iM501

Voice Interface Device

Product Datasheet
Feb 10, 2017

NOTICE

THIS DOCUMENT CONTAINS INFORMATION ON A PREPRODUCTION PRODUCT. SPECIFICATIONS AND PREPRODUCTION INFORMATION HEREIN ARE SUBJECT TO CHANGE WITHOUT NOTICE.

FORTÉMEDIA INCORPORATED AND ITS SUBSIDIARIES (FORTÉMEDIA) RESERVE THE RIGHT TO MAKE CORRECTIONS, MODIFICATIONS, ENHANCEMENTS, IMPROVEMENTS, AND OTHER CHANGES TO ITS PRODUCTS AND SERVICES AT ANY TIME AND TO DISCONTINUE ANY PRODUCT OR SERVICE WITHOUT NOTICE. CUSTOMERS SHOULD OBTAIN THE LATEST RELEVANT INFORMATION BEFORE PLACING ORDERS AND SHOULD VERIFY THAT SUCH INFORMATION IS CURRENT AND COMPLETE. ALL PRODUCTS ARE SOLD SUBJECT TO FORTÉMEDIA'S TERMS AND CONDITIONS OF SALE SUPPLIED AT THE TIME OF ORDER ACKNOWLEDGMENT.

FORTÉMEDIA ASSUMES NO LIABILITY FOR APPLICATIONS ASSISTANCE OR CUSTOMER PRODUCT DESIGN. CUSTOMERS ARE RESPONSIBLE FOR THEIR PRODUCTS AND APPLICATIONS USING FORTÉMEDIA COMPONENTS. TO MINIMIZE THE RISKS ASSOICATED WITH CUSTOMER PRODUCTS AND APPLICATIONS, CUSTOMERS SHOULD PROVIDE ADEQUATE DESIGN AND OPERATING SAFEGUARDS.

FORTÉMEDIA, INC. PRODUCTS ARE INTENDED FOR NEITHER LIFE SAVING NOR LIFE SUSTAINING APPLICATIONS AND FORTÉMEDIA, INC. THUS, ASSUMES NO LIABILITY IN SUCH USAGES. FORTÉMEDIA, INC. PRDUCTS MAY ONLY BE USED IN LIFE-SUPPORT DEVICES OR SYSTEMS WITH THE EXPRESS WRITTEN APPROVAL OF FORTÉMEDIA, INC., IF A FAILURE OF SUCH COMPONENTS CAN REASONABLY BE EXPECTED TO CAUSE THE FAILURE OF THAT LIFE-SUPPORT DEVICE OR SYSTEM, OR TO AFFECT THE SAFETY OR EFFECTIVENESS OF THAT DEVICE OR SYSTEM. LIFE SUPPORT DEVICES OR SYSTEMS ARE INTENDED TO BE IMPLANTED IN THE HUMAN BODY, OR TO SUPPORT AND/OR MAINTAIN AND SUSTAIN AND/OR PROTECT HUMAN LIFE. IF THEY FAIL, IT IS REASONABLE TO ASSUME THAT THE HEALTH OF THE USER OR OTHER PERSONS MAY BE ENDANGERED.

WE HEREIN DISCLAIM ANY AND ALL WARRANTIES, INCLUDING BUT NOT LIMITED TO WARRANTIES OF NON-INFRINGEMENT, REGARDING CIRCUITS, DESCRIPTIONS AND CHARTS STATED HEREIN.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

Revision History

Revision	Date	Description
0.0	06/23/2015	Initial Writing
1.0	06/30/2015	First version is ready
1.1	07/10/2015	Technical accuracy review is done
1.2	07/15/2015	Functionalities
1.2.1	07/23/2015	Details improve/correct
1.2.2	07/24/2015	Update block diagram
1.2.2	08/25/2015	Add the Pin layout, Pin assignment, and Package dimensions.
1.2.3	08/26/2015	Update Figure 13 and 14, fix typo in I2C example, change rise and fall times of I2C in table 7
1.2.4	09/04/2015	Update table 6
1.2.5	10/29/2015	Fix typo in pin list. Update the Application Diagrams.
1.2.6	11/10/2015	Update the Chapter 5
1.2.7	11/18/2016	Add section 5.5) LDO configuration for adjustable MIPs control
1.2.8	12/23/2016	Description on capacitor requirement per MIP setting
1.2.9	02/03/2017	Update section 5.5) LDO
1.3.0	02/07/2017	Editorial changes for first Release into Launch Package
1.3.1	02/09/2017	Editorial changes in format and abbreviations
1.3.2	02/10/2017	Editorial changes in Functionalities section

1. General Description

1.1 Overview:

Fortemedia iM501 integrates a Tensilica Hi-Fi Mini DSP core to provide voice-triggered wakeup function and post-wakeup voice processing. Working in conjunction with the Fortemedia microphone processor, iM205, iM501 can provide an ultimate power saving solution. There exists a total 474 Kbytes of on-chip memory such that multiple voice-trigger engines, smart microphone controls, and a minimum of 3 seconds of voice buffer could co-exist in the system. Together with the Fortemedia advanced microphone voice-enhancement technology, iM501 is a versatile Voice Interface Device improving both human-to-machine and human-to-human voice communications. The iM501 firmware framework also implements a VOS (Voice Operating System) for easy data path plug-in of 3rd-party algorithms.

1.1.1 Hardware Features

- 128 Kbytes IRAM (Instruction RAM), 250 Kbytes DRAM (Data RAM), 96 Kbytes SRAM (System RAM) with Auto Voice Buffer Scheme
  - Provide more than 3 seconds of dual voice buffer for seamless operation
  - Hardware assisted auto-buffers for MICs, DAC/TXs and RXs data to reduce system overhead
- Programmable LDO (0.9~1.32V) and PLL (1MHz ~ 150MHz), for MIPS and power efficiency requirements
- Support various sampling rates: 8K/16K/24K/32K/44.1K/48KHz
- Two PDM_DATA Input (support up to 4 Digital MICs) and two PDM_DATA Output pins
- Two TX (microphone) and one RX (AEC reference) of I2S are supported
- Microphone output can be I2S or PDM format
- 32K/512K/768K/1MHz PDM_CLKO with 16KHz sampling rate at power saving mode
- SPI/I²C for program and acoustic model download at speed up to 20MHz/400KHz
- AVD function to provide ultra-low current for wakeup function
- (iM501+iM205) can be SIP-integrated to provide 1 Analog MIC (AMIC) and 3 Digital MICs (DMIC) System-On-Chip (SoC).

1.1.2 Software Features

- VOS (Voice Operating System) for always-on ambience monitoring
- Programmable speech detection and voice triggers
- Beamforming, noise suppression and acoustic echo cancellation
- Acoustic ambience analysis

1.2 Typical Application

- Voice triggering applications for mobile devices
- Voice processing and analysis for wearable devices
- IoT voice control
- Smart Microphone system
- Automotive voice input processing

2. Block Diagram

2.1 Functional Block Diagram

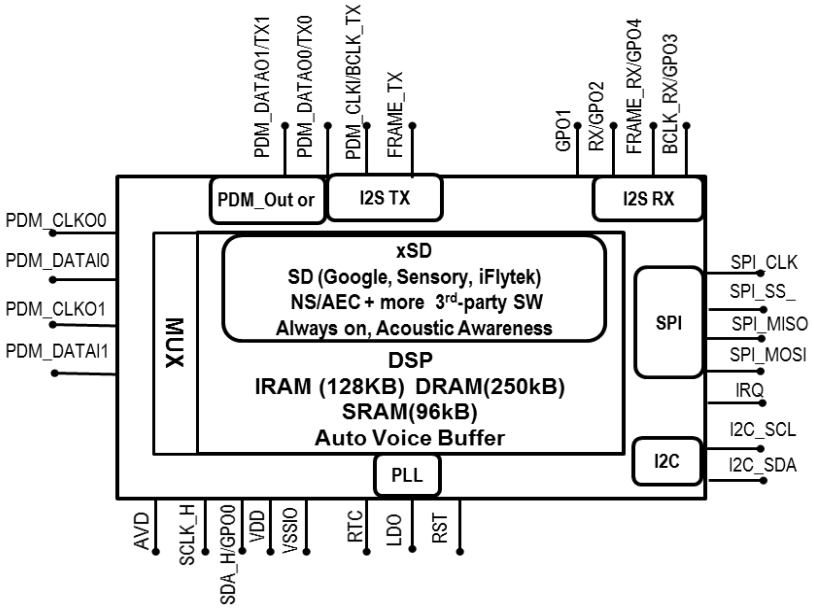

Figure 1. Functional Block Diagram of iM501

2.2 System Application Diagram

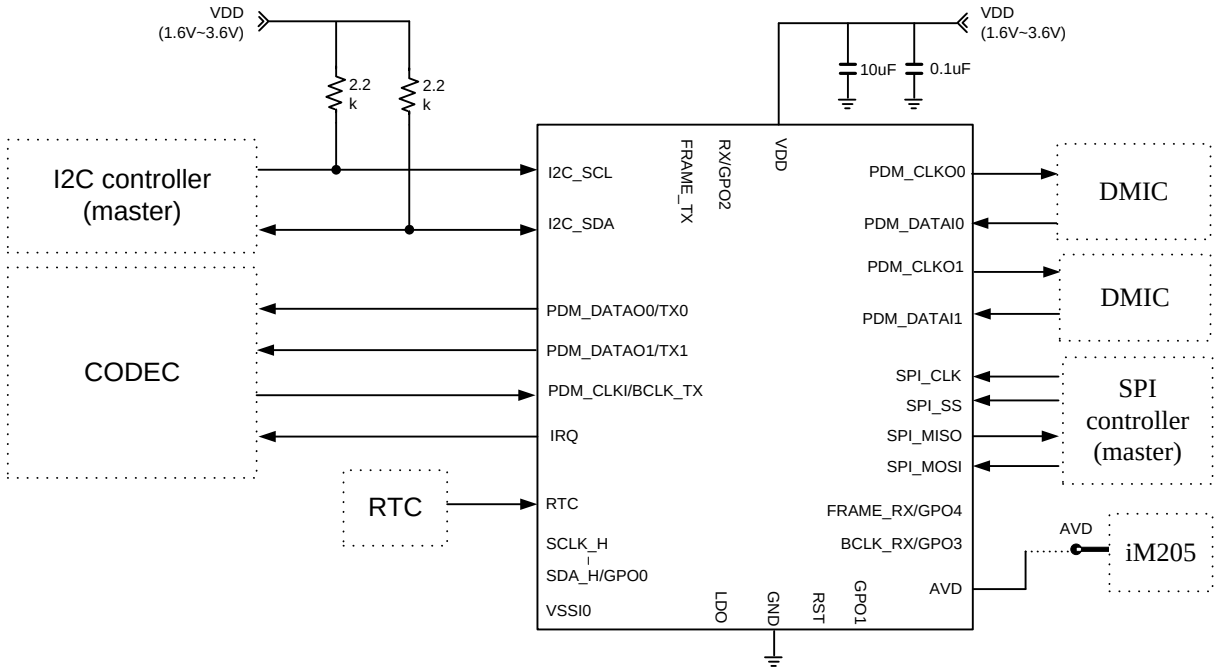

Figure 2. iM501 System Application Diagram with Microphones and Host

3. PIN Layouts

3.1 QFN32 Package

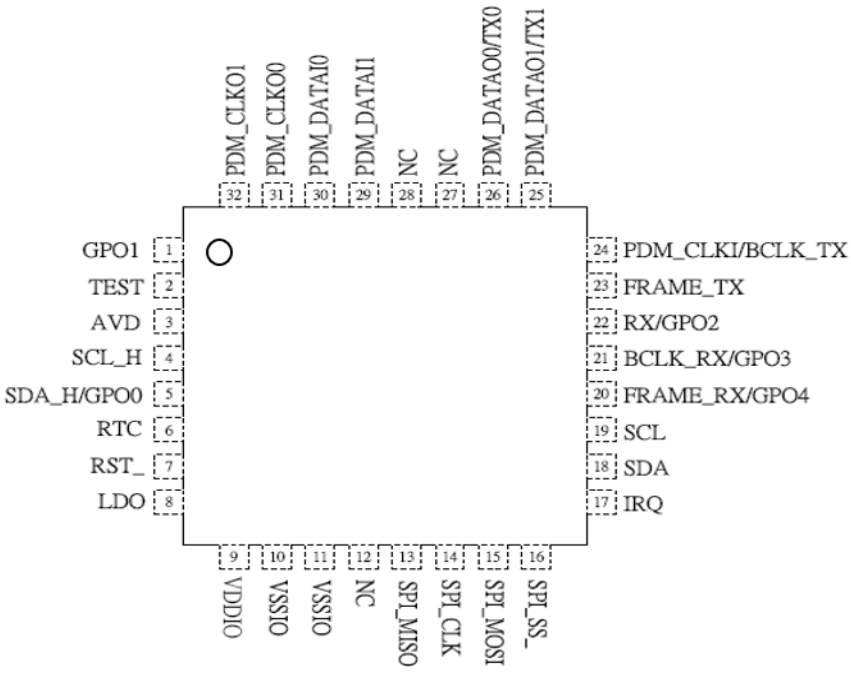

Figure 3. PIN Assignment of iM501 in QFN Package

3.2 CSP Package

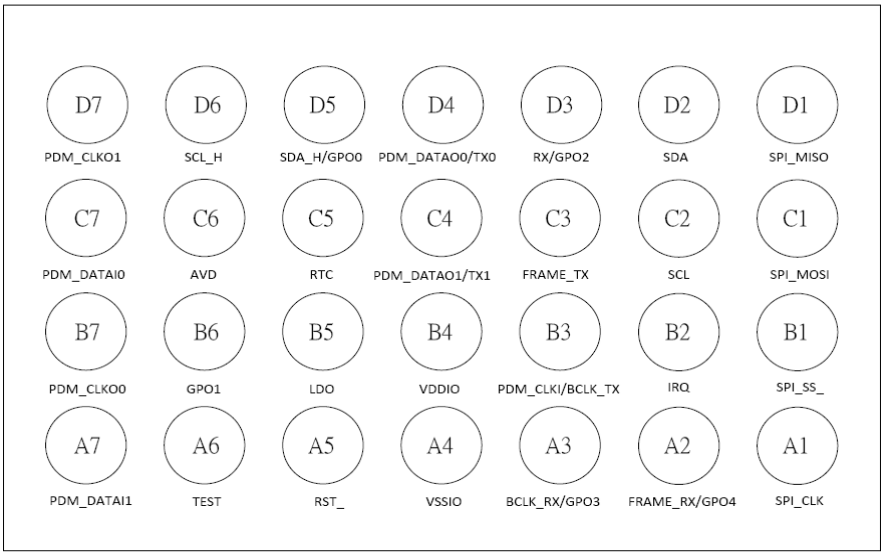

Figure 4. PIN Assignment of iM501 in CSP Package

4. PIN Description

Table 1. PIN Function Description

Pin Name	IO	Pin# in		Description
		QFN	CSP	
PDM_DATAI0	I	30	C7	PDM data from MIC0 and MIC1
PDM_DATAI1	I	29	A7	PDM data from MIC2 and MIC3.
PDM_CLKO0	O	31	B7	PDM clock output to microphone. PDM_CLKO0 is for always on microphone and can't be turned off. Frequency of PDM_CLKO0 can be 32K, 512K, 768K or 1.024MHz during power saving mode. PDM_CLKO0 is directly from PDM_CLKI0 pin when DSP is bypassed in normal mode
PDM_CLKO1	O	32	D7	2nd PDM clock to microphone. This clock can be turned off.
GPO1	O	1	B6	General purpose output
TEST	I	2	A6	Test pin for scan test. This pad has internal pull down resister
AVD	I	3	C6	AVD input from iM205 microphone, otherwise no connection or connect to high
SCL_H	O	4	D6	I2C clock to Fortemedia DMIC (iM205), otherwise no connection. This pad has internal pull up resister at 95K
SDA_H/GPO0	IO	5	D5	I2C data to Fortemedia DMIC (iM205), otherwise no connection. This pad has internal pull up resister at 95K
RTC	I	6	C5	32K clock input as reference in power saving mode.
RST_	I	7	A5	Reset, active low. This pad has internal pull up resister
LDO	P	8	B5	LDO output for internal core power and test purpose.
VDDIO	P	9	B4	Power supply 1.6~3.6V
VSSIO	G	10/11	A4	Double bounded ground pads.
N.C.		12		
SPI_MISO	O	13	D1	Data from SPI slave to SPI master
SPI_CLK	I	14	A1	SPI clock.
SPI_MOSI	I	15	C1	Data from SPI master to SPI slave
SPI_SS_	I	16	B1	SPI chip select. Active low
IRQ	O	17	B2	Wake-up trigger. Polarity is selectable.
SDA	IO	18	D2	I2C data connected to host.
SCL	I	19	C2	I2C clock connected to host. Up to 400 KBPS.
FRAME_RX/GPO4	IO	20	A2	Frame of RX of I2S or GPO4.
BCLK_RX/GPO3	IO	21	A3	BCLK of RX of I2S or GPO3
RX/GPO2	IO	22	D3	RX of I2S or GPO2
FRAME_TX	I	23	C3	Frame of TX of I2S
PDM_CLKI/BCLK_TX	I	24	B3	PDM clock from CODEC or BCLK of I2S. PDM clock range is from 1.024~4.096MHz.

Pin Name	IO	Pin# in		Description
PDM_DATAO1/TX1	O	25	C4	MIC2 and MIC3 to CODEC or TX1 of I2S.
PDM_DATAO0/TX0	O	26	D4	MIC0 and MIC1 to CODEC or TX0 of I2S.
N.C.		27		
N.C.		28		

5. On-Chip Major Function Description

5.1 I²C Interfaces

iM501 implements two I2C interfaces, one is used as I2C slave connected to host processor(AP/MCU), the other one is used as I2C host and connected to Fortemedia DMIC.

I²C slave bus (SCLK, SDA) is mainly used to download software and necessary initial configuration parameters during reset and power-up, and to transmit and receive control commands at run-time. The I²C slave bus can transfer up to 400 Kbps, its ID is 0xE2.

5.1.1 I²C master

I2C master implemented inside iM501 is to control Fortemedia DMIC only. The read/write operations of master I2C have been implemented by software inside iM501.

5.1.2 I²C Slave Command Format

The standard byte format of I2C data line must be 8–bit long. Each byte consists of 8 bits plus 1 acknowledge bit, with 1 bit per clock cycle. If operating as a receiver, it will return an Acknowledge (ACK) bit after each successful byte transfer, otherwise it returns a NOACK signal. Data transfer can be aborted if the master device generate a STOP condition to terminate a transfer. Each data transfer frame must start with a START or a RESTART symbol and ends by a STOP symbol.

Table 2. I²C START and STOP data transition

S: START	SDA transition from 1 to 0 when SCL=1
P: STOP	SDA transition from 0 to 1 when SCL=1

Within the data transfer frame, multiple command sequences are allowed and the maximum numbers of bytes per frame is 64K. Each command sequence starts with a command entry byte (e.g. 0x2B, as DRAM_WRITE), and a number of bytes per specific command. The following figures and tables summarize the details for the I²C command sequence.

Table 3. I²C Command Names

Command entry name	Command entry byte	#Bytes for each functional bytes		
		Address byte	Data byte	Total (cmd+addr+data)
IRAM_WRITE	0x0D	3	4	8
IRAM_READ	0x07	3	0	4
DRAM_WRITE	0x2B	3	2	6
DRAM_READ	0x27	3	0	4
REG_WRITE_1	0x48	1	1	3
REG_WRITE_2	0x4A	1	2	4
REG_READ	0x46	1	0	2

Table 4. Bit Definition for I²C Command Entry Byte

Data Length Field Name	1 Byte	
Bit#	Value	Description
D7 Data Only	1	Data only Command. The transfers after the bit set are data only, no address bytes, until the byte count register is zero.
	0	Normal Command.
D6 - D5 Source/Destination	00	IRAM Access.
	01	DRAM Access
	10	I2C Internal Register Access
	11	Reserved
D4		Reserved
D3 Read/Write	0	Read
	1	Write
D2-D1 Number of Bytes of Data	00	1 byte data write, or in Data Port Read mode.
	01	2 bytes data write
	10	4 bytes data write
	11	0 byte data (in Data Memory Read mode).
D0 Number of Bytes of Address	0	1 address byte for I2C Register Write, or 1 address byte for reading Data Port
	1	3 address bytes for accessing IRAM or DRAM

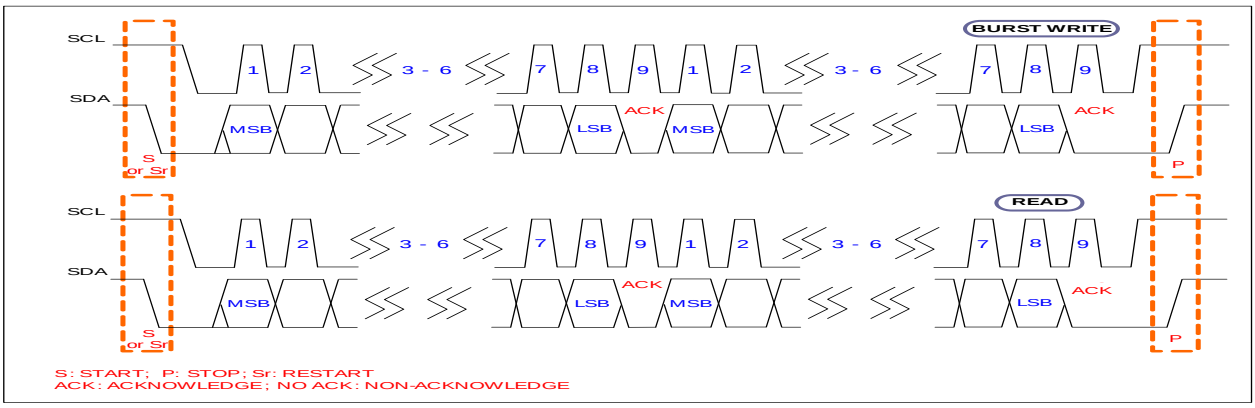

Figure 5. I²C Data Transfer Command Protocol

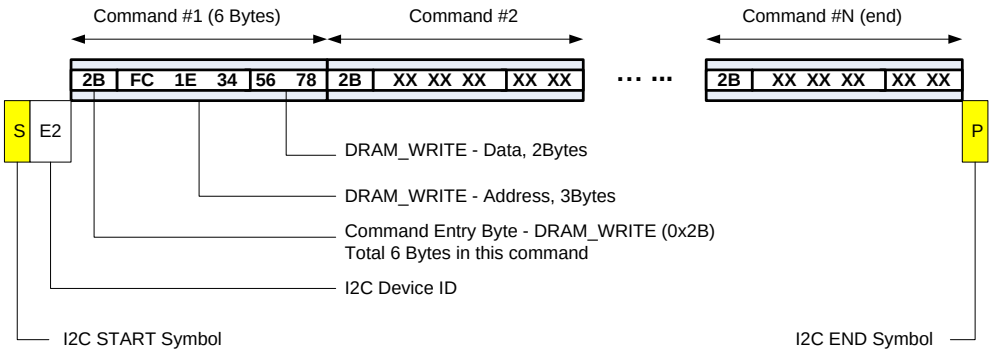

Figure 6. I²C Command Sequence

5.1.3 I²C Burst Modes

The burst mode provides very efficient data transfer between host and iM501. For example, transferring 64 Kbytes data (upload or download) only takes about 0.58 second (64K * 9/1M). To enable this mode:

- Register 0x12 must be set to 0x04 after power up, for host (I²C, not SPI) to access iM501.
- Register 0x0F is set to 0x05 for initial downloading when host is I²C.

The burst mode can be applied to both IRAM and DRAM, and for both upload (Read) and download (Write) operations.

5.1.4 I²C Burst Write

I2C burst write is implemented toward IRAM and DRAM only, but not for I2C registers.

I²C Bust Write of DRAM

The sequence to issue an I2C burst write of DRAM is:

1. **Program byte counter**: Byte counter indicates number of bytes to be written, its value must be multiple of 2 for burst write mode in DRAM. For instance, if 0x20 is written to byte counter, 32-byte will be written.

   Sample sequence to program byte counter with value 0x20 is: I2C_start -> ID W (0xE2) -> Command (0x4A, write I2C register with 1 byte address and 2-byte data) -> Address (0x08, byte counter) -> Data (0x20) -> Data (0x00) -> stop

2. **Write one word to generate starting address of burst write** (assuming 0x0FFC_6040 of DRAM, LSB of address is sent first): I2C_start -> ID W (0xE2) -> Command (0x2B, DRAM write, 3-address byte) -> address_1 (0x3E) -> address_2 (0x60) -> address_3(0xFC) -> Data_1 (0xCD, low byte for 0x0FFC_603E) -> data_2 (0xAB, high byte, 0xFFC_603F) → Stop

3. **Issue command to setup burst mode**: I2C_start -> ID W (0xE2) -> Command (0xA8, data only, DRAM, Write) -> stop

4. **Issue I2C Write** (DRAM write is 2-byte operation, and LSB of data is written first): I2C_start -> ID W (0xE2) -> data0 (0x0FFC_6040) -> data1 (0x0FFC_6041) -> data2 (0x0FFC_6042) -> … -> data30 (0x0FFC_605E) -> data31 (0x0FFC_605F) -> Stop

*Note: Data written in step 2 is not considered as part of byte counter

I²C Bust Write of IRAM

The sequence to issue an I²C burst write of IRAM is:

1. **Program byte counter**: Byte counter indicates number of bytes to be read, value of byte counter must be multiple of 4 for burst write mode of IRAM. For instance, if 0x20 is written to byte counter, 32-byte will be read out.

   Sample sequence to program byte counter with value 0x20 is: I2C_start -> ID W (0xE2) -> Command (0x4A, write I2C register with 1 byte address and 2-byte data) -> Address (0x08, byte counter) -> Data (0x20) -> Data (0x00) -> stop

2. **Write double words (4 bytes) to generate starting address of burst write** (assuming 0x1000_5344 of DRAM, LSB of address is sent first): I2C_start -> ID W (0xE2) -> Command (0x0D, IRAM write, 3-address byte) -> address_1 (0x40) -> address_2 (0x53) -> address_3(0x00) -> Data_0 (0xEF, lowest byte, 0x1000_5340) -> data_1 (0xCD, 2nd low byte, 0x1000_5341) -> Data_2 (0xAB, 2nd highest byte, 0x1000_5342) -> data_3 (0x89, highest byte, 0x1000_5343) -> Stop

3. **Issue burst read and indicate memory type**, IRAM or DRAM (setup burst mode): I2C_start -> ID W (0xE2) -> Command (0xA8, data only, DRAM, Write) -> stop

4. **Issue I2C Write** (IRAM write is 4-byte operation, data of lowest byte is written first): I2C_start -> ID W (0xE2) -> data0 (0x1000_5344) -> data1 (0x1000_5345) -> data2 (0x1000_5346) -> … -> data30 (0x1000_5362) -> data31 (0x1000_5363) ->Stop

   *Note: Data written in step 2 is not considered as part of byte counter

5.1.5 I²C Burst Read

I²C burst read is implemented toward IRAM and DRAM only, but not for I²C registers.

I²C Bust Read of DRAM

The sequence to issue an I²C burst read of DRAM is:

1. **Program byte counter**: Byte counter indicates number of bytes to be read, its value must be multiple of 4 for burst read mode. For instance, if 0x20 is written to byte counter, 32-byte will be read out.

   Sample sequence to program byte counter with value 0x20 is: I2C_start -> ID W (0xE2) -> Command (0x4A, write I2C register with 2-byte data and 1 byte address) -> address (0x08, byte counter) -> Data (0x20) -> Data (0x00) -> stop

2. **Send first address of burst read** (assuming 0x0FFC_6040 of DRAM, LSB of address is sent first): I2C_start -> ID W (0xE2) -> Command (0x27, DRAM read, 3-address byte) -> address_1 (0x40) -> address_2 (0x60) -> address_3(0xFC) -> Stop

3. **Issue burst read and indicate memory type**, IRAM or DRAM (setup burst mode): I2C_start -> ID W (0xE2) -> Command (0xA0, data only, DRAM) -> stop

4. **Issue I2C burst read command** (DRAM): I2C_start -> ID R (0xE3)

5. **Data will show serial on SDA pin**, data in lowest address shows first: 0x01 (address 0x0FFC_6040) -> 0x02 (address 0x0FFC_6041) -> 0x03 (address 0x0FFC_6042) -> … -> 0x20 (address 0x0FFC_605F)

I²C Bust Read of IRAM

The sequence to issue an I²C burst read of IRAM is:

1. **Program byte counter**: Byte counter indicates number of bytes to be read, its value must be multiple of 4 for burst read mode. For instance, if 0x20 is written to byte counter, 32-byte will be read out.

   Sequence to program byte counter with value 0x20: I2C_start -> ID W (0xE2) -> Command (0x4A, write I2C register with 2-byte data and 1 byte address) -> address (0x08, byte counter) -> Data (0x20) -> Data (0x00) -> stop

2. **Send first address of burst read** (assuming 0x1000_5340 of IRAM, LSB of address is sent out first): I2C_start -> ID W (0xE2) -> Command (0x07, IRAM read, 3-address byte) -> address_1 (0x40) -> address_2 (0x53) -> address_3(0x00) -> Stop

3. **Issue burst read command**: I2C_start -> ID W (0xE2) -> Command (0x80, data only, DRAM) -> stop

4. **Issue I2C burst read command** (IRAM): I2C_start -> ID R (0xE3)

5. **Data will show serial on SDA pin**, data in lowest address shows first: 0x01 (address 0x1000_5340) -> 0x02 (address 0x1000_5341) -> 0x03 (address 0x1000_5342) -> … -> 0x20 (address 0x1000_535F)

5.2 I²S Interface

There are TX0 and TX1 pins to support up to 4 microphone output. TX0 and TX1 pins are shared with PDM_DATAO0 and PDM_DATA1, respectively. When bit 1:0 of 0x0FFF_FF16 are set to "10", TX0 and TX1 of I²S are selected. Each TX pin (TX0, TX1) can support 2 channels (microphones).

TX0/TX1 and RX have independent FRAME and BCLK pins, so their clock rates of BCLK and FRAME can be different. I2S of iM501 supports TDM format from 2 to 8 slots. Data can be placed into any two slots of TX0 and TX1. RX data also supports 2 to 8 slots, the main purpose of RX is for AEC reference.

The setting of format of TX and RX is through register 0x0FFF_FF76 and 0x0FFF_FF78, respectively.

The format of TX and RX can be programmed. I2S provide a wide range of formats as below:

- Support 2 to 8 slots
- Slot length can be 16, 20, 24, 28, 32-bit
- Word length can be 16, 20, or 24-bit
- First-bit data delay can be 0 or 1
- Frame(LRCK) can be started with low or high
- Left/right alignment of word inside slot
- Bit order can be MSB first or LSB first
- FRAME(LRCK) width can be 1 to (slot number) * (slot length) - 1 cycles of BCLK
- Data can be transmitted/received at both rising or falling edge of BCLK

TX and RX can support up to 2 slots. Slot length can be 16-, 20-, 24-, 28-, 32-bit, it must be larger or equal to word length, which can be 16-, 20-, or 24-bits. The serial ports in iM501 can only be in slave mode (no master mode). The transmit data is padded with "0" when slot length is larger than word length.

5.3 PDM Data Input (Microphone Interface)

iM501 uses PDM interface (PDM_CLKO0, PDM_DATAI0, PDM_CLKO1, PDM_DATAI1) to communicate with external digital microphone(s). Please refer to the below diagram for more details.

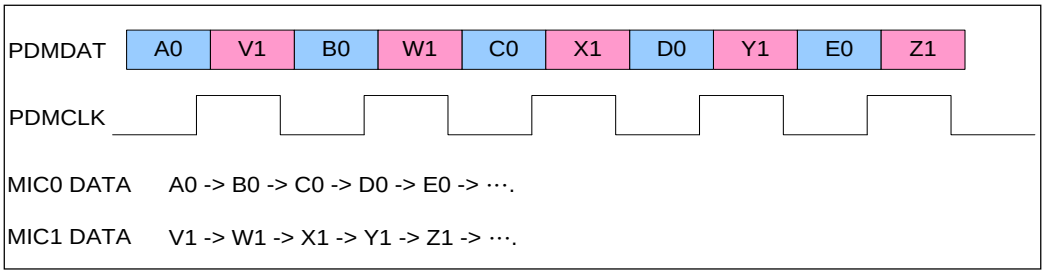

Figure 7. iM501 Digital Microphones Interface Clock and Data

The iM501 has capability to provide Fortemedia microphone or regular DMIC. The interface and clock rate (PDM_CLKO) to Fortemedia or regular DMIC are different.

5.3.1 Fortemedia DMIC

The major difference between Fortemedia DMIC and regular DMIC are the existence of I²C and AVD pins:

- Fortemedia DMIC can be programmed through I²C host of iM501,
- AVD pin is indicator to signal that there is an incoming voice stream during power saving mode.

After power up, host can initialize Fortemedia DMIC through iM501. Registers 0x0FFF_FF00, 0x0FFF_FF02 and 0x0FFF_FF04 are used for I2C communication with Fortemedia DMIC, or the host can ask DSP to do initialization of Fortemedia DMIC.

In normal operation, PDM_DATAO0/PDM_DATAO1 can directly come from DMIC or from DSP of iM501 by setting bit [1:0] of 0x0FFF_FF16. The clock rates of PDM_CLKO0/PDM_CLKO1 is same as PDM_CLKI when bit [1:0] of 0x0FFF_FF16 are set to 00 (DMIC bypass) or 2.048MHz when bit [1:0] are set to 01 (from DSP).

In Power Saving mode, clock rate of PDM_CLKO0/PDM_CLKO1 is either 32 KHz when AVD is not activated, or 1.024MHz when AVD is activated. When 32 KHz of PDM_CLKO0 is provided, PLL inside iM501 is powered down.

Hi-Fi Mini DSP and all digital filters are reset for around 33us (one cycle of 32 KHz clock) when AVD is activated. Memory-mapped registers or host registers (I2C, SPI) are not reset by AVD. Fortemedia DMIC takes about 4096 cycles of PDM_CLKO0 after AVD is activated to send out the 1st valid data.

Firmware will start from the vector at reset. Firmware should do minimum initialization when it is resumed from AVD to be able to execute “wakeup” function right away.

PDM_CLKO0 is for always-on DMIC, and PDM_CLKO1 can be turned off.

5.3.2 Regular DMIC

Regular DMIC has no I2C and no AVD. Only PDM_CLKO0/PDM_CLKO1 and PDM_DATAI0/ PDM_DATAI1 are used between iM501 and microphone.

In normal operation, PDM_CLKO0/PDM_CLKO1 can be same rate as PDM_CLKI when bit [1:0] of 0x0FFF_FF16 are set to "00", or 2.048MHz when bit [1:0] are set to "01" during system.

In Power Saving mode, PDM_CLKO0 can provide 512K, 768K, or 1.024MHz by setting PLL to generate CODEC_CLK at 2.048M, 3.072M, or 4.096MHz.

For Low Power mode when DMIC supports 512K, 768K or 1.024MHz, bit [13:12] of 0x0FFF_FF22 must be set accordingly to get 16 KHz sampling rate of PCM to DSP.

For regular DMIC with no AVD pin, PLL and digital filter of ADC are not powered down.

5.3.3 PDM Data Output (CODEC/Host Interface)

iM501 can send out up to 4 channels of microphone signal to CODEC or host, through PDM_CLKI, PDM_DATAO0, and PDM_DATAO1 pins. These pins are shared with BCLK_TX, TX0, and TX1, respectively.

The range of PDM_CLKI is from 1.024MHz to 4.096MHz. Both PDM_DATAO0 and PDM_DATAO1 can be tri-stated.

5.3.4 Clock Detector

There is a built-in clock detector in iM501. When frequency PDM_CLKI or BCLK_RX is less than 8 KHz, iM501 automatically enters Power Saving mode.

5.3.5 Interrupt to Host

iM501 generates IRQ to host. The IRQ signal can be level or pulse: IRQ can be high to low or low to high when level is selected, or high-low-high or low-high-low transitions when pulse is selected. Host needs to clear IRQ when level trigger is selected. IRQ's polarity is also programmable.

5.4 SPI Interface

The iM501 provides high speed data transfer between host and iM501 with speed up to 20 Mbps. The transfer protocol follows the sequence of "command (1 byte), address (3 bytes), and data counter (2 bytes)" when IRAM or DRAM are read/written.

For SPI register access, the transfer protocol is:

- Memory: command (1 byte) → address (3 bytes) -> data counter (2 bytes) -> data (n)
- Register: command (1 byte) → address (1 bytes) -> data (1 byte)

The maximum block for data transfer is 64K word and minimum is 1 word.

iM501 provides two hosts, I2C or SPI, to internal DSP. Bit [2] of host register of I²C at 0x12 is used to select which one as host of DSP.

Bit [1:0] of host register of I²C at 0x12 are used to configure the SPI interface:

- CPOL, at bit[0] of 0x20, determines polarity of SPI_CLK
- CPHA is to select clock edge for latching or transmitting SPI_MISO/SPI_MOSI.

SPI module must be reset when bit [1:0] is changed.

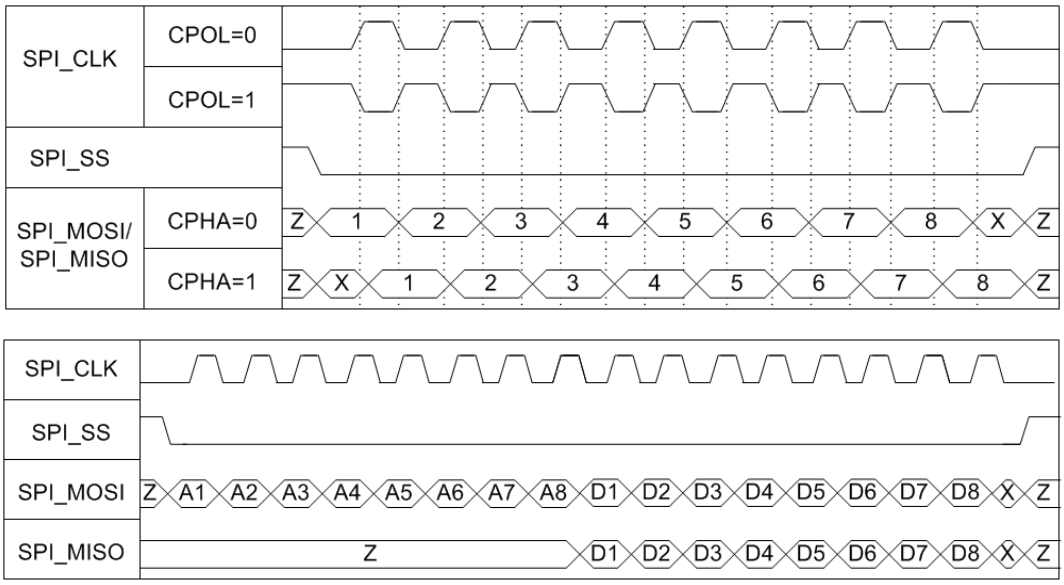

Figure 8. SPI Timing Sequences

5.4.1 SPI Protocol

COMMAND:

SPI supports read and write operations of IRAM and DRAM.

Table 5. Bit Definition for SPI Command Byte

	Bits	Reset	Description
Reserved	7:2		
Write	2	0	"0": read, "1": write
REG_SEL	1	0	"0": DSP memory, "1": SPI register
IRAM	0	0	"0": IRAM, "1": DRAM.

ADDRESS:

DRAM and IRAM have 3 address bytes, and SPI register has only 1 address byte. These 3 address bytes follows the command byte, with the least significant byte (LSB) at first and the most significant byte (MSB) at last. They define the first memory location to be read or written.

DATA Counter:

The 2-byte data counter specifies number of words to be read or written in IRAM or DRAM.

- The data counter must be multiple of 2 for IRAM (IRAM in Hi-Fi Mini is 32-bit access).
- The minimum number of memory access is 2 for IRAM and 1 for DRAM.
- The maximum transfer block is 128K byte for IRAM and DRAM.
- Don't need data counter if SPI register is read or written.

5.4.2 SPI Register Read/Write Sequence

Write SPI Register

To write an 8-bit data to SPI register at address ADD with data DAT, the sequence is:
0x06 (command) -> ADD (address) -> DAT (data)

For instance, the sequence to write 0x59 to SPI interrupt DSP register is:
0x06 (command) -> 0x01 (address) -> 0x59 (data)

Read SPI Register

To read an 8-bit data to SPI register at address ADD with data DAT, the sequence is:
0x02 (command) -> ADD (address) -> DAT (data)

For instance, the sequence to read SPI interrupt DSP register is:
0x02 (command) -> 0x01 (address) -> 0xXX (data)

5.4.3 SPI Burst Modes

iM501 provides SPI burst mode for both upload (Read) and download (Write) operations. Burst mode provide very efficient data transfer between host and IRAM/DRAM of iM501, for example, transferring 192 Kbytes data at SPI speed of 20Mbps only takes less than 100ms second.

To enable burst mode, the SPI host needs to set CPHA and CPOL bits at register 0x12 (Default of CPHA and CPOL are all "0"), and the Register 0x00 of SPI must be set to 0x05 for initial download.

5.4.4 SPI Burst Write

SPI Burst Write of IRAM

The sequence to issue a SPI burst write of 16-word of IRAM with contents as table below is:

ADDRESS of IRAM	Data of IRAM
0x1001_9803, 0x1001_9802, 0x1001_9801,0x1001_9800	0x04030201
0x1001_9807, 0x1001_9806, 0x1001_9805,0x1001_9804	0x08070605
0x1001_980B, 0x1001_980A, 0x1001_9809,0x1001_9808	0x0C0B0A09
0x1001_980F, 0x1001_980E, 0x1001_980D,0x1001_980C	0x100F0E0D
0x1001_9813, 0x1001_9812, 0x1001_9811,0x1001_9810	0x14131211
0x1001_9817, 0x1001_9816, 0x1001_9815,0x1001_9814	0x18171615
0x1001_981B, 0x1001_981A, 0x1001_9819,0x1001_9818	0x1C1B1A19
0x1001_981F, 0x1001_981E, 0x1001_981D,0x1001_981C	0x201F1E1D
0x1001_9823, 0x1001_9822, 0x1001_9821,0x1001_9820	0x24232221

04 (command) -> 00 (address, low) -> 98 (address, middle) -> 01 (address, high) -> 10 (low byte of word counter) -> 00 (high byte of word counter) -> 01 (data of 0x10019800) -> 02 (data of 0x10019801) -> … -> 1E (data of 0x1001981D) -> 1F (data of 0x1001981E) -> 20 (data of 0x1001981F) ->

SPI Burst Write of DRAM

The sequence to issue a SPI burst write of 16-word of DRAM with contents as table below is:

ADDRESS of DRAM	Data of DRAM
0xFCC0_9803, 0x0FFC_9802, 0x0FFC_9801,0x0FFC_9800	0x04030201
0xFCC0_9807, 0x0FFC_9806, 0x0FFC_9805,0x0FFC_9804	0x08070605
0xFCC0_980B, 0x0FFC_980A, 0x0FFC_9809,0x0FFC_9808	0x0C0B0A09
0xFCC0_980F, 0x0FFC_980E, 0x0FFC_980D,0x0FFC_980C	0x100F0E0D
0xFCC0_9813, 0x0FFC_9812, 0x0FFC_9811,0x0FFC_9810	0x14131211
0xFCC0_9817, 0x0FFC_9816, 0x0FFC_9815,0x0FFC_9814	0x18171615
0xFCC0_981B, 0x0FFC_981A, 0x0FFC_9819,0x0FFC_9818	0x1C1B1A19
0xFCC0_981F, 0x0FFC_981E, 0x0FFC_981D,0x0FFC_981C	0x201F1E1D

05 (command) -> 00 (address, low) -> 98 (address, middle) -> FC (address, high) -> 10 (low byte of word counter) -> 00 (high byte of word counter) -> 01 (data of 0x0FFC9800) -> 02 (data of 0x0FFC9801) -> … -> 1E (data of 0x0FFC981D) -> 1F (data of 0x0FFC981E) -> 20 (data of 0x0FFC981F) ->

5.4.5 SPI Burst Read

SPI burst read is implemented toward IRAM and DRAM only, but not for SPI registers.

SPI Burst Read of IRAM

The sequence to issue an I2C burst read of 16-word of IRAM from address 0x10019800 is: (Assuming IRAM address and contents are table below)

ADDRESS of IRAM	Data of IRAM
0x1001_9803, 0x1001_9802, 0x1001_9801,0x1001_9800	0x04030201
0x1001_9807, 0x1001_9806, 0x1001_9805,0x1001_9804	0x08070605
0x1001_980B, 0x1001_980A, 0x1001_9809,0x1001_9808	0x0C0B0A09
0x1001_980F, 0x1001_980E, 0x1001_980D,0x1001_980C	0x100F0E0D
0x1001_9813, 0x1001_9812, 0x1001_9811,0x1001_9810	0x14131211
0x1001_9817, 0x1001_9816, 0x1001_9815,0x1001_9814	0x18171615
0x1001_981B, 0x1001_981A, 0x1001_9819,0x1001_9818	0x1C1B1A19
0x1001_981F, 0x1001_981E, 0x1001_981D,0x1001_981C	0x201F1E1D
0x1001_9823, 0x1001_9822, 0x1001_9821,0x1001_9820	0x24232221

00 (command) -> 00 (address, low) -> 98 (address, middle) -> 01 (address, high) -> 10 (low byte of word counter) -> 00 (high byte of word counter) -> 01 (data of 0x10019800) -> 02 (data of 0x10019801) -> 03 (data of 0x10019802) -> … -> 1E (data of 0x1001981D) -> 1F (data of 0x1001981E) -> 20 (data of 0x1001981F) ->

SPI Burst Read of DRAM

The sequence to issue a SPI burst read of 16-word of DRAM with contents as table below is:

ADDRESS of DRAM	Data of DRAM
0xFCC0_9803, 0x0FFC_9802, 0x0FFC_9801,0x0FFC_9800	0x04030201
0xFCC0_9807, 0x0FFC_9806, 0x0FFC_9805,0x0FFC_9804	0x08070605
0xFCC0_980B, 0x0FFC_980A, 0x0FFC_9809,0x0FFC_9808	0x0C0B0A09
0xFCC0_980F, 0x0FFC_980E, 0x0FFC_980D,0x0FFC_980C	0x100F0E0D
0xFCC0_9813, 0x0FFC_9812, 0x0FFC_9811,0x0FFC_9810	0x14131211
0xFCC0_9817, 0x0FFC_9816, 0x0FFC_9815,0x0FFC_9814	0x18171615
0xFCC0_981B, 0x0FFC_981A, 0x0FFC_9819,0x0FFC_9818	0x1C1B1A19
0xFCC0_981F, 0x0FFC_981E, 0x0FFC_981D,0x0FFC_981C	0x201F1E1D

01 (command) -> 00 (address, low) -> 98 (address, middle) -> FC (address, high) -> 10 (low byte of word counter) -> 00 (high byte of word counter) -> 01 (data of 0x0FFC9800) -> 02 (data of 0x0FFC9801) -> … -> 1E (data of 0x0FFC981D) -> 1F (data of 0x0FFC981E) -> 20 (data of 0x0FFC981F) ->

5.5 LDO control

Given the feature of programmable LDO voltage and PLL, the iM501 provides a mechanism to adjust MIPS in terms of power consumption for various user applications. As shown in the following table, the iM501 can operate under distinct MIPS by setting the LDO register bits, LDO_CTL, in the bit [9:6] of 0x0FFF_FF22. Note that in order to control LDO, both PD_LDO in the bit 13 of 0x0FFF_FF1A and LDC_EN in the bit 10 of 0x0FFF_FF22 shall be set as 0. Also, an external cap at 2.2uF is required to add at the pin, VDDC when the targeting MIPS is higher than 120.

LDO_CTL hex value	LDO voltage*	Maximum MIPS*
4	1.10V	65
5	1.15V	80
6	1.20V	95
7	1.25V	110
8	1.30V	125
9	1.35V	140
A	1.40V	155

*Note: What defines in the table shall be referred to a typical value and subject to change upon chip variation.

6. Register Descriptions

6.1 I2C Registers

6.2 SPI Register

Host Address: 0x12
DSP I/O Address: N/A
Power-on Value: 0x00
Description: SPI setting

3	2	1	0

D3: SPI_RST
"1": reset SPI. This bit has to be written to "1", then to "0" whenever "CPHA" or "CPOL" are changed.
D2: I2C_mode
"0": Source of PIF master is from SPI. It means the memory down load and upload is through SPI interface.
"1": Source of PIF master is from I2C. It means the memory down load and upload is through I2C interface
D1: CPHA, or Clock phase of SPI.
D0: CPOL, or Clock polarity of SPI.

Note: 0x12 needs to be programmed to "0x0B", then 0x03 if system likes to set both "CPHA" and "CPOL" to "1"s.

6.3 DSP Register

6.3.1 TX of I2S configuration (0x0FFF_FF76)

	Bits	Read/Write	Reset	Description
Reserved	15:14			
TX_tri	13	Read/Write	0	"0": TX pin is driven, "1": TX pin is tri-stated.
Ocd_TX	12	Read/Write	0	"0": No delay. "1": one cycle delay.
R_latch_TX	11	Read/write	0	"0": Data is latched by falling edge of BCLK. "1": Data is latched by rising edge of BCLK.
Low_F_TX	10	Read/Write	0	"0": Frame starts at "high", "1": Frame starts at "low".
LSB_F_TX	9	Read/Write	0	"1": LSB first. "0": MSB first.
R_align_TX	8	Read/Write	0	"1": Right alignment, "0": Left alignment.
Word_lenth_TX	7:6	Read/Write	00	Length of word. "00": 16-bit, "01": 20-bit, "10":24-bit.
slot_lenth_TX	5:3	Read/write	000	**Length of slot. "011": 16-bit, "100": 20-bit,"101":24-bit,** **"110": 28-bit, "111": 32-bit**
Slot_num_TX	2:0	Read/Write	000	TX slot number. Number of slots = Slot_num[2:0] + 1;

6.3.2 RX of I2S configuration (0x0FFF_FF78)

	Bits	Read/Write	Reset	Description

Reserved	15:13			
Ocd_RX	12	Read/Write	0	"0": No delay. "1": one cycle delay.
R_latch_RX	11	Read/write	0	"0": Data is latched by falling edge of BCLK. "1": Data is latched by rising edge of BCLK.
Low_F_RX	10	Read/Write	0	"0": Frame starts at "high", "1": Frame starts at "low".
LSB_F	9	Read/Write	0	"1": LSB first. "0": MSB first.
R_align	8	Read/Write	0	"1": Right alignment, "0": Left alignment.
Word_lenth	7:6	Read/Write	00	Length of word. "00": 16-bit, "01": 20-bit, "10":24-bit.
slot_lenth_RX	5:3	Read/write	000	**Length of slot. "011": 16-bit, "100": 20-bit,"101":24-bit,** **"110": 28-bit, "111": 32-bit**
Slot_num_RX	2:0	Read/Write	000	RX slot number. Number of slots = Slot_num[2:0] + 1;

7. Voice Interface Device Functionalities

7.1 Software Components

The iM501 is a Voice Interface Device that integrates programmable feature sets including:

- VOS (Voice Operating System) for always-on ambience monitoring
- Supports programmable speech detection and voice triggers
- Noise suppression and acoustic echo cancellation
- Acoustic ambience analysis

In the following sections we shall describe each function briefly:

7.1.1 Voice Operating System

The iM501 design targets hosting of advanced voice processing technologies developed by Fortemedia, as well as easy integration of Third Party voice processing technologies - as firmware – onto the processor.

The runtime operating framework aims at providing flexibility and facilitates easy third party firmware integration. It manages the voice data I/O and speech history buffering, and prioritizes task executions. The operating environment is "voice-conscious" in a sense that it minimizes the delay incurred by buffering, yet maintains voice data history intelligently for performing ambience audio scene analysis.

7.1.2 Voice Trigger Function

The Voice-Trigger is for system wakeup or for signaling to a voice recognition engine the beginning of a session of man-machine dialogues. The iM501 system firmware allows for easy drop-in and integration of any Third-Party voice trigger and wake-up algorithms. After the iM501 reports the voice trigger event to the host processor, the host processor can retrieve the stored voice history for further voice analysis and recognition processing.

7.1.3 Voice Enhancement Function

The iM501 contains Fortemedia Advanced Microphone Array Processing (AMAP) algorithm for enhancing the incoming voice signal. The enhanced voice can benefit Voice Recognition functions to achieve higher recognition rates when man-machine interaction mode is on, and it will provide better voice quality in human-to-human conversation mode ambient noise and acoustic echoes eliminated.

The AMAP software components built into iM501 includes acoustic echo cancellation, multi-microphone beamforming, noise suppression, and intelligent voice enhancements.

7.1.4 Acoustic Ambience Analysis Function

The iM501 contains the Fortemedia Acoustic Ambience Analysis algorithm that analyzes the ambient sound signal in the voice buffer and reports higher level cues to upstream for sensor fusion or other applications.

8. Electrical Characteristics

8.1 7.1 DC Characteristics

Table 6. Absolute Maximum Ratings

Parameter	Symbol		Minimum	Typical	Maximum	Units
Power supply voltage	VDDIO		1.6		3.6	V
Ambient Operating Temperature	Ta		-20	-	+85	°C
Storage Temperature	Ts		-40		+125	°C
HBM ESD (Electrostatic Discharge)						
		Susceptibility Voltage				
All pins		2000				
MM ESD (Electrostatic Discharge)						
		Susceptibility Voltage				
All pins		200				
CDM ESD (Electrostatic Discharge)						
		Susceptibility Voltage				
All pins		500				

Table 7. Static Characteristics

Parameter	Symbol	For 1.8V I/O			For 2.5V I/O			For 3.3V I/O			Unit
		Min.	Typ.	Max.	Min.	Typ.	Max.	Min.	Typ.	Max.	
IO power supply	VDDIO	1.62	1.8	1.98	2.25	2.5	2.75	2.97	3.3	3.63	V
Hysteresis	-	0.17		0.3	0.2		0.3	0.22		0.3	V
Input low voltage	VIL			0.6			0.7			0.8	V
Input high Voltage	VIH	1.2			1.7			2.0			V
Output low voltage	VOL			0.4			0.4			0.4	V
Output high voltage	VOH	VDDIO-0.4			VDDIO -0.4			VDDIO -0.4			V
Input leakage current	ILI	-10		+10	-10		+10	-10		+10	uA
Output leakage current	ILO	-10		+10	-10		+10	-10		+10	uA
Output High current	Ioh		2.4			3.8			5		mA
Output low current	Iol		2.4			3.8			5		mA
Pull-up resistor	Pull-up		90			57			42		kΩ
Pull-down resistor	Pull-down		97			59			44		kΩ

8.2 AC Timing Characteristics

8.2.1 I²C Control Interface

Table 8. I²C Timing Parameters

Parameter	Symbol	Minimum	Typical	Maximum	Units
Clock Low pulse duration	$T_{w(9)}$	480	-	-	ns
Clock High pulse duration	$T_{w(10)}$	450	-	-	ns
Clock Frequency	f	100	-	400	KHz
Re-Start Setup Time	$T_{su(6)}$	260	-	-	ns
Start Hold Time	$T_{h(5)}$	260	-	-	ns
Data Setup Time	$T_{su(7)}$	50	-	-	ns
Data Hold Time	$T_{h(6)}$	0	-		ns
Rising Time	T_r	-	-	60	ns
Falling Time	T_f	-	-	60	ns
Stop Setup Time	$T_{su(8)}$	240	-	-	ns
Bus free time between a STOP and START condition	T_{buf}	500	-	-	ns
Pulse Width of Spikes Suppressed Input Filter	T_{sp}	10	-	-	ns

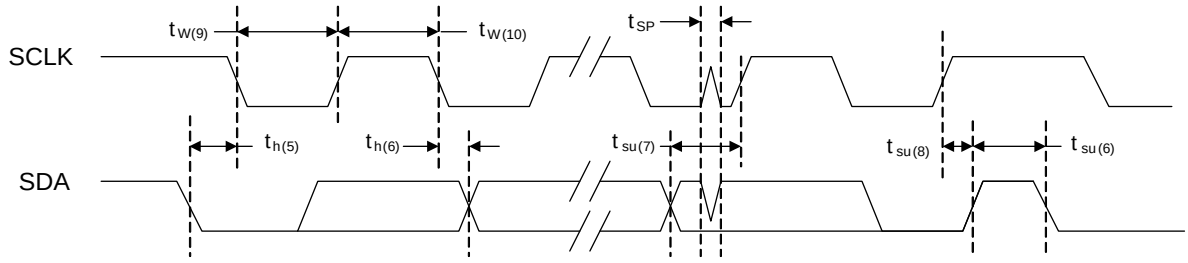

Figure 9. I2C Timing Diagram

8.2.2 PDM Control Interface

Table 9. Digital Microphone Interface Timing

Parameter	Symbol	MIN	TYP	MAX	UNIT
PDM_CLKOn cycle time	TCY_M	1.024	2.048	4.096	Mhz
PDM_CLKOn rise/fall time	TR_M, TF_M	5		30	ns
PDM_CLKOn duty cycle		45		55	%
PDM_DATIn (left) setup time	TLSU_M	15			ns
PDM_DATIn (left) hold time	TLH_M	10			ns
PDM_DATIn (right) setup time	TRSU_M	15			ns
PDM_DATIn (right) hold time	TRH_M	10			ns

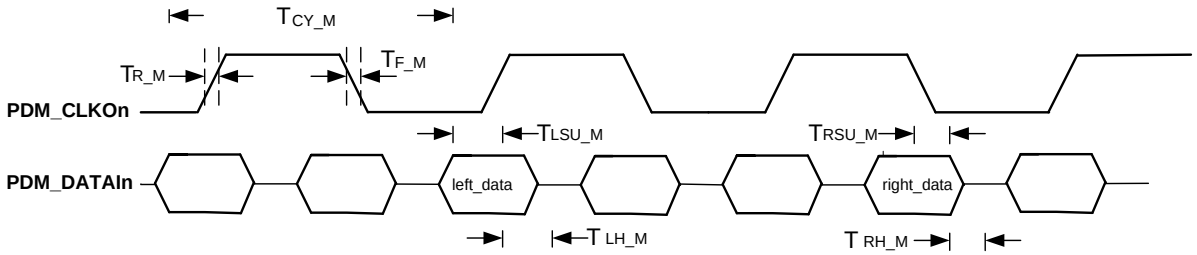

Figure 10. PDM Timing Diagram

Table 10. PDM (CODEC) Interface Timing

Parameter	Symbol	MIN	TYP	MAX	UNIT
PDM_CLKIn cycle time	T_{CY_C}	1.024	2.048	4.096	Mhz
PDM_CLKIn rise/fall time	T_{R_C}, T_{F_C}		10		ns
PDM_CLKIn duty cycle		45		55	%
PDM_DATOn (left) setup time	T_{LSU_C}	30			ns
PDM_DATOn (left) hold time	T_{LH_C}	15			ns
PDM_DATOn (right) setup time	T_{RSU_C}	30			ns
PDM_DATOn (right) hold time	T_{RH_C}	15			ns
PDM_DATOn data valid from rising edge of PDM_CLKIn	T_{REN}		30		ns
PDM_DATOn data valid from rising edge of PDM_CLKIn	T_{LEN}		30		ns

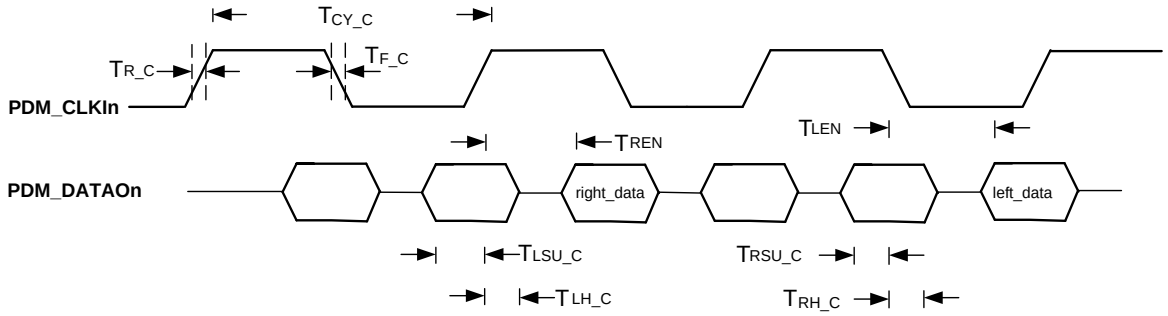

Figure 11. PDM (CODEC) Timing Diagram

8.2.3 SPI Control Interface

Table 11. SPI Timing Parameters

Parameter	Symbol	MIN	TYP	MAX	UNIT
SPI_SS_ falling edge to SPI_CLK rising edge	t_{SSU}	2			ns
SPI_CLK falling edge to SPI_SS_ rising edge	t_{SHO}	0			ns
SPI_CLK cycle time	t_{SCY}	50			ns
SPI_CLK pulse width low	t_{SCL}	10			ns
SPI_CLK pulse width high	t_{SCH}	10			ns
SPI_MOSI to SPI_CLK set-up time	t_{DSU}	3			ns
SPI_MOSI to SPI_CLK hold time	t_{DHO}	3			ns
SPI_CLK falling edge to SPI_MISO transition	t_{DL}			9	ns

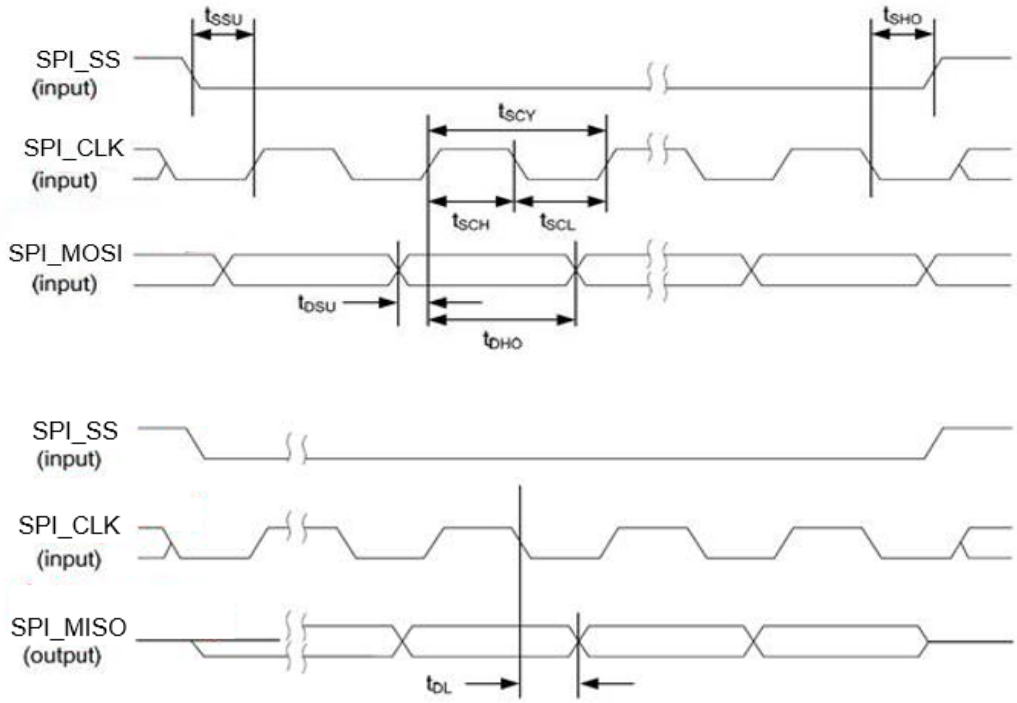

Figure 12. SPI Timing Diagrams

8.2.4 Power Up Sequence

Table 12. Power-up Sequence Timing Parameters

Parameter	Symbol	Minimum	Typical	Maximum	Units
VDDIO to VDDC delay time	T_{d_vio2vc}		200	-	µs
VDDC to internal power on reset	$T_{vddc2por\_}$		1024	-	cycle
PDM_CLKI off to power saving mode activated	Td_ckof2psm		130		us
PDM_CLKI on to resume normal mode	$T_{ckon2psm}$		8	-	cycle
LDO settle time from low to high	T_{ldos_l2h}		200	-	us
LDO settle time from high to low	T_{ldos_h2l}		2	-	ms
I2C ready after PDM_CLKI on	$T_{clk2I2C}$	-	3		ms
PLL lock in time		-	3		ms

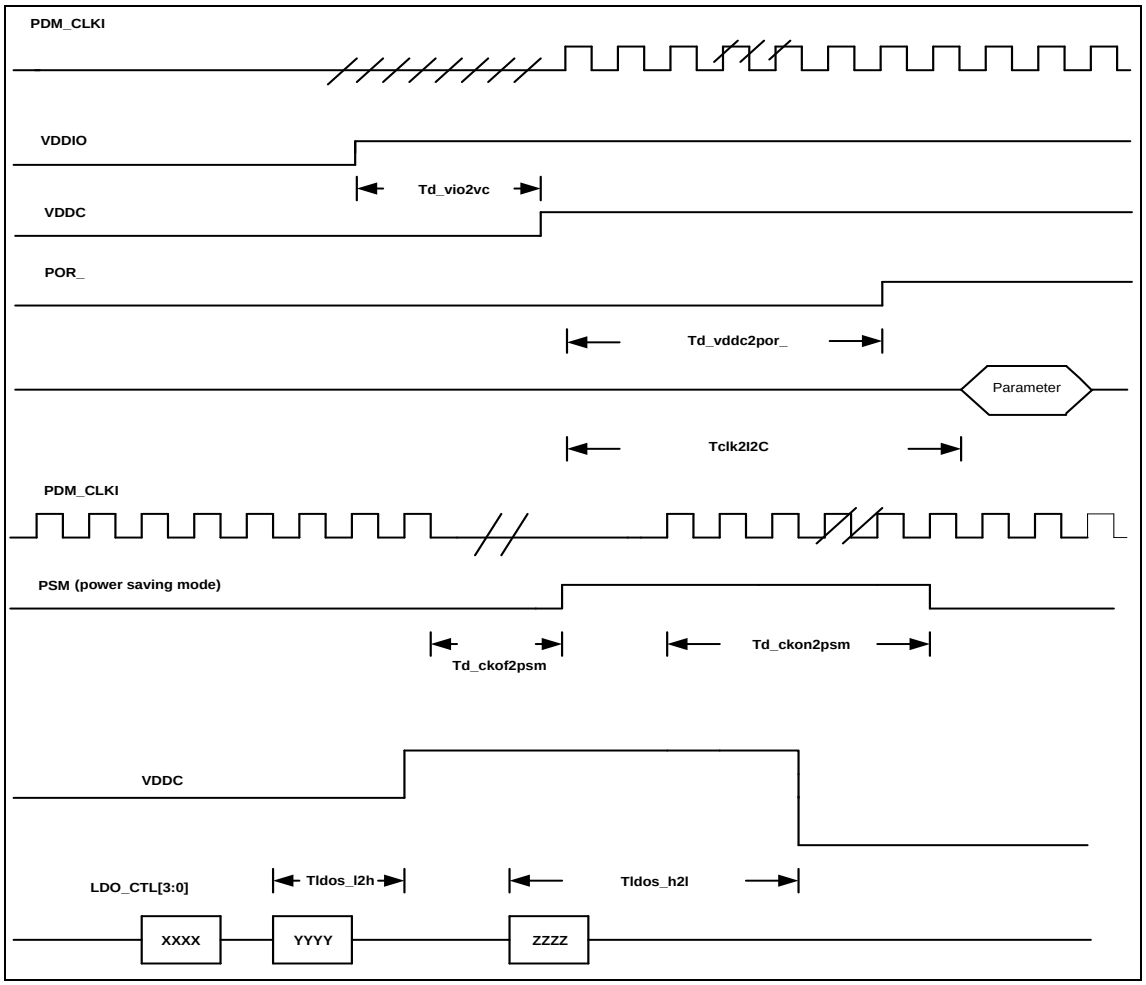

Figure 13. Power-up Sequence Timing Diagram

9. Circuit for iM501 Application Example

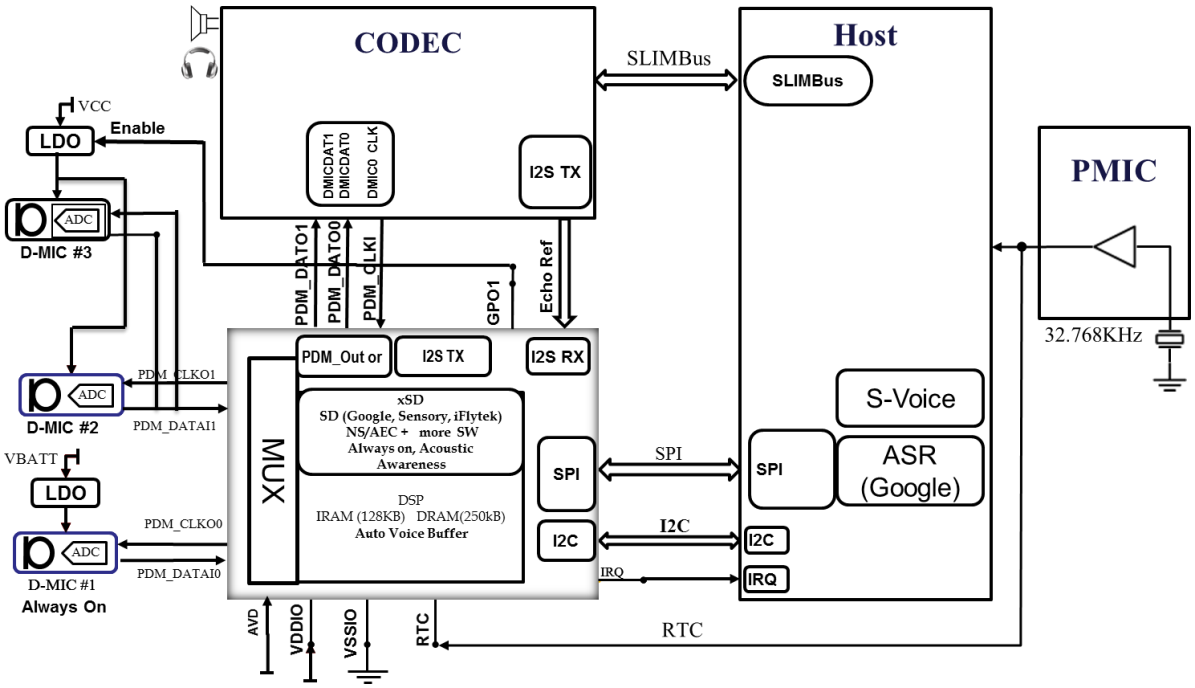

Figure 14. iM501 uses 3 DMICs for Mobile Device Wake-up and (AEC/NS) via PDM

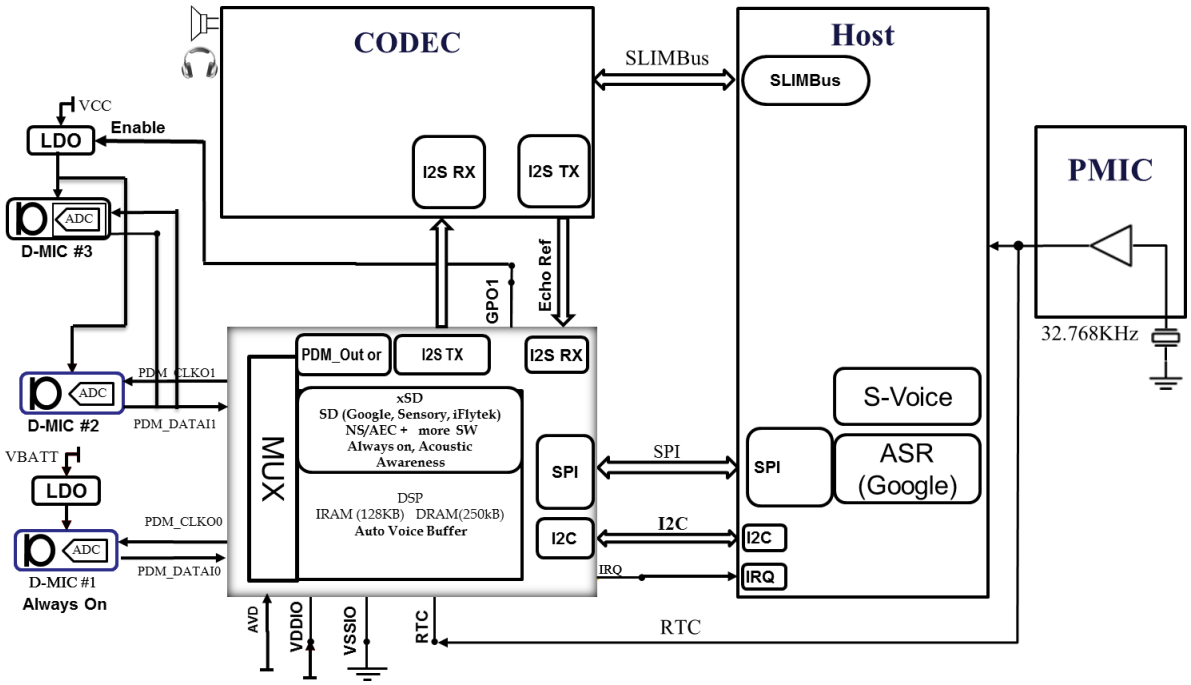

Figure 15. iM501 uses 3 DMICs for Mobile Device Wake-up and (AEC/NS) via I2S

10.Mechanical Dimensions

10.1 QFN Package (QFN32, 5mmx5mm):

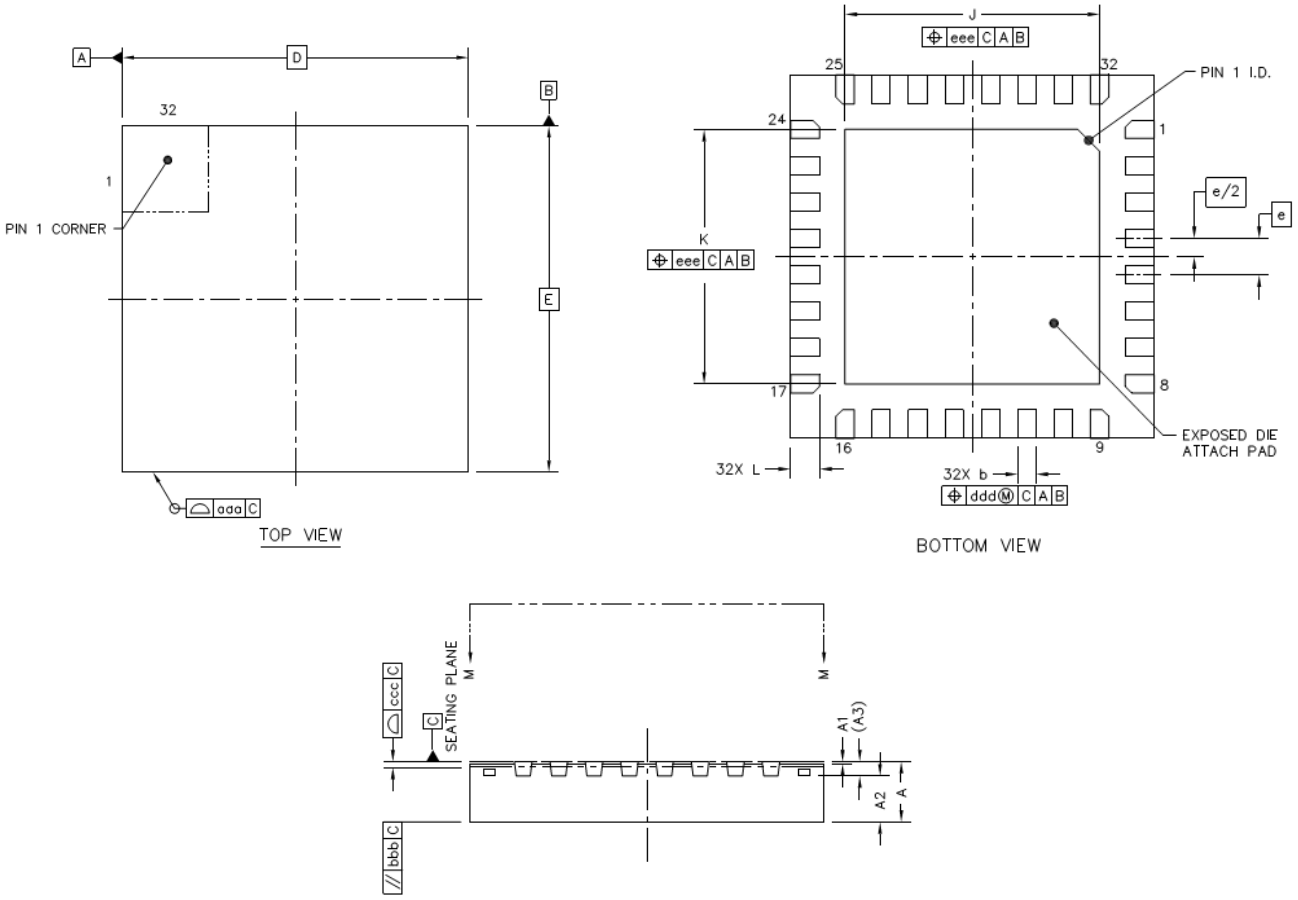

Figure 16. Top, Bottom and Side View of iM501 QFN Package

Table 13. Detailed Dimensions for iM501 in QFN Package

		Symbol	Dimension in millimeter (mm)		
			Min	Nom	Max
Total Thickness		A	0.8	0.85	0.9
Stand Off		A_1	0	0.035	0.05
Mold Thickness		A_2	---	0.65	0.67
L/F Thickness		A_3	0.203 REF		
Lead Width		b	0.2	0.25	0.3
Body Size	X	D	5 BSC		
	Y	E	5 BSC		
Lead Pitch		e	0.5 BSC		
EP Size	X	J	3.4	3.5	3.6
	Y	K	3.4	3.5	3.6
Lead Length		L	0.35	0.4	0.45
Package Edge Tolerance		aaa	0.1		
Mold Flatness		bbb	0.1		
Coplanarity		ccc	0.08		
Lead Offset		ddd	0.1		
Exposed PAD Offset		eee	0.1		

10.2 CSP (4x7 Ball 0.4mm Pitch):

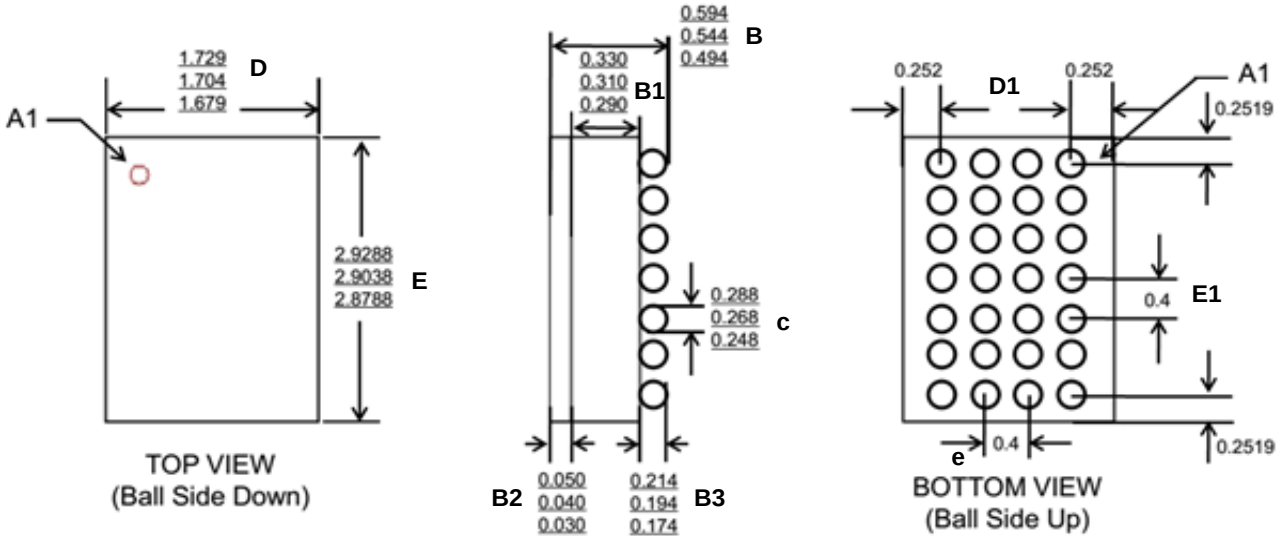

Figure 17. Top, Side and Bottom View of iM501 CSP Package

Table 14. Detailed Dimensions for iM501 in CSP Package

		Symbol	Dimension in millimeter (mm)		
			Min	Nom	Max
Total Thickness		B	0.494	0.544	0.594
Wafer Thickness		B_1	0.290	0.310	0.330
Stand Off		B_3	0.174	0.194	0.214
Film Thickness		B_2	0.030	0.040	0.050
Body Size	X	D	1.679	1.704	1.729
	Y	E	2.8788	2.9038	2.9288
Edge Ball Center to Center	X	D_1	1.175	1.2	1.225
	Y	E_1	2.375	2.4	2.425
Ball/Bump Pitch		e	0.4		
Ball Diameter (Size)		c	0.248	0.268	0.288
Ball/Bump Count			28		

11.Order Information

Table 15. Ordering Information

Part Number	Package	Status
iM501NE	QFN-32 (5mmx5mm)	Available
iM501CE	CSP 4x7 ball 0.4mm pitch	Available

Note: See page-9 for package and version identification.

12. Appendix-A: Example Code for iM501's I²C Communication

This chapter presents several pre-defined communication methods between iM501 and optional external components or other hosts.

12.1 Brief of iM501's I²C Protocol

I²C serial interface uses command protocol to communicate between host and iM501. There are two types of commands: address/data, and data-only. Each transfer will have one command byte, one or three address bytes, and up to four data bytes. If it's a data-only transfer, there is one command byte and up to 64k bytes of data, when byte counter is programmed to 'FFFF'. There is a command parser in iM501, which will interpret the command and access registers or memory as commanded.

The sequence of address and data is lowest byte first and highest byte last:

Command -> address_0 (lowest byte) -> address_1 (middle byte) -> address_2 (highest byte) -> Data_0 (lowest byte) -> Data_1 (2nd lowest byte) -> data_2 (2nd highest byte) -> data_3 (highest byte)

12.2 Example: writes 1 byte to internal register from I2C interface

```
Start;
ID W;                   -- E2
Write_byte      0x48    -- command byte, writes to internal register with 1 address bytes,
                           1 data bytes and indicates a write cycle
Write_byte      0x0E    -- address byte of internal register
Write_byte      0x0B    -- data byte of internal register
Stop;
(Write 0x0B to register 0x0E)
```

12.3 Example: writes 2 byte to internal register from I2C interface

```
Start;
ID W;                   --E2
Write_byte      0x4A    -- command byte, writes to internal register,
                           1 address byte, 2 data bytes, write cycle
Write_byte      0x08    -- address byte of internal register – byte counter
Write_byte      0x20    -- LSB of internal register
Write_byte      0x00    -- MSB of internal register
Stop;
(Write 0x0020 to register 0x08)
```

12.4 Example: reads 1 byte to internal register from I2C interface

```
Start;
ID W;
Write_byte      0x46    -- command byte, read internal register with 1 address byte,
                           no data byte and indicates a read cycle
Write_byte      0x0E    -- address byte of internal register
Stop;
Start;
ID R;                   -- E3
Read_byte
Stop;
```

(Internal register can only be read by 1 byte at a time)

12.5 Eexample: writes IRAM from I2C interface

(IRAM rage is from 0x1000_0000 to 0x1001_FFFF) ---Write 0xEFCDAB89 to 0x10013F80

```
Start;
ID;                 -- E2
Write_byte   0x0D   -- command byte, writes to IRAM with 3 address bytes,
                       4 data bytes and indicates a write cycle
Write_byte   0x80   -- IRAM lowest address byte
Write_byte   0x3F   -- IRAM 2nd lowest address byte
Write_byte   0x01   -- IRAM 2nd highest address byte
Write_byte   0x89   -- IRAM data, LSB
Write_byte   0xAB   -- IRAM data, second byte
Write_byte   0xCD   -- IRAM data, third byte
Write_byte   0xEF   -- IRAM data, fourth byte
```

ADDRESS of IRAM	Data of IRAM
0x1001_3F83, 0x1001_3F82, 0x1001_3F82, 0x1001_3F80	0xEFCDAB89

12.6 Example: reads IRAM from I2C interface

```
Start;
ID W;               -- E2
Write_byte   0x07   -- command byte, read IRAM with 3 address bytes, no data bytes
Write_byte   0x80   -- IRAM lowest address byte
Write_byte   0x3F   -- IRAM 2nd lowest address byte
Write_byte   0x01   -- IRAM 2nd highest address byte
Stop;
Start;
ID W;               -- E2
Write_byte   0x46   -- command byte, read internal with one address byte
Write_byte   0x0D   -- address, IDMA data register
Stop;
Start;
ID R;               --E3
Read_byte
Stop;
Start;
ID W;               -- E2
Write_byte   0x46   -- command byte, read internal with one address byte
Write_byte   0x0C   -- address, IDMA data register
Stop;
Start;
ID R;               -- E3
Read_byte
Stop;
Start;
ID W;               -- E2
Write_byte   0x46   -- command byte, read internal with one address byte
Write_byte   0x0B   -- address, IDMA data register
Stop;
Start;
ID R;               -- E3
Read_byte
Stop;
```

```
Start;
ID W;                   -- E2
Write_byte      0x46    -- command byte, read internal with one address byte
Write_byte      0x0A    -- address, IDMA data register
Stop;
Start;
ID R;                   -- E3
Read_byte
Stop;
```

12.7 Example: writes DRAM from I2C interface

```
Start;
ID W;                    -- E2
Write_byte      0x2B    --Command byte, write DRAM with 3 address bytes and 2 data bytes.
Write_byte      0x00     -- LSB address byte
Write_byte      0x00    --MSB address byte
Write_byte      0xFE    --Third address byte
Write_byte      0xCD    --LSB data byte
Write_byte      0xAB    --MSB data byte
Stop;
```

ADDRESS of DRAM	Data of DRAM
0x0FFE_0001,0x0FFE_0000	0xABCD

12.8 Example: reads DRAM from I2C interface

```
Start;
ID W;                   --E2
Write_byte      0x27    -- command byte, read IRAM with 3 address bytes, no data bytes
Write_byte      0x00    -- DRAM lowest address byte
Write_byte      0x00    -- DRAM 2nd lowest address byte
Write_byte      0xFE    -- DRAM 2nd highest address byte
Stop;
Start;
ID W;                   --E2
Write_byte      0x46    -- command byte, read internal with one address byte
Write_byte      0x0D    -- address, IDMA data register
Stop;
Start;
ID R;                   --E3
Read_byte
Stop;
Start;
ID W;                   --E2
Write_byte      0x46    -- command byte, read internal with one address byte
Write_byte      0x0C    -- address, IDMA data register
Stop;
Start;
ID R;                   --E3
Read_byte
Stop;
```

12.9 Example: download 32-byte data to IRAM using burst mode.

```
Start;
ID W;                          --E2
 Write_byte        0x4A        -- Command byte,
                                  write I2C host register with one address byte and two data bytes
Write_byte         0x08        -- address, byte counter
Write_byte         0x20        -- data,  LSB of byte counter
Write_byte         0x00        -- data, MSB of byte counter
Stop;
Start;
ID W;                           --E2
Write_byte         0x0D         --Command byte, write IRAM with 3 address bytes and four data bytes.
Write_byte         0x00         -- Lowest address byte
Write_byte         0x00         -- 2nd lowest address byte
Write_byte         0x00         -- 3rd lowest address byte
Write_byte         0XAB         -- lowest data byte
Write_byte         0xCD         -- 2nd lowest data byte
Write_byte         0xEF         -- 3rd  data byte
Write_byte         0x01         -- 4th  data byte
Stop;
```

ADDRESS of IRAM	Data of IRAM
0x1000_0003, 0x1000_0002, 0x1000_0001,0x1000_0000	0x01EFCDAB

```
Start;
ID W;                          --E2
Write_byte         0x88         --Command, data only, write IRAM
Stop;
Start;
ID W;                          --E2
Write_byte         0x00         --data byte
Write_byte         0x01         --data byte
Write_byte         0x02         --data byte
Write_byte         0x03         --data byte
Write_byte         0x04         --data byte
Write_byte         0x05         --data byte
Write_byte         0x06         --data byte
Write_byte         0x07         --data byte
Write_byte         0x08         --data byte
Write_byte         0x09         --data byte
Write_byte         0x0A          --data byte
Write_byte         0x0B         --data byte
Write_byte         0x0C          --data byte
Write_byte         0x0D          --data byte
Write_byte         0x0E         --data byte
Write_byte         0x0F         --data byte
Write_byte         0x11         --data byte
Write_byte         0x12         --data byte
Write_byte         0x13         --data byte
Write_byte         0x14         --data byte
Write_byte         0x15         --data byte
Write_byte         0x16         --data byte
Write_byte         0x17         --data byte
Write_byte         0x18         --data byte
Write_byte         0x19         --data byte
Write_byte         0x1A          --data byte
Write_byte         0x1B         --data byte
```

```
Write_byte      0x1C    --data byte
Write_byte      0x1D    --data byte
Write_byte      0x1E    --data byte
Write_byte      0x1F    --data byte
Stop;
```

ADDRESS of IRAM	Data of IRAM
0x1000_0003, 0x1000_0002, 0x1000_0001,0x1000_0000	0x01EFCDAB
0x1000_0007, 0x1000_0006, 0x1000_0005,0x1000_0004	0x03020100
0x1000_000B, 0x1000_000A, 0x1000_0009,0x1000_0008	0x07060504
0x1000_000F, 0x1000_000E, 0x1000_000D,0x1000_000C	0x0B0A0908
0x1000_0013, 0x1000_0012, 0x1000_0011,0x1000_0010	0x0F0E0D0C
0x1000_0017, 0x1000_0016, 0x1000_0015,0x1000_0014	0x13121110
0x1000_001B, 0x1000_001A, 0x1000_0019,0x1000_0018	0x17161514
0x1000_001F, 0x1000_001E, 0x1000_001D,0x1000_001C	0x1B1A1918
0x1000_0023, 0x1000_0022, 0x1000_0021,0x1000_0020	0x1F1E1D1C

12.10 Example: download 32-byte data to DRAM using burst mode

```
Start;
ID W;                   -- E2
Write_byte      0x4A    -- Command byte, write I2S host register with one address byte and two data
bytes
Write_byte      0x08    -- address, byte counter
Write_byte      0x20    --data, LSB of byte counter
Write_byte      0x00    --data, MSB of byte counter.
Stop;
Start;
ID W;                   -- E2
Write_byte      0x2B    --Command byte, write DRAM with 3 address bytes and 2 data bytes.
                        (Must be two byte data)
Write_byte      0x00    -- lowest address byte
Write_byte      0x00    -- 2nd lowest address byte
Write_byte      0xFE    -- 3rd lowest address byte
Write_byte      0xAB    --lower data byte
Write_byte      0xCD    --higher data byte
Stop;
```

ADDRESS of DRAM	Data of DRAM
0x0FFE_0001,0x0FFE_0000	0xCDAB

```
Start;
ID W;                   --E2
Write_byte      0xA8    --Command, data only, write DRAM
Stop;
Start;
ID W;                   -- E2
Write_byte      0x00    --data byte
Write_byte      0x01    --data byte
Write_byte      0x02    --data byte
Write_byte      0x03    --data byte
Write_byte      0x04    --data byte
Write_byte      0x05    --data byte
Write_byte      0x06    --data byte
Write_byte      0x07    --data byte
Write_byte      0x08    --data byte
```

```
Write_byte          0x09    --data byte
Write_byte          0x0A    --data byte
Write_byte          0x0B    --data byte
Write_byte          0x0C    --data byte
Write_byte          0x0D    --data byte
Write_byte          0x0E    --data byte
Write_byte          0x0F    --data byte
Write_byte          0x11    --data byte
Write_byte          0x12    --data byte
Write_byte          0x13    --data byte
Write_byte          0x14    --data byte
Write_byte          0x15    --data byte
Write_byte          0x16    --data byte
Write_byte          0x17    --data byte
Write_byte          0x18    --data byte
Write_byte          0x19    --data byte
Write_byte          0x1A    --data byte
Write_byte          0x1B    --data byte
Write_byte          0x1C    --data byte
Write_byte          0x1D    --data byte
Write_byte          0x1E    --data byte
Write_byte          0x1F    --data byte
Stop;
```

ADDRESS of DRAM	Data of DRAM
0x0FFE_0001,0x0FFE_0000	0xABCD
0x0FFE_0003,0x0FFE_0002	0x0100
0x0FFE_0005,0x0FFE_0004	0x0302
0x0FFE_0007,0x0FFE_0006	0x0504
0x0FFE_0009,0x0FFE_0008	0x0706
0x0FFE_000B,0x0FFE_000A	0x0908
0x0FFE_000D,0x0FFE_000C	0x0B0A
0x0FFE_000F,0x0FFE_000E	0x0D0C
0x0FFE_0011,0x0FFE_0010	0x0F0E
0x0FFE_0013,0x0FFE_0012	0x1110
0x0FFE_0015,0x0FFE_0014	0x1312
0x0FFE_0017,0x0FFE_0016	0x1514
0x0FFE_0019,0x0FFE_0018	0x1716
0x0FFE_001B,0x0FFE_001A	0x1918
0x0FFE_001D,0x0FFE_001C	0x1B1A
0x0FFE_001F,0x0FFE_001E	0x1D1C
0x0FFE_0021,0x0FFE_0020	0x1F1E

12.11 Example: read voice command using burst read mode.

```
(Read 32-byte of DRAM from 0x0FFE_0000 to 0x0FFE_001F)
Start;
ID W;                       -- E2
Write_byte      0x4A        -- Command byte, write I2C host register, one address byte, two data bytes
Write_byte      0x08        -- address, byte counter
Write_byte      0x20        -- data, LSB of byte counter
Write_byte      0x00        -- data, MSB of byte counter.
Stop;
Start;
ID W;                       -- E2
```

```
Write_byte      0x27      -- Command byte, read DRAM with 3-address byte
Write_byte      0xFE      -- address_3, 0xFE
Write_byte      0x00      -- address_2, 0x00
Write_byte      0x00      -- address_1, 0x00
Stop;
Start;
ID W;
Write_byte      0xA0      -- Command byte, “data only”, “DRAM” and “read” are set.
Stop;

Start;
ID R;                     --E3
read data1;               --0xCD (0x0FFE_0000)
read data2;               --0xAB (0x0FFE_0001)
read data3;               --0x00 (0x0FFE_0002)
read data4;               --0x01 (0x0FFE_0003)
read data5;               --0x02 (0x0FFE_0004)
read data6;               --0x03 (0x0FFE_0005)
read data7;               --0x04 (0x0FFE_0006)
read data8;               --0x05 (0x0FFE_0007)
read data9;               --0x06 (0x0FFE_0008)
read data10;              --0x07 (0x0FFE_0009)
read data11;              --0x08 (0x0FFE_000A)
read data12;              --0x09 (0x0FFE_000B)
read data13;              --0x0A (0x0FFE_000C)
read data14;              --0x0B (0x0FFE_000D)
read data15;              --0x0C (0x0FFE_000E)
read data16;              --0x0D (0x0FFE_000F)
read data17;              --0x0E (0x0FFE_0010)
read data18;              --0x0F (0x0FFE_0011)
read data19;              --0x10 (0x0FFE_0012)
read data20;              --0x11 (0x0FFE_0013)
read data21;              --0x12 (0x0FFE_0014)
read data22;              --0x13 (0x0FFE_0015)
read data23;              --0x14 (0x0FFE_0016)
read data24;              --0x15 (0x0FFE_0017)
read data25;              --0x16 (0x0FFE_0018)
read data26;              --0x17 (0x0FFE_0019)
read data27;              --0x18 (0x0FFE_001A)
read data28;              --0x19 (0x0FFE_001B)
read data29;              --0x1A (0x0FFE_001C)
read data30;              --0x1B (0x0FFE_001D)
read data31;              --0x1C (0x0FFE_001E)
read data32;              --0x1D (0x0FFE_001F)
Stop;
```