MSM6290™ Mobile Station Modem™

User Guide

80-VF866-3 Rev. C

March 19, 2008

Submit technical questions at:
https://support.cdmatech.com

Qualcomm Confidential and Proprietary

QUALCOMM Incorporated
5775 Morehouse Drive
San Diego, CA 92121-1714
U.S.A.

Terms and Conditions of Usage

This document contains descriptions of parts whose manufacture, use, sale, offer for sale, or importation into the United States is limited or prohibited by the February 5th, 2008 injunction against QUALCOMM Incorporated. This document should not be construed as an offer to sell such parts for use or importation into the U.S., nor should it be construed as assistance in making, using, selling, offering to sell, or the importation of any product in the U.S. containing such parts. This document is intended solely to provide information for those products that are outside the scope of the injunction. Recipient's download and/or use of the information in this document constitutes agreement with these terms.

Contents

10 JTAG Interface, Mode Select, and Emulation Considerations

11 Transport Stream Interface for DMB

Figures

Tables

Revision history

Revision	Date	Description
A	September 2007	Initial release
B	December 2007	■ Updated Section 1.2 to remove platform information and replace MediaFLO with UBM ■ Removed reference to Qualcomm providing AMSS software ■ Updated Section 1.3.3 with HSUPA support information ■ Removed the PM6658 power management IC and the PM6653 power management IC from the bulleted list at the beginning of Chapter 5 ■ Corrected the pin numbers in Figure 5-2 ■ Removed transceiver information under Section 7.3.1 ■ Updated Figure 7-2 ■ Updated Table 7-1 ■ Updated Figure 7-4, Figure 7-9, and Figure 7-10 ■ Removed the diagnostic (peripheral) information from Section 7.3.2 ■ Added information about USIM pins in Section 7.3.2.1 ■ Removed a note and Section 7.3.2.2, Peripheral application ■ Updated the DVB-H standard information in Figure 11-1 ■ Updated Section 11.4.1 and Section 11.4.2 to include information about an additional required tick ■ Updated the entire Section 11.4.3

Revision	Date	Description
C	March 2008	■ Updated MSM6290 device description in Section 1.2: Revised echo canceller information ■ Updated Section 1.3.1: Removed reference to 225 MHz and 3.6 Mbps ■ Updated Section 1.3.3: Revised 5.76 Mbps value to 4.5 Mbps ■ Updated Section 1.3.8: Added processing features ■ Updated Section 1.3.9: Revised bus clock speeds ■ Updated the Video telephony services: Qvideophone™ section: Added upscaling ■ Updated the Qcamcorder™ section: Removed H.264 ■ Updated Table 1-1: Revised ADSP and MDSP to 122.88 MHz ■ Updated the UART section: Added information on UART maximum throughput ■ Updated the USB section: Removed “diagnostic for USB“ and added information regarding HS USB controller achievable throughput ■ Added sections USB UICC and SDCC2 ■ Updated Figure 2-4: Removed two LCD IF to LCD Controller connections — A2[3] to ADV_N and A2[4] to WAIT_N ■ Updated section Camera interface timing requirement for MSM6290 module interface: Revised the highest supported PCLK rate from TBD to 80 MHz ■ Updated Section 2.4.4: Revised the highest supported PCLK rate from TBD to 80 MHz ■ Updated Section 2.5.1: Added a reference to the SD/MMC Application Note (80-V7837-1) ■ Revised Figure 2-12: Removed “for MMC” from SDCC1_DATA0 ■ Updated Table 2-7: Revised MMC signal names ■ Updated Section 2.5.3: Revised to 4-bit MMC support and specifications 4.2 ■ Updated Table 2-9: Changed to MMC version 4.2 ■ Updated Section 2.5.5: Added AMSS polling mechanism support ■ Updated Table 3-3: Changed HSEL_TIF description ■ Updated Table 4-1: Changed “Chip selects” ■ Removed Section 4.1.3, “External memory controller” ■ Updated Section 4.2.6.1: Removed word and Hword reference ■ Updated Section 4.2.7.7: Removed SRAM ■ Updated Table 5-1: Added Platform G ■ Updated Figure 7-4: Interchanged pin labels E2 and F4 ■ Updated Table 7-9: Interchanged UART values “DP_RX_DATA” and “CTS_N” ■ Updated Section 7.3.1: Added a note regarding HS USB interface expected throughput ■ Updated Section 7.3.3: Added PM6653 ■ Updated Figure 7-12: Revised digital die W14 internal connection to the HS-USB ■ Updated Figure 7-13: Added PM6653 ■ Updated Table 10-1: Revised first row description to “Engineering sample 1”; added a second row for additional samples ■ Updated Section 10-2: Changed from GPIO[102]

1 Introduction

1.1 Scope and intended audience

The Mobile Station Modem™ MSM6290™ user guide describes the memory interface, multimedia interface (includes digital camera, SD card, Bluetooth®), RF chip interface, USIM interface, user interface, mobile display digital interface (MDDI), codec acoustic interface, transport stream interface (TSIF), JTAG, ETM, and UART for handset/data-card module designers.

The MSM6290 specifications are included as a courtesy to procurement departments that purchase the device.

For more information about 384-ball NSP package specifications and manufacturing considerations, refer to the *BGA Package User Guide* (80-V2560-1).

1.2 Application description

The global expansion of 3G WCDMA (UMTS) networks has resulted in greater demand for high-speed, wireless data access. This demand has come in the form of wireless devices that send and receive data from the device's camera, camcorder, personal video player, MP3 audio player, gaming console, and phone. To efficiently support next-generation data speeds and functions, wireless devices must use application processors with high-performance modems.

To address this growing opportunity, Qualcomm has developed the MSM6290 chipset and system software solution. MSM6290 is the next generation of the MSM6290 ASIC.

The MSM6290 chipset solution integrates powerful applications processors into the Qualcomm market-proven wireless modem, offering increased processing capacity and lower power consumption. A complete system solution that operates on all UMTS networks worldwide, this chipset and system software offers a feature-rich set of enhanced multimedia data applications and state-of-the-art position location capabilities. The MSM6290 device supports streaming video, audio, still-image and video encoding and decoding, 2D and 3D graphics acceleration, and Java® acceleration. It also integrates a 5.0 megapixel camera interface, complete Bluetooth baseband support, and assisted or standalone GPS.

Major innovations incorporated into the MSM6290 device include 7.2 Mbps HSDPA, 2.0 Mbps HSUPA, EDGE DTM, SAIC, Rx diversity, equalizer, and TSIF (transport stream interface for broadcast).

NOTE The features above are distributed over multiple commercial releases and therefore, all releases may not include all features listed.

With the MSM6290 chipset solution, handset manufacturers can design sleek wireless devices that offer advanced image quality and resolution to provide enhanced 2D/3D animation, gaming, streaming video, videoconferencing, and more.

The Qualcomm MSM6290 chipset solution offers significantly increased processing speeds, high resolution video and graphics, and extended usage times for gaming and video applications; all on a single chip. This single-chip solution eliminates the need for a separate applications processor, which decreases parts count, reduces bill of material (BOM) costs, and supports the development of ultra-compact devices.

MSM6290 chipset solution benefits

- Integrates dedicated hardware cores into the MSM device, which eliminates the need for multimedia coprocessors. Offers superior image quality and resolution for mobile devices and extended application time.
- Enables impressive, user-compelling 2D/3D graphics, multimedia, animation, and video.
- Provides longer operating time for mobile devices when compared to other industry-solutions that use companion processors.
- Provides high-quality video and graphics with quarter video graphics array (QVGA) resolution, a four-fold improvement over quarter common intermediate format (QCIF)
- Offers a higher degree of integration (digital and analog functions on a single chip) and dedicated hardware cores, which decrease power consumption while increasing power and quality.
- Eliminates the need for intermediate frequency (IF) components, decreasing PCB area and reducing time-to market development and BOM costs.
- Enables a wide variety of location-based services and applications, including points of interest, personal navigation, and friend finder.
- Provides dedicated support for all market-leading codecs and other multimedia formats on a single platform to support carrier deployments around the world.
- Provides a platform for HSDPA, HSUPA, and EDGE and eliminates the need for an external processor for high-end features such as realtime video encoding and decoding.
- Offers TSIF (dedicated interface to external broadcast solutions).
- Broadcast interface can support UBM chipsets MBP1600 and MBP1610, DVB-H, T-DMB, and ISDB-T.
- Offers BT 2.0 and 2.1 through high-speed UART.
- High-speed USB.
- Two high-speed SD card controllers and USB UICC.

MSM6290 device description

The MSM6290 device integrates the ARM926EJ™-C processor, offering the ARM Jazelle™ Java hardware accelerator, two low-power, high-performance QDSP4u8 digital signal processor (DSP) cores, and a wideband stereo codec to support enhanced digital audio applications. The QDSP4u8 core eliminates the need for the multimedia companion processors normally required for video and audio-based applications, playing MP3 music files, MIDI synthesizer, video and still-image record and playback, and 2D/3D graphics functions. By removing the need for costly application coprocessors and memory subsystems, the MSM6290 chipset solution reduces BOM costs and increases standby and talk times. The MSM6290 chipset solution integrates both digital and analog functions into a single chip.

The MSM6290 system solution consists of the MSM6290 baseband processor, complete with WCDMA/HSDPA/HSUPA/GSM/GPRS/EDGE/GPS protocol stack and multimedia system software.

The MSM6290 device interfaces with:

- Platform F: RTR6285 quad-band UMTS with Rx Diversity
- Platform G: RTR6280 quad-band UMTS (no diversity and no GPS)
- European market: UMTS 900, AWS, 1900, and 2100
  Japanese market: UMTS 800, 1700, 1900, and 2100
  North American market: UMTS 850, AWS band, 1900, and 2100
- GSM/GPRS/EDGE 850, 900, 1800, and 1900
- GPS (not supported with Platform G)

The baseband and RF chipsets are supported by the latest powerOne™ series PM6658™ and PM6653 power management device.

AMSS software is designed to run on a Subscriber Unit Reference™ (SURF™) phone platform, an optional development platform used to evaluate, test, and debug AMSS software. The MSM6290 device is offered in a 384 ball, 0.4 mm pitch, lead-free NSP production package.

The MSM6290 solution includes multimedia features such as Qtunes™ digital audio that supports MP3 player software, and AAC/AAC plus/EAAC plus. Qualcomm offers an optimized software solution for the modem, along with system development software, verification, test, debug, calibration, manufacturing, and field test support using the UMTS designer development tools, reducing the time to bring a complete UMTS product to market.

The integrated wideband stereo codec converts an analog audio signal, either differential or single-ended, from the microphone into digital signals for the MSM6290 device's vocoder. The integrated codec also converts digital audio data from the vocoder into an analog audio signal, either differential or single-ended, for the earpiece. The internal vocoder supports all eight AMR modes, along with implementing an enhanced echo canceller which allows for full-duplex speakerphone operation. The vocoder also supports DTMF generation and detection, advanced noise suppression, audio AGC control, and automatic volume control (AVC).

Launchpad Suite of technologies

The Launchpad Suite™ of applications offers a cost-effective, scalable, and time-savings solution for wireless operators and manufacturers in providing advanced wireless data services. This seamlessly integrated solution enables advanced next-generation applications and services that incorporate multimedia, position location, connectivity, customized user interface, and storage capabilities. Launchpad features are available for each Qualcomm chipset, closely matching the specific functionality and agreed-upon cost-target objectives of manufacturers and wireless service operators worldwide.

The MSM6290 chipset solution supports the advanced feature set of the Qualcomm Launchpad Suite of technologies, including streaming video and audio, still-image and video encoding and decoding, 2D and 3D graphics acceleration, Java acceleration, and a megapixel camera interface. The MSM6290 IC also integrates gpsOne® functionality, featuring enhancements by SnapTrack® Inc. to support assisted and standalone GPS. This enables a wide variety of location-based services and applications, including points of interest, personal navigation, and friend finder.

gpsOne technology

gpsOne technology merges global positioning system (GPS) satellite and network information to provide a high-availability solution that offers industry-leading accuracy and performance. This solution performs well, even in very challenging environmental conditions where conventional GPS receivers fail, and provides a platform to enable wireless operators to address both location-based services and emergency mandates such as the United States Federal Communications Commission (FCC) E911 mandate.

In mobile-assisted mode, when a request for position location is issued, available network information is provided to the location server (e.g., cell-ID) and assistance is requested from the location server. The location server sends the assistance information to the handset. The handset/mobile unit measures the GPS observables and provides the GPS measurements along with available network data (that is appropriate for the given air interface technology) to the location server. The location server then calculates the position location and returns results to the requesting entity.

In mobile-based mode, the assistance data provided by the location server encompasses not only the information required to assist the handset in measuring the satellite signals, but also the information required to calculate the handset's position. Therefore, rather than provide the GPS measurements and available network data back to the location server, the mobile calculates the location on the handset and passes the result to the requesting entity.

In standalone (autonomous) mode, the handset demodulates the data directly from the GPS satellites. This mode has some reduced cold-start sensitivity, and a longer time to first fix as compared to the assisted modes. However, it requires no server interaction and works out of network coverage.

This combination of GPS measurements and available network information provides:

- High-sensitivity solution that works in all terrains: indoor, outdoor, urban, and rural
- High availability that is enabled by using both satellite and network information

Therefore, while network solutions typically perform poorly in rural areas and areas of poor cell geometry/density, and while unassisted, GPS-only solutions typically perform poorly indoors, the Qualcomm gpsOne solution provides optimal time to fix, accuracy, sensitivity, availability, and reduced network utilization in both of these environments, depending on the given condition.

The gpsOne solution in assisted modes provides cold-start GPS sensitivity that is an approximately 20 to 30 dB improvement over unassisted, conventional GPS receivers.

Compared to network solutions that require equipment at each cell site, the gpsOne solution integrates a complete GPS receiver in every MSM6290 chipset. This means that each handset is capable of position location without requiring expensive cell site equipment. This solution can be used to help operators address the FCC E911 mandate in the U.S. (and mandates planned for other countries), and also promotes consumer-priced, location-based services worldwide.

radioOne technology

The MSM6290 device interfaces directly with the new Qualcomm radioOne® RF ASICs. radioOne is a revolutionary technology for WCDMA/GSM transceivers that uses zero intermediate frequency (ZIF), or direct conversion architecture, for the wireless handset market. This direct conversion eliminates the need for large IF surface acoustic wave (SAW) filters and additional IF circuitry, which significantly reduces the handset BOM parts count, facilitating multiband and multimode handsets that can be produced in smaller form factors. radioOne technology also incorporates the frequency synthesis and passive elements used in converting baseband signals to and from RF. A single external local oscillator is used for the WCDMA/GSM receiver, which provides the capability to operate on systems around the world and simplifies part procurement.

BREW® solution

The MSM6290 device includes support for the Qualcomm BREW solution. The BREW solution is a complete product and business system for the development and over-the-air deployment of data services on wireless devices. The BREW solution provides the necessary tools and value-added services to developers, device manufacturers and wireless operators for application development and distribution, device configuration, and billing and payment.

SecureMSM Security Suite V2

The MSM6290 device includes support for the following SecureMSM™ features:

- Secure boot – protects against re-flashing attacks
  - Trusted third-party certificate authority (GeoTrust) provides code-signing service to manufacturers and Qualcomm.
  - Strong 2048-bit RSA keys are used.
  - Enables manufacturer and Qualcomm to ensure that their code is running unchanged on the device.
- Secure component framework
  - Capabilities-based architecture (access control for resource)
  - Partitioned software provides isolation of compromise/subvert.
  - Only known, trusted processes permitted to access materials in SFS.
- Secure storage
  - 128-bit hardware key for (AES) encryption of secure file system (SFS)
  - Operations using hardware key are run in on-chip RAM, making observation extremely difficult.
- Unchangeable hardware ID is unique to each MSM device.

The result is a very robust platform securing OMA DRM v2.0 services and e-commerce transactions. IMEI freeze and SIM lock protection benefit from hardware security.

1.3 MSM6290 device features

Figure 1-1 shows the MSM6290 system functional block diagram.

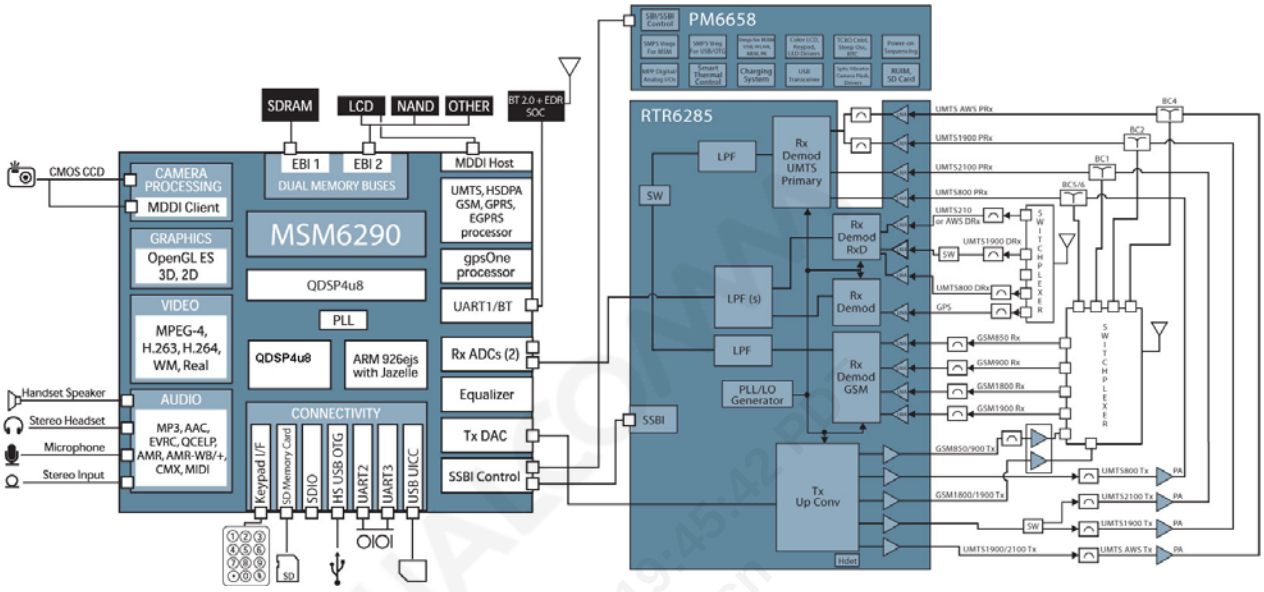


Figure 1-1 MSM6290 system functional diagram (Platform F)

1.3.1 MSM6290 device general features

NOTE The following features are a total feature list and are distributed over multiple commercial releases; therefore, all releases may not include all features listed.

- Support for multimode operation – HSDPA, HSUPA, quad band WCDMA (UMTS), and quad band GSM/GPRS/EDGE, GPS
- Support for HSDPA downlink up to 7.2 Mbps
- Support for HSUPA up to 2.0 Mbps
- Support for WCDMA (UMTS) uplink data rate up to 384 kbps
- High-performance ARM926EJ-S™ running at up to 297.6 MHz for 7.2 Mbps HSDPA
- ARM Jazelle Java hardware acceleration for faster Java-based games and other applets
- QDSP4u8 high-performance DSP cores
- Integrated gpsOne position location technology functionality
- Integrated high-speed UART for BT 2.0 and 2.1 for wireless connectivity to peripherals
- Qcamera™ with 30 fps WQVGA viewfinder resolution, and support for 5 MP camera sensors
- Direct interface to digital camera module with video front end (VFE) image processing
- True 3D graphics for advanced wireless gaming

- SecureMSM v2.0: includes support for open mobile alliance (OMA) DRM v2.0 and support for Qfuse.
- Audio on par with portable music players
- Vocoder support (AMR-NB, AMR-WB, GSM FR/HR/EFR, G.711)
- Advanced 10 × 10 mm, 0.4 mm pitch, 384-ball lead-free NSP packaging technology
- High-speed SD/SDIO hardware support
- USB UICC support

1.3.2 HSDPA features

The MSM6290 device supports the HSDPA release 5 standard:

- Supports HS-DSCH (HS-SCCH, HS-PDSCH and HS-DPCCH) in addition to the R99 transport channels as defined in 3GPP specifications.
- Supports a maximum of four simultaneous HS-SCCH channels as defined in 3GPP specifications.
- Supports a maximum of 10 HS-PDSCH channels and supports both QPSK and 16 QAM modulation.
- Supports CQI, and ACK/NACK on HS-DPCCH channel as defined in 3GPP specifications.
- Supports all incremental redundancy versions for HARQ, as defined in 3GPP specifications.
- Can switch between HS-PDSCH and DPCH channel resources, as directed by the network.
- Can be configured to support any of the two power classes 3 or 4 as defined in 3GPP R5 specifications (25.101).
- Supports network activation of compressed mode by SF/2 or HLS on the DPCH for conducting inter-frequency or inter-RAT measurements when the HS-DSCH is active.
- Supports STTD on both associated DPCH and HS-DSCH simultaneously.
- Supports CLTD mode 1 on the DPCH when the HS-PDSCH is active.
- Supports STTD on HS-SCCH when either STTD or CLTD Mode 1 are configured on the associated DPCH.
- Supports TFC selection limitation on the UL factoring in the transmissions on the HS-DPCCH as required in TS 25.133.

1.3.3 HSUPA features

HSUPA support in the March 2008 AMSS commercial release will be 2.0 Mbps (10 msec TTI). 2ms TTI (estimated achievable throughput rate of 4.5 Mbps) will be added in later CS release.

1.3.4 WCDMA R99 features

The MSM6290 device supports the W-CDMA FDD release 99, including the following features:

- All modes and data rates for W-CDMA frequency division duplex (FDD), with the following restrictions:
  - The downlink supports the following specifications:
    - Up to four physical channels, including the broadcast channel (BCH), if present
    - Up to three dedicated physical channels (DPCHs)
    - Spreading factor (SF) range support from 4 to 256
    - The following transmit diversity modes are supported:
      - Space-time transmit diversity (STTD)
      - Time-switched transmit diversity (TSTD)
      - Closed-loop feedback transmit diversity (CLTD)
- The uplink supports the following specifications:
  - The uplink provides the following UE support:
    - One physical channel, eight TrCH, and 16 TrBks starting at any frame boundary
    - A maximum data rate of 384 kbps
  - Full SF range support from 4 to 256
- SMS (CS and PS)
- PS data rate – 384 kbps DL/384 kbps UL
- CS data rate – 64 kbps DL/64 kbps UL
- AMR (all rates)

1.3.5 GSM features

The following GSM modes and data rates are supported by the MSM6290 device hardware. Support modes conform to release '99 specifications of the sub-feature.

- Voice features
  - FR
  - EFR
  - AMR
  - HR
  - A5/1, A5/2, and A5/3 ciphering

- Circuit-switched data features
  - 9.6 k
  - 14.4 k
  - Fax
  - Transparent and non-transparent modes for CS data and fax
  - No subrates are supported.

1.3.6 GPRS features

- Packet-switched data (GPRS)
  - E-DTM operation
  - Multi-slot class 12 data services
  - CS schemes: CS1, CS2, CS3, and CS4
  - GEA1, GEA2, and GEA3 ciphering
- Maximum of four Rx timeslots per frame

1.3.7 EDGE features

- EDGE E2 power class for 8 PSK
- E-DTM, multislot class 12
- Downlink coding schemes – CS 1-4, MCS 1-9
- Uplink coding schemes – CS 1-4, MCS 1-9
- BEP reporting
- SRB loopback and test mode B
- 8-bit, 11-bit RACH
- PBCCH support
- 1 phase/2 phase access procedures
- Link adaptation and IR
- NACC, extended UL TBF

1.3.8 MSM6290 device audio processing features

- Integrated wideband stereo codec
  - 16-bit DAC with typical 88 dB dynamic range
  - Supports sampling rates up to 48 kHz on the speaker path and 16 kHz on the microphone path
- VR – voicemail + voice memo
- Acoustic echo cancellation
- Audio AGC
- Audio codecs: AMR-NB, AMR-WB/WB+, MP3, MP3 2.5, AAC, AAC Plus, Enhanced AAC Plus, Microsoft® Windows® Audio v9, RealAudio® 8 (G2)
- QConcert™
- QAudioFX™
- CMX® 72-tone poly
- Dual microphone
- Speech codecs: AMR-NB (all rates), AMR-WB, GSM FR/HR/EFR, and G.711

1.3.9 MSM6290 microprocessor subsystem

- Industry standard ARM926EJ-S embedded microprocessor subsystem
  - 16 kB instruction and 16 kB data cache
  - Instruction set compatible with ARM7TDMI®
  - ARM version 5TEJ instructions
  - Higher performance 5 stage pipeline, Harvard-cached architecture
  - Higher internal CPU clock rate with on-chip cache
- Java hardware acceleration

- Enhanced memory support

  NOTE NOR/PSRAM is not supported on MSM6290.

  - 61.44 MHz, 81.92 MHz and 99.20 MHz bus clock for SDRAM
  - 32-bit SDRAM
  - Dual memory buses separating the high-speed memory subsystem (EBI1) from low-speed peripherals (EBI2) such as LCD panels
  - 1.8 V or 2.6 V memory interface support (excluding EBI1)
  - NAND FLASH memory interface
    - 8/16-bit data I/O width NAND Flash support
    - 1- or 4-bit ECC
    - 512-byte/2 KB page-size support
    - 2 chip-selects supported for NAND Flash
  - Boot from NAND
  - Low-power SDRAM (LP-SDRAM) interface
- Internal watchdog and sleep timers

1.3.10 Supported interface features

- High-speed USB for host functionality
- One high-speed universal asynchronous receiver transmitter (UART) serial ports and two legacy UARTs
- USIM controller
- USB UICC controller
- Two integrated 4-bit high-speed secure digital (SD) controller for SD cards
- 24-bit parallel and serial LCD interfaces
- General-purpose I/O pins
- External keypad interface

1.3.11 Supported multimedia features

- Provides additional general purpose MIPS by using:
  - Two QDSP4u8s
  - Dedicated hardware accelerators and compression engines
- Improves Java, BREW, and game performance
  - Integrated Java and 2D/3D graphics accelerator with sprite engine
- Enables various accessories via USB host connectivity
  - Integrated USB host controller functionality
- Enables compelling visual and audio applications

Qcamera

- High-quality digital camera processing, supporting CCD or CMOS image sensors up to 5 megapixels
- 30 fps WQVGA viewfinder

Qtv™

- Audio and video decoder that supports VOD, MOD and broadcast multimedia services
- Audio codecs supported: AMR-NB, AAC, AAC Plus, Enhanced AAC Plus, Windows Audio v9, RealAudio v8
- Integrated stereo wideband codec for music/digital clips
- CMX
- Video codecs supported: MPEG-4, H.263, H.264, Windows Media v9 and RealAudio v10

Video telephony services: Qvideophone™

- A two-way mobile video conferencing solution that delivers 15 fps at QCIF (QCIF to QVGA upscaling is available)
- Video codecs supported: MPEG-4 and H.263
- Audio codecs supported: AMR-NB
- QCIF to QVGA scaling

Qcamcorder™

- Realtime mobile video encoder
- Video codecs supported: MPEG-4, H.263
- Audio codecs supported: AMR-NB, AAC
- Recording performance: 15 fps at QVGA, 384 kbps
- Video telephony at 15 fps, QCIF resolution video encode at 15 fps at CIF for camcorder capability, video decode at 30 fps at CIF resolution, streaming or offline
- gpsOne
- Integrated gpsOne processing
- Standalone gpsOne mode in which the handset acts as a GPS receiver

CMX (MIDI and still image, animation, text, LED/vibrate support)

- 72 simultaneous polyphonic tones
- 44 kHz sampling rate
- 512 kB wave table
- Support of universal file formats
  - Standard MIDI format (SMF)
  - SP-MIDI
  - SMAF® audio playback (MA-2, MA-3, MA-5)
  - XMF/DLS
  - MFil (requires Docomo license)
- PNG decoder
- Pitch bend range support
- LED/vibrate support
- Scalable Vector Graphics (SVG – Tiny 1.1 + SVG Tiny 1.2)
- MLZ decoder
- Integrated PNG/SAF A.T.

Table 1-1 Summary of MSM6290 device features

Features	MSM6290 device
Processor	ARM926 EJ-S – 297.8 MHz[1] ADSP – 122.88 MHz MDSP – 122.88 MHz
Process technology	65 nm
Broadcast	TSIF (dedicated)
UART	One HS UART and two legacy UARTs
Secure Digital (SD)	Two HS SD interface for SD memory and SDIO cards
Bluetooth	BT 2.0 and 2.1 (through HS UART)
High-speed serial interface	MDDI
Security/digital rights management	OMA DRM v2.0 Qfuse supported Trusted boot only
Supported RF platforms	RF CMOS Platform F (Rx diversity) and Platform G (no diversity and no GPS)
gpsOne	Supported
16-bit burst NOR flash + 16-bit or 32-bit burst PSRAM	Not supported
8-bit or 16-bit NAND Flash + 32-bit SDRAM	Supported
16-bit SDRAM and NOR	Not supported
USB	HS-USB (host and peripheral)
Qcamcorder	15 fps at QVGA, 30 fps WQVGA viewfinder
Qtv (video decode)	30 fps at QVGA playback 15 fps at QVGA streaming
Qvideophone (video telephony)	15 fps at QCIF
Qcamera (camera interface)	5M pixel support at 30 fps WQVGA viewfinder
Audio/video decoders	MP3, MP3 2.5, AAC, AAC+, Enhanced AAC+ AMR-NB/WB/WB+ ADPCM, H.263, H.264, Windows Media, Real
2D/3D graphics HW acceleration	HW – 600 K triangles/sec, 90 mega pixels per second

1. The ARM will follow different frequency plans based on mode of operation. Refer to the *MSM6290 Device Specification* (80-VF866-1) for details.

1.4 MSM6290 ASIC overview

UMTS/HSDPA subsystem

The UMTS/HSDPA subsystem performs the digital release 99 June 2004 of the WCDMA FDD standard and release 5 specifications of HSDPA signal processing. See Section HSDPA features for more details.

GSM/GPRS subsystem

The GSM/GPRS/EGPRS subsystem reuses the MSM6250® GSM core. It performs the digital GSM signal processing and PA gain controls for GPRS support. The PA output level is controlled by an analog signal generated on the MSM device. In GSM mode, the power profile ramps up before the burst and ramps down after the burst. In GPRS mode, at the beginning of each burst (up to four active transmit slots), PA must be smoothly ramped up to some desired output power level, held at that level for the current slot, smoothly ramped down/up during the transition period and held to the new level for the next slot until the last slot. Then it must be smoothly ramped down to near-zero level. The MSM6290 device supports differential GSM PA power control output.

The RF interface communicates with the mobile station external RF circuits. Signals to these circuits control signal gain in the Rx and Tx signal path, control DC offset errors, and maintain the system frequency reference.

Stereo wideband codec

The MSM6290 device integrates a wideband voice/audio codec into the MSM. The codec supports two differential microphone inputs, one differential earphone output, one single-ended earphone output, and a differential analog auxiliary interface. The codec integrates the microphone and earphone amplifiers into the MSM6290 device, reducing the external component count to just a few passive components. The microphone (Tx) audio path consists of a two-stage amplifier with the gain of the second stage set externally. The Rx/Tx paths are designed to meet the ITU-G.712 requirements for digital transmission systems.

Vocoder subsystem

The MSM6290 QDSP4u8 supports AMR, FR, EFR, and HR. In addition, the QDSP4u8 has modules to support the following audio functions: DTMF tone generation, DTMF tone detection, Tx/Rx volume controls, Tx/Rx automatic gain control (AGC), Rx automatic volume control (AVC), ear seal echo canceller (ESEC), acoustic echo canceller (AEC), noise suppression (NS), and programmable, 13-tap, Type-I, FIR, Tx/Rx compensation filters. The MSM6290 device's integrated ARM926EJ-S processor downloads the firmware into the QDSP4u8 and configures the QDSP4u8 to support the desired functionality.

ARM microprocessor subsystem

The MSM6290 device uses an embedded ARM926EJ-S microprocessor. This microprocessor, through the system software, controls most of the functionality for the MSM device, including control of the external peripherals such as the keypad, LCD, RAM, ROM, and EEPROM devices. Through a generic single-wire serial bus interface (SSBI), the ARM926EJ-S configures and controls the functionality of the RTR6285, RTR6280, PM6658, and PM6653 devices.

UART

The MSM6290 device employs three UARTs. UART1 is a high-speed UART and has dedicated pins while UART2 and UART3 are legacy UARTs and share multiplexed pins.

UART1 is expected to achieve a maximum of up to 4 Mbps throughput rate while UART2 and UART3 can achieve a maximum of up to 230 kbps throughput.

USB

The MSM6290 device integrates a high-speed universal serial bus (USB) controller that supports both unidirectional and bidirectional transceiver interfaces. The USB controller acts as a USB peripheral communicating with the USB host.

Apart from the high-speed USB controller, the MSM6290 device also integrates a full-speed USB for USB UICC.

Theoretically the high-speed USB controller can achieve a maximum of up to a 480 Mbps throughput rate. But due to system overheard, it is expected to achieve much lower than a 480 Mbps throughput rate.

USB UICC

The secondary USB port on the MSM6290 device is used for the universal integrated circuit card (UICC). It is a SIM card with both USB and UART interfaces.

A UMTS USB SIM card is supported using a separate USB full-speed two-wire interface. Two new pins have been added to the MSM6290 device for this purpose. The MSM6290 device contains a new pad called VDD_PAD4, which can be operated at 2.85 V (or 1.8 V). Refer to the *MSM6290 Mobile Station Modem IC Device Specification* (80-VF866-1) for more information.

SDCC2

A second secure digital card controller (SDCC2) port has been added to the MSM6290 device to support the SD memory card and SDIO WLAN functionality simultaneously.

The SDCC2 block will support SD version 2.0, SDIO version 2.0, and MMC version 4.2 specifications. It also supports high-speed mode operations. GPIOs connected to the SDCC2 port are 3 V tolerant.

User interface

The MSM6290 device user interface comprises digital connections to the subscriber unit ringer transducer, keypad, and LCD.

General-purpose interface bus

The MSM6290 device has general-purpose bidirectional input/output pins. Some of the GPIO pins have alternate functions supported on them. The alternate functions include a USB interface, additional RAM, ROM, and general-purpose chip-selects, a parallel LCD interface, and a UART interface. The function of these pins is documented in the various software releases.

Mode select and JTAG interfaces

The mode pins to the MSM6290 device determine the overall operating mode of the ASIC. The options under the control of the mode inputs are native mode, which is the normal subscriber unit operation, ETM mode, which enables the built-in trace mode, and test mode for factory testing.

The MSM6290 device meets the intent of the ANSI/IEEE 1149.1A-1993 feature list. The JTAG interface can be used to test digital interconnects between devices within the handset during manufacture.

Camera interface

The camera interface allows the MSM6290 device to be connected to an external camera with no external logic.

MDDI

The mobile display digital interface (MDDI) is a cost-effective low-power solution that enables high-speed short-range communication with a display device using a digital packet data link. MDDI camera interface is also supported.

MDP

The mobile display processor (MDP) is a hardware accelerator primarily responsible for transferring an updated image from the MSM memory subsystem to the LCD module. It also supports QVGA (320 x 240) to WQVGA (400 x 240) frame scaling for display.

MPM

The MSM6290 device has the capability to use the mobile power manager (MPM) functionality to minimize current consumption during sleep. For customers that are planning to use this feature on future MSM6290-compatible MSM devices, the list of GPIOs that can be used for interrupts during sleep is provided in *MSM6290 Mobile Station Modem Device Specification* (80-VF866-1).

2 Multimedia Applications

2.1 Mobile display digital interface

2.1.1 MDDI overview

The MDDI is a cost-effective low-power solution that enables high-speed short-range communication between an MSM and a display device using a digital packet data link. MDDI simplifies the interconnection between a host processor and a display within a device to reduce the cost of these connections and improve reliability.

2.1.2 MDDI system architecture overview

2.1.2.1 Terminology

The devices connected by the MDDI link are called the host and client. Data going from the host to the client travels in the forward direction, and data from the client to the host travels in the reverse direction.

The MDDI host is the master of the host-to-client communication link. Most often the host sends display data to the client (forward traffic). The host enables reverse link communication by sending a special packet to the client that allows it to take over the bus for a specified duration and send packets to the host (reverse traffic). The time interval of the host polling the client is predetermined by the host and is based on the requirements of each specific application.

2.1.2.2 MDDI interface types

There are four types of MDDI interfaces. Type 1 uses two signal pairs and is suited for mobile phone, PDA, electronic game, and portable media player applications. Types 2 through 4 support higher data rates by sending multiple bits in parallel. The protocol allows any Type 1, 2, 3, or 4 host to communicate with any Type 1, 2, 3, or 4 client by negotiating to as high a rate as possible.

The least capable device is the determining factor for the performance of the link. In systems where the host and client are both capable of using Type 2, Type 3, or Type 4 interfaces, the system always begins operation as a Type 1 interface. The host determines the client's capability and negotiates to hand-off to either a Type 2, Type 3, or Type 4 mode as appropriate for the particular application. The MDDI host and client in MSM6290 support Type 1 mode which is more than adequate for use with internal displays and cameras.

2.1.2.3 MDDI data and strobe

The primary data path is via MDDI_Data0± and MDDI_Stb±. Each is a low-voltage differential signal carried by a differential pair of wires in the cable. There is only one transition on either the MDDI_Data0 pair or the MDDI_Stb pair for each bit sent over the interface. The host always drives the MDDI_Stb to the client, even when reverse data packets are sent. Data can flow in both the forward and reverse directions over the MDDI_Data pairs, although the host is always the master of the data link. The data and strobe signals are always operated in a differential mode to maximize noise immunity. While the MDDI host core can handle data rates of up to 400 Mbps, the maximum rate is 225 Mbps because of the maximum available system clock.

The devices connected by the MDDI link are called the host and client. Data going from the host to the client travels in the forward direction, and data from the client to the host travels in the reverse direction. The mobile device is the MDDI host and is the master of the host to client communication link. Most often the host sends display data to the client (forward traffic). The host enables reverse link communication by sending a special packet to the client that allows it take over the bus for a specified duration to send packets to the host (reverse traffic). The time interval of the host polling the client is pre-determined by the host and is based on the requirements of each specific application.

Figure 2-1 illustrates the MDDI physical connection of host and client, Type-1.

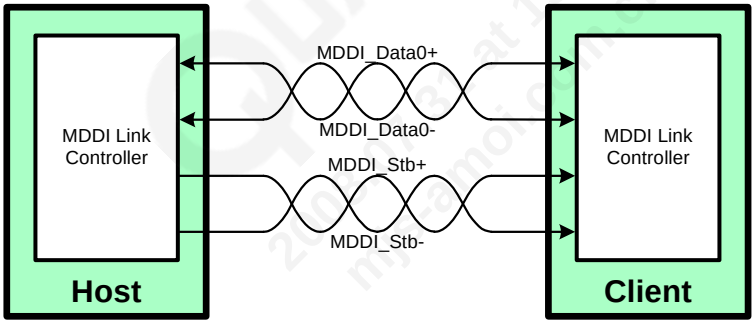


Figure 2-1 MDDI physical connection of host and display, type-1

Figure 2-2 illustrates the MDDI flip phone (handset vendors have the option of using the MDDI for the flip phone or the normal 12-20 wire interface used in a typical “candybar” phone). Pixel data from the camera is received and buffered in the camera interface block and formatted into MDDI digital pixel data in the same format that comes from the camera. This pixel data is then sent to the video front-end (VFE) block in the MSM device for further processing.

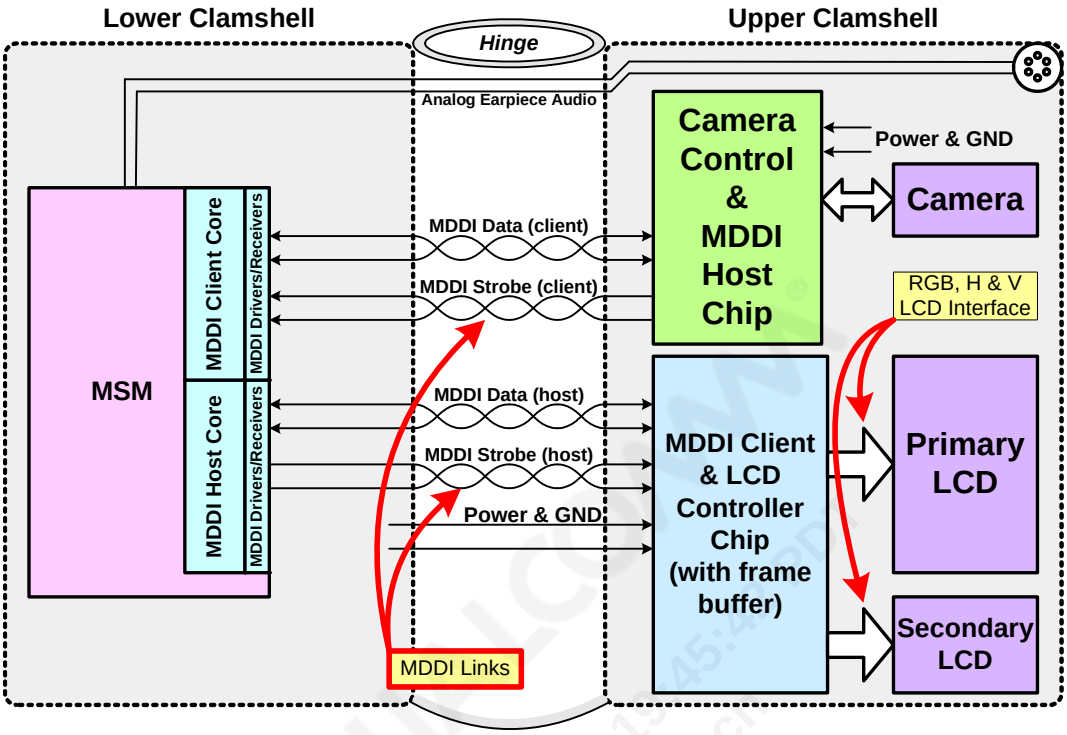


Figure 2-2 MDDI high-level block diagram

2.1.2.4 MDDI I/O pads interface

Figure 2-2 shows a top-level block diagram of the MDDI I/O system at the phone level. This diagram shows the direction and speed at which data can be transmitted over the two interfaces. Terms used in the illustration are explained as follows:

- The camera module and the LCD module would be located in the top portion (flip) of the phone. This chip (including the host interface contained within the camera module) would be produced by our MDDI licensees, and would be an integrated part of the phone camera module.
- "Host" represents the host drivers and receivers (refer also to Figure 2-3). MDDI host and client cores (also known by names of mddi_camif and MDDI) are located on the MSM. The inputs/outputs of the host and client drivers would be sent to these cores.

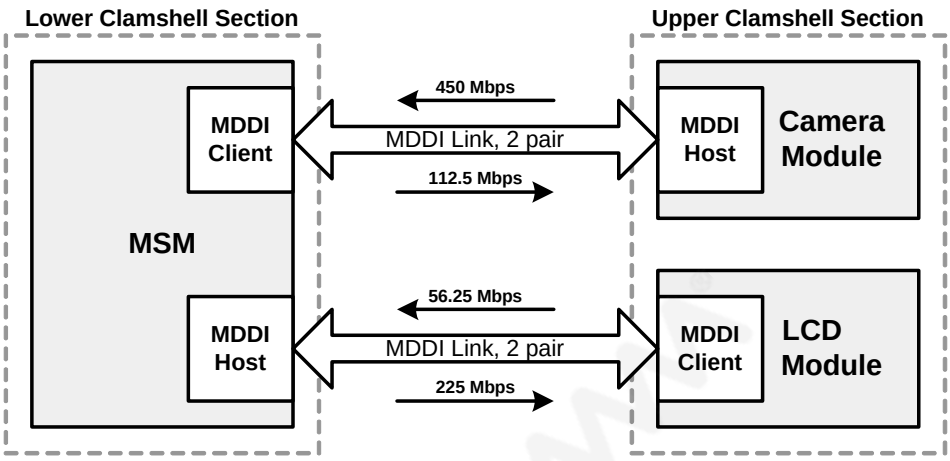


Figure 2-3 MDDI top-level block diagram

Table 2-1 MDDI I/O pad interface specification

Signal names	Description	E/I	I/O/P	Default state	Properties
MDDIH_DATP	MDDI data and strobe signals for the primary MDDI link that is connected to the MDDI host core	E	I/O	Z	Low swing differential signals
MDDIH_DATN		E	I/O	Z	
MDDIH_STBP		E	O	Z	
MDDIH_STBN		E	O	Z	
MDDIC_DATP	MDDI data and strobe signals for the camera MDDI link that is connected to the MDDI client core	E	I/O	Z	Low swing differential signals
MDDIC_DATN		E	I/O	Z	
MDDIC_STBP		E	I	X	
MDDIC_STBN		E	I	X	
VDD_MDDI	Pad power and ground signals for both the host and client pads	E	P	1.8 V	1.8 V, vdd_mddi
VSS_MDDIB		E	P	0 V	30 mA max current draw

2.2 MDP

2.2.1 Overview

The MDP is a hardware accelerator primarily responsible for transferring an updated image from the MSM memory subsystem to the LCD module. The necessity to transfer an updated image is an operation that is shared between software, video processing and graphics processing, so a common block helps to reduce redundant circuitry. The MDP is designed with the assumption that the LCD panel has an embedded LCD controller and a frame buffer. The image transfer is then the copying of the image from the MSM memory system to the frame buffer within the LCD module.

While the MDP is transferring an image to the LCD module, it can perform a final set of operations to the image. The set of operations that the MDP can perform has been chosen to maximize the efficiency of the MSM memory subsystems, typically removing two or more copy operations of the image to and from memory.

The MDP reduces redundant circuitry and offloads the ARM and aDSP from memory transfer operations and a certain set of graphics and video operations.

2.2.2 Features

- Frame update synchronization

  Tearing or drawing effects will result if the frame buffer write pointer gets ahead of the read pointer. In order to prevent this, the MDP tracks the LCD's frame buffer write pointer (via LCD_VSYNC and SST). MDP can throttle writes to the frame buffer so that updates occur only when safe. To assure correct operation, it is important to verify that the frame buffer update rate is longer than one refresh cycle (to prevent the write pointer getting ahead of the read pointer) and shorter than two refresh cycles (to prevent tearing). In addition, the frame buffer update has to be synchronized with the read pointer.

- It is important to note that in order to fully take advantage of this feature, the LCD must be capable of providing an LCD_VSYNC output to the MSM that the MDP will use to calculate the location of the read pointer of the frame buffer.

- Rotation (90, 180, 270 degree) and flip (left/right and top/bottom)

- In previous MSM devices, the ARM rotated and/or flipped the image before transferring it to the LCD. This was a very expensive operation at the hardware level with several MSM-memory transfer operations and a lot of used MIPS (unavoidable even with optimized software). The MDP, therefore, offloads the ARM and the multiple memory transfer operations. It is capable of providing 90, 180, and 270 degree rotations and left-right and top-bottom flips as it transfers an image from memory to the LCD panel.

- Video functions

  - Chroma upsampling
  - Color conversion (i.e., YCbCr 4:2:0 to RGB 4:4:4 format)
  - Image size downscale by n/2, n/4, and 3n/4
  - Image size upscale by two

- In previous MSM devices, the aDSP converted video frames from 4:2:0 to an RGB video output buffer. Since not all frames are displayed, this can be a waste of processing power. In addition, since the buffers are in the external memory, it can waste memory bus bandwidth. The MDP, therefore, offloads the aDSP from the above operation by color base converting the image as it is transferred to the LCD panel.
- MDP is also capable of n/2, n/4, and 3n/4 downscaling or by-2 upscaling.
- Graphics functions
  - Alpha-blend
  - Transparency
  - Text overlay
- Again, the MDP now offloads the ARM, which previously had to perform the alpha-blend, thus saving more MIPS and memory bandwidth. The MDP does this by having the ability to read from two sources, combine them, and then send the results to the display.
- In addition to the above graphical and image processing features, the MDP also has the following, higher level, features:
- Flexible LCD interface
- Script processing
- QVGA (320 x 240) to WQVGA (400 x 240) scaling

2.2.3 MDP LCD module interface

The block diagram of the interface between the MDP block and the LCD module is shown in Figure 2-4.

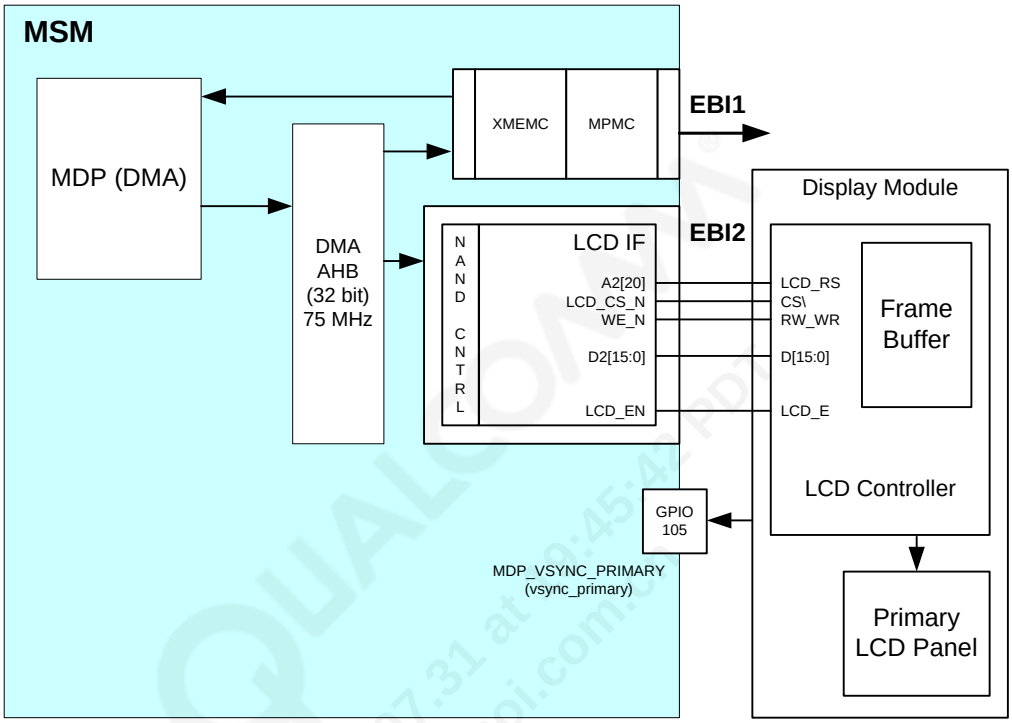


Figure 2-4 Block diagram of the MDP-to-LCD interface

The EBI2 interface available for the MDP LCD module available on the MSM6290 device.

The EBI2 interface is capable of supporting 16 bpp, 18 bpp, and 24 bpp.

Table 2-2 Expected MDP outputs for the RGB formats for EBI2

RGB format	Writes	Expected output
RGB 5:6:5	1	Cycle 1: [15:0] = R5R4R3R2R1 G6G5G4G3G2G1 B5B4B3B2B1
	2 (bus-sized, 2 pixels)	Cycle 1: [15:0] = R5R4R3R3R1 G6G5G4G3G2G1 B5B4B3B2B1 Cycle 2: [15:0] = R5R4R3R3R1 G6G5G4G3G2G1 B5B4B3B2B1
	2	Cycle 1: [7:0] = R5R4R3R3R1 G6G5G4 Cycle 2: [7:0] = G3G2G1 B5B4B3B2B1
	2 (bus-sized)	Cycle 1: [7:0] = G3G2G1 B5B4B3B2B1 Cycle 2: [7:0] = R5R4R3R2R1 G6G5G4
	4 (bus-sized, 2 pixels)	Cycle 1: [7:0] = G3G2G1 B5B4B3B2B1 Cycle 2: [7:0] = R5R4R3R3R2R1 G6G5G4 Cycle 3: [7:0] = G3G2G1 B5B4B3B2B1 Cycle 4: [7:0] = R5R4R3R2R1 G6G5G4

Table 2-2 Expected MDP outputs for the RGB formats for EBI2 (cont.)

RGB format	Writes	Expected output
RGB 6:6:6	1	Cycle 1: [18:0] = R6R5R4R3R2R1G6G5G4G3G2G1 B6B5B4B3B2B1
	2	Cycle 1: [15:0] = XXXXXXXR6R5R4R3R2R1G6G5G4 Cycle 2: [15:0] = XXXXXXXG3G2G1B6B5B4B3B2B1
	2 (bus-sized)	1) Cycle 1 [15:0] = R4R3R2R1G6G5G4G3G2G1 B6B5B4B3B2B1 Cycle 2 (same as CS cycle) [15:0] = XXXXXXXXXXXXXXR6R5 2) Cycle 1 [15:0] = G6G5G4G3G2G1 XXB6B5B4B3B2B1XX Cycle 2 (same as CS cycle) [15:0] = XXXXXXXXR6R5R4R3R2R1XX
	3	1) Cycle 1 [7:0] = XXR6R5R4R3R2R1 Cycle 2 [7:0] = XXG6G5G4G3G2G1 Cycle 3 [7:0] = XXB6B5B4B3B2B1 2) Cycle 1 [7:0] = R6R5R4R3R2R1XX Cycle 2 [7:0] = G6G5G4G3G2G1XX Cycle 3 [7:0] = B6B5B4B3B2B1XX
	4 (bus-sized)	1) Cycle 1 [7:0] = G2G1 B6B5B4B3B2B1 Cycle 2 (same as CS cycle) [7:0] = R4R3R2R1G6G5G4G3 Cycle 3 (same as CS cycle) [7:0] =XXXXXXR6R5 Cycle 4 (same as CS cycle) [7:0] = XXXXXXXX 2) Cycle 1 [7:0] = B6B5B4B3B2B1XX Cycle 2 (same as CS cycle) [7:0] = G6G5G4G3G2G1XX Cycle 3 (same as CS cycle) [7:0] = R6R5R4R3R2R1XX Cycle 4 (same as CS cycle) [7:0] = XXXXXXXX

Table 2-2 Expected MDP outputs for the RGB formats for EBI2 (cont.)

RGB format	Writes	Expected output
RGB 8:8:8	1	Cycle 1: [23:0] = R8R7R6R5R4R3R2R1 G8G7G6G5G4G3G2G1B8B7B6B5B4B3B2B1
	2	1) Cycle 1: [15:0] = R8R7R6R5R4R3R2R1 G8G7G6G5G4XXXX Cycle 2: [15:0] = G3G2G1B8B7B6B5B4B3B2B1XXXX 2) Cycle 1: [15:0] = XXXXR8R7R6R5R4R3R2R1 G8G7G6G5G4 Cycle 2: [15:0] = XXXXG3G2G1B8B7B6B5B4B3B2B1
	2	Cycle 1: [15:0] = XXXXXXXXR8R7R6R5R4R3R2R1 Cycle 2: [15:0] = G8G7G6G5G4G3G2G1 B8B7B6B5B4B3B2B1
	2 (bus-sized)	Cycle 1: [15:0] = G8G7G6G5G4G3G2G1 B8B7B6B5B4B3B2B1 Cycle 2 (same as CS cycle): [15:0] = XXXXXXXX R8R7R6R5R4R3R2R1
	3	Cycle 1: [7:0] = R8R7R6R5R4R3R2R1 Cycle 2: [7:0] = G8G7G6G5G4G3G2G1 Cycle 3: [7:0] = B8B7B6B5B4B3B2B
	4 (bus-sized)	Cycle 1: [7:0] = B8B7B6B5B4B3B2B Cycle 2 (same as CS cycle): [7:0] = G8G7G6G5G4G3G2G1 Cycle 3(same as CS cycle): [7:0] = R8R7R6R5R4R3R2R1 Cycle 4 (same as CS cycle): [7:0] = XXXXXXXX

NOTE Multiple writes will degrade the performance.

2.2.4 Hardware architecture

2.2.4.1 Major blocks within the MDP

Figure 2-5 shows the major blocks within the MDP.

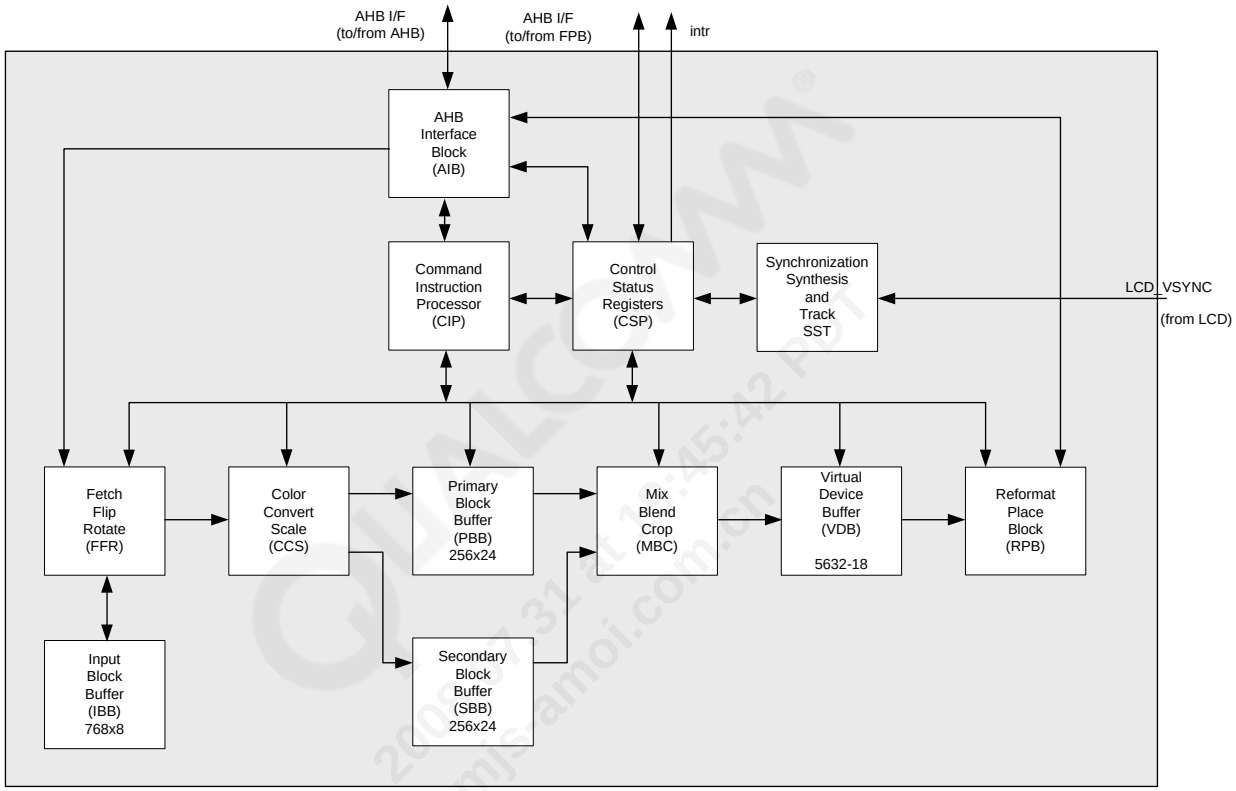


Figure 2-5 Major functional blocks within the MDP

AHB interface block

The AHB interface block (AIB) handles the AHB signaling and protocol for both the input and output of the MDP. Image data is input through the AIB from external memory and accepted in efficient four-beat bursts. After the image data has been processed through the MDP, it is presented back to the AIB and then passed along to the LCD panel. The data is output to the AHB system as a series of single pixels encapsulated in single beat word transfers.

Fetch flip and rotate

The fetch flip and rotate (FFR) block generates the addresses for the AIB block. After the AIB has received the input data, the FFR stores the data into the input block buffer (IBB) so that the image is in the correct orientation.

After the FFR has processed a block, then it has the option of repeating the sequence for several blocks. How far over to step for the next block and how many blocks to process in the stripe are programmable through instructions processed by the CIP.

After the FFR has processed a stripe, it then has the option of repeating the sequence for several stripes. How far over to step for the next stripe and how many stripes to process for the image are programmable through instructions processed by the CIP.

IBB

The IBB is the storage location for the raw block read from memory. It is fed on the input side by the FFR and the data is extracted on the output side by the CCS. The image in the IBB is in the correct orientation for the panel.

Color convert scale

The color convert scale (CCS) takes the data from the IBB and converts it to an 18 or 24-bpp format and then stores the pixel in either the PBB or the SBB. The CCS can also either scale the image up or down prior to storage. The two destinations allow the MBC block to mix and combine two separate images.

Primary block buffer

The primary block buffer (PBB) holds the primary image block. Typically an image block is a 16 × 16 block of pixels, but sometimes it can be less if the MDP is processing an edge of an image. In the case where there is neither mixing nor text overlay, the PBB holds the only image block. In either case, the image block sits in the PBB until there is room in the VDB for storage.

Secondary block buffer

The secondary block buffer (SBB) holds the secondary image block. In the case where there is no mixing nor text overlay, then the SBB sits idle.

Mix blend crop

The mix blend crop (MBC) takes the image blocks from the PBB and SBB, combines the images and then places the final image block into the VDB. The MBC can alpha-blend or perform transparency transformations on the way to the VDB.

Virtual device buffer

The virtual device buffer (VDB) accepts blocks of pixels on the input and produces pixel data on the output in a row format that the LCD expects.

Reformat place block

The reformat place block (RPB) extracts the pixels from the VDB, reformats them and then places them on the MDP AHB bus. From there the pixels go to the LCD on the EBI2 interface.

Command instruction processor

The command instruction processor (CIP) is the brain of the MDP. Software passes the MDP an address that points to the beginning of a script. The commands and instructions in the script define the functionality of the MDP for the next refresh cycle of the LCD. Sometimes the MDP will only update a small portion of the screen and sometimes the MDP will modify the whole screen. The MDP processes the command until it comes across a HALT command, at which time it goes idle again — waiting for the next script pointer from software.

Synchronization synthesis and tracking

The synchronization synthesis and tracking (SST) block generates a local copy of the read pointer so that the MDP knows when to perform its updates without tearing the display. The SST locks to the vertical synchronization pulse from the display and uses its location to synchronize the internal counters inside the SST. The SST does not have to exactly lock to the panel timing, but it has to be within a few lines of accuracy. The SST can restart counting every frame so that errors do not propagate.

Control and status registers

The control and status registers (CSR) block allows software to monitor and control the MDP. Most of the parameters within the main processing chain of the MDP are controlled by the scripts, but functions that are completely static are controlled through the CSR. For debug purposes, all of the parameters that can be set through the CIP can be monitored through the CSR.

2.2.5 MDP-related design considerations

The various design considerations relating the MDP are summarized below:

- The MDP is designed to work with LCD panels that have an LCD controller and a frame buffer
- The MDP will not work with memory-mapped LCD modules
- The MDP will work with parallel LCD panels that satisfy the first two conditions
- The recommended, and typical, refresh rate for the LCD is 60 Hz (it should be noted that changing the refresh rate (in either direction) can lead to tearing)
- The frame buffer rate should be greater than one refresh cycle but less than two refresh cycles
- MDP can support maximum resolution up to QVGA. However, MDP can also support QVGA (320 x 240) to WQVGA (400 x 240) scaling.

To take full advantage of the capabilities of the MDP, it is strongly recommended that LCD modules that output an external VSYNC signal are used. This signal is connected to the MSM6290 device via GPIO105 (primary) or GPIO104 (secondary) and is what the MDP uses to calculate the read pointer's location and to prevent tearing.

2.3 Camera module interface

2.3.1 Overview

The camera interface (Figure 2-6) is intended for direct connection of a camera sensor to the MSM6290 device. It is therefore primarily used in Qcamera, Qcamcorder, Qtv, and Qvideophone applications. The MSM device accepts raw Bayer pattern data (preferably), performs the required image processing (RGB triplet generation, color space conversion, auto white balance, auto exposure, gamma correction, etc.), and prepares the image for capture or transmission. The MSM6290 device is also capable of supporting a YUV 4:2:2 input from the sensor.

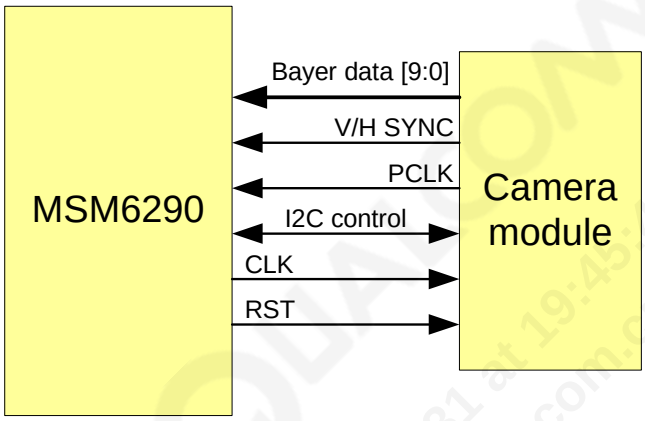


Figure 2-6 Camera module interface

2.3.2 General description

The camera module interface consists of the following:

- 10-bit data bus for the pixel data information
- Horizontal and vertical synchronization signals
- Two-wire I2C bus as a control path between the MSM device and the camera module

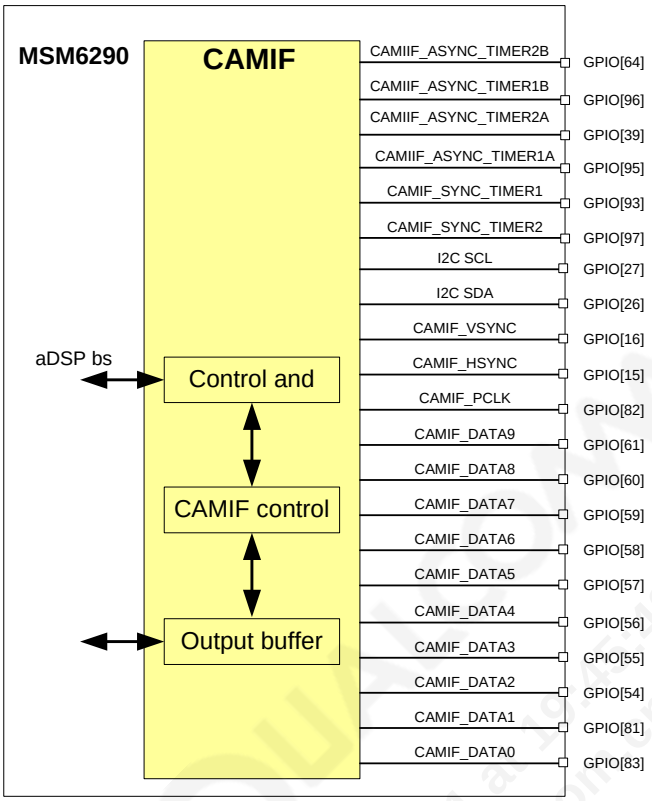


Figure 2-7 MSM6290 camera interface

Table 2-3 I/O and power supply signals to/from the MSM6290 device

Signal name	I/O/B/T/P	Description	Notes
MCLK	O	Master output clock	Generated by MSM or an external oscillator
MRST	O	Master reset	Generated by MSM (or external circuitry)
D[9:2]	I/T	8 MSBs of 10-bit data – the 8 bits that should be used in an 8-bit-only connection	Bayer input data
D[1:0]	I/T	2 LSBs of 10-bit data – should be left unconnected for an 8-bit-only connection	Bayer input data
PCLK	I/T	Pixel clock output	Pixel rate clock
HSYNC	I/T	Horizontal sync	
VSYNC	I/T	Vertical sync	
SDA	B	Serial control data	Serial control data
SCL	B	Serial control clock	Serial control clock
IOVCC	P	Digital I/O power	2.5 to 2.7 V
DVCC	P	Digital sensor core	1.8 V power
GND	P	Module GND	Multiple analog/digital GND signals

MCLK: master clock input

This is the main clock into the camera module. This clock can be generated by the MSM device (CAMCLK_PO — primary, GP_CLK — secondary) or through an external clock source. In case of an external clock source, the module manufacturer is responsible for sourcing this clock. The output pixel clock, PCLK, normally generated by the camera module, is derived from this clock. The typical frequency of this clock varies depending on the requirements of the camera module.

MRST: master reset input

This is an optional reset line that can be provided from the MSM device (or external clock circuitry) into a camera module. The signal is optional and can be active high or active low, depending on the camera module requirements.

Camera interface timing requirement for MSM6290 module interface

- **D [9:2]** – 8 MSB bits of the output data bus from the camera module for a 10-bit I/F.
- **D [1:0]** – 2 LSB bits of the output data bus from the camera module for a 10-bit I/F. In case of a camera module with 8-bit output capability, these bit positions are idle and will not be connected. Bit 9:2 will provide the 8-bit bus.
- **PCLK** – Pixel clock output from the camera module. This is the output pixel clock from the camera module. The MSM device uses this clock to latch in the output data from the camera module. It is assumed that this clock is continuous with a programmable function to gate the clock off as power saving options. The supported PCLK rate can now be as high as 80 MHz.
- **HSYNC** – Horizontal synchronization signal. This is the line synchronization signal output from the camera module. Unlike previous MSM devices, the new dedicated hardware allows for automatic tracking of this signal by the MSM6290 device.
- **VSYNC** – Vertical synchronization signal. This is the Frame synchronization signal output from the camera module. Unlike previous MSM devices, the new dedicated HW allows for automatic tracking of this signal by the MSM6290 device.
- **SDA** – Serial data line of the I2C bus. The standard required pull-up resistor for this signal is placed on the MSM device side of the connection. It is not necessary for the camera module to have a pull-up resistor on this signal.
- **SCL** – Serial clock line of the I2C bus. The standard required pull-up resistor for this signal is placed on the MSM device side of the connection. It is not necessary for the camera module to have a pull-up resistor on this signal.
- **Synchronous timers** – Two single-line synchronous timers for flash control and mechanical shutter
- **Asynchronous timers** – Two pairs of asynchronous timers for auto-focus applications

2.4 Clock requirements

The camera module is driven by a master clock input that is provided from either the MSM device or an external oscillator. It is important to verify that the jitter on the clock input (to the camera) does not violate the sensor specs. In addition, because the PCLK input to the MSM device is derived from the MCLK, it is important to verify the PCLK quality is good and that any jitter that is carried over from the MCLK does not violate the MSM specifications.

2.4.1 CAMCLK_PO

If an MSM clock source is chosen, customers should make sure to use the CAMCLK_PO output from GPIO13. This dedicated clock is specifically designed to provide a clock input to sensors. It has several convenient features such as software gating that allows users to control turning the clock on and off cleanly, and it is also glitch-free. Because it was designed specifically for sensor use, and because the PCLK input to the MSM device may be derived from it, the CAMCLK_PO will be characterized and tested along with the rest of the camera interface to ensure correct and reliable operation.

If an MSM output clock to the sensor is desired, it is recommended that the CAMCLK_PO be used due to the previously mentioned features.

The CAMCLK_PO clock can be derived from various clock sources which can then be further divided by a modulo divider or by an M/N:D counter. It is important to note the limitations of the M/N counter. Customers should make sure to test their desired clock output and to verify that it conforms to the sensor requirements. For minimal jitter and minimal duty-cycle degradation, even integer dividers should be used. If other dividers are desired, Qualcomm will work with customers to determine the feasibility of different M/N values and the resulting clock quality.

The CAMCLK_PO settings are configured in the CAMCLK_PO_CLK_MD and CAMCLK_PO_CLK_NS.

2.4.2 Frame synchronization capabilities

The data output on the 8- or 10-bit bus should by synchronized to a set of frame and line synchronization signals, recognizable by the MSM device. These synchronization signals are:

- HSYNC – Horizontal synchronization signal, indicating the start and end of a line of video
- VSYNC – Vertical synchronization signal, indicating the start and end of a frame of video

The usual mode of operation is active physical synchronization. This is where the MSM receives the HSYNC and VSYNC as inputs from the camera module. It then uses those signals in the internal VFE to correctly parse and rebuild the frames. However, though rarely used, the MSM6290 device is also capable of supporting embedded synchronization. This is a mode in which a set of unique flags is embedded into the data stream, indicating the line or frame boundaries. In the case where the camera module embeds the synchronization codes into the data stream, the HSYNC and VSYNC signals are not required to be present.

The preferred type of data format is raw Bayer RGB data (though YCbCr 4:2:2 is also supported).

2.4.3 Frame synchronization timing

Figure 2-8 shows the relationship between the vertical and horizontal sync signals. The timing of the arrival of the first valid active video line with respect to VSYNC signal is programmable. All synchronization signals throughout this document are shown as active low. However, the MSM device can be programmed to accept both active high or active low pulses for HSYNC, VSYNC and PCLK signals.

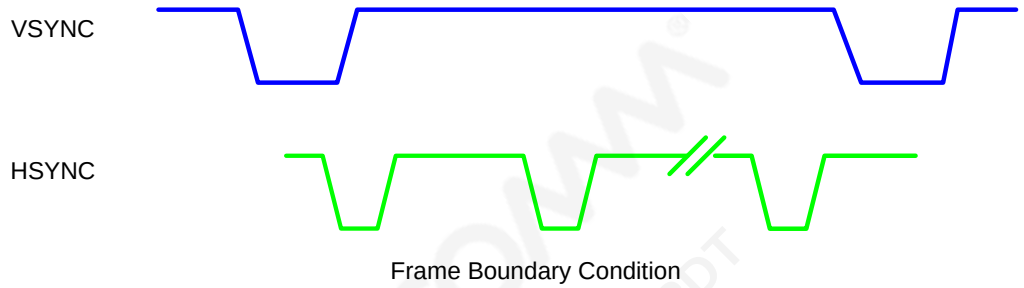


Figure 2-8 Horizontal and vertical synchronization signals

Figure 2-9 and Figure 2-10 indicate the start of frame conditions. In Figure 2-9 it is assumed that the camera module outputs its first active line immediately after the VSYNC pulse, where Figure 2-10 shows the condition where a number of inactive lines can precede the first active line of the frame. The MSM devices can interface with modules with either capability.

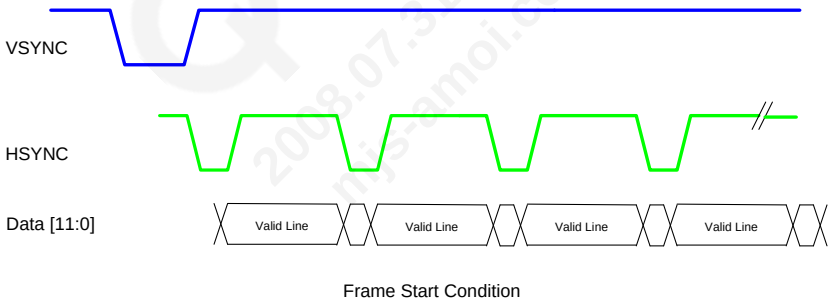


Figure 2-9 Frame start condition

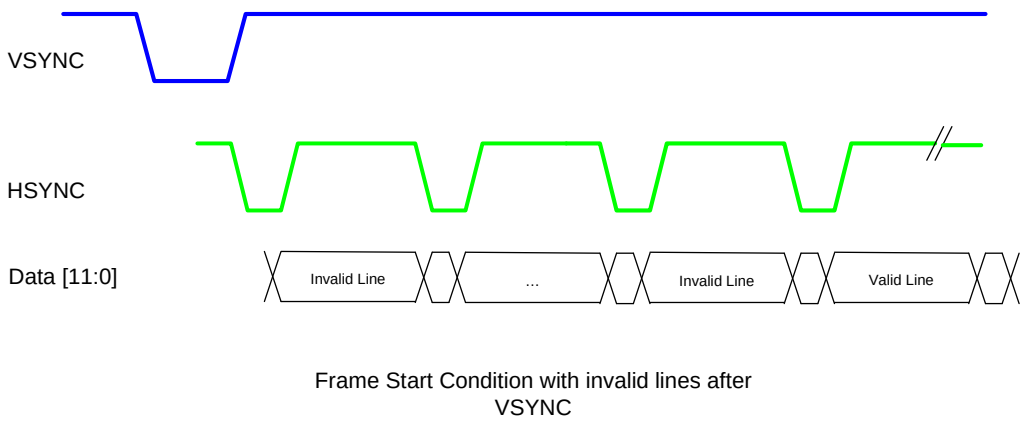


Figure 2-10 Frame start condition with invalid lines after VSYNC

The MSM device is also flexible in handling the different end-of-frame conditions, as indicated in Figure 2-9 and Figure 2-10. In Figure 2-9 it is assumed that the camera module outputs active line up until immediately before VSYNC pulse, with no invalid lines present during the active VSYNC period. Figure 2-10 shows the condition where a number of inactive lines can precede the VSYNC pulse. The MSM devices can interface with modules with either capability.

2.4.4 VFE hardware

The MSM6290 device features dedicated VFE hardware. These hardware blocks decrease much of the image processing load that was previously handled by the DSP. As a result, the VFE hardware has greatly increased the speed and capability of the multimedia applications on the MSM6290 device. Some of the important capabilities of the VFE and the CAMIF are highlighted in the following:

- The supported PCLK rate can be up to 80 MHz – This is a significant improvement from some of the previous MSM device's capability and opens the door to true DSC capability. Using raw Bayer, the MSM6290 device can support up to a 5-MP image (camera snapshot). More of the MSM device's capabilities relating to different multimedia applications are described in Section 2.4.5.
- The interface is capable of using the HSYNC and VSYNC inputs and tracking them – This is an important improvement over some of the previous MSM devices where time consuming tuning of the internal Sync signals had to be done for correct interfacing. As a result, this capability greatly reduces the engineering efforts spent on interfacing and is much more user friendly. In addition, this reduces restrictions and adds more flexibility to the customers' sensor selection, since HSYNC and VSYNC are now allowed to vary.

However, there are a couple of restrictions on the HSYNC and VSYNC inputs that should be noted.

- VSYNC pulse width has to be no less than 5 PCLKs; HSYNC pulse width has to be no less than 2 PCLK.
- The actual number of valid bytes during the active portion of a line has to remain constant. This restriction is not new and is based on the fact that the number of valid pixels given to the VFE should stay constant when in one mode of operation.

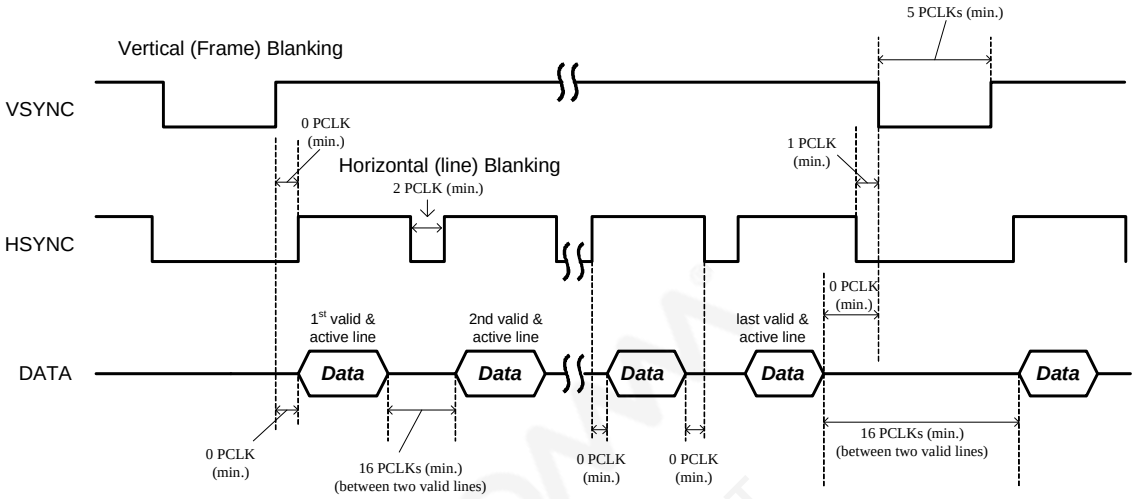


Figure 2-11 Minimum time requirement on HBlank, VBlank, and interval between two valid lines

2.4.5 Multimedia applications using the camera interface

This section describes the different multimedia applications that can be used on the MSM6290 device. The following features use the camera interface, and possibly the MDP.

2.4.5.1 Qcamera

The Qcamera application is the most popular and standard of the multimedia applications. The camera function is the integration of a camera sensor onto the cell-phone module, using the MSM camera interface, to implement a digital camera application that is capable of taking, formatting, and storing digital pictures. The Qcamera application capabilities are summarized in Table 2-4.

Table 2-4 Qcamera capabilities summary

Parameter	Capture	Preview
Interface	10 bits (8 optional)	10 bits (8 optional)
Maximum resolution – frame rate	5 MP at TBD fps[1]	WQVGA – 30 fps[2]

1. FPS is dictated by maximum pixel clock frequency.
2. Adding overlay to display may degrade performance.

2.4.5.2 Qcamcorder

The Qcamcorder application uses the same hardware setup and interface as Qcamera but it is used to record whole segments of digital film. The Qcamcorder application capabilities are summarized in Table 2-5.

Table 2-5 Qcamcorder capabilities summary

Parameter	Preview	Record
Interface	10 bits (8 optional)	10 bits (8 optional)
Maximum resolution – frame rate	QVGA – 30 fps[1]	QVGA video encode – 15 fps

1. Adding overlay to the display may degrade performance.

2.4.5.3 Qtv

The Qtv application is used to decode playback video clips from memory.

Table 2-6 Qtv capabilities summary

Parameter	fps
Video decode – playback	MPEG4/H263/H264: 30 fps QVGA (upscaling to WQVGA) WMV, Real:15 fps QVGA
Video decode – streaming	MPEG4/H263/H264:15 fps QVGA WMV, Real:15 fps QCIF

2.4.5.4 Qvideophone

The Qvideophone application is a combination of the Qcamcorder and Qtv applications. It is used for a two-way "videophone" application where a person on one phone can talk to and view another person at the same time the second person can talk to and view the first person. Because it is a two-way application, it requires both video encoding and decoding. The Qvideophone application is capable of 15 fps at QCIF resolution (video simultaneous encode + decode). It can also support QCIF to QVGA upscaling.

2.5 SD/MMC card interface

2.5.1 Introduction

A secure digital (SD) memory card is a flash-based memory card specifically designed to meet the security, capacity, performance, and environmental requirements inherent to newly emerging audio and video consumer electronic devices.

The MSM6290 device support two 4-bit SD (or 1-bit) interface that conforms to the SD physical layer specification. The physical form factor, pin assignments, and data transfer protocol are forward-compatible with the multimedia card (MMC) while including some enhancements. These include higher bandwidth, copyright protection (via secure version), and functional expansion through SDIO. Secure digital card communications are based upon an advanced 9-pin interface consisting of a clock, a bidirectional command pin, four or one bidirectional data pin(s), and three power pins.

The MSM6290 IC device's secure digital card controller (SDCC2) is implemented using an ARM PrimeCell PL180UN core that provides deserialization of data and buffering between the SD card and the system. The PL180 is modified to include additional features, such as:

- Flow control: clock stopping when the system is not able to keep up with the SD bus
- SDIO support: primarily passing interrupts from the SDIO device to the processor
- DM interface and support: a rate controlled request acknowledge interface of the data mover

The *SD Physical Layer Spec* (v 2.0) requires an MCLK range of 0 to 52 MHz, yielding a maximum transfer rate of 208 Mbits/s. SD cards, however, use primarily flash-based memory so the overall throughput is limited by the flash-memory performance and CPU overhead.

The following subsections describe the SD/MMC interface of the MSM6290 device, its requirements, and its capabilities. Refer to the *Application Note MultiMedia Card/Secure Digital Card* (80-V7837-1) for details on SD/MMC.

2.5.2 Interface description

The MSM6290 device has a 4-bit SD interface as shown in Figure 2-12. It supports 4 bits of data and a command signal. In addition, a master clock output is provided by the MSM device to be used as SDCC_CLK or MMC_CLK. This clock is designed for use with the SD interface and is what customers should use with the SD/MMC cards. Figure 2-12 and Table 2-7 illustrate and describe the SD/MMC interface.

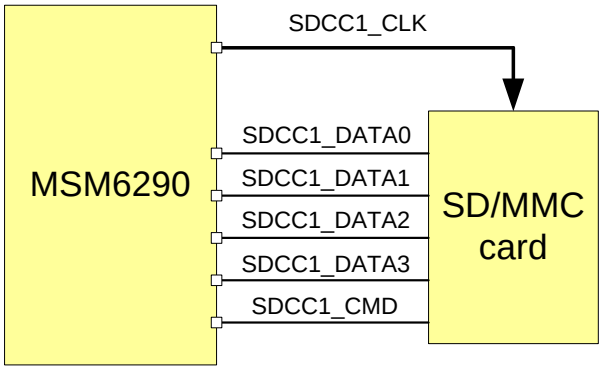


Figure 2-12 SDCC1 and MMC interface connections on the MSM device

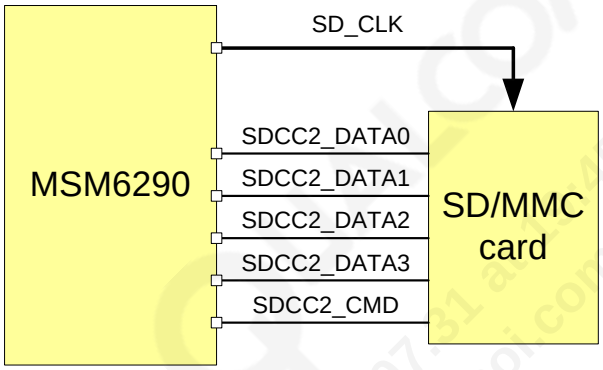


Figure 2-13 SDCC2 interface connections on the MSM device

Table 2-7 SDCC1 interface signals and GPIOs used

GPIO#	I/O	SD signal name	MMC signal name
GPIO[30]	I/O	SDCC1_CMD	MMC_CMD
GPIO[31]	O	SDCC1_CLK	MMC_CLK
GPIO[32]	I/O	SDCC1_DAT[0]	MMC_DATA[0]
GPIO[99]	I/O	SDCC1_DAT[1]	MMC_DATA[1]
GPIO[100]	I/O	SDCC1_DAT[2]	MMC_DATA[2]
GPIO[101]	I/O	SDCC1_DAT[3]	MMC_DATA[3]

Table 2-8 SDCC2 interface signals and GPIOs used

GPIO#	I/O	SD signal name
GPIO[43]	I/O	SDCC2_CMD
GPIO[44]	O	SDCC2_CLK
GPIO[28]	I/O	SDCC2_DAT[0]
GPIO[24]	I/O	SDCC2_DAT[1]
GPIO[23]	I/O	SDCC2_DAT[2]
GPIO[22]	I/O	SDCC2_DAT[3]

2.5.3 Interface capabilities

As mentioned, the SD interface supports SDCC1 and SDCC2 (according to the SD physical layer specification 2.0) up to 4-bit data mode. The SDCC1 interface is also capable of supporting 4-bit MMC according to MMC specifications 4.2. While the same hardware controller is used (the SD host controller block integrated within the MSM), the initialization for SD cards and MMCs are different. The AMSS will auto-detect which card is inserted (SD or MMC, or no card) and will proceed accordingly.

A second secure digital card controller (SDCC2) port has been added to the MSM6290 device to support the SD memory card and SDIO WLAN functionality simultaneously. It supports high-speed mode operations.

The SD card features are:

- Supported standards

Table 2-9 Supported standards

	1-bit	4-bit	Max operating frequency (MHz)
SDIO v2.0	Yes	Yes	50
SD v2.0	Yes	Yes	50
MMC v4.2	Yes	Yes	52

- Two SD card interfaces – SDCC1 and SDCC2
- Both the interfaces support high-speed mode operations
- Support for simultaneous use of SD memory card and SDIO WLAN
- SW configurable edge sampling (falling or rising)
- Clock gating for power saving (and a power-saving option to always turn the clock off when bus is idle)
- Flow control option to prevent overflow and underflow
- SDCC_CLK output up to 52 MHz
- Support for 2.5 V to 3.0 V operation

2.5.4 Clock output

As mentioned, the MSM device supplies a master clock output through GPIO31, the SDCC1_CLK or GPIO44 the SDCC2_CLK. This master clock is designed to work with the SD interface and can go up to 52 MHz.

The SDCC_CLK output is configured by the SDCC_CLK_MD and SDCC_CLK_NS registers. These registers are used to select a source and then divide it using either a modulo divider (div2 or div4) or an M/N:D counter. The M/N value is basically the fraction of the source frequency that the output frequency should be. The D value is used to select a specific duty cycle and for a 50% duty cycle D should be equal to N/2.

For the cleanest output, an even divisor (i.e., source divided by 2, 4, 6, 8, etc.) should be used or at least whole integer divisors. If odd fractions are used, the output frequency will have jitter, which will vary depending on the M and N values used. It is important to make sure the required frequency can be achieved with jitter that is acceptable to the clocked device (SD or MMC card).

NOTE When required, applications engineering will help customers verify and optimize the M, N, and D values. Send technical questions to https://support.cdmatech.com.

2.5.5 MMC/SD card detection

Software based detection (polling) mechanism is implemented in AMSS.

Card insertion: AMSS checks for the presence of a card by periodically sending commands only when the application calls for it through the CMD line. In other words, even though the card is inserted, the card does not do anything until the application calls for it.

Card removal: AMSS checks for the removal of the card by periodically sending commands. If there is no response, then the card was removed or is malfunctioning.

NOTE For the hardware detection mechanism, it is necessary to install a 1 kΩ resistor on the DATA3 line. Refer to the *MSM6290 Baseband Reference Schematic* (80-VF866-41) for proper connections. AMSS does not support this feature. It only supports a polling mechanism for card insertion and removal.

2.5.6 SDIO interface

The 4-bit SD interface is also capable of supporting SDIO applications. The hardware supports SDIO according to the SDIO card specification version 2.0, including interrupt support in all modes.

3 ARM Microprocessor and Peripherals

The MSM6290 device contains an embedded ARM926EJ-S microprocessor and supporting peripherals. The supporting peripherals include the memory and peripheral interface controller, sleep controller, clock controller, watchdog controller, and watchdog circuit. The ARM926EJ-S used in the MSM6290 device is a high-performance, low-power microprocessor. Some of the features of the ARM microprocessor include a five-stage pipelined RISC architecture, both 32-bit ARM and 16-bit THUMB instruction sets; a 32-bit address bus; and a 32-bit internal data bus. For more information about using the ARM926EJ-S, see the Advanced RISC Machines publication *ARM926EJ-S Data Sheet* (document number ARM DDI 0029E.) The ARM processor is configured for little-endian mode and supports a 32- or 16-bit external data bus.

3.1 AHB system

The MSM6290 device support several types of voice, video, and data features. Because of this, the MSM6290 require significant processing power and memory-access bandwidth. The MSM6290 advanced high-performance bus (AHB) system is leveraged from the MSM6200® AHB system with modifications that increase performance.

The MSM6290 device uses the ARM926EJ-S microprocessor, which has a target maximum speed of 297.8 MHz. (This speed supports all multimedia features available for MSM6290.) The ARM926EJ-S brings increased processing power by using a Harvard cache architecture that provides a processor subsystem that includes ARM9EJ-S integer core, a memory management unit (MMU), separate instruction and data SYS AHB bus interfaces, and data caches.

Figure 3-1 illustrates the bus architecture. Master capability (initiating reads or writes) is shown as an arrow from a block to the bus. Slave capability is shown by an arrow from the bus to the block.

The MSM6290 device adds the following new blocks with AHB master capability:

- Video
- Graphics (also has slave port)
- SD controller (DMA)
- Mobile display processor (MDP)

The following slave is added:

- Internal memory (graphics Z buffer, boot RAM, low-power viewfinder)

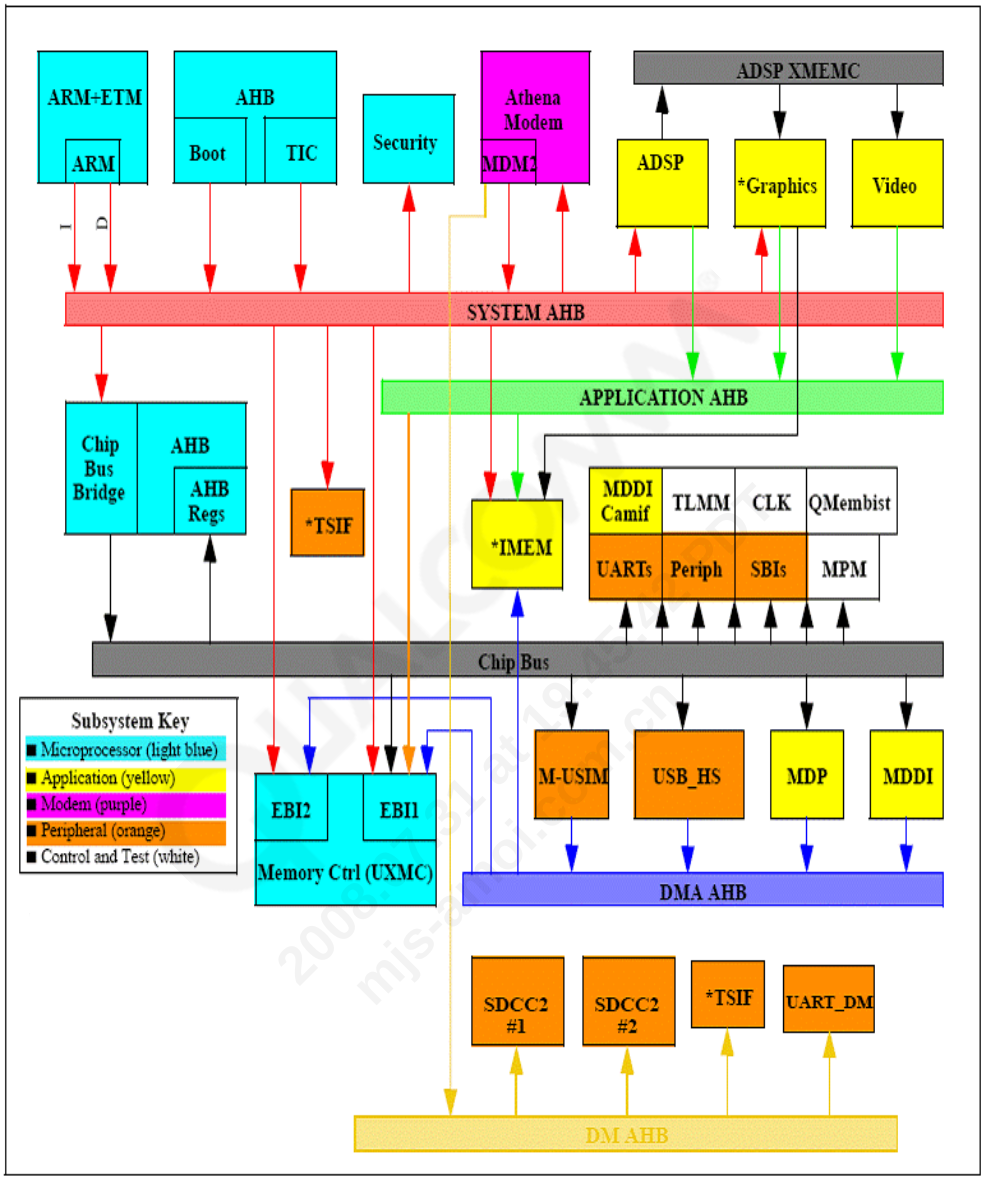


Figure 3-1 AHB system architecture

3.1.1 AHB buses

The architecture employs three AHB buses: system AHB, application AHB, and DMA AHB. The system AHB is dedicated to the ARM9™ for functional operation.
Hardware-register configuration is only accessible from this bus. The application AHB handles multimedia data traffic to, and from, EBI1 and is unaffected by EBI2 accesses.

The DMA AHB handles DMA data transfers controlled by the ARM. Its primary function is to transfer screen updates from EBI1 main memory to an LCD. It is also used for low bandwidth I/O devices (SD card and USB).

3.1.2 Arbitration

There are two levels of arbitration in the system when accessing external devices. A master must first win arbitration to access its AHB bus and then win arbitration in the multi-ported memory controller for EBI1 or EBI2.

3.2 Features

- ARM926EJ-S integration
- ARM926EJ-S operating up to a maximum frequency of 297.6 MHz for MSM6290
- Multi-layer AHB system operating up to a maximum frequency of 75 MHz (for multimedia features) and 61 MHz (for limited multimedia features)
- Two external memory interfaces with arbitration for the multi-layer AHB system and memory controllers
- Support for external memory and peripherals access for the QDSP4u8 processors and the graphics processor
- Support for SDRAM
- Bootup from NAND flash device
- Debugging capabilities at target speeds

3.3 ARM926EJ-S

3.3.1 Features

The ARM926EJ-S is a single-clock, high-performance, Java-enabled, synthesizable core. Its features include:

- Harvard caches
- 16 kB instruction and 16 kB data caches, each four-way associative supported on the MSM6290 device
- Memory management unit (MMU)
- Harvard AHB interface
- ARM and THUMB modes supported
- Jazelle ARM9EJ-S core (Java support)
- Embedded trace macrocell (ETM) support
- Low power features

3.3.2 Burst access

The ARM926EJ-S supports burst transfers to its slaves, specifically for cache operations. Table 3-1 shows the burst transfer types supported.

Table 3-1 ARM burst transfer support

HBURST[2:0]	Description	Operation
SINGLE	Single transfer of word, half-word, or byte	Single transfer of word NC instruction fetches Page-table walk read Continuation of a burst that either lost grant or received a split/retry response
INCR4	4-word incrementing burst	Half-line cache writeback Instruction prefetch (if enabled) 4-word burst NCNB, NCB, WT, WB write
INCR8	8-word incrementing burst	Full line cache writeback 8-word burst NCNB, NCB, WT, WB write
WRAP8	8-word wrapping burst	Cache line fill

Burst transfers coupled with the use of page mode, burst mode, and SDRAM memories help to increase system performance.

3.4 Memory map and memory map decoder

The memory map indicates how the different slaves are assigned memory address ranges on the system. The memory map decoder or decoder in an AMBA system is used to perform a centralized address decoding function. This decoding generates select lines to each of the system bus slaves, indicating that a read or write access to the slave is required.

The MSM6290 device supports *only* trusted boot.

In trusted boot mode, the MSM6290 boots from its internal boot ROM located at 0xFFFF0000. Additional details are provided in Section 3.7.

3.4.1 Memory map

Table 3-2 Memory map (for NAND boot)

Memory block	Base address	Size (MB)
BOOT MODE=1 (NAND)		
XMEM1_CS_N[2]	0x00000000	128
XMEM1_CS_N [3]	0x08000000	128
XMEM1_CS_N[0]	0x10000000	128
XMEM1_CS_N[1]	0x18000000	128
LCD_CS_N	0x20000000	128
XMEM2_CS_N[0]	0x28000000	128
XMEM2_CS_N[1]	0x30000000	128
XMEM2_CS_N[2]	0x38000000	128
XMEM2_CS_N[3]	0x40000000	128
UART_DM	0x48000000	128
SDCC1	0x50000000	128
NAND	0x60000000	128
MOT_EST	0x68000000	128
ADSP	0x70000000	128
IMEM	0x78000000	128
MSM CORE	0x80000000	128
TSIF	0x88000000	128
MODEM	0x90000000	128
HDLC	0x98000000	128
CRYPTO	0xA0000000	128
GRAPHICS	0xA8000000	128
RESERVED	0xB0000000	128
SDCC2	0XB8000000	128
RESERVED	0XC0000000	256

3.5 Reset and pause

The reset pause block is responsible for:

- Generating a synchronous reset for the ARM and other advanced microcontroller bus architecture (AMBA) peripherals
- Providing status information concerning the source of the last reset (pin or watchdog generated)
- Providing a software-controllable mechanism for stalling the ARM for a specified amount of time

This block generates reset signals for the ARM and the system. It also controls the clock to the ARM microprocessor, providing the ability to reset or pause the ARM. Software can use the pause mechanism to halt the microprocessor for a specified amount of time.

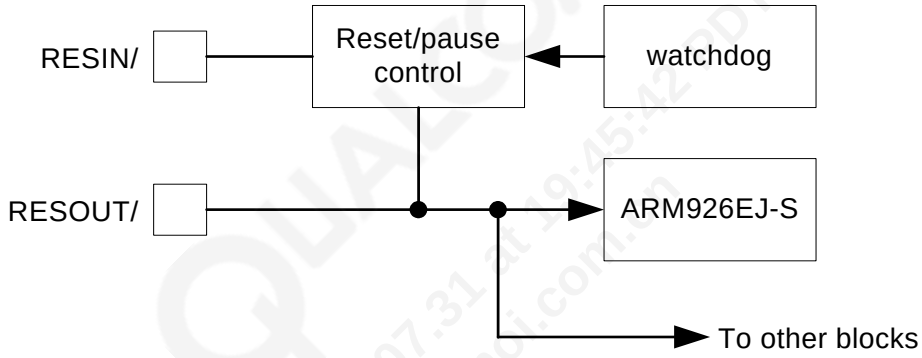


Figure 3-2 Reset generation

This block combines the two reset signals, power-on reset (RESIN_N) and watchdog reset, and provides a synchronously de-asserting reset pulse to the rest of the chip. HRESET_N is guaranteed to be asserted for a minimum of two CLK pulses regardless of the duration of the RESIN_N pulse. The source of the last reset is stored in bit 0 of register RESET_STATUS.

This block also contains the circuits to pause the ARM. This can be done by de-asserting HREADY which stalls the microprocessor until HREADY asserts.

Software pauses the processor by writing to a 16-bit counter through register PAUSE_TIMER. This action immediately de-asserts HREADY for 4 cycles. ARM is paused for 4 + PAUSE_TIMER periods. For example, if TCXO is a 19.2 MHz clock (52 ns period), then writing the value 10 to PAUSE_TIMER pauses the microprocessor for 52 ns * 4 + 10 * 52 ns = 728 μs.

This timer is meant for introducing small delays in the system software. Note that if the microprocessor clock frequency is sufficiently low, delays introduced by this timer can exceed the watchdog timer duration resulting in a watchdog reset of the system.

3.6 Watchdog timer

The watchdog timer is a 21-bit counter running on the sleep controller clock regime that enables the mobile station to recover from unexpected hardware or software anomalies. Unless the microprocessor periodically resets the watchdog timer, the watchdog timer resets the mobile station. The watchdog timer is disabled and reset by grounding the WDOG_EN pin. Asserting the RESIN_N pin also resets the watchdog timer. The watchdog timer is enabled by leaving the WDOG_EN pin unconnected (it has an internal pull-up) or by connecting it to V_{DD_P}. The watchdog timer is disabled by hardware as soon as the ARM processor enters debug mode (indicated by the DBGACK pin of the processor transitioning from low to high). To re-enable the watchdog timer, apply a system reset to clear the state of the internal signal wdog_disable.

The watchdog circuit pulses the signal WATCHDOG_EXPIRED when the watchdog timer has expired. In NATIVE mode, this signal is combined with the RESIN_N pin to generate RESOUT_N and the internal reset for the MSM device.

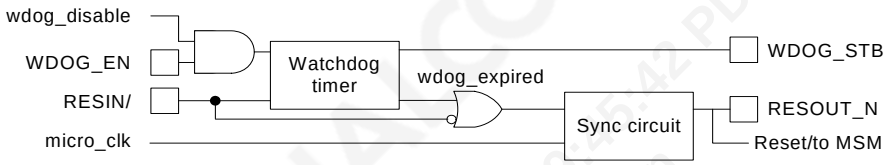


Figure 3-3 Watchdog timer configuration

3.6.1 Sleep mode

In sleep mode, the microprocessor cannot reset the watchdog timer. The sleep controller contains an auto-kicker that resets the watchdog every 32 sleep clock cycles. The output of the sleep controller auto-kicker is also routed to output via the WDOG_STB pin. To enable the auto-kicker, the microprocessor must write a sequence of 1, 0 to the SLEEP_CTL:AUTO_KICK_ARM bit. When the sleep counter reaches its terminal count, the auto-kicker circuit is disabled.

3.6.2 Non-sleep mode

In non-sleep mode, the microprocessor must reset the watchdog timer at least once every 213 ms. If the microprocessor does not reset the watchdog timer in this time, the watchdog timer expires and asserts an internal RESIN_N signal to the system. The WDOG_EXPIRED signal lasts between 53.3 ms and 106 ms (53.3 ms for a 4.8 MHz clock rate and 53.3 to 106 ms when the sleep crystal is used). To reset the watchdog timer, the microprocessor must write a sequence of 1, 0, 1 to the SLEEP_CTL:WATCH_DOG bit.

3.6.3 General-purpose timer operation

When the microprocessor is powered down after programming a general-purpose timer delay that exceeds the watchdog timer interval, the general-purpose timer auto-kicker must be enabled to prevent the watchdog from resetting the ARM microprocessor. The auto-kicker is disabled when the general-purpose timer reaches its terminal count.

3.7 Boot methodology

3.7.1 Introduction

In AMSS 6290 software and firmware, images are stored in NAND flash memory. Both 8-bit and 16-bit NAND flash are supported. There are two external memory interfaces on chip, EBI1 and EBI2. EBI1 supports SDRAM, while EBI2 supports NAND flash.

Normal non-trusted boot is not supported on the MSM6290 device; only trusted boot is supported. To support trusted boot, an on-chip BOOT ROM has been added. When trusted boot mode is enabled, regardless of the type of flash used, the MSM device boots from on-chip BOOT ROM. If secure-boot efuse is blown, the boot code copies the AMSS image from the flash, hashes the image, and authenticates the image. Because the image has to be authenticated before use, the code is considered trusted, free of viruses and hacker code. Booting directly from flash compromises security because flash is subject to tampering. If the secure-boot efuse is not blown, the authentication process is skipped by software, and a secure platform cannot be provided, even if booting from on-chip ROM.

3.7.2 AHB bus memory map

The memory map indicates how the different slaves are assigned memory address ranges on the system. The memory map decoder, or decoder in an AMBA system, is used to perform a centralized address decoding function. This decoding generates select lines to each of the system bus slaves, indicating that a read or write access to the slave is required.

For trusted boot, the chip always boots from internal boot ROM (mapped at 0xFFFF0000 to 0xFFFFFFFF), then the primary boot loader (PBL) in ROM downloads the secondary boot loader (SBL) from NAND flash.

The BOOT_MODE pin value should be 1, indicating that the chip should boot from NAND flash; the decoder maps the memory regions differently.

When booting from NAND flash, the two RAM1 regions (RAM1(0) and RAM1(1)) are mapped to the lowest addresses, so that an SDRAM can be used at RAM1(0), to which the ARM processor can copy the entire AMSS software from the NAND flash. The decoder must properly decode accesses under the different configurations allowed by these options.

Table 3-3 lists the HSEL signal descriptions.

Table 3-3 MSM6290 decoder HSEL signal descriptions

HSEL signal	Address range	Description	Size
HSEL_ROM(0)	0x0000 0000 - 0x07FF FFFF 0x1000 0000 - 0x17FF FFFF	Boot mode 0 Boot mode 1	128 MB
HSEL_ROM(1)	0x0800 0000 - 0x0FFF FFFF 0x1800 0000 - 0x1FFF FFFF	Boot mode 0 Boot mode 1	128 MB
HSEL_RAM(0)	0x1000 0000 - 0x17FF FFFF 0x0000 0000 - 0x07FF FFFF	Boot mode 0 Boot mode 1	128 MB
HSEL_RAM(1)	0x1800 0000 - 0x1FFF FFFF 0x0800 0000 - 0x0FFF FFFF	Boot mode 0 Boot mode 1	128 MB
HSEL_LCD	0x2000 0000 - 0x27FF FFFF	LCD/color LCD controller	128 MB
HSEL_GP0	0x2800 0000 - 0x2FFF FFFF	General purpose memory space	128 MB
HSEL_GP1	0x3000 0000 - 0x37FF FFFF	General purpose memory space	128 MB
HSEL_GP2	0x3800 0000 - 0x3FFF FFFF	General purpose memory space	128 MB
HSEL_GP3	0x4000 0000 - 0x47FF FFFF	General purpose memory space	128 MB
HSEL_UART_DM	0x4800 0000 - 0x4FFF FFFF	UART1 data mover	128 MB
HSEL_SDCC1	0x5000 0000 - 0x5FFF FFFF	SDCC1	256 MB
HSEL_NAND	0x6000 0000 - 0x 67FF FFFF	NAND controller	128 MB
MOT_EST	0x6800 0000 - 0x 6FFF FFFF		128 MB
HSEL_ADSP	0x7000 0000 - 0x77FF FFFF	Select for application QDSP4 (memories and registers)	128 MB
HSEL_IMEM	0x7800 0000 - 0x7FFF FFFF	Internal memory	128 MB
HSEL_MSM	0x8000 0000 - 0x87FF FFFF	Select for bus peripherals	128 MB
HSEL_TSIF	0x8800 0000 - 0x8FFF FFFF	TSIF	128 MB
HSEL_MODEM	0x9000 0000 - 0x97FF FFFF	Modem	128 MB
HSEL_HDLC	0x9800 0000 - 0x9FFF FFFF	HDLC	128 MB
HSEL_CRYPTO	0xA000 0000 - 0xA7FF FFFF	Crypto engine	128 MB
HSEL_GRAPHICS	0xA800 0000 - 0xAFFF FFFF	Graphics slave	128 MB
RESERVED1	0xB000 0000 - 0xB7FF FFFF	Reserved region 1	128 MB
HSEL_SDCC2	0xB800 0000 - 0xBFFF FFFF	SDCC2	128 MB
RESERVED4	0xC000 0000 - 0xFFFF FFFF	Reserved area	256 MB

3.7.3 Boot pin requirements

Three pins are used during boot for the MSM6290 device:

- BOOT_MODE: input pin that controls the source of boot code. NAND flash via EBI2 if ‘1’.
- BOOT_MODE2: input pin that indicates if the NAND flash in the system has 8-bit IO (if ‘0’) or 16-bit IO (if ‘1’).
- BOOT_MODE3 input pin that indicates if normal boot (if ‘0’) or trusted boot (if ‘1’). Normal boot is not supported so BOOT_MODE3 must always be set to ‘1’.
- RESIN_N: input pin that causes the MSM to reset. In reset state when ‘0’ and out of reset state when ‘1’.
- DP_TX_DATA: output pin that indicates the boot status when booting up from NAND. This pin functions as the UART data output pin when not in NAND boot mode.

The definitions of the MSM6290 device boot pins are summarized in Table 3-4.

Table 3-4 Bootup pins

Pin name	Direction	Value	Functions
BOOT_MODE	IN	‘0’	■ Reserved
		‘1’	■ Selects to boot from the NAND Flash on EBI2. ■ Forces the ARM926 processor to start executing code from location 0xFFFF0000, which is aliased to the on-chip BOOT SRAM if TRUSTED_BOOT pin is ‘0’; aliased to the BOOT ROM if TRUSTED_BOOT pin is ‘1’. ■ Forces the DP_TX_DATA pin to be used by the boot sequencer to report boot- up status right after power-on reset. Software can later program the DP_TX_DATA pin to be used by UART as the data output pin after the bootup process is completed.
BOOT_MODE2	IN	‘0’	8-bit NAND Flash
		‘1’	16-bit NAND Flash
BOOT_MODE3	IN	‘0’	Not supported.
		‘1’	Execute trusted boot from BOOT ROM aliased to 0xFFFF0000.
RESIN_N	IN	Active low	When asserted (‘0’), it puts the MSM hardware in poweron reset mode. The deassertion of this pin (‘1’) initiates the bootup process.
DP_TX_DATA	OUT	‘0’	When this pin remains at ‘0’ (not toggling), it indicates that the boot sequencer is still in the process of loading the first block of data (ARM’s boot code) from the FLASH, and has not encountered an error condition yet. This pin is powered on to this value when BOOT_MODE = ‘1’.
		‘1’	When this pin remains at ‘1’ (not toggling), it indicates that the boot sequencer has finished loading the first block of data (ARM’s boot code) form the NAND Flash to the on-chip boot SRAM without errors and ARM has been released from reset.
		HCLK/2	When this pin toggles at HCLK/2 frequency, it indicates that the boot sequencer has aborted its operation due to a page-read error condition detected by the NAND Flash controller right after reading a page. Usually it is due to uncorrectable ECC errors in the page.
		HCLK/4	When this pin toggles at HCLK/4 frequency, it indicates that the boot sequencer has aborted its operation due to receiving an AHB ERROR response when accessing the NAND Flash controller.
		HCLK/8	When this pin toggles at HCLK/8 frequency, it indicates that the boot sequencer has aborted its operation due to not receiving the “op_done_irq” interrupt signal from the NAND Flash controller for 8k TCXO cycles after the boot sequencer activates a NAND Flash device reset or a page read. For TCXO running at 20 MHz, 8 k cycle takes 409,600 ns.
		HCLK/16	When this pin toggles at HCLK/16, it indicates that the trusted boot has failed. The ARM9 is required to write to a register in the Security block to indicate that trusted boot failed.

Note: The values of BOOT_MODE, BOOT_MODE2, and BOOT_MODE3 pins are captured in registers on the rising edge of the RESET_N signal. The MSM6290 device use a boot mode based on the value of BOOT_MODE, BOOT_MODE2, and BOOT_MODE3 pins. These values are captured by registers that cannot be programmed or accessed. The boot mode cannot be altered once ARM starts the bootup process.

3.7.4 Hardware support

To accommodate trusted boot, the boot sequencer in MSM6290 device is shown in Figure 3-4.

3.7.4.1 Boot sequencer

The finite state machine in the boot sequencer examines BOOT_MODE, BOOT_MODE2, and BOOT_MODE3 states upon MSM reset. Depending on the value of BOOT_MODE (NAND boot) and BOOT_MODE3 (trusted boot), the boot sequencer determines to boot from BOOT ROM or NAND flash.

3.7.4.2 Failed trusted boot register

A single bit in the FAILED_TRUSED_BOOT register indicates that authentication failed. The ARM9 is required to write to this register indicating that trusted boot failed. This indication is put out on the DP_TX_DATA pin by toggling that pin at the frequency HCLK/16.

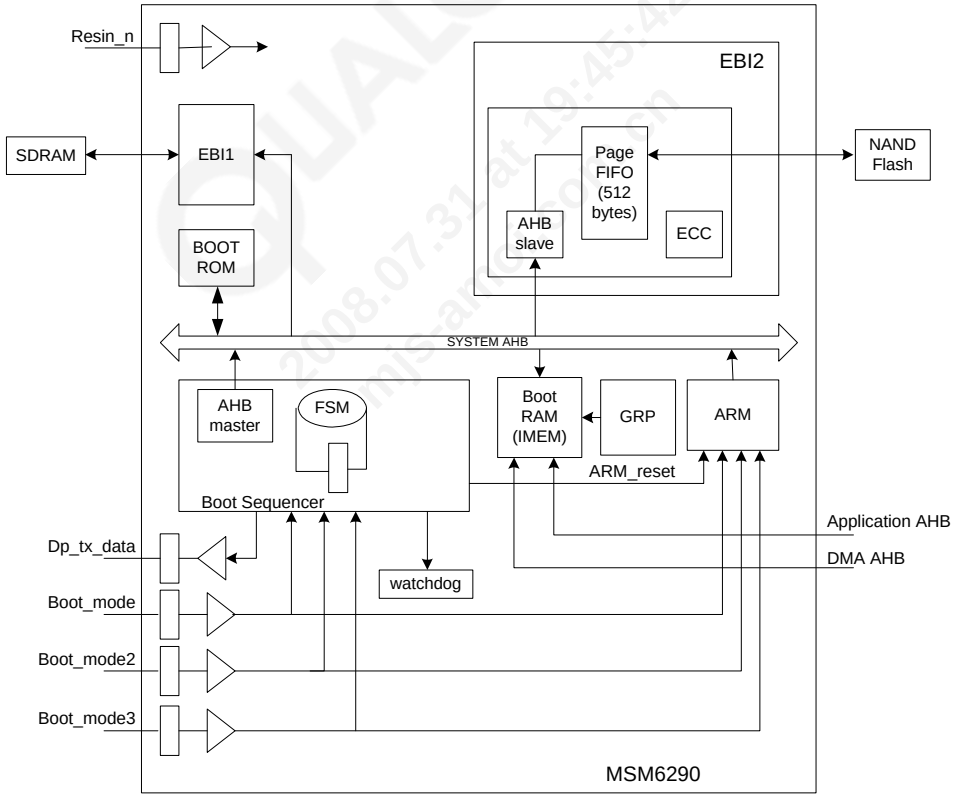


Figure 3-4 Trusted boot hardware support

3.7.5 Bootup procedure

3.7.5.1 Trusted boot with NAND flash

This boot procedure applies when BOOT_MODE3 = '1' and BOOT_MODE = '1'. During this boot mode, BOOT ROM is mapped to 0xFFFF0000, and IMEM is mapped to its normal address space, 0x78000000.

The boot-up procedure is described as follows:

1. The boot sequencer takes ARM out of reset and makes ARM to start execution from 0xFFFF0000.
2. ARM executes the primary boot loader residing in BOOT ROM, starting at 0xFFFF0000.
3. The primary boot loader instructs the NAND controller to use:
   - 8-bit NAND mode if BOOT_MODE2 = 0
   - 16-bit NAND mode if BOOT_MODE2 = 1
4. The primary boot loader performs auto-detection of NAND page size (512 vs. 2 k), and NAND type (NAND vs. SuperAND).
5. The primary boot loader copies config data from NAND to IMEM.
6. The primary boot loader hashes the secondary boot loader in ARM and copies it to SDRAM.
7. The primary boot loader performs three RSA calculations in ARM.
   a. Performs RSA decrypt on secondary signature and extracts hash.
   b. Compares extracted hash with original.
   c. Hashes attestation certificate and compares against RSA decrypt on the signature.
   d. Hashes intermediate certificate and compares against RSA decrypt on the signature.
8. The primary boot loader transfers control to the secondary boot loader.
9. The secondary boot loader hashes AMSS using SHA-1 accelerator (resides in Crypto Engine) and copies it to SDRAM/SRAM.
10. The secondary boot loader performs three RSA calculations in ARM.
    a. Performs RSA decrypt on secondary signature and extracts hash.
    b. Compares extracted hash with original.
    c. Hashes attestation certificate and compares against RSA decrypt on the signature.
    d. Hashes intermediate certificate and compares against RSA decrypt on the signature.
11. Hands over control to AMSS.

3.7.6 Other requirements

Every time the poweron reset is asserted then de-asserted, the whole boot-up process is always activated. The poweron reset can be caused by the resin_n pin or the watchdog.

- TCXO (19.2 MHz) is the only clock available to MSM internal hardware during the boot-up process, and is used as the source of HCLK.
- Only NAND flash devices with a page size of 512 bytes and 2048 bytes are supported. The NAND flash devices with a page size of 256 bytes are not supported.
- NAND devices with I/O width of both 8 bits and 16 bits are supported.
- It is a requirement on the NAND flash vendors that the defects in the first 16 pages of data in the NAND flash be within ECC's correction capability (1 and 4 bit per ECC codeword). In current NAND flash controller design, each ECC codeword is 128 bytes long, so each page contains four code words. The ECC correction capability is therefore 4-bit errors per page evenly distributed into the four code words.
- The MMU is not turned on until the entire application software has been transferred to SDRAM.
- Two chip selects are supported for NAND devices.

3.7.7 MSM device reset scheme summary

- The sync logic synchronizes the trailing edge of resin_n to the HCLK. This logic resides in the reset_pause block in the amba_ahb.
- When the boot_mode pin is 0 and boot_mode3 pin is either 0 or 1, the resin_n pin can directly reset the ARM processor.
- When the boot_mode pin is 1 and boot_mode3 pin is 1 (trusted boot), the resin_n pin can directly reset the ARM processor.
- The boot_mode2 pin indicates the I/O width of the NAND flash. Boot_mode2=0 indicates 8-bit I/O. Boot_mode=1 indicates 16-bit I/O.
- Both the resin_n pin and the watchdog (wdog_expire signal) can cause the MSM to reset.
- When the mode[1] pin is 1 (test mode), the ARM processor is held in reset.

4 Memory Interface

The external bus interface 1 (EBI1) is a high-performance bus that supports a wide variety of memories. With a higher performance ARM9 microprocessor in the MSM6290 device, the system memory bandwidth becomes a bottleneck in running memory-intensive applications such as video and graphics. In addition, the liquid crystal display (LCD) accesses consume a significant portion of the memory bandwidth.

The MSM6290 device was designed to provide two distinct memory interfaces. EBI1 was targeted to support high-speed synchronous memory devices. EBI2 was targeted to support slower asynchronous devices, such as the LCD or NAND flash.

The ARM926EJ-S microprocessor is a cached processor, and all the accesses to external memory will be burst accesses of four or eight 32-bit words when the memory region is declared to be cacheable/bufferable. To take advantage of this higher performance microprocessor, the data from the memories have to satisfy the requirements for these burst accesses.

Using the phone's traditional asynchronous memories results in access times of around 70 – 90 ns, which provide very poor performance. Page memories improve access times to 30 – 40 ns; however, they are still not sufficient for the new applications.

The MSM6290 device is meant to provide a flexible memory interface to support high-speed synchronous memories operating at much higher frequencies than possible with asynchronous memories. The memory interface design is also flexible to support the traditional page mode and asynchronous memory devices for a low-power phone design.

- **The MSM6290 software does not support burst PSRAM devices on EBI1.** (Any reference to burst PSRAM devices refers only to hardware capability of the MSM6290 device.)
- **The MSM6290 software does not support MLC NAND.**

4.1 External bus interface 1

4.1.1 Memories supported on EBI1

The following memories are supported on EBI1:

- 16/32-bit SDRAM (low-power SDRAM)
- PSRAM is not supported. There may be potential MIPS issues when running Bluetooth and video telephony concurrently with memory configurations other than 32-bit SDRAM. **Any reference to burst PSRAM devices refers only to hardware capability of the MSM6290 device.There is no system level or software support for PSRAM.**

4.1.2 EBI1 system

The EBI1 system includes two controllers, each targeting different memory types:

- The multiport memory controller (MPMC), also referred to as the PL172 core, is primarily intended to support low-power synchronous DRAM.
- The external memory controller (XMEMC) is primarily intended to support synchronous burst mode devices.

Figure 4-1 shows the hierarchy of the EBI1 block.

The EBI1 core operates at the bus clock (HCLK). This means that any synchronous memories that reside on the EBI1 bus (external to the MSM device) must also be capable of operating at this frequency. For example, if HCLK = 75 MHz, then the SDRAM/burst flash also must be capable of operating at 75 MHz.

The two memory controllers support a wide range of memory devices to enable maximum flexibility in designing an MSM6290 IC-based phone targeting both a low-power and high-performance solution.

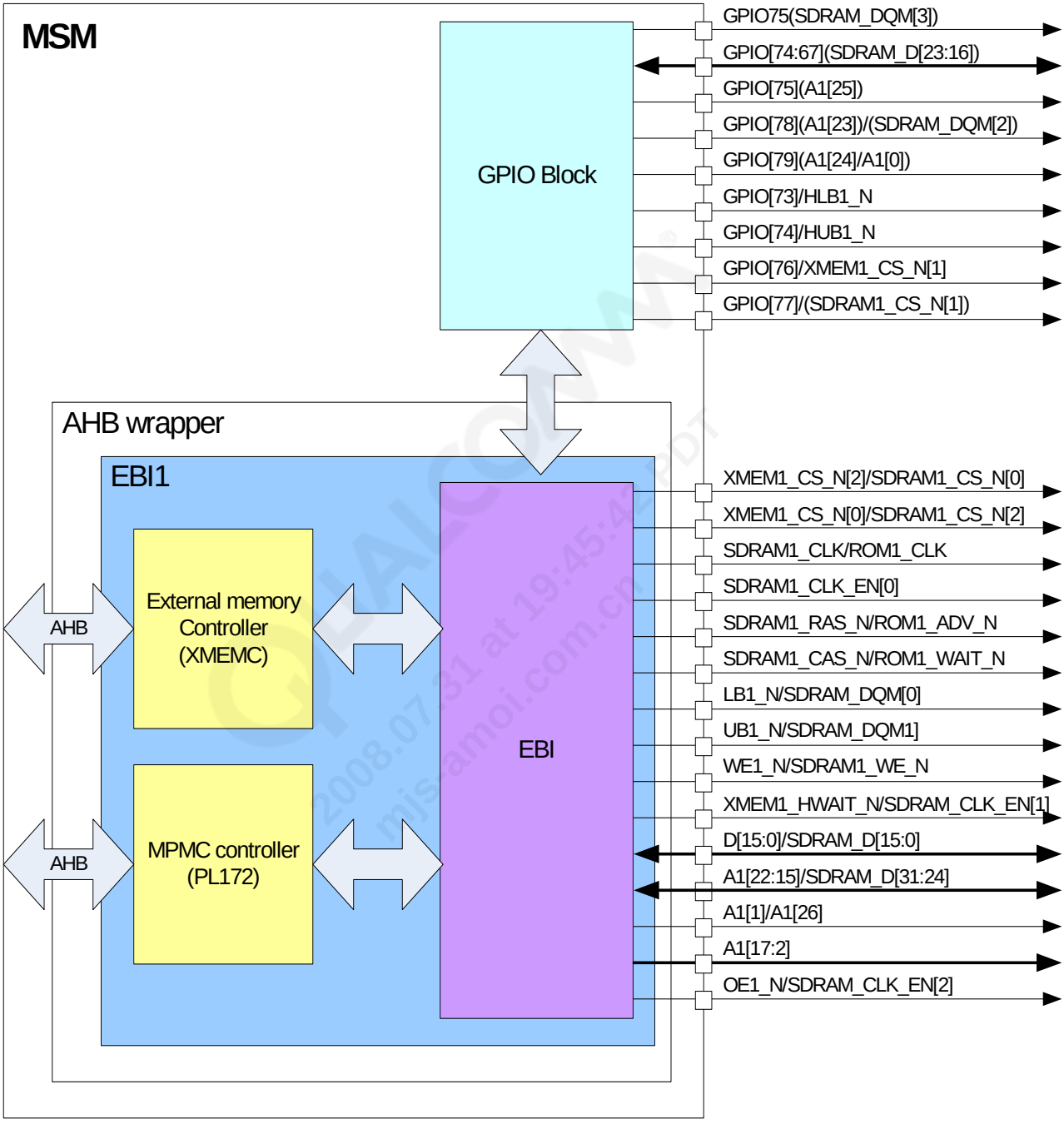


Figure 4-1 EBI1 hierarchy and connections to external memory

Table 4-1 Memories supported on EBI1 by the memory controller

Memory type	Max size per chip select	Chip selects that support this memory	Comments
MPMC			
32-bit SDRAM (monolithic device)	32 MB	SDRAM1_CS_N0, SDRAM1_CS_N1, SDRAM1_CS_N2, and SDRAM1_CS_N3	SDRAM is supported only on RAM chip selects.
32-bit SDRAM (using 2x16 SDRAM devices)	128 MB	SDRAM1_CS_N0, SDRAM1_CS_N1, SDRAM1_CS_N2, and SDRAM1_CS_N3	SDRAM is supported only on RAM chip selects.

Table 4-2 shows the different memory combinations that can be supported, based on the different memories that can be supported by the two memory controllers.

NOTE The primary memory configurations that are targeted for the MSM6290 device are indicated by the shaded regions.

Table 4-2 Supported primary EBI1 memory configurations

Bootup bus	Memory configuration	Active controller	Comments
EBI2	16-bit/32-bit SDRAM + NAND(EBI2)	MPMC	NAND (EBI2) used for code storage, SDRAM used for code execution. All static memories must reside on EBI2.
EBI2	16/32-bit SDRAM + 16-bit OneNAND (EBI2)	MPMC	OneNAND is a device from Samsung that is a storage device with a NOR style interface. The NAND controller in EBI2 is bypassed and all NAND commands are issued directly to the oneNAND device by software.

4.1.2.1 Bootup configuration

On powerup, the XMEM controller will always be the controller that is selected.

- To switch to the MPMC, software will have to write a value of 01 to the EBI1_MEM_CTLR_SEL_CMD register.
- To switch back to the XMEMC again, software will have to write a value of 00 to the EBI1_MEM_CTLR_SEL_CMD register.
- Software can determine which memory controller is active by reading the EBI1_MEM_CTLR_SEL_STATUS register.

Table 4-3 Address map of boot modes

Chip-selects	EBI2 8-bit mode NAND boot	EBI2 16-bit mode NAND boot
EBI1_CS_N[0]	0x1000 0000	0x1000 0000
EBI1_CS_N[1]	0x1800 0000	0x1800 0000
EBI1_CS_N[2]	0x0000 0000	0x0000 0000
EBI1_CS_N[3]	0x0800 0000	0x0800 0000
EBI2_LCD_CS_N	0x2000 0000	0x2000 0000
EBI2_CS_N[0]	0x2800 0000	0x2800 0000
EBI2_CS_N[1]	0x3000 0000	0x3000 0000
EBI2_CS_N[2]	0x3800 0000	0x3800 0000
EBI2_CS_N[3]	0x4000 0000	0x4000 0000

4.1.2.2 Chip selects

EBI1 provides support for the following four chip selects:

- XMEM1_CS_N[0]
- XMEM1_CS_N[1]
- XMEM1_CS_N[2]
- XMEM1_CS_N[3]

The **EBI1_STDY_SEL** register is used to select whether XMEM1_CS_N[3:1] will be static chip selects from XMEMC (poweron-reset value) or dynamic (MPMC).

Table 4-4 indicates which registers are responsible for configuring the parameters corresponding to a particular chip select based on the active memory controller.

Table 4-4 Register names associated with the EBI1 chip selects

Chip select pin name	XMEMC[1]	MPMC
XMEM1_CS_N[0] / SDRAM1_CS_N[2]	EBI1_CS0_CFGx	MPMC_DY_CONFIG2
XMEM1_CS_N[2] / SDRAM1_CS_N[0]	EBI1_CS2_CFGx	MPMC_DY_CONFIG0
XMEM1_CS_N[1] / SDRAM1_CS_N[3]	EBI1_CS1_CFGx	MPMC_DY_CONFIG1
XMEM1_CS_N[3][2]/ SDRAM1_CS_N[1]	EBI1_CS3_CFGx	MPMC_DY_CONFIG3

1. x = 0 or 1.
2. Supports 16-bit SDRAM devices only.

4.1.2.3 Switching memory controllers

Prior to switching from the XMEMC to the MPMC, the following registers must be initialized:

- MPMC_CONTROL
- MPMC_CONFIG

4.1.3

4.1.4 MPMC overview

The MPMC is an ARM memory controller (PL172), which is embedded within EBI1. The purpose of embedding this controller is to support SDRAM memory parts.

The MPMC is setup to support two SDRAM chip selects. Refer to Table 4-4 for details on SDRAM chip selects. MPMC also supports two SDRAM clock enable signals.

4.1.5 Requirements

The MPMC is predominantly used as an SDRAM controller. This section highlights important MPMC requirements useful for understanding, configuring, and using this peripheral.

4.1.5.1 System requirements

The MPMC controller supports most of the SDRAM commands. These commands are issued through configuration registers within the MPMC controller. This section focuses on the description of the self-refresh and power-down modes supported within the MPMC.

Self-refresh mode

The purpose of the self-refresh command is to place the SDRAM into a self-refresh mode. When the SDRAM is placed into a self-refresh mode, it is able to retain its data without external clocking. This is most desired when the MSM device powers itself down.

Software-driven self-refresh

The list below provides step-by-step instructions on how the MPMC controller can place the SDRAMs into a self-refresh mode through software control.

- Program the MPMC configuration register MPMC_DY_CNTL to issue an SDRAM NORMAL operation command (0x0003).
- Program the MPMC configuration register MPMC_DY_CNTL to issue the self-refresh command (0x0007).
- Program the MPMC configuration register MPMC_DY_CNTL to enable clock-gating to SDRAM (0x0040).
- Poll the MPMC_STATUS configuration register (bit [2], 0x4) to observe when the MPMC issued the self-refresh command to the SDRAMs.

At this point, the SDRAMs will be placed into self-refresh mode, and the clock to the SDRAMs will be gated off. The SDRAMs will stay in self-refresh mode until the MPMC is programmed to bring the SDRAMs out of self-refresh mode:

To have the MPMC place the SDRAMs back into auto-refresh mode:

- Program the MPMC configuration register MPMC_DY_CNTL to issue two NOP commands while disabling SDRAM clock-gating (0x0107).
- Program the MPMC configuration register MPMC_DY_CNTL to issue an SDRAM NORMAL operation command (0x0003).

4.1.5.2 Software requirements

16-bit SDRAM initialization requirements

- The SDRAM address bit 0 (SDRAM1_A[0]) is being shared with A1[24] (used for PSRAM) that comes out on GPIO79. This GPIO must be configured correctly prior to beginning the SDRAM initialization.
- The external clock going to the SDRAM (referred to as the MPMCCLK in the ARM document) will be supported in one mode: MPMCCLK = HCLK.
- The SDRAM initialization must be done only after the memory controller has been switched to the MPMC.
- The SDRAM initialization **must be done prior to the cache initialization** so, the cache must be disabled while doing the SDRAM initialization.
- If static memory such as flash or SRAM is being used, then the initialization must be done prior to the memory controller switch.

32-bit SDRAM initialization requirements

In addition to the requirements for the 16-bit SDRAM initialization, the following step must be taken:

- The GPIOs corresponding to DQM[2], DQM[3], SDRAM_D1[23:16] must be configured.

4.1.5.3 Hardware requirements

Powerdown mode support using SDRAM1_CLK_EN [0:3] pins

These pins are used primarily to power down the SDRAMs or configure the SDRAM to operate in self-refresh mode.

It is important to know that when SDRAM MCPs with two chip selects are used, there **must** be two individual clock enable (CKE) inputs as well, one for each SDRAM. This is necessary to be able to independently put all the SDRAMs in self-refresh mode.

MPMC dynamic memory read strategy

The MPMC dynamic memory read strategy defines the way the MPMC controller presents the control signals and data to the SDRAM.

MPMC 16 and 32-bit dynamic read/write timing

The MPMC controller's clock mode is based on the rom1_clk (clock to SDRAM) being in-phase in relation to HCLK. To meet the setup requirements of the SDRAM, the control and data signals for the next memory access are delayed by some delta amount in relation to the rising edge of the rom1_clk (see Figure 4-2). For memory accesses, the control and data signals will have more than a half a clock cycle of setup time to the SDRAM.

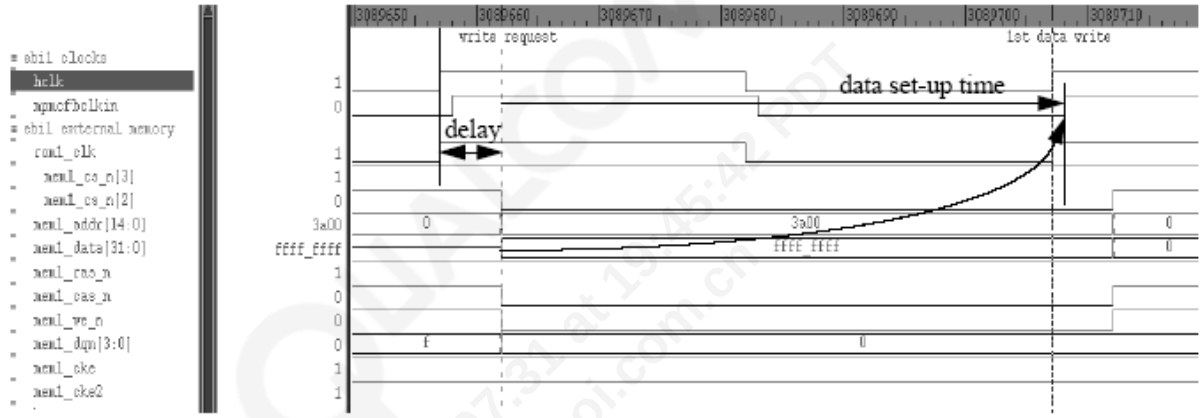


Figure 4-2 MPMC write timing diagram

An example of a read is illustrated in Figure 4-3. In this diagram, the read control signals have over a half a clock-cycle of setup time to the external memory clock (rom1_clk).

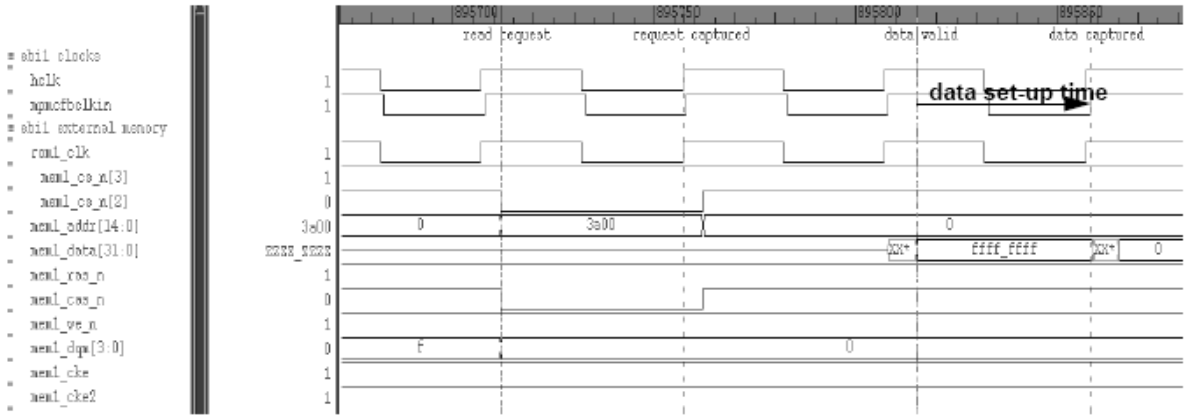


Figure 4-3 MPMC read timing diagram

4.1.6 EBI1 clock skew block

The synchronous memories, such as SDRAM, require a clock to be fed into the memory. This clock is provided by the MSM device, and is needed to ensure that the clock going out to the memory is skewed (delayed) to meet the setup/hold time requirements on the control/address/data signals with respect to the clock.

The EBI1 clock (pin name: ROM1_CLK/SDRAM1_CLK) coming out of the MSM device goes through the path shown in Figure 4-4. The ROM1_CLK clock going out to the external memory is a delayed version of the on-chip clk (HCLK) due to the external loading on the clock line and the delays due to on-chip routing and pads.

For reads from external synchronous memories, the read data coming back into the MSM device will be “registered” using the feed-back clock (FB_CLK). The EBI1 clock skew block applies to both controllers in EBI1: XMEMC and MPMC.

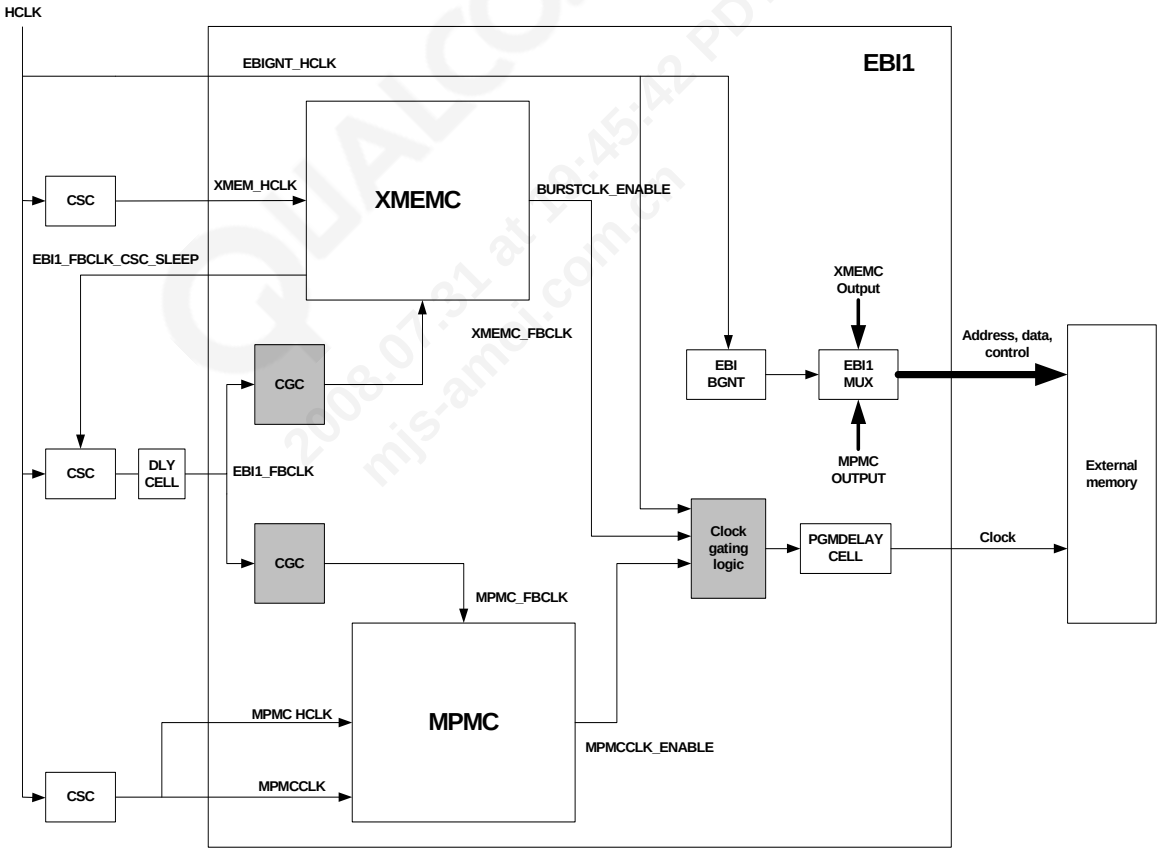


Figure 4-4 EBI1 external clock generation

4.2 EBI2

The EBI2 is one of the MSM6290 device's two external memory interfaces. It is targeted to be the interface for slow peripheral devices (e.g., LCD) and the NAND flash memory.

4.2.1 EBI2 features

- Support for asynchronous flash. Page mode is **not** supported. Programmable wait states for access, hold, and recovery for all chip selects.
- Interface support for byte addressable 16-bit devices (UB_N and LB_N signals).
- EBI2 can be made to operate in HCLK/2 or HCLK/4 modes.
- Support for 8-bit and 16-bit wide NAND flash.
- Support for two chip selects that can be configured as NAND chip selects.
- Support for parallel, port mapped LCD interfaces (Intel and Motorola timing), (18-bit, 16-bit, and 8-bit).
- Support for system word, hword, and byte accesses to 16-bit and 8-bit wide devices on all chip selects (bus sizing supported for all interfaces with the exception of NAND).
- Support for memory accesses (with exception of NAND) by SYS AHB and DMA AHB bus. NAND is accessed through a special controller.
- Support for any generic external peripheral whose interface timing is the same as that of asynchronous memories.
- SW controllable write protect feature (against user mode writes).
- NAND controller change to allow a software programmable command that can be issued to the NAND device.
  - The flexibility of this feature allows the MSM device to support other NAND devices that may require a new command not supported by the controller.
- Support for an EBI2 pin interface for either 1.8 V or 2.6 V pad power supply voltage.

4.2.2 Chip selects

Table 4-5 EBI2 chip selects

Chip select	Memory types supported
XMEM2_CS_N[0]	Asynchronous flash, and primary NAND flash
XMEM2_CS_N[1]	Asynchronous flash, and secondary NAND flash
LCD2_CS_N	Parallel LCD Port map devices with either Motorola or Intel type of interface timing
XMEM2_CS_N[3:2]	Async flash and any other async memory device

4.2.3 Access types

In the MSM6290 IC's AHB system, three different transfer sizes can be attempted: word, Hword, and byte transfers. If the external memory is being accessed, the memory bus width can be equal to, less than, or greater than the transfer size. Table 4-6 shows the transfers supported for each chip select.

Table 4-6 Access types by chip select

Chip select	32x16	16	8ubx16	8lbx16	32x8	16x8	8
XMEM2_CS_N[0] (XMEM controller)	Y	Y	Y	Y	Y	Y	Y
XMEM2_CS_N[0] (NAND controller)	N	Y	N	N	N	N	Y
XMEM2_CS_N[1] (XMEM controller)	Y	Y	Y	Y	Y	Y	Y
XMEM2_CS_N[1] (NAND controller)	N	Y	N	N	N	N	Y
LCD2_CS_N	Y	Y	N	N	Y	Y	Y
XMEM2_CS_N [3:2]	Y	Y	Y	Y	Y	Y	Y

4.2.4 EBI2 system

In order to support all the device types mentioned, the interface requires two different controllers:

- External memory controller (EBI2_XMEM): Used to access LCD and generic asynchronous interface devices
- NAND controller: Used to access NAND flash devices

The interface also allows accesses by multiple requesters. An arbiter controls the access of the requesters to the controllers and to the external memory interface pins. For EBI2 the requesters are:

- SYS AHB bus: It can access both the external memory controller and the NAND controller. The only master on this bus attempting accesses to EBI2 is the ARM processor.
- DMA AHB bus: It can access the external memory controller only.
- NAND controller: This is a different type of requester than the previous two. The NAND controller initiates requests to the EBI2 after being configured by the SYS AHB bus.

Arbitration is required to allow the access of multiple requesters to the external memory interface. The arbiter uses fixed priority schemes that are programmed into two priority modes as shown in Table 4-7. Arbitration requires polling the different requests placed by the different requesters and granting the access to that request with the higher priority. Granting an access requires controlling the data path from the bus to the controller and from the controller to the pins.

Table 4-7 EBI2 priority options

Option	Priority 1 (higher)	Priority 2	Priority 3 (lower)
Mode 0 EBI2_CFG[0] = 0 Powerup default	SYS	DMA	NAND
Mode 1 EBI2_CFG[0] = 1	DMA	SYS	NAND

4.2.5 EBI2 external memory controller (asynchronous devices)

This interface provides access to asynchronous flash and to any memory-mapped peripheral that requires similar timing.

4.2.5.1 Features

- Support of bus sizing operation to allow word, Hword, and byte accesses to 16-bit and 8-bit memories.
- Support for recovery/turnover wait states (RECOVERY) insertion to the beginning of the current chip select access when the previous access was a read to a different chip select, or a when the current access is a write and the previous access was a read to the same chip select.
- Support for address to chip select assertion setup wait states (CS_SETUP) at the beginning of the access. This feature helps support some types of pseudo-RAMs on the market. When accesses to the chip select using these features are consecutive, this feature also provides the high time for chip select required by the mentioned pseudo-RAMs. In general, it can be used to provide extra setup time between address and chip select assertion if needed.
- The WAIT_RD and WAIT_WR wait states require a minimum programmed value of 1. That is WAIT_RD(min) = 1, WAIT_WR(min) = 1.
- Since the RECOVERY and CS_SETUP wait states are applied at the beginning of the access, only the field with the greatest number of wait states takes effect.
- Support for hold time between the we_n signal rising and the address/cs_n/data signals changing (HOLD_WR wait states). A minimum of 1 hold wait state is required (HOLD_WR = 1) for proper operation of the interface.
- Support for hold time between the oe_n signal rising and the address/cs_n signals changing (HOLD_RD wait states). Required by specific types of peripherals. Minimum value required (and most common value to use) is HOLD_RD = 0.

4.2.5.2 Bootup specification

4.2.5.2.1 Using bootup timer

If the BUSY_N pin is tied to logic 1 or is asserted high within 213 ms, the bootup will be delayed until the expiration of the 12-bit counter operating at a frequency of 19.2 MHz.

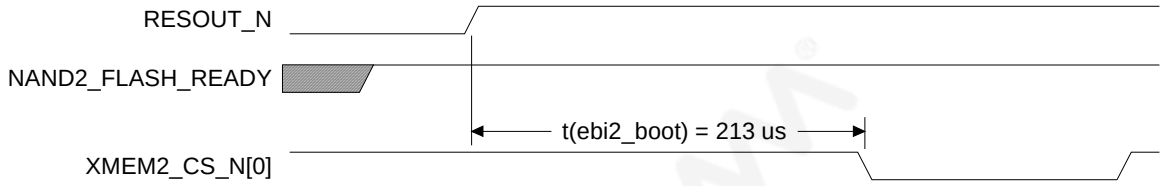


Figure 4-5 EBI2 bootup time specification with NAND2_FLASH_READY tied to logic 1

4.2.5.2.2 Using BUSY_N pin

If the BUSY_N pin is driven to logic 0 for a period longer than 213 ms, the bootup will be delayed until the assertion of logic 1 on the BUSY_N pin. This mode is useful when the boot up time for the PseudoNAND devices exceeds the 213 ms period provided by the timer.

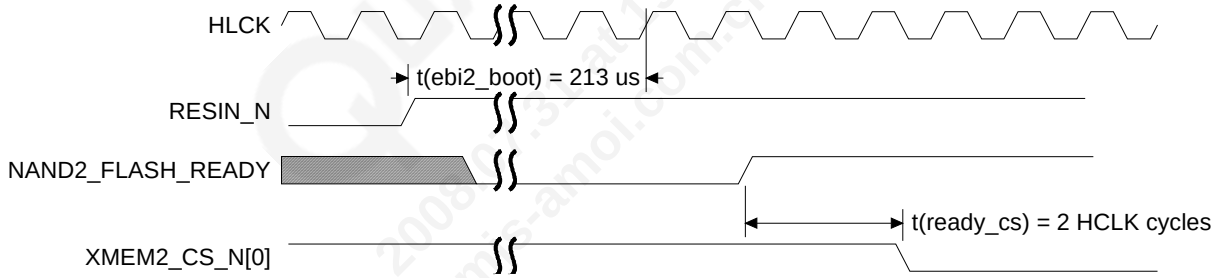


Figure 4-6 EBI2 NAND bootup scheme with NAND2_FLASH_READY monitoring

4.2.5.3 Configuration registers

All EBI2 external memory controller configuration registers are required to be initialized before the chip select is to be used.

Table 4-8 Chip select configuration registers for EBI2 external memory controller

Chip select	Configuration register
XMEM2_CS_N[0]	GP0_CFG0 GP0_CFG1
XMEM2_CS_N[1]	GP1_CFG0 GP1_CFG1
XMEM2_CS_N[2]	GP2_CFG0 GP2_CFG1
XMEM2_CS_N[3]	GP3_CFG0 GP3_CFG0

4.2.5.4 Access description

Any chip select is asserted for:

- (WAIT_RD + HOLD_RD + 1) AHB clk cycles for reads
- (WAIT_WR + HOLD_WR + 1) AHB clk cycles for writes

4.2.5.5 WAIT cycles

WAIT cycles affect the length of **all** accesses to the memory (native or bus sized). With all other types of wait state fields set to zero the access length is WAIT + 1. we_n and oe_n assert for the amount of cycles on the WAIT field.

4.2.5.6 HOLD cycles

The HOLD cycles are extra cycles inserted after WE_N or OE_N deasserting.

For a write, the MSM device data is driven while WE_N is low. Once WE_N deasserts, the data is not driven and keepers retain the data until another driver drives new data. CS_N remains asserted during this time, and ADDRESS remains valid as well.

For a read, the memory data is driven by the memory device while OE_N is low. Once OE_N deasserts, the data is not driven and keepers retain the data until another driver drives new data. However, data is not latched by the MSM device until the end of the HOLD cycles. CS_N remains asserted during this time, and ADDRESS remains valid as well. This feature is not typical, and may be required only by a few special peripherals. In the case when it is used, care must be taking to assure the keepers can keep the data for the HOLD cycles until the MSM device latches it.

4.2.5.7 Recovery cycles

Recovery cycles are used, if needed, at the beginning of an access that follows a read to avoid data bus driver contention between the memory driver just getting off the bus and the new driver of the bus. The RECOVERY cycles applied are those programmed on the memory whose read just finished. During the RECOVERY cycles the ADDRESS of the next access is valid, however, the CS_N and WE_N or OE_N do not assert until the end of the cycles.

The value programmed depends on the specific memory OE_N to data-hi-z parameter.

4.2.5.8 CS_SETUP cycles

CS_SETUP cycles allow for providing ADDRESS setup time to the assertion of CS_N, having an effect similar to that of recovery. During the CS_SETUP cycles, the ADDRESS of the next access is valid, however, the CS_N and WE_N or OE_N do not assert until the end of the cycles.

CS_SETUP affects the beginning of every native access, and only the first access of a bus sized transfer. In this case, both CS_SETUP and RECOVERY cycles are required, only the greater of the two fields takes effect.

4.2.6 External memory controller (LCD support)

The LCD controller on the MSM6290 device is upgraded to offer more flexibility. It is now capable of supporting 24-bit LCD interface (write-only). In the MSM6290 device, there is an assigned chip select, LCD2_CS_N, which supports port-mapped LCD interfaces with either the Motorola (6800) or the Intel (8080) parallel interface types of timing. An additional LCD_EN signal is required for Motorola's interface. The port-mapped (Intel type) parallel LCD device could be connected to any chip select in EBI2. Bus-sizing capability was added to the MSM6290 device to optimize LCD accesses. Since EBI2 supports 8-bit, 16-bit, 18-bit, and 24-bit LCDs (write-only), most of the external LCD controllers are supported.

4.2.6.1 Features

- Support for both Motorola and Intel types of port-mapped parallel LCD interfaces.
- Support of a bus sizing operation to allow 24-bit write accesses to a 24-bit LCD interface.
- Support for access times of up to 2000 ns (at 75 MHz AHB bus speed).
- Separate configuration control for write and read operations.
- For Motorola interface mode, support lcd_enable high times of up to 1000 ns (at 75 MHz AHB bus speed).
- For Intel interface mode, support for a hold time from WR_N deasserting to CS_N/RS/ADDRESS changing and from OE_N deasserting to CS_N/RS/ADDRESS changing.
- For Intel interface mode, support for setup time from RS/ADDRESS stable to CS_N/OE_N/WE_N asserting.
- Support for 24-bit (write-mode only).

4.2.6.2 Configuration registers

Table 4-9 Chip select configuration registers for EBI2 external memory controller (LCD)

Chip select	Configuration register
LCD_CS_N	LCD_CFG0 LCD_CFG1

Note: The MSM device must be configured in 24-bit mode via the LCD_CFG1 register (setting bit 26: LCD_24_BIT_MODE to 1) to use an 18-bit LCD via the EBI2 address bits (A2[2:1]) as D2[17:16].

4.2.6.3 Access description

Intel timing (8080) type of accesses are similar to that of the EBI2 external memory controller asynchronous accesses with the difference that there is not DELTA wait states.

For Intel (8080) timing, any chip select is asserted for the following:

- (WAIT_RD + HOLD_RD + 1) AHB clock cycles for reads and (WAIT_WR + HOLD_WR + 1) AHB clock cycles for writes

  Additional cycles are added to the overall time the transfer occupies the bus through the RECOVERY and CS_SETUP cycles.

For Motorola (6800) type of accesses, the chip select is asserted for:

- (WAIT_RD + 1) AHB clock cycles for reads and (WAIT_WR + 1) AHB clock cycles for writes

  Additional cycles are added to the overall time the transfer occupies the bus through the RECOVERY cycles.

4.2.6.4 LCD_E_SETUP

For Motorola timing, the LCD_E_SETUP field indicates how many cycles exist between the assertion of the LCD_CS_N and the assertion of the LCD_E signal.

4.2.6.5 LCD_E_HIGH

For Motorola timing, the LCD_E_HIGH field indicates for how many cycles LCD_E is asserted.

4.2.7 NAND flash memory interface

4.2.7.1 Overview

The MSM6290 device supports NAND flash memory to improve data storage capability (see Figure 4-9).

The following is an overview of the NAND flash controller:

- Generic NAND flash memory interface supports 16-bit and 8-bit NAND flash devices from manufacturers such as Samsung, Toshiba, etc.
- Both 512-byte page and 2048-byte page NAND devices are supported.
- SLC NAND flash devices supported.
- Software programmable command for NAND device
- Can support two separate NAND devices
- NAND interface uses the EBI2 ports. The NAND interface shares these ports with other external devices, i.e., external SRAM.

- Interface performs one page at a time data transfers to/from NAND device. No sequential page accesses supported.
- Interface has an internal SRAM buffer for one data page (512/2048 bytes) with the ARM processor directly accessing this internal buffer memory through the AHB bus through 32-bit word accesses.
- Error correction coding (ECC): Reed-Solomon code ECC is used to support MLC based devices. The Reed-Solomon code ECC has 4-symbol correction per 512 bytes of data, where a symbol consists of eight bits of user data and two padded zero bits. Therefore, the correction capability is equivalent to four bytes per 512 bytes of user data. A 512-byte page is considered a single ECC codeword, and a 2 KB page is considered four ECC codewords by the ECC engine. ECC function can be disabled by the MSM device processor.

4.2.7.2 Single-level cell technology

The single-level cell (SLC) technology is the original mass storage solid state technology used in flash devices. Every storage transistor has two different states, and therefore contains one bit of information. The SLC is sometimes referred to as 2LC since each transistor has two states. Based on the data reliability level of the SLC technology, the vendors recommend the Hamming code with 1-bit correction per 512 bytes of data as the ECC scheme to achieve acceptable overall data reliability.

4.2.7.3 NAND controller architecture

The NAND flash devices is sharing EBI2 pins with other types of memory devices controlled by XMEMC. The signals from the controller go through the multiplex logic in the XMEMC before they reach to the EBI2 pins. The NAND controller contains five major blocks:

1. **AHB interface**: This block is a slave interface to the 32-bit AHB bus. It contains all the software control/configuration registers and an MPU module.
2. **Command sequencer:** This block controls the flow of the operation requested by an AHB bus master. This block also maintains an REQ/ACK protocol with the DM (Data Mover) to support DM's command queue function.
3. **Buffer controller:** This block controls the FIFO read/write functions of the two 512-byte RAMs.
4. **NAND timing interface:** This block controls the specific timing of the NAND flash interface signals, such as wait states programmed by software. This block also contains the ECC encoder and syndrome computer.
5. **ECC decoder:** This block performs the Reed-Solomon code decoding algorithm and error corrections into the internal buffers. In all other situations, the ARM processor is the only bus master that accesses the NAND controller. The controller provides a set of AHB-accessible registers for the bus masters to configure the controller, activate operations and read operation status. For data transfers to/from the NAND device, the controller also provides two 528-byte AHB-accessible page buffers for the bus masters to use as a data transfer FIFO.

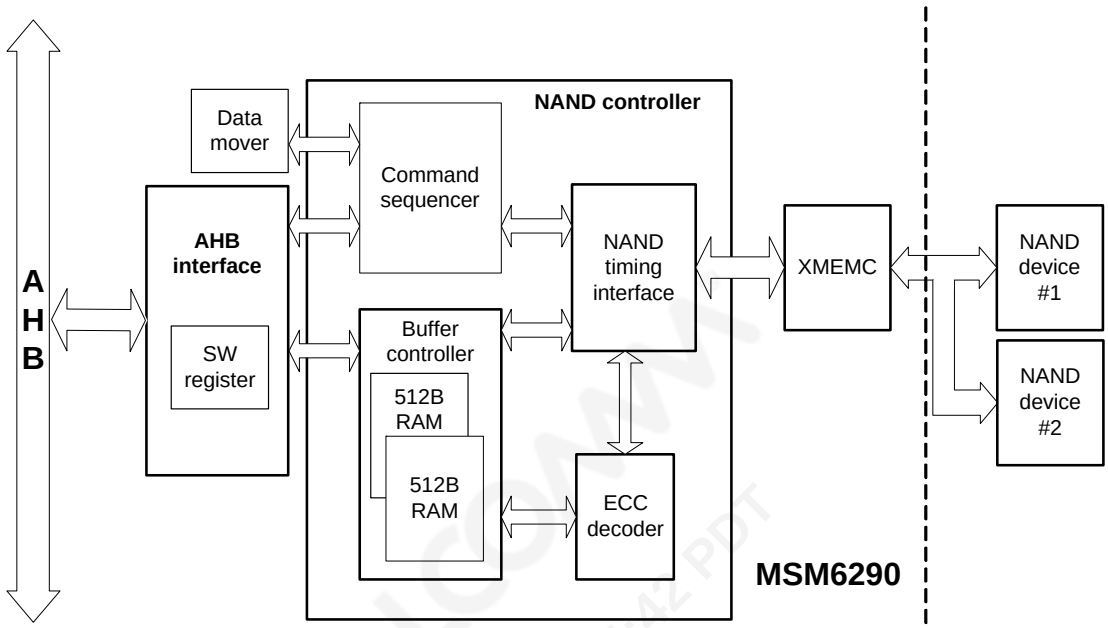


Figure 4-7 NAND controller block diagram

4.2.7.4 Support for two NAND flash devices

The NAND controller treats the whole NAND flash storage space as a series of pages. By default, the controller assumes a single device is used in the system. If there are two NAND devices attached to the MSM device, then the configuration of each device (I/O width and page size) as well as the boundary between the two devices has to be programmed by software. The boundary of the two devices is defined as the first page of the second device.

The controller's access to each device is completely page-based; i.e., once a page access to a device begins, the access must be completed before the controller can start another page access to the same or the other device. Therefore, the controller never operates on two devices simultaneously.

4.2.7.5 ECC for NAND 8/16-bit interface (512-byte page)

The MSM6290 device uses Reed-Solomon coding for error correction and decoding. The ECC module provides Reed-Solomon encoding and error correction (decoding) of the 512 bytes data packets (+ 10 parity bytes for decoder). The module operates in two modes:

1. 8-bits x 512 input block length (+ 8-bits x 10 parity length for decoder)
2. 16-bits x 256 input block length (+ 16-bits x 5 parity length for decoder)

It should be noted that the bad block indicator is moved to word 0 (from word 5) in MSM6290 16-bit NAND interface connected to 512 bytes per page NAND devices.

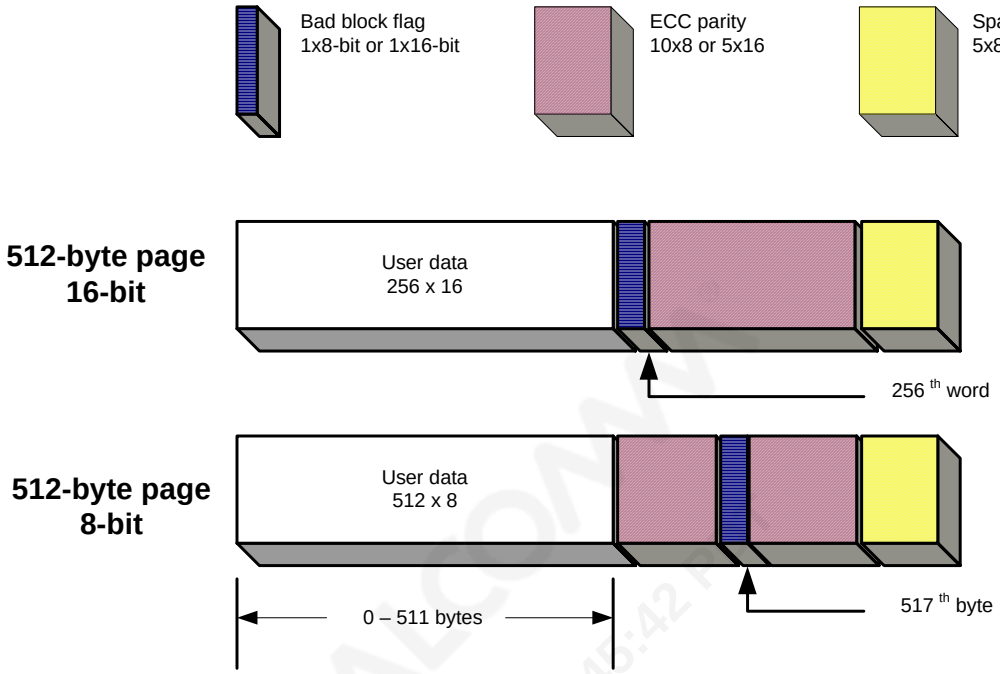


Figure 4-8 Page format description (8-bit/16-bit – 512-byte page)

Table 4-10 ECC allocation – 8-bit NAND interface (512-byte page)

Byte	ECC allocation	Notes
Byte_00	ECC Parity_0	10 parity length for Reed-Solomon decoder (8-bit interface)
Byte_01	ECC Parity_1	
Byte_02	ECC Parity_2	
Byte_03	ECC Parity_3	
Byte_04	ECC Parity_4	
Byte_05	Bad block indicator (0xFF)	
Byte_06	ECC Parity_5	
Byte_07	ECC Parity_6	
Byte_08	ECC Parity_7	
Byte_09	ECC Parity_8	
Byte_10	ECC Parity_9	
Byte_11 to Byte_15	Unused	

Table 4-11 ECC allocation – 16-bit NAND interface (512-byte page)

Word	ECC allocation	Notes
Word_00	Bad block indicator (0xFFFF)	
Word_01	ECC Parity_0	5 parity length for Reed-Solomon decoder (16-bit interface)
Word_02	ECC Parity_1	
Word_03	ECC Parity_2	
Word_04	ECC Parity_3	
Word_05	ECC Parity_4	
Word_6 to Word_7	Unused	

4.2.7.6 ECC for NAND 8/16-bit interface (2kB page)

The ECC module provides Reed-Solomon encoding and error correction (decoding) of the 512 bytes data packets (+ 10 parity bytes for decoder). The module operates in two modes:

1. 8 bits x 512 input block length (+ 8 bits x 10 parity length for decoder)
2. 16 bits x 256 input block length (+ 16 bits x 5 parity length for decoder)

The MSM6290 device has different ECC/spare-area layout than the early devices (as shown in Figure 4-9).

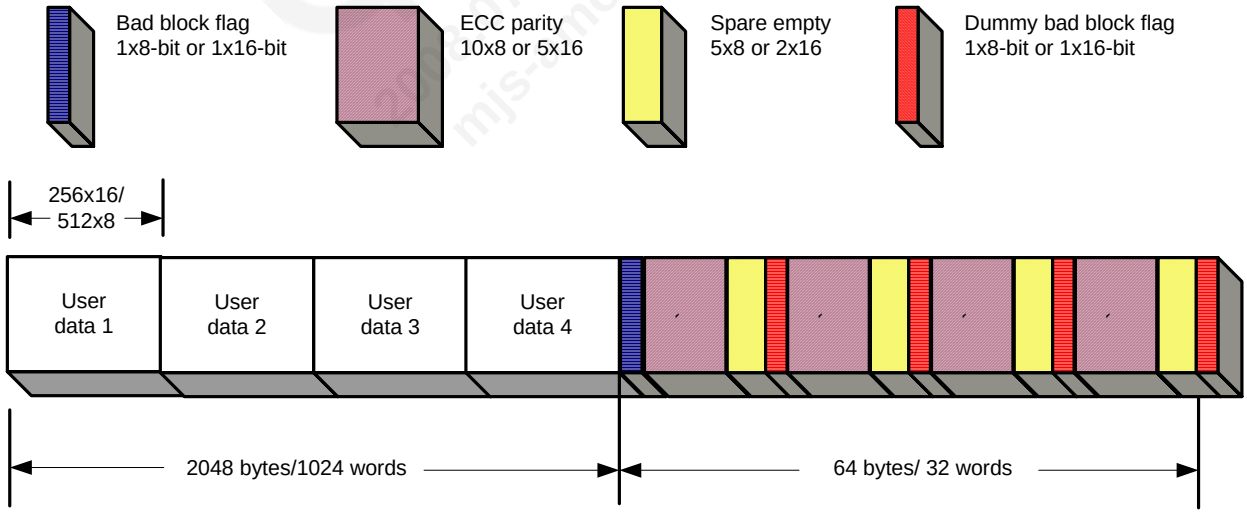


Figure 4-9 Page format – MSM6290 (8-bit/16-bit – 2-KB page)

Table 4-12 ECC allocation in 8-bit NAND interface (2 K page)

Byte	ECC allocation	Notes
Byte_00	Bad block indicator (0xFF)	
Byte_01_Byte_10	ECC0	The bit allocation is same as 8-bit 512-byte page format.
Byte_11 to Byte_15	Unused	
Byte_16	Dummy bad block flag	
Byte_17 to Byte_26	ECC1	The bit allocation is same as 8-bit 512-byte page format.
Byte_27 to Byte_31	Unused	
Byte_32	Dummy bad block flag	
Byte_33 to Byte_42	ECC2	The bit allocation is same as 8-bit 512-byte page format.
Byte_43 to Byte_47	Unused	
Byte_48	Dummy bad block flag	
Byte_49 to Byte_58	ECC3	The bit allocation is same as 8-bit 512-byte page format.
Byte_59 to Byte_63	Unused	

Table 4-13 ECC allocation in 16-bit NAND interface (2 K page)

Word	ECC allocation	Notes
Word_00	Bad block indicator (0xFFFF)	
Word_01 to Word_05	ECC0	Bit allocation is same as 16-bit 512-byte page format.
Word_06 to Word_07	Unused	
Word_08	Dummy bad block flag	
Word_09 to Word_13	ECC1	Bit allocation is same as 16-bit 512-byte page format.
Word_04 to Word_15	Unused	
Word_16	Dummy bad block flag	
Word_17 to Word_21	ECC2	Bit allocation is same as 16-bit 512-byte page format.
Word_22 to Word_23	Unused	
Word_24	Dummy bad block flag	
Word_25 to Word_29	ECC3	Bit allocation is same as 16-bit 512-byte page format.
Word_30 to Word_31	Unused	

4.2.7.7 EBI2 interface access arbitration

The NAND controller shares the same EBI2 interface pins as the EBI2 XMEMC controller. All memory devices (NAND Flash, LCD etc.) that are controlled by these two controllers are connected to the same EBI2 interface. Also, both the SYS_AHB bus and the DMA_AHB bus can access the EBI2 interface through the XMEMC controller. Therefore, there is an arbitration logic in the EBI2 block that arbitrates among the SYS_AHB bus, the DMA_AHB bus and the NAND controller for accessing the EBI2 interface. The SYS_AHB bus has the highest priority and the NAND controller the lowest.

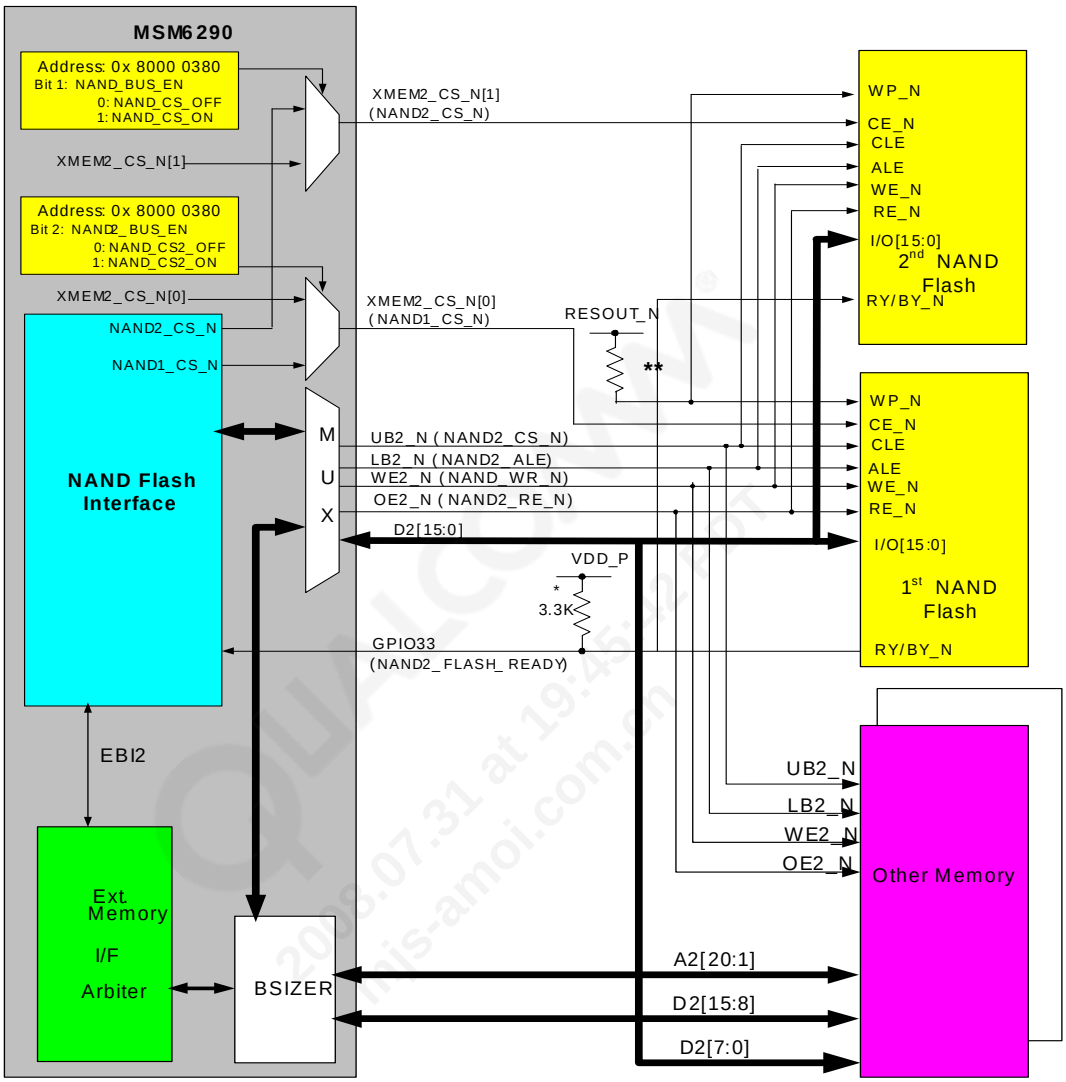
The NAND controller, upon receiving a software command, begins to perform a sequence of read/write transfers to the NAND device through the EBI2 interface pins. Each individual NAND transfer corresponds to a single strobe on the OE2_N pin (read) or the WE2_N pin (write) while the XMEM2_CS_N0 pin (the chip-enable, or CE_N, for NAND Flash) is active. To initiate a transfer, the NAND controller sends the xport_req signal to the EBI2 arbiter to request an access and waits for the arbiter to issue the xport_grant signal.

Once the xport_grant signal is received, the NAND controller performs a single transfer on the EBI2 interface to the NAND device and, during the last clock cycle of this transfer, it asserts the "ready2yield" signal to allow the arbiter to re-arbitrate for the next access. If the xport_grant signal is still asserted in the next clock cycle after ready2yield is asserted, then the NAND controller continues with the next transfer or, if the xport_grant signal is no longer asserted (which means a higher-priority request has been granted), then the NAND controller keeps the xport_req signal asserted until it receives the xport_grant signal and performs the next transfer. This process continues until the NAND controller completes the whole sequence of read/write transfers for the software command.

There is an exception to this rule. In a NAND Flash "page_read" or "flag_read" operation, after the controller completes the last address transfer, the NAND device enters into a "read-busy" state, which is indicated by the ready_busy_n signal being asserted and usually lasts for about 10 to 25 microseconds before the controller is allowed to issue read transfers to read out the page data from the NAND device. Most of the NAND devices on the market do not require the chip-enable, or CE_N signal to remain asserted during the "read-busy" state, but some legacy devices do.

To be backward compatible, the controller continues to support these legacy devices with a special mode of operation which is enabled by setting NAND_FLASH_CFG1[1] to '1.' In this case, when the controller performs a "page_read" or "flag_read," the "ready2yield" signal is not asserted at the end of each transfer until after the "read-busy" state is finished. This way, the NAND controller, once granted by the EBI2 arbiter, would keep the EBI2 interface occupied while it issues the command and address writes to the device and while the device stays in "read-busy" state. Once the "read-busy" state is finished, the controller goes back to the policy of always asserting "ready2yield" signal at the end of each transfer.

For this mode (NAND_FLASH_CFG1[1]='1'), a watchdog timer is enabled to ensure the controller does not get stuck and hog the EBI2 interface for too long if the device "read-busy" state becomes unusually long. The timer starts counting when the NAND device enters into the readbusy state (ready_busy_n='0') during either a "page_read" or "flag_read" operation, and is reset back to zero when the NAND device finishes the "read-busy" state (ready_busy_n='1'). If the "read-busy" is not finished when the counter expires, the NAND controller assumes that the NAND device is non-responsive and aborts the on-going operation, sets bit 31 (page-err) and bit 3 (operation-err) in the NAND_FLASH_STATUS register, sends an op_done_irq interrupt signal to the microprocessor, resets all registers to poweron default state, and forces the NAND controller into idle state. The size of the timer is 13-bit wide, and thus counts 8192 HCLK cycles before expiration. For HCLK = 20 MHz, this time period is 409.6 microsecond. For HCLK = 100 MHz, this time period is 81.92 microsecond.



*** Note: This pull up resistor affects the rise time of the RY/BY_N signal.**
**** Note: Value based on manufacturer specification**

Figure 4-10 Interface connection NAND Flash and MSM6290 device

4.3 OneNAND interface

- OneNAND memories are supported only on EBI2.
- Boot from OneNAND is supported on XMEM2_CS_N[3]
- Currently, AMSS 6290 supports oneNAND on one chip select only. This would limit the maximum density of OneNAND to 2 Gb.
- Only non-multiplexed oneNAND devices are supported.
- oneNAND + NAND flash configuration is not supported in the MSM6290 device.

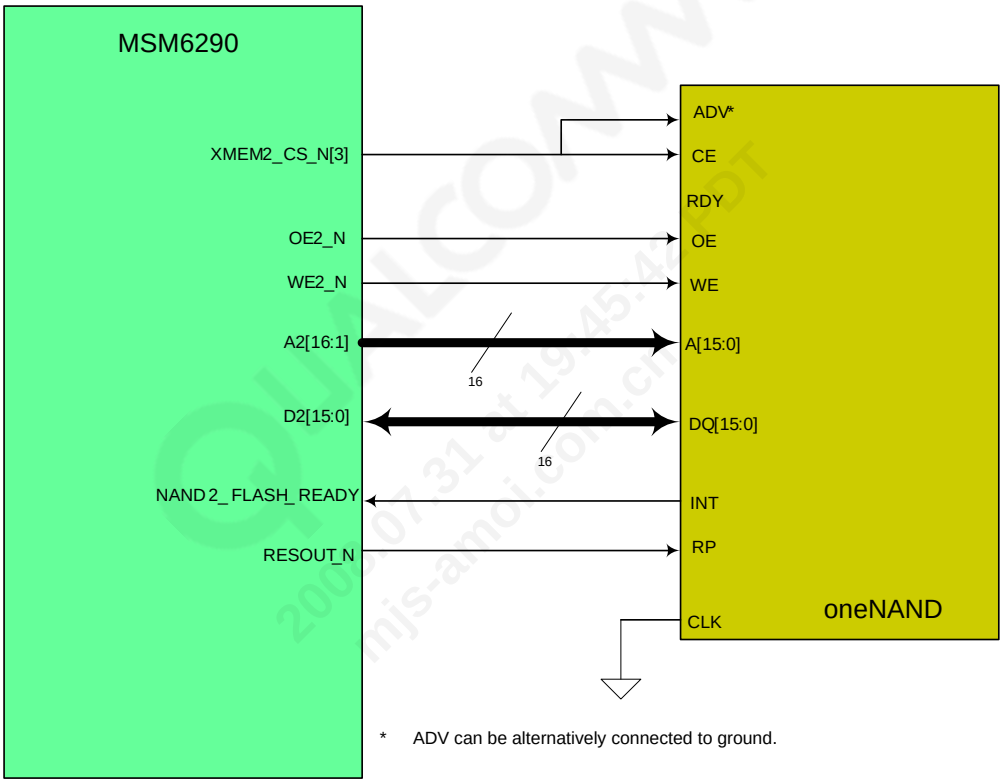


Figure 4-11 MSM6290 OneNAND interface details

5 RF Interface

The MSM6290 devices interface with all MSM6290-series radioOne ICs. There are two planned RF platforms supported within the MSM6290 chipset, schedules of which are available in the software and product release documents. The MSM6290 radioOne chipset includes:

- MSM6290 chipset with Platform F (RFCMOS):
  - RTR6285
  - Support for:
    - US: UMTS 850, AWS, 1900, 2100 (bands 5, 4, 2, 1)
    - JP: UMTS 800, 1700, 1900, 2100 (bands 6, 9, 2, 1)
    - EU: UMTS 900, AWS, 1900, 2100 (bands 8, 3, 2, 1)
    - GSM/GPRS/EDGE 850/900/1800/1900
    - GPS
  - MSM6290 Mobile Station Modem IC
- MSM6290 chipset with Platform G:
  - RTR6285
  - Support for:
    - US: UMTS 850, AWS, 1900, 2100 (bands 5,4,2,1)
    - JP: UMTS 800, 1700, 1900, 2100 (bands 6,9,2,1)
    - EU: UMTS 900, AWS, 1900, 2100 (bands 8,3,2,1)
    - GSM/GPRS/EDGE 850/1800/1900
  - MSM6290 Mobile Station Modem IC

The MSM6290 chipsets support UMTS 850/800 MHz, 900 MHz, AWS, 1900 MHz, and 2100 MHz, and GSM/GPRS/EGPRS 850/900/1800/1900 MHz handset designs. Contact Qualcomm UMTS RF marketing for details on the system configuration. All receivers and transmitters use the radioOne ZIF architecture to eliminate intermediate frequencies, directly converting signals between RF and baseband.

All MSM6290 RFICs are highly integrated and fulfill specific functions; functional requirements are partitioned between the ICs to yield complete, optimal transceiver implementations. Overall MSM6290 performance depends upon the combined, complementary performance of all the ICs in the chipset.

This section describes the MSM6290 RF interconnections, organized by function.

5.1 MSM6290 chipset RF interfaces

Descriptions of the interfaces between the MSM6290 IC and the RTR6285 and RTR6280 RF IC are presented in this section.

5.1.1 SSBI

RFIC operating modes and circuit parameters are MSM-controlled through the proprietary SSBI. The application programming interface (API) is used to implement SSBI commands. The API is documented in AMSS software. See the applicable AMSS software documentation for details.

The SSBI provides efficient initialization, control of device operating modes and parameters, and verification of programmed parameters. The MSM device's SSBI controller is the master while the RFICs and PM IC are slaves. The single-wire SBI signals have the following dedicated function:

- **SSBI1_DATA –** The single-wire interface for RTR6285 and RTR6280 IC
- **SSBI0_DATA –** Unused
- **SSBI2_DATA –** The single wire interface for PM6658 and PM6653 IC

The SSBI1_DATA signal is a direct connection between the MSM6290 IC and the RFIC, routed using standard high-speed bus layout techniques (the bus clock rate could approach 10 MHz). Routing of this signals should avoid sensitive analog, LO, and RF signal traces and circuits. The SSBI2_DATA signal is a direct connection between the MSM6290 ICs and the PM6658 and PM6653 device. The SSBI0_DATA signal is unused.

5.1.2 Receiver functions

Table 5-1 Platform F and G RFIC receive parameters-MSM controlled via SSBI

RFIC	Controlled parameters
RTR6285 and RTR6280	■ Primary operating modes (UMTS, quad-GSM/EDGE, GSM, and either compress mode, warmup, Tx, or Rx/Tx) ■ Acquisition control and detector modes ■ Charge pump parameters ■ Circuits gains, bias currents, and output drive levels ■ Transceiver LO generation mode and parameters ■ Transceiver VCO controls ■ PLL1, PLL2, and PLL3 parameters ■ Rx gain control (LNA, downconverter gain steps) ■ Lock detect is reported to the MSM device through logic circuits and the SSBI ■ Calibration functions

5.1.3 Transmitter functions

The transmit baseband signals are generated by digital-to-analog converter (DAC) circuits within the MSM6290 ICs. Their performance is also highly dependent upon the DAC reference signal provided by the RTR6285 and RTR6280 IC, making the DAC_REF line critical. These five signals are extremely sensitive, analog signals and must be treated accordingly in the PCB layout:

- **I_OUT** – Transmitter in-phase analog output (positive); a differential signal that complements I_OUT_N.
- **I_OUT_N** – Transmitter in-phase analog output (negative); a differential signal that complements I_OUT.
- **Q_OUT** – Transmitter quadrature analog output (positive); a differential signal that complements Q_OUT_N.
- **Q_OUT_N** – Transmitter quadrature analog output (negative); a differential signal that complements Q_OUT.
- **DAC_REF** – Reference input to the MSM Tx data DACs. The DAC_REF line is bypassed to VDD_A with a 2200 μF capacitor located near the MSM pin. Bypassing this reference signal to the supply voltage has proven to be more effective than bypassing to ground.

5.1.4 GSM transmitter functions

5.1.4.1 GSM PA control signal

- **GSM_PA_EN** – PA enable signal for GSM PA.
- **GSM_PA_BAND** – Select GSM PA band.
- **PA_POWER_CTL** – PA power control DAC output. The analog control pin to control GSM PA controller.
- **PA_PWR_CTL_M** – Not needed.
- **GSM_PA_DAC_REF** – GSM PA DAC reference input.

5.1.5 UMTS transmitter functions

The RTR6285 and RTR6280 IC is supplemented by off-chip circuits to complete the transmit paths. The UMTS transmit path includes an RF bandpass filter, power amplifier, isolator, duplexer, and the antenna switch. The following control signals are required for the UMTS transmit output devices:

- **PA_ON[2:0]** – Turns the appropriate PA on and off as required to support the desired operation while consuming minimum DC current. PA_ON transitions high (active) a programmable interval before the beginning of a transmission, and remains high for a short time after the transmission ends.
  - PA_ON0 is used for the UMTS 1900 or 2100.
  - PA_ON1 for UMTS 850 MHz.
  - PA_ON2

- **PA_RANGE[1:0]** – Control signals that step the active PA mode and bias, greatly reducing DC current consumption under low-power operating conditions. PA_R0 is used to switch the PA power supply between the battery and the PMIC switcher. PA_R1 is used to switch the mode of the WCDMA PAs.

5.1.6 Off-chip VCTCXO functions

The voltage-controlled temperature-compensated crystal oscillator (VCTCXO) provides the reference frequency for all RFIC synthesizers as well as numerous functions within the MSM6290 ICs. This reference signal is buffered by the PM6658 or PM6653 IC and applied to the MSM device via its TCXO pin. *The MSM6290 chipset requires a 19.2 MHz nominal VCTCXO frequency.*

The VCTCXO frequency is controlled by the MSM6290 frequency tracking circuit's TRK_LO_ADJ PDM signal in the same manner as the transmit gain control. A two-pole RC lowpass filter is recommended on this control line:

- The first RC segment should be located near the MSM6290 ICs and provide a 1.22-millisecond time constant (R = 2.0 kΩ, C = 0.033 μF). This is the dominate-pole, providing significant rejection of the digital waveform's AC components.
- The second RC segment should be located near the VCTCXO and provide a 1-microsecond time constant (R = 100 Ω, C = 0.01 μF). This pole has little affect on system performance, but removes high frequency noise picked-up as the control signal is routed across the PCB.

The filtered PDM signal results in an analog control signal into the VCTCXO tuning port whose voltage is directly proportional to the density of the digital bit stream. The MSM device varies the pulse density to change the analog control voltage that sets the oscillator frequency — all within a feedback control loop that minimizes handset frequency drift relative to the network.

The PM6658 IC controls the handset powerup sequence, including a special VCTCXO warmup interval before other circuits are turned on. This warmup interval (as well as other TCXO controller functions) is enabled by the MSM6290 IC's TCXO_EN line, a direct connection between the MSM and PM ICs. The PM6658 VREG_TCXO regulated output voltage is used to power the VCTCXO and is enabled before most other regulated outputs.

- **TCXO** – Frequency reference input signal from the VCTCXO, buffered by the PM6658 IC.
- **TRK_LO_ADJ** – Digital PDM bit stream that is filtered to create an analog control signal; minimizes VCTCXO frequency drift relative to the network.
- **TCXO_EN** – Control signal that enables the PM6658 or PM6653 TCXO controller circuits. Use MSM6290 GPIO[94] for TCXO_EN.

5.1.7 Antenna control signal

- **ANT_SEL[3:0]_N** – Antenna switch module control signals

Refer to the Platform F RF schematic for control signals.

5.1.8 GRFC allocation

GRFCs will have different functions depending on the RF platform that is being used. Table 5-2 shows the function and GRFC mapping.

Table 5-2 GRFC allocation on Platform F

GRFC no.	Pin name	RFCMOS platform F
GRFC[14]	GPIO[80]	Not used
GRFC[13]	GPIO[14]	Not used
GRFC[12]	GPIO[103]	Not used
GRFC[11]	GPIO[102]	Reserved
GRFC[10]	TX_ON	TX_ON
GRFC[9]	GPIO[12]	ANT_SEL3_N
GRFC[8]	GPIO[11]	ANT_SEL2_N
GRFC[7]	GPIO[10]	ANT_SEL1_N
GRFC[6]	GPIO[9]	ANT_SEL0_N
GRFC[5]	GPIO[8]	GSM_SAW_SW_MODE
GRFC[4]	GPIO[7]	DRX_MODE_SELECT_C
GRFC[3]	GPIO[6]	INTER_RX_SW_SEL2
GRFC[2]	GPIO[5]	GSM_PA_BAND
GRFC[1]	GPIO[4]	GSM_PA_EN
GRFC[0]	GPIO[3]	INTER_RX_SW_SEL1
–	GPIO[2]	PA_ON1
–	GPIO[29]	PA_ON2
–	GPIO[41]	DRX_MODE_SELECT_A
–	GPIO[65]	DRX_MODE_SELECT_B
–	GPIO[19]	INTER_TX_SW_SEL2
–	GPIO[0]	INTER_TX_SW_SEL1

5.1.9 PMIC interface

The PM6658 and PM6653 IC have a REF_OUT pin to provide PA DAC reference voltage for the MSM6290 devices.

The MSM6290 pin AD26 (GSM_PA_DAC_REF) must connect the PM6658 or PM6653 pin B4 (REF_OUT) with a 0.1 µF capacitor.

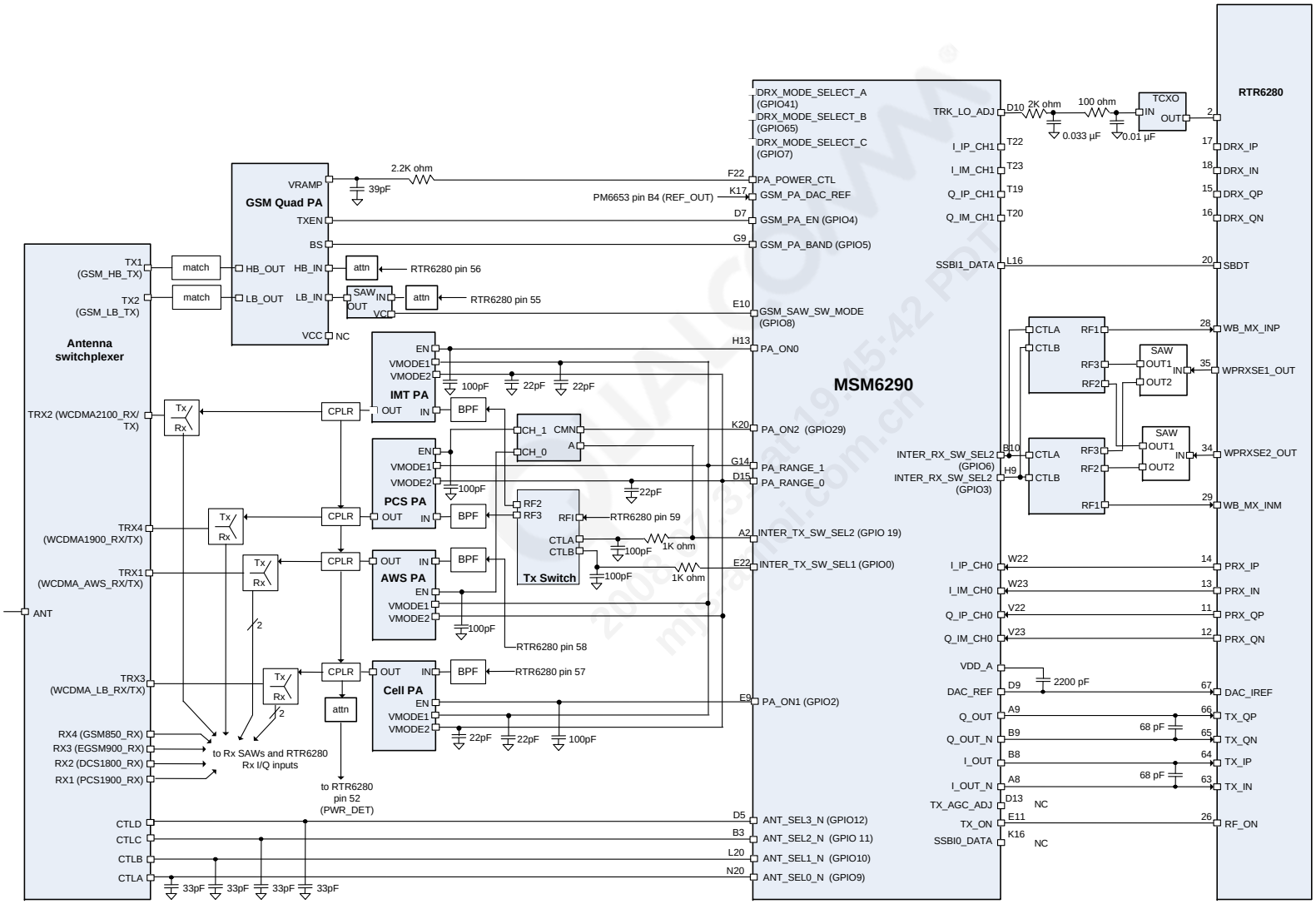


Figure 5-1 Platform G RF interface diagram

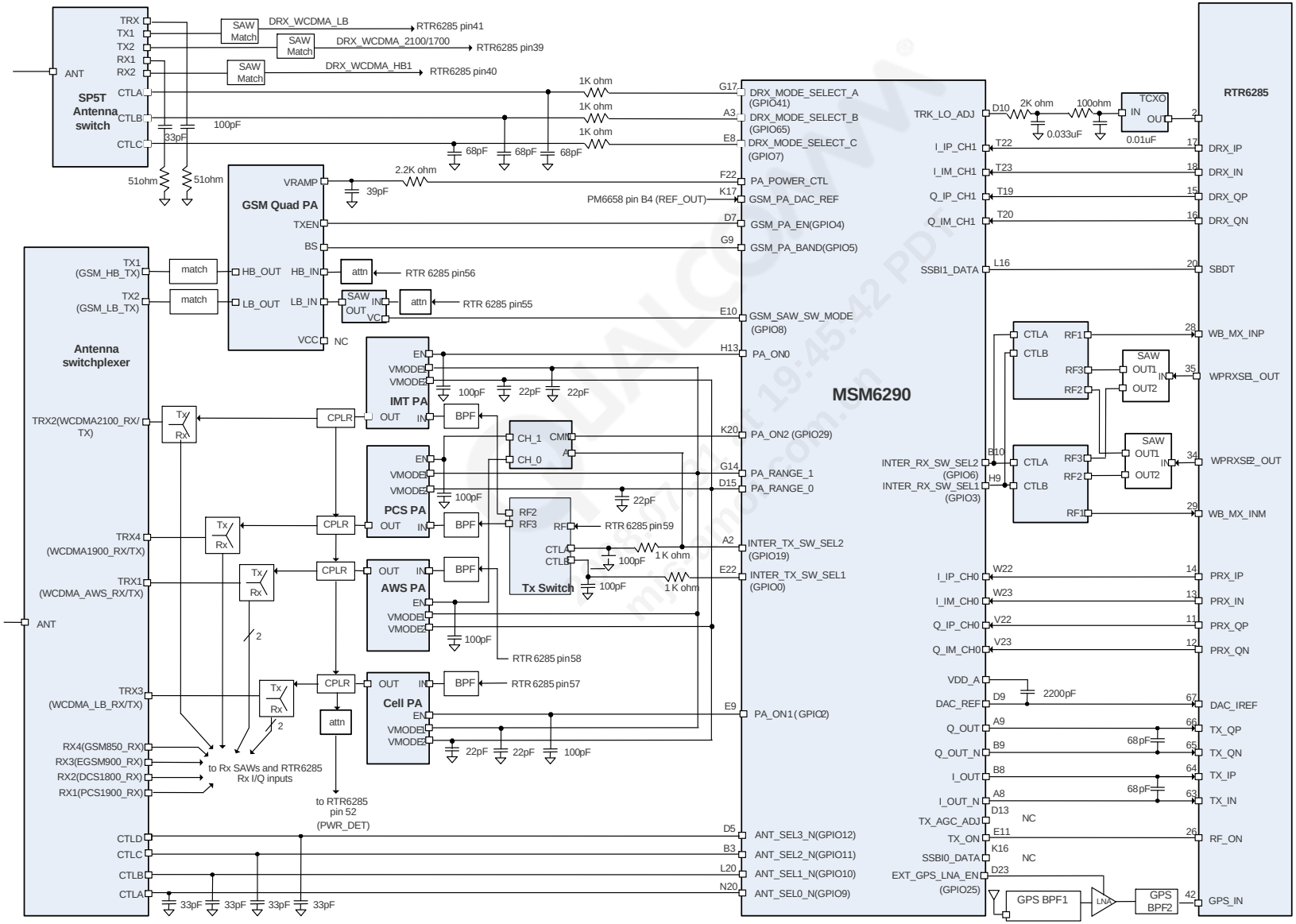


Figure 5-2 Platform F RF interface diagram

6 Stereo Wideband Codec

NOTE All diagrams showing MSM pin numbers refer to the NSP package, unless otherwise specified.

The MSM6290 audio front end is comprised of the stereo wideband codec, PCM interface, and additional DSP audio processing.

The stereo wideband codec allows the MSM6290 device to support stereo music/ringer melody applications in addition to the 8 kHz voice band applications on the forward link. In the audio transmit path, the device operates as 13-bit linear converter with software selectable 8 kHz and 16 kHz sampling rate. In the audio receive path, the device operates as a software selectable 13-bit or 16-bit linear converter with software selectable 8 kHz, 16 kHz, 22.05 kHz, 24 kHz, 32 kHz, 44.1 kHz, or 48 kHz sampling rate. Through software, the Rx path can be configured as either a mono or stereo output.

New to the MSM6290 device is a transmit (Tx) ADC path that now supports stereo wideband sampling.

The integrated codec contains all of the required conversion and amplification stages for the audio front end. The codec operates as a 13-bit linear codec with the transmit (Tx) and receive (Rx) filters designed to meet ITU-T G.712 requirements. The codec includes a programmable sidetone path for summing a portion of the Tx audio into the Rx path. An on-chip voltage/current reference is provided to generate the precise voltages and currents required by the codec. This circuit requires a single capacitor of 0.1 µF to be connected between the CCOMP and GND pins. The on-chip voltage reference also provides a microphone bias voltage required for electret condenser microphones typically used in handset applications. The MICBIAS output pin is designed to provide 1.8 V DC while delivering as much as 1 mA of current. Audio decoder summing and headset switch detection are included.

The codec interface includes the amplification stages for both the microphone and earphone. On the transmit (Tx) path the interface supports two differential microphone inputs, a differential auxiliary input, and a stereo line input. On the receive (Rx) path the interface supports one differential earphone output, a stereo single-ended headphone output, one differential auxiliary output and stereo single-ended line outputs.

The codec is configured by the codec SBI registers. The codec interface is shown in Figure 6-1.

Also part of the audio front end is the PCM interface. The PCM interface allows for an external codec to be used instead of the internal codec. This interface can be used in I2S mode which will allow for an external stereo DAC to be used.

Finally the audio front end includes additional DSP audio processing that does gains, filtering and other audio processing.

The DSP audio processing is configured through the QDSP5000 command types and is not directly controlled by the microprocessor.

Details of all on-chip audio functions (Rx and Tx) and their interfaces to the external transducers (earphones or speakers and microphones) are provided throughout this chapter. A few high-level audio comments are provided here:

- Stereo music/ringer melody applications are supported in addition to the 8 kHz voice applications on the forward link (Rx). The Rx output is software configurable as mono or stereo. Additional software selections include 13-bit or 16-bit linear conversion with sampling at 8, 16, 22.05, 24, 32, 44.1, or 48 kHz. One differential earphone output, a stereo single-ended headphone output, one auxiliary output, and a stereo single-ended line output are supported.
- In the audio transmit path, a 13-bit linear converter is provided with a software selectable sampling rate of 8 or 16 kHz depending on the audio codec used. The MSM6290 IC is one of the first MSM product that supports stereo wideband sampling. Two differential microphone inputs, a differential auxiliary input, and a stereo line input are supported.
- Rx and Tx filters meet ITU-T G.712 requirements.
- The PCM interface allows an external codec to be used instead of the internal codec; this supports I2S modes that allow an external stereo DAC to be used. Currently only I2S output mode (SDAC) is supported.
- Audio decoder summing and headset switch detection are included.

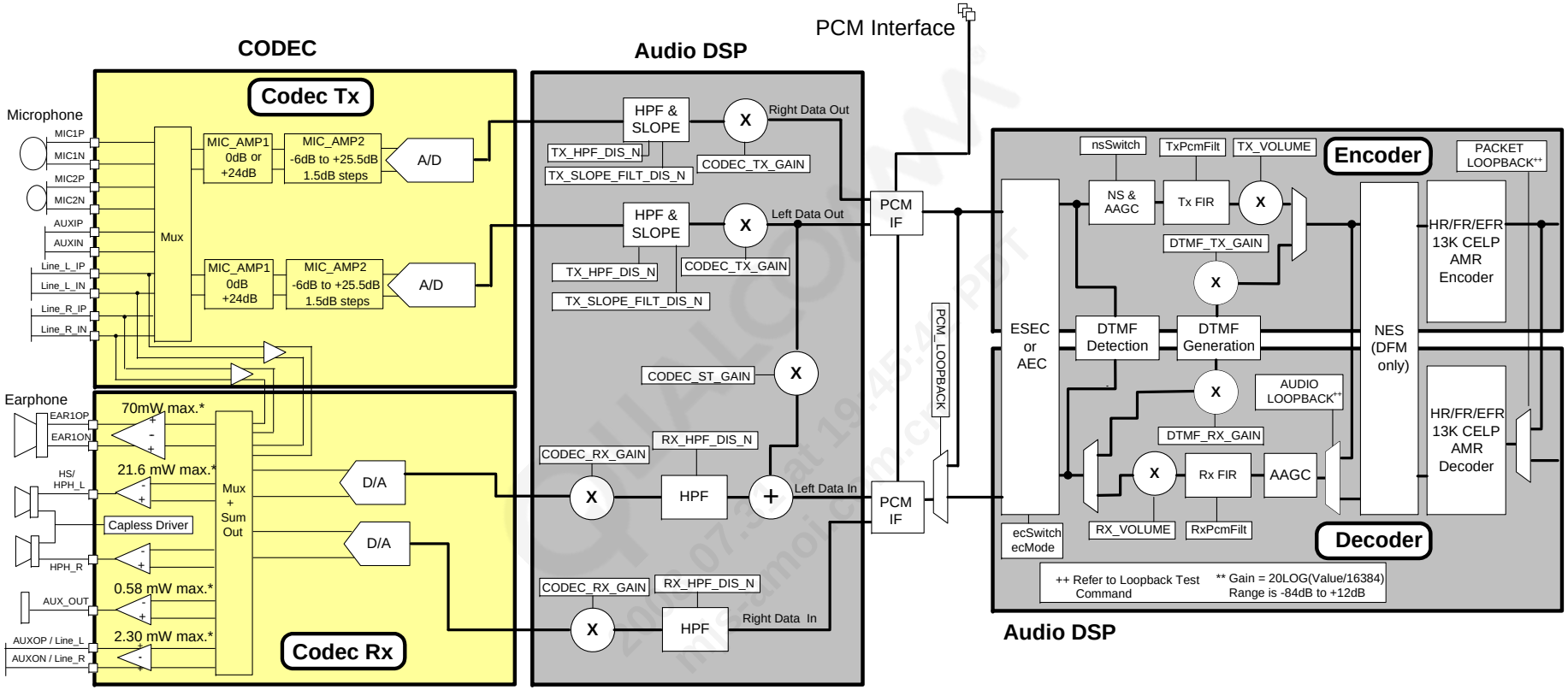


Figure 6-1 Codec, audio DSP, and PCM interface

6.1 Audio connections

MSM6290 audio connections (Figure 6-2) fall primarily into seven categories:

1. Microphone inputs
2. Auxiliary input
3. Line inputs
4. Earphone outputs
5. Stereo headphone outputs
6. Auxiliary output
7. Line outputs

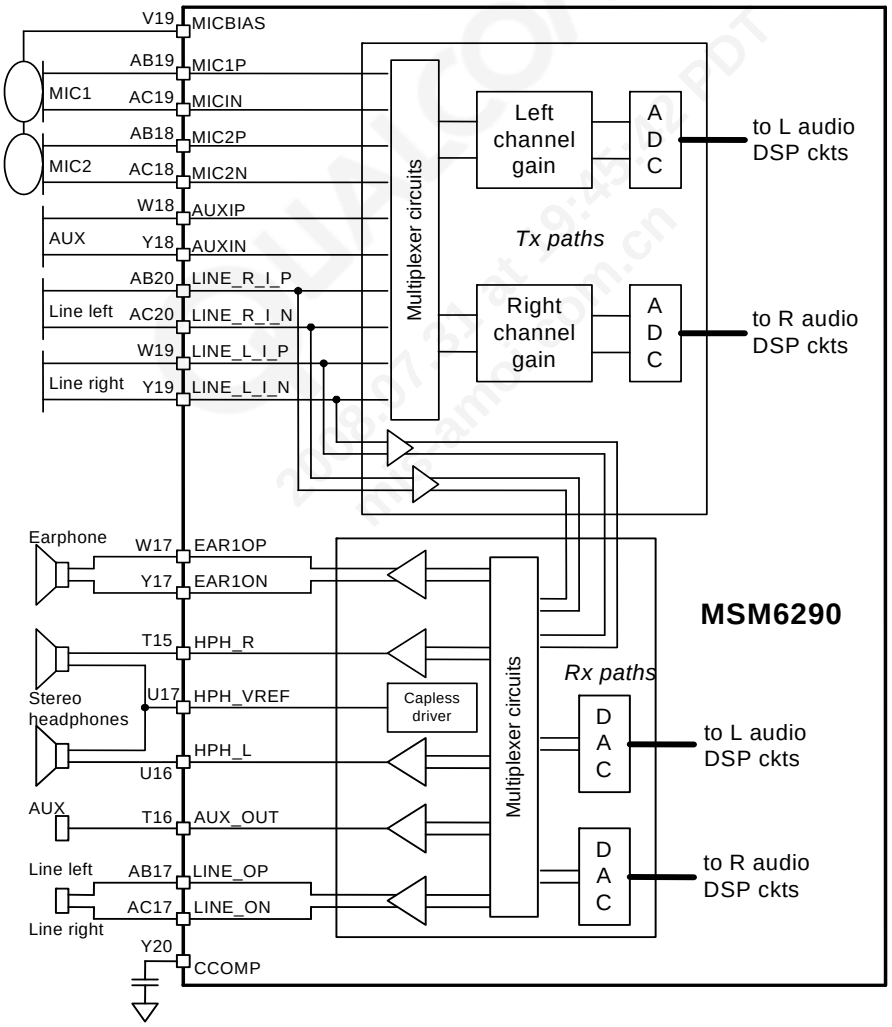


Figure 6-2 Analog codec front end with audio connections

The audio pin assignments are listed in Table 6-1.

Table 6-1 Audio connections

Signal name	Pin #	I/O	Voltage	Description	Comments
Microphone inputs					
MIC1P	AB19	I	Analog	Microphone #1 differential positive (+) input	
MIC1N	AC19	I	Analog	Microphone #1 differential negative (–) input	
MIC2P	AB18	I	Analog	Microphone #2 differential positive (+) input	
MIC2N	AC18	I	Analog	Microphone #2 differential negative (–) input	
MICBIAS	V19	B	Analog	Microphone bias supply	No decoupling cap
Auxiliary input					
AUXIP	W18	I	Analog	Auxiliary differential positive (+) input	
AUXIN	Y18	I	Analog	Auxiliary differential negative (–) input	
Line inputs					
LINE_R_I_P	AB20	I	Analog	Right channel stereo function with 3 options	Options: (1) line in; (2) microphone, (3) summing function to Rx
LINE_R_I_N	AC20	I	Analog		
LINE_L_I_P	W19	I	Analog	Left channel stereo function with 3 options	
LINE_L_I_N	Y19	I	Analog		
Earphone outputs					
EAR1OP	W17	O	Analog	Earphone differential positive (+) output	
EAR1ON	Y17	O	Analog	Earphone differential negative (–) output	
Stereo headphone outputs					
HPH_R	T15	O	Analog	Stereo headphone right output or negative (–) headphone out	
HPH_L	U16	O	Analog	Stereo headphone left output or positive (+) headphone out	
HPH_VREF	U17	I	Analog	Headphone common mode voltage	
Auxiliary output					
AUX_OUT	T16	O	Analog	Auxiliary output to carkit, PMIC, or external speaker	Single-ended
Line outputs					
LINE_OP	AB17	O	Analog	Positive (+) line (LINE_OUT_L) output or stereo left channel output	
LINE_ON	AC17	O	Analog	Negative (–) line (LINE_OUT_R) output or stereo right channel output	
Other audio-related pin					
CCOMP	Y20	I	Analog	External decoupling cap for CODEC voltage reference	0.1 µF recommended

An on-chip voltage/current reference generates the precise voltages and currents required by audio functions. This reference circuit requires a single 0.1 µF bypass capacitor connected from the CCOMP pin to PCB GND. The audio circuits also provide a microphone bias voltage (MICBIAS) required for electret condenser microphones typically used in handset applications. This 1.8 V output provides as much as 1 mA of current.

6.2 Tx path inputs and gains

The Tx paths begin with an off-chip microphone. The microphone outputs can be applied to the MSM6290 IC using one of its two differential microphone inputs, its differential auxiliary input, or its stereo line input. Interconnect circuit details are provided in Section 6.5 and its subsections; on-chip Tx input processing is shown in fair detail within Figure 6-1 (the block labeled “Tx paths”).

The desired audio Tx inputs are selected and routed to the appropriate right or left channel gain stages. There are two gain stages on both the left and right paths. The first stage (MIC_AMP1) is programmable for either 0 dB or +24 dB gain. The second stage (MIC_AMP2) has programmable gain from -6 dB to +25.5 dB in 1.5 dB steps. These gains are controlled by SBI registers.

There are additional gain stages after the analog-to-digital converters, that are available in Tx path audio DSP, explained in Section 6.4.1. The “Tx paths” block ends with the ADC circuits and their digital outputs to the audio DSP block.

It is evident from Figure 6-1 that any Tx input can be routed back to any Rx path’s driver amplifier output. The most direct path is provided within the “Tx paths” block: line left and line right inputs can be buffered (AUX_PGA) and routed directly to a driver amplifier output.

6.3 Rx path outputs

The Rx paths end with an off-chip speaker. The speaker inputs can be driven by the MSM6290 IC using its differential earphone output, its stereo headphone outputs, its single-ended auxiliary output, or its (new) dedicated stereo line output. Interconnect circuit details are provided in Section 6.5 and its subsections; on-chip Rx output processing is shown in fair detail within Figure 6-1 (the block labeled “Rx paths”).

The “Rx paths” block begins with digital inputs from the audio DSP block that are applied to the DAC circuits. The DAC analog outputs are routed to the desired driver amplifier as selected by the multiplexer circuits via SBI registers.

The earphone driver provides a mono-differential output and the auxiliary output is single-ended mono. The headphone and line-out drivers provide stereo single-ended outputs that can also be configured as mono differential pairs. Unused driver circuits are disabled with their outputs placed in a high-Z state. A side-tone path is available that redirects sound from the Tx path to the Rx path.

Gains for Rx path are prior to the digital-to-analog converters in the Rx path audio DSP, explained in Section 6.4.2.

6.4 Audio DSP

The audio DSP block includes Tx and Rx functions; each are discussed in this section. The audio DSP functions are shown in fair detail within Figure 6-1 (the block labeled “Audio DSP”).

In addition to the Tx and Rx specific functions, a side-tone path is provided from the Tx path to the Rx path. This makes it possible to add a portion of the Tx audio into the receive path; the path of the injected Tx signal includes a programmable gain stage having a range of 0 dB to -96 dB (controlled by the QDSP5000 DMA parameter CODEC_ST_GAIN contained within the AFE_CMD_INT_CODEC_GAIN_CONFIG register). A programmed value of 0x4000 results in -12 dB of gain while 0x0000 mutes the side-tone. The gain calculation for CODEC_ST_GAIN is:

Gain = 20*LOG(CODEC_ST_GAIN/16384) – 12 dB

Qualcomm recommends this side-tone technique to discourage users from talking too loudly into their phones.

6.4.1 Tx path audio DSP

Transmit data from the microphone is digitally filtered with an ITU G.712 compliant filter that attenuates input signals outside the 3400 Hz baseband and decimates the data rate to 8 kHz. There are two optional digital filters on the Tx path prior to the vocoder: a slope filter and a highpass filter. The slope filter (Figure 6-3) is designed to provide pre-emphasis for the high frequency audio prior to the vocoder.

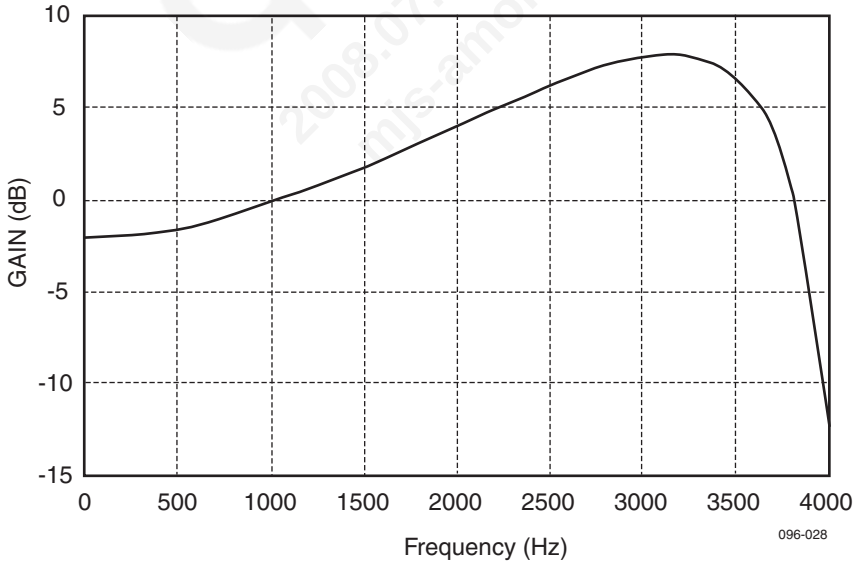


Figure 6-3 Slope filter response

The highpass filter (Figure 6-4) provides at least 30 dB of attenuation below 120 Hz. This has more significance than in previous MSM designs because the MSM6290 IC does not have the external gain circuit that included a highpass filter. It is recommended that the internal MSM6290 highpass filter be enabled for all applications where a microphone is the input source, thereby acting as a wind noise rejection filter.

The two filters are individually enabled via codec configuration commands (TX_HPF_DIS_N and TX_SLOPE_FILT_DIS_N).

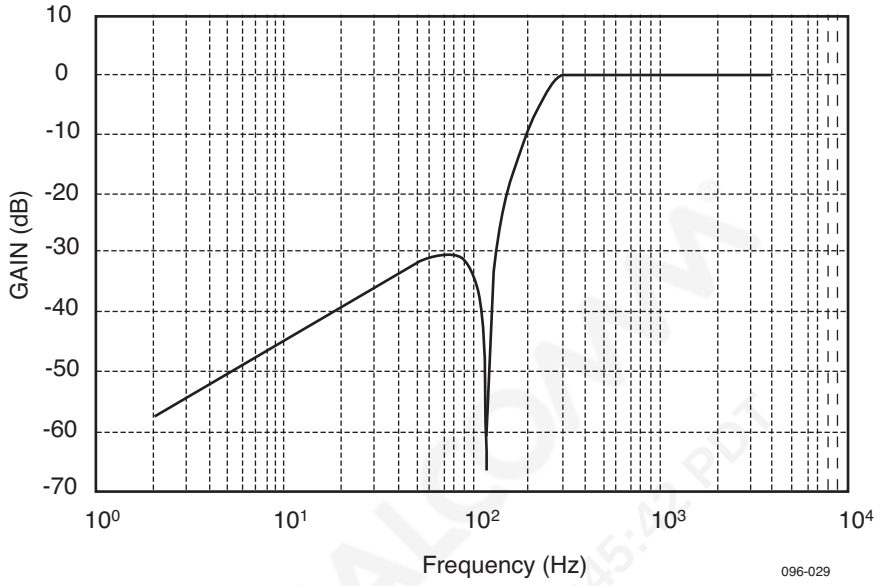


Figure 6-4 Highpass filter response

In addition to gain stages within the "Tx paths" block, the DSP block provides two gain stages in the Tx path:

- The first stage has a range of -84 dB to +12 dB and is programmed using CODEC_TX_GAIN within the AFE_CMD_INT_CODEC_GAIN_CONFIG register.
- The second stage has a range of -84 dB to +12 dB and is programmed using TX_VOLUME within the VOLUME_CTRL_CMD register.

For CODEC_TX_GAIN and TX_VOLUME, a programmed value of 0x4000 provides unity gain and a programmed value of 0x0000 mutes the Tx audio data. The gain calculation is:

Gain = 20*LOG(CODEC_TX_GAIN{or TX_VOLUME}/16384)

6.4.2 Rx path audio DSP

The receive DSP input is digitally filtered with an ITU G.712 compliant filter. The filter response is flat out to 3400 Hz, but still provides at least 14 dB attenuation at 3.98 kHz to ensure adequate image rejection.

A user-selectable highpass filter is available for rejection of low-frequency noise. This filter provides at least 30 dB of attenuation below 120 Hz and is identical to the Tx highpass filter having the frequency response shown in CHECK THIS OUT. The codec configuration command RX_HPF_DIS_N selects this highpass filter.

The Rx audio path includes two gain stages on either side of the PCM interface, each with a range of +12 dB to -84 dB prior to the digital-to-analog conversion. These QDSP5000 DMA parameters are CODEC_RX_GAIN (in AFE_CMD_INT_CODEC_GAIN_CONFIG) and RX_VOLUME (in VOLUME_CTRL_CMD).

CODEC_RX_GAIN is a fixed gain used to adjust for the sensitivity of the speaker driver. RX_VOLUME is the parameter that changes when the phone user adjusts the volume level of the speaker driver. A programmed value of 0x4000 is +0 dB of gain and a programmed value of 0x0000 mutes the Rx audio data. The gain calculation for these is:

Gain = 20*LOG(CODEC_RX_GAIN or RX_VOLUME /16384) dB

6.5 Recommended Tx and Rx gain settings

Tx and Rx path gain setting recommendations are defined in this section.

Tx path gain settings

The following four Tx path audio gains are recommended when using a microphone input for a voice call in handset mode:

1. Set MIC_AMP1 (0 or 24 dB) gain in conjunction with MIC_AMP2 gains such that the output of MIC_AMP2 is 23 dB to 25 dB below the saturation point (3.626 Vpp) of the analog-to-digital converter. In a typical phone design the MIC_AMP1 will be set at 24 dB.
2. CODEC_TX_GAIN should be set to approximately +4 dB.
3. The TX_VOLUME setting depends upon whether the TX AGC is turned on or not:
   - If Tx AGC is off, TX_VOLUME should be set to approximately 0 dB.
   - If Tx AGC is turned on with a -21 dB compression threshold, then TX_VOLUME should be set to approximately +3 dB. In this case there should also be 0 dB AGC static gain applied (assuming the previous recommendations are followed).
4. Most input gain adjustments should be implemented using MIC_AMP2 gain. This should be set such that the mobile meets the send loudness rating (SLR) and distortion requirements of the 3GPP/3GPP2 standards and the carriers. If further slight fine tuning is required, due to the MIC_AMP2 having 1.5 dB gain steps, CODEC_TX_GAIN can be set in the range of +3 to +5 dB.

Rx path gain settings

The following two Rx path audio gains are recommended for a voice call in handset mode:

1. The RX_VOLUME setting depends upon whether the Rx AGC is turned on or not:
   - If Rx AGC is off, the maximum RX_VOLUME level should be set to approximately 0 dB.
   - If Rx AGC is turned on with a -21 dB compression threshold, then the RX_VOLUME maximum level should be set to approximately +6 dB.
2. Generally, the CODEC_RX_GAIN should be set to any value within its usable range such that the receive loudness rating (RLR) and noise requirements of the 3GPP/3GPP2 standards and the carriers are met. *However, if values outside of the range of the ±12 dB are needed, handset designers should contact Qualcomm for assistance.*

6.6 PCM interface

The MSM6290 PCM interface can be used in two modes: 1) the default mode is its auxiliary PCM that runs at 128 kHz; 2) the other mode is its primary PCM that runs at 2.048 MHz. Primary PCM is disabled at powerup or when RESIN_N is asserted, but clearing bit 4 in the CODEC_CTL register enables the primary PCM mode. Both PCM modes are discussed in this section, followed by additional PCM topics.

6.6.1 Auxiliary PCM

The auxiliary PCM interface enables communication with an external codec to support hands-free applications. Linear, μ-law, and A-law codecs are supported by the auxiliary PCM interface.

The auxiliary codec port operates with standard long-sync timing and a 128 kHz clock. The AUX_PCM_SYNC runs at 8 kHz with 50% duty cycle. Most μ-law and A-law codecs support the 128 kHz AUX_PCM_CLK bit clock.

The ONES_DETECT state machine is clocked by CLKOUT of the internal ARM926EJ-S microprocessor. The logic is controlled by CODEC_CTL:ONES_POLARITY with a single output of WEB_MISC_RD:ONES_DETECT. If ONES_POLARITY = 0, then a high on AUX_DIN guarantees ONES_DETECT = 0. If ONES_POLARITY = 1, then a low on AUX_DIN guarantees ONES_DETECT = 1.

In addition, this logic generates an AUX_PCM_DIN_INT interrupt that is asserted when the value on the AUX_PCM_DIN pin matches that of the ONES_POLARITY bit. This interrupt is enabled when the bit AUX_PCM_DIN_INT_EN within the IRQ_MASK_0 or FIQ_MASK_0 register is set (1); clear (0) this bit to disable the interrupt.

6.6.2 Primary PCM

The aux codec port also supports 2.048 MHz PCM data and sync timing for linear, μ-law, and A-law codecs that match the sync timing — this is called the primary PCM interface (or just PCM interface). PCM interfaces can be configured and controlled by either direct register access through the CODEC_CTL register, or by the QDSP4000 codec configuration command. Using the codec configuration command is the preferred method and is the only method discussed here.

The PCM_CLK pin provides the clock for receive (PCM_DIN) and transmit (PCM_DOUT) codec PCM data; its direction is defined by PCM_CLK_DIR. When clear (0), the PCM_CLK pin becomes an input (PCM_CLK comes from the PCM_CLK pin). When set (1), the PCM_CLK pin becomes an output (PCM_CLK comes from the clock and sync generator).

The PCM_SYNC pin direction is also programmable, defined by PCM_SYNC_DIR within the same register. When clear (0), the PCM_SYNC pin becomes an input (PCM_SYNC comes from the PCM_SYNC pin; PCM_SYNC must be latched in). When set (1), the PCM_SYNC pin becomes an output (PCM_SYNC comes from the clock and sync generator).

The polarities of PCM_CLK and PCM_SYNC are inverted by setting (1) PCM_CLK_SENSE. When PCM_CLK_SENSE is set (1) the codec interface latches the PCM_SYNC on the rising edge of the PCM_CLK.

6.6.3 I2S introduction

The I2S bus provides a serial link especially for digital audio. The bus only has to handle audio data, while the other signals, such as sub-coding and control, are transferred separately. Since the transmitter and receiver have the same clock signal for data transmission, the transmitter, as the master, has to generate the bit clock, word select (WS) signal and data. In complex systems, there may be several transmitters and receivers controlling digital audio data flow between the various ICs. Transmitters then, have to generate data under the control of an external clock, and so act as slave.

6.6.3.1 Serial data

Serial data is transmitted in two's complement with the MSB first. The MSB is transmitted first because the receiver and transmitter may have different word lengths. When the system word length is greater than the transmitter word length, the word is truncated (LSBs are set to '0') for data transmission. The MSB has a fixed position, while the position of LSB depends on word length. The transmitter always sends the MSB of the next word one clock period after the WS changes.

Serial data sent by the transmitter may be synchronized with either the trailing (high to low) or the leading (low to high) edge of the clock signal. However, the serial data must be latched onto the receiver at the leading edge of the serial clock signal, and hence there are certain restrictions when transmitting data that is synchronized on the leading edge.

6.6.3.2 Word select

The WS line indicates the channel being transmitted.

- WS = 0; channel 1 (left)
- WS = 1; channel2 (right)

WS may change on either the trailing or leading edge of the serial clock, but it does not need to be symmetrical. In the slave, this signal is latched on the leading edge of the clock signal. The WS line changes one clock period before the MSB is transmitted. This allows the slave transmitter to derive synchronous timing of the serial data that will be set up for transmission. It also enables the receiver to store the previous word and clear the input for the next word.

6.6.4 Using AUX_PCM to interface with an external stereo DAC

The MSM6290 pins used for the AUX_PCM interface (Table 6-2) can also be used to interface with an external stereo DAC (SDAC) to play stereo sound or music (MP3 or MIDI for example). The I2S bus in output mode provides a serial link specifically for digital audio; it handles the transfer of audio data and transports it to the interface.

NOTE Using the I2S bus in input mode with AAC encoding will be supported in a future software release.

Table 6-2 GPIO assignments for AUX_PCM and SDAC interfaces

GPIO	Pin #	AUX_PCM functionality	SDAC interface functionality
GPIO102	D4	AUX_PCM_SYNC	SDAC_L_R_N
GPIO80	E23	AUX_PCM_CLK	SDAC_CLK
GPIO14	L23	AUX_PCM_DIN	SDAC_MCLK
GPIO103	G7	AUX_PCM_DOUT	SDAC_DOUT

The stereo DAC has a 16-bit per channel output at several sampling rates from 8 to 48 kHz. The MSM/SDAC interface will output PCM data based upon the sample rate of the AAC contents. Additional SDAC interconnect details are explained below.

SDAC_DOUT: The serial PCM data stream for both channels are output from the MSM through this pin. The data is transmitted in two's complement with the MSB first. This minimizes the loss of data when the transmitter word length (16 bits) is different than the receiver device's word length, and is handled in one of two ways:

- When the transmitted word length is greater than the receiver word length, the bits after the receiver's LSB are ignored — the rest of the transmitter LSBs are ignored.
- When the transmitted word length is less than the receiver word length, the receiver's missing LSB will be set to zero initially, so they will remain at zero.

It is important to note that the above scenarios are true specifically when the system consists of just one transmitter (the master). In a complex system where there are several transmitters, it is possible that the system word length will be greater than a specific transmitter's word length. In that case, the LSBs of the system word will be set to '0' for data transmission.

SDAC_L_R_N: This signal specifies the present data stream's intended stereo channel: 1 specifies the left channel, 0 specifies the right.

SDAC_CLK: This is the bit clock that can be generated by the MSM (as explained in Section 6.6.4.1 and then supplied to the external stereo DAC. Alternatively it can be generated by the external stereo DAC and then provided to the MSM device.

The SDAC_CLK frequency is dependent upon the number of bits per channel and the selected sampling rate. For example, for two channels, 16 bits each, and a chosen 48 kHz sampling frequency, the frequency would be:

FSDAC_CLK = (16 + 16) * 48 k = 1536000 b/s = 1.536 MHz

Table 6-3 provides example SDAC_CLK frequencies for a two channel, 16-bit output at various sampling rates.

Table 6-3 Sampling frequency and SDAC_CLK rates

Sampling frequency	Required SDAC_CLK frequency
32 kHz	1.024 MHz
44.1 kHz	1.4112 MHz
48 kHz	1.536 MHz

NOTE The I2S bus supports a maximum frequency of 12.888 MHz.

SDAC_MCLK: an optional clock output from the MSM device to the external stereo DAC. This clock does not affect the MSM device; it merely provides handset designers with the option of using the MSM device to provide the external device with a master clock signal (MCLK).

6.6.4.1 SDAC_CLK setting

If the MSM6290 IC is chosen as the clock source for the SDAC_CLK, the SDAC_CLK signal can be configured using the SDAC_CLK_MD and SDAC_CLK_NS registers. The SDAC_CLK can be configured to run from various input clock sources (TCXO, Sleep Xtal, 196.608 PLL, etc.); each source can be divided by fixed values such as 2 or 4, or can be divided by an M/N:D counter that provides many more options.

When using the M/N:D counter, the M/N is the division ratio and the M and N values must be programmed correctly. The D value controls the output duty cycle. For a duty cycle of 50%, D should equal N/2. Using the M/N:D counter has limitations and can cause mild-to-serious jitter on the clock signal. In order to guarantee minimal jitter, even integer divisors should be used.

For convenience, a 1.024 MHz clock is available from a 2.048 MHz source divided by two; in this case the M/N:D counter is bypassed.

To create an output frequency *f*, with a given duty cycle, D, the following sequence must be used to initialize the SDAC M/N:D counter:

1. Solve for decimal values for M and N. The value of M cannot exceed the value of N.
2. Convert the decimal values of M and D into 16-bit values.
3. Write the 16-bit M value into SDAC_MD_REG[31:16] and D value into SDAC_MD_REG[0:15]
4. Write the one’s complement of N minus M into the SDAC_NS_REG[31:16]

To assure proper implementation and adequate output clock quality, contact Qualcomm for all M/N ratios.

6.6.5 Using AUX_PCM to interface with an external stereo ADC

Interfacing the MSM6290 device to an external stereo ADC provides a mechanism to input stereo digital audio into the MSM devices’ audio processing subsystem. The audio is subsequently processed by a stereo audio encoder. Applications for this interface include but are not limited to the following:

- Record stereo FM
- Stereo camcorder mic support
- Stereo line-in support

The MSM6290 device interfaces to a stereo ADC IC. This external IC must support the following feature:

- Stereo ADC IC must be able to operate in I2S master mode. This means that the external IC is responsible for generating the bit and left/right clocks, also known as continuous serial and WS clocks, respectively.

The external SADC interface system uses the same four GPIOs as the SDAC system. These pins are traditionally used as AUX_PCM pins.

NOTE I2S input mode (SADC) has not been tested yet.

6.7 External analog interface details

Four example audio interface configurations are shown in this section:

- Handset
- Headset
- Carkit
- Line-in/line-out

6.7.1 Handset interfaces

Typical handset interfaces are shown in Figure 6-5. The earphone output pins are connected directly to the handset's earphone with two bypass capacitors connected to ground. The capacitance is selected depending on the design, typically less than 100 pF. The output power for the differential EAR1 output is typically 70 mW for a full-scale +3 dBm0 sine wave into a 32 Ω speaker.

Both microphone pins require 2.2 kΩ bias resistors and 0.1 µF AC-coupling capacitors. The positive microphone terminal is connected to the MSM6290 MICBIAS pin through one of the 2.2 kΩ resistors; this 1.8 V output provides 1 mA of bias current for the microphone. MICBIAS supports multiple microphones simultaneously up to 1 mA.

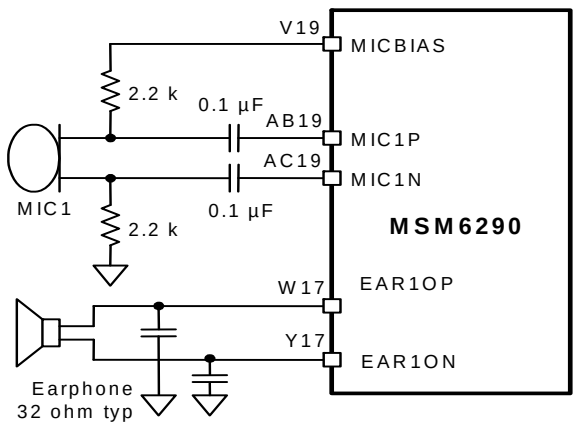


Figure 6-5 Typical handset microphone and earphone interface

6.7.2 Headset interfaces

The most basic handset configuration is shown in Figure 6-6. This configuration uses an AC-coupled mono earphone interface and a standard single-ended microphone interface. The output power for the single-ended HPH_L/HPH_R output is typically 21.6 mW per side for a full-scale +3 dBm0 sine wave into a 16 Ω speaker.

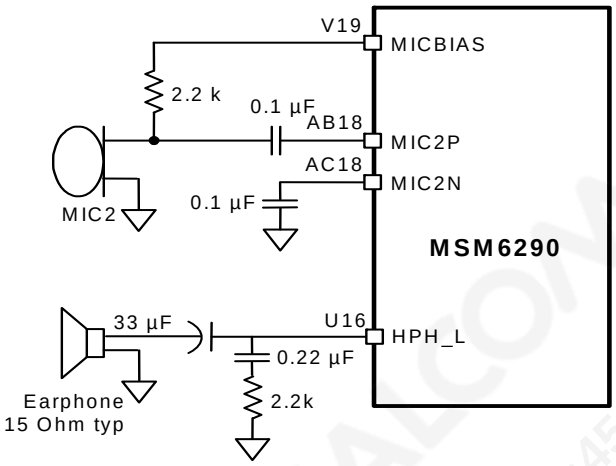


Figure 6-6 Basic headset interfaces

A few alternative earphone configurations are given in the following paragraphs. If the load capacitance is greater than 100 pF due to earphones with different capacitive load used, a RC shunt network (0.22 µF and 22 Ω) is recommended to prevent oscillations as shown on Figure 6-6.

Stereo earphone with AC-coupling

For stereo earphone configurations, the HPH_R and HPH_L outputs to drive the right and left speakers, respectively. Each speaker is also connected to a common ground. AC-coupling capacitors having values from 33 to 470 µF are used. If an RF bypass capacitor is required, add the capacitor at each speaker input as shown in Figure 6-7. If the load capacitance is greater than 100 pF due to earphones with a different capacitive load used, a shunt RC network is added to both MSM outputs to prevent oscillations; recommended values are 0.22 µF and 22 Ω.

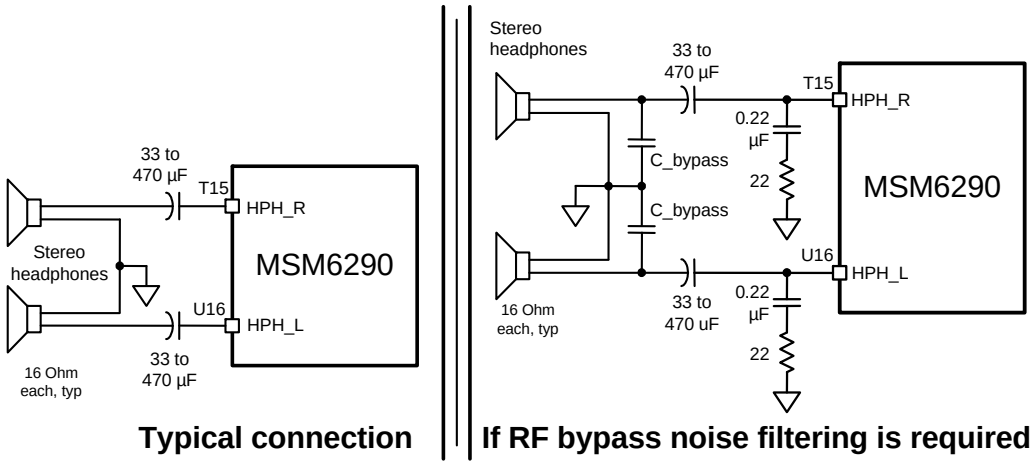


Figure 6-7 Stereo earphone interfaces

Depending on the actual board design, RF bypass capacitors can be added to filter unwanted RF noise couples into the earphone.

Mono-differential earphone

The recommended circuit for driving a mono-differential earphone is shown in Figure 6-8. Both the HPH_R and HPH_L outputs are used to drive a single 32-Ω speaker. If an RF bypass capacitor is required, add the capacitor at each speaker input as shown in Figure 6-8. If the load capacitance is greater than 100 pF due to earphones with a different capacitive load used, a shunt RC network is added across the MSM output pins; recommended values are 0.22 µF, 44 Ω, and 0.22 µF.

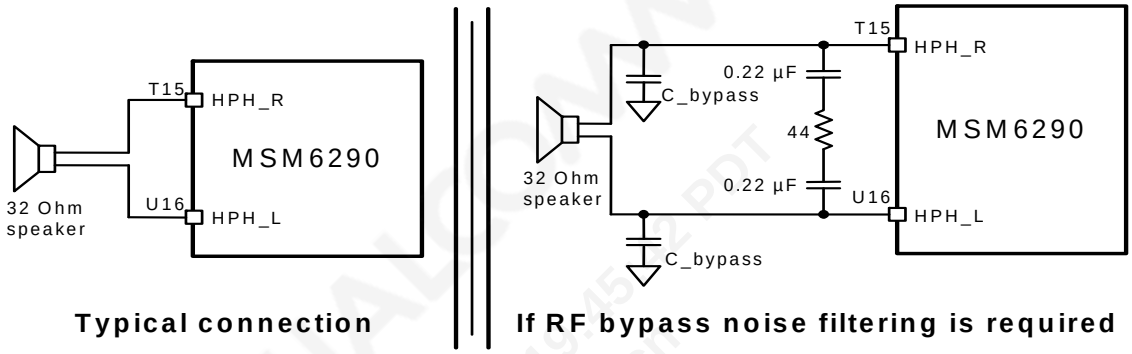


Figure 6-8 Mono-differential earphone interface

Stereo earphone using the capless driver

The recommended circuit for a stereo earphone using the MSM device's capless driver is shown in Figure 6-9. Again, both the HPH_R and HPH_L outputs are used to drive the right and left speakers, respectively. But in this case, each speaker is also connected to a common reference signal from the MSM device (HPH_VREF). If RF bypass capacitor is required, a shunt capacitor can be added at each speaker and ground. If the load capacitance is greater than 100 pF due to earphones with a different capacitive load used, a RC shunt network is added between both outputs and the reference voltage to prevent oscillations; recommended values are 0.22 µF and 22 Ω.

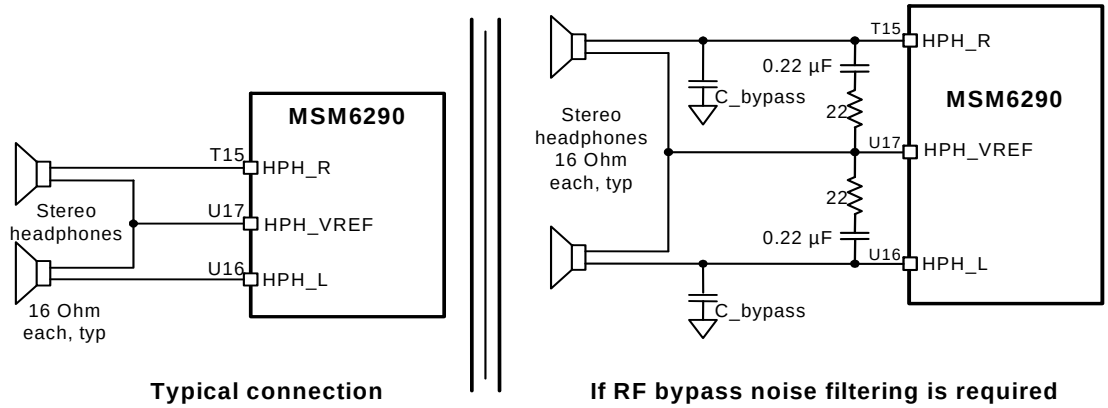


Figure 6-9 Stereo earphone interfaces using the capless driver

Alternative microphone configurations are available as well; these are given in the following paragraphs.

Basic microphone with headset switch detect (HSSD)

The MSM6290 IC is able to detect when a headset switch is depressed, even if it is in a sleep mode. The most basic implementation that allows this feature is shown in Figure 6-10. In this case, the MICBIAS circuit is internally enabled for the sleep mode detection mode whereby the MSM detects the action of the headset switch via an interrupt.

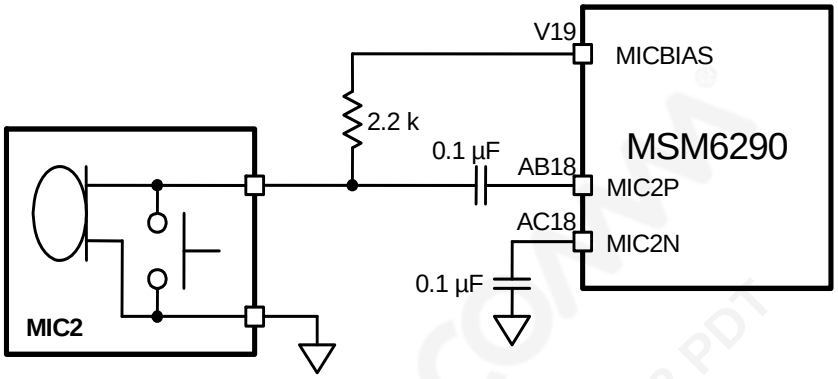


Figure 6-10 Basic microphone with switch

Microphone with headset switch using the capless driver

A circuit that exploits the MSM capless driver (Figure 6-10) can be used rather than the basic circuit. In this case the microphone bias is provided by an on-board power supply, thereby freeing up the MICBIAS pin to function as a HSSD detection circuit. Headset switch depressed is reported as an interrupt. A 0.1 µF capacitor to ground at MIC2N is required to filter out unintended switch press. In addition, the HPH_VREF pin is configured internally as ground to enable the headset switch detection to work in sleep mode.

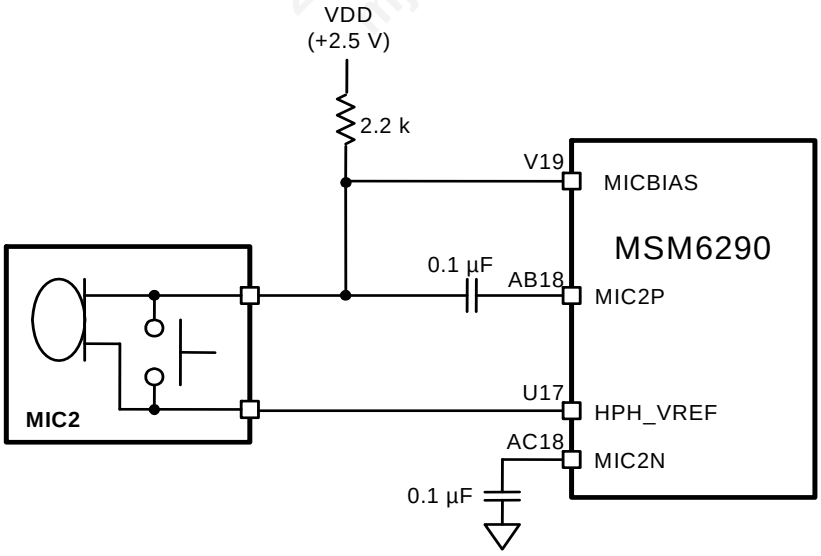


Figure 6-11 Microphone with switch using the capless driver

Microphone with separate switch output pin

If the microphone has a separate switch output pin, that extra pin is connected to an MSM6290 GPIO (Figure 6-12).

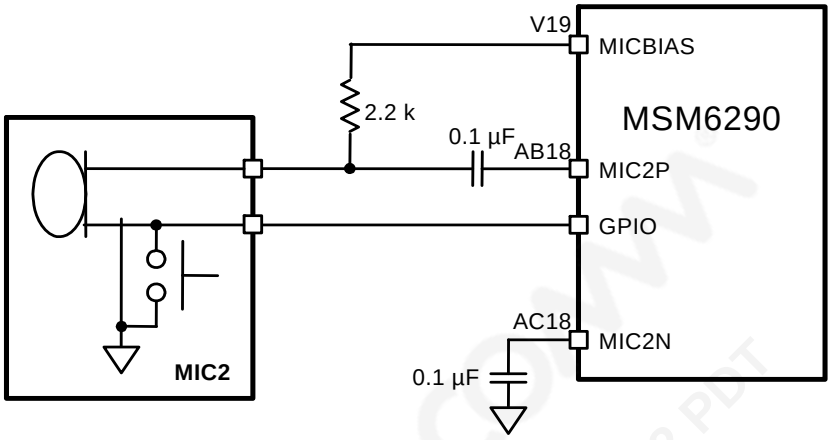


Figure 6-12 Microphone with separate switch output pin

6.7.3 Auxiliary I/O interface (carkit)

The MSM6290 device provides an analog auxiliary interface for analog carkit applications (Figure 6-13).

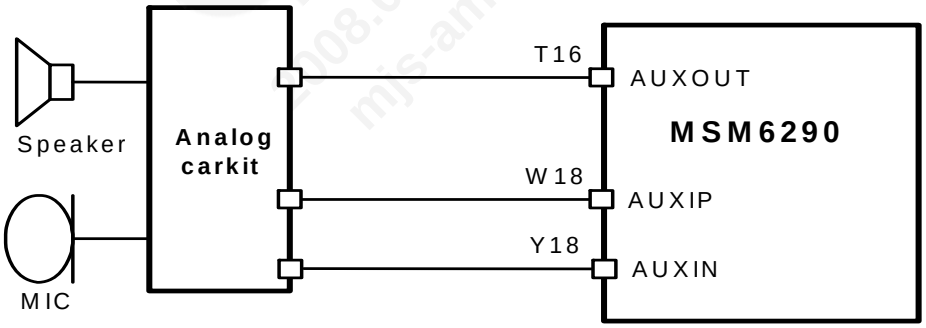


Figure 6-13 Typical analog carkit application

6.7.4 Line input to output audio

The MSM6290 codec interface also supports stereo line in routed through AUX_PGA, to HPH_L/HPH_R or LINE_OUT_L/LINE_OUT_R or any other audio outputs desired (Figure 6-14 left or right; left is shown in the figure).

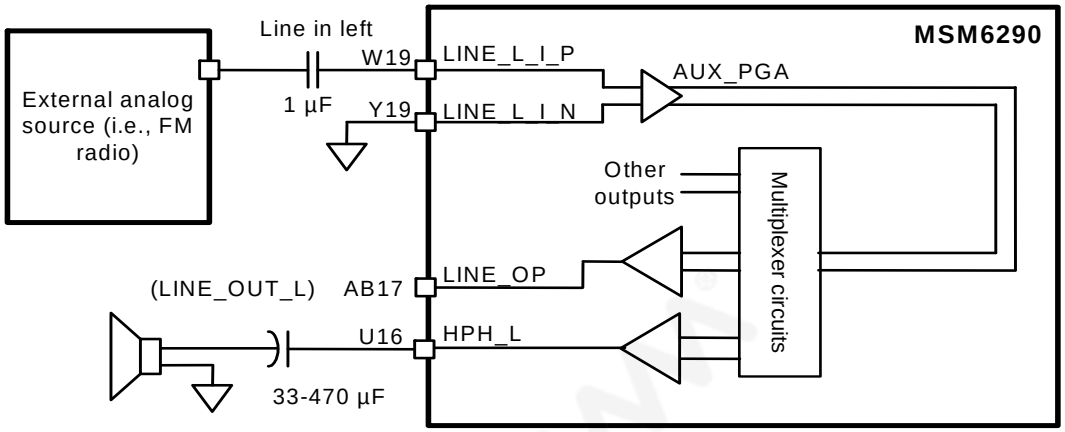


Figure 6-14 Typical line-in to line-out usage

6.8 Interface to an external speaker amplifier (PM6658 or PM6653 IC)

The PM6658 and PM6653 device can also be used as a speaker amplifier. The PM6658 or PM6653 speaker driver output power is rated at 500 mW. To use this feature as an amplifier of an audio output, be sure to set the appropriate speaker driver analog and digital gains, and set the analog high-pass filter corner at the resonant frequency of the far-field speaker transducer.

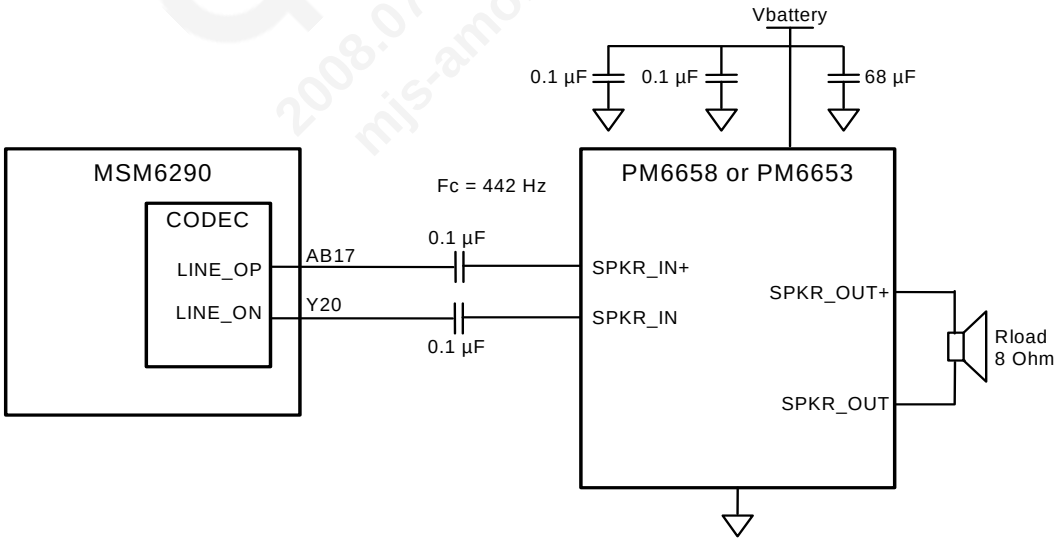


Figure 6-15 Using the PM6658 or PM6653 speaker driver

7 UART, USIM, and USB Interfaces

7.1 UART interface

The MSM6290 is capable of providing up to three UART ports. Each UART (universal asynchronous receiver transmitter) communicates with serial data ports that conform to the RS-232 interface protocol. With a properly written and user-defined download program, the UART can be used as the handset's serial data port for test and debug and can support additional interface functions such as an external keypad or ringer. If the handset uses EEPROM or flash memory, the UART can be used to load and/or upgrade system software.

Support for three UART interfaces in parallel with the following requirements:

- UART1: high-speed, maximum of up to 4 Mbps using the UART1DM data mover interface and up to 230 kbps for data services using the UART1 interface.
- UART2: maximum speed of 115 kbps.
- UART3: maximum speed of 115 kbps.

The serial data port is a UART channel (Figure 7-1). The UART processes both transmitted and received data with separate transmit (Tx) and receive (Rx) FIFOs (512-byte each for UART1). Supporting circuits include interrupt control, clock source, bit rate generator (BRG), and microprocessor interface. Each of these UART functions is described later in this chapter.

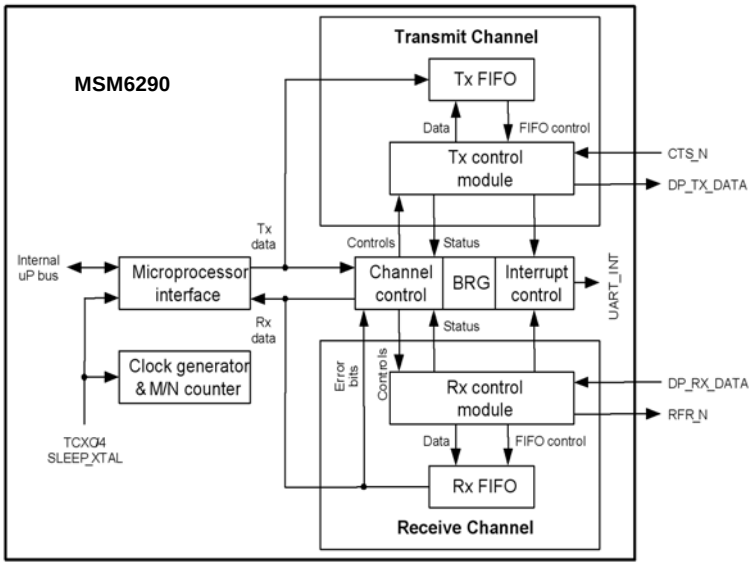


Figure 7-1 UART functional block diagram

7.1.1 UART features

The UART has several features that are common to both transmit and receive modes:

- Hardware handshaking
- Programmable parameters:
  - Data size
  - Stop bits
  - Parity
  - Bit rate
  - Selectable clock source
    - Command registers that control the broad UART functions such as enable and disable (of the Tx and/or Rx channel), start break, stop break, and reset (of the Tx and/or Rx channels).

All UART signals use GPIOs; this allows them to be configured for alternate functions as discussed throughout this chapter.

7.1.2 UART connections

UART connections include up to two ports (refer to Figure 7-2 and Table 7-1).

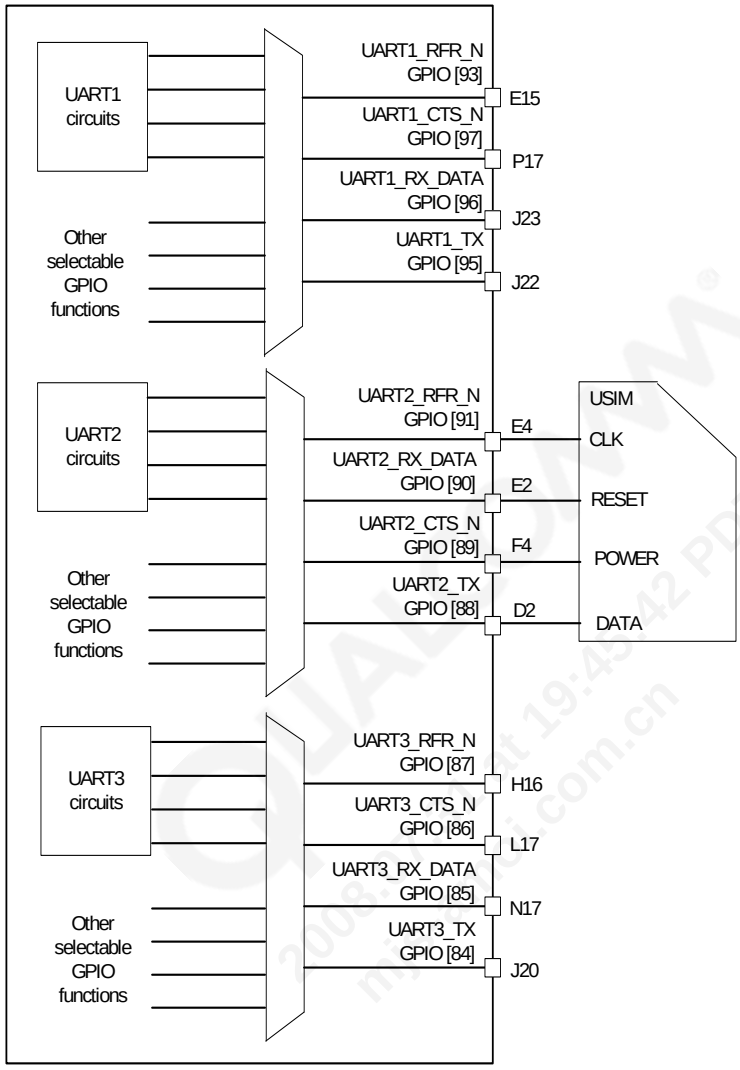


Figure 7-2 UART connections

Table 7-1 UART connections

Signal name	Pin #	I/O	Voltage	Description	Comments
UART1 or UART1DM					
UART1_RFR_N GPIO[93]	E15	O	Digital	Ready for receiving	
UART2_CTS_N GPIO[97]	P17	I	Digital	Clear to send	
UART2_RX_DATA GPIO[96]	J23	I	Digital	Receive serial data input	
UART1_TX_DATA GPIO[95]	J22	O	Digital	Transmit serial data output	

Table 7-1 UART connections (cont.)

Signal name	Pin #	I/O	Voltage	Description	Comments
UART2					
UART2_RFR_N GPIO[91]	E4	O	Digital	Ready for receiving	GPIO[91] must be used for USIM_CLK
UART2_RX_DATA GPIO[90]	E2	I	Digital	Receive serial data input	GPIO[90] must be used for USIM_RESET
UART2_CTS_N GPIO[89]	F4	I	Digital	Clear to send	GPIO[89] must be used for USIM_PWR_EN
UART2_TX GPIO[88]	D2	O	Digital	Transmit serial data output	GPIO[88] must be used for USIM_DATA
UART3					
UART3_RFR_N GPIO[87]	N16	O	Digital	Ready for receiving	
UART3_CTS_N GPIO[86]	L17	I	Digital	Clear to send	
UART3_RX_DATA GPIO[85]	N17	I	Digital	Receive serial data input	
UART3_TX_DATA GPIO[84]	J20	O	Digital	Transmit serial data output	

7.1.3 UART transmitter

The UART transmit channel contains a 512-byte Tx FIFO and a Tx control module. The Tx FIFO accepts parallel data from the microprocessor. The Tx control module reads data from the Tx FIFO and sends it out serially, adding a start bit, optional parity bits, and stop bit(s).

The TXRDY bit in the status register (SR) sets (1) whenever the Tx FIFO has space available. The TXLEV interrupt is asserted when the Tx FIFO has fewer than the number of characters programmed in the transmit FIFO watermark register (TFWR).

Idling or disabling the Tx channel holds the DP_TX_DATA pin in a marking state (high). Enabling the Tx channel with a character waiting in the Tx FIFO loads that character into the Tx shift register. Next, a start bit is transmitted (Tx low) for one bit time, followed by each bit in the character (LSB first) then an optional parity bit followed by stop bits (Tx high).

If the Tx FIFO is not empty after the previous transmission, the Tx control module begins another transmission by loading the next Tx FIFO character into the Tx shift register. If the Tx FIFO is empty after the previous transmission, the SR's TXEMT (Tx empty) bit is set (1). Setting TXEMT indicates an under-run in the Tx channel. The TXEMT bit clears (0) once the transmit shift register has a new character from the Tx FIFO.

The transmitting channel can also be enabled or disabled using the command register (CR). When disabled, the Tx channel continues transmitting any characters in the Tx shift register until the register is empty. When transmission ends, the DP_TX_DATA pin goes into a marking state (high).

If the CR issues a start break command after the Tx channel transmits all the characters in the Tx FIFO and shift register, the Tx channel forces the DP_TX_DATA pin low. The DP_TX_DATA pin remains low until the CR issues a 'Stop Break' command.

When the CR issues a reset transmitter command, the Tx channel ceases transmissions. The DP_TX_DATA pin enters a marking state and flushes the Tx FIFO.

And finally, the CTS control feature can be turned on by programming the UART mode register 1 (MR1). When on, the Tx channel checks the CTS_N input before transmitting a character. If CTS_N is high, the Tx channel stops transmitting and continues marking. If CTS_N is low, transmission begins or continues. If CTS_N goes high in the middle of character transmission, the Tx channel waits for a completed transmission before entering the marking state. The Tx channel can generate a programmable interrupt whenever CTS_N changes states.

7.1.4 UART receiver

The receive channel contains a 512-byte Rx FIFO and an Rx control module. Each byte in the Rx FIFO has two bits of corresponding status information. Serial data is input via the DP_RX_DATA pin which is pulled low internally. The Rx control module converts the serial data into parallel data and loads it into the Rx FIFO.

If the Rx FIFO is not full, the control module writes the received character to the Rx FIFO. When the Rx control module writes a received character to an empty Rx FIFO, the RXRDY bit in the status register (SR) is set (1). If the Rx FIFO becomes full, the RXFULL bit sets (1). The RXFULL bit clears (0) when a character is read from the Rx FIFO. The RXRDY bit clears (0) when the Rx FIFO becomes empty once again. When the Rx FIFO has more characters than what was programmed in the receive FIFO watermark register (RFWR), the Rx channel asserts a RXLEV interrupt. If a character is waiting in the Rx FIFO for a programmed time period (STALE_TIMEOUT), an RXSTALE interrupt occurs. The Rx channel asserts an RXHUNT interrupt whenever the Rx channel receives a character that matches the value found in the hunt character register (HCR).

Two status bits are associated with each 10-bit data word in the Rx FIFO. One status bit defines the received break condition; the other status bit is the logical-OR of a parity error or framing error. MR2:ERROR_MODE determines the character error mode or the block error mode. In character error mode, the SR's error bits apply only to the byte waiting to be read from the Rx FIFO. In block mode, the SR's error bits are the logical-OR of all incoming error bits since the command register last issued a 'Reset Error Status' command. Whether in character error mode or block mode, reading the status register has no effect on the Rx FIFO.

If the Rx FIFO is full and the Rx channel receives a new character, that character remains in the Rx shift register until a position becomes available in the Rx FIFO. Any additional incoming characters are lost until space becomes available in the Rx FIFO. When characters are lost, an overrun error occurs and SR:OVERRUN is set (1). Once SR:OVERRUN is set (1), it remains set until the CR issues a reset error status command. An overrun error does not affect the Rx FIFO's contents.

The receive channel can also be disabled using the CR. Incoming characters are now ignored, but characters already in the Rx FIFO remain unchanged and can be read by the microprocessor. Having the CR issue a reset receiver command flushes the Rx FIFO and clears (0) the status bits. The receive channel remains disabled until it is re-enabled, also by setting the appropriate bits on the CR.

The automatic ready-for-receiving (RFR_N) mode can be enabled on the UART's mode register 1 (MR1). In this mode, the RFR_N pin goes high when the Rx FIFO level is the same or greater than the value programmed in MR1. When the Rx FIFO level falls below the programmed level, the RFR_N pin returns to a low state. This feature prevents overruns by connecting the channel's RFR_N pin to a transmitting device CTS_N pin.

7.1.5 UART clock source

Each of the two UART interfaces has its own clock source chosen within the MISC_CLK_SELI register. The respective UART clock is then derived from this chosen clock source using the M/N counter present in each UART block. This counter is programmed using the UART_MREG, UART_NREG, and UART_DREG for each respective UART interface. Writing the value 0 (zero) to the M registers cause the M/N counter to output a 0 Hz clock, effectively turning off the clock and putting the UART into its power save mode. When the clock is disabled, none of the UART registers are accessible except for the M/N counter registers. Refer to the *MSM6290 Mobile Station Modem Software Interface* (80-VF866-2) for more information.

The selected UART2 clock source is also used for USIM operation. Handset designers must consider the interface's operating mode (UART2 or USIM) when selecting the clock source.

7.1.6 UART2 and UART3 bit rate generator

The desired bit rate for the receive and transmit channels is selected using the UART_CSR register (again, each UART has its own). Depending upon the selected clock source and selected M/N values, the fields in the UART_CSR register should be programmed to achieve the desired bit rate for either direction (transmit or receive). It is important to note that the maximum achievable MSM6290 bit rate is 115.2 kbps.

The bit rate generator (BRG) generates enables to the Tx and Rx channels that are 16 times the nominal bit rate. The BRG selects one of the 16 possible bit rates as defined in clock select register (Table 7-2) and sends the selected bit rate to the Tx channel (CSR bits [3:0]) and Rx channel (CSR bits [7:4]). The bit rate values listed in the table assume a system clock frequency of 1.8432 MHz and are generated by dividing the UART system clock.

Table 7-2 UART bit rate selection

CSR bits	Settings	Bit rate (bps) TCXO/4	Bit rate (bps) TCXO
Rx: bit [7:4] Tx: bit [3:0]	0000	75	300
	0001	150	600
	0010	300	1.2 k
	0011	600	2.4 k
	0100	1.2 k	3.6 k
	0101	2.4 k	4.8 k
	0110	3.6 k	7.2 k
	0111	4.8 k	9.6 k
	1000	7.2 k	14.4 k
	1001	9.6 k	19.2 k
	1010	14.4 k	28.8 k
	1011	19.2 k	38.4 k
	1100	28.8 k	57.6 k
	1101	38.4 k	115.2 k
	1110	57.6 k	230.4 k[1]
	1111	115.2 k	460.8 k[1]

1. Not tested

7.1.7 UART interrupts

The UART can assert six separate interrupts: DELTA_CTS, TXLEV, RXLEV, RXSTALE, RXBREAK, and RXHUNT. The interrupt register (ISR) shows each interrupt's status bit independent of the interrupt mask register (IMR) bit state. The mask interrupt status register (MISR) returns the bitwise AND of the ISR and IMR registers. The interrupt functions in the UART_ISR register can be enabled or disabled by setting the appropriate bits in the IMR. Furthermore, the conditions for the RXSTALE interrupt can be defined by the user on the UART_IPR register.

7.1.8 IrDA interface

The MSM6290 device contains an IrDA transceiver (Figure 7-3) that interfaces between the UART and the RX_DATA and TX_DATA pins. The IrDA feature is available and fully implemented in hardware for the UART1 interface, but is not supported by Qualcomm software — handset designers must configure it themselves to fit their needs. The name, address, bit field, and functionality details of each IrDA register are given in Table 7-3.

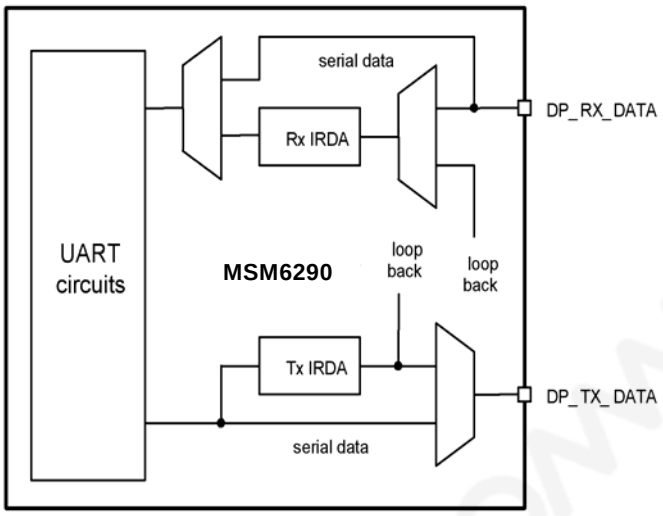


Figure 7-3 IrDA transceiver within the UART function

The IrDA block implemented behind the UART1 pins is shown in Figure 7-3. The IrDA transceiver interfaces with the UART on one side and with an external infrared transducer module on the other side through the RX_DATA and TX_DATA pins. The IrDA transceiver consists of two modules: a transmitter and a receiver.

The IrDA transmitter converts serial UART data in to the IrDA format for transmission. IrDA specification requires that a logic one (1) be transmitted as a logic zero (0), and that a logic zero (0) be transmitted as a pulse whose nominal width is $3/16^{th}$ of the bit rate, with a minimum width of 1.41 µs.

The IrDA receiver detects pulses transmitted from an external IrDA transmitter. According to the IrDA specifications, a logic one (1) be transmitted as a logic zero (0), and that a logic zero (0) be transmitted as a pulse whose nominal width is $3/16^{th}$ of the bit rate, with a minimum width of 1.41 µs.

UART1: UART_IRDA (0x8000 0038)

Table 7-3 UART_IrDA register details for UART interface

Bit	Functional description/settings
3	Reserved
2	Inverts the polarity of the IRDA modulated signal on the DP_TX_DATA pin 0 = non-inverted polarity 1 = inverted polarity
1	Inverts the polarity of the IRDA modulated signal on the DP_TX_DATA pin 0 = non-inverted polarity 1 = inverted polarity
0	Enables or bypasses the IRDA transceiver 0 = IRDA transceiver bypassed; other bits of this register are ignored 1 = IRDA transceiver enabled

7.1.9 UART2 and USIM differences

The UART2 interface has a 64-byte FIFO in the transmit and receive chains. The UART2 interface can be used as an interface for the USIM function. Thus, the user has the capability to convert the UART interface into USIM features. This is done by setting the appropriate bits on the SIM_CFG registers, and using the appropriate lines and signals associated with the external UART interface (as shown in Figure 7-4).

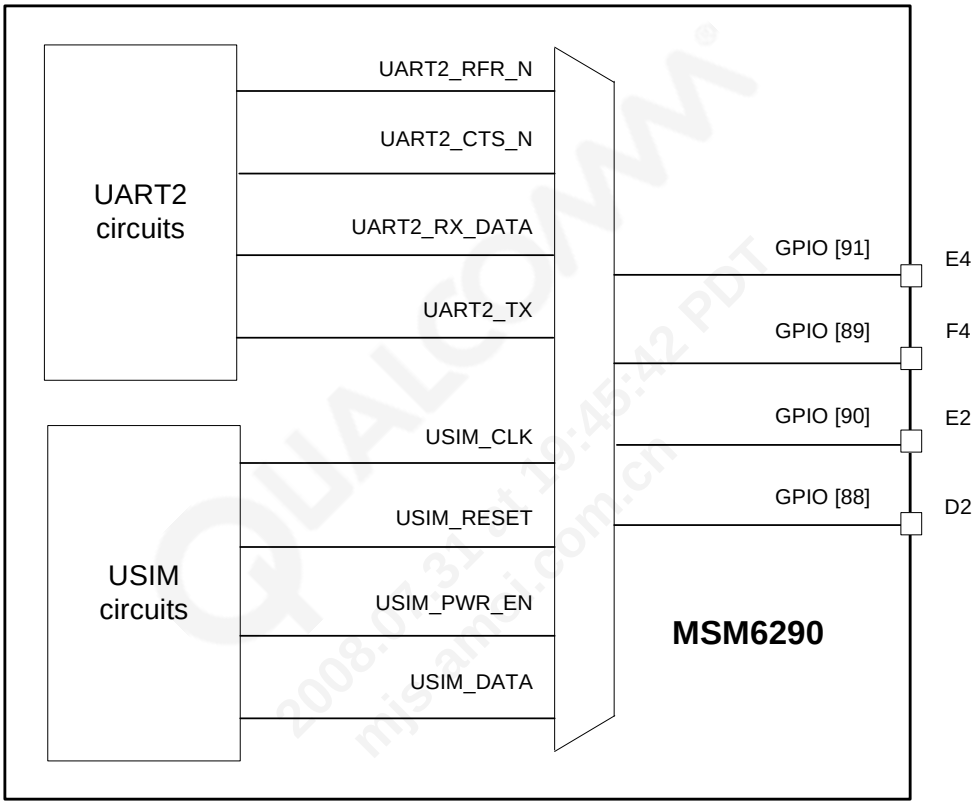


Figure 7-4 Multiplexing arrangement for UART2 and USIM circuits

7.1.10 UART data mover interface (UART1DM)

In order to support a 4 Mbps UART interface and a 1.15 Mbps IrDA data rate, the MSM6290 device includes the UART1DM block for connections to the fast peripheral bus. The UART1DM block is necessary because the standard UART1 interface is not fast enough and is connected to the slow peripheral bus. Additional UART1DM interface features include:

- Up to 4 Mbps maximum speed on UART1 interface
- Medium data rate for IrDA (1.152 Mbps)
- Rate-controlled data mover (separate CRCI channel for Rx/Tx)
- Separate Tx/Rx FIFOs (implemented in one SRAM)
- 32-bit wide AHB interface
- Traditional level interrupts directly to a CPU when no DM is available

The UART1DM interface architecture (Figure 7-5) is based upon UART1, and includes several major functional blocks: registers, controller, memory (RAM), synchronization, clock and bit-rate generator, Tx channel, and Rx channel.

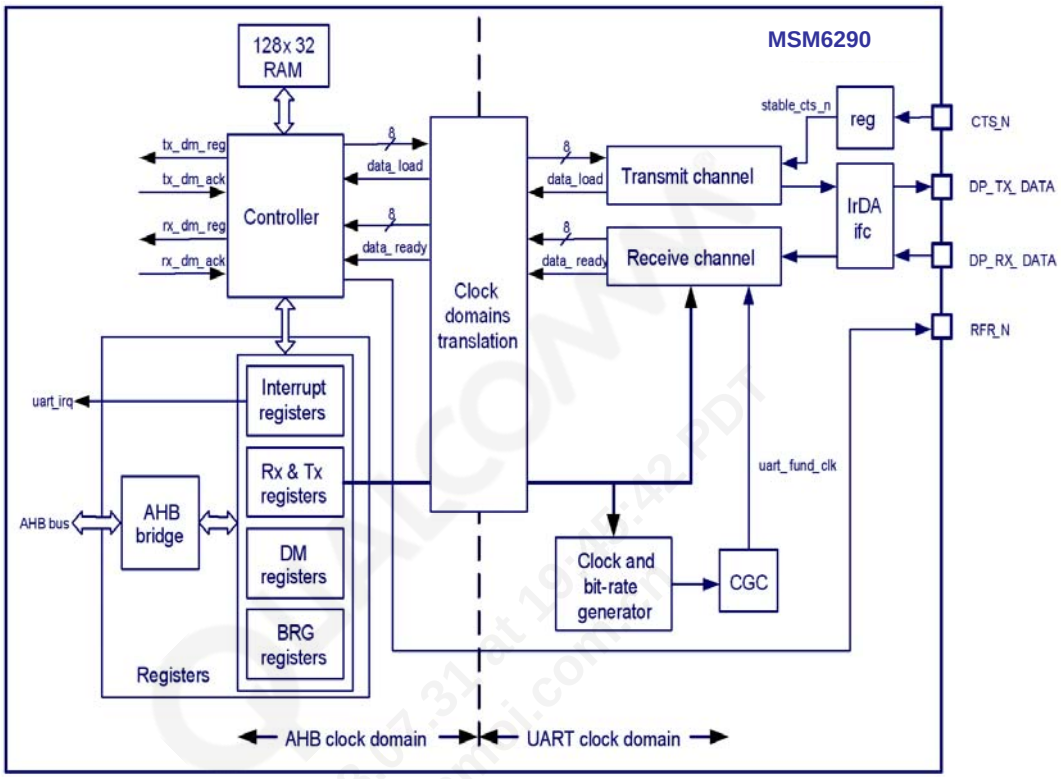


Figure 7-5 UART1DM top-level architecture

Registers

The register block implements an AHB slave. Write and read transactions are enabled from the AHB bus to two internal targets: (1) write/read registers, and (2) the Tx/Rx FIFOs that are implemented as a single RAM.

Controller

The main function of the control block is Rx/Tx FIFO management. The Tx/Rx FIFOs are located within a single RAM and up to four accesses can occur simultaneously: (1) read from the AHB, (2) write to the AHB, (3) read from the Tx channel, and (4) write from the Rx channel. The controller also provides FIFO data packing/unpacking — this is needed since the RAM width is 32 bits while the Rx and Tx channels use 8-bit characters.

Memory

A 128 × 32 RAM is included; it functions as the Tx/Rx FIFOs.

Synchronization (clock domain translations)

The synchronization block is responsible for all clock domain translations. The RAM, registers, and controller run off the AHB clock (in the AHB domain), while Rx and Tx channels run off the UART clock (in the UART domain).

7.1.10.1 Bit rate generator block description

The input clock to the UART can be one of the following frequencies:

- 20 MHz
- 192 MHz (192/4 MHz, 192/3 MHz, 192/2 MHz)
- 266 MHz (266/4 MHz, 266/3 MHz, 266/2 MHz)

Table 7-4 lists the possible values for m, n, and d in order to achieve the requested frequency.

Table 7-4 Clock selection table

UART input clock	20 (MHz)	192 (MHz)	192 (MHz)	192 (MHz)	266 (MHz)	266 (MHz)	266 (MHz)
Data rate	Low UART rates	430.4 (kbit/sec)	1.15 (Mbit/sec)	4 (Mbit/sec)	430.4 (kbit/sec)	1.15 (Mbit/sec)	4 (Mbit/sec)
Target clock (16*DRate)	1.8432 (MHz)	5.5296 (MHz)	18.432 (MHz)	64 (MHz)	5.5296 (MHz)	18.432 (MHz)	64 (MHz)
m	288	18	12	1	611	1152	32
n	3215	625	125	3	29392	16625	133
d	1562	312	62	1	8191	8191	66.5
Duty cycle (average)	50	50	50	33	27	49	50.38
Resulting clock	1.8432 (MHz)	5.5296 (MHz)	18.4320 (MHz)	64 (MHz)	5.5296 (MHz)	18.432 (MHz)	64 (MHz)

In UART1DM, BRG provides enable to the clock gating cell (CGC), which drives both Tx and Rx channels. Such design requires that uart_clk frequency will be 18.4 MHz for 1.15 Mbit/sec data rate. The clock generation is presented below.

The only blocks in the design that operate at high frequency (up to 266 MHz) are the M/N counter and pulse derivation (single flip-flop) circuits shown in Figure 7-1. All other blocks operate at uart_fund_clk (up to 64 MHz for required 4 Mbit/sec). In such implementation, the duty cycle of the uart_fund_clk isn't 50%. Because negative-edge flip-flops are not used, this is not a problem.

As Figure 7-5 shows, the UART receives the source clock and, with the M/N counter, produces the UART fundamental clock (or UART internal clock). The M/N counter is controlled by a set of three registers: M, N, and D. All Tx and Rx logic operates at fundamental frequency. The fundamental clock goes to the divider, which produces rate enables. The rate enable is a single clock (fundamental clock) pulse, which is the UART bit-rate frequency multiplied by 16. For example, if the UART transmits at a rate of 75 bit/sec, the frequency of the rate enable will be 75*16 = 1200 Hz. The multiplication factor of 16 is due to the implementation of the UART.

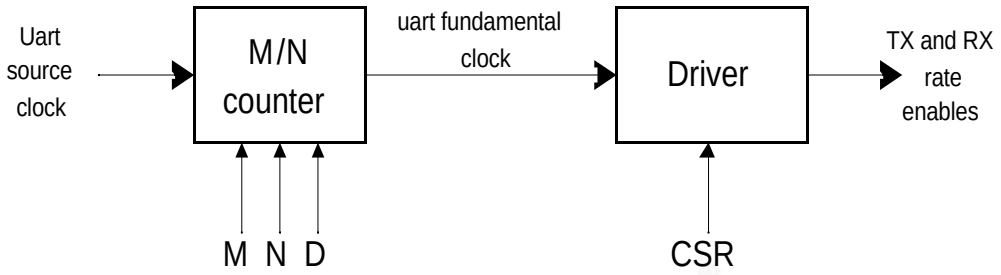


Figure 7-6 Bit-rate generator configuration

The rate enables are produced from the fundamental clock. That's why the frequency of the fundamental clock should be the highest UART bit rate multiplied by 16 (i.e., the fundamental clock frequency can't be lower than rate enables frequency).

The divider includes several counters operated at the fundamental frequency, which produces rate enables. Table 7-5 lists the possible outputs of the divider.

Table 7-5 CSR value vs. clock division

CSR value	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Division by	1536	768	384	192	96	48	32	24	16	12	8	6	4	3	2	1

For example, if a CSR value of four was selected, the rate enable frequency will be the fundamental clock frequency divided by 96, and the UART bit rate will be the rate enable frequency divided by 16.

The UART in the MSM6290 device should support several bit rates:

- Standard rates from 75 to 115.2 kbit/sec
- 1.152 Mbit/sec (IRDA medium rate)
- Optionally 4 Mbit/sec bit rate

Table 7-6 lists the fundamental clock frequencies.

Table 7-6 Supported bit rates vs. fundamental clock

Rates	Fundamental clock frequency
75-115.2 kbit/sec	1.8432 MHz
1.152 Mbit/sec	18.432 MHz
4 Mbit/sec	64 MHz

In the first case (for 75-115.2 kbit/sec rates), the CSR selects one of the rates listed in Table 7-7. In the case of 1.152 Mbit/sec or 4 Mbit/sec, the CSR value is constant 0xF.

Table 7-7 Bit rates vs. CSR value (UART fundamental clk = 1.8432 MHz)

CSR	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Bit rate	75	150	300	600	1.2 K	2.4 K	3.6 K	4.8 K	7.2 K	9.6 K	14.4 K	19.2 K	28.8 K	38.4 K	57.6 K	115.2 K

After the fundamental clock is chosen and UART source clock is known, the M/N counter registers should be calculated.

The basic idea is: M/N = fundamental clock frequency/UART source frequency

=> M = fund. clock/GCD (fund. clock, source clock)

=> N = source clock/GCD (fund. clock, source clock)

Where GCD is the greatest common divider.

D = N/2 (for 50% duty cycle).

Pay attention to the M/N register's width. Sometimes, ideal values of M and N exceed the maximum possible value of the register.

The last phase, after M, N and D are calculated, is calculation of the MREG, NREG, and DREG (i.e., actual value, which is written to the registers). In order to simplify HW implementation, NREG includes NOT(N-M) value (this value is also supplied by the script).

7.1.10.2 UART1DM operation

As already mentioned, the UART1DM is connected to the AHB bus and includes two rate-controlled (CRCI) data mover interfaces (one for Rx transfers and one for Tx transfers). All configuration registers are in the AHB clock domain, thereby optimizing the AHB bus utilization throughout the configuration phase. Most registers are configuration registers so they do not require synchronization — they undergo a soft reset along with the UART block after the configuration phase and will only be enabled after a transfer.

The Rx path of the UART is totally independent from the Tx path so the UART block can receive data during an active Tx transfer. The only Rx/Tx commonality is the shared SRAM that implements the data paths' FIFOs. Sharing the SRAM to implement the Tx FIFO and the Rx FIFO occurs during the initialization phase and the memory space is not necessarily shared equally.

The UART1DM block is designed to work with or without the rate controlled data mover request acknowledge interface. One of four methods used to set up a Tx or Rx transfer with or without the CRCI DM interface (detailed below).

Method 1: Tx transfer when disabling TX-DM CRCI channel

1. Initialize the UART.
2. Soft reset the UART.
3. The UART sends an interrupt each time the FIFO holds less than a pre-programmed number of characters. This interrupt signals the CPU that a new data burst can be sent to the UART Tx block.
4. The UART transmits character-by-character as long as there is a valid character in the transmit FIFO.

Method 2: Tx transfer when enabling TX-DM CRCI channel

1. Initialize the UART.
2. Soft reset the UART.
3. Initialize the DM and make sure that the EBI memory includes the transfer.
4. Enable the TX-DM mode then initialize the UART's TX_data_length counter.
5. The UART sends a tx_dm_req to the DM each time there is enough space for a new data burst in the transmit FIFO. The DM responds to the request by sending an acknowledge and then sends the data burst via the AHB bus to the transmit FIFO. The UART continues to send requests until the TX_data_length counter is zero; the transfer is then complete.
6. The UART transmits character-by-character as long as there is a valid character in the transmit FIFO.

Method 3: Rx transfer when disabling RX-DM CRCI channel

1. Initialize the UART.
2. Soft reset the UART.
3. The UART starts collecting characters as they are received via the receive channel and writes them to the Rx FIFO. The UART sends an interrupt each time the FIFO holds more than a pre-programmed number of characters (level interrupt) or whenever no characters are received over a pre-programmed duration while the FIFO is empty (stale interrupt). This interrupt signals the CPU that newly received data can be read from the Rx FIFO.

Method 4: Rx transfer when enabling RX-DM CRCI channel

1. Initialize the UART.
2. Soft reset the UART.
3. Initialize the DM.
4. Enable the RX-DM mode.
5. Enable the UART Rx path.
6. The UART starts receiving characters and a request is sent to the DM each time the Rx FIFO is ready for a new data burst. After each character transfer, the stale timer is loaded; when a stale time-out occurs (no new characters were received) the last burst is filled with non-valid characters and a request is sent to the DM. An interrupt is sent to the ARM immediately after transferring the last burst to the DM. The UART also stores the number of valid characters received up to the point that the stale interrupt bit was cleared or reset.

The UART1DM Tx and Rx channels are similar to the UART channels except that the FIFO controls and IRQ generation have been moved to the control block.

7.1.11 IrDA interface

The IRDA interface block converts Rx data from IrDA format to UART format, and Tx data from UART to IrDA format. The conversion is performed if IrDA is enabled (IRDA[0]); otherwise, the data is fed through. UART1DM IrDA block diagram is shown in Figure 7-7.

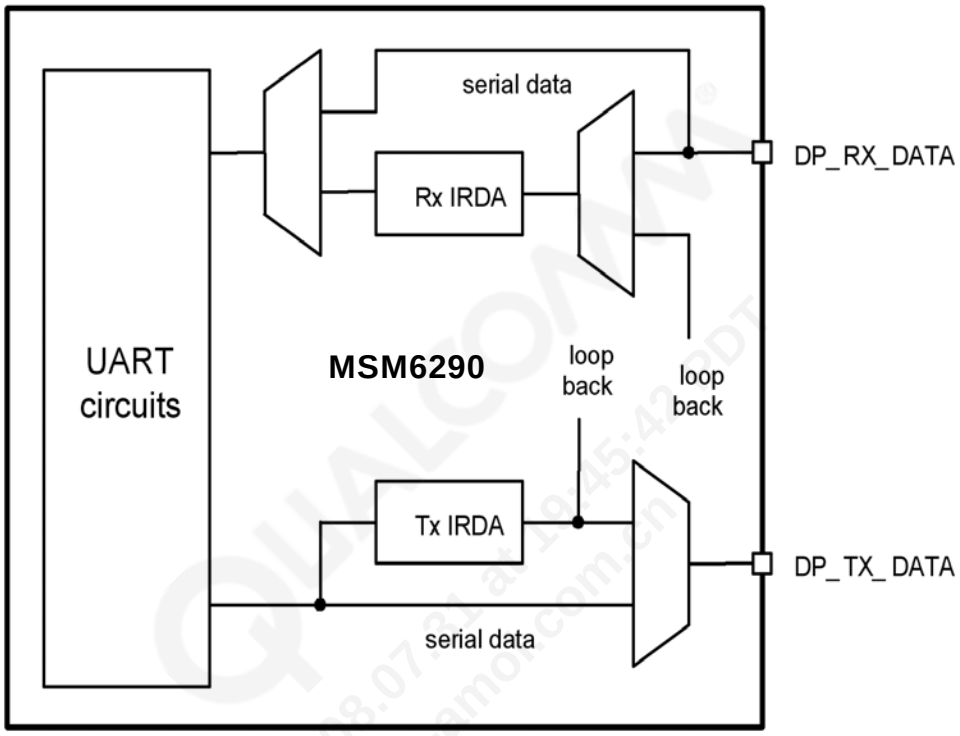


Figure 7-7 UART1DM IrDA block diagram

The IrDA interface through the UART1DM supports 1.15 Mbits/sec. medium data rate. In order to support this data rate, the UART must run at a high clock frequency of 16*1.15 = 18.4 MHz.

For 1.15 Mbits/sec. data rate, the CSR value needs to be 0x0F when IrDA is enabled. In order to support 115.2 kbit/sec. rate, the fundamental clock should multiplied by 2 (requires changes in the M, N, D, MND registers) and CSR should be configured to 0xE.

The UART1DM IrDA interface can be enabled by setting bit0 in the UART1DM_IRDA register. The medium rate can also be enabled using this register. By setting bit 4, we can enable medium rate (1.15 Mbits/s) IrDA data transfer.

7.1.11.1 IrDA timing specification

The IRDA for MSM6290 supports the 1.15 Mbit/sec medium data rate. In general, that means the UART must run at the higher clock of 16*1.15 = 18.4 MHz.

According to the IrDA physical layer standard for all signaling rates up to and including 115.2 kbit/s, the pulse duration is 3/16 of the bit duration. For 0.576 Mbit/s and 1.152 Mbit/s, the pulse durations are 1/4 of the bit duration.

The CSR value cannot be 0xF when the IRDA is enabled. In order to support 115.2 kbit/sec rate, the fundamental clock should be multiplied by two (requires changes in the M, N, and D registers), and CSR should be configured to 0xE.

Figure 7-8 shows the IRDA waveform.

Table 7-8 lists the IRDA rate and pulse duration specifications.

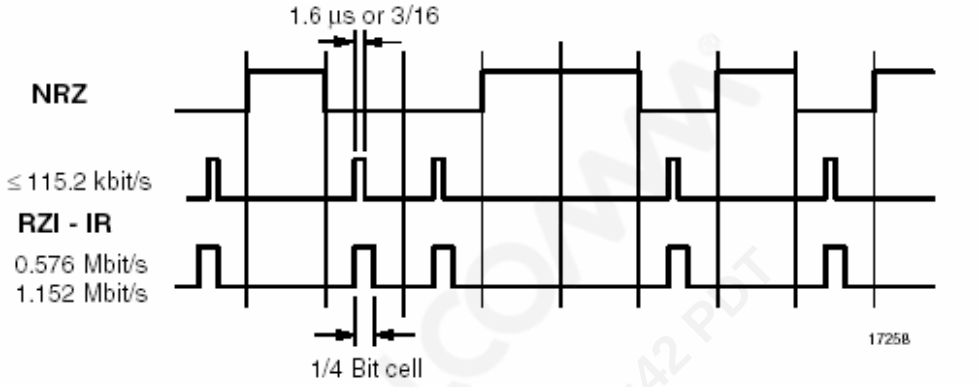


Figure 7-8 IRDA waveform

Table 7-8 IRDA rate and pulse duration specification

Signaling rate	Modulation	Rate tolerance (% of rate)	Pulse duration minimum	Pulse duration nominal	Pulse duration maximal
2.4 kbits/s	RZI*)	±0.87	1.41 µs	78.13 µs	88.55 µs
9.6 kbits/s	RZI*)	±0.87	1.41 µs	19.53 µs	22.13 µs
19.2 kbits/s	RZI*)	±0.87	1.41 µs	9.77 µs	11.07 µs
38.4 kbits/s	RZI*)	±0.87	1.41 µs	4.88 µs	5.96 µs
57.6 kbits/s	RZI*)	±0.87	1.41 µs	3.26 µs	4.34 µs
115.2 kbits/s	RZI*)	±0.87	1.41 µs	1.63 µs	2.23 µs
0.576 Mbit/s	RZI*)	±0.1	295.2 ns	434.0 ns	520.8 ns
1.152 Mbit/s	RZI*)	±0.1	147.6 ns	217.0 ns	260.4 ns

7.2 USIM interface

7.2.1 General description

The USIM is a smart card for UMTS/GSM cellular applications. The USIM provides the required subscription information to allow the mobile equipment to attach to a GSM or UMTS network. The USIM also provides the subscriber's verification procedures as well as authentication methods for network authentication during the attach procedures. The USIM card can be inserted into any UMTS/GSM USIM equipped handset, allowing the user to receive or make calls, and receive other subscribed services from any USIM equipped handset, thus enabling more handset independence for the user.

7.2.2 USIM interface description

Upon powerup or after a soft-reset, the clock and data lines to the USIM will be active as they go through the initialization process. This will occur one slot at a time.

After initialization, the operation of the two slots depends on whether a card is detected in a slot or not. When operating with a card, the data line always stays in the active “high” (marking state). Even though the line is not actively transmitting between accesses, it is still active. The clock, on the other hand, is only on when actively reading the card, and is turned off between accesses. The state of the clock when it is turned off depends on the characteristics of the card, and may be high or low. However, even between accesses, the interface is still active. The data, reset, and power lines all remain high (when reset is active low). The clock is turned off in order to save power and is turned on in order to access the card.

If for some reason the slot cannot be communicated with (card not inserted, card not recognized, broken connection, etc.), the interface is deactivated. In this state, all lines are low (again, clock state depends on characteristics of card), and there is no chance to operate or communicate with the card without re-initializing.

Once a USIM card is inserted and initialized on powerup, the interface is always on (as described above), even when the MSM device is in sleep mode. This characteristic can give rise to current consumption problems. It is important to understand that only upon powerup of the MSM, and not during regular operation, can a USIM card be recognized and initialized.

7.2.3 USIM implementation

USIM can be implemented on the MSM6290 device using the UART2 interface. The MSM6290 device supplies USIM_CLK, USIM_DATA, USIM_PWR_EN, and USIM_RESET. All pins are hidden behind UART2. A brief summary of the functional mode MUX at the pins for UART2 is shown in Table 7-9.

Table 7-9 GPIO pins for USIM/UART2

Primary pin	USIM	UART2
GPIO90	USIM_RESET	DP_RX_DATA
GPIO91	USIM_CLK	RFR_N
GPIO88	USIM_DATA	DP_TX_DATA
GPIO89	USIM_PWR_EN	CTS_N

In addition to those signals provided to USIM by the MSM device, there is also a SIM_CFG register on the UART2 interface that can be programmed by the user for different SIM functions.

7.3 USB interface

7.3.1 Introduction

The MSM device contains a USB interface. This interface is compliant with the USB 2.0 specification.

When two devices are connected via a USB interface, one of the devices must act as a host, and the other device must act as a peripheral. The host is responsible for initiating and controlling traffic on the bus. The USB specification requires personal computers (PCs) to act as hosts, and other devices such as printers, keyboards, mice, etc. to act as peripherals.

The USB 2.0 specification requires hosts such as PCs to support all three USB speeds, namely low-speed (1.5 Mbps), full-speed (12 Mbps) and high-speed (480 Mbps). The USB 2.0 specification allows peripherals to support any one or more of these speeds.

NOTE Theoretical high-speed USB interface throughput is 480 Mbps. However, due to system overhead, expected throughput will be much lower than 480 Mbps.

The MSM device is compliant with USB 2.0 specification. The MSM device supports full-speed and high-speed when acting as a peripheral, and low-speed, full-speed, or high-speed as a host.

There are two USB controllers embedded in the MSM6290 device.

1. Primary USB controller
2. Secondary USB controller

Both are high-speed USB controllers. The primary USB port has an integrated controller and PHY, and can operate at all three speeds. The secondary USB port has an integrated controller. It can only operate at full-speed. Figure 7-9 shows the architecture for both primary and secondary USB controllers.

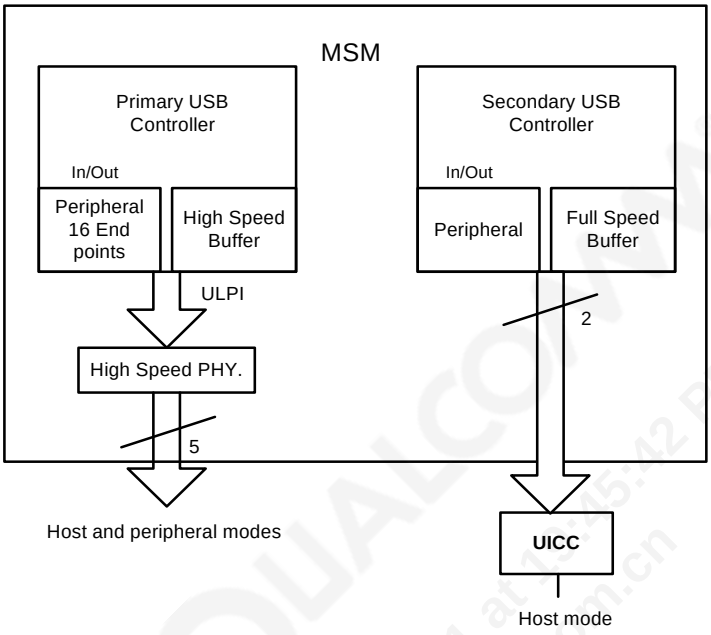


Figure 7-9 Primary and secondary USB controller architecture in the MSM6290 device

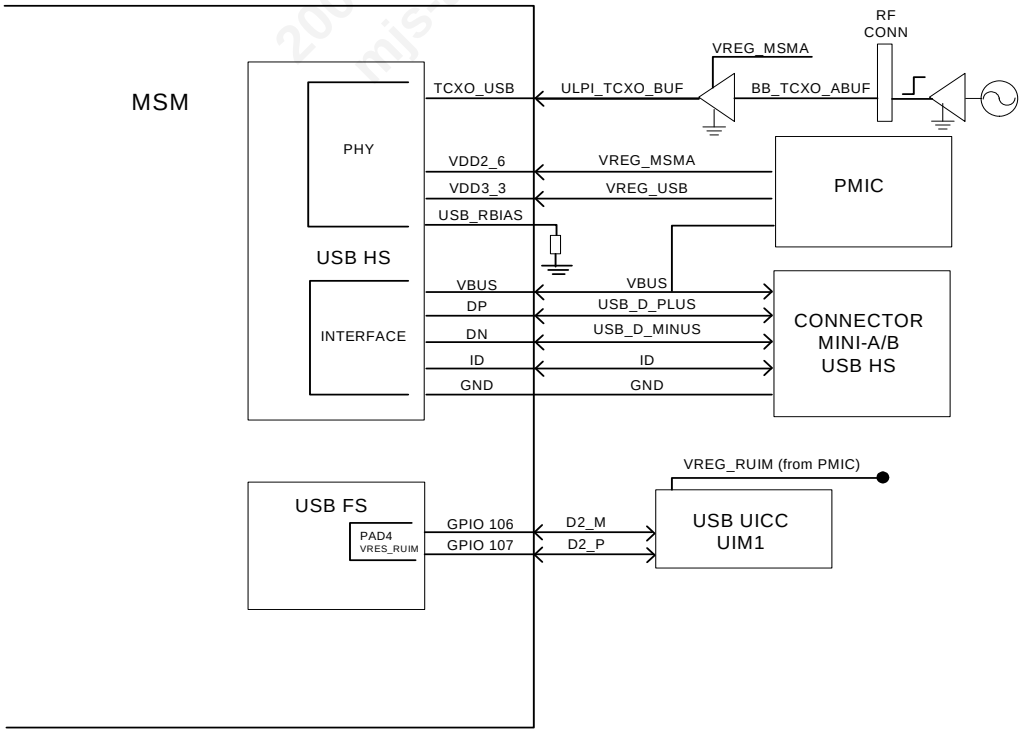


Figure 7-10 High-level diagram of USB architecture

7.3.1.1 Primary USB controller

Below are the primary USB controller features:

- High-speed USB controller (specification 2.0)
- Acts as both host and peripheral.
- In peripheral mode, the controller has 16 IN/OUT pairs of endpoints. In host mode, the number of possible endpoints is determined by software, and can be greater than 16.
- Built-in high-speed PHY
- Five USB pins dedicated for high-speed USB interface. They are:
  - USB_VBUS
  - USB_DP
  - USB_DN
  - USB_ID
  - USB_GND

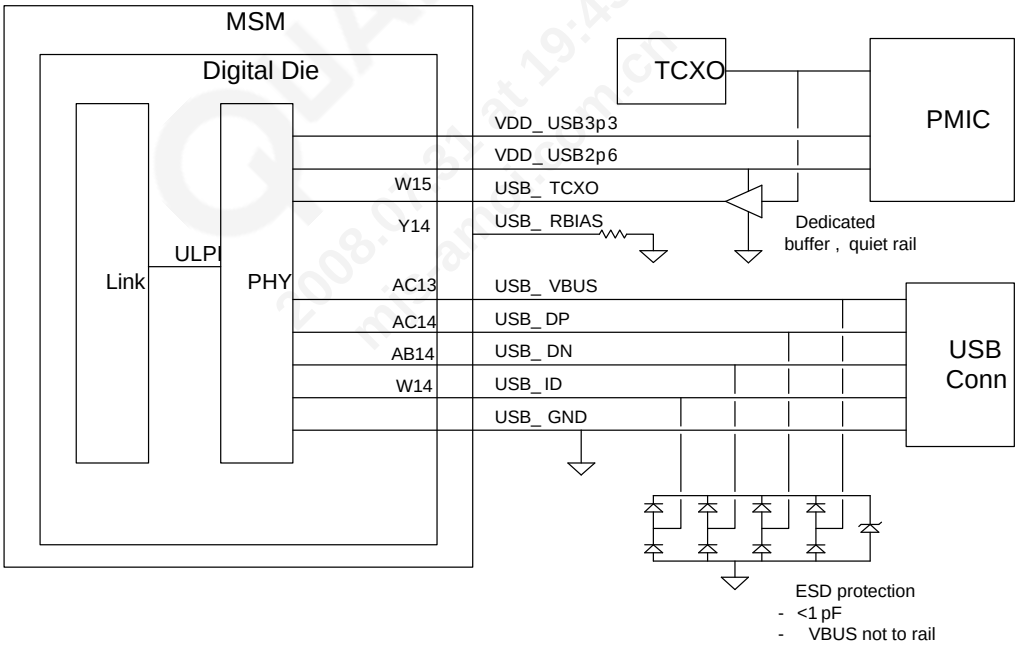


Figure 7-11 High-level USB architecture

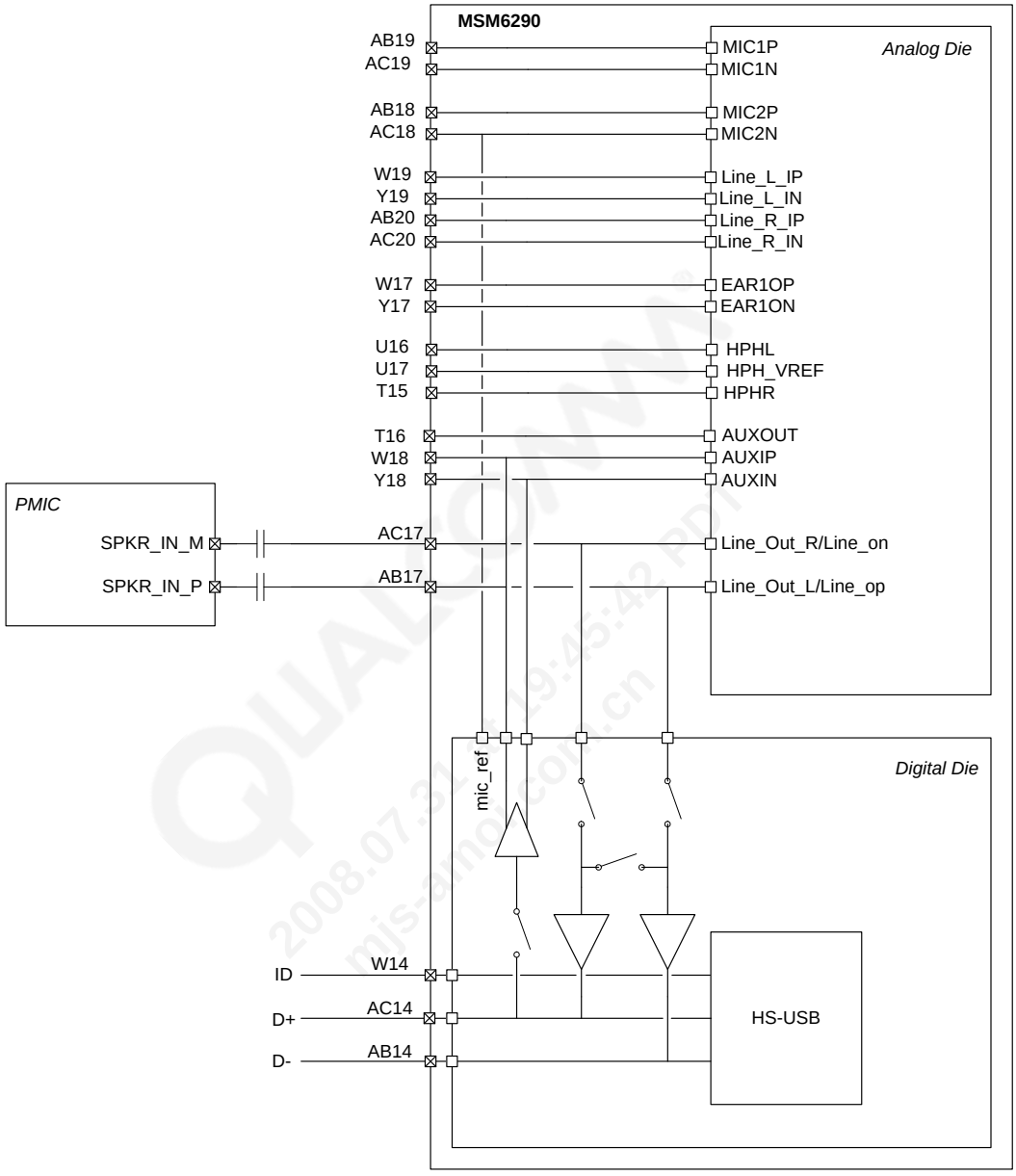


Figure 7-12 USB audio routing diagram

7.3.1.2 Peripheral applications

Given the appropriate software, the primary USB controller can support the below applications while acting as peripheral:

- Mass storage
- MTP for music and video content transfers
- CDC/ACM for modem
- CDC/ECM for data services
- CDC/OBEX for NMEA and diagnostic
- SICD for PictBridge

7.3.1.3 Host applications

Given the appropriate software, the primary USB controller can support the below applications while acting as a host:

- HID supporting keyboard
- Mouse and gamepad controller connectivity
- Mass storage supporting USB flash drive and HDD connectivity

7.3.1.4 Endpoints

The primary controller supports 16 IN/OUT pairs of endpoints when acting as a peripheral. When acting as a host, the number of possible endpoints is determined by software, and can be greater than 16.

Each endpoint in either host or peripheral mode can be configured for control, interrupt, or bulk. Isochronous endpoints are supported in hardware, but are not yet used by any software application.

7.3.2 Secondary USB controller

Below are the secondary USB controller features:

- The secondary controller has the same logic and software interface as the primary USB port.
- The secondary port does not have an integrated high-speed PHY.
- The secondary port has smaller buffers than the primary port and only a full-speed transceiver interface. Thus, the secondary port can only support full-speed operation.
- It supports the USB UICC (host mode) functionality.

7.3.2.1 Host application

The secondary USB controller can support USB UICC operation while acting in host mode. Due to a maximum buffer size limitation in the secondary USB controller, the USB UICC interface's maximum speed is up to full speed USB interface (12 Mbps).

Two pins are dedicated for this operation along with the three USIM pins. Those are:

- UICC_USB_VP
- UICC_USB_VM
- USIM_RESET
- USIM_CLK
- USIM_DATA

All five pins are tolerant to 3.0 V and are connected to the VDD_P4 voltage pad, which can be operated at 2.85 V (typ) or 1.8 V (typ).

7.3.3 PMIC interface

The MSM6290 device interfaces to a PM6658/PM6653 power management IC (PMIC) using the architecture shown in Figure 7-13.

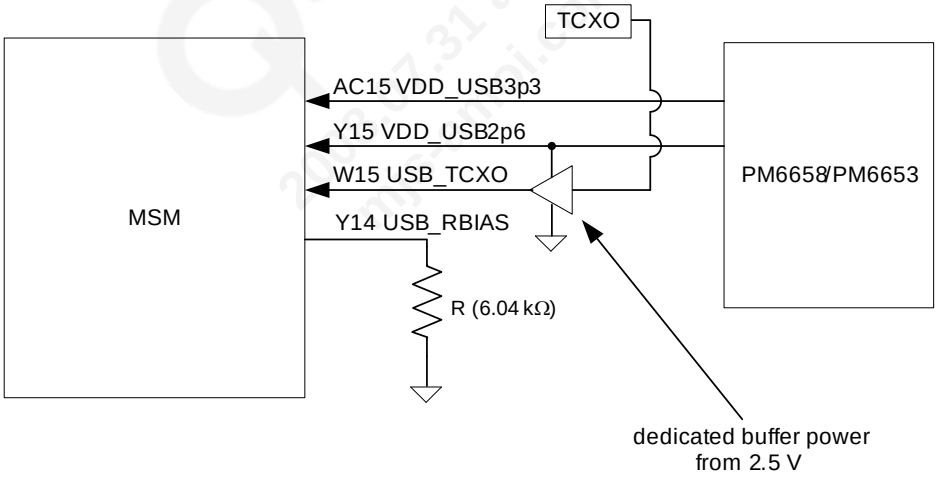


Figure 7-13 PMIC interface

Two power supplies are required from the PM6658/PM6653 device for both primary and secondary USB operations. Those are VDD_USB3P3 and VDD_USB2P6. One resistor (USB_RBIAS) is required for biasing purposes.

8 User Interface

The user interface of the MSM6290 devices consists of:

- Keypad[4:0]
- Ringer
- M/N counter
- General purpose clock
- I2C
- HKADC

8.1 Keypad interface

The KEYSENSE[4:0] pins can be used to connect a matrix keypad to the MSM6290 devices. The KEYSENSE[4:0] inputs assert a KEYSENSE_INT if any of the pins are pulled low. Figure 8-1 shows the circuits associated with KEYSENSE[4:0].

External pull ups are required on these lines. See the *MSM6290 Baseband Reference Schematic* (80-VF866-41) for more details.

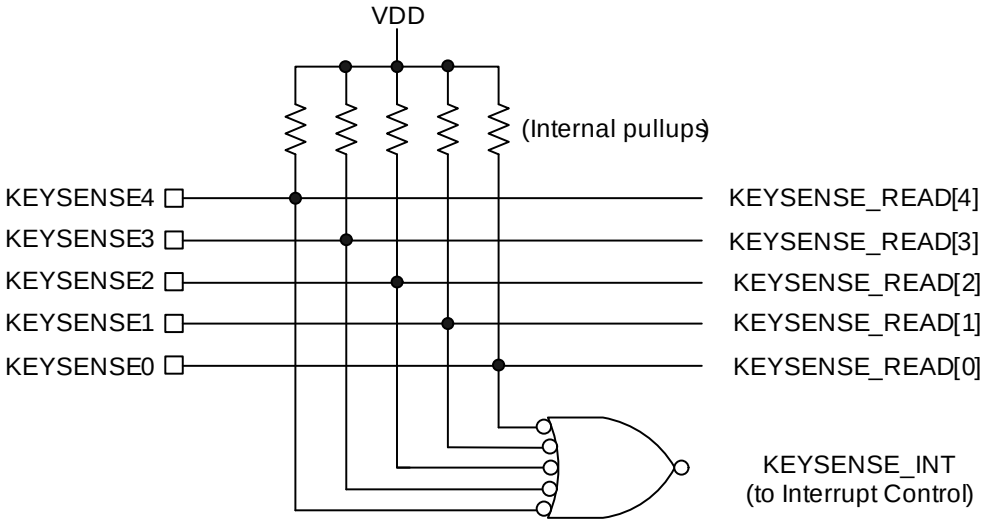


Figure 8-1 KEYSENSE[4:0] circuits

8.2 Ringer

The ringer generation circuit is programmed to output single tones or DTMF tone pairs on the RINGER pin. The ringer generation circuit produces and sums two different user-programmable frequencies. Single tones can be generated by programming two tones of the same frequency. The ringer function is disabled when RESOUT is asserted.

Figure 8-2 shows the ringer generation circuit block diagram. Each M/N counter is clocked by a 300 kHz clock derived from a 19.2 MHz TCXO. The ringer generation circuit also controls the envelope for the ring output waveform. The signal on the RINGER pin is a digital pulse stream. The RINGER output generally requires an external booster circuit, such as the one shown in Figure 8-3, to drive a sound transducer.

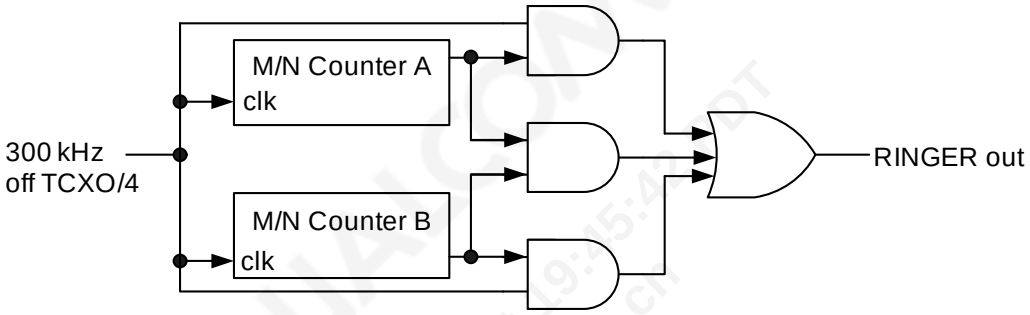


Figure 8-2 Ringer generation circuit

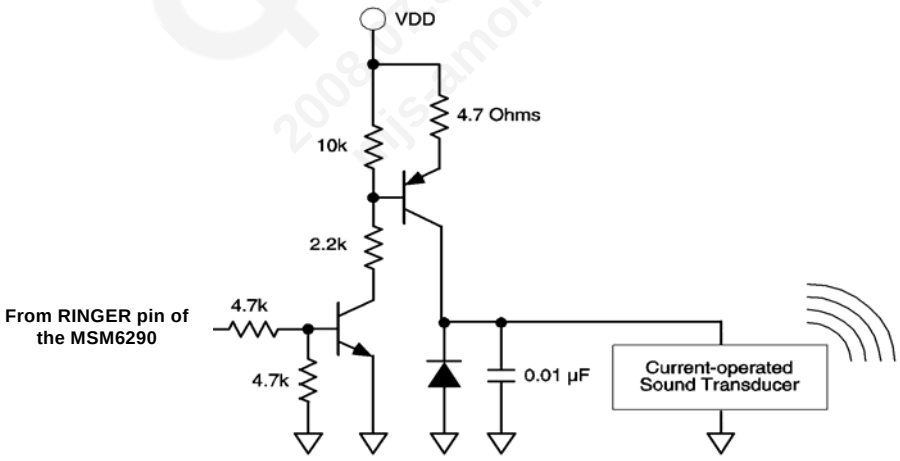


Figure 8-3 External driver circuit example

To generate DTMF tones on the RINGER output, M, N, and D values must be written into the two identical M/N counters. M is a 6-bit value, N is a 13-bit value, and D is a 12-bit value. The 6-bit M value (RINGER_MN_A_MDIV:RINGER_MN_A_MDIV) is written along with the enable for the counter (RINGER_MN_A_MDIV:RINGER_MN_A_EN) to the A counter M-value register. RINGER_MN_A_NDIV is written to the A counter N-value register as the one's-complement of the N – M value. The D value in RINGER_MN_A_DUTY is written to the A counter D-value register. These register allocations are identical for the B counter. Table 8-1 lists the decimal values for M and N which are used to program the M/N counters for DTMF tones. The frequencies produced are accurate to better than 1.5% of their specified values for a TCXO of 19.2 MHz.

Table 8-1 Standard DTMF frequencies and ringer programming values

DTMF frequency (Hz)	M/N (dec)	M (hex)	N (hex)	1's complement of (N minus M) (hex)	D duty-cycle (50% of N) (hex)
350	7 / 6000	07	1770	0896	BB8
440	7 / 4772	07	12A4	0D62	952
480	8 / 5000	08	1388	0C7F	9C4
620	16 / 7742	10	1E3E	01D1	F1F
697	17 / 7317	11	1C95	037B	E4A
770	20 / 7792	14	1E70	01A3	F38
852	12 / 4225	0C	1081	0F8A	840
941	9 / 2869	09	0B35	14D3	5A9
1209	32 / 7940	20	1F04	011B	F82
1336	6 / 1347	6	0543	0AC2	2A1
1477	26 / 5281	1A	14A1	0B78	A50
1633	23 / 4225	17	1081	0F95	840

The DTMF frequency is found by multiplying the ratio M/N (decimal equivalent values) by the clock frequency of 300 kHz. The one's-complement of (N – M) value is the actual value for programming the N register to get the desired N value. The D value acts as a loudness control. Maximum RINGER loudness is set by using a D value which is 50% of the value in the N column. Softer ringing is achieved by decreasing or increasing the D value.

Each counter (A and B) is enabled by setting (1) bit 6 of the M register. For counter A, this is RINGER_MN_A_EN in the RINGER_MN_A_MDIV register. For counter B, it is RINGER_MN_B_EN in the RINGER_MN_B_MDIV register. Setting bit 6 of the M register resets both counters (CNTR_A and CNTR_B) to start counting at the same time. Both counters can be programmed to generate the same or different frequencies. Clearing (0) bit 6 disables both counters. These M register bits have an edge detect for a zero-to-one transition. To generate a single tone, the counters are phase-locked by clearing bit 6 of both M registers every time the counters are programmed. Bit 6 must be cleared (0) before the N and D registers are programmed and remain cleared for at least 1 ms. The last step of the programming sequence is to set each bit 6 along with its corresponding M value. This triggers the synchronization mechanisms. The standard DTMF keypad tone combinations are listed in Table 8-2.

Table 8-2 Keypad frequencies

Keypad	Counter A frequency (Hz)	Counter B frequency (Hz)
1	697	1209
2	697	1336
3	697	1477
4	770	1209
5	770	1336
6	770	1477

Table 8-2 Keypad frequencies (cont.)

Keypad	Counter A frequency (Hz)	Counter B frequency (Hz)
7	852	1209
8	852	1336
9	852	1477
*	941	1209
0	941	1336
#	941	1477
A	697	1633
B	770	1633
C	852	1633
D	941	1633

8.3 M/N counter

GP_MN is a general-purpose M/N counter. The nominal output frequency is (M/N) × TCXO/4. Figure 8-4 shows the block diagram of the GP_MN M/N counter.

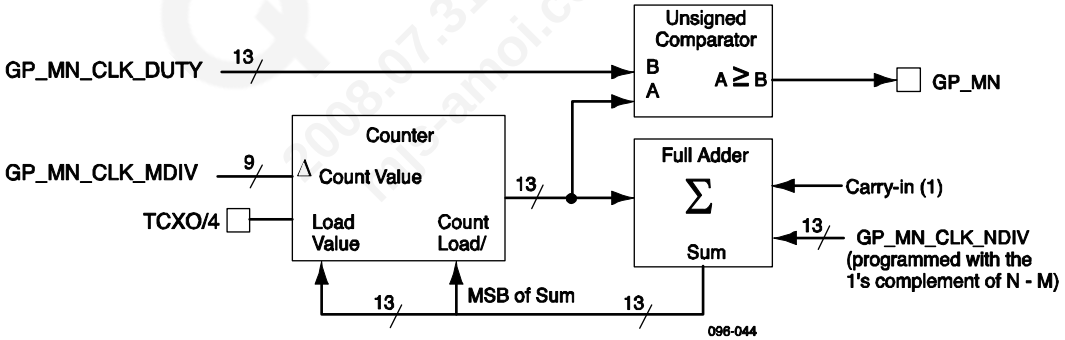


Figure 8-4 GP_MN block diagram

To create an output frequency, *f*, with a given duty cycle, *D*, the following sequence must be used to initialize the GP_MN M/N counter:

1. Solve for decimal values for M and N. The value of M cannot exceed the value of N.
2. Convert the decimal values for M and N into a 9-bit value for M and a 13-bit value for N.
3. Write the 9-bit M value into GP_MN_CLK_MDIV.
4. Write the 13-bit result of the one's complement value of N – M into GP_MN_CLK_NDIV. In other words, GP_MN_CLK_NDIV = 0x1FFF – (N – M).
5. Write the desired 13-bit duty cycle value, *D*, into GP_MN_CLK_DUTY. A 50% duty cycle output waveform is generated when: GP_MN_CLK_DUTY = N/2.

GP_MN can also be used as a programmable digital output. Table 8-3 shows a set of values and the resulting GP_MN output state.

Table 8-3 Using GP_MN as a digital output

GP_MN output state	GP_MN_CLK_MDIV	GP_MN_CLK_NDIV	GP_MN_CLK_DUTY
LOW	0x00	0x1000	0x1FFF
HIGH	0x00	0x1000	0

8.4 General purpose clock

The general purpose clock (GP_CLK) can be MUXed out of GPIO19 or GPIO44 by selecting the appropriate GPIO_CFG[5:2] value. For GPIO19, this value is xx10 and for GPIO44 it is xx10 (where x is “do not care”).The clock speed of GP_CLK is selected by programming the GP_CLK_MD and GP_CLK_NS registers. The clock source is selected by the CLK_SRC_SEL bits and can be TCXO (19.2 MHz), Sleep XTAL (32.768 kHz), USB XTAL (48 MHz), PLL0, PLL1, PLL2, or PLL3. This clock is then passed through a pre-divider of 1, 2, or 4 (as selected by the PRE_DIV_SEL bits). The divided clock is then either sent to the GP_CLK M/N, or the M/N can be bypassed with the MNCNTR_SEL bit. The M_VAL, N_VAL, and D_VAL bits set the divide ratio for the M/N counter.

The maximum speed that the pin driver can support is approximately 75 MHz.

8.4.1 Overview

The I^2C-bus supports any IC fabrication process (NMOS, CMOS, bipolar). Two wires, serial data (SDA) and serial clock (SCL), carry information between the devices connected to the bus. Each device is recognized by a unique address (whether it’s a microcontroller, LCD driver, memory or keyboard interface) and can operate as either a transmitter or receiver, depending on the function of the device.

A simplified version of the I^2C bus operation is as follows:

1. The master generates a START condition, signaling all ICs on the bus to listen for data.
2. The master writes a 7-bit address, followed by a read/write bit to select the device as a transmitter or receiver.
3. The receiver sends an acknowledge bit over the bus. The transmitter must read this bit to determine whether or not the addressed device is on the bus.
4. Depending on the value of the read/write bit, any number of 8-bit messages can be transmitted or received by the master. These messages are specific to the I2C device used. After eight message bits are written to the bus, the transmitter will receive an acknowledge bit. This message and acknowledge transfer continues until the entire message is transmitted.
5. The message is terminated by the master with a STOP condition. This frees the bus for the next master to begin communications.

8.4.2 General characteristics

Both SDA and SCL are bidirectional lines, connected to a positive supply voltage via a current-source or pull-up resistor. When the bus is free, both lines are HIGH. The output stages of devices connected to the bus must have an open-drain or open-collector to perform the wired-AND function. Data on the I^2C-bus can be transferred at rates of up to 100 kbps in the standard-mode, up to 400 kbps in the fast-mode, or up to 3.4 Mbps in the high-speed mode. The number of interfaces connected to the bus is solely dependent on the bus capacitance limit of 400 pF.

NOTE Current MSM6290 chipsets do not support high-speed mode. Also, it is difficult to achieve full 400 kbps in the fast mode.

For further details on I^2C, see *I2C Controller and I2C Bus Specification in MSM6XXX Chipsets* (80-V7836-1).

8.4.3 Implementation

The serial data is connected to the GPIO26 and the serial clock is connected to GPIO27 of the MSM6290 devices.

Table 8-4 I2C pins

Primary pin	I2C signals
GPIO26	I2C_SDA
GPIO27	I2C_SCL

8.5 HKADC

The MSM6290 device has an on-chip 8-bit analog-to-digital converter (ADC) that is available for digitizing analog signals representing parameters such as battery voltage, temperature, and RF power levels. These parameters support handset-level housekeeping functions — various tasks that must be performed to keep the "house," or handset, in order. Thus the term housekeeping ADC (HKADC) is used.

The MSM6290 TS/HKADC is an 8-bit successive-approximation circuit (Figure 8-5).

The resolution of the ADC is 8-bit.

The TS/HKADC block diagram is shown in Figure 8-5. The TS/HKADC contains four main sub-blocks: a 6-channel MUX, a sample and hold circuit, a digital-to-analog converter, and a comparator latch block.

- Six channel MUX: TS/HKADC contains a six channel MUX.
  - Six inputs are available as general-purpose inputs serving house keeping functions.
  - The analog multiplexer is controlled by the MUX_SEL_IN bits within the TSHK_INTF_CONFIG register.
- Sample and hold circuit: this sample-and-hold circuit samples the input voltage and holds it at one of the input terminals of the comparator. This sampled voltage is held until the end of the full 8-bit conversion.
- Digital-to-analog converter: The digital-to-analog (DAC) converter is designed to provide 8-bits of resolution.
- Comparator and latch

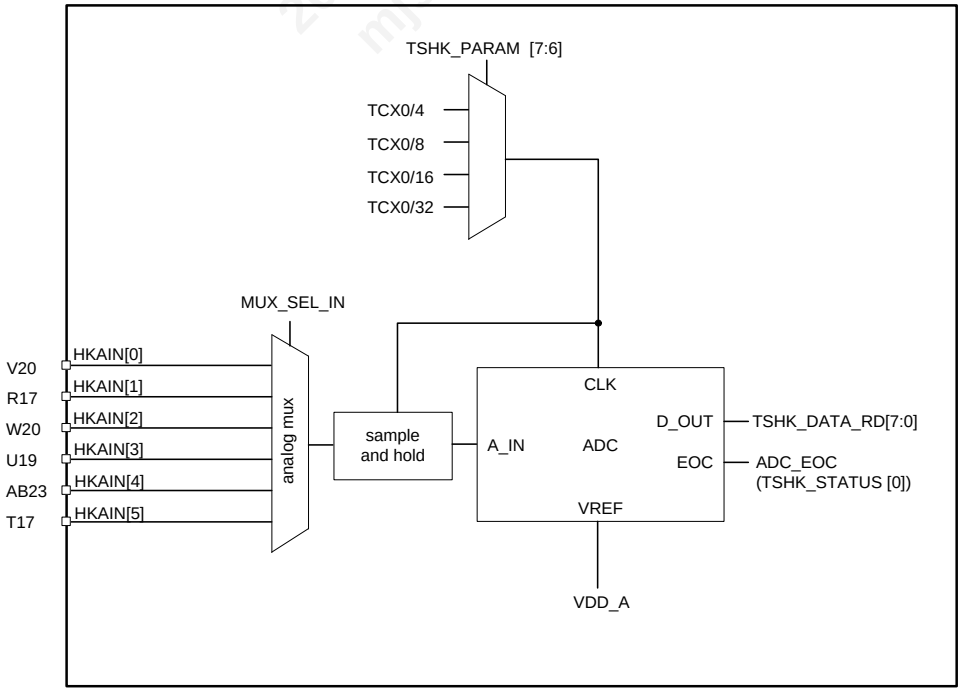


Figure 8-5 HKADC architecture

The output of the multiplexer is routed to a sample-and-hold circuit that provides the ADC input; this sampled voltage is held until the end of the full 8-bit conversion. As mentioned earlier, the ADC is implemented using a 8-bit successive-approximation circuit.

8.5.1 Analog input voltage range

The ADC input voltage must be within the range of 0 to VREF volts where VREF is VDD_A. The ADC transfer function depends upon the VREF, as shown in Table 8-5.

Table 8-5 MSM6290 ADC transfer

Input voltage range	8-bit digital output result HKADC_DATA_RD[7:0]
1xx/VDD_A	
>VDD_A	0xFF
(4095/4095)VDD_A)	0xFF
(2048/4095)VDD_A	0x80
0.000	0x00

8.5.2 HKADC operation

Four clock sources are available for the HKADC. A clock source based on the frequency of TCXO- divided-by-four is the default clock source for the HKADC. The alternate clock sources can be selected by TSHK_PARAM [7:6].

The analog-to-digital conversion process of the HKADC requires 16 cycles of the HKADC Clk.

The general HKADC conversion process is shown in Figure 8-6. The HKADC must be enabled and powered up for at least five microseconds prior to the beginning of any conversion. Once the powerup requirement has been met, the first step in the conversion process is to switch the analog input multiplexer to the desired channel by writing the MUX_SEL_IN bits in the TSHK_INTF_CONFIG (analog SBI) register. This takes one HKADC clock cycle to accomplish before a conversion is initiated.

A conversion is initiated by writing to the TSHK_COMMAND_WR register. The analog signal level to be converted is acquired during the next three HKADC clock cycles. This time is required for acquisition. There is also a sampling time requirement, which is the time required to charge the internal sampling capacitor holding the DC level for the remainder of the conversion process. Eight clock cycles are required after signal acquisition to determine the 8-bit final result. There are four clock cycles of latency at the end of the conversion process to allow the result to propagate to the TSHK_DATA_RD [7:0] register. When TSHK_STATUS [0] sets (1), the 8-bit result is available for reading from the TSHK_DATA_RD [7:0] register.

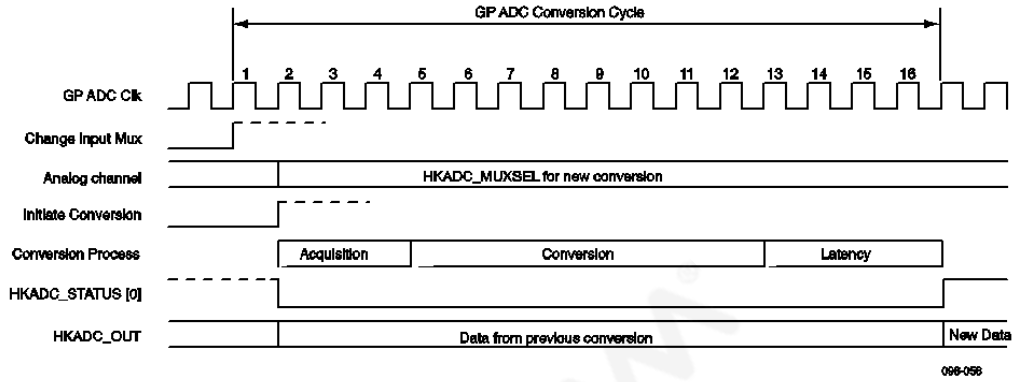


Figure 8-6 General HKADC conversion process

8.5.3 HKADC conversion time

The conversion time is defined by the following formula:

$$Ttotal = Tsampling + Tacquisition + N * T_{ADC\_clock}$$

Programmable values within this formula are:

- Tsampling is programmable using TSHK_DIG_CONFIG [7:6]. The default value is 01, which programs the sampling interval to 24 * (8/TCXO). Other suggested values set the multiple to 36 or 48.
- N is the ADC resolution, 8-bit resolution is the default value, only resolution supported by the MSM6290 device.
- The last programmable variable is T_{ADC_clock} — the sampling clock. This value is programmed using TSHK_PARAM [7:6], with 600 kHz, 1.2 MHz, 2.4 MHz, and 4.8 MHz options. The default value is 2.4 MHz — the only frequency value for which the HKADC is intended to work.

Table 8-6 Example of HKADC

Programmed or calculated value	Result
TSHK_DIG_CONFIG[7:6]=10	Sampling interval=36*(8/TCXO)
TSHK_DIG_CONFIG[3:2]=00	N = resolution = 8 bits
TSHK_PARAM[7:6] = 00	T_{ADC_clock} = TCXO/8
T_{ADC_clock} = TCXO/8 = 19.2/8 MHz	2.4 MHz
$T_{sampling}$ = 36*(8/TCXO)	15 µsec
$T_{acquisition}$ = 1*T_{ADC_clock} = 1/ADC_clock	0.417 µsec
N*T_{ADC_clock}	3.333 µsec
T_{total} = $T_{sampling}$ + $T_{acquisition}$ + N*T_{ADC_clock}	18.747 µsec

8.5.4 HKADC analog interface considerations

For HKAIN[5:0] input pins that are not selected by the HKADC input multiplexer, the impedance is very high and the pins of the MSM6290 device present essentially no load to external circuits connected to these pins. Figure 8-7 illustrates the equivalent circuit of the HKADC input and the circuits that are designed to drive the HKAIN [5:0] pins. The input multiplexer selects an input pin by closing S1. When a conversion is initiated, S2 stays closed until the end of the acquisition interval (three HKADC Clk cycles). After acquisition, S2 opens and the DC voltage to be converted is held on the sampling capacitor, Cs.

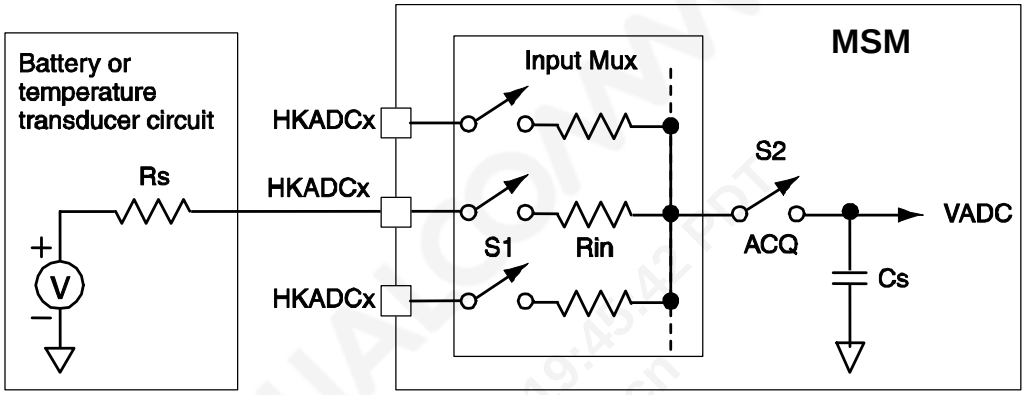


Figure 8-7 Equivalent circuit for HKADC input and external voltage sources

During the acquisition interval, the external circuit connected to the selected HKAIN pin must supply enough current to charge Cs through the 5 k (maximum) on-resistance of the input multiplexer. For accurate conversion results, the voltage on Cs must settle to within 0.25% of its final value (~1/2 LSB) in acquisition interval. Cs is approximately 12 pF.

The relationship that must be met is given by:

$$7 * (Rs + Rin) * Cs < (48 / CLK\_IN)$$

The right side does not change and has a fixed value. Therefore, for a CLK_IN of TXCO/4, we can rewrite the equation as follows:

$$7 * (Rs + Rin) * Cs < 10\ \mu s$$

This relationship is used to determine the maximum source resistance of the external circuits driving the HKAIN input pin.

Examples of this relationship are shown in Table 8-7.

Table 8-7 Recommended source resistor maximum values

f_{ADC_clock}	Rs max	CLK conditions	Vref conditions
600 kHz	50 k	TCXO = 19.2 MHz	VDD_A is selected for V_{REF}; VDD_A source resistance <50.
1.2 MHz	25 k	TCXO = 19.2 MHz	

9 Clock Regime

The MSM6290 devices derive all internal clock sources from three clock inputs, TCXO and SLEEP_XTAL. The TCXO clock input supports 19.2 MHz. An integrated PLL and M/N counter is used to create the required clock sources when the TCXO frequency is 19.2 MHz. The SLEEP_XTAL can support a 32.768 kHz clock source to drive the sleep controller during periods when most of the MSM6290 devices are powered down and the TCXO is disabled.

9.1 TCXO

The MSM6290 devices integrate a phase-locked loop and an M/N counter to derive CHIPX16 and CHIPX8 from the TCXO clock input.

9.2 Codec PLL

A separate fractional PLL will be available and can be used to generate the jitter-sensitive clocks for the wideband audio codec. The supersets of frequencies that need to be generated by the codec PLL are:

- 11.2896 MHz – Required for the noise shaper in the codec, is used to generate the 5.6448 MHz clock also required by the codec interface.
- 12.288 MHz – Required for the noise shaper in the codec, is used to generate the 8.192 MHz, 6.144 MHz, 4.096 MHz, 2.048 MHz, and 1.024 MHz clocks also required by the codec interface.

9.3 Sleep crystal circuit for 32.768 kHz

The MSM6290 devices contain a sleep oscillator circuit which can be used as a clock source when other clocks are disabled to conserve power, however it is highly recommended to use the PM6658 or PM6653 to drive SLEEP_XTAL_IN of the MSM instead of connecting a crystal directly to the MSM device.

Bits 11:10 of SLEEP_OSC_CTL control the gain of the internal inverter in the MSM6290 SLEEP oscillator circuit. Bits 11:10 are collectively called GAIN:SLEEP_OSC_CTL. Table 9-1 shows the gain settings of the inverter.

Table 9-1 Sleep oscillator inverter – relative gain settings

GAIN:SLEEP_OSC_CTL (bits 11:10 of SLEEP_OSC_CTL)	Relative gain
00	Minimum
01	
10	
11	Maximum

Bit 9 of SLEEP_OSC_CTL, called RF_BYPASS controls the feedback resistance between the input and output of the internal inverter. Set (1) SLEEP_OSC_RF_BYPASS to disable the feedback path. Clear (0) SLEEP_OSC_RF_BYPASS to enable the feedback path.

Bit 8 of SLEEP_OSC_CTL, called RD_BYPASS controls the series resistance between the output of the internal inverter, and the SLEEP_XTAL_OUT pin. Set (1) this bit to disable series resistance between the inverter's output and the SLEEP_XTAL_OUT pin. Clear (0) this bit to enable series resistance between the inverter's output and the SLEEP_XTAL_OUT pin. This bit **must** be set to a logic zero when a passive crystal is used with the internal inverter.

Table 9-2 summarizes the suggested settings of bits 11:8 of SLEEP_OSC_CTL. Switches S1 and S2 refer to the switches in the figures shown below in the next section. SLEEP_OSC_CTL is a write-once register; the register values can only be changed on the first write after system reset. Subsequent writes to SLEEP_OSC_CTL have no effect.

Table 9-2 Suggested SLEEP_OSC_CTL settings of bits 11:8

Sleep crystal frequency and configuration	Switch and gain settings	SLEEP_OSC_CTL, bits 11:8
32.768 kHz, passive crystal	S1 closed, S2 open Gain minimum	0000
32.768 kHz, external, active oscillator circuit	S1 open, S2 open Gain maximum	1011

Figure 9-1 shows an example 32.768 kHz sleep passive crystal circuit configuration with load capacitors. The inverter's gain is at its lowest setting. S1 is closed, enabling the inverter feedback path. S2 is open, enabling the series resistance between the inverter's output and SLEEP_XTAL_OUT.

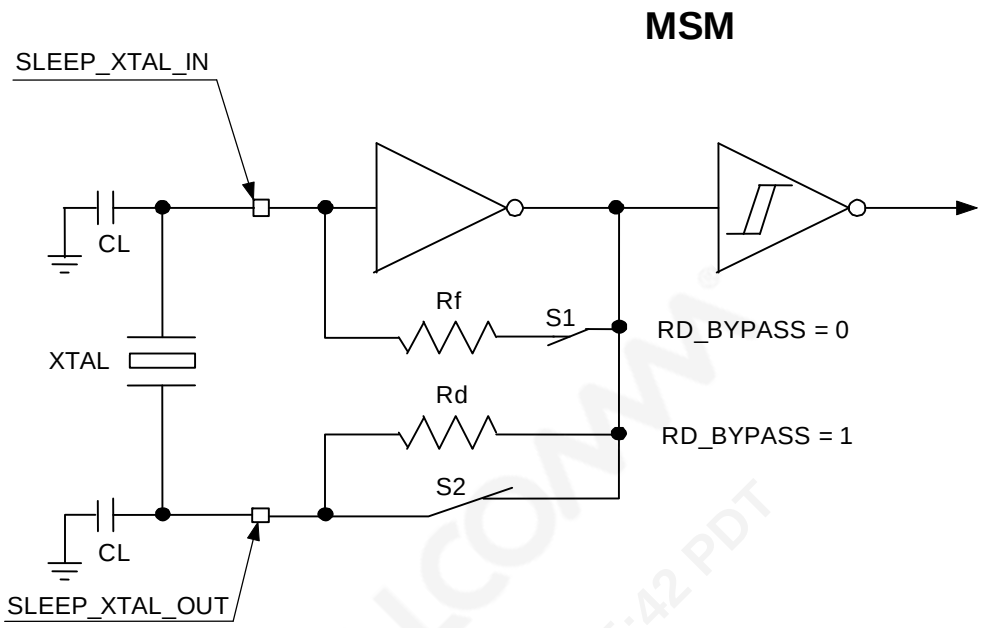


Figure 9-1 Sleep oscillator circuit with passive crystal

The sleep oscillator circuit also supports an external, active oscillator circuit.

To configure the sleep oscillator circuit for an external active oscillator:

1. Set the gain of the inverter at its maximum, producing fast transitions at the input of the output of the inverter.
2. Open S1, disabling the inverter's feedback path, thus reducing power consumption.
3. Open S2, enabling the series resistance between the output of the inverter and the SLEEP_XTAL_OUT pin.
4. Leave the SLEEP_XTAL_OUT pin unconnected.

If the active oscillator has an open-drain output, care must be taken in choosing the pull-up resistor. The pull-up resistor must be less than 20 k. If the output of the external oscillator circuit drives both high and low, the pull-up resistor is not required.

9.4 USB oscillator circuit for 48 MHz

AMSS 6290 generates a 48 MHz clock using the MSM6290 internal PLL. A USB oscillator circuit is no longer needed.

9.5 Subsystem clock regimes

9.5.1 Clock block architecture

The clock block in the MSM6290 devices is a single-phase clock generator that is responsible for providing all clocks required by internal cores and external interfaces. Via a frequency synthesizer constructed using two fractional phase-locked loops (GPLL) and two custom integer PLL(DCPLL), all required WCDMA, GPS, GSM, ARM, QDSP and miscellaneous peripheral clocks are generated from the two mandatory external oscillators, TCXO and sleep crystal oscillator.

The MSM device's PLLs are used to generate several different stable, low-jitter clock sources. Other clock sources include on-chip ring-oscillators and external clock sources.

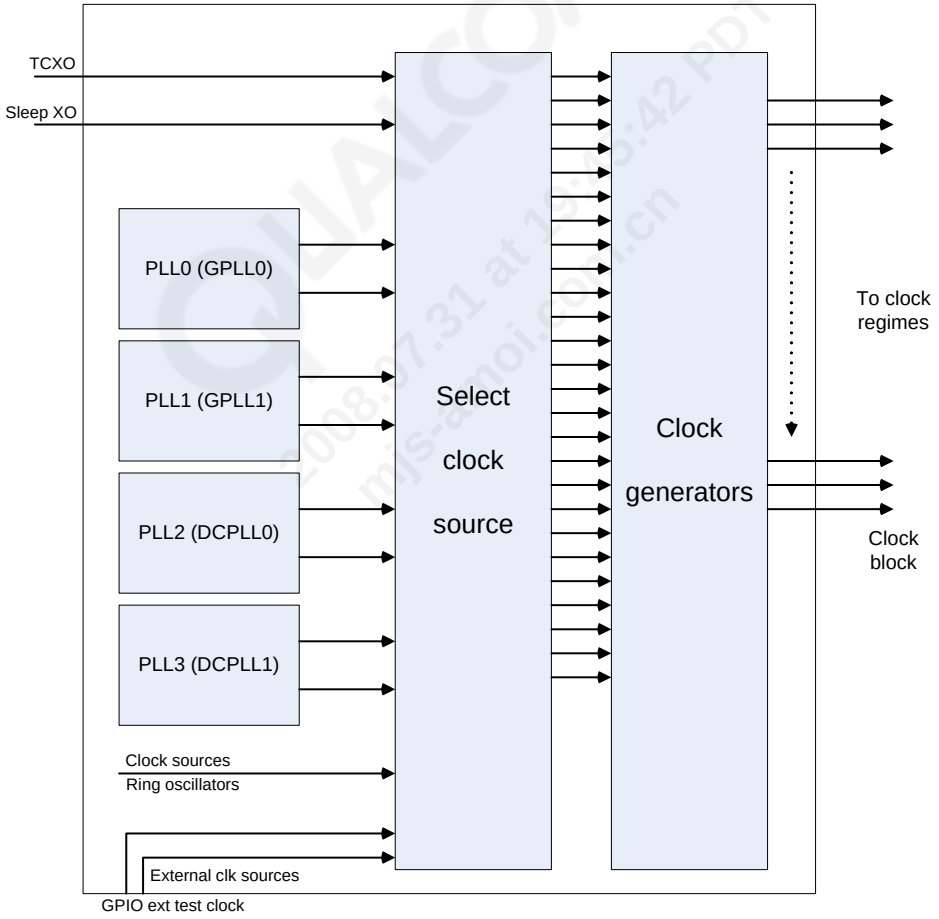


Figure 9-2 Clock block architecture

9.5.2 Clock regimes

The MSM6290 clock generator block has more than 50 clock regimes. The clock source for each regime is chosen by a programmable MUX structure under microprocessor control. The names of each of the clock regimes, along with other details about them, are shown in Table 9-3. Even though several of the clock regimes have similar sources, they are separated to provide independent sleep control and gating. Table 9-3 defines the clock origins.

Table 9-3 Descriptions of clock origins

Clock	Description
CHIPX16	Output of the M/N counter and TCXO PLL, path depends on TCXO frequency
CHIPX8	Output of the M/N counter and TCXO PLL, path depends on TCXO frequency
PLLOUT	Output of the TCXO PLL
GPS	Output of the TCXO PLL
TCXO/4	TCXO input divided by 4
TCXO	TCXO input
Sleep	Sleep oscillator output

10 JTAG Interface, Mode Select, and Emulation Considerations

10.1 JTAG interface

This section describes and explains the JTAG interface on the MSM6290 devices, the features it supports, its adherence to the IEEE standard 1149.1-1993, and its usage information.

10.1.1 JTAG standard overview

The JTAG interface conforms to the IEEE 1149.1A-1993 standard. That standard calls for a component with circuitry that allows test instructions and data to be fed into it and respective results to be read out of it in a serial format. It also calls for a test access port to be available on a design, and for boundary-scan architecture to be implemented so that the above requirements are fulfilled.

This testing circuitry will be used for board-level testing, and will have to accomplish the following:

- Confirm that each component on the board performs its function correctly
- Confirm all components are interconnected in the correct manner
- Confirm the entire design behaves as intended

As previously mentioned, this is done using a boundary-scan architecture technique. This technique involves the inclusion of a shift register stage (or cell) adjacent to each component pin so that signals at the component's boundaries can then be tested, controlled, and observed. Thus, the boundary-scan cells will be connected in a serial manner as a long chain, and will behave as an overall shift register. More information about the boundary-scan cells is provided in Section 10.1.5.3.1.

NOTE For more detailed information on this standard and its guidelines, please refer to IEEE Standard 1149.1A-1993. Also see *JTAG/ETM Interface for ARM9-based MSM Devices Application Note* (80-V7838-1).

10.1.2 MSM device JTAG interface

The JTAG interface on the MSM6290 devices aid in mobile station board-level testing and debugging.

Referring to the *IEEE 1149.1A-1993 manual: IEEE Standard Test Access Port and Boundary-Scan Architecture*, the compliance clause defines how compliance with this standard is "switched on" or "switched off."

In these MSM devices, the test access port (TAP) selection is done as part of the MODE selection for the device. For normal operation, and to select the ARM JTAG TAP, the MODE[1:0] pins are left unconnected (they are pulled low internally). To disable the watchdog, the WDOG_EN pin must be pulled low externally; the WDOG_EN pin is pulled high internally to enable the watchdog for normal operation. To select the MSM device scan chain TAP, the MODE [1:0] pins must be pulled high externally.

NOTE This implementation does not conform to the 1149.1-90, but does meet the intent of IEEE 1149.1A-93.

It is important to note that the TAP controller that is not selected is held in the TEST_LOGIC_RESET state. Because the JTAG port is shared, the TMS and TCK (see Section 10.1.3) pins are not seen by both controllers. TRST_N is pulled low internally on the MSM device.

As defined by the standard, the JTAG interface of these MSM devices allows test instructions and data to be shifted into the MSM device, and the test results to be read out in a serial format. The JTAG interface on the MSM devices consists of four main functional elements: a TAP, a TAP controller, an instruction register, and three test data registers. Figure 10-1 and Figure 10-2 provide illustrations of the MSM device JTAG interface and its block diagram.

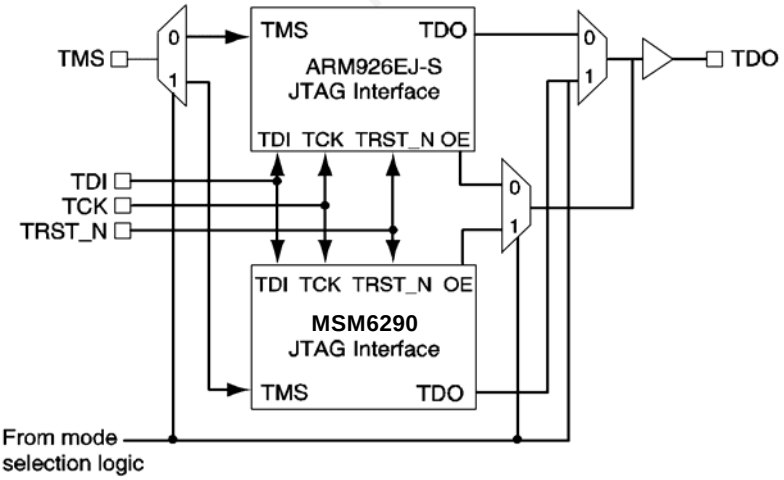


Figure 10-1 MSM device JTAG interface

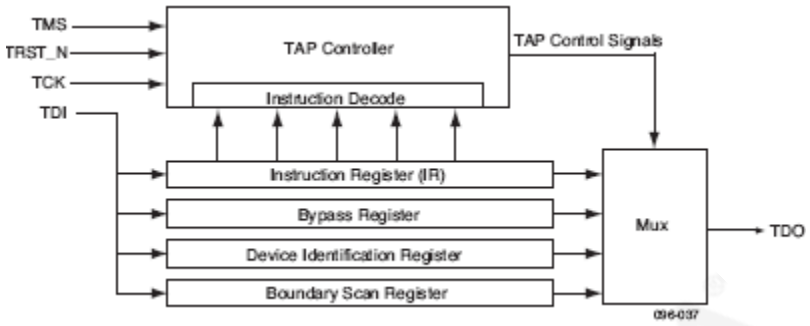


Figure 10-2 JTAG interface block diagram

10.1.3 Test access port

The JTAG interface is accessed through the TAP. The TAP includes the following input/output (I/O) pins:

- **TCK** – Test clock input
- **TMS** – Test mode select input
- **TDI** – Test data input
- **TDO** – Test data output
- **TRST_N** – Test reset input

The TAP dedicates its I/O pins to the JTAG interface. The nature of these pins and any internal pulls they may have is described in Section 10.1.3.1 through Section 10.1.3.5; these pins have no other use on the MSM device.

10.1.3.1 TCK

TCK is a user-defined clock input for the JTAG interface. It is independent of the system clock, and it supplies a clock input to the serial test path between TDI and TDO. TCK also permits shifting of test data concurrent with MSM device operation. The fact that it is independent ensures that the shifting of test data in and out of the TAP does not change the state of the MSM device. If the TCK signal is stopped at zero, stored-state devices contained in the test-logic retain their state indefinitely.

10.1.3.2 TMS

The test mode input is sampled on the rising edge of TCK, and is decoded by the TAP controller to set up all operations. The TMS input is pulled-high internally, so that it if it is unconnected, it will have the same effect as if a logic-high was applied, thus ensuring that the MSM devices continue operating unhindered by the test logic. Therefore, to enter test modes, a logic zero must be applied externally.

10.1.3.3 TDI

TDI is the TAP input for test instruction or test input (which will be in serial format). It is sampled on the rising edge of TCK. Data is pushed into the TAP through TDI and propagates to TDO without being inverted. Furthermore, it will be pulled high internally, so that if left unconnected, the TDI input will appear as a logic high. This ensures that open-circuit faults in the board-level and the serial data path force a defined logic state into the TAP.

10.1.3.4 TDO

TDO outputs the TAP serial data. Changes in the state of the signal driven through TDO occur only on the falling edge of TCK. This is done to ensure race-free operation (since changes on TMS and TDI occur only on the rising edge of TCK). The TDO driver will be inactive at all times (in this case, it will be tri-stated) unless data scanning is in progress.

10.1.3.5 TRST_N

When this signal is driven to a logic low, it will asynchronously reset the TAP controller and drive it to the test-logic-reset state, and asynchronously initialize all other test logic (as is required by the test-logic-reset state). If it is unconnected, it will be pulled down internally to make sure it does not interfere with normal device operation.

10.1.4 TAP controller

The TAP controller is a synchronous finite state machine that responds to changes in the TMS and TCK signals. It controls a sequence of operations. Its state machine has 16 states, allowing both data and instructions to be shifted. There are states for capturing, shifting, updating data and instructions, a state for running tests, and a reset state. For a detailed description of the different states and a state diagram, see IEEE Standard 1149.1a-1993.

The actions of test logic (both instruction and data registers) will occur on either the rising or falling edge of TCK in each respective controller state. More information on this is provided in IEEE Standard 1149.1a-1993.

The TAP controller initializes when it is in the test-logic-reset state. There are two ways to get it to that state:

- Applying a logic low to TRST_N brings the TAP controller asynchronously into the test-logic-reset state.
- Holding the TMS high while allowing at least five rising edges of TCK to occur brings the TAP controller synchronously into the test-logic-reset state (worst-case time to reach it from any other state).

After the test-logic-reset state is reached, all test logic disables and normal MSM device operations resume.

For board-level designs where JTAG testing is not used, TRST_N must be a logic-low. Before using the MSM devices in a mobile station application, TRST_N must be pulled low at powerup, or the TAP controller must be in the test-logic-reset state.

10.1.5 Data registers

Three registers in the MSM device JTAG interface are data registers: device identification register, bypass register, and boundary-scan register.

10.1.5.1 Device identification register

The device identification register provides a way for the manufacturer, part number, and version of a component to be determined through the TAP. One application of this is to distinguish the manufacturers of components on an assembled board when more than one sourcing is used.

The device identification register used on the JTAG interface of the MSM device is a 32-bit register holding the device's manufacturer identity code (bits[11:1] – administered by JEDEC; Qualcomm's is 0x070), part number (bits[27:12] – Qualcomm-generated number), and version data (bits[31:28] — a Qualcomm-generated number — 0x1 for the production version). The least significant bit (bit 0) is the device identification register's start bit, and is set to a logic 1.

Table 10-1 MSM6290 Device identification register

Bits	Name	Description
31:0	PART_ID	0x001C00E1: MSM6290 configuration – Engineering sample 1
31:0	PART_ID	0x101C00E1: MSM6290 configuration – Engineering sample 2, 3, and commercial sample

Selecting the device identification register loads its value into the shift register on the rising edge of TCK in the capture-DR state. It has a parallel input, but it does not have a parallel output. Figure 10-3 shows the structure of the device ID register.

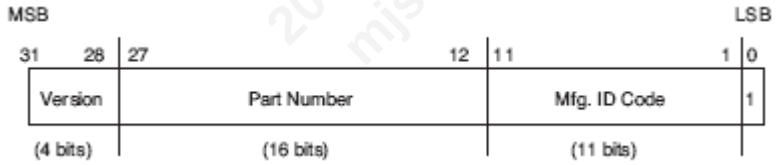


Figure 10-3 Data structure for device identification register

10.1.5.2 Bypass register

The bypass register is a single-stage shift register that is located between the TDI and the TDO. It is used to route the scan data directly from TDI to TDO without going through the entire boundary-scan register. Bypassing the boundary-scan register allows quicker movement of data to and from other board components. The operation of the bypass register will have no effect on the operation of the on-chip system logic. It does not have a parallel output.

Selecting the bypass register loads the shift register with a logic low on the rising edge of TCK, following entry into the capture-DR state.

10.1.5.3 Boundary-scan register

The boundary-scan register allows board interconnection testing to detect typical defects, like opens and shorts. The boundary-scan register connects, via boundary-scan cells (see Section 10.1.5.3.1), between each digital I/O pin and the MSM devices internal circuits. Therefore, it gives the tester the capability to access and/or sample system I/O signals when testing system logic. The boundary-scan register also handles parallel input and outputs.

10.1.5.3.1 Boundary-scan cells

The boundary-scan register is comprised of boundary-scan cells. These cells access digital signals at the MSM device. Each cell has a single-shift register stage and single serial input and output terminals, and is connected to the one before it and after it in the boundary-scan register. In addition, each cell has a parallel input/output (which is how the entire register handles parallel input/output) terminal through which data is sampled and/or latched. Figure 10-4 shows the basic structure of the boundary-scan cell.

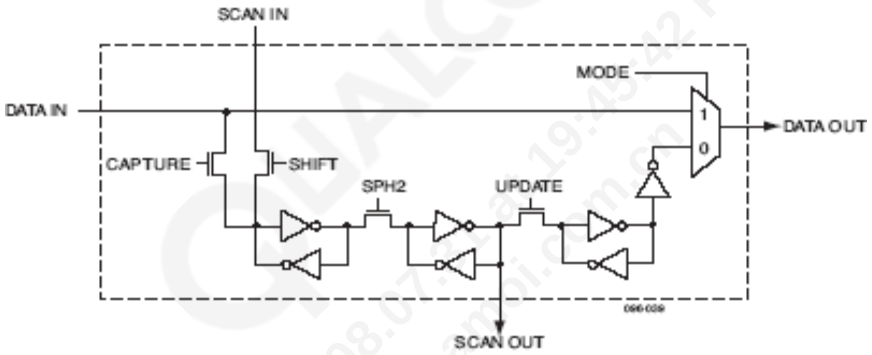


Figure 10-4 Typical structure of a boundary-scan cell

10.1.6 Instruction register

The instruction register (IR) is a 7-bit shift register having a serial input and parallel latched output. It holds JTAG instructions that were shifted into the JTAG interface and allows instructions to be serially loaded into the test logic during an IR scan cycle. A JTAG instruction shifts through TDI, LSB first until it is in the IR. In the IR, the instruction is decoded and the test to be performed and the data register to be used are selected. The parallel output holds the current instruction, and is updated on the falling edge of TCK (when in the update-IR state), or asynchronously upon entry into the test-logic-reset state. The basic functions used and supported on the MSM devices JTAG interface are BYPASS, SAMPLE/PRELOAD, EXTEST, and IDCODE.

NOTE For more information on JTAG Instruction registers, see *JTAG/ETM Interface for ARM9-based MSM Devices Application Note* (80-V7838-1).

10.1.7 JTAG selection

Because both the embedded ARM microprocessor and the MSM devices have JTAG capability, the user has the ability to select which JTAG interface they wish to use. This is done using the MODE[1:0] pins. As previously explained, for ARM JTAG, they should be left unconnected since they are pulled-low internally, and for MSM JTAG, pulled high externally. It is important to remember that WDOG_EN can expire during JTAG operation, and therefore should be accounted for either by grounding its pins externally (thereby disabling it), or by software. Figure 10-5 and Figure 10-6 show the example pin configurations for setting those MODE pins.

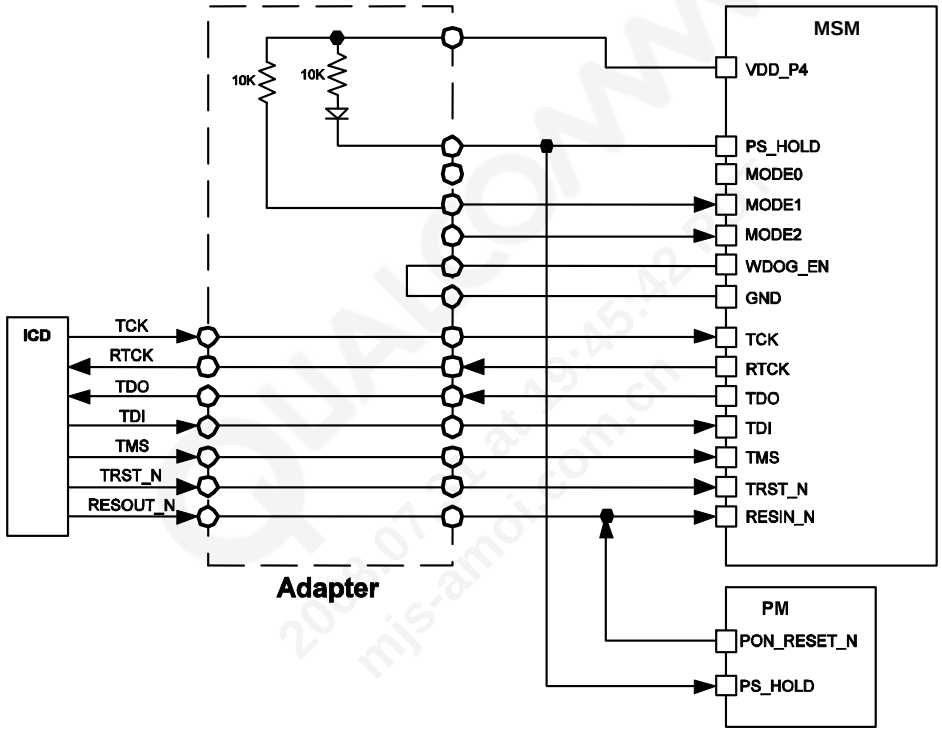


Figure 10-5 JTAG and MSM pin connections for selecting MSM BSDL scan chain, WDOG disabled

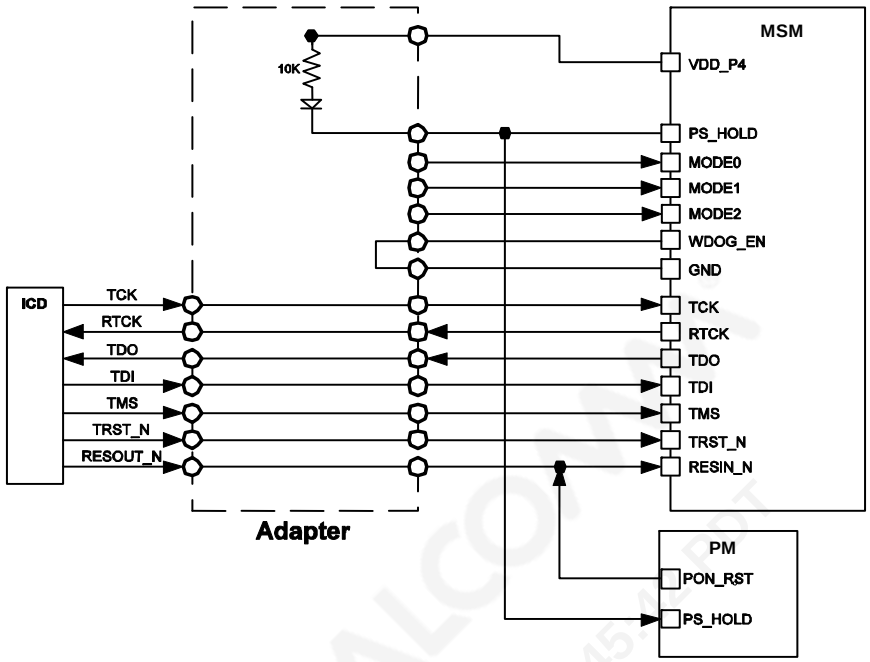


Figure 10-6 JTAG/MSM pin connections for selecting ARM JTAG, WDOG disabled

10.2 Embedded trace macrocell

This section describes the operation of the embedded trace macrocell (ETM) in the ARM family of processors, and how it is implemented and used on MSM devices. It is important to note that to achieve tracing at target speed for the MSM device (225 MHz), the deMUX ETM mode should be used.

10.3 ETM overview

The ETM is a block that provides instruction and data trace for the ARM family of microprocessors. The use of ETM in a system allows real-time debugging. The respective MSM devices use ETM9, which is the block that is used by the ARM9 microprocessors.

The ETM consists of two parts:

- **Trace port** – The trace port broadcasts trace information (either instruction trace or data trace)
- **Triggering facilities** – This controls the ETM to filter and control trace operations

ETM is designed to be connected directly to the ARM core it is tracing. The trace port connects to a software debugger, which configures the ETM and processes the information the ETM provides.

NOTE For more information regarding the ETM, see the ARM ETM9 manuals and documentation at www.arm.com.

An example debugging environment is shown in Figure 10-7.

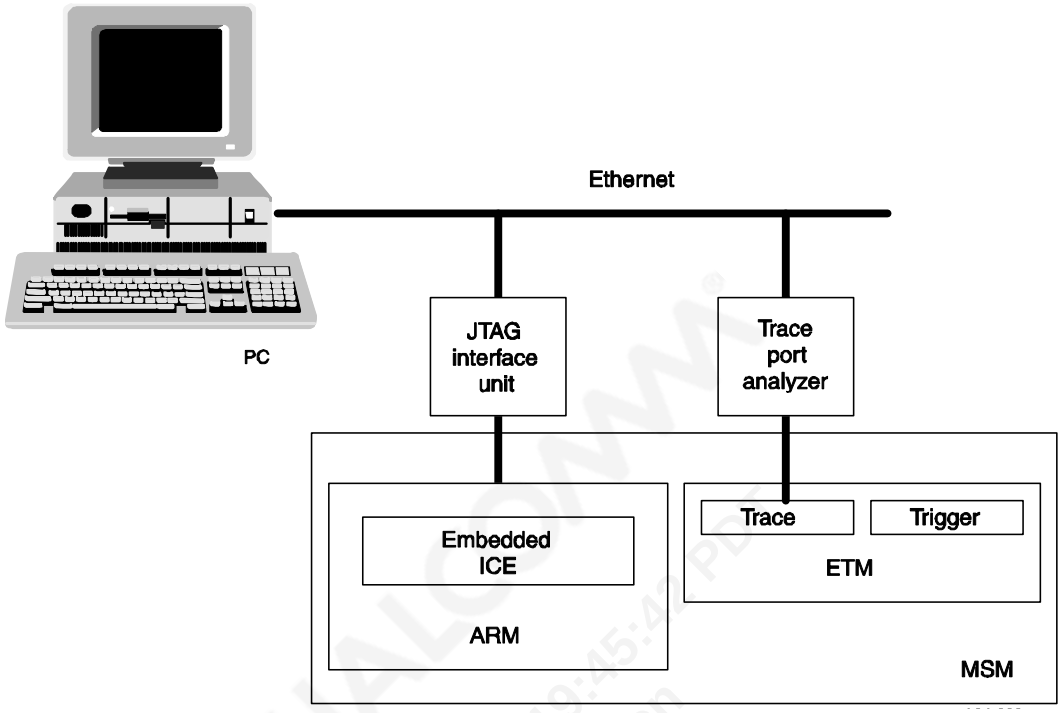


Figure 10-7 Example debugging environment using ETM

NOTE For more details, see *JTAG/ETM Interface for ARM9-based MSM Devices Application Note* (80-V7838-1).

10.4 ETM architecture

The architecture of the ETM is shown in Figure 10-8.

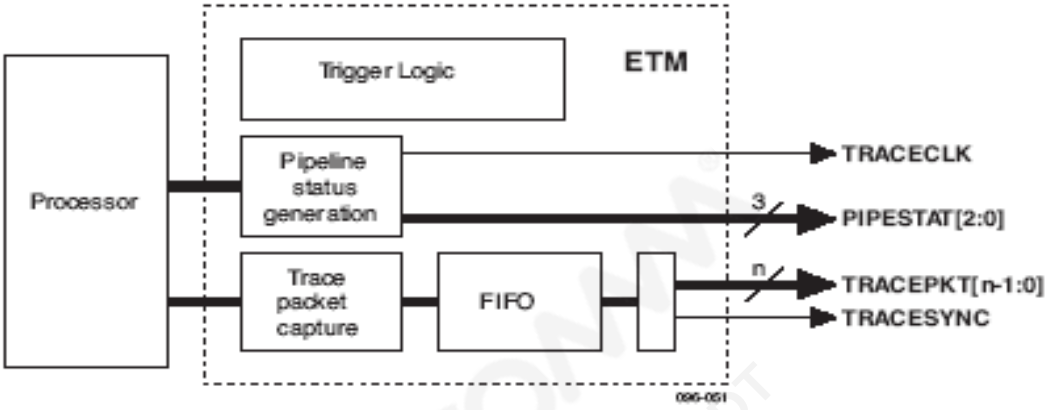


Figure 10-8 Basic ETM architecture

The trace packet can be 8, or 16 bits, with the bigger packets allowing higher trace data bandwidth.

The trace port is composed of four signals:

- *PIPESTAT (2:0)* – Three pipeline status pins
- *TRACEPKT* – An n-pin trace packet port, where n can be 4, 8, or 16 pins
- *TRACESYNC* – A trace packet synchronization pin
- *TRACECLK* – A clock signal with the same frequency as the processor clock

The GPIO pins used for each of the above signals are described in Section 10.6.

In addition to the ETM modes supported in previous MSM devices that use ETM7, the MSM6290 device, which uses ETM9, also implements a wide (de-multiplexed) trace port. This is achieved by clocking the trace port at half the processor operating frequency and routing trace port outputs to pairs of output pins.

10.5 ETM features

The ETM configuration on the MSM devices supports several different options. However, since the target speed of the MSM device is approximately 297.6 MHz, full emulation at this speed is not realizable due to limitations introduced by the pads' maximum toggling frequency. Therefore, 8-bit deMUX mode would be the best choice for target speed emulation needs. The configuration options supported by the MSM6290 devices are:

- 16-bit normal mode
- 8-bit normal mode
- 8-bit deMUX mode (recommended)
- Half-rate clocking mode (as well as the full clocking mode)

10.5.1 16-bit normal mode

When this mode is used, the ARM processor should be run at a low speed (the maximum toggling speed allowed by the pads). In 16-bit normal mode, only one trace port analyzer (TPA) is required. The ETM pins are toggled at the clock rate of the ARM.

10.5.2 8-bit normal mode

Camera interface input data (8 bits) share pins with some ETM trace packet pins. Therefore, the ETM can be configured to run in 8-bit mode. It uses eight fewer pins, which are then used as camera interface input data. In all other aspects, this is identical to the 16-bit normal mode.

10.5.3 8-bit deMUX mode (recommended ETM mode)

When operating the microprocessor at the target speed, it is difficult to toggle ETM pins at high-speed. In order to achieve tracing at the target speed (~297.6 MHz), the ETM needs to be configured in 8-bit deMUX mode. Therefore, this is the mode that should be selected when emulation at target speed is desired. In this mode, the trace port analyzer captures the data at half the ARM's clock rate. The deMUX mode configuration is shown in Figure 10-9.

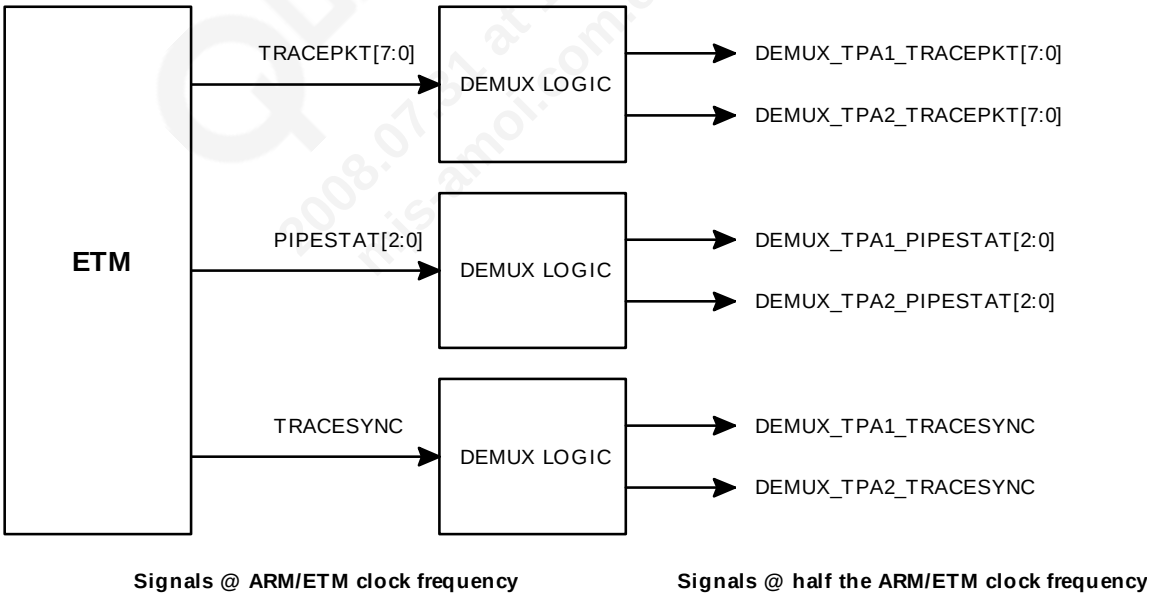


Figure 10-9 ETM 8-bit deMUX mode

The cost of this mode is an additional trace packet analyzer (TPA), four more pins, and less trace (not full). A total of 25 pins are required for ETM in this mode.

10.5.4 Half-rate clocking

The ETM also supports half-rate clocking mode. When in this mode, the TRACECLK to the TPA is one-half the rate of the ARM clock in normal mode, and one-quarter the rate of the ARM clock in deMUX mode. The primary purpose for the half-rate clocking mode is to reduce the signal transition rate on the TRACECLK pin. When this mode is selected, the TPA samples the trace data signals on both edges of the clock, instead of only the rising edge in full-rate clocking mode.

10.6 ETM implementation on the MSM devices

The MSM devices are capable of implementing the maximum trace port size (16 bits) and the maximum FIFO size.

For ETM debugging, the MSM devices are set into a dedicated debug mode called ETM mode. This mode is set through the MODE pins. To set ETM mode MODE[1:0] should be set to 0b´01. In ETM mode, the MSM devices operate in native mode, but with selected GPIO pins dedicated to the ETM trace port (see Table 10-2).

The trace data and clock in the different modes are implemented as follows: In deMUX mode, the trace data (TRACESYNC, PIPESTAT (2:0), TRACE_PKT (15:0)) are deMUXed to the X and Y path. A programmable delay cell is implemented in the clock path, so that the clock path delay is adjustable. This ensures timing for the TPA. In normal mode, the X and Y paths are the same.

The ETM interface uses the GPIO pins on their respective MSM devices. A total of 25 pins (hidden behind the GPIO pins) are needed to support the different ETM modes. The ETM configuration register's PORTMODE (1:0) is used to select normal or de-multiplexed configuration. The PORTSIZE[2:0] bits are used to select between 8- or 16-bit modes. For more information about the configuration of the ETM, please refer to the appropriate ARM documentation. Table 10-2 lists and describes the relevant GPIO pins and the different signals for each ETM mode for the MSM device.

Table 10-2 GPIO[66:42] signals in ETM mode

GPIO	8-bit DeMUX	8-bit normal	16-bit normal
GPIO[66] (ETM_TRACECLK)	ETM_TRACECLK	ETM_TRACECLK	ETM_TRACECLK
GPIO[65] (ETM_TRACESYNC)	ETM_X_TRACESYNC	ETM_X_TRACESYNC	ETM_X_TRACESYNC
GPIO[64] (ETM_PIPESTAT2)	ETM_X_PIPESTAT2	ETM_X_PIPESTAT2	ETM_X_PIPESTAT2
GPIO[63] (ETM_PIPESTAT1)	ETM_X_PIPESTAT1	ETM_X_PIPESTAT1	ETM_X_PIPESTAT1
GPIO[62] (ETM_PIPESTAT0)	ETM_X_PIPESTAT0	ETM_X_PIPESTAT0	ETM_X_PIPESTAT0
GPIO[61:54] (ETM_TRACE_PKT [15:8])	ETM_Y_TRACE_PKT[7:0]	Not used Can be used for CAMIF_DATA[9:2]	ETM_X_TRACE_PKT[15:8]
GPIO[53:46] (ETM_TRACE_PKT [7:0])	ETM_X_TRACE_PKT[7:0]	ETM_X_TRACE_PKT[7:0]	ETM_X_TRACE_PKT[7:0]
GPIO[45] (ETM_TRACESYNC_B)	ETM_Y_TRACESYNC	N/A	N/A
GPIO[44] (ETM_PIPESTAT0B)	ETM_Y_PIPESTAT0	N/A	N/A
GPIO[43] (ETM_PIPESTAT1B)	ETM_Y_PIPESTAT1	N/A	N/A
GPIO[42] (ETM_PIPESTAT2B)	ETM_Y_PIPESTAT2	N/A	N/A

Note: GPIO[41](GPIO2_GROUP_INT), GPIO[40](PM_INT_N), and GPIO[39](ETM_KEYSENSE_IRQ) are for emulation on the SURF boards in ETM mode.

11 Transport Stream Interface for DMB

The MSM6290 interfaces with the Qualcomm UBM that allows handsets and other portable devices to receive audio and/or video (TV) broadcast programming. In general, DMB systems use one of two options to deliver the streaming signals: 1) a satellite to TV tower link followed by terrestrial broadcast to handsets, or 2) a direct link from a satellite to handsets.

NOTE UBM uses different interfaces based upon the standard (MediaFLO, ISDB-T or DVB-H) used. Refer to the *MBP1600 Device Specification* (80-VD180-1) for further details.

As DMB systems are being developed, it becomes clear that *different digital mobile broadcast standards are being adopted around the world*. Table 11-1 lists some of the currently established DMB standards.

Table 11-1 Digital mobile broadcast standards

Standard	Region(s)	Technology and mode	Bandwidth	Band
ISDB-T Integrated Service Digital Broadcasting - Terrestrial transmission (ARIB STD-B31)	Japan	OFDM®	6 MHz or 6/13 MHz	UHF
S-DMB Satellite Digital Mobile Broadcast (ARIB STB-B41)	Korea	4XWCDMA	25 MHz	S-Band (2.6 GHz)
DVB-H Digital Video Broadcast-Handheld	US Europe	COFDM	5, 6, 7, 8 MHz	UHF/VHF

The handset's DMB implementation (Figure 11-1) uses a separate, non-CDMA radio link that has its own antenna. Regardless of which DMB standard is used, an ITU-T *H.222.0 Transport Stream* (HTS) should present to the MSM6290 TSIF interface. The MSM6290 IC then decodes the HTS for audio and/or visual presentation.

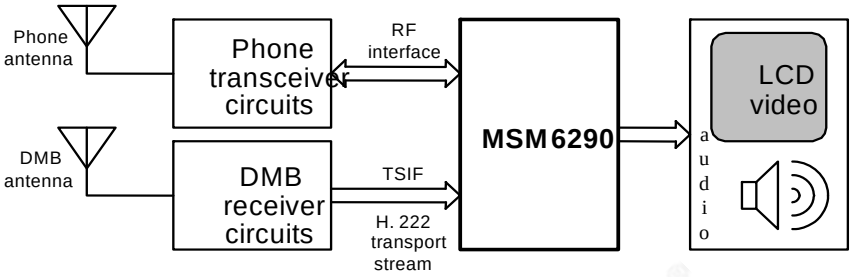


Figure 11-1 MSM6290 DMB handset architecture

HTS makes use of fixed-length 188 byte packets that are specified in H.222.0 (ISO/IEC 13818-1). While this standard specifies HTS content, it does not specify an appropriate component-to-component signaling mechanism.

11.1 TSIF features

The MSM6290 TSIF features include:

- 3-pin or 6-pin serial interface
- Supports external clocks up to 4 MHz at a maximum data rate of 0.5 Mbytes per second
- The data mover (DM) transfers HTS packets directly from the external interface to system memory
- Enhanced fallback and/or debug support using a software-based copy mechanism when HTS packets are transferred directly from the external interface to system memory
- Reports the status of each HTS packet transferred to memory via the DM or software copy
- Optional interrupts for critical events: loss-of-sync, packet available, and packet overflow.
- Provides a 27 MHz based TSIF clock reference (TCR) that functions as a system clock reference (SCR) as defined in H.222.0
- Provides a TSIF time stamp (TTS) based upon the TCR as part of the status for each packet

11.2 TSIF connections

TSIF functions can be implemented by the MSM6290 device using many of the same GPIOs used for UART, USB, and/or UIM interfaces (Table 11-2). The clock, enable, and data lines are required and make up the three-line TSIF implementation. An additional three lines (sync, error, and null) can be added to form the optional six-line TSIF implementation.

Table 11-2 TSIF connections

Signal name	Pin #	I/O	Voltage	Description	Comments
TSIF_CLK GPIO [87]	N16	I	Digital	TSIF clock for DMB	UART, USB, and/or UIM functions can also be supported by these same GPIOs.
TSIF_EN GPIO [86]	L17	I	Digital	TSIF enable for DMB	
TSIF_DATA GPIO[85]	N17	I	Digital	TSIF data for DMB	
TSIF_SYNC GPIO [84] or GPIO [80] or GPIO [29]	J20 or E23 or K20	I	Digital	Optional TSIF sync at packet beginning for DMB	
TSIF_ERROR GPIO[84]	J20	I	Digital	Optional TSIF packet error flag for DMB	
TSIF_NULL GPIO [84] or GPIO [17] or GPIO [14]	J20 or H22 or L23	I	Digital	Optional TSIF null flag for DMB	

11.3 TSIF hardware architecture

The MSM6290 TSIF hardware (Figure 11-2) includes software accessible registers: status and control, time-limit, and TSIF clock reference. The status and control register is used to configure and control the TSIF as well as provide status during operation. When the TSIF is started via this register, it effectively enables the state machines to simultaneously perform continuously overlapping external hardware interface (EHI) input, and output operations as conceptually detailed in Section 11.4.

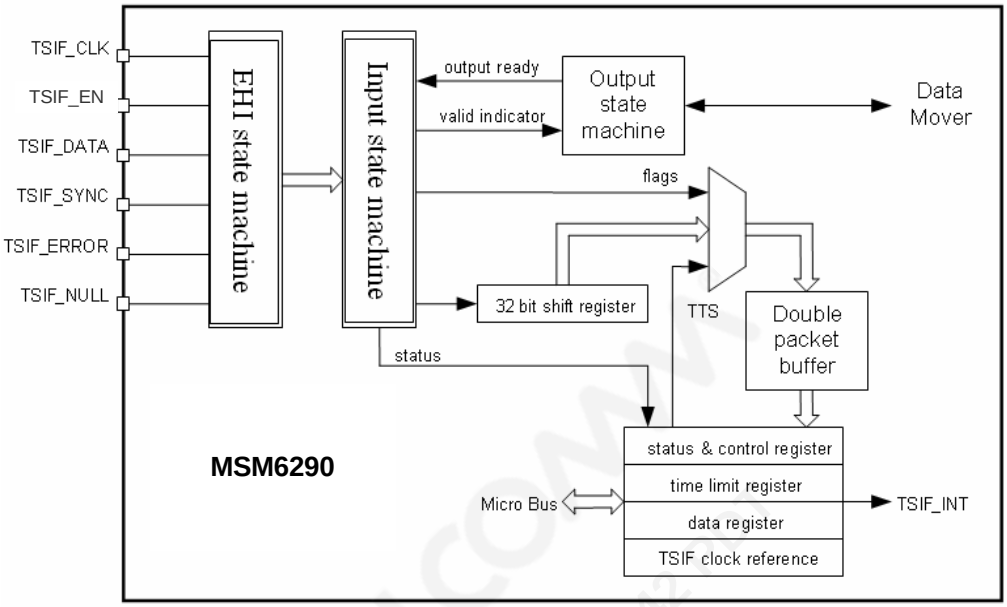


Figure 11-2 TSIF hardware architecture

As serial data enters via TSIF_DATA, it is shifted into a 32-bit word register. Each completed word is stored in the double packet buffer until a 188-byte packet has been stored. Then the flag values (first-packet-flag, overflow-flag, post-tsif_en-flag, error_flag, null_flag) and TTS are stored in the buffer and the output operation is alerted with a valid-indicator that a packet and status are stored in the buffer. The input operation then flips the buffers and begins to enter the next packet.

At valid-indicator, the output operation makes a request to the data mover (DM) to move a packet to system memory. There is an assumption here that the DM has been previously programmed to move a 192-byte packet when a request is received. The DM then reads the packet via the micro interface with 32-bit reads from the double packet buffer. The read is performed in a single 192-byte block. The output operation then raises output-ready to signal that it has successfully written the last packet to memory and is ready for the next one. Each data packet appears in memory as illustrated in Figure 11-3.

As described above, the first 188 bytes are data; the last four bytes are time stamps (3 bytes) and flags (1 byte). Byte 191 contains the following flags in the following bit positions:

- Bit 0, *valid flag*: always set (1) to indicate valid HTS packet and status. By pre-clearing this byte to 0, software can always determine when a valid packet is in memory.
- Bit 1, *first packet flag*: high indicates the first packet of a new stream or the first valid packet after one or more packets were lost.
- Bit 2, *overflow flag*: high indicates that the input operation had a packet to provide to the output but the output operation was not done with its last packet – one or more packets were lost. The current packet is valid and is associated with the first packet that could be passed to the output.
- Bit 3, *error flag*: indicates the state of the TSIF_ERROR input when this packet was input.
- Bit 4, *null flag*: indicates the state of the TSIF_NULL input when this packet was input.

- Bit 5, *post TSIF_EN flag*: indicates the state of TSIF_EN the next TSIF_CLK after the last data bit of the packet.

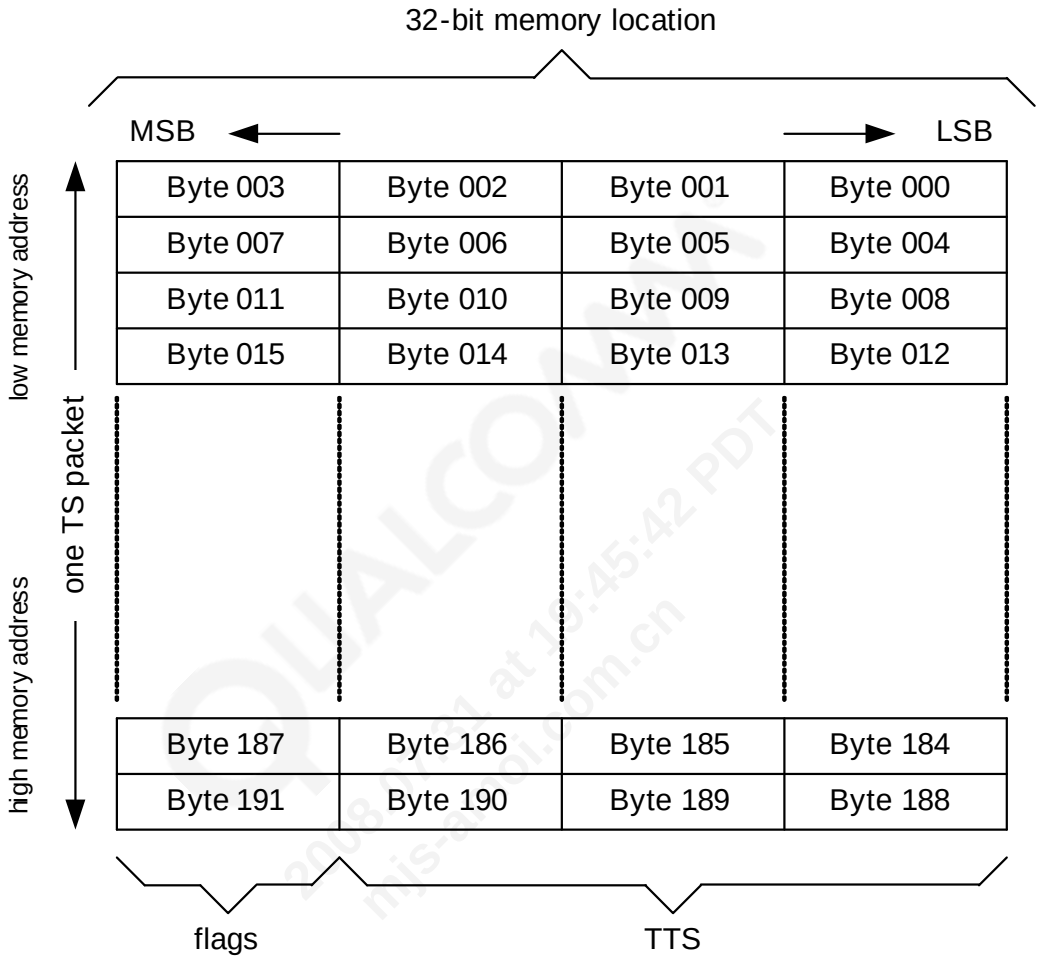


Figure 11-3 TSIF system memory packet

11.4 TSIF external hardware interface modes

There are two modes available for interfacing the MSM6290 transport stream functions to external hardware; both are described in this section, followed by a discussion of the sampling of optional TSIF signals.

11.4.1 TSIF mode 1

Mode 1 uses just three TSIF signals (Figure 11-4). The TSIF_CLK frequency depends upon the digital mobile broadcast system being supported. When TSIF_EN is asserted, as indicated by a low-to-high transition, it must remain asserted for at least 1504 TSIF_CLK cycles. For most DMB applications, TSIF_EN de-asserts after 1504 ticks (even though it's not a mode 1 requirement). The data captured during these 1504 clocks is called a "valid packet." The low-to-high transition of TSIF_EN serves as a new-packet-indicator.

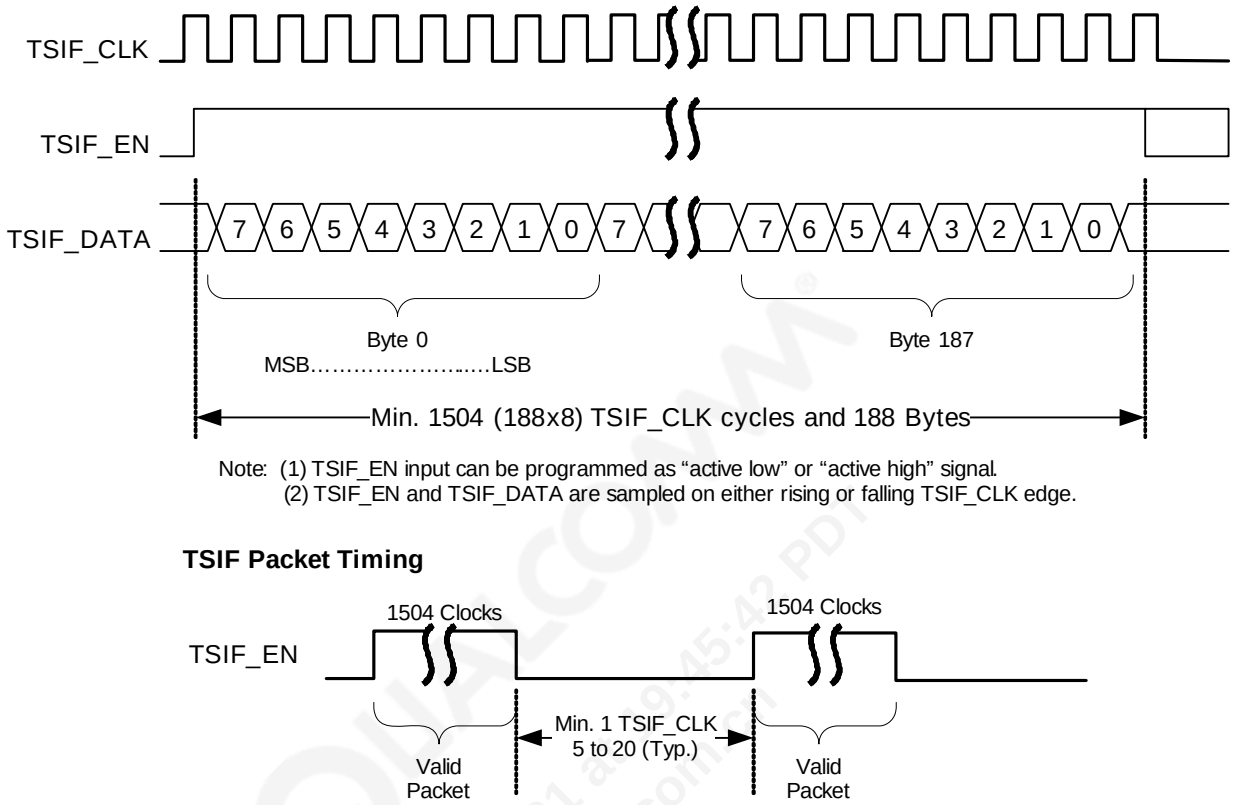


Figure 11-4 TSIF mode 1 signals and timing

After the TSIF external hardware interface (EHI) is enabled, the receipt of the first valid packet is associated with a first-packet-flag. This flag notifies the processing resource that the associated packet is the first of a series. Should the EHI enter an error condition of some kind that results in the loss of one or more packets, this flag is set again to indicate the start of a new series.

One possible error condition is for TSIF_EN to assert and then de-assert in less than 1504 clocks. Under this case, all data captured under this "short" TSIF_EN is discarded and a loss-of-sync error is recorded. Additionally, an optional time-limit may be employed that is started each time a valid-packet is captured. If the time-limit expires before a new valid-packet is received, then a lost-sync error is recorded.

Bit order within the byte is always MSB (most significant bit) to LSB (least significant bit) as illustrated.

Under mode 1, the time between TSIF_EN assertion (or valid packet) is always at least one clock cycle but is generally on the order of 5 to 20 clock cycles depending upon the DMB system being supported. However, it can conceptually be much longer. The only firm requirement is that a low-to-high transition of TSIF_EN must be followed by 1504 ticks of TSIF_EN remaining high. Furthermore, one additional tick is needed after tsif_en is deasserted. Thus, a total of 1505 ticks are needed. As an option (discussed above), a time-limit between valid packets can be used to indirectly limit the length of time between TSIF_EN low-to-high transitions.

11.4.2 TSIF mode 2

For mode 2 operation (Figure 11-5), the signal TSIF_SYNC provides the new-packet indicator function and TSIF_EN is a free format signal that may assert or de-assert on any clock period. Only the low-to-high transition of TSIF_SYNC is needed to generate the new-packet-indicator. The length of TSIF_SYNC assertion time is a "don't care." The signal TSIF_EN may be high or low at new-packet indicator, and could even be tied high and a valid-packet (188 bytes, 1504 bits) would still be captured at each new-packet indicator. The only TSIF_EN requirement is that it must stay high for at least 1504 clocks between assertions of TSIF_SYNC. If not, a lost-sync error occurs. The time-limit and first-packet-flag concepts presented in the previous section still apply. Furthermore, one additional tick is needed after tsif_en is deasserted. Thus, a total of 1505 ticks are needed.

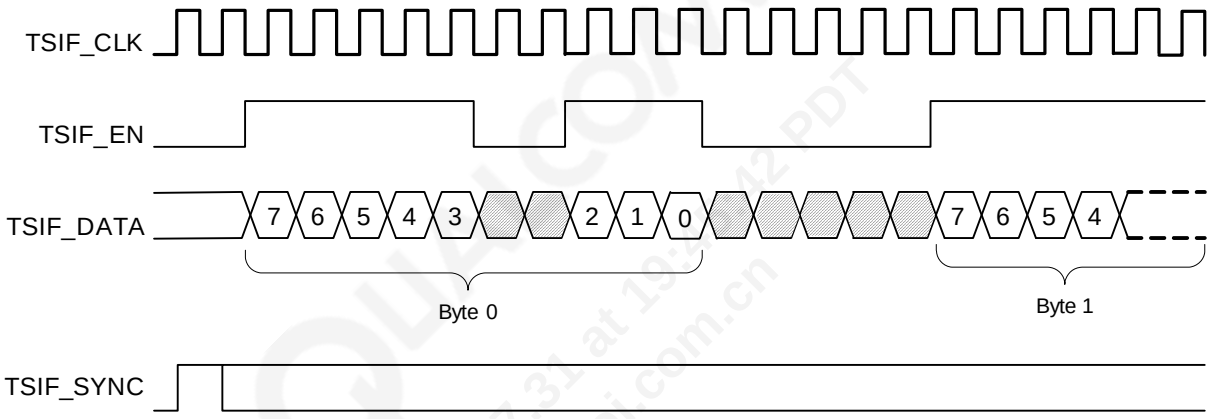


Figure 11-5 TSIF mode 2 signals and timing

11.4.3 Sampling optional TSIF signals

An H.222.0 transport stream provides an explicit mechanism for reporting a packet error or a null packet. However, source demodulators output discrete side-band signals for these functions as illustrated in Figure 11-6. The null (tsif_null) signal indicates a null or loaded packet. These signals do not affect the operational characteristics of the modes already presented. They are just sampled along with tsif_data at one of the locations indicated, that is, the first or last bit of the packet.

As the values on these signals are just sampled and reported, they do not need to be used for the functions implied by their names. Any per packet information may be sampled and recorded. For example, the inputs could be used to identify individual channels in a multichannel stream.

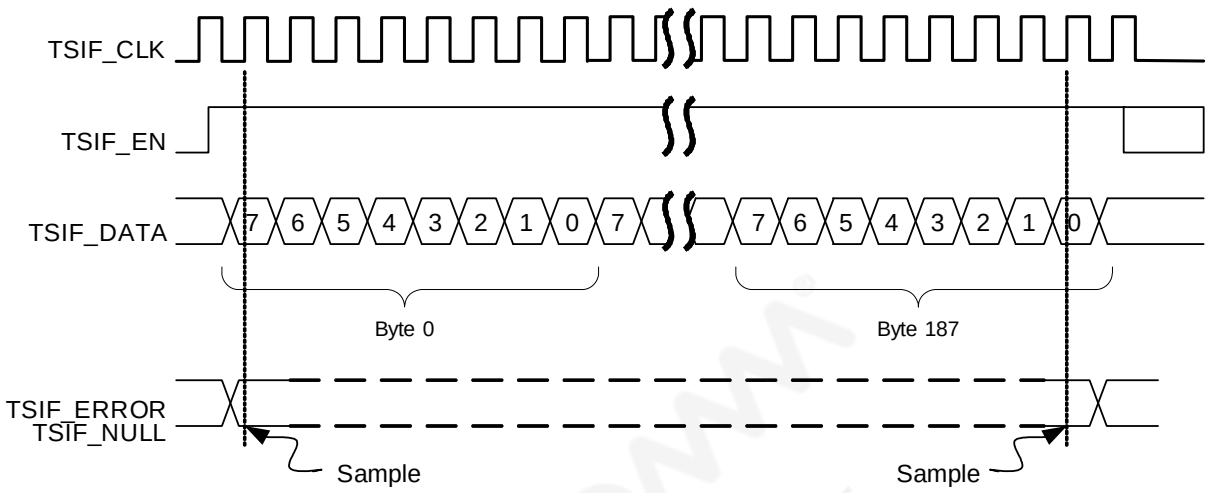


Figure 11-6 Sampling optional TSIF signals