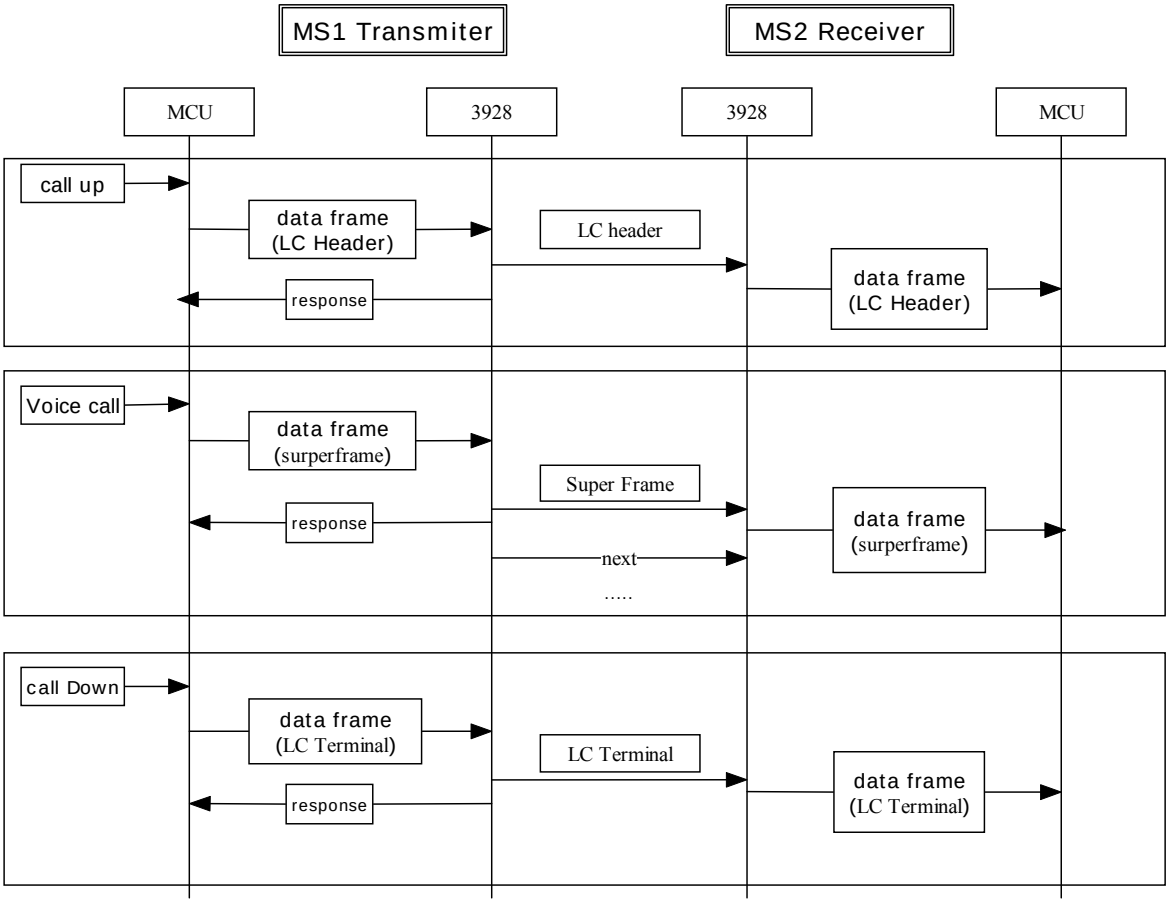

Figure 2-1 DMR MS Call Flow

The MCU informs CT3258 the types of data frame to send through command DMR_DATA_FRAME and then waits for response from CT3258. CT3258 put the data frame for transmission and sends a response in the form of (0x17 0x0a). The next DMR_DATA_FRAME won't be sent unless a response (0x17 0x0A) is received. The interval between two DMR_DATA_FRMAE is around 60 milliseconds.

Only one DMR_DATA_FRAME command with LC contents is needed to start the voice communication at the TX side. The subsequent super frames use the same LC contents. Similarly, DMR_DATA_FRAME is only reported once upon receiving super frames at the RX side. The MCU knows the call end condition from either by receiving a terminal frame, or by receiving CT3258 response of 1B 08.

2.3.4 Base Station Activation

In order to place voice or data calls through the base station or repeater, the mobile station should first acquire synchronization by monitor signals from the base station. If the base station is in idle state, the mobile station should first send a base station activation message to activate the base station. This is done by MCU sending a DMR_DATE_FRAME with CSBK contents. The diagram below shows the base station activation procedure.

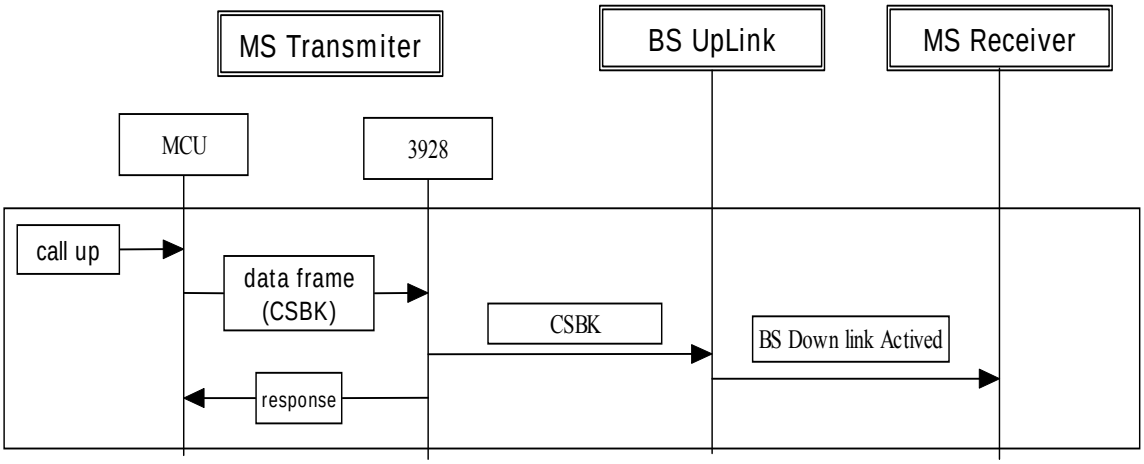

Figure 2-2 Base Station Activation

2.3.5 Listen Before Transmitting

If the base station is activated, it constantly broadcasts base station signals. The mobile station should first acquire timing synchronization from the base station before it can start or receive a call. Timing synchronization is done by monitoring the SYNC pattern and the CACH signals from the base station.

The MCU on the mobile station first sends command DMR_CALL_SLOT to inform CT3258 which slot to monitor, and waits CT3258 for DMR_SLOT_FOUND response. As soon as CT3258 acquire synchronization, it sends DMR_SLOT_FOUND response, first with slot verified flag unset, then after the preset number of CACH is received, with the slot verified flag set. At this point, the mobile station is aligned with the base station. CT3258 can report any data frames from the far end through DMR_DATA_FRAME message. The MCU can also issue DMR_DATA_FRAME command to CT3258 to send to the far end.

The diagram below shows a timing acquisition process.

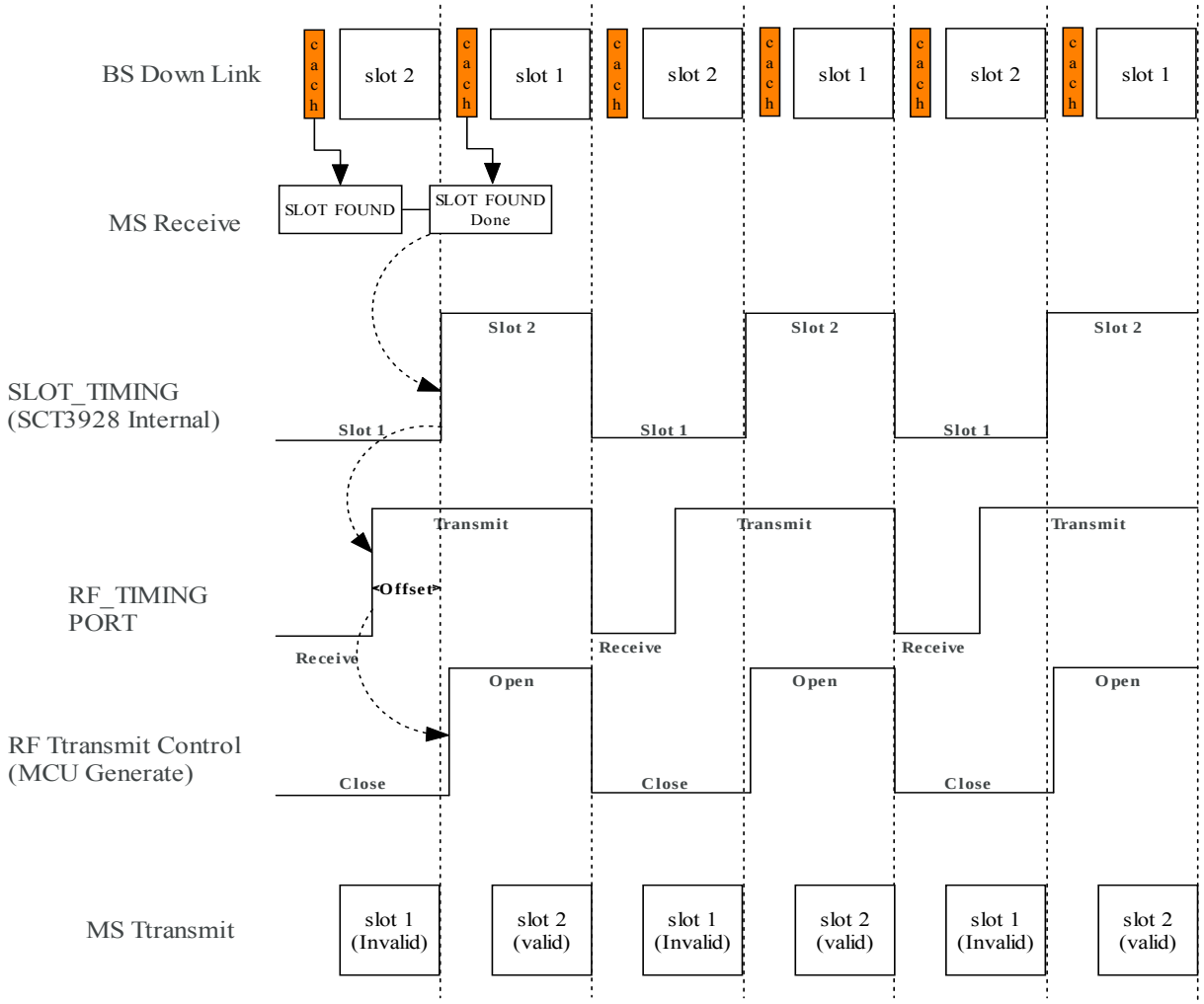

Figure 2-3 Signal Timing Relationship

After CT3258 acquires synchronization from the base station, it generates an internal 60 millisecond cyclic timing signal, SLOT_TIMING. In the TX mode, CT3258 controls the timing on the modulation output port so that it is in synchronization with the base station timing. In the meantime, it also outputs a SLOT_TIMING signal through the RF_TIMING port to control the RF circuit. The RF_TIMING signal can be identical to the SLOT_TIMNG. It can also be ahead or lag the SLOT_TIMING signal. The offset are programmable through command DMR_OFFSET. The MCU can then generate necessary timing signals to control the RF transceiver and the power amplifier. In the case of SCT3700, the switching of RF transceiver and controlled by the CT3258.

2.4 Typical Call Sessions with CT3258 in DPMR Mode

2.4.1 Digital Voice Call

A voice call involves the transmitter and the receiver. In both side, the MCU is the main controller, controlling the RF circuit, the MMI and CT3258. Before the call, the MCU should set the RF circuit in TX mode for the transmitter and in RX mode for the receiver. The sub-section below only describes the interaction between the MCU and CT3258.

2.4.1.1 Initial Setting for both the Transmitter and the Receiver

1. Set Vocoder type with VOCODER_SEL
2. Set CT3258 to Easy mode with PROCESS_MODE command
3. Set Color Code corresponding to the RF Channel with DPMR_CC command
4. Set addressing mode to either “Initial Address Mode” or “Configured Addressing Mode” with ADDRESS_MODE command
5. Set Own ID with DIGC_OWN_ID_BIN or DPMR_OWN_ID_BCD (“Configured Addressing Mode” only)

2.4.1.2 Initial Setting for the Receiver

1. Set the demodulator gain with DEMOD_GAIN command
2. Set the types of field to report when receiving a call with REPORT_FIELD command

2.4.1.3 Initial Setting for the Transmitter

1. Set the modulator gain with MOD_GAIN command

2.4.1.4 Transmitter Side, Start a Call

1. Set CT3258 in TX or DUPLEX with WORK_MODE command
2. Set called ID and own ID with DPMR_CALLED_ID and DPMR_OWN_ID
3. Start the call with CALL_START, for peer-to-peer voice call

2.4.1.5 Receiver Side, Start a Call

1. Set CT3258 in RX or DUPLEX with WORK_MODE command
2. When detecting carrier, inform CT3258 that carrier is ready with CARRIER_READY command

After the call from transmitter arrives, CT3258 reports useful information. The MCU decides

whether to accept or discard the call.

2.4.1.6 Transmitter Side, End a Call

1. Drop the call with CALL_STOP command
2. When the END message is sent out, CT3258 responds with ACK_MESSAGE (0x17 0x0A)

2.4.1.7 Receiver Side, End a Call

1. Drop the carrier with CARRIER_READY command

2.4.2 Voice Call with Slow Data

1. Start the call the same way as voice only call described in the previous section except that the communication mode(the M in M_V_F_EP) is set to Call with Slow Data
2. After the calls start, the MCU first send a packet with DPMR_SLD field.
3. When CT3258 begin transmitting, it fills slow data in the super frame and transmits slow data along with the voice frames. As soon as the slow data is being transmitted, it sends a DPMR_SLD response packet to the MCU, at 80 millisecond interval.

Packet Header	START_BYTES
	LENGTH
	TYPE (Read Near End)
Fields	DPMR_SLD
Parity	FBYTE
	PARITY_BYTE

Table 2-9 Example Packet for Request Near End Slow Data

4. Upon receiving DPMR_SLD response, the MCU write DPMR_SLD to CT3258, also at 80 millisecond interval.
5. At the receiver, upon receiving super frame with SLOW data, CT3258 extract slow data and report to the MCU with DPMR_SLD packet at 80 millisecond interval.
6. When all data has been transmitted, the MCU has to wait the response packet of the last frame DPMR_SLD.
7. Drop the call with CALL_STOP command
8. When the END message is sent out, CT3258 responds with ACK_MESSAGE (0x17 0x0A)

2.4.3 Voice Call Recording

1. Enable voice recording function with command VOCODER_IO_SET at the transmitter for near end voice call recording (IO_SET = 0x20), or at the receiver for far end voice call recording (IO_SET = 0x04)

2. Start the call the same way as voice only call described in the previous section.
3. When the call starts, CT3258 start to send encoded voice data with CHAN_D field, at 80 millisecond interval.

Packet Header	START_BYTES
	LENGTH
	TYPE (Read Near End)
Fields	CHAN_D
Parity	FBYTE
	PARITY_BYTE

Table 2-10 Example Packet for Voice Recoding

4. Upon receiving CHAN_D, the MCU should process the message and save its contents. The MCU does not send a response to CHAN_D.
5. The process repeats at 80 millisecond interval.
6. Drop the call with CALL_STOP command
7. When the END message is sent out, CT3258 responds with ACK_MESSAGE (0x17 0x0A)
8. Disable the voice recording by setting IO_SET = 0x00

2.4.4 Voice Call Play Back

1. Enable voice play back function with command VOCODER_IO_SET at the transmitter for far end voice play back (IO_SET = 0x40), or at the receiver for near end voice play back (IO_SET = 0x08)
2. Start the call the same way as voice only call described in the previous section.
3. When the call starts, MCU starts to send encoded voice data with CHAN_D field, at 80 millisecond interval.

Packet Header	START_BYTES
	LENGTH
	TYPE (Read Near End)
Fields	CHAN_D
Parity	FBYTE
	PARITY_BYTE

Table 2-11 Example Packet for Play Back

4. Upon receiving CHAN_D, CT3258 send back the response to CHAN_D.
5. The process repeats at 80 millisecond interval.
6. Drop the call with CALL_STOP command
7. When the END message is sent out, CT3258 responds with ACK_MESSAGE (0x17 0x0A)
8. Disable the voice recording by setting IO_SET = 0x00

2.4.5 Data Call with Type 1 or Type 2 Data

1. Start the call the same way as voice only call described in the previous section except that the communication mode (the M in M_V_F_EP) is set to Type 1 or Type 2 data call. The DPMR_HT_CI has to be configured as section 1.2.53.
2. After the calls start, the MCU first send a packet with DIGC_DATA_FRAME field, which includes up to 36 bytes (in 80 ms) of data. In the DIGC_DATA_FRAME fields, user should also indicate whether valid data are present and the length of the valid data. The user should also indicate whether this is the last data frame.
3. When CT3258 begin transmitting, it fills data in the TCH frame and starts transmitting data. As soon as data is being transmitted, it sends a DIGC_DATA_FRAME response packet to the MCU, at 80 millisecond interval.

Packet Header	START_BYTES
	LENGTH
	TYPE (Read Near End)
Fields	DIGC_DATA_FRAME
Parity	FBYTE
	PARITY_BYTE

Table 2-12 Example Packet for Request Near End Type 1 and Type 2 Data

4. Upon receiving DPMR_DTAT_FRAME response, the MCU write DIGC_DATA_FRAME with more data to CT3258, also at 80 millisecond interval.
5. At the receiver, upon receiving DPMR pay load, CT3258 extracts data from TCH frames and report to the MCU with DIGC_DATA_FRAME packet at 80 millisecond interval.
6. When all data has been transmitted, the MCU responds to data query with DIGC_DATA_FRAME with indication that no more data is to transmit.
7. Drop the call with CALL_STOP command
8. When the END message is sent out, CT3258 responds with ACK_MESSAGE (0x17 0x0A)

2.4.6 Voice Call with Appended Data

The Voice Call with Appended Data starts as voice call and ends as a Type 2 Data Call.

1. Start the call the same way as voice only call described in the previous sections except that the communication mode(the M in M_V_F_EP) is set to Voice Call with Appended Data. The DPMR_HT_CI has to be configured as section 1.2.53.
2. Proceed to voice communication as a normal voice call
3. When the user releases the PTT, the MCU first send a packet with DIGC_DATA_FRAME field, which includes up to 36 bytes (in 80 ms) of data. In the DIGC_DATA_FRAME fields, user should also indicate whether valid data are present and the length of the valid data. The user should also indicate whether this is the last data frame.

4. When CT3258 begin transmitting, it fills data in the TCH frame and starts transmitting data. As soon as data is being transmitted, it sends a DIGC_DATA_FRAME response packet to the MCU, at 80 millisecond interval.
5. Upon receiving DPMR_DTAT_FRAME response the MCU write DIGC_DATA_FRAME with more data to CT3258, also at 80 millisecond interval.
6. At the receiver, upon receiving DPMR pay load, CT3258 extracts data from TCH frames and report to the MCU with DIGC_DATA_FRAME packet at 80 millisecond interval.
7. When all data has been transmitted, the MCU responds to data query with DIGC_DATA_FRAME with indication that no more data is to transmit.
8. Drop the call with CALL_STOP command
9. When the END message is sent out, CT3258 responds with ACK_MESSAGE (0x17 0x0A)

2.4.7 Data Call Type 3 (for future release)

Type 3 Data calls involve a call set up and disconnect procedure as described below:

1. Station A: Send connection request
2. Station B: Send ACK
3. Station A: Send data frames
4. Station B: Send ACK
5. Station A: Send more data frames
6. Station B: Send ACK

…

7. Station A: Send Disconnect request

The interactions of MCU and CT3258 to complete a data call are described below.

2.4.7.1 Initialization of Data Call Type 3

The initialization of Type3 data call is the same as voice calls.

2.4.7.2 Connection Request

1. Set Station A in TX mode and Station B in RX mode with WORK_MODE command
2. Set call information (CI) for appropriate data format with DPMR_CI_PM command
3. Send CONNECT command in station A
4. Set Carrier Ready in station B
5. CT3258 in station B report HT_CI to station B MCU
6. Station B MCU becomes aware of a data call request, and get ready to send an ACK
7. When the CONNECT message is sent out in station A, CT3258 responds with ACK_MESSAGE (10)

2.4.7.3 Connection Acknowledge

1. Set Station B in TX mode and Station A in RX mode with WORK_MODE command
2. Set CI to indicate ACK or NACK (with NACK reason) with DPMR_CI_PM.
3. Send DPMR_ACK command in station B
4. Set Carrier Ready in station A
5. CT3258 in station A report HT_CI to the MCU
6. Station A MCU becomes aware of the acknowledgement from station B, and get ready to send data.
7. When the Connect Acknowledgement message is sent out in station B, CT3258 responds with ACK_MESSAGE (10)

2.4.7.4 Sending Type 3 Data Frames

1. Set Station A in TX mode and Station B in RX mode with WORK_MODE command
2. Set CI for appropriate data format with DPMR_CI_PM command
3. Start the call with CALL_START
4. After the calls start, the MCU first send a packet with DIGC_DATA_FRAME field with Type 3 data indication, which includes up to 48 bytes (in 80 ms) of data. In the DIGC_DATA_FRAME fields, user should also indicate whether valid data are present and the length of the valid data. The user should also indicate whether this is the last data frame.
5. CT3258 fills data in the data frame and start transmission. As soon as data is being transmitted, it sends a DIGC_DATA_FRAME query packet to the MCU to request for more data. This process continues at 80 milli-second interval.
6. Upon receiving DIGC_DATA_FRAME query, the MCU write DIGC_DATA_FRAME to CT3258, also at 80 millisecond interval.
7. At station B, upon receiving data frames, CT3258 extract data and report to the MCU with DIGC_DATA_FRAME packet at 80 millisecond interval.
8. Station B MCU assembles the data frames from the DIGC_DATA_FRAME fields.
9. When all data in current data frame has been transmitted, the MCU responds to data query with DIGC_DATA_FRAME with indication that no more data is to transmit.
10. Drop the call with CALL_STOP command
11. When the END message is sent out, CT3258 responds with ACK_MESSAGE (0x17 0x0A)

2.4.7.5 Data Acknowledgement

1. Set Station B in TX mode and Station A in RX mode with WORK_MODE command
2. Set CI to indicate ACK or NACK (with NACK reason) with DPMR_CI_PM.
3. Send DPMR_ACK command in station B
4. Set Carrier Ready in station A
5. CT3258 in station A report HT_CI to the MCU
6. Station A MCU becomes aware of the acknowledgement from station B, and get ready to send more data packet.
7. When the Data Acknowledgement message is sent out in station B, CT3258 responds with

ACK_MESSAGE (0x17 0x0A) to MCU B.

2.4.7.6 Disconnect

1. Set Station A in TX mode and Station B in RX mode with WORK_MODE command
2. Send DISCONNECT command in station A.
3. Set Carrier Ready in station B
4. CT3258 in station B report HT_CI to the MCU
5. Station B MCU becomes aware of the Disconnect request from station A, and start disconnect procedure
6. Station B disconnect the cal
7. When the DISCONNECT message is sent out in station A, CT3258 responds with ACK_MESSAGE (0x17 0x0A)

2.4.8 Short Appended Data (TS 102 658 Only)

ETSI TS 102 658 defines a special message for transmitting short data. It is constructed by a connection request HEADER frame, one to four appended data frames, and an END frame. To send Short Appended Data, the MCU can follow the procedure below.

2.4.8.1 Send Short Appended Data

Set Station A in TX mode with WORK_MODE command

1. Set call information (CI) for appropriate data format with DPMR_CI_PM command
2. Send the contents of the short appended data with DIGC_DATA_FRAME command with type set to Short Appended Data.
3. Send SEND_AD command in station A
4. When the CONNECT message is sent out in station A, CT3258 responds with ACK_MESSAGE (10)
5. The MCU set Station A in RX mode to get ready to receive an ACK.

2.4.8.2 Receive Short Appended Data

1. Set Station B in RX mode with WORK_MODE command
2. Set Carrier Ready in station B
3. CT3258 in station B receives short appended data and report to station B MCU with DIGC_DATA_FRAME
4. Set Station B in TX mode to get ready to send an ACK.

2.4.9 Analog Voice Call in DPMR Mode

CT3258 support dual mode operation with digital and analog calls. For analog call, CT3258 supports CTCSS/DCS sub-audio signals.

2.4.9.1 Receiver Side, Start a Call

1. Set CT3258 to analog call mode with PROCESS_MODE command
2. Set CT3258 in RX or DUPLEX with WORK_MODE command
3. Set CTCSS/DCS settings with SUB_AUDIO command
4. When detecting carrier, inform CT3258 that carrier is ready with CARRIER_READY

After the call from transmitter arrives, CT3258 reports whether CTCSS/DCS is matched or not with SUB_AUDIO command.

2.4.9.2 Transmitter Side, Start a Call

1. Set CT3258 to analog call mode with PROCESS_MODE command
2. Set CT3258 in TX or DUPLEX with WORK_MODE command
3. Set CTCSS/DCS settings with SUB_AUDIO command
4. Start the call with CALL_START command

2.4.9.3 Receiver Side, End a Call

1. Drop the carrier with CARRIER_READY command

2.4.9.4 Transmitter Side, End a Call

1. Drop the call with CALL_STOP command

2.4.10 Handling of Maintenance Message (TS 102 658 Only)

With ETSI TS 102 658, the BS may send multiple maintenance messages such as preservation messages, Idle messages or Guard messages. CT3258 may report each occurrence of these messages, or elect to only report the same message once to save traffic between CT3258 and the MCU. The election is controlled by command DPMR_PROTOCOL_OPTION.

2.4.11 Automatic Analog and DPMR Call Detection

To set up for automatic analog and DPMR call detection, the receiver is set up the same ways as a digital calls, with the following exceptions:

1. Set CT3258 to Easy mode with PROCESS_MODE command, with mixed call bit enabled: i. e., the process mode as 131.
2. Set CTCSS/DCS settings with SUB_AUDIO command

The transmitter side is set up exactly as normal analog or DPMR call.

With automatic analog and DPMR call detection, CT3258 monitors the RF channel, and simultaneously detects FS1/FS2 pattern for DPMR calls, and CTCSS/DCS signaling for analog calls. If DPMR call is found, it sends CALL_MATCH message (even if the called ID or CC code does not match). If CTCSSS/DCS signaling is found, it send SUB_AUDIO message.

2.4.12 Audio Muting for Un-matched Calls

By defaults, CT3258 enables audio as soon as a DPMR or an analog call is received, even if called ID or CTCSS/DCS code does not match. It is up to the MCU to decide whether to take the call or not. If the MCU decides to take to call, it enables audio PA. If not, it turns off the audio PA, and send carrier lost to CT3258.

Optionally, the CT3258 can mute the audio until a correct call match is achieved. This can be done by setting Mute Flag when sending the Carrier Ready command at the receiver. When Mute Flag is set, CT3258 sends zeros to its audio out line until a called ID of the incoming call matches the own ID of itself, or the CTCSS/DCS code of the transmitter and receiver match.

2.5 Typical Call Sessions with CT3258 in DMR Mode

2.5.1 DMR Voice Call in DMR Easy Mode

A voice call involves the transmitter and the receiver. In both side, the MCU is the main controller, controlling the RF circuit, the MMI and CT3258. Before the call, the MCU should set the RF circuit in TX mode for the transmitter, and in RX mode for the receiver. The sections below only describe the interaction between the MCU and CT3258.

Note that if SCT3700 support is enabled, DMR Easy mode is not supported.

2.5.1.1 Initial Setting for both the Transmitter and the Receiver

1.Set Vocoder type with VOCODER_SEL, 4 for AMBE+2
Command ID: VOCODER_SEL
Send Command: 84 A9 61 00 02 00 10 04
Receive Data:84 A9 61 00 02 00 10 00

2.Set CT3258 to digital mode with PROCESS_MODE command
Command ID: PROCESS_MODE
Send Command: 84 A9 61 00 02 00 1A 03
Receive Data:84 A9 61 00 02 00 1A 00

3.Set Color Code corresponding to the RF Channel with DMR_CC command
Send Command: 84 A9 61 00 02 08 77 01

Receive Data:84 A9 61 00 02 08 77 00

4.Configure the call option with DMR_CALL_OPTION
Command ID: DMR_CALL_OPTION
Send Command: 84 A9 61 00 06 08 79 41 00 00 00 00
Receive Data:84 A9 61 00 02 08 79 00
(In this particular command, CT3258 is set to do SQ detection; slotted mode is used, with unused slot filled with zeros; SLOT verified number is 1).

5.Set Own ID with DIGC_OWN_ID_BIN or DMR_OWN_ID_BCD
Command ID: DIGC_OWN_ID_BIN
Send Command: 84 A9 61 00 04 08 50 XX XX XX
Receive Data:84 A9 61 00 02 08 50 00

2.5.1.2 Initial Setting for the Receiver

1. Set the demodulator gain with DEMOD_GAIN command
2. Set the types of field to report when receiving a call with REPORT_FIELD command
Command ID: REPORT_FIELD
Send command: 84 A9 61 00 02 00 1D XX
Receive Data: 84 A9 61 00 02 00 1D 00

2.5.1.3 Initial Setting for the Transmitter

1. Set the modulator gain with MOD_GAIN command

2.5.1.4 Transmitter Side, Start a Call

1.Set CT3258 in TX with WORK_MODE command
Command ID: WORK_MODE
Send Command: 84 A9 61 00 04 00 18 02 00 00
Receive Data: 84 A9 61 00 02 00 18 00

2.Set called ID and own ID with DIGC_CALLED_ID and DIGC_OWN_ID
Command ID: DIGC_OWN_ID_BIN
Send Command: 84 A9 61 00 04 08 50 XX XX XX
Receive Data: 84 A9 61 00 02 08 50 00

Command ID: DMR_CALLED_ID_BIN
Send Command: 84 A9 61 00 04 08 51 XX XX XX
Receive Data: 84 A9 61 00 02 08 51 00

3.Set FLC information

Command ID: DMR_FLC
Send Command: 84 A9 61 00 04 08 76 00 00 00
Receive Data:84 A9 61 00 02 08 76 00

(In this particular command, FLCO as group call, FID is standard)

4.Start the call with DMR_CALL_START, for peer-to-peer voice call
Command ID: DMR_CALL_START
Send Command: 84 A9 61 00 01 08 78
receive command: 84 a9 61 00 02 08 78 00

2.5.1.5 Receiver Side, Start a Call

1.Set CT3258 in RX or DUPLEX with WORK_MODE command
Command ID: WORK_MODE
Send Command: 84 A9 61 00 04 00 18 01 00 00
Receive Data:84 A9 61 00 02 00 18 00

2.When detecting carrier, inform CT3258 that carrier is ready with CARRIER_READY command

Send Command: 84 A9 61 00 02 00 19 {0x01 or 0x02}
Receive Data:84 A9 61 00 02 00 19 00

After the call from transmitter arrives, CT3258 reports useful information. The MCU decides whether to accept or discard the call.

2.5.1.6 Receiver Side, after a Call is Received

Receive Data:84 A9 61 00 02 30 7F 87

- DMR_SLOT_FOUND message. The content is TACT in the CACH, BS call, SLOT verify not completed

Receive Data:84 A9 61 00 02 30 1B 04

- Indicate voice SYNC is received

Receive Data:84 A9 61 00 02 30 7B 11

- DMR_SLOT_TYPE message. CC is 1. Data type is Voice LC Header

Receive Data:84 A9 61 00 04 30 76 00 00 00

- DMR_FLC message

Receive Data:84 A9 61 00 04 30 50 xx xx xx

- DIGC_CALLED_ID_BIN

Receive Data:84 A9 61 00 04 30 51 xx xx xx

- DIGC_OWN_ID_BIN

Receive Data:84 A9 61 00 05 30 7D xx xx xx xx

- DIGC_CALLED_ID_BCD

Receive Data:84 A9 61 00 05 30 7E xx xx xx xx

- DIGC_OWN_ID_BCD

Receive Data:84 A9 61 00 02 30 27 00

- CALL_MATCH message, indicating a valid DMR call is received, and called ID and CC are matched.

2.5.1.7 Transmitter Side, End a Call

1. Drop the call with CALL_STOP command
   Command ID: DMR_CALL_STOP
   Send Command: 84 A9 61 00 01 03 21
   receive command: 84 A9 61 00 02 03 21 00

2. When the END message is sent out, CT3258 responds with ACK_MESSAGE (0x17 0x0A)

2.5.1.8 Receiver Side, End a Call

When the transmitter stops the call, CT3258 receives LC terminator from the far end, and report it with DMR_SLOT_TYPE to the MCU.

Receive Data:84 A9 61 00 02 30 7B 11

- DMR_SLOT_TYPE message. CC is 1. Data type is Voice LC Terminator.

2. After receiving DMR_SLOT_TYPE, the receiver can then drop the carrier with CARRIER_READY command
   Command ID: DMR_CARRIER_READY
   Send Command: 84 A9 61 00 02 00 19 00
   Receive Data:84 A9 61 00 02 00 19 00

2.5.2 DMR Voice Call in DMR Layer 2 Mode

A voice call involves the transmitter and the receiver. In both side, the MCU is the main controller, controlling the RF circuit, the MMI and CT3258. Before the call, the MCU should set the RF circuit in TX mode for the transmitter and in RX mode for the receiver. The sub-section below only describes the interaction between the MCU and CT3258.

2.5.2.1 Initial Setting for both the Transmitter and the Receiver

1. Set Vocoder type with VOCODER_SEL. An up to 2-3 seconds delay is required to down load the vocoder from the flash.
   Command ID: VOCODER_SEL
   Send Command: 84 A9 61 00 02 00 10 04
   Receive Data:84 A9 61 00 02 00 10 00

2. Set CT3258 to layer 2 mode with PROCESS_MODE command
   Command ID: PROCESS_MODE
   Send Command: 84 A9 61 00 02 00 1A 02
   Receive Data:84 A9 61 00 02 00 1A 00

3. Set Color Code corresponding to the RF Channel with DMR_CC command
   Command ID: DMR_CC
   Send Command: 84 A9 61 00 02 08 77 01
   Receive Data:84 A9 61 00 02 08 77 00

4. Set DMR call options with DMR_CALL_OPTION command
   Command ID: DMR_CALL_OPTION
   Send Command: 84 A9 61 00 06 08 79 41 00 10 00 00
   Receive Data:84 A9 61 00 02 08 79 00

(In this particular command, CT3258 is set to do SQ detection; slotted mode, with unused slot filled with zeros; SLOT verified number is 1).

2.5.2.2 Initial Setting for the Receiver

- Set the demodulator gain with DEMOD_GAIN command
- Set the types of field to report when receiving a call with REPORT_FIELD command

2.5.2.3 Initial Setting for the Transmitter

1. Set the modulator gain with MOD_GAIN command

2.5.2.4 Receiver Side, Start a Call

1. Set CT3258 in RX or DUPLEX with WORK_MODE command
   Command ID: WORK_MODE_RX
   Send Command: 84 A9 61 00 04 00 18 01 00 00
   Receive Data:84 A9 61 00 02 00 18 00
2. Set which slot to receive with DMR_CALL_SLOT command
   Send Command: 84 A9 61 00 02 08 6F 01

Receive Data:84 A9 61 00 02 08 6F 00
(For mobile to mobile call, the slot number is 0)

3. When detecting carrier, inform CT3258 that carrier is ready with CARRIER_READY command
   Command ID: DMR_CARRIER_READY
   Send Command: 84 A9 61 00 02 00 19 {0x01 or 0x02}
   Receive Data:84 A9 61 00 02 00 19 00

After the call from transmitter arrives, CT3258 reports useful information. The MCU decides whether to accept or discard the call.

2.5.2.5 Transmitter Side, Start a Call

1. Set which slot to transmit with DMR_CALL_SLOT command. If MS call is to be placed, the slot number is 0. Otherwise, slot 1 or slot 2 is selected
   Command ID: DMR_CALL_SLOT
   Send Command: 84 A9 61 00 02 08 6F 81
   Receive Data: 84 A9 61 00 02 08 6F 00

2. Set CT3258 in TX mode with WORK_MODE command
   Command ID: WORK_MODE_TX
   Send Command: 84 A9 61 00 04 00 18 02 00 00
   Receive Data: 84 A9 61 00 02 00 18 00

3. Send LC with DIGC_DATA_FRAME (CC can be sent in the same packet)
   Command ID: DIGC_DATA_FRAME
   Send Command: 84 A9 61 00 0E 08 77 01 43 01 09 00 00 00 00 00 01 00 00 01
   Receive Command: 84 A9 61 00 01 20 43

4. Start the call by sending super frame with DIGC_DATA_FRAME
   Command ID: PROTOCOL_DATA_FRAME
   Send Command: 84 A9 61 00 0E 08 77 01 43 11 09 00 00 00 00 00 01 00 00 01
   Receive Command: 84 A9 61 00 01 20 43

2.5.2.6 Receiver Side, after a Call is Received

Receive Data: 84 A9 61 00 02 30 7F 87

- DMR_SLOT_FOUND message. The content is TACT in the CACH, BS call, SLOT verify not completed

Receive Data:84 A9 61 00 02 30 7F A9

- DMR_SLOT_FOUND message. The content is TACT in the CACH, BS call, SLOT verify completed, with no need to invert slot numbers in previous DMR_SLOT_FOUND

message. At this point, the mobile station is synchronized with the base station. If the mobile station wishes to start a call, it needs to check the AT bit in the TACT of CACH, and proceed if it is not busy.

Receive Data: 84 A9 61 00 0E 30 77 01 43 01 09 00 00 00 00 00 01 00 00 01

- This message contains two fields, DMR_CC and DIGC_DATA_FRAME. CC is 0x01. The content in the data frame is data burst at SLOT1, length of 9, and Voice LC header.

Receive Data: 84 A9 61 00 0E 30 77 01 43 11 09 00 00 00 00 00 01 00 00 01

- This message contains two fields, DMR_CC and DIGC_DATA_FRAME. CC is 0x01. The content in the data frame is voice super frame at SLOT1, length of 9, and Voice LC header. Once CT3258 receives and reports one voice super frame, it won't report subsequent voice super frame. But it continues to track the synchronization of the base station, and report 0x1B 0x08 if the synchronization is lost, which the MCU can use as an indication of call lost.

2.5.2.7 Transmitter Side, End a Call

1. Stop the call by sending Terminator LC. Once CT3258 has sent all message to the far end, it reports 17 0A to the MCU

   Command ID: PROTOCOL_DATA_FRAME

   Send Command: 84 A9 61 00 0c 08 43 02 09 00 00 00 00 00 01 00 00 01

   Receive Command: 84 A9 61 00 01 20 43

   Receive Command: 84 A9 61 00 02 20 17 0A

2.5.2.8 Receiver Side, End a Call

When the transmitter stops the call, CT3258 receives LC terminator from the far end, and report it with DIGC_DATA_FRAME to the MCU

Receive Data:84 A9 61 00 0E 30 77 01 43 02 09 00 00 00 00 00 01 00 00 01

- This message contains two fields, DMR_CC and DIGC_DATA_FRAME. CC is 0x01. The content in the data frame is data burst at SLOT1, length of 9, and Voice LC Terminator.

1. Drop the carrier with CARRIER_READY command

   Command ID: DMR_CARRIER_READY

   Send Command: 84 A9 61 00 02 00 19 00

   Receive Data:84 A9 61 00 02 00 19 00

2.5.3 Analog Voice Call in DMR Mode

CT3258 support dual mode operation with digital and analog calls. For analog call, CT3258 supports CTCSS/DCS sub-audio signals.

2.5.3.1 Receiver Side, Start a Call

1. Set CT3258 to analog call mode with PROCESS_MODE command
2. Set CT3258 in RX or DUPLEX with WORK_MODE command
3. Set CTCSS/DCS settings with SUB_AUDIO command
4. When detecting carrier, inform CT3258 that carrier is ready with CARRIER_READY

After the call from transmitter arrives, CT3258 reports whether CTCSS/DCS is matched or not with SUB_AUDIO command.

2.5.3.2 Transmitter Side, Start a Call

1. Set CT3258 to analog call mode with PROCESS_MODE command
2. Set CT3258 in TX or DUPLEX with WORK_MODE command
3. Set CTCSS/DCS settings with SUB_AUDIO command
4. Start the call with DMR_CALL_START command

2.5.3.3 Receiver Side, End a Call

1. Drop the carrier with CARRIER_READY command

2.5.3.4 Transmitter Side, End a Call

1. Drop the call with CALL_STOP command

2.6 Gain Calibration

This is the first step of calibration. It calibrates the receiver path to calculate SQ_LEVEL correctly without being affected by gains variation in the analog path..

For SCT3700, follow these steps:

1. Feed -60dbm FM signal (eg. by HP8920) to receiver and make chip run in receiver mode
2. Adjust “GAIN_ADJUST” till WB_RSSI equals to -60dbm

Note that if external LNA is used, it needs to be disable when doing GAIN calibration.

Command ID: READ_RSSI

Send Command: 84 A9 61 00 01 20 6C 00

Receive Data: 84 A9 61 00 07 23 6C 00 3B 00 37 00 00

For example, 0x3B-137 = -78dbm, which means there is -18db mismatch. So we set GAIN_ADJUST to 0xee, and send

Command ID: WRITE_RSSI

Send Command: 84 A9 61 00 0D 00 6C EE 00 00 30 10 00 04 00 10 00 04 00

Receive Data:84 A9 61 00 02 00 6C 00

After that when we read again, it results,

Command ID: READ_RSSI

Send Command: 84 A9 61 00 01 20 6C 00

Receive Data: 84 A9 61 00 07 23 6C 00 4C 00 36 00 00

This time 0x4c-137 = -61dbm is almost correct, +/-1db is ignorable.

3. Enable external LNA if it is used

Command ID: 3700CONFIG

Send Command: 84 A9 61 00 03 00 6D 0F 00 00

Receive Data:84 A9 61 00 02 00 6D 00

4. Adjust "EXT LNA_GAIN" till WB_RSSI equals to -60dbm

Command ID: readrssi

Send Command: 84 A9 61 00 01 20 6C 00

Receive Data: 84 A9 61 00 07 23 6C 00 6D 00 18 00 00

0x6D-137 = -28dbm, which means 32db mismatch. So we set EXTLNA_VAL[5:0]=0x20, EXTLNA_VAL[7:6]=3 which is determined by 0x20-(0x4C-0x3B)=15dB

Bit7-Bit6	Value
11	12<value
10	6< value≤12
01	0<value≤6

Command ID: WRITE_RSSI

Send Command: 84 A9 61 00 0D 00 6C EE E0 00 30 10 00 04 00 10 00 04 00

Receive Data:84 A9 61 00 02 00 6C 00

Then read WB_RSSI again, where 0x4D-137 = -60dbm.

Command ID: READ_RSSI

Send Command: 84 A9 61 00 01 20 6C 00

Receive Data: 84 A9 61 00 07 23 6C 00 4D 00 18 00 00

Up till now, RX gain calibration is done.

2.7 DC and IQ Calibration

DC and IQ calibration of SCT3700 can be performed after gain calibration is done. Make sure that "GAIN ADJUST" and "EXT LNA GAIN" are set correctly before proceeding to this section.

Before starting, the user should be familiar with these 3 commands.

Command ID: DC_OFFSET_READ
Send Command: 84 A9 61 00 02 20 39 05
Receive Command:84 A9 61 00 0f 20 39 “RXDCI” “RXDCQ” “AMPI” “AMPQ” “TXDC_high” ”TXDC_low” “AMPCOMP”

Command ID: DC_OFFSET_WRITE
Send Command: 84 A9 61 00 0d 00 39 “ADC_DCI” “ADC_DCQ” “DAC_DCI” “DAC_DCQ” “DAC_GAINI” “DAC_GAINQ”
Receive Command:84 A9 61 00 00 02 39 00

Command ID: IQCOMP_COEF_WRITE
Send Command: 84 A9 61 00 0A 00 42 “cali_setting” “rx_coef_amp” “rx_coef_phi” “tx_coef_amp” “tx_coef_pha”
Receive Command:84 A9 61 00 00 02 42 00

1. Boot CT3258. Don’t select any vocoder.
   Put SCT3700 into calibration loop back mode with command
   Command ID: WORK_MODE_CALIBRATOIN
   Send Command: 84 A9 61 00 04 00 18 43 00 00
   Receive Command:84 A9 61 00 02 00 18 00

2. Set the SCT3700 RF to transmit a carrier at frequency Fc+Fif (Fif = 1050 kHz for example).

3. Set the SCT3700 RF to receiver at frequency Fc. After the down conversion by the RF front end, IQ signals of frequency Fif are at the input of CT3258.

4. Disable IQ calibration filter.
   Command ID: IQCOMP_COEF_WRITE
   Send Command: 84 A9 61 00 0A 00 42 **44** 00 00 00 00 00 00 00 00
   Receive Command:84 A9 61 00 02 00 42 00

5. Disable IF dc filter
   Command ID: MISC_GAIN
   Send Command: 84 A9 61 00 0D 00 6B 08 00 08 00 08 00 00 00 01 00 00 00
   Receive Command:84 A9 61 00 02 00 6B 00
   Clear all the dc offset
   Command ID: DC_OFFSET_WRITE
   Send Command: 84 A9 61 00 0d 00 39 00 00 00 00 00 00 00 00 00 00 00 00
   Receive Command:84 A9 61 00 00 02 39 00

6. Force PGA gain to “0”

Command ID: RSSI_WRITE

Send Command: 84 A9 61 00 0d 00 6C “GAIN ADJUST” “EXT LNA GAIN” 80 00 10 00 04 00 10 00 04 00

Receive Data:84 A9 61 00 02 00 6C 00

Note that “GAIN ADJUST” and “EXT LNA GAIN” should be set to the values obtained from last section.

7. Read the DC offset “RXDCI” and “RXDCQ” with command DC_OFFSET read command (100 milliseconds average in this example)

Command ID: DC_OFFSET_READ

Send Command: 84 A9 61 00 02 20 39 05

Receive Command:84 A9 61 00 0f 20 39 “**RXDCI**” “**RXDCQ**” “AMPI” “AMPQ” “TXDC_high” ”TXDC_low” “AMPCOMP”

8. Write the above DC compensation value “**RXDCI**” and “**RXDCQ**” with DC_OFFSET write command

Command ID: DC_OFFSET_WRITE

Send Command: 84 A9 61 00 0d 00 39 “**ADC_DCI**” “**ADC_DCQ**” “DAC_DCI” “DAC_DCQ” “DAC_GAINI” “DAC_GAINQ”

Receive Command:84 A9 61 00 00 02 39 00

The user can start with ADC_DCI = RXDCI, ADC_DCQ = RXDCQ. DAC_DCI and DAC_DCQ should be zero.

9. Repeat 6 and 7 until minimum RXDCI and RXDCQ is obtained. This completes the RX DC calibration for this PGA level.

10. Put RX IQ compensation filter in calibration mode by issue IQCOMP_COEF command with RX_IQ_MODE bit set to 1 and TX_IQ_MODE bit to 1

Command ID: IQCOMP_COEF_WRITE

Send Command: 84 A9 61 00 0A 00 42 **44** “rx_coef_amp” 00 00 00 00 00 00

Receive Command:84 A9 61 00 00 02 42 00

Start rx_coef_amp = 0;

Then configure rx_coef_amp = “AMPCOMP” from step 15.

11. Next, start doing RX IQ calibration for this PGA level. Please note whenever you send the “39” command, you should make sure that you use the correct “**ADC_DCI**” “**ADC_DCQ**” accordingly.

12. Read the amplitude of the signal at I and Q path with command DC_OFFSET read command (100 milliseconds average in this example). Here we should configure the “DAC_GAINI”

"DAC_GAINQ" to make sure amplitude big enough (**"AMPI"** / "**AMPQ**"is larger than 0x1000 and smaller than 0x7000).

Command ID: DC_OFFSET_READ
Send Command: 84 A9 61 00 02 20 39 05
Receive Command:84 A9 61 00 0f 20 39 "DCI" "DCQ" "AMPI" "AMPQ" "TXDC_high" "TXDC_low" "**AMPCOMP**"

Send Command: 84 A9 61 00 0d 00 39 "**ADC_DCI**" "**ADC_DCQ**" "DAC_DCI" "DAC_DCQ" **"DAC_GAINI" "DAC_GAINQ"**
Receive Command:84 A9 61 00 00 02 39 00

13. Write AMPCOMP until AMPI and AMPQ are almost equal.
Command ID: IQCOMP_COEF_WRITE
Send Command: 84 A9 61 00 0A 00 42 **44** "AMPCOMP" 00 00 00 00 00 00
Receive Command:84 A9 61 00 00 02 42 00

14. Put RX IQ compensation filter in normal mode by issue IQCOMP_COEF command with RX_IQ_MODE bit set to 0
Command ID: IQCOMP_COEF_WRITE
Send Command: 84 A9 61 00 0A 00 42 **00** "rx_coef_amp" "rx_coef_phi" 00 00 00 00
Receive Command:84 A9 61 00 00 02 42 00

Start rx_coef_phi = 0;
Then configure rx_coef_phi = "AMPCOMP" from step 18.

15. Read the amplitude of the signal at I and Q path with command DC_OFFSET read command (100 milliseconds average in this example)
Command ID: DC_OFFSET_READ
Send Command: 84 A9 61 00 02 20 39 05
Receive Command:84 A9 61 00 09 20 39 "DCI" "DCQ" "AMPI" "AMPQ""TXDC_high" "TXDC_low" "**AMPCOMP**"

16. Repeat step 17, 18 until AMPI and AMPQ are equal.

17. To complete whole RX DC and RX IQ calibration for all PGA levels, please repeat step 7 to step 16 each time with different PGA setting. PGA gain totally has 6 steps from 0-5.
Command ID: forceagcgain
Send Command: 84 A9 61 00 0d 00 6C "GAIN ADJUST" "EXT LNA GAIN" 80 00 10 00 04 00 10 00 04 00
Repeat step 7-16

Command ID: RSSI_WRITE
Send Command: 84 A9 61 00 0d 00 6C "GAIN ADJUST" "EXT LNA GAIN" 81 00 10 00 04 00

10 00 04 00

Repeat step 7-16

Command ID: RSSI_WRITE

Send Command: 84 A9 61 00 0d 00 6C “GAIN ADJUST” “EXT LNA GAIN” 82 00 10 00 04 00 10 00 04 00

Repeat step 7-16

Command ID: RSSI_WRITE

Send Command: 84 A9 61 00 0d 00 6C “GAIN ADJUST” “EXT LNA GAIN” 83 00 10 00 04 00 10 00 04 00

Repeat step 7-16

Command ID: forceagcgain

Send Command: 84 A9 61 00 0d 00 6C “GAIN ADJUST” “EXT LNA GAIN” 84 00 10 00 04 00 10 00 04 00

Repeat step 7-16

Command ID: forceagcgain

Send Command: 84 A9 61 00 0d 00 6C “GAIN ADJUST” “EXT LNA GAIN” 85 00 10 00 04 00 10 00 04 00

Repeat step 7-16

18. Next step is to calibrate TX DC offset. Set “DAC_GAINI” “DAC_GAINQ” to 0. This results in zero tx baseband I/Q out.

Command ID: DC_OFFSET_WRITE

Send Command: 84 A9 61 00 0d 00 39 “ADC_DCI” “ADC_DCQ” “DAC_DCI” “DAC_DCQ” **00 00 00 00**

Receive Command:84 A9 61 00 00 02 39 00

Command ID: DC_OFFSET_READ

Send Command: 84 A9 61 00 02 20 39 05

Receive Command:84 A9 61 00 0f 20 39 “RXDCI” “RXDCQ” “AMPI” “AMPQ” “TXDC_high” ”TXDC_low” “AMPCOMP”

The initial value for “DAC_DCI” and “DAC_DCQ” should be zero (00 00 00 00)

19. Adjust “DAC_DCI” and “DAC_DCQ” and until minimum DC leakage is obtained (indicated by ”TXDC_high” “TXDC_low”). “TXDC_high” and “TXDC_low” constitute a 32 bit value that indicates the DC leakage energy. The larger the value, the more the DC leakage.

Suggested procedure:

You can adjust DAC_DCQ first to get minimum leakage. For example use 0x0100 as a coarse step and 0x0010 as a fine step.

Then adjust DAC_DCI to get minimum leakage.

Again fine adjust DAC_DCQ.

And Fine adjust DAC_DCI again till leakage is minimum.

This complete the SCT3700 RX IQ calibration as well and RX DC and TX DC calibration process. The next step is TX IQ calibration. However, in most cases, TX IQ calibration is not necessary as SCT3700 is used in zero IF mode. Skip the steps below unless there is large TX IQ mismatch and SCT3700 is used in low IF mode.

20. Set the SCT3700 RF to transmit a carrier at frequency Fc. Now both SCT3700 TX and RX are using the same carrier frequency.

21. Generate a 1050 Hz sinusoidal I Q signal at the CT3258 TX side, with zero amplitude
    Command ID: MISC_GAIN
    Send Command: 84 A9 61 00 0D 00 6B 08 00 08 00 08 00 18 00 08 00 08 00
    Receive Command:84 A9 61 00 02 00 6B 00
22. Put TX IQ compensation filter by issue IQCOMP_COEF command with TX_IQ_MODE bit set to 1
    Command ID: IQCOMP_COEF_WRITE
    Send Command: 84 A9 61 00 0A 00 42 04 “rx_coef_amp” “rx_coef_phi” “tx_coef_amp” 00 00
    Receive Command:84 A9 61 00 00 02 42 00

23. Read the amplitude of the signal at I and Q path with command DC_OFFSET read command (100 milliseconds average in this example)
    Command ID: DC_OFFSET_READ
    Send Command: 84 A9 61 00 02 20 39 05
    Receive Command:84 A9 61 00 09 20 39 “DCI” “DCQ” “AMPI” “AMPQ”

24. Repeat 23, 24 until AMPI and AMPQ are equal

25. Put TX IQ compensation filter in normal mode by issue IQCOMP_COEF command with TX_IQ_MODE bit to 0
    Command ID: IQCOMP_COEF_WRITE
    Send Command: 84 A9 61 00 0A 00 42 00 “rx_coef_amp” “rx_coef_phi” “tx_coef_amp” “tx_coef_phi”
    Receive Command:84 A9 61 00 00 02 42 00

26. Read the amplitude of the signal at I and Q path with command DC_OFFSET read command (100 milliseconds average in this example)

Command ID: DC_OFFSET_READ
Send Command: 84 A9 61 00 02 20 39 05
Receive Command:84 A9 61 00 09 20 39 “DCI” “DCQ” “AMPI” “AMPQ”

27. Repeat 26, 27 until AMPI and AMPQ are equal. This completes the TX IQ calibration.

2.8 Error Handling

CT3258 responds to every packet received from the MCU, within T_rsp time, even it has parity error. Currently T_rsp = 40 milliseconds.

When the received packet has parity error, CT3258 responds with ACK_MESSAGE with parity error indication.

If the MCU does not receive a message from the CT3258 within T_rsp time, or receives an ACK_MESSAGE with parity error indication, or the received response message has parity error, it can consider the message lost, and resend the message if necessary.

For messages that initiated by CT3258 (not as response to MCU command), the MCU checks parity error if parity check is enabled. If parity error fails, it simply discards the message, without sending a parity error response. This generally falls into two categories:

1. The message is DIGC_DATA_FRAME or DPMR_SLD, with which CT3258 query for new data to be transmitted to the far end. In the case, if parity error fails, but the MCU can figure from the context that it is a DIGC_DATA_FRAME or DPMR_SLD message, it send query result as if there is no parity error. If the MCU can not figure out from the context, it simply drops the packet without sending a response.
2. The message is a CT3258 report message generated by CT3258. These types of messages do not require responses from the MCU. When there is a parity error, the MCU simply drops the message. If the MCU needs any specific information, it can send a query message to the DSP.

2.9 DPMR Standard User Interface

CT3258 supports the Standard User Interface described in ANNEX A of ETSI TS 102 490.

With the Standard User Interface, the mobile user uses a standard telephone key pad, with digits ‘0’ to ‘9’, plus ‘*’ and ‘#’ keys, to dial another mobile’s number. Each mobile is assigned a 7 digital number, and one or several group numbers. At the receiver, CT3258 match the “Called Number” of the incoming call to its own individual number and group numbers. It reports a call match to the MCU if there is a match. Otherwise, it reports a not-match.

Dialing with wild characters, abbreviated dialing and mask dialing are supported.

2.9.1 Message Used for Standard User Interface

2.9.1.1 Enable Standard User Interface

To enable the Standard User Interface, the user uses message with field ADDRESSING_MODE, and set the mode to Configured Address Mode. The user can also set a mask for mask dialing with the same field. By default, mask dialing is disabled (mask length is 0).

To enable layer 3 processing, so that call match is performed at the receiver, the user should set the process mode to 3 with PROCESS_MODE field.

The follow message enables the Standard User Interface and enables layer 3 processing

84 A9 61 00 04 00 1A 03 28 01

2.9.1.2 Set Own Numbers

The user can set its own number with DPMR_OWN_ID_BCD field. For example, the following message set the own number to 1234567.

84 A9 61 00 05 00 53 12 34 56 70

The user can also set one or several group number with GROUP_ID field. For example, the following message set the first group number to 1122334. GROUP_ID is 1.

84 A9 61 00 05 00 29 11 22 33 41

2.9.1.3 Set Dialed Numbers

2.9.1.3.1 Full 7 digit Number

The user can pass the dialed number to CT3258 with DPMR_CALLED_ID_BCD field. For example, the following message set the dialed number to 1234567.

84 A9 61 00 05 00 52 12 34 56 70

2.9.1.3.2 Wild Number Dialing

The user can also uses wild character '*' to dial a group of numbers, by replacing '*' with 'A'. For example, the following message set the dialed number to 123456*.

84 A9 61 00 05 00 52 12 34 56 A0

With this dialed number, called station with number '1234560' to '1234569' will respond.

2.9.1.3.3 Abbreviated Dialing

The user does not always have to dial the full number of the called station. By using abbreviated dialing, the user can only dial the digits of the called station number that is different than this own number. For example, if caller's own number is 1234567, and the called stations number is 1234568, the user only has to dial a single digit '8'. The message sent to CT3258 is thus:

84 A9 61 00 05 00 52 8F FF FF F0.

Note that the un-dialed digits in DPMR_CALLED_ID_BCD is filled with 'F'.

Upon receiving this message, CT3258 forms the complete Called ID by extracting the un-dialed from its Own ID. In this example, the full Called ID, 1234568 will be sent to the far end.

2.9.1.3.4 Masked Dialing

In some situation, the user is limited by the number of digits he/she can dial. To do this, the MCU sends a mask number (corresponding to the number of digits that is masked out for the user to dial) with ADDRESSING_MODE field.

For example, the following message limit the number of digital user can dial to 3.
84 A9 61 00 02 00 28 09.

If caller's own number is 1234567, and the user dials '789', the message sent to CT3258 is thus:

84 A9 61 00 05 00 52 78 9F FF F0.

Upon receiving this message, CT3258 forms the complete Called ID by extracting the un-dialed from its Own ID. In this example, the full Called ID, 1234789 will be sent to the far end.

If the number of digits user dial exceeds the maximum number of digits to dial, the MSB is discarded. In the previous examples, if user dials '99789', the message sent to CT3258 is:

84 A9 61 00 05 00 52 99 78 9F F0.

CT3258 will remove '99' and form the complete digit as '1234789'.

2.9.1.4 Report Call Match

If layer 3 processing is enabled, the receiver matches the called number of incoming call to its own individual number or group number, and report the matching status with CALL_MATCH fields.

CALL_MATCH is reported as long as there is an incoming call, whether or not the call is matched or not.

CALL_MATCH also report the status of Color Code match.

For group call, CALL_MATCH also report the group number that matches the call. For group call with wild characters, the group number is 0.

For examples, the following reported message from CT3258:

84 A9 61 00 02 00 27 00

tells the MCU that an individual call is matched, and Color Code is also matched;

84 A9 61 00 02 00 27 01

tells the MCU that an individual call is not matched, but Color Code is matched;

84 A9 61 00 02 00 27 02
indicates that a group call with wild characters is received, and Color Code is also matched;

84 A9 61 00 02 00 27 1A
indicates that a group call with group number 1 is received, but Color Code is not matched;

It is possible that a valid call may have a mis-match on the Color Code, as Color code, unlike the Called ID, is not FEC protected. When the MCU receives a call with un-matched Color Code, it may poll SCT3250 for Color Code in subsequent voice frames through reading DPMR_CC after waiting 160 milli-seconds to see if it is indeed a valid call.

The Called ID and Own ID information is embedded in the header frame (80 milliseconds) and in 4 CCH frames (320 milliseconds). For normal calls with valid header frame, call match is done within 80 milliseconds. However, with late arrival calls, CT3258 has to extract call information over 4 error free CCH frames. Adding the time to synchronize FS2, the minimum time to report a call match is over 500 milliseconds. In case of high BER (bit error rate), the SYNC time can be longer.

2.10DMR Number and Dialing Plan

CT3258 supports the Numberling Plan described in ANNEX C of ETSI TS 102 361-2.

With the Standard User Interface, the mobile user uses a standard telephone key pad, with digits ‘0’ to ‘9’, plus ‘*’ and ‘#’ keys, to dial another mobile’s number. Each mobile is assigned a 7 digital number, and one or several group numbers. At the receiver, CT3258 match the “Called Number” of the incoming call to its own individual number and group numbers. It reports a call match to the MCU if there is a match. Otherwise, it reports a not-match.

2.10.1 Message Used for DMR Dialing Plan

2.10.1.1 Set Own Numbers

The user can set its own number with DMR_OWN_ID_BCD field. For example, the following message set the own number to 1234567.

84 A9 61 00 05 00 7E 01 23 45 67

The user can also set one or several group number with GROUP_ID field. For example, the following message set the first group number to 1122334.

84 A9 61 00 05 00 29 11 22 33 41

2.10.1.2 Set Dialed Numbers

2.10.1.2.1 Full 8 digit Number

The user can pass the dialed number to CT3258 with DMR_CALLED_ID_BCD field. For example, the following message set the dialed number to 1234567.

84 A9 61 00 05 00 7D 01 23 45 67

2.10.1.2.2.All Call Dialing

The user can use All Call Dialing. See the derails in the table.

BCD ID	Remark
84 A9 61 00 05 00 7D 00 aa aa(00******)	All Talkgroup ID0
84 A9 61 00 05 00 7D 01 aa aa(01******)	All Talkgroup ID1
etc.	etc.
84 A9 61 00 05 00 7D 09 aa aa(09******)	All Talkgroup ID9
84 A9 61 00 05 00 7D 0a aa aa(0*******)	All Talkgroup

2.10.1.3 Report Call Match

In Easy Mode, the receiver matches the called number of incoming call to its own individual number or group number, and report the matching status with CALL_MATCH fields.

CALL_MATCH is reported as long as there is an incoming call, whether or not the call is matched

or not.

CALL_MATCH also report the status of Color Code match.

For group call, CALL_MATCH also report the group number that matches the call.

For examples, the following reported message from CT3258:

84 A9 61 00 02 00 27 00

tells the MCU that an individual call is matched, and Color Code is also matched;

84 A9 61 00 02 00 27 01

tells the MCU that an individual call is not matched, but Color Code is matched;

84 A9 61 00 02 00 27 02
indicates that a group call is received, and Color Code is also matched;

84 A9 61 00 02 00 27 1A
indicates that a group call with group number 1 is received, but Color Code is not matched;

It is possible that a valid call may have a mis-match on the Color Code, as Color code, unlike the Called ID, is not FEC protected. When the MCU receives a call with un-matched Color Code, it may poll SCT3250 for Color Code in subsequent frames.

The Called ID and Own ID information is embedded in the header frame (30 milliseconds) and in 6 voice frames (360 milliseconds). For normal calls with valid header frame, call match is done within 30 milliseconds. However, with late arrival calls, CT3258 has to extract call information over 6 error free voice frames, or 360 milliseconds. In case of high BER (bit error rate), the SYNC time can be longer.

2.11Power Saving Mode

The host can put CT3258 into low power modes with command CHIP_LOWPOWER (field type 0x15). The low power modes are listed below.

0: Normal
1: Reserved
2: Reserved
3: Power down certain peripheral of CT3258 including serial port, DMA and timers.
4: CT3258 enters IDLE mode
5: CT3258 enters Sleep mode

6: CT3258 enters Halt mode

If CT3258 is in power saving mode lower or equal to 4, it can be waken up with command CHIP_LOWPOWER with field value 0 (normal power mode). CT3258 resume all states before it enters the lower power mode.

If CT3258 enters power saving mode 5, it can be waken up with INT0. CT3258 resume all states before it enters the lower power mode.

If CT3258 enters power saving mode 6, it can be waken up with NMI. Once NMI is received by CT3258, it restarts the CT3258. All information sent to CT3258 is lost and need to be reinitialized. The difference of reset with NMI and reset with RESET chip is that firmware would not need to be reloaded with NMI.

The same command can be used to power down the external vocoder and codec by setting bit 5 and bit 5 of field value.

2.12 Codec Selection

CT3258 has native support for the following two codecs:

TI TLV320AIC3204
Wolfson WM8758B

TLV320AIC3204 and WM8758B are stereo codecs. With a stereo codec, the left ADC is connected to Microphone input, the right ADC is connected to the FM demod port. The right DAC is connected to the VCTCXO port, while the left DAC is connected to the audio line port in the RX mode and the VCO port in the TX mode.

The codec can be selected with CODEC_SELECT command (field ID 0x2D).

Other codecs can be used with CT3258 if it supports I2C interface and conforms to the serial format of CT3258 high speed serial port.

2.13 Codec Configurations

The host can send I2C command to codec through CT3258. The commands are sent to CT3258 via HPI interface, with bit 6 of TYPE field set to 1. Upon receiving such commands, CT3258 extracts information from the packet after the TYPE fields, and adds I2C address of the codec, and sends the information to the codec via I2C interface.

To configure codec, the host first sends the I2C address of the codec to CT3258 with command SET_I2C_ADDRESS. It then can send codec configuration command directly to the codec by

setting the packet type to be 0x40.

For example, to send command to WM8758B to reset it, the user can use the following commands:
Command ID: SET_I2C_ADDRESS
Send Command: 84 A9 61 00 02 03 2E 34
Receive Command:84 A9 61 00 02 03 2E 00
Command ID: CODEC_COMMAND
Send Command: 84 A9 61 00 02 40 00 00
Receive Command:84 A9 61 00 02 40 17 00

When CT3258 receives the above command from the host, it extract “00 00” from the command, and 0x34 to the beginning, and sends “0x34, 0, 0” to the codec via I2C interface.

2.14 Debug Mode

2.14.1 Setting Up Two Point Modulation

CT3258 has two outputs in TX mode connecting to VCO and VCTCXO port of the RF unit to facilitate 2 point modulation. The user should refer to “CT3258 Application Notes – Analog Interface Setup” for details on how to adjust two point modulation.

2.14.2 BER Test

To check the link quality, CT3258 can be configured to do BER test.

The user should refer to “CT3258 Application Notes – Analog Interface Setup” for details on how to configure for BER test..

2.14.3 Loop Back

To verify the different part of CT3258 hardware and software, a set of loop backs can be enabled. The following loop back modes are especially useful in verifying external codec and vocoder connection of CT3258.

2.14.3.1 Audio Codec Loop Back and Modem Codec Loop Back

With audio codec loop back, the audio signal from the microphone goes through the ADC in the TX path and loops back to the DAC in the RX path, and sent out to the headphone output.

With modem codec loop back, the modem signal from the DEMOD_Q port goes through the ADC in the RX path and loops back to the DAC in the TX path, and sent out to the MOD_Q port.

The user can apply signal generated at the input and monitor the output with a scope. The correct operation of audio codec loop back and modem codec loop indicate the correct connection of external codec and the analog audio and modem path.

To set up for audio codec loop back and modem codec loop back, the user should follow these steps:

Send command to set CT3258 in audio codec loop back mode:
ALOOP_CODEC_LOOP: 61 00 04 00 18 03 11 00

The user should then be able to hear his voice from the headphone, and see the signal from a scope.

2.14.3.2 Audio Vocoder Loop Back

With audio vocoder loop back, the audio signal from the microphone goes through the ADC, the decimator, the voice encoder in the TX path and loops back to the voice decoder, the interpolator and the DAC in the RX path, and sent out to the headphone output.

Correct vocoder loop back indicate correct operation of the external vocoder (e.g. AMBE3000).

To set up for audio vocoder loop back, the user should follow these steps:

Send command to set CT3258 in audio vocoder loop back mode:
ALOOP_VOCODER_LOOP: 61 00 04 00 18 03 03 00

The user should then be able to hear his voice from the headphone.

