TEXAS INSTRUMENTS

TAS1020

USB Streaming Controller

Data Manual

February 2001

DAV Digital Audio - Digital Speakers

SLAS263A

IMPORTANT NOTICE

Texas Instruments and its subsidiaries (TI) reserve the right to make changes to their products or to discontinue any product or service without notice, and advise customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgment, including those pertaining to warranty, patent infringement, and limitation of liability.

TI warrants performance of its products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Customers are responsible for their applications using TI components.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards must be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance or customer product design. TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such products or services might be or are used. TI's publication of information regarding any third party's products or services does not constitute TI's approval, license, warranty or endorsement thereof.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations and notices. Representation or reproduction of this information with alteration voids all warranties provided for an associated TI product or service, is an unfair and deceptive business practice, and TI is not responsible nor liable for any such use.

Resale of TI's products or services with *statements different from or beyond the parameters* stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service, is an unfair and deceptive business practice, and TI is not responsible nor liable for any such use.

Also see: Standard Terms and Conditions of Sale for Semiconductor Products. www.ti.com/sc/docs/stdterms.htm

Mailing Address:

Texas Instruments
Post Office Box 655303
Dallas, Texas 75265

Contents

List of Illustrations

List of Tables

1 Introduction

The TAS1020 integrated circuit (IC) is a universal serial bus (USB) peripheral interface device designed specifically for applications that require isochronous data streaming. Applications include digital speakers, which require the streaming of digital audio data between the host PC and the speaker system via the USB connection. The TAS1020 device is fully compatible with the USB Specification Version 1.1 and the USB Audio Class Specification.

The TAS1020 uses a standard 8052 microcontroller unit (MCU) core with on-chip memory. The MCU memory includes 8K bytes of program memory ROM that contains a boot loader program. At initialization, the boot loader program downloads the application program code to a 6K RAM from either the host PC or a nonvolatile memory on the printed-circuit board (PCB). The MCU handles all USB control, interrupt and bulk end-point transactions. In addition, the MCU can handle USB isochronous end-point transactions.

The USB interface includes an integrated transceiver that supports 12 Mb/s (full speed) data transfers. In addition to the USB control end-point, support is provided for up to seven in end-points and seven out end-points. The USB end-points are fully configurable by the MCU application code using a set of end-point configuration blocks that reside in on-chip RAM. All USB data transfer types are supported.

The TAS1020 device also includes a codec port interface (C-Port) that can be configured to support several industry standard serial interface protocols. These protocols include the audio codec (AC) ’97 Revision 1.X, the audio codec (AC) ’97 Revision 2.X and several inter-IC sound (I^2S) modes.

A direct memory access (DMA) controller with two channels is provided for streaming the USB isochronous data packets to/from the codec port interface. Each DMA channel can support one USB isochronous end-point.

An on-chip phase lock loop (PLL) and adaptive clock generator (ACG) provide support for the USB synchronization modes, which include asynchronous, synchronous and adaptive.

Other on-chip MCU peripherals include an inter-IC control (I^2C) serial interface, and two 8-bit general-purpose input/output (GPIO) ports.

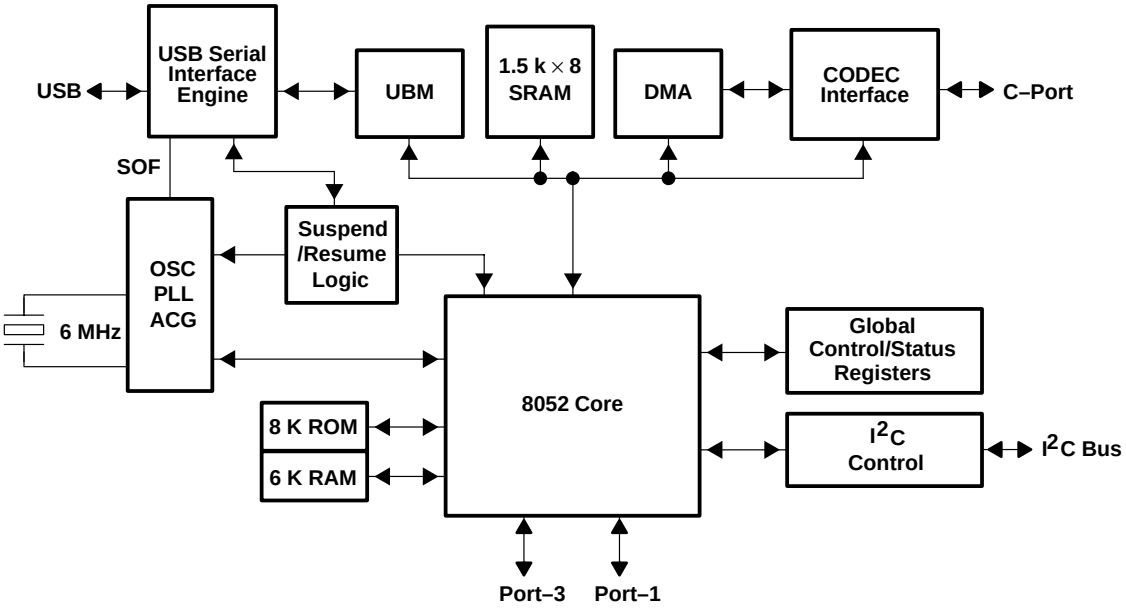
The TAS1020 device is implemented in a 3.3-V 0.25 μm CMOS technology.

1.1 Features

- **Universal Serial Bus (USB)**
  - USB specification version 1.1 compatible
  - USB audio class specification 1.0 compatible
  - Integrated USB transceiver
  - Supports 12 Mb/s data rate (full speed)
  - Supports suspend/resume and remote wake-up
  - Supports control, interrupt, bulk, and isochronous data transfer types
  - Supports up to a total of seven in end-points and seven out end-points in addition to the control end-point
  - Data transfer type, data buffer size, single or double buffering is programmable for each end-point
  - On-chip adaptive clock generator (ACG) supports asynchronous, synchronous and adaptive synchronization modes for isochronous end-points
  - To support synchronization for streaming USB audio data, the ACG can be used to generate the master clock for the codec

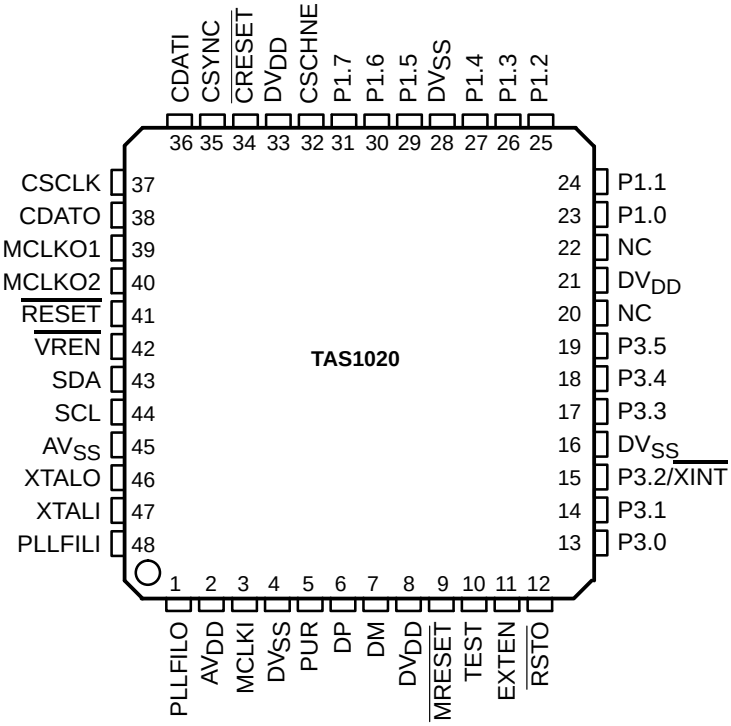
- **Micro-Controller Unit (MCU)**
  - Standard 8052 8-bit core
  - 8K bytes of program memory ROM that contains a boot loader program and a library of commonly used USB functions
  - 6016 bytes of program memory RAM which is loaded by the boot loader program
  - 256 bytes of internal data memory RAM
  - Two GPIO ports
  - MCU handles all USB control, interrupt, and bulk end-point transfers
- **DMA Controller**
  - Two DMA channels to support streaming USB audio data to/from the codec port interface
  - Each channel can support a single USB isochronous end-point
  - In the I^2S mode the device can support DAC/ADCs at different sampling frequencies
  - A circular programmable FIFO used for isochronous audio data streaming
- **Codec Port Interface**
  - Configurable to support AC'97 1.X, AC'97 2.X, AIC or I^2S serial interface formats
  - I^2S modes can support a combination of one DAC and/or two ADCs
  - Can be configured as a general-purpose serial interface
  - Can support bulk data transfer using DMA for higher throughput
- **I^2C Interface**
  - Master only interface
  - Does not support a multimaster bus environment
  - Programmable to 100 kb/s or 400 kb/s data transfer speeds
  - Supports I^2C arbitration and wait states to accomodate slow slaves
- **General Characteristics**
  - High performance 48-pin TQFP Package
  - On-chip phase-locked loop (PLL) with internal oscillator is used to generate internal clocks from a 6 MHz crystal input
  - Reset output available which is asserted for both system and USB reset
  - External MCU mode supports application firmware development
  - 8K ROM with boot loader program and commonly used USB functions library
  - 3.3 V core and I/O buffers

1.2 Functional Block Diagram



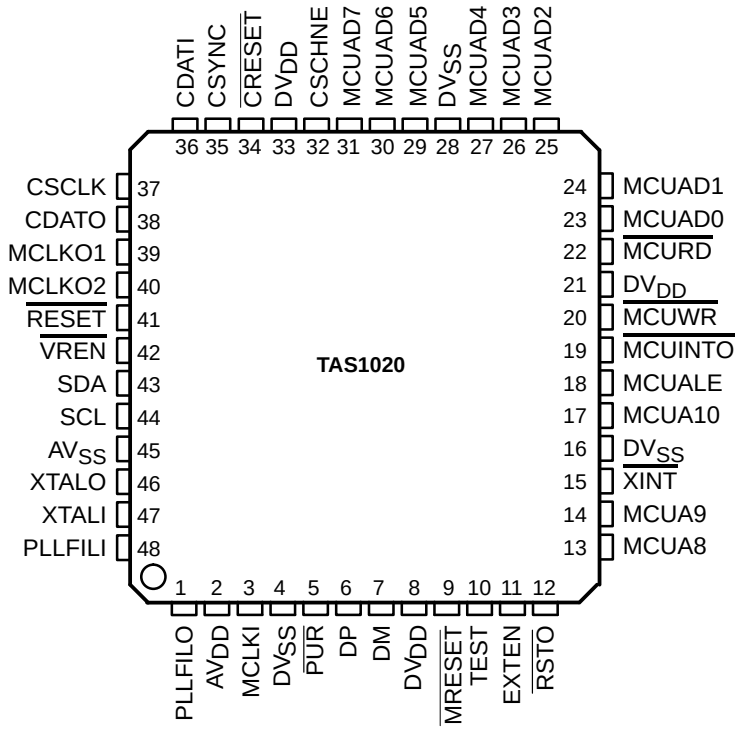
1.3 Terminal Assignments – Normal Mode

PFB PACKAGE (Normal Mode)
(TOP VIEW)

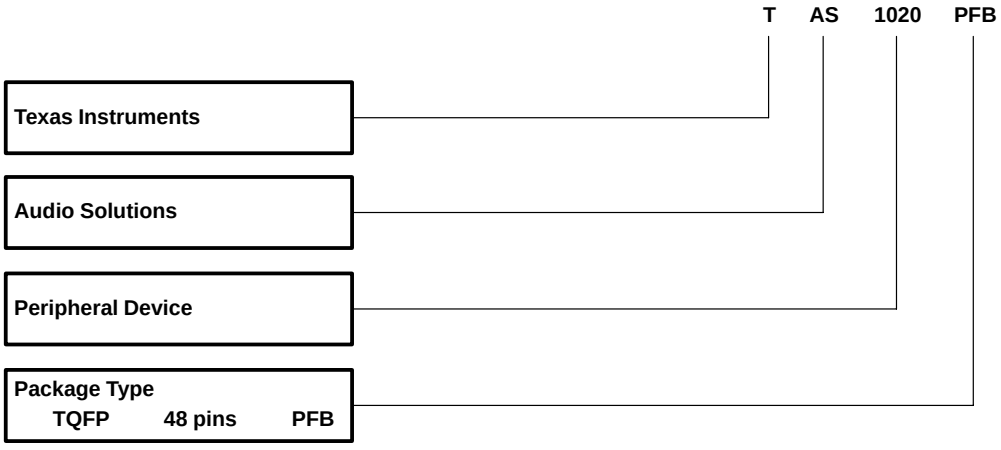


1.4 Terminal Assignments – External MCU Mode

PFB PACKAGE (External Mode)
(TOP VIEW)



1.5 Ordering Information



1.6 Terminal Functions – Normal Mode

TERMINAL			I/O	DESCRIPTION
NAME	PIN TYPE	NO.		
AV_{DD}	Power	2		3.3-V analog supply voltage
AV_{SS}	Power	45		Analog ground
CSCLK	CMOS	37	I/O	Codec port interface serial clock: CSCLK is the serial clock for the codec port interface used to clock the CSYNC, CDATO, CDATI, $\overline{\text{CRESET}}$, AND CSCHNE signals.
CSYNC	CMOS	35	I/O	Codec port interface frame sync: CSYNC is the frame synchronization signal for the codec port interface.
CDATO	CMOS	38	I/O	Codec port interface serial data out
CDATI	CMOS	36	I/O	Codec port interface serial data in
$\overline{\text{CRESET}}$	CMOS	34	I/O	Codec port interface reset output
CSCHNE	CMOS	32	I/O	Codec port interface secondary channel enable
DP	CMOS	6	I/O	USB differential pair data signal plus. DP is the positive signal of the bidirectional USB differential pair used to connect the TAS1020 device to the universal serial bus.
DM	CMOS	7	I/O	USB differential pair data signal minus. DM is the negative signal of the bidirectional USB differential pair used to connect the TAS1020 device to the universal serial bus.
DV_{DD}	Power	8, 21, 33		3.3-V digital supply voltage
DV_{SS}	Power	4, 16, 28		Digital ground
EXTEN	CMOS	11	I	External MCU mode enable: Input used to enable the device for the external MCU mode
MCLKI	CMOS	3	I	Master clock input. An input that can be used as the master clock for the codec port interface or the source for MCLKO2.
MCLKO1	CMOS	39	O	Master clock output 1: The output of the ACG that can be used as the master clock for the codec port interface and the codec.
MCLKO2	CMOS	40	O	Master clock output 2: An output that can be used as the master clock for the codec port interface and the codec used in I^2S modes for receive. This clock signal can also be used as a miscellaneous clock.
$\overline{\text{MRESET}}$	CMOS	9	I	Master reset: An active low asynchronous reset for the device that resets all logic to the default state
NC		20,22		Not used
P1.[0:7]	CMOS	23, 24, 25, 26, 27, 29, 30, 31	I/O	General-purpose I/O port [bits 0 through 1]: A bidirectional 8-bit I/O port with an internal 100 µA active pullup
P3.[0:6]	CMOS	13, 14, 15, 17, 18, 19	I/O	General-purpose I/O port [bits 0 through 1]: A bidirectional I/O port with an internal 100 µA active pullup
PLLFILI	CMOS	48	I	PLL loop filter input: Input to on-chip PLL from external filter components
PLLFILO	CMOS	1	O	PLL loop filter output: Output from on-chip PLL to external filter components
PUR	CMOS	5	O	USB data signal plus pullup resistor connect. PUR is used to connect the pullup resistor on the DP signal to 3.3-V or a 3-state. When the DP signal is connected to 3.3-V the host PC detects the connection of the TAS1020 device to the universal serial bus.
$\overline{\text{RESET}}$	CMOS	41	O	General-purpose active-low output which is memory mapped
$\overline{\text{RSTO}}$	CMOS	12	O	Reset output: An output that is active while the master reset input or the USB reset is active
SCL	CMOS	44	O	I^2C interface serial clock
SDA	CMOS	43	I/O	I^2C interface serial data
TEST	CMOS	10	I	Test mode enable: Factory test mode
$\overline{\text{VREN}}$	CMOS	42	O	General-purpose active-low output which is memory mapped
$\overline{\text{XINT}}$	CMOS	15	I	External interrupt: An active low input used by external circuitry to interrupt the on-chip 8052 MCU
XTALI	CMOS	47	I	Crystal input: Input to the on-chip oscillator from an external 6-MHz crystal
XTALO	CMOS	46	O	Crystal Output: Output from the on-chip oscillator to an external 6-MHz crystal

1.7 Terminal Functions – External MCU Mode

TERMINAL			I/O	DESCRIPTION
NAME	PIN TYPE	NO.		
AV_{DD}	Power	2	–	3.3-V Analog supply voltage
AV_{SS}	Power	45	–	Analog ground
CSCLK	CMOS	37	I/O	Codec port interface serial clock: CSCLK is the serial clock for the codec port interface used to clock the CSYNC, CDATO, CDATI, CRESET AND CSCHNE signals.
CSYNC	CMOS	35	I/O	Codec port interface frame sync: CSYNC is the frame synchronization signal for the codec port interface.
CDATO	CMOS	38	I/O	Codec port interface serial data output
CDATI	CMOS	36	I/O	Codec port interface serial data input
CRESET	CMOS	34	I/O	Codec port interface reset output
CSCHNE	CMOS	32	I/O	Codec port interface secondary channel enable
DP	CMOS	6	I/O	USB differential pair data signal plus: DP is the positive signal of the bidirectional USB differential pair used to connect the TAS1020 device to the universal serial bus.
DM	CMOS	7	I/O	USB differential pair data signal minus. DM is the negative signal of the bidirectional USB differential pair used to connect the TAS1020 device to the universal serial bus.
DV_{DD}	Power	8, 21, 33	–	3.3-V Digital supply voltage
DV_{SS}	Power	4, 16, 28	–	Digital ground
EXTEN	CMOS	11	I	External MCU mode enable: Input used to enable the device for the external MCU mode. This signal uses a 3.3 V TTL/LVCMOS input buffer.
MCLKI	CMOS	3	I	Master clock input: An input that can be used as the master clock for the codec port interface or the source for MCLKO2.
MCLKO1	CMOS	39	O	Master clock output 1: The output of the ACG that can be used as the master clock for the codec port interface and the codec.
MCLKO2	CMOS	40	O	Master clock output 2: An output that can be used as the master clock for the codec port interface and the codec. This clock signal can also be used as a miscellaneous clock.
MRESET	CMOS	9	I	Master reset: An active low asynchronous reset for the device that resets all logic to the default state.
MCUAD [0:7]	CMOS	23, 24, 25, 26, 27, 29, 30, 31	I/O	MCU multiplexed address/data bit 0: Multiplexed address bit 0/data bit 0 for external MCU access to the TAS1020 external data memory space.
MCUA [8:10]	CMOS	13, 14, 17	I/O	MCU address bus: Multiplexed address bus for external MCU access to the TAS1020 external data memory space.
MCUALE	CMOS	18	I	MCU address latch enable: Address latch enable for external MCU access to the TAS1020 external data memory space.
MCUINTO	CMOS	19	O	MCU interrupt output: Interrupt output to be used for external MCU INTO input signal. All internal TAS1020 interrupt sources are read together to generate this output signal.
MCUWR	CMOS	20	I	MCU write strobe: Write strobe for external MCU write access to the TAS1020 external data memory space.
MCURD	CMOS	22	I	MCU read strobe: Read strobe for external MCU read access to the TAS1020 external data memory space.
PLLFILI	CMOS	48	I	PLL loop filter input: Input to on-chip PLL from external filter components.
PLLFILO	CMOS	1	O	PLL loop filter output: Output to on-chip PLL from external filter components.
PUR	CMOS	5	O	USB Data signals plus pullup resistor connect. PUR is used to connect the pullup resistor on the DP signal to 3.3V to a 3-state. When the DP signal is connect to a 3.3V, the host PC should detect the connection of the TAS1020 device to the universal serial bus.
RESET	CMOS	41	O	General-purpose active-low output which is memory mapped
RSTO	CMOS	12	O	Reset output: An output that is active while the master reset input or the USB reset is active.
SCL	CMOS	44	O	I^2C interface serial clock
SDA	CMOS	43	I/O	I^2C interface serial data input/output
TEST	CMOS	10	I	Test mode enable: Factory text mode

1.7 Terminal Functions – External MCU Mode (Continued)

TERMINAL			I/O	DESCRIPTION
NAME	PIN TYPE	NO.		
$\overline{\text{VREN}}$	CMOS	42	O	General-purpose active-low output which is memory mapped.
$\overline{\text{XINT}}$	CMOS	15	I	External interrupt: An active low input used by external circuitry to interrupt the on-chip 8052 MCU.
XTALI	CMOS	47	I	Crystal input: Input to the on-chip oscillator from an external 6-MHz crystal.
XTALO	CMOS	46	O	Crystal output: Output from the on-chip oscillator to an external 6-MHz crystal.

1.8 Device Operation Modes

The EXTEN and TEST pins define the mode that the TAS1020 is in after reset.

MODE	EXTEN	TEST
Normal mode – internal MCU	0	0
External MCU mode	1	0
Factory test	0	1
Factory test	1	1

1.9 Terminal Assignments for Codec Port Interface Modes

The codec port interface has five modes of operation that support AC97, I^2S, and AIC codecs. There is also a general-purpose mode that is not specific to a serial interface. The mode is programmed by writing to the mode select field of the codec port interface configuration register 1 (CPTCNF1). The codec port interface terminals CSYNC, CSCLK, CDATO, CDATI, $\overline{\text{CRESET}}$, and CSCHNE take on functionality appropriate to the mode programmed as shown in the following table.

TERMINAL		GP Mode 0		AIC Mode 1		AC '97 v1.X Mode 2		AC '97 v2.X Mode 3		I^2S Mode 4		I^2S Mode 5	
NO.	NAME												
35	CSYNC	CSYNC	I/O	$\overline{\text{FS}}$	O	SYNC	O	SYNC	O	LRCK	O	LRCK1	O
37	CSCLK	CSCLK	I/O	SCLK	O	BIT_CLK	I	BIT_CLK	I	SCLK	O	SCLK1	O
38	CDATO	CDATO	O	DOUT	O	SD_OUT	O	SD_OUT	O	SDOUT1	O	SDOUT1	O
36	CDATI	CDATI	I	DIN	I	SD_IN	I	SD_IN1	I	SDIN1	I	SDIN2	I
34	$\overline{\text{CRESET}}$	$\overline{\text{CRESET}}$	O	$\overline{\text{RESET}}$	O	$\overline{\text{RESET}}$	O	$\overline{\text{RESET}}$	O	$\overline{\text{CRESET}}$	O	SCLK2	O
32	CSCHNE	NC	O	FC	O	NC	O	SD_IN2	I	SDIN2	I	LRCK2	O

NOTES: 1. Signal names and I/O direction are with respect to the TAS1020 device. The signal names used for the TAS1020 terminals for the various codec port interface modes reflect the nomenclature used by the codec devices.
2. NC indicates no connection for the terminal in a particular mode. The TAS1020 device drives the signal as an output for these cases.
3. The CSYNC and CSCLK signals can be programmed as either an input or an output in the general-purpose mode.

2 Description

2.1 Architectural Overview

2.1.1 Oscillator and PLL

Using an external 6-MHz crystal, the TAS1020 derives the fundamental 48-MHz internal clock signal using an on-chip oscillator and PLL. Using the PLL output, the other required clock signals are generated by the clock generator and adaptive clock generator.

2.1.2 Clock Generator and Sequencer Logic

Utilizing the 48-MHz input from the PLL, the clock generator logic generates all internal clock signals, except for the codec port interface master clock (MCLK) and serial clock (CSCLK) signals. The TAS1020 internal clocks include the 48-MHz clock, a 24-MHz clock, a 12-MHz clock and a USB clock. The USB clock also has a frequency of 12-MHz. The USB clock is the same as the 12-MHz clock when the TAS1020 is transmitting data and is derived from the data when the TAS1020 is receiving data. To derive the USB clock when receiving USB data, the TAS1020 utilizes an internal digital PLL (DPLL) that uses the 48-MHz clock.

The sequencer logic controls the access to the SRAM used for the USB end-point configuration blocks and the USB end-point buffer space. The SRAM can be accessed by the MCU, USB buffer manager (UBM), or DMA channels. The sequencer controls the access to the memory using a round-robin fixed priority arbitration scheme. This means that the sequencer logic generates grant signals for the MCU, UBM, and DMA channels at a predetermined fixed frequency.

2.1.3 Adaptive Clock Generator (ACG)

The adaptive clock generator is used to generate a master clock output signal (MCLKO) to be used by the codec port interface and the codec device. To synchronize data sent to or received from the codec to the USB frame rate, the MCLKO signal generated by the adaptive clock generator must be used. The synchronization of the MCLKO signal to the USB frame rate is controlled by the MCU by programming the adaptive clock generator frequency value. The MCLKO frequency is monitored by the MCU and updated as required. For asynchronous operation, an external source can be used to generate a master clock input signal (MCLKI) to be used by the codec port interface. In this scenario, the codec device also uses the same master clock signal (MCLKI).

2.1.4 USB Transceiver

The TAS1020 provides an integrated transceiver for the USB port. The transceiver includes a differential output driver, a differential input receiver, and two single ended input buffers. The transceiver connects to the USB DP and DM signal terminals.

2.1.5 USB Serial Interface Engine (SIE)

The serial interface engine logic manages the USB packet protocol requirements for the packets being received and transmitted on the USB by the TAS1020 device. For packets being received, the SIE decodes the packet identifier field (PID) to determine the type of packet being received and to ensure the PID is valid. For token packets and data packets being received, the SIE calculates the packet cycle redundancy check (CRC) and compares the value to the CRC contained in the packet to verify that the packet was not corrupted during transmission. For token packets and data packets being transmitted, the SIE generates the CRC that is transmitted with the packet. For packets being transmitted, the SIE also generates the synchronization field (SYNC) that is an eight bit filed at the beginning of each packet. In addition, the SIE generates the correct PID for all packets being transmitted. Another major function of the SIE is the overall serial-to-parallel conversion of the data packets being received and the parallel-to-serial conversion of the data packets being transmitted.

2.1.6 USB Buffer Manager (UBM)

The USB buffer manager provides the control logic that interfaces the SIE to the USB end-point buffers. One of the major functions of the UBM is to decode the USB function address to determine if the host PC is addressing the TAS1020 device USB peripheral function. In addition, the end-point address field and direction signal are decoded to determine which particular USB end-point is being addressed. Based on the direction of the USB transaction and the end-point number, the UBM will either write or read the data packet to/from the appropriate USB end-point data buffer.

2.1.7 USB Frame Timer

The USB frame timer logic receives the start of frame (SOF) packet from the host PC each USB frame. Each frame, the logic stores the 11-bit frame number value from the SOF packet in a register and asserts the internal SOF signal. The frame number register can be read by the MCU and the value can be used as a time stamp. For USB frames in which the SOF packet is corrupted or not received, the frame timer logic will generate a pseudo start of frame (PSOF) signal and increment the frame number register.

2.1.8 USB Suspend and Resume Logic

The USB suspend and resume logic detects suspend and resume conditions on the USB. This logic also provides the internal signals used to control the TAS1020 device when these conditions occur. The capability to resume operation from a suspend condition with a locally generated remote wake-up event is also provided.

2.1.9 MCU Core

The TAS1020 uses an 8-bit microcontroller core that is based on the industry standard 8052. The MCU is software compatible with the 8052, 8032, 80C52, 80C53, and 87C52 MCUs. The 8052 MCU is the processing core of the TAS1020 and handles all USB control, interrupt and bulk end-point transfers. In addition, the MCU can also be the source or sink for USB isochronous end-point transfers.

2.1.10 MCU Memory

In accordance with the industry standard 8052, the TAS1020 MCU memory is organized into program memory, external data memory and internal data memory. A boot ROM program is used to download the application code to a 6K byte RAM that is mapped to the program memory space. The external data memory includes the USB end-point configuration blocks, USB data buffers, and memory mapped registers. The total external data memory space available is 1.5K bytes. A total of 256 bytes are provided for the internal data memory.

2.1.11 USB End-Point Configuration Blocks and End-Point Buffer Space

The USB end-point configuration blocks are used by the MCU to configure and operate the required USB end-points for a particular application. In addition to the control end-point, the TAS1020 supports a total of seven in end-points and seven out end-points. A set of six bytes is provided for each end-point to specify the end-point type, buffer address, buffer size, and data packet byte count.

The USB end-point buffer space provided is a total of 1440 bytes. The space is totally configurable by the MCU for a particular application. Therefore, the MCU can configure each buffer based on the total number of end-points to be used, the maximum packet size to be used for each end-point, and the selection of single or double buffering.

2.1.12 DMA Controller

Two DMA channels are provided to support the streaming of data for USB isochronous and bulk end-points. Each DMA channel can support one USB isochronous and bulk end-points, either in or out. The DMA channels are used to stream data between the USB end-point data buffers and the codec port interface. The USB end-point number and direction can be programmed for each DMA channel. Also, the codec port interface time slots to be serviced by each DMA channel can be programmed.

2.1.13 Codec Port Interface

The TAS1020 provides a configurable full duplex bidirectional serial interface that can be used to connect to a codec or another device for streaming USB isochronous data. The interface can be configured to support several different industry standard protocols, including AC '97 1.X, AC '97 2.X, AIC, and I^2S. The TAS1020 also has a general-purpose mode to support other protocols.

2.1.14 I^2C Interface

The I^2C interface logic provides a two-wire serial interface that the 8052 MCU can use to access other ICs. The TAS1020 is an I^2C master device only and supports single byte or multiple byte read and write operations. The interface can be programmed to operate at either 100 kbps or 400 kbps. In addition, the protocol supports 8-bit or 16-bit addressing for accessing the I^2C slave device memory locations. The TAS1020 supports I^2C wait states. This means slaves can assert wait state on the I^2C bus by pulling the SCL low.

2.1.15 General-Purpose IO Ports (GPIO)

The TAS1020 provides two general-purpose IO ports that are controlled by the internal 8052 MCU. The two ports are port 1 and port 3. Port 3 bit location 2 is used as the external interrupt ($\overline{\text{XINT}}$) input to the TAS1020.

Each bit of both ports can be independently used as either an input or output. Hence each port bit consists of an output buffer, an input buffer, and a pullup resistor. The pullup resistors for each GPIO port can be disabled using the PUDIS bits in the global control register. Any port 3 pin can be used to wake up the host PC in low-power suspend mode.

2.1.16 Interrupt Logic

The interrupt logic monitors the various conditions that can cause an interrupt and asserts the interrupt 0 (INTO) input to the 8052 MCU accordingly. All of the TAS1020 internal interrupt sources and the external interrupt ($\overline{\text{XINT}}$) input are OR'ed together to generate the INT0 signal. An interrupt vector register is used by the MCU to identify the interrupt source.

2.1.17 Reset Logic

An external master reset ($\overline{\text{MRESET}}$) input signal that is asynchronous to the internal clocks is used to reset the TAS1020 logic. In addition to the master reset, the TAS1020 logic can be reset with the USB reset from the host PC. The TAS1020 also provides a reset output ($\overline{\text{RSTO}}$) signal that can be used by external devices. This signal is asserted when either a master reset or USB reset occurs.

2.2 Device Operation

The operation of the TAS1020 is explained in the following sections. For additional information on USB, refer to the Universal Serial Bus Specification version 1.1.

2.2.1 Clock Generation

The TAS1020 requires an external 6-MHz crystal and PLL loop filter components connected as shown in Figure 4-1 to derive all the clocks needed for both USB and codec operation. Using the low frequency 6-MHz crystal and generating the required higher frequency clocks internal to the IC is a major advantage regarding EMI.

2.2.2 Device Initialization

After a power-on reset is applied to the TAS1020 device, the 8052 MCU executes a boot loader program from the 8K byte boot ROM mapped to the program memory space. During device initialization, the boot loader program downloads the application program code from an external EEPROM through the I^2C interface. This requires that a binary image of the application code be written to the 6K byte code RAM in the TAS1020 device.

All memory mapped registers are initialized to a default value as defined in Appendix A, *MCU Memory and Memory-Mapped Registers*. The TAS1020 device powers up with a default function address of zero and disconnected from the USB.

2.2.2.1 Boot Load from EEPROM

Loading the application code from an external serial EEPROM requires a preprogrammed memory device containing an informative header and the application code. While the application code is being downloaded, the TAS1020 remains disconnected from the USB. When the code download is complete, execution of the application code connects the TAS1020 to the USB. In this situation, the TAS1020 enumerates using the vendor ID and product ID contained in the application code.

2.2.2.2 EEPROM Header

For both application code and USB device information stored in a EEPROM device, a common header format is used that precedes the data payload. Table 2-1 shows the format and information contained in the header.

Table 2–1. EEPROM Header

OFFSET	TYPE	SIZE	VALUE
0	headerChksum	1	Check sum of the header by adding the header, excluding the header cheksum, in bytes.
1	HeaderSize	1	Size of the header including strings if applied
2	Signature	2	Signature: 0x1234
4	VenderID	2	USB Vendor ID
6	ProductID	2	USB Vendor ID
8	ProductVersion	1	Product version
9	FirmwareVersion	1	Firmware version
10	UsbAttributes	1	USB attributes: Bit 0: If set to 1, the header includes all three strings: language, manufacture, and product strings, if set to 0, the header does not include any string. Bit 2: If set to 1, the device can be self powered, if set to 0, it is not self powered. Bit 3: If set to 1, the device can be bus powered, if set to 0, it is not be bus powered. Bit 1 and 4 ... 7: Not used.
11	MaxPower	1	Maximum power the device needs in units of 2 mA.
12	Attribute	1	Device attributes: Bit 3: If set to 1, the devices EEPROM support 400 kHz, if set to 0, it can not support 400 MHz. Bit 0: If set to 1, the CPU speed runs at 24 MHz, if set to 0, the CPU speed runs at 12 MHz. Bit 1, 2 and 4 ... 7: Not used.
13	WPageSize	1	Maximum I^2C write page size
14	DataType	1	This value defines if the device is application EEPROM, device EEPROM. 0x01: Application EEPROM 0x02: Device EEPROM Other values are invalid.
15	RpageSize	1	Maximum I^2C read page size. If the value is zero, the whole payLoadSize is read in one I^2C read setup.
16	payLoadSize	2	Size of the application, if using EEPROM as an application EEPROM, otherwise the value is 0.
0xxx	Language string	4	Language string in standard USB string format if applied.
0xxx	Manufacture string	...	Manufacture string in standard USB string format if applied.
0xxx	Product string	...	Product string in standard USB string format if applied.
0xxx	Application Code	...	Application code if applied

The signature field is used for the detection of a EEPROM device connected to the TAS1020. The header size field supports future updates of the header. Data begins right after the header. The version field identifies the header version. The EEPROM type field identifies the specific EEPROM device being used. The data type field describes the nature of data stored in the EEPROM (application code or USB device information). The data size field holds the

length of the data payload starting from the end of the header. The check sum field contains the check sum for the data payload portion of the EEPROM.

2.2.2.3 EEPROM Data Type

The two types of data that are stored in the EEPROM are application code and USB device information.

2.2.2.3.1 Application Code

Application firmware is stored as a binary image of the code. The binary image is mapped to the MCU program memory space starting at address zero and is stored in the EEPROM as a continuous linear block starting after the header information. A utility program is available to convert a file in Intel hexadecimal format to a binary image data file and appends it to the header.

2.2.2.4 EEPROM Device Type

The TAS1020 boot loader program supports several different types of serial EEPROM devices. The boot loader program automatically identifies the EEPROM type from the header information and uses the correct serial interface protocol accordingly. The boot loader program uses an I^2C slave device address of A0h for the serial EEPROM device.

These EEPROM devices require an I^2C device address in addition to a two byte data word address. These devices require the full 7-bit I^2C device address. Depending on the memory size of the EEPROM device being used, the most significant three or four bits of the two byte data word address are don't care bits.

2.2.3 USB Enumeration

USB enumeration is accomplished by interaction between the host PC software and the TAS1020 code. After power-on reset the boot loader code first reads the information from the EEPROM, then runs the application code. The application code connects the TAS1020 to the USB. During the enumeration, the application code identifies the device as an application specific device and the host loads the appropriate host driver(s). The boot loader and application code both use the CONT, SDW and FRSTE bits to control the enumeration process. The function connect (CONT) bit is set to a 1 by the MCU to connect the TAS1020 device to the USB. When this bit is set to a 1, the USB data plus pullup resistor (PUR) output signal is enabled, which connects the pullup on the PCB to the TAS1020 3.3-V digital supply voltage. When this bit is cleared to a 0, the PUR output is in the 3-state mode. This bit is not affected by a USB reset. The shadow the boot ROM (SDW) bit is set to a 1 by the MCU to switch the MCU memory configuration from boot loader mode to normal operating mode. The function reset enable (FRSTE) bit is set to a 1 by the MCU to enable the USB reset to reset all internal logic including the MCU. However, the shadow the ROM (SDW) and the USB function connect (CONT) bits will not be reset. When this bit is set, the reset output ($\overline{\text{RSTO}}$) signal from the TAS1020 device is also active when a USB reset occurs. This bit is not affected by USB reset.

2.2.4 USB Reset

The TAS1020 can detect a USB reset condition. When the reset occurs, the TAS1020 responds by setting the function reset (RSTR) bit in the USB status register (USBSTA). If the corresponding function reset bit in the USB interrupt mask register is set, an MCU interrupt will be generated and the USB function reset (0x17) vector will appear in the interrupt vector register (VECINT).

The function reset enable bit (FRSTE) in the USB control register (USBCTL) is used to control the extent to which the internal logic is reset. The function reset enable bit is set to a 1 by the MCU to enable the USB reset to reset all internal logic including the MCU. However, the shadow the ROM (SDW) and the USB function connect (CONT) bits will not be reset. When the FRSTE bit is set, the reset output ($\overline{\text{RSTO}}$) signal from the device also is active when a USB reset occurs. This bit is not affected by USB reset.

2.2.5 USB Suspend and Resume Modes

A suspend condition is defined as a constant idle state on the bus for more than 3 ms. A USB device must actually be in the suspend state no more than 10 ms after the suspend condition is detected. There are two ways for the TAS1020 device to exit the suspend mode: 1) detection of USB resume signaling and 2) detection of a local remote wake-up event.

2.2.5.1 USB Suspend Mode

When a suspend condition is detected on the USB, the suspend/resume logic sets the function suspend request bit (SUSR) in the USB status register. As a result, the function suspend request interrupt (SUSR) is generated. To enter the low-power suspend state and disable all TAS1020 device clocks, the MCU firmware must set the idle mode bit (IDL), which is bit 0 in the MCU power control (PCON) register. Note that the IDL bit must be set in the main ( ) routine, not in the suspend interrupt service routine. The instruction that sets the IDL bit is the last instruction executed before the MCU goes to idle mode. In idle mode, the MCU status is preserved. Note that the low power suspend state is a state in which the TAS1020 clocks are disabled and the IC will consume the least amount of power possible.

2.2.5.2 USB Resume Mode

When the TAS1020 is in a suspend state, any non-idle signaling on the USB is detected by the suspend/resume logic and device operation resumes. When the resume signal is detected, the TAS1020 clocks are enabled, the function resume request bit (RESR) will be set, and the function resume request interrupt (RESR) will be generated. The function resume request interrupt to the MCU automatically clears the idle mode bit in the PCON register. As a result, MCU operation resumes servicing the new interrupt. After the RETI from the ISR, the next instruction to be executed is the one following the instruction that set the IDL bit. Note that if the low-power suspend state was not entered by setting the IDL bit, the clocks are already enabled, and the IDL bit is already cleared.

2.2.5.3 USB Remote Wake-Up Mode

The TAS1020 device has the capability to remotely wake-up the USB by generating resume signaling upstream. Note that this feature must be enabled by the host software with the SET_FEATURE DEVICE_REMOTE_WAKEUP request. The remote wake-up resume signaling must not be generated until the suspend state has been active for at least 5 ms. In addition, the remote wake-up resume signaling must be generated for at least 1ms but for no more than 15 ms. When the TAS1020 is in the low power suspend state, asserting the external interrupt input ($\overline{\text{XINT}}$) to the device enables the clocks and generates the XINT interrupt. The XINT interrupt to the MCU automatically clears the idle mode bit in the PCON register. As a result, MCU operation resumes servicing the new interrupt. After the RETI from the ISR, the next instruction to be executed is the one following the instruction that set the IDL bit. Note that if the low-power suspend state was not entered by setting the IDL bit, the clocks is already enabled and the IDL bit is already cleared. When the firmware sets the remote wake-up request bit (RWUP) in the USB control register, the suspend/resume logic generates the resume signaling upstream on the USB.

The remote wakeup in the TAS1020 is set to level triggered, a default value. The bit TCON.2 in the MCU determines whether it is an edge triggered or a level triggered. 1 indicates edge triggered, 0 indicates level triggered. The default value is 0. In order to use the remote wakeup function, RWUP bit has to be set in the USBCTL register. If XINT pin is used for remote wakeup, XINTEN bit has to be set in the GLOBCTL register. If one of the port 3 pins is used for remote wakeup, the bit associated with the pin has to be unmasked in the P3MSK register.

2.2.6 Adaptive Clock Generator (ACG)

The adaptive clock generator is used to generate two programmable master clock output signals (MCLKO and MCLKO2) that can be used by the codec port interface and the codec device. The ACG can be used to generate the master clock for the codec for USB asynchronous, synchronous, and adaptive modes of operation. However, for the USB asynchronous mode of operation, an external clock can be used to drive the MCLKI signal of the TAS1020. In this scenario, the MCLKI signal is used as the clock source for the codec port interface instead of the clock output from the ACG.

A block diagram of the adaptive clock generator is shown in Figure 2–1. The frequency synthesizer circuit generates a programmable clock with a frequency range of 12–25 MHz. The output of the frequency synthesizer feeds the divide-by-M circuit, which can be programmed to divide by 1 to 16. As a result, the frequency range of the MCLKO signal is 750 kHz to 25 MHz. The duty cycle of the MCLKO signal is 50% for all programmable MCLKO frequencies.

As shown in Figure 2–1, the C-Port serial clock (CSCLK) is derived by setting the *divide by B* value in codec interface configuration register CPTNCF4 [2:0] and the C-Port serial clock2 (CSCLK2) is derived by setting the *divide by B* value in codec port receive interface register 4 CPTRXCNF4 [2:0].

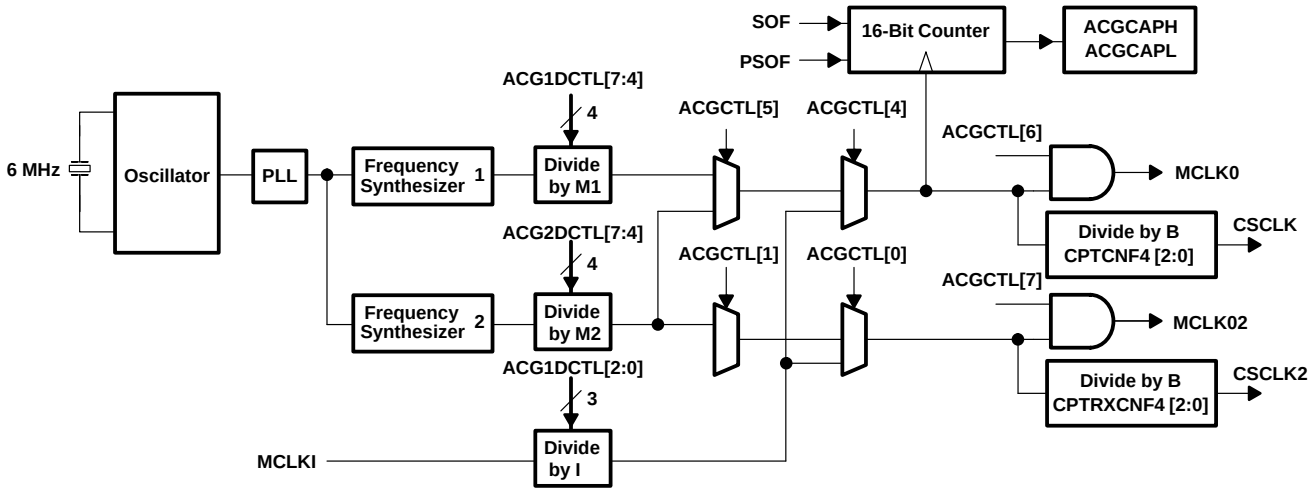


Figure 2–1. Adaptive Clock Generator

The ACG is controlled by the following registers. Refer to section A.5.3 for details.

FUNCTIONAL REGISTER	ACTUAL BYTE-WIDE REGISTERS		
24-bit frequency register #1	ACG1FRQ2	ACG1FRQ1	ACG1FRQ0
16-bit capture register		ACGCAPH	ACGCAPL
8-bit synthesizer 1 divider control register			ACG1DCTL
8-bit ACG control register			ACGCTL
24-bit frequency register #2	ACG2FRQ2	ACG2FRQ1	ACG2FRQ0
8-bit synthesizer 2 divider control register			ACG2DCTL

The main functional modules of the ACG are described in the following sections.

2.2.6.1 Programmable Frequency Synthesizer

The 24-bit ACG frequency register value is used to program the frequency synthesizer. This programming results in high resolution to accurately select the desired codec master clock frequency. The value of the frequency register may be updated by the MCU while the ACG is running. In audio applications, the firmware can adjust the frequency value by ±LSB or more to lock onto the USB start-of-frame (SOF) signal and achieve a synchronous mode of operation. The 24-bit frequency register value is updated and used by the frequency synthesizer only when MCU writes to the ACGFRQ0 register.

Depending on the application, a smaller number of bits for controlling the synthesizer frequency can be chosen. The frequency resolution also depends on the actual frequency being used. In general, the frequency resolution is less for higher frequencies and more for lower frequencies. This is due to the fact that the 208 ps frequency resolution becomes more significant compared to the period at higher frequencies than at lower frequencies. The resolution increases with the number of bits used to represent the frequency as the quantization error reduces because more bits are used to represent a fractional number.

The clock frequency of the MCLKO output signal is calculated by using the formula:

For N > 24 and N < 50, MCLKO frequency = (25/N) × 192/8 MHz

For N = 50, MCLKO frequency = 96/8 MHz

Where N is the value in the 24-bit frequency register (ACGFRQ). The value of N can range from 24 to 50. The six most significant bits of the 24-bit frequency register are used to represent the integer portion of N, and the remaining 18 bits of the frequency register are used to represent the fractional portion of N. An example is shown below.

Example Frequency Register Calculation
Suppose the desired MCLKO frequency is 24.576 MHz. Using the above formula, N = 24.4140625 decimal. To determine the binary value to be written to the ACGFRQ register, separately convert the integer value (24) to 6-bit binary and the fractional value (4140625) to 18-bit binary. As a result, the 24-bit binary value is 011000.011010100000000000.

The corresponding values to program into the ACGFRQ registers are:

ACGFRQ2 = 01100001b = 61h

ACGFRQ1 = 10101000b = A8h

ACGFRQ0 = 00000000b = 00h

Keep in mind that writing to the ACGFRQ0 register loads the frequency synthesizer with the new 24-bit value.

Example Frequency Resolution Calculation
To illustrate the frequency resolution capabilities of the ACG, the next possible higher and lower frequencies for MCLKO can be calculated.

To get the next possible higher frequency of MCLKO equal to 24.57600384 MHz, increase the value of N by 1 LSB. Thus, N = 011000.011010100000000001 binary.

To get the next possible lower frequency of MCLKO equal to 24.57599600 MHz, decrease the value of N by 1 LSB. Thus, N = 011000.011010011111111111 binary.

For this example with a nominal MCLKO frequency of 24.576 MHz, the frequency resolution is approximately 4 Hz.

Table 2-2 lists typically used frequencies and corresponding ACG frequency registers.

Table 2–2. ACG Frequency Registers

SYNTHESIZED CLOCK OUTPUT	ACG1FRQ2/ACG2FRQ2	ACG1FRQ1/ACG2FRQ1	ACG1FRQ0/ACG2FRQ0
25 MHz	0x60	0	0
24.576 MHz	0x61	0×A8	0x0F
22.579 MHz	0x6A	0x4B	0x20
18.432 MHz	0x82	0x35	0x55
16.934 MHz	0x8D	0xBA	0x09
16.384 MHz	0x92	0x7C	0x00
12.288 MHz	0xC3	0x50	0x00
12 MHz	0xC8	0	0

2.2.6.2 Capture Counter and Register

The capture counter and register circuit consists of a 16-bit free running counter which runs at the capture clock frequency. The capture clock source can be selected by using the MCLKCP bit in the ACGCTL register to select either the MCLKO or MCLKO2 signal. With each USB start-of-frame (SOF) signal or pseudo-start-of-frame (PSOF) signal, the capture counter value is stored into the 16-bit capture register. This value is valid until the next SOF or PSOF signal occurs (~1 ms). The MCU can read the 16-bit capture register value by reading the ACGCAPH and ACGCAPL registers.

2.2.7 USB Transfers

The TAS1020 device supports all the USB data transfer types: control, bulk, interrupt, and isochronous. In accordance with the USB specification, endpoint zero is reserved for the control endpoint and is bidirectional. In addition to the control endpoint, the TAS1020 is capable of supporting up to 7 in endpoints and 7 out endpoints. These additional endpoints can be configured as bulk, interrupt, or isochronous endpoints. The MCU handles all control, bulk, and interrupt endpoint transactions. In addition, the MCU can handle isochronous endpoint transactions, such as a rate feedback endpoint to the host PC. However, for streaming isochronous data between the host PC and the codec interface port, the DMA channels are provided.

2.2.7.1 Control Transfers

Control transfers are used for configuration, command, and status communication between the host PC and the TAS1020 device. Control transfers to the TAS1020 device use in endpoint 0 and out endpoint 0. The three types of control transfers are control write, control write with no data stage, and control read. Note that the control endpoint must be initialized before connecting the TAS1020 device to the USB.

2.2.7.1.1 Control Write Transfer (Out Transfer)

The host PC uses a control write transfer to write data to the USB function. A control write transfer consists of a setup stage transaction, at least one out data stage transaction, and an in-status stage transaction.

The steps to be followed for a control write transfer are as follows:

Setup Stage Transaction:

1. MCU initializes in endpoint 0 and out endpoint 0 by programming the appropriate USB endpoint configuration blocks. This entails programming the buffer size and buffer base address, selecting the buffer mode, enabling the endpoint interrupt, initializing the TOGGLE bit, enabling the endpoint, and clearing the NACK bit for both in endpoint 0 and out endpoint 0.
2. The host PC sends a setup token followed by the setup data packet addressed to out endpoint 0. If the data is received without an error, then the UBM writes the data to the setup data packet buffer, set the setup stage transaction (SETUP) bit to a 1 in the USB status register, return an ACK handshake to the host PC, and assert the setup stage transaction interrupt. Note that as long as the setup transaction stage (SETUP) bit is set to a 1, the UBM returns a NACK handshake for any data stage or status stage transactions transactions regardless of the endpoint 0 NACK or STALL bit values.
3. The MCU services the interrupt and reads the setup data packet from the buffer then decodes the command. If the command is not supported or valid, the MCU should set the STALL bit in the out endpoint 0 configuration byte and the endpoint 0 configuration byte before clearing the setup stage transaction (SETUP) bit. This causes the device to return a STALL handshake for any data stage or status stage transactions. After reading the data packet and decoding the command, the MCU clears the interrupt, which automatically clears the setup stage transaction status bit. The MCU also sets the TOGGLE bit in the out endpoint 0 configuration byte to a 1. For control write transfers, the PID used by the host for the first out data packet is a DATA1 PID and the TOGGLE bit must match.

Data Stage Transaction:

1. The host PC sends an out token packet followed by a data packed addressed to out endpoint 0. If the data packet is received without errors then the UBM writes the data to the endpoint buffer, updates the data count value, toggles the TOGGLE bit, sets the NACK bit to a 1, returns an ACK handshake to the host PC, and asserts the endpoint interrupt.
2. The MCU services the interrupt and reads the data packet from the buffer. To read the data packet, the MCU first must obtain the data count value. After reading the data packet, the MCU must clear the interrupt and clear the NACK bit to allow the reception of the next data packet from the host PC.

3. If the NACK bit is set to a 1 when the in token packet is received, the UBM simply returns a NAK handshake to the host PC. If the STALL bit is set to a 1 when the in token packet is received, the UBM simply returns a STALL handshake to the host PC. If a CRC or bit stuff error occurs when the data packet is received, then no handshake is returned to the host PC.

Status Stage Transaction:

1. For in endpoint 0, the MCU updates the data count value to zero, sets the TOGGLE bit to 1, then clears the NACK bit to a 0 to enable the data packet to be sent to the host PC. Note that for a status stage transaction a null data packet with a DATA1 PID is sent to the host PC.
2. The host PC sends an in token packet addressed to in endpoint 0. After receiving the in token, the UBM transmits a null data packet to the host PC. If the data packet is received without errors by the host PC, then an ACK handshake is returned. The UBM then toggles the TOGGLE bit, sets the NACK bit to a 1, and asserts the endpoint interrupt.
3. If the NACK bit is set to a 1 when the in token packet is received, the UBM simply returns a NAK handshake to the host PC. If the STALL bit is set to a 1 when the in token packet is received, the UBM simply returns a STALL handshake to the host PC. If no handshake packet is received from the host PC, then the UBM prepares to retransmit the same data packet again.

2.2.7.1.2 Control Write With No Data Stage Transfer (Out Transfer)

The host PC uses a control write transfer to write data to the USB function. A control write with no data stage transfer consists of a setup stage transaction and an in status stage transaction. For this type of transfer, the data to be written to the USB function is contained in the two byte value field of the setup stage transaction data packet.

The steps to be followed for a control write with no data stage transfer are as follows:

1. MCU initializes in endpoint 0 and out endpint 0 by programming the appropriate USB endpoint configuration blocks. This entails programming the buffer size and buffer base address, selecting the buffer mode, enabling the endpoint interrupt, initializing the TOGGLE bit, enabling the endpoint, and clearing the NACK bit for both in endpoint 0 and out endpoint 0.

Setup Stage Transaction:

2. The host PC sends a setup token packet followed by the setup data packet addressed to out endpoint 0. If the data is received without an error then the UBM writes the data to the setup data packet buffer, sets the setup state transaction (SETUP) bit to a 1 in the USB status register, returns an ACK handshake to the host PC, and asserts the setup stage transaction interrupt. Note that as long as the setup transaction (SETUP) bit is set to a 1, the UBM returns a NAK handshake for any data stage or status stage transactions regardless of the endpoint 0 NACK or STALL bit values.
3. The MCU services the interrupt and reads the setup data packet from the buffer then decodes the command. If the command is not supported or valid, the MCU sets the STALL bit in the out endpoint 0 configuration byte and the in endpoint 0 configuration byte before clearing the setup stage transaction (SETUP) bit. This causes the device to return a STALL handshake for an data stage or status stage transactions. After reading the data packet and decoding the command, the MCU clears the interrupt, which automatically clears the setup stage transaction status bit.

Data Stage Transaction (s): Note, there are no data stage transactions for this type of transfer.

Status Stage Transaction:

1. For in endpoint 0, the MCU updates the data count value to zero, sets the TOGGLE bit to 1, then clears the NACK bit to a 0 to enable the data packet to be sent to the host PC. Note that for a status stage transaction a null data packet with a DATA1 PID is sent to the host PC.
2. The host PC sends an in taken packet addressed to in endpoint 0. After receiving the in token, the UBM transmits a null data packet to the host PC. If the data packet is received without errors by the host PC, then an ACK handshake is returned. The UBM then toggles the TOGGLE bit, sets the NACK bit to a 1, and asserts the endpoint interrupt.

3. If the NACK bit is set to a 1 when the in token packet is received, the UBM simply returns a NAK handshake to the host PC. If the STALL bit is set to a 1 when the in token packet is received, the UBM simply returns a STALL handshake to the host PC. If no handshake packet is received from the host PC, then the UBM prepares to retransmit the same data packet again.

2.2.7.1.3 Control Read Transfer (In Transfer)

The host PC uses a control read transfer to read data from the USB function. A control read transfer consists of a setup stage transaction, at least one in data stage transaction, and an out-status stage transaction.

The steps to be followed for a control read transfer are as follows:

1. MCU initializes in endpoint 0 and out endpoint 0 by programming the appropriate USB endpoint configuration blocks. This entails programming the buffer size and buffer base address, selecting the buffer mode, enabling the endpoint interrupt, initializing the TOGGLE bit, enabling the endpoint, and clearing the NACK bit for both in endpoint 0 and out endpoint 0.

Setup Stage Transaction:

2. The host PC sends a setup token followed by the setup data packet addressed to out endpoint 0. If the data is received without an error, then the UBM writes the data to the setup data packet buffer, sets the setup stage transaction (SETUP) bit to a 1 in the USB status register, returns an ACK handshake to the host PC, and asserts the setup stage transaction interrupt. Note that as long as the setup transaction stage transactions (SETUP) bit is set to a 1, the UBM returns a NACK handshake for any data stage or status stage transactions regardless of the endpoint 0 NACK or STALL bit values.
3. The MCU services the interrupt and reads the setup data packet from the buffer then decodes the command. If the command is not supported or valid, the MCU sets the STALL bit in the out endpoint 0 configuration byte and the endpoint 0 configuration byte before clearing the setup stage transaction (SETUP) bit. This causes the device to return a STALL handshake for any data stage or status stage transactions. After reading the data packet and decoding the command, the MCU clears the interrupt, which automatically clears the setup stage transaction status bit. The MCU also sets the TOGGLE bit in the out endpoint 0 configuration byte to a 1. For control read transfers, the PID used by the host for the first out data packet is a DATA1 PID.

Data Stage Transaction:

1. The data packet to be sent to the host PC is written to the in endpoint 0 buffer by the MCU. The MCU also updates the data count value then clears the in endpoint 0 NACK bit to a 0 to enable the data packet to be sent to the host PC.
2. The host PC sends an out token packet addressed to out endpoint 0. After receiving the in token, the UBM transmits the data packet to the host PC. If the data packet is received without an error by the host PC, then an ACK handshake is returned. The UBM will then toggle the TOGGLE bit, set the NACK bit to a 1 and assert the endpoint interrupt.
3. The MCU services the interrupt and prepares to send the next data packet to the host PC.
4. If the NACK bit is set to a 1 when the in token packet is received, the UBM simply returns a NAK handshake to the host PC. If the STALL bit is set to a 1 when the in token packet is received, the UBM simply returns a STALL handshake to the host PC. If no handshake packet is received from the host PC, then the UBM prepares to retransmit the same data packet again.
5. MCU continues to send data packets until all data has been sent to the host PC.

Status Stage Transaction:

1. For output endpoint0, the MCU sets the TOGGLE bit to 1, then clears the NACK bit to a 0 to enable the data packet to be sent to the host PC. Note that for a stage transaction a null data packet with the DATA1 PID is sent to the host PC.
2. The host PC sends an out token packet addressed to out endpoint0. If the data packet is received without an error the UBM updates the data count value, toggles to the TOGGLE bit, sets the NACK bit to a 1, returns an ACK handshake to the host PC, and asserts the endpoint interrupt.

3. The MCU services the interrupt. If the status transaction completed successfully, then the MCU clears the interrupt and clears the NACK bit.

4. If the NACK bit is set to a 1 when the in token packet is received, the UBM simply returns a NAK handshake to the host PC. If the STALL bit is set to a 1 when the in token packet is received, the UBM simply returns a STALL handshake to the host PC. If a CRC or bit stuff error occurs when the data packet is received, then no handshake is returned to the host PC.

2.2.7.2 Interrupt Transfers

The TAS1020 supports interrupt data transfers both to and from the host PC. Devices that need to send or receive a small amount of data with a specified service period should use the interrupt transfer type. In endpoints 1 through 7 and out endpoints 1 through 7 can all be configured as interrupt endpoints.

2.2.7.2.1 Interrupt Out Transaction

The steps to be followed for a interrupt out transaction are as follows:

1. MCU initializes one of the out endpoints as an out interrupt endpoint by programming the appropriate USB endpoint configuration block. This entails programming the buffer size and buffer base address, selecting the buffer mode, enabling the endpoint interrupt, initializing the toggle bit, enabling the endpoint and clearing the NACK bit.

2. The host PC sends an out token packet followed by a data packet addressed to the out endpoint. If the data is received without an error then the UBM writes the data to the endpoint buffer, updates the data count value, toggles the toggle bit, sets the NACK bit to a 1, returns an ACK handshake to the host PC, and asserts the endpoint interrupt.

3. The MCU services the interrupt and reads the data packet from the buffer. To read the data packet, the MCU first must obtain the data count value. After reading the data packet, the MCU clears the interrupt and clears the NACK bit to allow the reception of the next data packet from the host PC.

4. If the NACK bit is set to a 1 when the data packet is received, the UBM simply returns a NACK handshake to the host PC. If the STALL bit is set to a 1 when the in token packet is received, the UBM simply returns a STALL handshake to the host PC. If a CRC or bit stuff error occurs when the data packet is received, then no handshake is returned to the host PC.

   NOTE: In double buffer mode for interrupt out transactions, the UBM selects between the X and Y buffer based on the value of the toggle bit. If the toggle bit is a 0, the UBM writes the data packet to the X buffer. If the toggle bit is a 1, the UBM writes the data packet to the Y buffer. When a data packet is received, the MCU determines which buffer contains the data packet by reading the toggle bit. However, when using double buffer mode, the possibility exists for data packets to be received and written to both the X and Y buffer before the MCU responds to the endpoint interrupt. In this case, by simply using the toggle bit to determine which buffer contains the data packet does not work. Hence, in double buffer mode, the MCU reads the X buffer NACK bit, the Y buffer NACK bit, and the toggle bit to determine the status of the buffers.

2.2.7.2.2 Interrupt In Transaction

The steps to be followed for an interrupt in transaction are as follows:

1. MCU initializes one of the in endpoints as an in interrupt endpoint by programming the appropriate USB endpoint configuration block. This entails programming the buffer size and buffer base address, selecting the buffer mode, enabling the endpoint interrupt, initializing the toggle bit, enabling the endpoint, and setting the NACK bit.

2. The data packet to be sent to the host PC is written to the buffer by the MCU. The MCU also updates the data count value then clears the NACK bit to 0 to enable the data packet to be sent to the host PC.

3. The host PC sends an in token packet addressed to the in endpoint. After receiving the in token, the UBM transmits the data packet to the host PC. If the data packet is received without errors by the host PC, then an ACK handshake is returned. The UBM then toggles the toggle bit, sets the NACK bit to a 1, and asserts the endpoint interrupt.

4. The MCU services the interrupt and prepares to send the next data packet to the host PC.

5. If the NACK bit is set to a 1 when the in token packet is received, the UBM simply returns a NACK handshake to the host PC. If the STALL bit is set to a 1 when the in token packet is received, the UBM simply returns a STALL handshake to the host PC. If no handshake packet is received from the host PC, then the UBM prepares to retransmit the same data packet again.

   NOTE: In double buffer mode for interrupts in transactions, the UBM selects between the X and Y buffer based on the value of the toggle bit. If the toggle bit is a 0, the UBM will read the data packet to the X buffer. If the toggle bit is a 1, the UBM will read the data packet to the Y buffer.

2.2.7.3 Bulk Transfers

The TAS1020 supports bulk data transfers both to and from the host PC. Devices that need to send or receive a large amount of data without a suitable bandwidth should use the bulk transfer type. In endpoints 1 through 7 and out endpoints 1 through 7 can be configured as bulk endpoints. TAS1020 supports single and double buffering for bulk transfers.

2.2.7.3.1 Bulk Out Transaction Using MCU

The steps to be followed for a bulk out transaction are as follows:

1. MCU initializes one of the out endpoints as an out bulk endpoint by programming the appropriate USB endpoint configuration block. This entails programming the buffer size and buffer base address, selecting the buffer mode, enabling the endpoint interrupt, initializing the toggle bit, enabling the endpoint, and clearing the NACK bit.

2. The host PC sends an out token packet followed by a data packet addressed to the out endpoint. If the data is received without an error then the UBM writes the data to the endpoint buffer, updates the data count value, toggles the toggle bit, sets the NACK bit to a 1, returns an ACK handshake to the host PC, and asserts the endpoint interrupt.

3. The MCU services the interrupt and reads the data packet from the buffer. To read the data packet, the MCU first needs to obtain the data count value. After reading the data packet, the MCU clears the interrupt and clears the NACK bit to allow the reception of the next data packet from the host PC.

4. If the NACK bit is set to a 1 when the data packet is received, the UBM simply returns a NACK handshake to the host PC. If the STALL bit is set to a 1 when the data packet is received, the UBM simply returns a STALL handshake to the host PC. If a CRC or bit stuff error occurs when the data packet is received, then no handshake is returned to the host PC.

   NOTE: In double buffer mode for bulk out transactions, the UBM selects between the X and Y buffer based on the value of the toggle bit. If the toggle bit is a 0, the UBM writes the data packet to the X buffer. If the toggle bit is a 1, the UBM writes the data packet to the Y buffer. When a data packet is received, the MCU determines which buffer contains the data packet by reading the toggle bit. However, when using double buffer mode, data packets may be received and written to both the X and Y buffer before the MCU responds to the endpoint interrupt. In this case, simply using the toggle bit to determine which buffer contains the data packet does not work. Hence, in double buffer mode, the MCU reads the X buffer NACK bit, the Y buffer NACK bit, and the toggle bit to determine the status of the buffers.

2.2.7.3.2 Bulk In Transaction Using MCU

The steps to be followed for a bulk in transaction are as follows:

1. MCU initializes one of the in endpoints as an in bulk endpoint by programming the appropriate USB endpoint configuration block. This entails programming the buffer size and buffer base address, selecting the buffer mode, enabling the endpoint interrupt, initializing the toggle bit, enabling the endpoint and setting the NACK bit.
2. The data packet to be sent to the host PC is written to the buffer by the MCU. The MCU also updates the data count value then clears the NACK bit to a 0 to enable the data packet to be sent to the host PC.
3. The host PC sends an in token packet addressed to the in endpoint. After receiving the in token, the UBM transmits the data packet to the host PC. If the data packet is received without errors by the host PC, then an ACK handshake is returned. The UBM then toggles the toggle bit, sets the NACK bit to a 1, and asserts the endpoint interrupt.
4. The MCU services the interrupt and prepares to send the next data packet to the host PC.
5. If the NACK bit is set to a 1 when the in token packet is received, the UBM simply returns a NAK handshake to the host PC. If the STALL bit is set to a 1 when the in token packet is received, the UBM simply returns a STALL handshake to the host PC. If no handshake packet is received from the host PC, then the UBM prepares to retransmit the same data packet again.

   NOTE: In double buffer mode for bulk in transactions, the UBM selects between the X and Y buffer based on the value of the toggle bit. If the toggle bit is a 0, the UBM reads the data packet to the X buffer. If the toggle bit is a 1, the UBM reads the data packet to the Y buffer.

2.2.7.3.3 Bulk Out Transaction Through DMA

This transaction is used for mass storage class USB applications to move bulk data through DMA to C-Port. C-Port can be programmed in General Purpose mode or AC97 mode. When in general purpose mode, SYNC is disabled when no data is there to send out. In the AC97 mode, Tag fields are de-asserted when no valid data is in the buffer to be sent out.

A sample for the Bulk Out transaction is the number of bytes to be transferred in a codec frame.

Two modes of transfer are possible. The first one is the software handshake mode using MCU and external controller (DSP) and the second one is no handshake mode where MCU intervention is less for higher rates of transfer.

Software Handshake Using MCU and External Controller (DSP)

The following steps explain the operation:

1. hsken (Handshake enable) bit in DMA channel control definition register = 1.
2. UBM after receiving BULK packet, it puts it into the buffer, sets the NACK bit and the data count in the data count register, and generates an interrupt to the MCU.
3. MCU after receiving interrupt can decide to use DMA to send the packet to C-Port.
4. MCU decides how many BULK OUT packets need to be handled by DMA without MCU intervention. MCU writes this value (16-bit) to DMAPCNT1, MDMAPCNT0 registers.
5. MCU configures the C-Port to be in AC97/GP modes. If C-Port is used as GP mode, then MCU must set the CptBlk bit in GLOBCTRL register.
6. Then, MCU enables the DMA.
7. DMA then transfers data through the C-Port and once it finishes the number of packets programmed by MCU, it disables itself and then generates DMA0 or DMA1 interrupt to MCU.
8. MCU after receiving DMA interrupt will know that the DMAPCNT packets are sent out to the C-Port and takes control again.

9. Handshaking between MCU and DSP can be done in software at a higher level, depending on the application.

No Handshake Mode

If the MCU intervention in the above handshake mode limits the bandwidth on the BULK out endpoint, no Handshake mode can be used.

1. hsken (Handshake enable) bit in DMA channel control definition register = 0.
2. MCU configures the C-Port to be in AC97/GP modes. If C-Port is used as GP mode, then MCU must set the CptBlk bit in GLOBCTRL register.
3. MCU configures the endpoint configuration registers and enables the DMA.
4. Now handshaking is only between DMA and UBM. DMA moves data once it is filled by UBM. This continues until MCU disables DMA.

2.2.7.3.4 Bulk In Transaction Through DMA

TAS1020 does not support BULK IN through DMA.

2.2.7.4 ISO OUT/IN. USB to/from C-Port Through DMA

2.2.7.4.1 ISO Out

Here the ISO packets from the host are received by UBM and put into the buffer. DMA collects the data from the buffer and streams it through the C-Port to a codec or a DSP. DMA looks at the UBM writer pointers to make a decision about the availability of ISO data in the buffer. C-Port can be programmed into different modes based on the application and the codecs being used. Here MCU must program the endpoint configuration registers, DMA and C-Port. DMA also updates the buffer content registers which is the difference of UBM writer pointer and DMA read pointer after the end of every USB frame.

- **UBM Operation**
  - Reads endpoint configuration byte. Checks if endpoint is enabled. If not enabled, ignores the ISO packet.
  - If it is ISO endpoint, checks the endpoint number with the endpoint number in the DMA channel 0/1 definition register. If either of the DMA channels is used, UBM behaves as described below. If not, the ISO traffic is directed to MCU, which the MCU is able to handle it in a different way.
  - Reads OEPBBAXn. This is the starting address for the circular buffer being used for the endpoint. For the first ISO OUT packet UBM initializes the write pointer to OEPBBAXn. UBM locally stores this starting address as MCU can relocate the buffer anytime. Each time an ISO OUT packet comes, UBM compares this local buffer starting address with OEPBBAXn. If this is not same, UBM re-initializes the write pointer to OEPBBAXn.
  - Reads OEPSIZn. This is the circular buffer size for the ISO endpoint. This is used to wrap-around the UBM write pointer to the starting address of the buffer when it reaches the end of the buffer.
  - Buffer end address = buffer starting address + buffer size
  - UBM proceeds with reading the data from the SIE and storing the data into the circular buffer in the shared RAM. The 11-bit write pointer is incremented after receiving each byte. Once the buffer write pointer reaches the buffer end address, UBM rewinds the write pointer to the starting buffer address.
  - At the end of transaction, UBM updates the sample count in the OEPDCNTX or OEPDCNTY register based on the LSB of frame counter.
  - NAK bit not used for ISO to DMA.

– If UBM write pointer overtakes the DMA read pointer, UBM sets the buffer overflow (OVF) bit in the ISO endpoint configuration register.

– UBM write pointer is reset to the buffer starting address once DMA is disabled.

DMA Operation

- DMA must be enabled before getting any ISO packets from the Host. Typically, MCU enables the DMA before playing audio to the device.
  – After DMA is enabled, it reads the DMA channel control register to get the endpoint number and direction.
  – Based on the endpoint number, it reads the endpoint configuration register. If it is ISO endpoint, it works as follows.
  – DMA reads the OEPBBAXn, OEPBSIZn registers to determine the buffer starting address and buffer size.
  – DMA reads the C-port registers to understand the C-port mode.
  – DMA reads the DMA time slot assignment registers.
  – DMA, for the first time, keeps monitoring the OEPDCNTXn or OEPDCNTYn register to see if an ISO OUT packet is received. Once a new ISO OUT packet is received, DMA waits for next SOF/PSOF signaling and goes to the next step.
  – DMA starts with the read pointer equal to the buffer starting address. Once an ISO OUT packet is received, the UBM pointer is incremented very fast as the ISO OUT is packet is received at 12 Mb/sec.
  – DMA starts reading the samples out of the buffer and sends to the C-port till it catches up with the UBM pointer.
  – Each time DMA receives SOF/PSOF signaling, it completes streaming the current sample, and calculates the buffer content based on UBM write pointer and DMA read pointer. Buffer content is in number of bytes. This buffer size is written into DmaBCnt0/DMABCnt1 registers.

  NOTE: In AC97 mode, if buffer under-run happens, DMA should send this information to the C-port to clear the tag bits for the corresponding slots to indicate that the data in those codec frames is invalid. For I2S, AIC, GP modes, it must send zeros. C-port logic sends zeros if DMA does not stream anything to the C-port for a codec frame.

2.2.7.4.2 ISO IN

The serial data coming from the codec/DSP is received by the C-Port. DMA moves the bytes from C-Port holding registers into the buffer and updates the endpoint data count registers. UBM packetizes the data and sends the packets to host when it receives an ISO IN request. MCU has to program the endpoint configuration registers, DMA and C-Port. The following section explains how DMA and UBM works.

UBM Operation

- After ISO IN token is received, UBM reads the endpoint configuration register if the endpoint is enabled. If not enabled, it ignores the IN request.
- UBM reads the IEPBBAXn, IEPBSIZXn registers to get the circular buffer starting address and buffer size and calculates the buffer maximum address.
- UBM reads IEPDCNTXn or IEPDCNTYn depending on the LSB of the frame counter register to determine how many samples (bytes) need to be transferred to the host.
- UBM initializes the read pointer to one of the 11-bit pointers from DMA(depending on the LSB of frame counter) to start reading the samples out of the buffer. If the address reaches buffer maximum address UBM rewinds the read pointer to the buffer starting address.

- After the last sample is moved to the SIE transmit holding registers, UBM clears the data count register for the IEPDCNTXn or IEPDCTNYn registers.

DMA Operation

- In general, DMA must packetize the serial data coming from the C-port for 1 ms before UBM sends it to the host when an ISO IN token is received.
- If DMA draws packet boundaries when it receives SOF/PSOF, then it might be in the middle of receiving a sample when an IN token is received. Hence, DMA must finish packetizing way before the SOF/PSOF. For the worst case, at 8kHz, there are 8 codec frames in 1 USB frame. So, to make room for one 8-kHz codec frame, the *IN packet switch point* should occur at least 12000% 8 = 1500 USB clocks before SOF. Taking the variation in USB clocks per SOF into account, the IN packet switch event is set to 1511 USB_clocks before SOF/PSOF.
- After DMA is enabled, it reads all the DMA control registers, DMA time slot registers, the endpoint base address, and size registers.
- DMA write pointer is initialized to the buffer start address.
- As DMA packetizes continuously regardless of host sending ISO IN tokens, DMA must pass pointers to UBM to clearly tell the UBM about the packet boundaries for each USB frame.
- Two data count registers are required in the IN direction.
- DMA waits for *IN packet switch event* to start its operation. At this time DMA stores its write pointer to convey the buffer start address for this packet to UBM.
- DMA has to pass two pointers based on the LSB of the frame counter.
- DMA stores the samples into the buffer, incrementing its 11-bit write pointer.
- Once it receives next packet switch event, the DMA puts the last sample into the buffer and updates the IEPDCNTXn or IEPDCNTYn data count register with the number of samples received based on the LSB of the frame counter.

2.2.8 Microcontroller Unit

The 8052 core used in the TAS1020 is based on the industry standard 8052 MCU and is software compatible with the 8052, 8032, 80C52, 80C53, and 87C52 MCUs. Therefore, refer to a standard 8052 data manual for more details, if needed.

2.2.9 External MCU Mode Operation

The external MCU mode of operation is provided for firmware development using an in-circuit emulator (ICE). In the external MCU mode, the internal 8052 MCU core of the TAS1020 is disabled. Also in the external MCU mode, the GPIO ports are used for the external MCU data, address, and control signals. Refer to section 1.7, *Terminal Functions – External MCU Mode*, for details. In this mode, the external MCU or ICE is able to access the memory mapped IO registers, the USB configuration blocks and the USB buffer space. Refer to section 1.8, *Device Operation Modes*, for information regarding the various modes of operation.

Texas Instruments has developed the TAS1020 evaluation module (EVM) to allow customers to develop application firmware and to evaluate device performance. The EVM board provides a 40-pin dip socket for an ICE and headers to allow expansion of the system in a variety of ways.

2.2.10 Interrupt Logic

The 8052 MCU core used in the TAS1020 supports all the standard interrupt sources. The five standard MCU interrupt sources are timer 0, timer 1, serial port, external 1 (INT1), and external 0 (INT0).

All of the additional interrupt sources within the TAS1020 device are read together to generate the INT0 signal to the MCU. Refer to the interrupt vector register for more details on the other TAS1020 interrupt sources.

The other interrupt sources are: the eight USB in end-points, the eight USB out end-points, USB function reset, USB function suspend, USB function resume, USB start-of-frame, USB pseudo start-of-frame, USB setup stage transaction, USB setup stage transaction over-write, codec port interface transmit data register empty, codec port interface receive data register full, I^2C interface transmit data register empty, I^2C interface receive data register full, and the external interrupt input.

The interrupts for the USB in end-points and USB out end-points can be masked. An interrupt for a particular end-point occurs at the end of a successful transaction to that end-point. A status bit for each in and out end-point also exists. However, these status bits are read only, and therefore, these bits are intended to be used for diagnostic purposes only. After a successful transaction to an end-point, both the interrupt and status bit for an end-point are asserted until the interrupt is cleared by the MCU.

The USB function reset, USB function suspend, USB function resume, USB start-of-frame, USB pseudo start-of-frame, USB setup stage transaction, and USB setup stage transaction over-write interrupts can all be masked. A status bit for each of these interrupts also exists. Refer to the USB interrupt mask register and the USB status register for more details. Note that the status bits for these interrupts are read only. For these interrupts, both the interrupt and status bit are asserted until the interrupt is cleared by the MCU.

The codec port interface transmit data register empty, codec port interface receive data register full, I^2C Interface transmit data register empty, and I^2C interface receive data register full interrupts can all be masked. A status bit for each of these interrupts also exists. Note that the status bits for these interrupts are read only. However, for these interrupts, the status bits are not cleared automatically when the interrupt is cleared by the MCU. Refer to the codec port interface control/status register and the I^2C interface control/status register for more details.

The external interrupt input ($\overline{\text{XINT}}$) is also read together with the on-chip interrupt sources. An enable bit exists for this interrupt in the global control register. This interrupt does not have a status bit.

2.2.11 DMA Controller

The TAS1020 provides two DMA channels for transferring data between the USB end-point buffers and the codec port interface. The DMA channels are provided to support the streaming of data for USB isochronous end-points only. Each DMA channel can be programmed to service only one isochronous end-point. The end-point number and direction are programmable using the DMA channel control register provided for each of the DMA channels.

A DMA channel can also be used to send bulk endpoint data to the C-port for higher throughput. For more details refer to the *Bulk Out Transaction Using DMA* section 2.2.7.3.3.

The codec port interface time slots to be serviced by a particular DMA channel must also be programmed. For example, an AC'97 mode stereo speaker application uses time slots 3 and 4 for audio playback. Therefore, the DMA channel used to move the audio data to the codec port interface must set time slot assignment bits 3 and 4 to a 1. Each DMA channel is capable of being programmed to transfer data for time slots 0 through 13 using the two DMA channel time slot assignment registers provided for each DMA channel.

The number of bytes to be transferred for each time slot is also programmable. The number of bytes used must be set based on the desired audio data format.

2.2.12 Codec Port Interface

The codec port interface is a configurable serial interface used to transfer data between the TAS1020 IC and a codec device. The serial protocol and formats supported include AC '97 1.0, AC '97 2.0, and several I^2S modes. In addition, a general-purpose mode is provided that can be configured to various user defined serial interface formats. The TAS1020 also provides a bulk mode where DMA can send bulk data to C-port directly.

Configuration of the interface is accomplished using the four codec port interface configuration registers: CPTCNF1, CPTCNF2, CPTCNF3, and CPTCNF4. In I^2S mode 5, CPTRXCNF2, CPTRXCNF3, and CPTRXCNF4 are used to configure the C-port in receive direction. Refer to section A.5.4 for more details on these registers. The serial interface is a time division multiplexed (TDM) time slot *based* scheme. The basic serial format is programmed by setting the number of time slots per codec frame and the number of serial clock cycles (or bits) per time slot. The interface in

all modes is bidirectional and full duplex. For some modes, both audio data and command/status data are transferred via the serial interface. The sources of the USB end-point data buffer are the transmit data and destination of the receive data for all audio data time slots. Transfer of the audio data packets to/from the USB end-point data buffers and the codec port interface is controlled by one or more of the DMA channels. Remember that each DMA channel can be assigned to one USB isochronous end-point. The source and/or the destination of the command/status address and data values is the MCU.

The features of the codec port interface that can be configured are:

- The mode of operation
- The number of time slots per codec frame
- The number of serial clock cycles for slot 0
- The number of serial clock cycles for all slots other than slot 0
- The number of valid data bits per audio data time slot
- The time slots to be used for command/status address and data
- The serial clock (CSCLK) frequency in relation to the codec master clock (MCLK) frequency
- The source of the serial clock signal (internally generated or an input from the codec device)
- The source of the codec master clock signal used to generate the internal serial clock signal (internally generated by the ACG or an input to the TAS1020 device)
- The polarity, duration, and direction of the codec frame sync signal
- The relationship between the codec frame sync signal and the serial clock signal
- The relationship between the codec frame sync signal and the serial data signals
- The relationship between the serial clock signal and the serial data signals
- The use of zero padding or a 3-state level for unused time slots and/or bits
- The byte ordering to be used

2.2.12.1 Audio Codec (AC) ’97 1.0 Mode of Operation

In AC ’97 1.0 mode, the codec port interface can be configured as an AC link serial interface to the AC ’97 codec device. Refer to the Audio codec ’97 Specification Revision 1.03 for additional information. The AC link serial interface is a time division multiplexed (TDM) *slot based* serial interface that is used to transfer both audio data and command/status data between the TAS1020 IC and the codec device.

Table 2–3. Terminal Assignments for Codec Port Interface AC ’97 1.0 Mode

TERMINAL		AC ’97 VERSION 1.0 MODE 2	
NO.	NAME		
35	CSYNC	SYNC	O
37	CSCLK	BIT_CLK	I
38	CDATO	SD_OUT	O
36	CDATI	SD_IN	I
34	$\overline{\text{CRESET}}$	$\overline{\text{RESET}}$	O
32	CSCHNE	NC	O

In this mode, the codec port interface is configured as a bidirectional full duplex serial interface with a fixed rate of 48 kHz. Each 48-kHz frame is divided into 13 time slots, with the use of each time slot predefined by the Audio codec ’97 Specification. Each time slot is 20 serial clock cycles in length except for time slot 0, which is only 16 serial clock cycles. The serial clock, which is referred to as the BIT_CLK for AC ’97 modes, is set to 12.288 MHz. Based on the length of each slot, there is a total of 256 serial clock cycles per frame at a frequency of 12.288 MHz. As a result the frame frequency is 48 kHz. For the AC ’97 modes, the BIT_CLK is input to the TAS1020 device from the codec. The BIT_CLK is generated by the codec from the master clock (MCLK) input. The codec MCLK input, which can be generated by the TAS1020 device, must be a frequency of 24.576 MHz. The start of each 48-kHz frame is

synchronized to the rising edge of the SYNC signal, which is an output of the TAS1020 device. The SYNC signal is driven high each frame for the duration of slot 0. See Figure 2–2 for details on connecting the TAS1020 to a codec device in this mode.

Figure 2–2. Connection of the TAS1020 to an AC ’97 Codec

The AC link protocol defines slot 0 as a special slot called the *tag slot* and defines slots 1 through 12 as *data slots*. Slot 1 and slot 2 are used to transfer command and status information between the TAS1020 device and the codec. Slot 1 and slot 2 of the outgoing serial data stream are defined as the command address and command data slots, respectively. These slots are used for writing to the control registers in the codec. Slot 1 and slot 2 of the incoming serial data stream are defined as the status address and status data slots, respectively. These slots are used for reading from the control registers in the codec.

Unused or reserved time slots and unused bit locations within a valid time slot are filled with zeros. Since each data time slot is 20 bits in length, the protocol supports 8-bit, 16-bit, 18-bit, or 20-bit data transfers.

2.2.12.2 Audio Codec (AC) ’97 2.0 Mode of Operation

The basic serial protocol for the AC ’97 2.0 mode is the same as the AC ’97 1.0 mode. The AC ’97 2.0 mode, however, offers some additional features. In this mode, the TAS1020 provides support for multiple codec devices and also on-demand sampling. The TAS1020 can connect directly to two AC ’97 codecs as shown in Figure 2–3. Note that if only one codec is used, then the SD_IN2 input (pin 40) should be tied to DVSS.

Table 2–4. Terminal Assignments for Codec Port Interface AC ’97 2.0 Mode

TERMINAL		AC ’97 VERSION 2.0 MODE 3	
NO.	NAME		
35	CSYNC	SYNC	O
37	CSCLK	BIT_CLK	I
38	CDATO	SD_OUT	O
36	CDATI	SD_IN1	I
34	$\overline{\text{CRESET}}$	$\overline{\text{RESET}}$	O
32	CSCHNE	SD_IN2	I

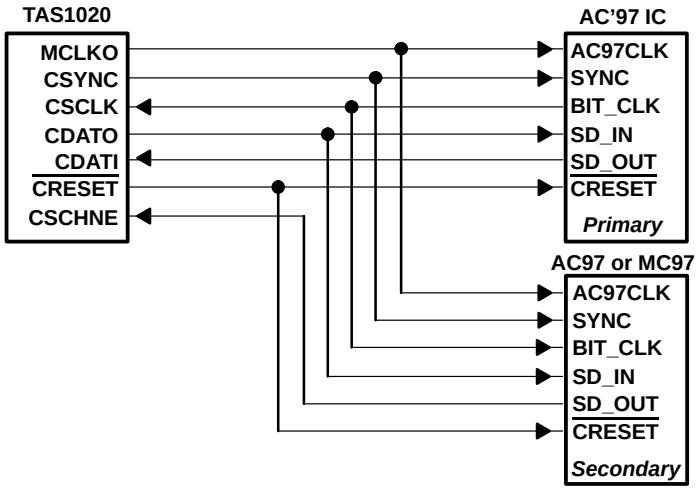


Figure 2–3. Connection of the TAS1020 to Multiple AC '97 Codecs

2.2.12.3 Inter-IC Sound (I^2S) Modes of Operation

The TAS1020 offers two I^2S modes of operation. However, the serial format is the same for these two I^2S modes. The difference in the I^2S modes is simply the number of serial data outputs and/or serial data inputs supported. For instance, in codec port interface mode 4, there is one serial data output (SDOUT1) and two serial data inputs (SDIN1, SDIN2). Hence, mode 4 can be used to connect the TAS1020 device to a codec with one stereo DAC and two ADCs. Table 2–5 shows the TAS1020 codec terminal assignments and the respective signal names for each of the I^2S modes.

Table 2–5. Terminal Assignments for Codec Port Interface I^2S Modes

TERMINAL		I^2S MODE 4		I^2S MODE 5	
NO.	NAME				
35	CSYNC	LRCK	O	LRCK1	O
37	CSCLK	SCLK	O	SCLK1	O
38	CDATO	SDOUT1	O	SDOUT1	O
36	CDATI	SDIN1	I	SDIN2	I
34	$\overline{\text{CRESET}}$	$\overline{\text{CRESET}}$	O	SCLK2	O
32	CSCHNE	SDIN2	I	LRCK2	O

In all I^2S modes, the codec port interface is configured as a bidirectional full duplex serial interface with two time slots per frame. The frame sync signal is the left/right clock (LRCK) signal. Time slot 0 is used for the left channel audio data, and time slot 1 is used for the right channel audio data. Both time slots must be set to 32 serial clock (SCLK) cycles in length giving an SCLK-to-LRCK ratio of 64. The serial clock frequency is based on the audio sample rate and the codec master clock (MCLK) frequency. For example, when using an audio sample rate (FS) of 48 kHz and an MCLK frequency of 12.288 MHz (256xFS), the SCLK frequency must be set to 3.072 MHz (64xFS). Note that the codec frame sync, the audio sample rate (FS), and the LRCK are all synonymous.

The LRCK signal has a 50% duty cycle. The LRCK signal is low for the left channel time slot and is high for the right channel time slot. In addition, the LRCK signal is synchronous to the falling edge of the SCLK. Serial data is shifted out on the falling edge of SCLK and shifted in on the rising edge of SCLK. There is a one SCLK cycle delay from the edge of the LRCK before the most significant bit of the data is shifted out for both the left channel and right channel.

For the I^2S modes of the codec port interface, there is a 24-bit transmit and 24-bit receive shift register for each SDOUT and SDIN signal, respectively. As a result, the interface can actually support 16-bit, 18-bit, 20-bit or 24-bit transfers. The interface pads the unused bits automatically with zeros.

The I2S protocol does not provide for command/status data transfers. Therefore, when using the TAS1020 device with a codec that uses an I2S serial interface for audio data transfers, the TAS1020 I2C serial interface can be used for codec command/status data transfers.

In addition, the TAS1020 codec port interface is very flexible. As a result, many variations of the serial interface protocol can be configured including an SCLK-to-LRCK ratio of 32.

2.2.12.4 *Mapping of DMA Time Slots to Codec Port Interface Time Slots for I2S Modes*

The I2S serial data format requires two time slots (left channel and right channel) for each serial data output or input. As discussed in the previous section. Each of the serial data outputs and/or inputs has a unique left channel time slot (slot number 0) and right channel time slot (slot number 1). For the I2S modes of operation, the DMA channel time slot assignments must be mapped to the different left channel and right channel time slots for the serial data outputs and inputs. Each DMA channel has fourteen time slot bits, which are time slot assignment bits 0 through 13. Table 2–6 and 2–7 show the codec port interface time slot numbers and the corresponding time slot numbers for the DMA channels.

As an example, suppose that Codec port interface mode 4 is to be used with one serial data output and two serial data inputs. The DMA channel to be programmed to support the serial data output must have time slot assignment bits 0 and 1 set to 1. The DMA channel to be programmed to support the serial data input must have time slot assignment bits 0 and 2 set to 1.

Table 2–6. SLOT Assignments for Codec Port Interface I2S Mode 4

SERIAL DATA	Codec PORT INTERFACE TIME SLOT NUMBER		DMA CHANNELS(s) TIME SLOT NUMBER	
	LEFT CHANNEL	RIGHT CHANNEL	LEFT CHANNEL	RIGHT CHANNEL
SDOUT1	0	1	0	1
SDIN1	0	1	0	2
SDIN2	0	1	1	3

Table 2–7. SLOT Assignments for Codec Port Interface I2S Mode 5

SERIAL DATA	Codec PORT INTERFACE TIME SLOT NUMBER		DMA CHANNELS(s) TIME SLOT NUMBER	
	LEFT CHANNEL	RIGHT CHANNEL	LEFT CHANNEL	RIGHT CHANNEL
SDOUT1	0	1	0	1
SDIN2	0	1	0	1

2.2.12.5 General-Purpose Mode of Operation

In the general-purpose mode the codec port interface can be configured to various user defined serial interface formats using the pin assignments shown in Table 2–8. This mode gives the user the flexibility to configure the TAS1020 to connect to various codecs and DSPs that do not use a standard serial interface format.

Table 2–8. Terminal Assignments for Codec Port Interface General-Purpose Mode

TERMINAL		GP MODE 0	
NO.	NAME		
35	CSYNC	CSYNC	I/O
37	CSCLK	CSCLK	I/O
38	CDATO	CDAT0	O
36	CDATI	CDAT1	I
34	$\overline{\text{CRESET}}$	$\overline{\text{CRESET}}$	O
32	CSCHNE	NC	O

2.2.12.6 AIC Mode

AIC mode is used to connect TAS1020 to a voice codec which has AIC interface as described in TI's TLV320AIC10 data sheet.

2.2.12.7 Bulk Mode

TAS1020 supports bulk data transfer through the codec port using one of the two available DMA channels. The codec port needs to be configured in AC97 or general purpose modes for the operation. This mode is selected by writing a 1 to CPTBLK bit of the CPTCNF4 register. When the codec port is running in general purpose bulk mode, CSYNC is active only when valid data is present in the current frame. In 'AC97 bulk mode, CSYNC is active all the time, but the tag bits in slot 0 indicate the valid data.

2.2.13 I^2C Interface

The TAS1020 has a bidirectional two-wire serial interface that can be used to access other ICs. This serial interface is compatible with the I^2C (Inter IC) bus protocol and supports both 100-kbps and 400-kbps data transfer rates. The TAS1020 is a master only device that does not support a multimaster bus environment (no bus arbitration) or wait state insertion. Hence this interface can be used to access I^2C slave devices including EEPROMs and codecs. For example, if the application program code is stored in an EEPROM on the PCB, then the MCU will download the code from the EEPROM to the TAS1020 on-chip RAM using the I^2C interface. Another example is the control of a codec device that uses an I^2S interface for audio data transfers and an I^2C interface for control register read/write access.

2.2.13.1 Data Transfers

The two-wire serial interface uses the serial clock signal, SCL, and the serial data signal, SDA. As stated above, the TAS1020 is a master only device, and therefore, the SCL signal is an output only. The SDA signal is a bidirectional signal that uses an open-drain output to allow the TAS1020 to be wire-ORed with other devices that use open-drain or open-collector outputs.

All read and write data transfers on the serial bus are initiated by a master device. The master device is also responsible for generating the clock signal used for all data transfers. The data is transferred on the bus serially one bit at a time. However, the protocol requires that the address and data be transferred in byte (8-bit) format with the most-significant bit (MSB) transferred first. In addition, each byte transferred on the bus is acknowledged by the receiving device with an acknowledge bit. Each transfer operation begins with the master device driving a start condition on the bus and ends with the master device driving a stop condition on the bus.

The timing relationship between the SCL and SDA signals for each bit transferred on the bus is shown in Figure 3-7. As shown, the SDA signal must be stable while the SCL signal is high, which also means that the SDA signal can only change states while the SCL signal is low.

The timing relationship between the SCL and SDA signals for the start and stop conditions is shown in Figure 3-8. As shown, the start condition is defined as a high-to-low transition of the SDA signal while the SCL signal is high. Also as shown, the stop condition is defined as a low-to-high transition of the SDA signal while the SCL signal is high.

When the TAS1020 is the device receiving data information, the TAS1020 acknowledges each byte received by driving the SDA signal low during the acknowledge SCL period. During the acknowledge SCL period, the slave device must stop driving the SDA signal. If the TAS1020 is unable to receive a byte, the SDA signal is not driven low and is pulled high external to the TAS1020 device. A high during the SCL period indicates a not-acknowledge to the slave device. The acknowledge timing is shown in Figure 3-9.

Read and write data transfers by the TAS1020 device can be done using single byte or multiple byte data transfers. Therefore, the actual transfer type used depends on the protocol required by the I^2C slave device being accessed.

2.2.13.2 Single Byte Write

As shown is Figure 2-4, a single byte data write transfer begins with the master device transmitting a start condition followed by the I2C device address and the read/write bit. The read/write bit determines the direction of the data transfer. For a write data transfer, the read/write bit must be a 0. After receiving the correct I2C device address and the read/write bit, the I2C slave device responds with an acknowledge bit. Next, the TAS1020 transmits the address byte or bytes corresponding to the I2C slave device internal memory address being accessed. After receiving the address byte, the I2C slave device again responds with an acknowledge bit. Next, the TAS1020 device transmits the data byte to be written to the memory address being accessed. After receiving the data byte, the I2C slave device again responds with an acknowledge bit. Finally, the TAS1020 device transmits a stop condition to complete the single byte data write transfer.

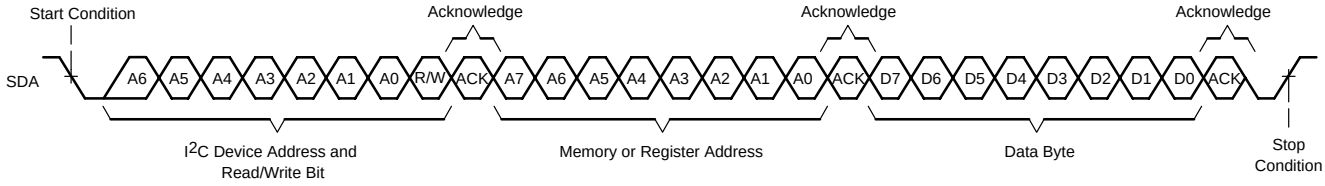


Figure 2–4. Single Byte Write Transfer

2.2.13.3 Multiple Byte Write

A multiple byte data write transfer is identical to a single byte data write transfer except that multiple data bytes are transmitted by the TAS1020 device to the I2C slave device as shown in Figure 2-5. After receiving each data byte, the I2C slave device responds with an acknowledge bit.

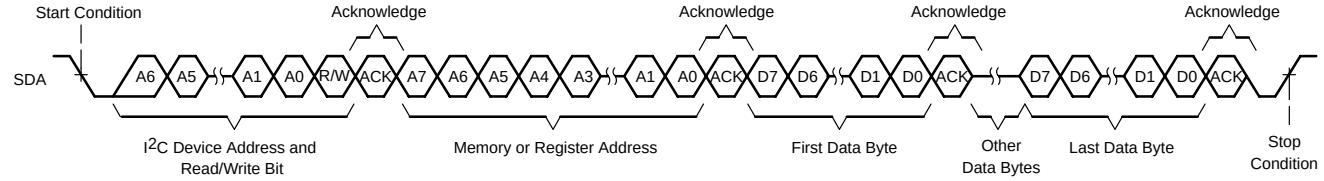


Figure 2–5. Multiple Byte Write Transfer

2.2.13.4 Single Byte Read

As shown in Figure 2-6, a single byte data read transfer begins with the TAS1020 device transmitting a start condition followed by the I^{2}C device address and the read/write bit. For the data read transfer, both a write followed by a read are actually performed. Initially, a write is performed to transfer the address byte or bytes of the internal memory address to be read. As a result, the read/write bit must be a 0. After receiving the I^{2}C device address and the read/write bit, the I^{2}C slave device responds with an acknowledge bit. Also, after sending the internal memory address byte or bytes, the TAS1020 device transmits another start condition followed by the I^{2}C slave device address and the read/write bit again. This time the read/write bit is a 1 indicating a read transfer. After receiving the I^{2}C device address and the read/write bit the I^{2}C slave again responds with an acknowledge bit. Next, the I^{2}C slave device transmits the data byte from the memory address being read. After receiving the data byte, the TAS1020 device transmits a not-acknowledge followed by a stop condition to complete the single byte data read transfer.

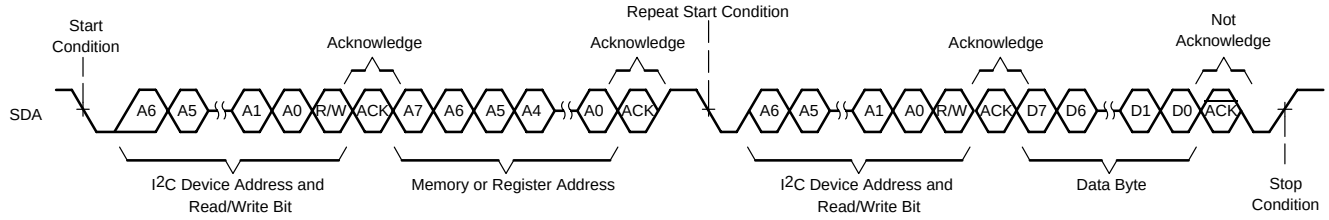


Figure 2–6. Single Byte Read Transfer

2.2.13.5 Multiple Byte Read

A multiple byte data read transfer is identical to a single byte data read transfer except that multiple data bytes are transmitted by the I^{2}C slave device to the TAS1020 device as shown in Figure 2-7. Except for the last data byte, the TAS1020 device responds with an acknowledge bit after receiving each data byte.

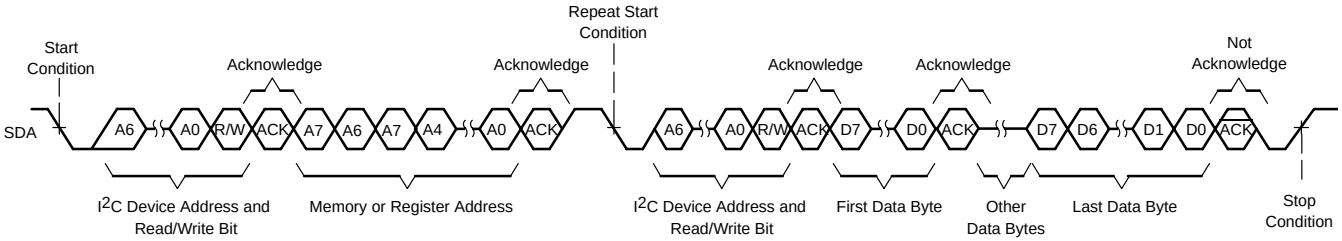


Figure 2–7. Multiple Byte Read Transfer

3 Electrical Specifications

3.1 Absolute Maximum Ratings Over Operating Temperature Ranges (unless otherwise noted)†

Supply voltage range, DV_{DD} . . . . . –0.5 to 3.6 V
Input voltage range, V_I: 3.3 V TTL/LVCMOS . . . . . –0.5 V to DV_{DD} + 0.5 V
Output voltage range, V_O: 3.3 V TTL/LVCMOS . . . . . –0.5 V to DV_{DD} + 0.5 V
Continuous power dissipation . . . . . See Dissipation Rating Table
Operating free air temperature . . . . . 0°C to 70°C
Storage temperature range . . . . . –65°C to 150°C
Lead temperature 1,6 mm (1/16 inch) from case for 10 seconds . . . . . 300°C

† Stresses beyond those listed under “absolute maximum ratings” may cause permanent damage to the device. These are stress ratings only and functional operation of the device at these or any other conditions beyond those indicated under “recommended operating conditions” is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

DISSIPATION RATING TABLE

PACKAGE	$T_A \leq 25°C$ POWER RATING	DERATING FACTOR ABOVE $T_A = 25°C$	$T_A = 70°C$ POWER RATING
TQFP	0.923 W	10.256 mW/°C	0.461 W

3.2 Recommended Operating Conditions

		MIN	NOM	MAX	UNIT
Digital supply voltage, DV_{DD}		3	3.3	3.6	V
Analog supply voltage, AV_{DD}		3	3.3	3.6	V
High-level input voltage, V_{IH}	CMOS inputs	0.7 V_{CC}			V
Low-level input voltage, V_{IL}	CMOS inputs	0		0.2 V_{CC}	V
Input voltage, V_I	CMOS inputs	0		DV_{DD}	V
Output voltage, V_O	CMOS inputs	0		DV_{DD}	V
Input transition time, t_t (t_r and t_f, 10% to 90%)		0		6	ns

3.3 Electrical Characteristics Over Recommended Operating Conditions (unless otherwise noted)

PARAMETER			TEST CONDITIONS	MIN	TYP	MAX	UNIT
V_{OH}	High-level output voltage, P3 [0-7]		I_{OH} = –4 mA	DV_{DD}–0.5			V
V_{OL}	Low-level output voltage, P3 [0-7]		I_{OL} = 4 mA			0.5	V
V_{OH}	High-level output voltage, P1 [0-7]		I_{OH} = –8 mA	DV_{DD}–0.5			V
V_{OL}	Low-level output voltage, P1 [0-7]		I_{OL} = 8 mA			0.5	V
I_{OZ}	High-impedance output current					±20	µA
I_{IL}	Low-level input current	Pullup disabled	$V_I = V_{IL}$			–20	µA
		Enabled			–250		
I_{IH}	High-level input current	Pullup disabled	$V_I = V_{IH}$			20	µA
		Enabled			20		
I_{DD}	Digital supply voltage DV_{DD} (3.3 V)		CPU clock 12 MHz		45.9		mA
			CPU clock 24 MHz		50.9		
			Suspend†		196		µA
	Analog supply voltage AV_{DD} (3.3 V)		Normal		14.7		mA
			Suspend		24		nA

† In this 196 µA measurement, the bulk or suspend current (190 µA) is delivered to the USB cable through PUR pin. The remaining 6 µA is consumed by the device. As described in section 7.2.3 of USB 1.1 specification, *When computing suspend current, the current from VBus through the pull-up and pullown resistors must be included*.

3.4 Timing Characteristics

3.4.1 Clock and Control Signals Over Recommended Operating Conditions (unless otherwise noted)

PARAMETER		TEST CONDITIONS	MIN	MAX	UNIT
f_{MCLKO1}	Clock frequency, MCLKO1	C_L = 50 pF, See Note 1	1	25	MHz
f_{MCLKO2}	Clock frequency, MCLKO2	C_L = 50 pF, See Note 1	1	25	MHz
f_{MCLKI}	Clock frequency, MCLKI	See Note 1	5	25	MHz
$t_{w(L)}$	Pulse duration, $\overline{\text{XINT}}$ low	C_L = 50 pF	0.2	10	µs

NOTE 1: Worst case duty cycle is 45/55.

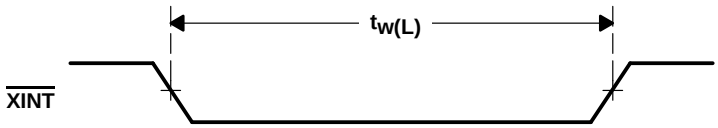


Figure 3–1. External Interrupt Timing Waveform

3.4.2 USB Transceiver Signals Over Recommended Operating Conditions (unless otherwise noted)

PARAMETER		TEST CONDITIONS	MIN	MAX	UNIT
t_r	Transition rise time for DP or DM		4	20	ns
t_f	Transition fall time for DP or DM		4	20	ns
t_{RFM}	Rise/fall time matching	$(t_r/t_f) \times 100$	90%	110%	
$V_{O(CRS)}$	Voltage output signal crossover		1.3	2	V

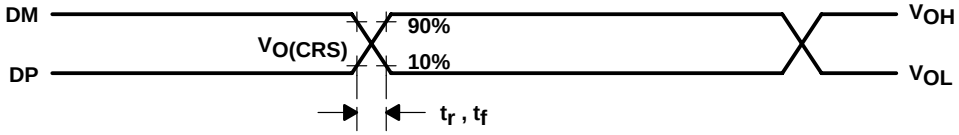


Figure 3–2. USB Differential Driver Timing Waveform

3.4.3 Codec Port Interface Signals (AC '97 Modes), T_A = 25°C, DV_{DD} = 3.3 V, AV_{DD} = 3.3 V

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
f_{BIT_CLK}	Frequency, BIT_CLK	See Note 1		12.288		MHz
t_{cyc1}	Cycle time, BIT_CLK	See Note 1		81.4		ns
$t_{w1(H)}$	Pulse duration, BIT_CLK high	See Note 1	36	40.7	45	ns
$t_{w1(L)}$	Pulse duration, BIT_CLK low	See Note 1	36	40.7	45	ns
f_{SYNC}	Frequency, SYNC	C_L = 50 pF		48		kHz
t_{cyc2}	Cycle time, SYNC	C_L = 50 pF		20.8		µs
$t_{w2(H)}$	Pulse duration, SYNC high	C_L = 50 pF		1.3		µs
$t_{w2(L)}$	Pulse duration, SYNC low	C_L = 50 pF		19.5		µs
t_{pd1}	Propagation delay time, BIT_CLK rising edge to SYNC, SD_OUT and $\overline{\text{RESET}}$	C_L = 50 pF			15	ns
t_{su}	Setup time, SD_IN to BIT_CLK falling edge		10			ns
t_h	Hold time, SD_IN from BIT_CLK falling edge		10			ns

NOTE 1: Worst case duty cycle is 45/55.

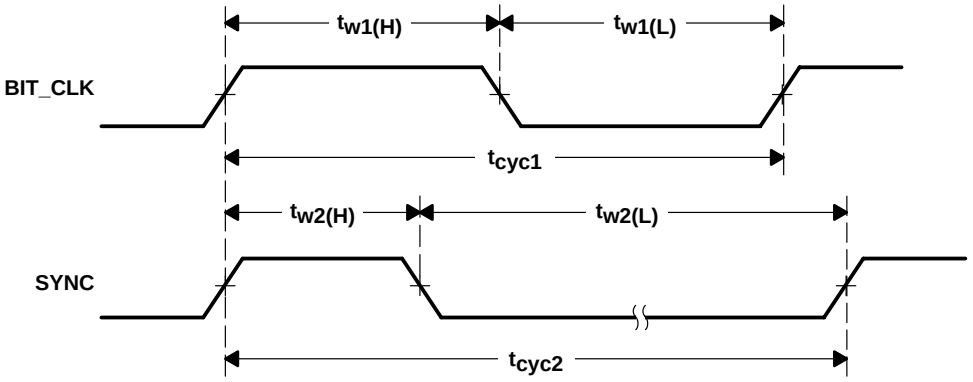


Figure 3–3. BIT_CLK and SYNC Timing Waveforms

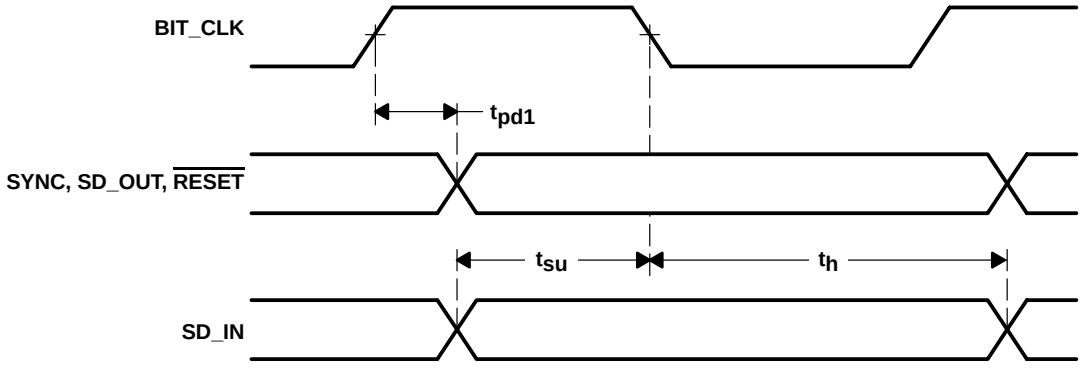


Figure 3–4. SYNC, SD_IN, and SD_OUT Timing Waveforms

3.4.4 Codec Port Interface Signals (I^2S Modes) Over Recommended Operating Conditions (unless otherwise noted)

		TEST CONDITIONS	MIN	MAX	UNIT
f_{SCLK}	Frequency, SCLK	C_L = 50 pF	(32)F_S	(64)F_S	MHz
t_{cyc}	Cycle time, SCLK	C_L = 50 pF, See Note 1	1/(64)F_S	1/(32)F_S	ns
t_{pd}	Propagation delay, SCLK falling edge to LRCLK and SDOUT	C_L = 50 pF		15	ns
t_{su}	Setup time, SDIN to SCLK rising edge		10		ns
t_h	Hold time, SDIN from SCLK rising edge		10		ns

NOTE 1: Worst case duty cycle is 45/55.

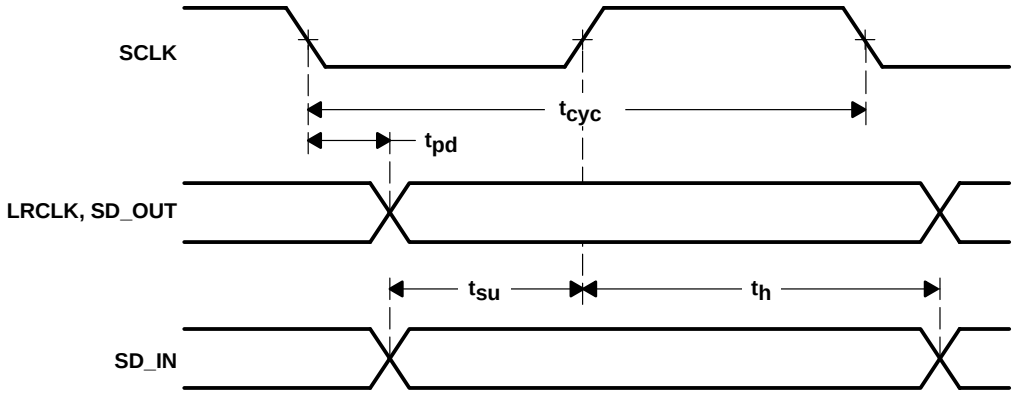


Figure 3–5. I^2S Mode Timing Waveforms

3.4.5 Codec Port Interface Signals (General-Purpose Mode) Over Recommended Operating Conditions (unless otherwise noted)

		TEST CONDITIONS	MIN	MAX	UNIT
f_{CSCLK}	Frequency, CSCLK	C_L = 50 pF	0.125	25	MHz
t_{cyc}	Cycle time, CSCLK	C_L = 50 pF, See Note 2	0.040	8	µs
t_{pd}	Propagation delay, CSCLK to CSYNC, CDATO, CSCHNE and $\overline{\text{CRESET}}$	C_L = 50 pF		15	ns
t_{su}	Setup time, CDATI to CSCLK		10		ns
t_h	Hold time, CDATI from CSCLK		10		ns

NOTE 2: The timing waveforms in Figure 3-6 show the CSYNC, CDATO, CSCHNE, and $\overline{\text{CRESET}}$ signals generated with the rising edge of the clock and the CDATI signal sampled with the falling edge of the clock. The edge of the clock used is programmable. However, the timing characteristics are the same regardless of which edge of the clock is used.

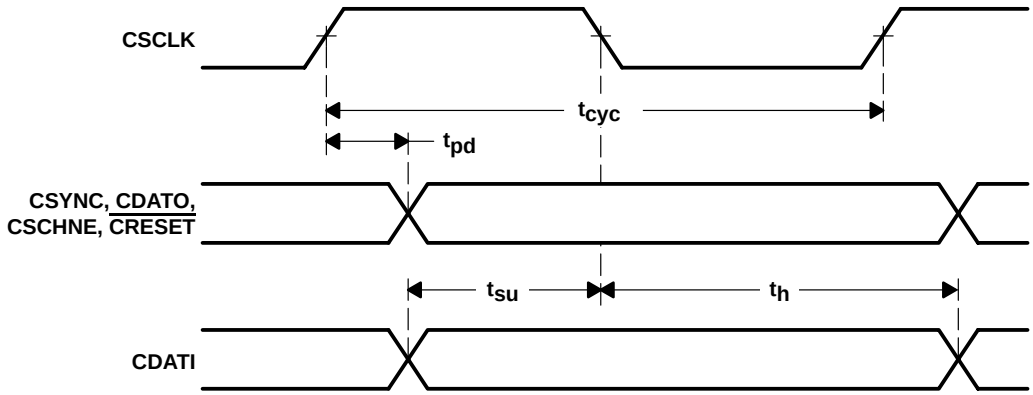


Figure 3–6. General-Purpose Mode Timing Waveforms

3.4.6 I²C Interface Signals Over Recommended Operating Conditions (unless otherwise noted)

PARAMETER		TEST CONDITIONS	STANDARD MODE		FAST MODE		UNIT
			MIN	MAX	MIN	MAX	
f_{SCL}	Frequency, SCL		0	100	0	400	kHz
$t_{w(H)}$	Pulse duration, SCL high		4		0.6		µs
$t_{w(L)}$	Pulse duration, SCL low		4.7		1.3		µs
t_r	Rise time, SCL and SDA			1000		300	ns
t_f	Fall time, SCL and SDA			300		300	ns
t_{su1}	Setup time, SDA to SCL		250		100		ns
t_{h1}	Hold time, SCL to SDA		0		0		ns
t_{buf}	Bus free time between stop and start condition		4.7		1.3		µs
t_{su2}	Setup time, SCL to start condition		4.7		0.6		µs
t_{h2}	Hold time, start condition to SCL		4		0.6		µs
t_{su3}	Setup time, SCL to stop condition		4		0.6		µs
C_L	Load capacitance for each bus line			400		400	pF

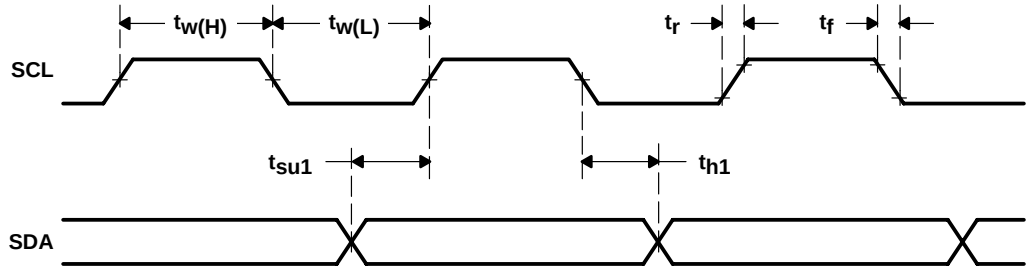


Figure 3–7. SCL and SDA Timing Waveforms

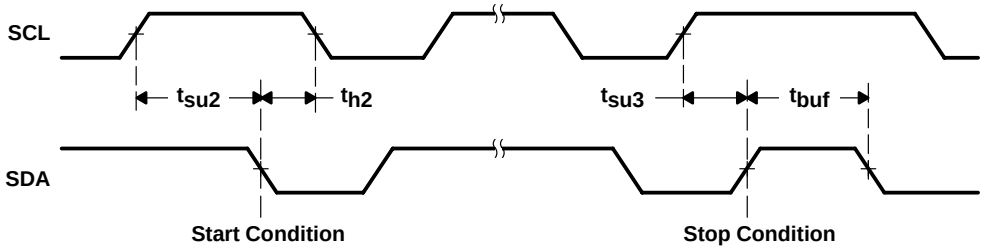


Figure 3–8. Start and Stop Conditions Timing Waveforms

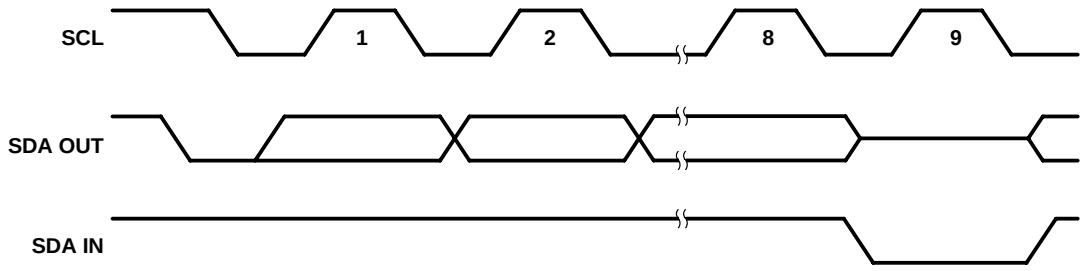


Figure 3–9. Acknowledge Timing Waveform

4 Application Information

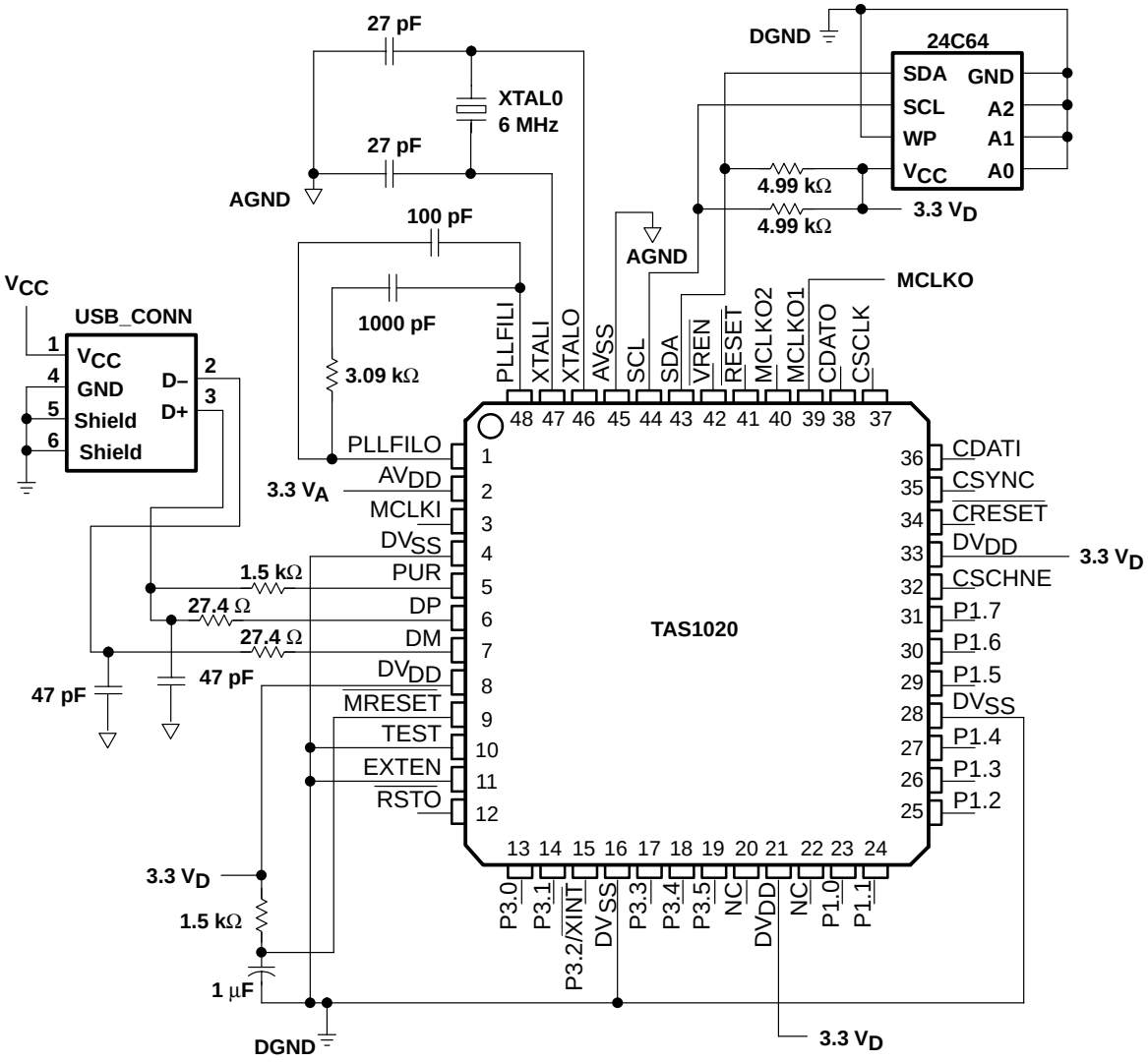


Figure 4–1. Typical TAS1020 Device Connections

Appendix A
MCU Memory and Memory-Mapped Registers

This section describes the TAS1020 MCU memory configurations and operation. In general, the MCU memory operation is the same as the industry standard 8052 MCU.

A.1 MCU Memory Space

The TAS1020 MCU memory is organized into three individual spaces: program memory, external data memory, and internal data memory. The total address range for the program memory and the external data memory spaces is 64K bytes each. The total address range for the internal data memory is 256 bytes.

The read only program memory contains the instructions to be executed by the MCU. The TAS1020 uses a 8K boot ROM as the program memory during initialization. The boot ROM program code downloads the application program code from a nonvolatile memory (that is, EEPROM) on the peripheral PCB. The application program code is written to an 6K RAM mapped to the external data memory space. After downloading the application program code to RAM, the boot ROM enables the normal operating mode by setting the ROM disable (SDW) bit (refer to memory configuration register) to enable program code execution from the 6K RAM instead of the boot ROM. In the normal operating mode, the boot ROM is still mapped to program memory space starting at address 8000h. Refer to Figures A–1 and A–2 for details.

The external data memory contains the data buffers for the USB end-points, the configuration blocks for the USB end-points, the setup data packet buffer for the USB control end-point, and memory mapped registers. The data buffers for the USB end-points, the configuration blocks for the USB end-points and the setup data packet buffer for the USB control end-point are all implemented in RAM. The memory mapped registers used for control and status registers are implemented in hardware with flip-flops. The data buffers for the USB end-points are a total of 1304 bytes, the configuration blocks for the USB end-points are a total of 128 bytes, the setup packet buffer for the USB control end-point is 8 bytes, and the memory mapped registers space is 80 bytes. The total external data memory space used for these blocks of memory is 1520 bytes. In addition to these memory blocks, a 6K (6016 bytes) RAM is mapped to the external data memory space in the boot loader mode of operation. The 6K RAM is read/write in this mode and is used to store the application program code during download by the boot ROM. In the normal mode of operation, the 6K RAM is mapped to the program memory space and is read-only.

A.2 Internal Data Memory

The internal data memory space is a total of 256 bytes of RAM, which includes the 128 bytes of special function registers (SFR) space. The internal data memory space is mapped in accordance with the industry standard 8052 MCU. The internal data memory space is mapped from 00h to FFh with the SFRs mapped from 80h to FFh. The lower 128 bytes are accessible with both direct and indirect addressing. However, the upper 128 bytes, which is the SFR space, is only accessible with direct addressing.

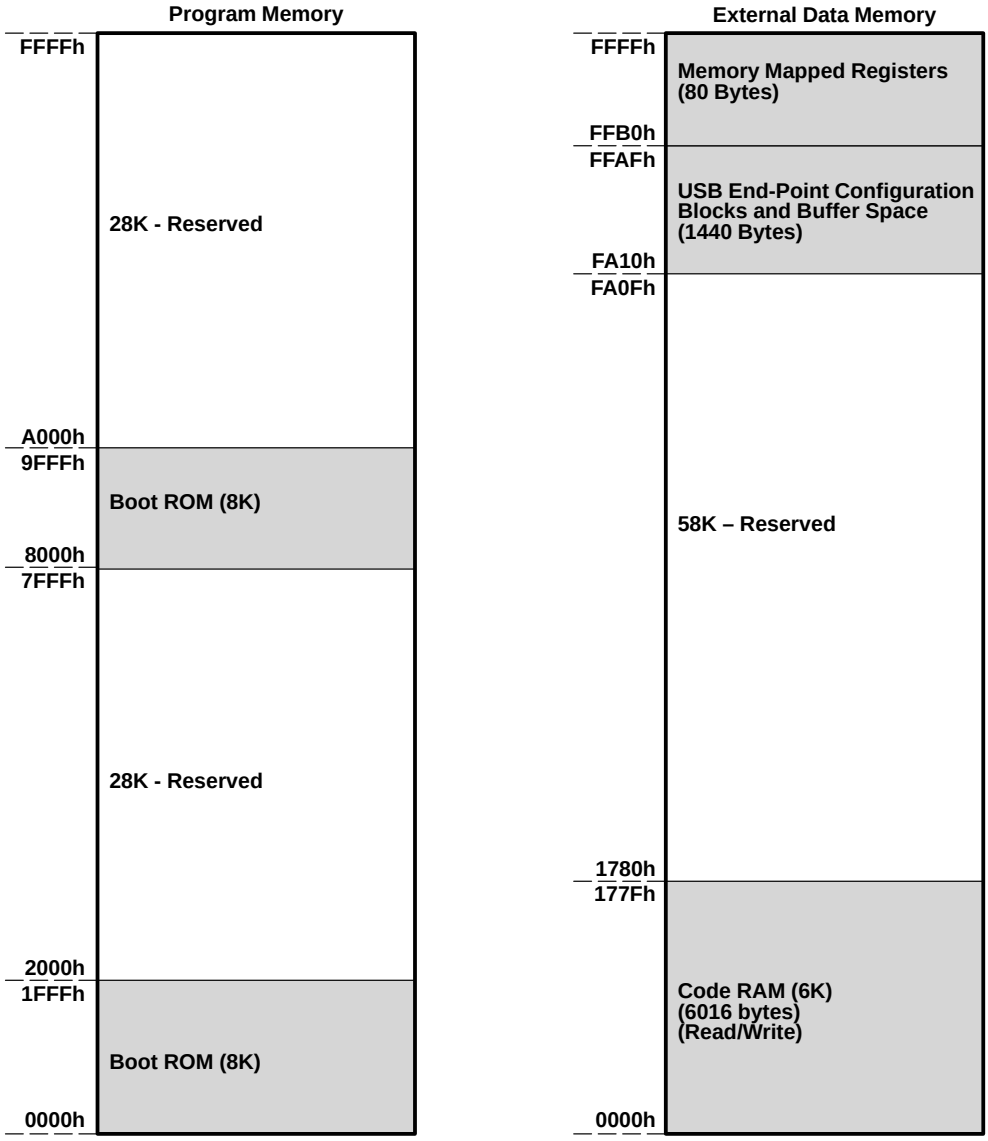


Figure A–1. Boot Loader Mode Memory Map

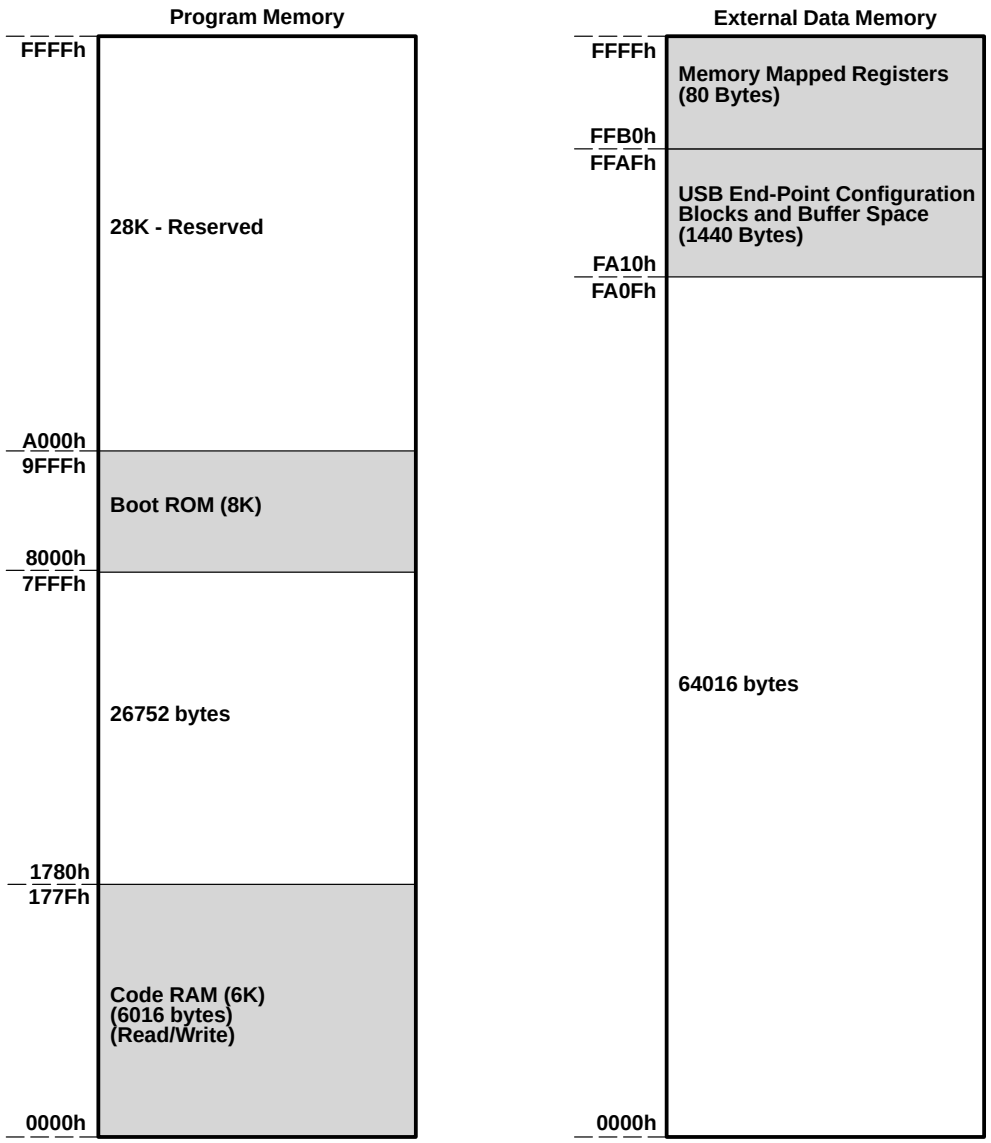


Figure A–2. Normal Operating Mode Memory Map

A.3 External MCU Mode Memory Space

When using an external MCU for firmware development, only the USB configuration blocks, the USB buffer space, and the memory mapped registers are accessible by the external MCU. See Section A.4 for details. In this mode, only address lines A0 to A10 are input to the TAS1020 device from the external MCU. Therefore, the USB buffer space and the memory mapped registers in the external data memory space are not fully decoded since all sixteen address lines are not available. Hence, the USB buffer space and the memory mapped registers are actually accessible at any 2K boundary within the total 64K external data memory space of the external MCU. As a result, when using the TAS1020 in the external MCU mode, nothing can be mapped to the external data memory space of the external MCU except the USB buffer space and the memory mapped registers of the TAS1020 device.

A.4 USB End-Point Configuration Blocks and Data Buffers Space

A.4.1 USB End-Point Configuration Blocks

The USB end-point configuration space contains 16 blocks of 8 bytes which define configuration, buffer location, buffer size, and data count for 16 (8 input and 8 output) USB end-points. The MCU, UBM, and DMA all have access to these configuration blocks.

The device defines an end-point of a USB pipe by initializing the configuration block configuration byte. It defines the location of the pipe X and Y buffers in end-point data buffer space by writing to the X buffer base address byte and Y buffer base address byte. Base addresses are octlet (8-byte) aligned. The device sets the X and Y buffer size to allocate fixed sized buffers for the pipe. Both X and Y buffer size must be greater than or equal to the USB packet size associated with the end-point. If the buffer size is greater than the USB packet size, each buffer will independently recirculate.

A.4.2 Data Buffers Space

The end-point data buffer space (1304 bytes) provides rate buffering between the USB and codecs attached to the TAS1020. Buffers are defined in this space by base address pointers and size descriptors in the USB end-point configuration blocks. The MCU also has access to this space.

Among the 1304 bytes of endpoint data buffer (FA10h-FF27h), the locations (FA10h-FAC7h) are used by current TAS1020 ROM mode. If ROM code service is used,the DMA data buffer loations available for customer's application are limited to 1120 bytes (FAC8h-FF27h) which should be enough for applications upto 5 channels, 48 kHz sampling rate with 16 bits per sample or 3 channels, 48 kHz sampling rate with 24 bits per sample.

The UBM associates USB end-points with buffers in the end-point data buffer space by looking up configuration for an end-point in USB end-point configuration space. A particular DMA channel is associated with a buffer through an end-point number in the DMA channel's control register.

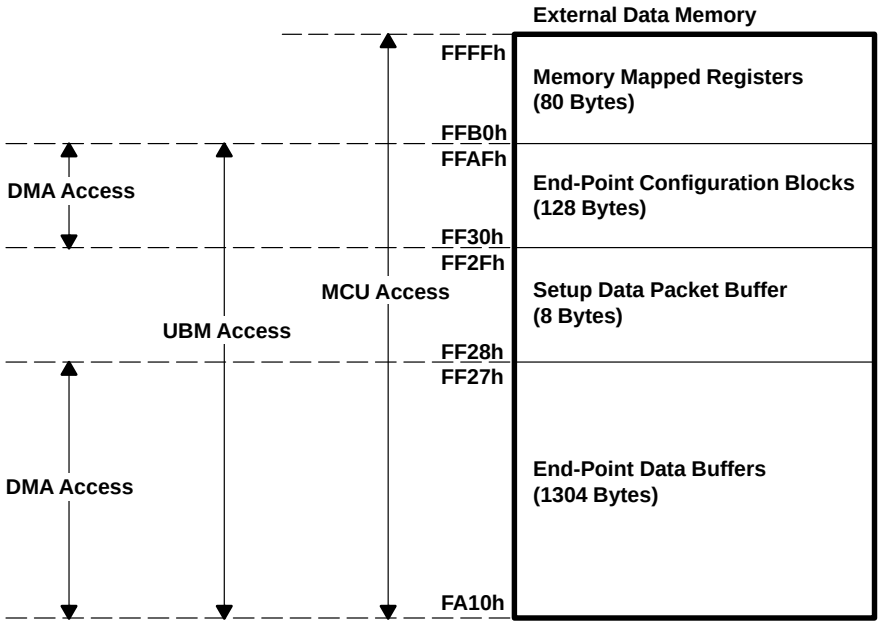


Figure A–3. USB End-Point Configuration Blocks and Buffer Space Memory Map

Table A–1. USB End-Point Configuration Blocks Address Map

ADDRESS	MNEMONIC	NAME
FFAFh	OEPDCNTY0	Out end-point 0 - Y buffer data count byte
FFAEh	Reserved	Reserved for future use
FFADh	OEPBBAY0	Out end-point 0 - Y buffer base address byte
FFACh	Reserved	Reserved for future use
FFABh	OEPDCNTX0	Out end-point 0 - X buffer data count byte
FFAAh	OEPBSIZ0	Out end-point 0 - X and Y buffer size byte
FFA9h	OEPBBAX0	Out end-point 0 - X buffer base address byte
FFA8h	OEPCNF0	Out end-point 0 – configuration byte
FFA7h	OEPDCNTY1	Out end-point 1 - Y buffer data count byte
FFA6h	Reserved	Reserved for future use
FFA5h	OEPBBAY1	Out end-point 1 - Y buffer base address byte
FFA4h	Reserved	Reserved for future use
FFA3h	OEPDCNTX1	Out end-point 1 - X buffer data count byte
FFA2h	OEPBSIZ1	Out end-point 1 - X and Y buffer size byte
FFA1h	OEPBBAX1	Out end-point 1 - X buffer base address byte
FFA0h	OEPCNF1	Out end-point 1 – configuration byte
FF9Fh	OEPDCNTY2	Out end-point 2 - Y buffer data count byte
FF9Eh	Reserved	Reserved for future use
FF9Dh	OEPBBAY2	Out end-point 2 - Y buffer base address byte
FF9Ch	Reserved	Reserved for future use
FF9Bh	OEPDCNTX2	Out end-point 2 - X buffer data count byte
FF9Ah	OEPBSIZ2	Out end-point 2 - X and Y buffer size byte
FF99h	OEPBBAX2	Out end-point 2 - X buffer base address byte
FF98h	OEPCNF2	Out end-point 2 – configuration byte
FF97h	OEPDCNTY3	Out end-point 3 - Y buffer data count byte
FF96h	Reserved	Reserved for future use
FF95h	OEPBBAY3	Out end-point 3 - Y buffer base address byte
FF94h	Reserved	Reserved for future use
FF93h	OEPDCNTX3	Out end-point 3 - X buffer data count byte
FF92h	OEPBSIZ3	Out end-point 3 - X and Y buffer size byte
FF91h	OEPBBAX3	Out end-point 3 - X buffer base address byte
FF90h	OEPCNF3	Out end-point 3 – configuration byte
FF8Fh	OEPDCNTY4	Out end-point 4 - Y buffer data count byte
FF8Eh	Reserved	Reserved for future use
FF8Dh	OEPBBAY4	Out end-point 4 - Y buffer base address byte
FF8Ch	Reserved	Reserved for future use
FF8Bh	OEPDCNTX4	Out end-point 4 - X buffer data count byte
FF8Ah	OEPBSIZ4	Out end-point 4 - X and Y buffer size byte
FF89h	OEPBBAX4	Out end-point 4 - X buffer base address byte
FF88h	OEPCNF4	Out end-point 4 – configuration byte
FF87h	OEPDCNTY5	Out end-point 5 - Y buffer data count byte
FF86h	Reserved	Reserved for future use
FF85h	OEPBBAY5	Out end-point 5 - Y buffer base address byte
FF84h	Reserved	Reserved for future use
FF83h	OEPDCNTX5	Out end-point 5 - X buffer data count byte
FF82h	OEPBSIZ5	Out end-point 5 - X and Y buffer size byte
FF81h	OEPBBAX5	Out end-point 5 - X Buffer Base Address Byte
FF80h	OEPCNF5	Out end-point 5 – configuration byte

Table A–1. USB End-Point Configuration Blocks Address Map (Continued)

ADDRESS	MNEMONIC	NAME
FF7Fh	OEPDCNTY6	Out end-point 6 - Y buffer data count byte
FF7Eh	Reserved	Reserved for future use
FF7Dh	OEPBBAY6	Out end-point 6 - Y buffer base address byte
FF7Ch	Reserved	Reserved for future use
FF7Bh	OEPDCNTX6	Out end-point 6 - X buffer data count byte
FF7Ah	OEPBSIZ6	Out end-point 6 - X and Y buffer size byte
FF79h	OEPBBAX6	Out end-point 6 - X buffer base address byte
FF78h	OEPCNF6	Out end-point 6 – configuration byte
FF77h	OEPDCNTY7	Out end-point 7 - Y buffer data count byte
FF76h	Reserved	Reserved for future use
FF75h	OEPBBAY7	Out end-point 7 - Y buffer base address byte
FF74h	Reserved	Reserved for future use
FF73h	OEPDCNTX7	Out end-point 7 - X buffer data count byte
FF72h	OEPBSIZ7	Out end-point 7 - X and Y buffer size byte
FF71h	OEPBBAX7	Out end-point 7 - X buffer base address byte
FF70h	OEPCNF7	Out end-point 7 – configuration byte
FF6Fh	IEPDCNTY0	In end-point 0 - Y buffer data count byte
FF6Eh	Reserved	Reserved for future use
FF6Dh	IEPBBAY0	In end-point 0 - Y buffer base address byte
FF6Ch	Reserved	Reserved for future use
FF6Bh	IEPDCNTX0	In end-point 0 - X buffer data count byte
FF6Ah	IEPBSIZ0	In end-point 0 - X and Y buffer size byte
FF69h	IEPBBAX0	In end-point 0 - X buffer base address byte
FF68h	IEPCNF0	In end-point 0 – configuration byte
FF67h	IEPDCNTY1	In end-point 1 - Y buffer data count byte
FF66h	Reserved	Reserved for future use
FF65h	IEPBBAY1	In end-point 1 - Y buffer base address byte
FF64h	Reserved	Reserved for future use
FF63h	IEPDCNTX1	In end-point 1 - X buffer data count byte
FF62h	IEPBSIZ1	In end-point 1 - X and Y buffer size byte
FF61h	IEPBBAX1	In end-point 1 - X buffer base address byte
FF60h	IEPCNF1	In end-point 1 – configuration byte
FF5Fh	IEPDCNTY2	In end-point 2 - Y buffer data count byte
FF5Eh	Reserved	Reserved for future use
FF5Dh	IEPBBAY2	In end-point 2 - Y buffer base address byte
FF5Ch	Reserved	Reserved for future use
FF5Bh	IEPDCNTX2	In end-point 2 - X buffer data count byte
FF5Ah	IEPBSIZ2	In end-point 2 - X and Y buffer size byte
FF59h	IEPBBAX2	In end-point 2 - X buffer base address byte
FF58h	IEPCNF2	In end-point 2 – configuration byte
FF57h	IEPDCNTY3	In end-point 3 - Y buffer data count byte
FF56h	Reserved	Reserved for future use
FF55h	IEPBBAY3	In end-point 3 - Y buffer base address byte
FF54h	Reserved	Reserved for future use
FF53h	IEPDCNTX3	In end-point 3 - X buffer data count byte
FF52h	IEPBSIZ3	In end-point 3 - X and Y buffer size byte
FF51h	IEPBBAX3	In end-point 3 - X buffer base address byte
FF50h	IEPCNF3	In end-point 3 – configuration byte

Table A–1. USB End-Point Configuration Blocks Address Map (Continued)

ADDRESS	MNEMONIC	NAME
FF4Fh	IEPDCNTY4	In end-point 4 - Y buffer data count byte
FF4Eh	Reserved	Reserved for future use
FF4Dh	IEPBBAY4	In end-point 4 - Y buffer base address byte
FF4Ch	Reserved	Reserved for future use
FF4Bh	IEPDCNTX4	In end-point 4 - X buffer data count byte
FF4Ah	IEPBSIZ4	In end-point 4 - X and Y buffer size byte
FF49h	IEPBBAX4	In end-point 4 - X buffer base address byte
FF48h	IEPCNF4	In end-point 4 – configuration byte
FF47h	IEPDCNTY5	In end-point 5 - Y buffer data count byte
FF46h	Reserved	Reserved for future use
FF45h	IEPBBAY5	In end-point 5 - Y buffer base address byte
FF44h	Reserved	Reserved for future use
FF43h	IEPDCNTX5	In end-point 5 - X buffer data count byte
FF42h	IEPBSIZ5	In end-point 5 - X and Y buffer size byte
FF41h	IEPBBAX5	In end-point 5 - X buffer base address byte
FF40h	IEPCNF5	In end-Point 5 – configuration byte
FF3Fh	IEPDCNTY6	In end-point 6 - Y buffer data count byte
FF3Eh	Reserved	Reserved for future use
FF3Dh	IEPBBAY6	In end-point 6 - Y buffer base address byte
FF3Ch	Reserved	Reserved for future use
FF3Bh	IEPDCNTX6	In end-point 6 - X buffer data count byte
FF3Ah	IEPBSIZ6	In end-point 6 - X and Y buffer size byte
FF39h	IEPBBAX6	In end-point 6 - X buffer base address byte
FF38h	IEPCNF6	In end-point 6 – configuration byte
FF37h	IEPDCNTY7	In end-point 7 - Y buffer data count byte
FF36h	Reserved	Reserved for future use
FF35h	IEPBBAY7	In end-point 7 - Y buffer base address byte
FF34h	Reserved	Reserved for future use
FF33h	IEPDCNTX7	In end-point 7 - X buffer data count byte
FF32h	IEPBSIZ7	In end-point 7 - X and Y buffer size byte
FF31h	IEPBBAX7	In end-point 7 - X buffer base address byte
FF30h	IEPCNF7	In end-point 7 – configuration byte

A.4.3 USB Out End-Point Configuration Bytes

This section describes the individual bytes in the USB end-point configuration blocks for the out end-points. A set of 8 bytes is used for the control and operation of each USB out end-point. In addition to the USB control end-point, the TAS1020 supports up to a total of seven out end-points.

A.4.3.1 USB Out End-Point – Y Buffer Data Count Byte (OEPDCNTYx)

The USB out end-point Y buffer data count byte contains the 7-bit value used to specify the amount of data received in a data packet from the host PC. The no acknowledge status bit is also contained in this byte.

Bit	7	6	5	4	3	2	1	0
Mnemonic	NACK	DCNTY6	DCNTY5	DCNTY4	DCNTY3	DCNTY2	DCNTY1	DCNTY0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

BIT	MNEMONIC	NAME	DESCRIPTION
7	NACK	No acknowledge	The no acknowledge status bit is set to a 1 by the UBM at the end of a successful USB out transaction to this end-point to indicate that the USB end-point Y buffer contains a valid data packet and that the Y buffer data count value is valid. For control, interrupt, or bulk end-points, when this bit is set to a 1, all subsequent transactions to the end-point will result in a NACK handshake response to the host PC. Also for control, interrupt, and bulk end-points to enable this end-point to receive another data packet from the host PC, this bit must be cleared to a 0 by the MCU. For isochronous end-points, a NACK handshake response to the host PC is not allowed. Therefore, the UBM ignores this bit in reference to receiving the next data packet. However, the MCU or DMA must clear this bit before reading the data packet from the buffer.
6:0	DCNTY(6:0)	Y Buffer data count	The Y buffer data count value is set by the UBM when a new data packet is written to the Y buffer for the out end-point. The 7-bit value is set to the number of bytes in the data packet for control, interrupt or bulk end-point transfers and is set to the number of samples in the data packet for isochronous end-point transfers. To determine the number of samples in the data packet for isochronous transfers, the bytes per sample value in the configuration byte is used. The data count value is read by the MCU or DMA to obtain the data packet size.

A.4.3.2 USB Out End-Point – Y Buffer Base Address Byte (OEPBBAYx)

The USB out end-point Y buffer base address byte contains the 8-bit value used to specify the base memory location for the Y data buffer for a particular USB out end-point.

Bit	7	6	5	4	3	2	1	0
Mnemonic	BBAY10	BBAY9	BBAY8	BBAY7	BBAY6	BBAY5	BBAY4	BBAY3
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

BIT	MNEMONIC	NAME	DESCRIPTION
7:0	BBAY(10:3)	Y Buffer base address	The Y buffer base address value is set by the MCU to program the base address location in memory to be used for the Y data buffer. A total of 11 bits is used to specify the base address location. This byte specifies the most significant 8 bits of the address. All 0s are used by the hardware for the three least significant bits.

A.4.3.3 USB Out End-Point – X Buffer Data Count Byte (OEPDCNTXx)

The USB out end-point X buffer data count byte contains the 7-bit value used to specify the amount of data received in a data packet from the host PC. The no acknowledge status bit is also contained in this byte.

Bit	7	6	5	4	3	2	1	0
Mnemonic	NACK	DCNTX6	DCNTX5	DCNTX4	DCNTX3	DCNTX2	DCNTX1	DCNTX0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

BIT	MNEMONIC	NAME	DESCRIPTION
7	NACK	No acknowledge	The no acknowledge status bit is set to a 1 by the UBM at the end of a successful USB out transaction to this end-point to indicate that the USB end-point X buffer contains a valid data packet and that the X buffer data count value is valid. For control, interrupt, or bulk end-points, when this bit is set to a 1, all subsequent transactions to the end-point will result in a NACK handshake response to the host PC. Also for control, interrupt, and bulk end-points to enable this end-point to receive another data packet from the host PC, this bit must be cleared to a 0 by the MCU. For isochronous end-points, a NACK handshake response to the host PC is not allowed. Therefore, the UBM ignores this bit in reference to receiving the next data packet. However, the MCU or DMA must clear this bit before reading the data packet from the buffer.
6:0	DCNTX(6:0)	X Buffer data count	The X buffer data count value is set by the UBM when a new data packet is written to the X buffer for the out end-point. The 7-bit value is set to the number of bytes in the data packet for control, interrupt, or bulk end-point transfers and is set to the number of samples in the data packet for isochronous end-point transfers. To determine the number of samples in the data packet for isochronous transfers, the bytes per sample value in the configuration byte is used. The data count value is read by the MCU or DMA to obtain the data packet size.

A.4.3.4 USB Out End-Point – X and Y Buffer Size Byte (OEPBSIZx)

The USB out end-point X and Y buffer size byte contains the 8-bit value used to specify the size of the two data buffers to be used for this end-point.

Bit	7	6	5	4	3	2	1	0
Mnemonic	BSIZ7	BSIZ6	BSIZ5	BSIZ4	BSIZ3	BSIZ2	BSIZ1	BSIZ0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

BIT	MNEMONIC	NAME	DESCRIPTION
7:0	BSIZ(7:0)	Buffer size	The X and Y buffer size value is set by the MCU to program the size of the X and Y data packet buffers. Both buffers are programmed to the same size based on this value. This value is in 8-byte units. For example, a value of 18h results in the size of the X and Y buffers each being set to 192 bytes.

A.4.3.5 USB Out End-Point – X Buffer Base Address Byte (OEPBBAXx)

The USB out end-point X buffer base address byte contains the 8-bit value used to specify the base memory location for the X data buffer for a particular USB out end-point.

Bit	7	6	5	4	3	2	1	0
Mnemonic	BBAX10	BBAX9	BBAX8	BBAX7	BBAX6	BBAX5	BBAX4	BBAX3
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

BIT	MNEMONIC	NAME	DESCRIPTION
7:0	BBAX(10:3)	X Buffer base address	The X buffer base address value is set by the MCU to program the base address location in memory to be used for the X data buffer. A total of 11 bits is used to specify the base address location. This byte specifies the most significant 8 bits of the address. All 0s are used by the hardware for the three least significant bits.

A.4.3.6 USB Out End-Point – Configuration Byte (OEPCNFx)

The USB out end-point configuration byte contains the various bits used to configure and control the end-point. Note that the bits in this byte take on different functionality based on the type of end-point defined. The control, interrupt, and bulk end-points function differently than the isochronous end-points.

A.4.3.6.1 USB Out End-Point – Control, Interrupt or Bulk configuration byte

This section defines the functionality of the bits in the USB out end-point configuration byte for control, interrupt, and bulk end-points.

Bit	7	6	5	4	3	2	1	0
Mnemonic	OEPEN	ISO	TOGGLE	DBUF	STALL	OEPIE	—	—
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

BIT	MNEMONIC	NAME	DESCRIPTION
7	OEPEN	End-Point enable	The end-point enable bit is set to a 1 by the MCU to enable the out end-point.
6	ISO	Isochronous end-point	The isochronous end-point bit is set to a 1 by the MCU to specify the use of a particular out end-point for isochronous transactions. This bit must be cleared to a '0' by the MCU to use a particular out end-point for control, interrupt, or bulk transactions.
5	TOGGLE	Toggle	The toggle bit is controlled by the UBM and is toggled at the end of a successful out data stage transaction if a valid data packet is received and the data packet PID matches the expected PID.
4	DBUF	Double buffer mode	The double buffer mode bit is set to a 1 by the MCU to enable the use of both the X and Y data packet buffers for USB transactions to a particular out end-point. This bit must be cleared to a 0 by the MCU to use the single buffer mode. In the single buffer mode, only the X buffer is used.
3	STALL	Stall	The stall bit is set to a '1' by the MCU to stall end-point transactions. When this bit is set, the hardware will automatically return a stall handshake to the host PC for any transaction received for the end-point. An exception is the control end-point setup stage transaction, which must always be received. This requirement allows a Clear_Feature_Stall request to be received from the host PC. Control end-point data and status stage transactions however can be stalled. The stall bit is cleared to a '0' by the MCU if a Clear_Feature_Stall request or a USB reset is received from the host PC. For a control write transaction, if the amount of data received is greater than expected, the UBM will set the stall bit to a 1 to stall the end-point. When the stall bit is set to a 1 by the UBM, the USB out end-point 0 interrupt will be generated.
2	OEPIE	Interrupt enable	The interrupt enable bit is set to a 1 by the MCU to enable the out end-point interrupt. See section A.5.7.1 for details on the out end-point interrupts.
1:0	—	Reserved	Reserved for future use

A.4.3.6.2 USB Out End-Point – Isochronous Configuration Byte

This section defines the functionality of the bits in the USB out end-point configuration byte for isochronous end-points.

Bit	7	6	5	4	3	2	1	0
Mnemonic	OEPEN	ISO	OVF	BPS4	BPS3	BPS2	BPS1	BPS0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

BIT	MNEMONIC	NAME	DESCRIPTION
7	OEPEN	End-Point enable	The end-point enable bit is set to a 1 by the MCU to enable the out end-point.
6	ISO	Isochronous end-point	The isochronous end-point bit is set to a 1 by the MCU to specify the use of a particular out end-point for isochronous transactions. This bit must be cleared to a 0 by the MCU for a particular out end-point to be used for control, interrupt, or bulk transactions.
5	OVF	Overflow	The overflow bit is set to a 1 by the UBM to indicate a buffer overflow condition has occurred. This bit is used for diagnostic purposes only and is not used for normal operation. This bit can only be cleared to a 0 by the MCU.
4:0	BPS(4:0)	Bytes per sample	The bytes per sample bits are used to define the number of bytes per isochronous data sample. In other words, the total number of bytes in an entire audio codec frame. For example, a PCM 16-bit stereo audio data sample consists of 4 bytes. There are two bytes of left channel data and two bytes of right channel data. For a four channel system using 16-bit data, the total number of bytes is 8, which is the isochronous data sample size. 00h = 1 byte, 01h = 2 bytes, …, 1Fh = 32 bytes

A.4.4 USB In End-Point Configuration Bytes

This section describes the individual bytes in the USB end-point configuration blocks for the in end-points. A set of 8 bytes is used for the control and operation of each USB in end-point. In addition to the USB control end-point, the TAS1020 supports up to a total of seven in end-points.

A.4.4.1 USB In End-Point – Y Buffer Data Count Byte (IEPDCNTYx)

The USB in end-point Y buffer data count byte contains the 7-bit value used to specify the amount of data to be transmitted in a data packet to the host PC. The no acknowledge status bit is also contained in this byte.

Bit	7	6	5	4	3	2	1	0
Mnemonic	NACK	DCNTY6	DCNTY5	DCNTY4	DCNTY3	DCNTY2	DCNTY1	DCNTY0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

BIT	MNEMONIC	NAME	DESCRIPTION
7	NACK	No acknowledge	The no acknowledge status bit is set to a 1 by the UBM at the end of a successful USB in transaction to this end-point to indicate that the USB end-point Y buffer is empty. For control, interrupt, or bulk end-points, when this bit is set to a 1, all subsequent transactions to the end-point will result in a NACK handshake response to the host PC. Also for control, interrupt, and bulk end-points to enable this end-point to transmit another data packet to the Host PC, this bit must be cleared to a 0 by the MCU. For isochronous end-points, a NACK handshake response to the host PC is not allowed. Therefore, the UBM ignores this bit in reference to sending the next data packet. However, the MCU or DMA must clear this bit after writing a data packet to the buffer.
6:0	DCNTY(6:0)	Y Buffer data count	The Y buffer data count value is set by the MCU or DMA when a new data packet is written to the Y buffer for the in end-point. The 7-bit value is set to the number of bytes in the data packet for control, interrupt, or bulk end-point transfers and is set to the number of samples in the data packet for isochronous end-point transfers. To determine the number of samples in the data packet for isochronous transfers, the bytes per sample value in the configuration byte is used.

A.4.4.2 USB In End-Point – Y Buffer Base Address Byte (IEPBBAYx)

The USB in end-point Y buffer base address byte contains the 8-bit value used to specify the base memory location for the Y data buffer for a particular USB in end-point.

Bit	7	6	5	4	3	2	1	0
Mnemonic	BBAY10	BBAY9	BBAY8	BBAY7	BBAY6	BBAY5	BBAY4	BBAY3
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

BIT	MNEMONIC	NAME	DESCRIPTION
7:0	BBAY(10:3)	Y Buffer base address	The Y buffer base address value is set by the MCU to program the base address location in memory to be used for the Y data buffer. A total of 11 bits is used to specify the base address location. This byte specifies the most significant 8 bits of the address. All 0s are used by the hardware for the three least significant bits.

A.4.4.3 USB In End-Point – X Buffer Data Count Byte (IEPDCNTXx)

The USB in end-point X buffer data count byte contains the 7-bit value used to specify the amount of data received in a data packet from the host PC. The no acknowledge status bit is also contained in this byte.

Bit	7	6	5	4	3	2	1	0
Mnemonic	NACK	DCNTX6	DCNTX5	DCNTX4	DCNTX3	DCNTX2	DCNTX1	DCNTX0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

BIT	MNEMONIC	NAME	DESCRIPTION
7	NACK	No acknowledge	The no acknowledge status bit is set to a 1 by the UBM at the end of a successful USB in transaction to this end-point to indicate that the USB end-point X buffer is empty. For control, interrupt, or bulk end-points, when this bit is set to a 1, all subsequent transactions to the end-point will result in a NACK handshake response to the host PC. Also for control, interrupt, and bulk end-points to enable this end-point to transmit another data packet to the host PC, this bit must be cleared to a 0 by the MCU. For isochronous end-points, a NACK handshake response to the host PC is not allowed. Therefore, the UBM ignores this bit in reference to sending the next data packet. However, the MCU or DMA must clear this bit after writing a data packet to the buffer.
6:0	DCNTX(6:0)	X Buffer data count	The X buffer data count value is set by the MCU or DMA when a new data packet is written to the X buffer for the in end-point. The 7-bit value is set to the number of bytes in the data packet for control, interrupt, or bulk end-point transfers and is set to the number of samples in the data packet for isochronous end-point transfers. To determine the number of samples in the data packet for isochronous transfers, the bytes per sample value in the configuration byte is used.

A.4.4.4 USB In End-Point – X and Y Buffer Size Byte (IEPBSIZx)

The USB in end-point X and Y buffer size byte contains the 8-bit value used to specify the size of the two data buffers to be used for this end-point.

Bit	7	6	5	4	3	2	1	0
Mnemonic	BSIZ7	BSIZ6	BSIZ5	BSIZ4	BSIZ3	BSIZ2	BSIZ1	BSIZ0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

BIT	MNEMONIC	NAME	DESCRIPTION
7	BSIZ(7:0)	Buffer size	The X and Y buffer size value is set by the MCU to program the size of the X and Y data packet buffers. Both buffers are programmed to the same size based on this value. This value should be in 8 byte units. For example, a value of 18h results in the size of the X and Y buffers each being set to 192 bytes.

A.4.4.5 SB In End-Point – X Buffer Base Address Byte (IEPBBAXx)

The USB in end-point X buffer base address byte contains the 8-bit value used to specify the base memory location for the X data buffer for a particular USB in end-point.

Bit	7	6	5	4	3	2	1	0
Mnemonic	BBAX10	BBAX9	BBAX8	BBAX7	BBAX6	BBAX5	BBAX4	BBAX3
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

BIT	MNEMONIC	NAME	DESCRIPTION
7:0	BBAX(10:3)	X Buffer base address	The X buffer base address value is set by the MCU to program the base address location in memory to be used for the X data buffer. A total of 11 bits is used to specify the base address location. This byte specifies the most significant 8 bits of the address. All 0s are used by the hardware for the three least significant bits.

A.4.4.6 USB In End-Point – Configuration Byte (IEPCNFx)

The USB in end-point configuration byte contains the various bits used to configure and control the end-point. Note that the bits in this byte take on different functionality based on the type of end-point defined. Basically, the control, interrupt and bulk end-points function differently than the isochronous end-points.

A.4.4.6.1 USB In End-Point – Control, Interrupt or Bulk Configuration Byte

This section defines the functionality of the bits in the USB in end-point configuration byte for control, interrupt, and bulk end-points.

Bit	7	6	5	4	3	2	1	0
Mnemonic	IEPEN	ISO	TOGGLE	DBUF	STALL	IEPIE	—	—
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

BIT	MNEMONIC	NAME	DESCRIPTION
7	IEPEN	End-Point enable	The end-point enable bit is set to a 1 by the MCU to enable the in end-point. This bit does not affect the reception of the control end-point setup stage transaction.
6	ISO	Isochronous end-point	The isochronous end-point bit is set to a 1 by the MCU to specify the use of a particular in end-point for isochronous transactions. This bit must be cleared to a 0 by the MCU to use a particular in end-point for control, interrupt, or bulk transactions.
5	TOGGLE	Toggle	The toggle bit is controlled by the UBM and is toggled at the end of a successful in data stage transaction if a valid data packet is transmitted. If this bit is a 0, a DATA0 PID is transmitted in the data packet to the host PC. If this bit is a 1, a DATA1 PID is transmitted in the data packet.
4	DBUF	Double buffer mode	The double buffer mode bit is set to a 1 by the MCU to enable the use of both the X and Y data packet buffers for USB transactions to a particular in end-point. This bit must be cleared to a 0 by the MCU to use the single buffer mode. In the single buffer mode, only the X buffer is used.
3	STALL	Stall	The stall bit is set to a 1 by the MCU to stall end-point transactions. When this bit is set, the hardware will automatically return a stall handshake to the host PC for any transaction received for the end-point.
2	IEPIE	Interrupt enable	The interrupt enable bit is set to a 1 by the MCU to enable the in end-point interrupt. See section A.5.7.2 for details on the in end-point interrupts.
1:0	—	Reserved	Reserved for future use.

A.4.4.6.1 USB In End-Point – Isochronous Configuration Byte

This section defines the functionality of the bits in the USB in end-point configuration byte for isochronous end-points.

Bit	7	6	5	4	3	2	1	0
Mnemonic	IEPEN	ISO	OVF	BPS4	BPS3	BPS2	BPS1	BPS0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

BIT	MNEMONIC	NAME	DESCRIPTION
7	IEPEN	End-Point enable	The end-point enable bit is set to a 1 by the MCU to enable the in end-point.
6	ISO	Isochronous end-point	The isochronous end-point bit is set to a 1 by the MCU to specify the use of a particular in end-point for isochronous transactions. This bit must be cleared to a 0 by the MCU for a particular in end-point to be used for control, interrupt, or bulk transactions.
5	OVF	Overflow	The overflow bit is set to a 1 by the UBM to indicate a buffer overflow condition has occurred. This bit is used for diagnostic purposes only and is not used for normal operation. This bit can only be cleared to a 0 by the MCU.
4:0	BPS(4:0)	Bytes per sample	The bytes per sample bits are used to define the number of bytes per isochronous data sample. In other words, the total number of bytes in an entire audio codec frame. For example, a PCM 16-bit stereo audio data sample consists of 4 bytes. There are two bytes of left channel data and two bytes of right channel data. For a four channel system using 16-bit data, the total number of bytes is 8, which is the isochronous data sample size. 00h = 1 byte, 01h = 2 bytes, …, 1Fh = 32 bytes

A.4.5 USB Control End-Point Setup Stage Data Packet Buffer

The USB control end-point setup stage data packet buffer is the buffer space used to store the 8-byte data packet received from the host PC during a control end-point transfer setup stage transaction. Refer to Chapter 9 of the USB Specification for details on the data packet.

Table A–2. USB Control End-Point Setup Data Packet Buffer Address Map

ADDRESS	NAME
FF2Fh	wLength – Number of bytes to transfer in the data stage
FF2Eh	wLength – Number of bytes to transfer in the data stage
FF2Dh	wIndex – Index or offset value
FF2Ch	wIndex – Index or offset value
FF2Bh	wValue – Value of a parameter specific to the request
FF2Ah	wValue – Value of a parameter specific to the request
FF29h	bRequest – Specifies the particular request
FF28h	bmRequestType – Identifies the characteristics of the request

A.5 Memory-Mapped Registers

The TAS1020 device provides a set of control and status registers to be used by the MCU to control the overall operation of the device. This section describes the memory-mapped registers.

Table A–3. Memory Mapped Registers Address Map

ADDRESS	MNEMONIC	NAME	SECTION	PAGE
FFFFh	USBFADR	USB function address register	A.5.1.1	A–17
FFFEh	USBSTA	USB status register	A.5.1.2	A–18
FFFDh	USBIMSK	USB interrupt mask register	A.5.1.3	A–19
FFFCh	USBCTL	USB control register	A.5.1.4	A–19
FFFBh	USBFNL	USB frame number register (low-byte)	A.5.1.5	A–20
FFFAh	USBFNH	USB frame number register (high-byte)	A.5.1.6	A–20
FFF9h	ACG2FRQ0	Adaptive clock generator2 frequency register (Byte 0)	A.5.3.6	A–25
FFF8h	ACG2FRQ1	Adaptive clock generator2 frequency register (Byte 1)	A.5.3.7	A–25
FFF7h	ACG2FRQ2	Adaptive clock generator2 frequency register (Byte 2)	A.5.3.8	A–26
FFF6h	ACG2DCTL	Adaptive clock generator2 divider control register	A.5.3.9	A–26
FFF5h	Reserved	Reserved for future use		
FFF4h	DMABCNT1H	DMA buffer content register (high-byte) (channel 1)	A.5.2.5	A–22
FFF3h	DMABCNT1L	DMA buffer content register (low-byte) (channel 1)	A.5.2.4	A–21
FFF2h	DMABPCT0	DMA bulk packet count register (low-byte)	A.5.2.6	A–22
FFF1h	DMABPCT1	DMA bulk packet count register (high-byte)	A.5.2.7	A–22
FFF0h	DMATSL1	DMA time slot assignment register (low-byte) (channel 1)	A.5.2.1	A–20
FFEFh	DMATSH1	DMA time slot assignment register (high-byte) (channel 1)	A.5.2.2	A–21
FFEEh	DMACTL1	DMA control register (channel 1)	A.5.2.3	A–21
FFEDh	Reserved	Reserved for future use		
FFECh	DMABCNT0H	DMA current buffer content register (high-byte) (channel 0)	A.5.2.5	A–22
FFEBh	DMABCNT0L	DMA current buffer content register (low-byte) (channel 0)	A.5.2.4	A–21
FFEAh	DMATSL0	DMA time slot assignment register (low-byte) (channel 0)	A.5.2.1	A–20
FFE9h	DMATSH0	DMA time slot assignment register (high-byte) (channel 0)	A.5.2.2	A–21
FFE8h	DMACTL0	DMA control register (channel 0)	A.5.2.3	A–21
FFE7h	ACG1FRQ0	Adaptive clock generator1 frequency register (byte 0)	A.5.3.1	A–24
FFE6h	ACG1FRQ1	Adaptive clock generator1 frequency register (byte 1)	A.5.3.2	A–24
FFE5h	ACG1FRQ2	Adaptive clock generator1 frequency register (byte 2)	A.5.3.3	A–24
FFE4h	ACGCAPL	Adaptive clock generator1 MCLK capture register (low byte)	A.5.3.4	A–25
FFE3h	ACGCAPH	Adaptive clock generator1 MCLK capture register (high byte)	A.5.3.5	A–25
FFE2h	ACG1DCTL	Adaptive clock generator1 divider control register	A.5.3.10	A–26
FFE1h	ACGCTL	Adaptive clock generator control register	A.5.3.11	A–27
FFE0h	CPTCNF1	Codec port interface configuration register 1	A.5.4.1	A–27
FFDFh	CPTCNF2	Codec port interface configuration register 2	A.5.4.2	A–28
FFDEh	CPTCNF3	Codec port interface configuration register 3	A.5.4.3	A–29
FFDDh	CPTCNF4	Codec port interface configuration register 4	A.5.4.4	A–30
FFDCh	CPTCTL	Codec port interface control and status register	A.5.4.5	A–31
FFDBh	CPTADR	Codec port interface address register	A.5.4.6	A–32
FFDAh	CPTDATL	Codec port interface data register (low-byte)	A.5.4.7	A–32
FFD9h	CPTDATH	Codec port interface data register (high-byte)	A.5.4.8	A–33
FFD8h	CPTVSLL	Codec port interface valid slots register (low-byte)	A.5.4.9	A–33
FFD7h	CPTVSLH	Codec port interface valid slots register (high-byte)	A.5.4.10	A–34

Table A–3. Memory-Mapped Registers Address Map (Continued)

ADDRESS	MNEMONIC	NAME	SECTION	PAGE
FFD6h	CPTRXCNF2	Codec port interface receive configuration register 2	A.5.4.11	A–34
FFD5h	CPTRXCNF3	Codec port interface receive configuration register 3	A.5.4.12	A–35
FFD4h	CPTRXCNF4	Codec port interface receive configuration register 4	A.5.4.13	A–36
FFD3h	Reserved	Reserved for future use		
FFD2h	Reserved	Reserved for future use		
FFD1h	Reserved	Reserved for future use		
FFD0h	Reserved	Reserved for future use		
FFCFh	Reserved	Reserved for future use		
FFCEh	Reserved	Reserved for future use		
FFCDh	Reserved	Reserved for future use		
FFCCh	Reserved	Reserved for future use		
FFCBh	Reserved	Reserved for future use		
FFCAh	P3MSK	Mask register for P3	A.5.5.1	A–36
FFC9h	Reserved	Reserved for future use		
FFC8h	Reserved	Reserved for future use		
FFC7h	Reserved	Reserved for future use		
FFC6h	Reserved	Reserved for future use		
FFC5h	Reserved	Reserved for future use		
FFC4h	Reserved	Reserved for future use		
FFC3h	I2CADR	I^2C interface address register	A.5.6.1	A–36
FFC2h	I2CDATI	I^2C interface receive data register	A.5.6.2	A–37
FFC1h	I2CDATO	I^2C interface transmit data register	A.5.6.3	A–37
FFC0h	I2CCTL	I^2C interface control and status register	A.5.6.4	A–38
FFBFh	Reserved	Reserved for future use		
FFBEh	Reserved	Reserved for future use		
FFBDh	Reserved	Reserved for future use		
FFBCh	Ch0WrPtrL	UBM write pointer (low-byte) (8 bits)	A.5.2.8	A–22
FFBBh	Ch0WrPtrH	UBM write pointer (high-byte) (3 bits)	A.5.2.9	A–23
FFBAh	Ch0RdPtrL	DMA read pointer (low-byte) (8 bits)	A.5.2.10	A–23
FFB9h	Ch0RdPtrH	DMA read pointer (high-byte) (3 bits)	A.5.2.11	A–23
FFB8h	Ch1WrPtrL	UBM write pointer (low-byte) (8 bits)	A.5.2.8	A–22
FFB7h	Ch1WrPtrH	UBM write pointer (high-byte) (3 bits)	A.5.2.9	A–23
FFB6h	Ch1RdPtrL	DMA read pointer (low-byte) (8 bits)	A.5.2.10	A–23
FFB5h	Ch1RdPtrH	DMA read pointer (high-byte) (3 bits)	A.5.2.11	A–23
FFB4h	OEPINT	USB out end-point interrupt register	A.5.7.1	A–39
FFB3h	IEPINT	USB in end-point interrupt register	A.5.7.2	A–39
FFB2h	VECINT	Interrupt vector register	A.5.7.3	A–40
FFB1h	GLOBCTL	Global control register	A.5.7.4	A–41
FFB0h	MEMCFG	Memory configuration register	A.5.7.5	A–41

A.5.1 USB Registers

This section describes the memory-mapped registers used for control and operation of the USB functions. This section consists of six registers used for USB functions.

A.5.1.1 USB Function Address Register (USBFADR – Address FFFFh)

The USB function address register contains the current setting of the USB device address assigned to the function by the host. After power-on reset or USB reset, the default address will be 00h. During enumeration of the function by the host, the MCU should load the assigned address to this register when a USB Set_Address request is received by the control end-point.

Bit	7	6	5	4	3	2	1	0
Mnemonic	—	FA6	FA5	FA4	FA3	FA2	FA1	FA0
Type	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION
7	—	Reserved	Reserved for future use
6:0	FA(6:0)	Function address	The function address bit values are set by the MCU to program the USB device address assigned by the host PC.

A.5.1.2 USB Status Register (USBSTA – Address FFFEh)

The USB status register contains various status bits used for USB operations.

Bit	7	6	5	4	3	2	1	0
Mnemonic	RSTR	SUSR	RESR	SOF	PSOF	SETUP	—	STPOW
Type	R	R	R	R	R	R	R	R
Default	0	0	0	0	0	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION
7	RSTR	Function reset	The function reset bit is set to a 1 by hardware in response to the host PC initiating a USB reset to the function. When a USB reset occurs, all of the USB logic blocks, including the SIE, UBM, frame timer, and suspend/resume are automatically reset. The function reset enable (FRSTE) control bit in the USB control register can be used to enable the USB reset to reset all TAS1020 logic, except the shadow the ROM (SDW) and the USB function connect (CONT) bits. When the FRSTE control bit is set to a 1, the reset output ($\overline{\text{RSTO}}$) signal from the TAS1020 device will also be active when a USB reset occurs. This bit is read only and is cleared when the MCU writes to the interrupt vector register.
6	SUSR	Function suspend	The function suspend bit is set to a 1 by hardware when a USB suspend condition is detected by the suspend/resume logic. See section 2.2.5 for details on the USB suspend and resume operation. This bit is read only and is cleared when the MCU writes to the interrupt vector register.
5	RESR	Function resume	The function resume bit is set to a 1 by hardware when a USB resume condition is detected by the suspend/resume logic. See section 2.2.5 for details on the USB suspend and resume operation. This bit is read only and is cleared when the MCU writes to the interrupt vector register.
4	SOF	Start-of-frame	The start-of-frame bit is set to a 1 by hardware when a new USB frame starts. This bit is set when the SOF packet from the host PC is detected, even if the TAS1020 frame timer is not locked to the host PC frame timer. This bit is read only and is cleared when the MCU writes to the interrupt vector register. The nominal SOF rate is 1 ms.
3	PSOF	Pseudo start-of-frame	The pseudo start-of-frame bit is set to a 1 by hardware when a USB pseudo SOF occurs. The pseudo SOF is an artificial SOF signal that is generated when the TAS1020 frame timer is not locked to the host PC frame timer. This bit is read only and is cleared when the MCU writes to the interrupt vector register. The nominal pseudo SOF rate is 1 ms.
2	SETUP	Setup stage transaction	The setup stage transaction bit is set to a 1 by hardware when a successful control end-point setup stage transaction is completed. Upon completion of the setup stage transaction, the USB control end-point setup stage data packet buffer should contain a new setup stage data packet.
1	—	Reserved	Reserved for future use
0	STPOW	Setup stage transaction over-write	The setup stage transaction over-write bit is set to a 1 by hardware when the data in the USB control end-point setup data packet buffer is over-written. This scenario occurs when the host PC prematurely terminates a USB control transfer by simply starting a new control transfer with a new setup stage transaction.

A.5.1.3 USB Interrupt Mask Register (USBMSK – Address FFFDh)

The USB interrupt mask register contains the interrupt mask bits used to enable or disable the generation of interrupts based on the corresponding status bits.

Bit	7	6	5	4	3	2	1	0
Mnemonic	RSTR	SUSR	RESR	SOF	PSOF	SETUP	—	STPOW
Type	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
Default	0	0	0	0	0	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION
7	RSTR	Function reset	The function reset interrupt mask bit is set to a 1 by the MCU to enable the USB function reset interrupt.
6	SUSR	Function suspend	The function suspend interrupt mask bit is set to a 1 by the MCU to enable the USB function suspend interrupt.
5	RESR	Function resume	The function resume interrupt mask bit is set to a 1 by the MCU to enable the USB function resume interrupt.
4	SOF	Start-of-frame	The start-of-frame interrupt mask bit is set to a 1 by the MCU to enable the USB start-of-frame interrupt.
3	PSOF	Pseudo start-of-frame	The pseudo start-of-frame interrupt mask bit is set to a 1 by the MCU to enable the USB pseudo start-of-frame interrupt.
2	SETUP	Setup stage transaction	The setup stage transaction interrupt mask bit is set to a 1 by the MCU to enable the USB setup stage transaction interrupt.
1	—	Reserved	Reserved for future use
0	STPOW	Setup stage transaction over-write	The setup stage transaction over-write interrupt mask bit is set to a 1 by the MCU to enable the USB setup stage transaction over-write interrupt.

A.5.1.4 USB Control Register (USBCTL – Address FFFCh)

The USB control register contains various control bits used for USB operations.

Bit	7	6	5	4	3	2	1	0
Mnemonic	CONT	FEN	RWUP	FRSTE	—	—	—	SDW_OK
Type	R/W	R/W	R/W	R/W	R	R	R	R/W
Default	0	0	0	0	0	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION
7	CONT	Function connect	The function connect bit is set to 1 by the MCU to connect the TAS1020 device to the USB. As a result of connecting to the USB, the host PC should enumerate the function. When this bit is set, the USB data plus pullup resistor (PUR) output signal is enabled, which will connect the pullup on the PCB to the TAS1020 3.3-V supply voltage. When this bit is cleared to 0, the PUR output is in the 3-stated. This bit is not affected by a USB reset.
6	FEN	Function enable	The function enable bit is set to 1 by the MCU to enable the TAS1020 device to respond to USB transactions. If this bit is cleared to 0, the UBM will ignore all USB transactions. This bit is cleared by a USB reset.
5	RWUP	Remote wake-up	The remote wake-up bit is set to 1 by the MCU to request the suspend/resume logic to generate resume signaling upstream on the USB. This bit is used to exit a USB low-power suspend state when a remote wake-up event occurs. After initiating the resume signaling by setting this bit, the MCU should clear this bit within 2.5 μs.
4	FRSTE	Function reset enable	The function reset enable bit is set to 1 by the MCU to enable the USB reset to reset all internal logic including the MCU. However, the shadow the ROM (SDW) and the USB function connect (CONT) bits will not be reset. When this bit is set, the reset output (RSTO) signal from the TAS1020 device will also be active when a USB reset occurs. This bit is not affected by USB reset.
3	—	Reserved	Reserved for future use.
2	—	Reserved	Reserved for future use.
1	—	Reserved	Reserved for future use.
0	SDW_OK	SDW bit confirm	This bit is used as a confirmation bit to prevent a user to spuriously clear the SDW bit in the MEMCFG register. This bit must be set to 1 before clearing the SDW bit to switch from normal mode to boot mode. This bit is not affected by USB reset.

A.5.1.5 USB Frame Number Register (Low-byte) (USBFNL – Address FFFBh)

The USB frame number register (low byte) contains the least significant byte of the 11-bit frame number value received from the host PC in the start-of-frame packet.

Bit	7	6	5	4	3	2	1	0
Mnemonic	FN7	FN6	FN5	FN4	FN3	FN2	FN1	FN0
Type	R	R	R	R	R	R	R	R
Default	0	0	0	0	0	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION
7:0	FN(7:0)	Frame number	The frame number bit values are updated by hardware each USB frame with the frame number field value received in the USB start-of-frame packet. The frame number can be used as a time stamp by the USB function. If the TAS1020 frame timer is not locked to the host PC frame timer, then the frame number is incremented from the previous value when a pseudo start-of-frame occurs.

A.5.1.6 USB Frame Number Register (High-byte) (USBFNH – Address FFFAh)

The USB frame number register (high byte) contains the most significant 3 bits of the 11-bit frame number value received from the host PC in the start-of-frame packet.

Bit	7	6	5	4	3	2	1	0
Mnemonic	—	—	—	—	—	FN10	FN9	FN8
Type	R	R	R	R	R	R	R	R
Default	0	0	0	0	0	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION
7:3	—	Reserved	Reserved for future use.
2:0	FN(10:8)	Frame number	The frame number bit values are updated by hardware each USB frame with the frame number field value received in the USB start-of-frame packet. The frame number can be used as a time stamp by the USB function. If the TAS1020 frame timer is not locked to the host PC frame timer, then the frame number is incremented from the previous value when a pseudo start-of-frame occurs.

A.5.2 DMA Registers

This section describes the memory-mapped registers used for the two DMA channels. Each DMA channel has a set of three registers.

A.5.2.1 DMA Time Slot Assignment Register (Low-byte) (DMATSL1 – Address FFF0h) (DMATSL0 – Address FFEAh)

Bit	7	6	5	4	3	2	1	0
Mnemonic	TSL7	TSL6	TSL5	TSL4	TSL3	TSL2	TSL1	TSL0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION
7:0	TSL(7:0)	Time slot assignment	The DMA time slot assignment bits are set to 1 by the MCU to define the codec port interface time slots supported by this DMA channel.

A.5.2.2 DMA Time Slot Assignment Register (High-byte) (DMATSH1 – Address FFEFh) (DMATSH0 – Address 0xFFE9)

Bit	7	6	5	4	3	2	1	0
Mnemonic	BPTS1	BPTS0	TSL13	TSL12	TSL11	TSL10	TSL9	TSL8
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION
7:6	BPTS(1:0)	Bytes per time slot	The bytes per time slot bits are used to define the number of bytes to be transferred for each time slot supported by this DMA channel. 00b = 1 byte, 01b = 2 bytes, 10b = 3 bytes, 11b = 4 bytes
5:0	TSL(13:8)	Time slot assignment	The DMA time slot assignment bits are set to 1 by the MCU to define the codec port interface time slots supported by this DMA channel.

A.5.2.3 DMA Control Register (DMACTL1 – Address FFEEh) (DMACTL0 – Address FFE8h)

Bit	7	6	5	4	3	2	1	0
Mnemonic	DMAEN	HSKEN	—	—	EPDIR	EPNUM2	EPNUM1	EPNUM0
Type	R/W	R/W	R	R	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION
7	DMAEN	DMA enable	The DMA enable bit is set to a 1 by the MCU to enable this DMA channel. Before enabling the DMA channel, all other DMA channel configuration bits must be set to the desired value.
6	HSKEN	Handshake enable	This bit is relevant for BULK data transfer in the OUT direction through DMA. MCU must set this bit to a 1 to enable the handshake mode for the data transfer. If MCU sets this bit, MCU has to enable DMA for each received BULK OUT packet. DMA, once enabled, transfers the BULK OUT packet to the C-port, disables itself and generates an interrupt to the MCU. If MCU clears this bit, DMA handles the BULK OUT data transfer to the C-port without MCU intervention. For more details, refer to section 2.2.7.3.3.
5	—	Reserved	Reserved for future use
4	—	Reserved	Reserved for future use
3	EPDIR	USB end-point direction	The USB end-point direction bit controls the direction of data transfer by this DMA channel. The MCU should set this bit to a 1 to configure this DMA channel to be used for a USB in end-point. The MCU must clear this bit to a 0 to configure this DMA channel to be used for a USB out end-point.
2:0	EPNUM(2:0)	USB end-point number	The USB end-point number bits are set by the MCU to define the USB end-point number supported by this DMA channel. Keep in mind that end-point 0 is always used for the control end-point, which is serviced by the MCU and not a DMA channel. 001b = End-Point 1, 010b = End-Point 2, …, 111b = End-Point 7

A.5.2.4 DMA Current Buffer Content Register (Low-Byte) (DMABCNT1L – Address FFE3h) (DMABCNT0L– Address FFEBh)

Bit	7	6	5	4	3	2	1	0
Mnemonic	Size 7	Size 6	Size 5	Size 4	Size 3	Size 2	Size 1	Size 0
Type	R	R	R	R	R	R	R	R
Default	0	0	0	0	0	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION
7:0	Size(7:0)	Buffer content	This register shows the buffer content (bytes) for an ISO OUT end-point. This register is updated every SOF and is stable for the following USB frame which MCU can read it to implement USB audio synchronization.

A.5.2.5 DMA Current Buffer Content Register (High-Byte) (DMABCNT1H – Address FFE4h) (DMABCNT0H – Address FFECh)

Bit	7	6	5	4	3	2	1	0
Mnemonic	Size 15	Size 14	Size 13	Size 12	Size 11	Size 10	Size 9	Size 8
Type	R	R	R	R	R	R	R	R
Default	0	0	0	0	0	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION
7:0	Size(15:8)	Buffer content	This register shows the buffer content (bytes) for an ISO OUT end-point. This register is updated every SOF and is stable for the following USB frame which MCU can read it to implement USB audio synchronization.

A.5.2.6 DMA Bulk Packet Count Register (Low-byte) (DMABPCT0 – Address FFF2h)

Bit	7	6	5	4	3	2	1	0
Mnemonic	PCNT7	PCNT6	PCNT5	PCNT4	PCNT3	PCNT2	PCNT1	PCNT0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION
7:0	PCNT (7:0)	Bulk packet count	This register shows the number of BULK OUT packets DMA has to handle in handshake mode. MCU writes to this register before enabling the DMA to program the DMA to handle up to 64K BULK packets without MCU intervention. MCU can read this register anytime.

A.5.2.7 DMA Bulk Packet Count Register (High-byte) (DMABPCT1 – Address FFF1h)

Bit	7	6	5	4	3	2	1	0
Mnemonic	PCNT15	PCNT14	PCNT13	PCNT12	PCNT11	PCNT10	PCNT9	PCNT8
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION
7:0	PCNT (15:8)	Bulk packet count	This register shows the number of BULK OUT packets DMA has to handle in handshake mode. MCU writes to this register before enabling the DMA to program the DMA to handle up to 64K BULK packets without MCU intervention. MCU can read this register anytime.

A.5.2.8 UBM Write Pointer (Low Byte) (Ch0WrPtrL – Address FFBCh) (Ch1WrPtrL – Address FFB8h)

Bit	7	6	5	4	3	2	1	0
Mnemonic	WRPTR7	WRPTR6	WRPTR5	WRPTR4	WRPTR3	WRPTR2	WRPTR1	WRPTR0
Type	R	R	R	R	R	R	R	R
Default	0	0	0	0	0	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION
7:0	WRPTR(7:0)	UBM Write Pointer	This register contains 8 LSB bits of 11-bit UBM Write Pointer of the isochronous OUT endpoint buffer. MCU can read this register anytime. This 11-bit UBM Write Pointer WRPTR can be used in conjunction with the corresponding 11-bit Chn DMA RDPTR to estimate the isochronous OUT endpoint buffer size.

A.5.2.9 UBM Write Pointer (High Byte) (Ch0WrPtrH – Address FFBBh) (Ch1WrPtrH – Address FFB7h)

Bit	7	6	5	4	3	2	1	0
Mnemonic	—	—	—	—	—	WRPTR10	WRPTR9	WRPTR8
Type	—	—	—	—	—	R	R	R
Default	—	—	—	—	—	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION
2:0	WRPTR(10:8)	UBM Write Pointer	This register contains 3 MSB bits of 11-bit UBM Write Pointer of the isochronous OUT endpoint buffer. MCU can read this register anytime. This 11-bit UBM Write Pointer WRPTR can be used in conjunction with the corresponding 11-bit Chn DMA RDPTR to estimate the isochronous OUT endpoint buffer size.

A.5.2.10 DMA Read Pointer (Low Byte) (Ch0RdPtrH – Address FFBAh) (Ch1RrPtrH – Address FFB6h)

Bit	7	6	5	4	3	2	1	0
Mnemonic	RDPTR7	RDPTR6	RDPTR5	RDPTR4	RDPTR3	RDPTR2	RDPTR1	RDPTR0
Type	R	R	R	R	R	R	R	R
Default	0	0	0	0	0	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION
7:0	RDPTR(7:0)	DMA Read Pointer	This register contains 8 LSB bits of 11-bit DMA channel n (n can be 0 or 1) Read Pointer of the Isochronous OUT endoint buffer. MCU can read this register anytime. This 11-bit CHn DMA Read pointer RDPTR can be used in conjunction with the corresponding 11-bit UBM Write Pointer WRPTR to estimate the isochronous OUT endpoint buffer size.

A.5.2.11 DMA Read Pointer (High Byte) (Ch0RdPtrH – Address FFB9h) (Ch1RrPtrH – Address FFB5h)

Bit	7	6	5	4	3	2	1	0
Mnemonic	—	—	—	—	—	WRPTR10	WRPTR9	WRPTR8
Type	—	—	—	—	—	R	R	R
Default	—	—	—	—	—	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION
2:0	RDPTR(10:8)	DMA Read Pointer	This register contains 3 MSB bits of 11-bit channel n (n can be 0 or 1) Read Pointer of the Isochronous OUT endoint buffer. MCU can read this register anytime. This 11-bit CHn DMA RDPTR can be used in conjunction with the corresponding 11-bit UBM Write Pointer WRPTR to estimate the isochronous OUT endpoint buffer size.

A.5.3 Adaptive Clock Generator Registers

This section describes the memory-mapped registers used for two adaptive clock generators for their controls and operations.

A.5.3.1 Adaptive Clock Generator1 Frequency Register (Byte 0) (ACG1FRQ0 – Address FFE7h)

The adaptive clock generator frequency register (byte 0) contains the least significant byte of the 24-bit ACG frequency value. The adaptive clock generator frequency registers, ACG1FRQ0, ACG1FRQ1, and ACG1FRQ2, contain the 24-bit value used to program the ACG1 frequency synthesizer. The 24-bit value of these three registers is used to determine the codec master clock output (MCLKO) signal frequency. See section 2.2.8 for the operation details of the adaptive clock generator including instructions for programming the 24-bit ACG frequency value.

Bit	7	6	5	4	3	2	1	0
Mnemonic	FRQ7	FRQ6	FRQ5	FRQ4	FRQ3	FRQ2	FRQ1	FRQ0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION
7:0	FRQ(7:0)	ACG frequency	The ACG frequency bit values are set by the MCU to program the ACG frequency synthesizer to obtain the desired codec master clock output (MCLKO) signal frequency.

A.5.3.2 Adaptive Clock Generator1 Frequency Register (Byte 1) (ACGFRQ1 – Address FFE6h)

The adaptive clock generator frequency register (byte 1) contains the middle byte of the 24-bit ACG 1 frequency value.

Bit	7	6	5	4	3	2	1	0
Mnemonic	FRQ15	FRQ14	FRQ13	FRQ12	FRQ11	FRQ10	FRQ9	FRQ8
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION
7:0	FRQ(15:8)	ACG frequency	The ACG frequency bit values are set by the MCU to program the ACG frequency synthesizer to obtain the desired codec master clock output (MCLKO) signal frequency.

A.5.3.3 Adaptive Clock Generator1 Frequency Register (Byte 2) (ACG1FRQ2 – Address FFE5h)

The adaptive clock generator frequency register (byte 2) contains the most significant byte of the 24-bit ACG frequency value.

Bit	7	6	5	4	3	2	1	0
Mnemonic	FRQ23	FRQ22	FRQ21	FRQ20	FRQ19	FRQ18	FRQ17	FRQ16
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION
7:0	FRQ(23:16)	ACG frequency	The ACG frequency bit values are set by the MCU to program the ACG frequency synthesizer to obtain the desired codec master clock output (MCLKO) signal frequency.

A.5.3.4 Adaptive Clock Generator1 MCLK Capture Register (Low byte) (ACG1CAPL – Address FFE4h)

The adaptive clock generator MCLK capture register (low byte) contains the least significant byte of the 16-bit codec master clock (MCLK) signal cycle count that is captured each time a USB start of frame (SOF) occurs. The value of a16-bit free running counter, which is clocked with the MCLK signal, is captured at the beginning of each USB frame. The source of the MCLK signal used to clock the 16-bit timer can be selected to be either the MCLKO signal or the MCLKO2 signal. See section 2.2.10 for the operation details of the adaptive clock generator.

Bit	7	6	5	4	3	2	1	0
Mnemonic	CAP7	CAP6	CAP5	CAP4	CAP3	CAP2	CAP1	CAP0
Type	R	R	R	R	R	R	R	R
Default	0	0	0	0	0	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION
7:0	CAP(7:0)	ACG MCLK capture	The ACG MCLK capture bit values are updated by hardware each time a USB start of frame occurs. This register contains the least signification byte of the 16-bit value.

A.5.3.5 Adaptive Clock Generator1 MCLK Capture Register (High-byte) (ACGCAPH – Address FFE3h)

The adaptive clock generator MCLK capture register (high byte) contains the most significant byte of the 16-bit codec master clock (MCLK) signal cycle count that is captured each time a USB start of frame (SOF) occurs.

Bit	7	6	5	4	3	2	1	0
Mnemonic	CAP15	CAP14	CAP13	CAP12	CAP11	CAP10	CAP9	CAP8
Type	R	R	R	R	R	R	R	R
Default	0	0	0	0	0	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION
7:0	CAP(15:8)	ACG MCLK capture	The ACG MCLK capture bit values are updated by hardware each time a USB start of frame occurs. This register contains the most signification byte of the 16-bit value.

A.5.3.6 Adaptive Clock Generator2 Frequency Register (Byte 0) (ACG2FRQ0 – Address FFF9h)

The adaptive clock generator control registers ACG2FRQ0, ACG2FRQ1, and ACG2FRQ2, contain the 24-bit value used to program the ACG2 frequency synthesizer.

Bit	7	6	5	4	3	2	1	0
Mnemonic	FRQ7	FRQ6	FRQ5	FRQ4	FRQ3	FRQ2	FRQ1	FRQ0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION
7:0	FRQ(7:0)	ACQ2 frequency	The ACG2 frequency bit values are set by the MCU to program the ACG2 frequency synthesizer.

A.5.3.7 Adaptive Clock Generator Frequency Register (Byte 1) (ACG2FRQ1 – Address FFF8h)

Bit	7	6	5	4	3	2	1	0
Mnemonic	FRQ15	FRQ14	FRQ13	FRQ12	FRQ11	FRQ10	FRQ9	FRQ8
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION
7:0	FRQ(15:8)	ACQ2 frequency	The ACG2 frequency bit values are set by the MCU to program the ACG2 frequency synthesizer.

A.5.3.8 Adaptive Clock Generator Frequency Register (Byte 2) (ACG2FRQ2 – Address FFF7h)

Bit	7	6	5	4	3	2	1	0
Mnemonic	FRQ23	FRQ22	FRQ21	FRQ20	FRQ19	FRQ18	FRQ17	FRQ16
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION
7:0	FRQ(23:16)	ACQ2 frequency	The ACG2 frequency bit values are set by the MCU to program the ACG2 frequency synthesizer.

A.5.3.9 Adaptive Clock Generator2 Divider Control Register (ACG2DCTL – Address FFF6h)

Bit	7	6	5	4	3	2	1	0
Mnemonic	DIVM3	DIVM2	DIVM1	DIVM0	–	–	–	–
Type	R/W	R/W	R/W	R/W	R	R	R	R
Default	0	0	0	0	0	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION
7:0	DIVM(3:0)	Divide by M value	The divide by M control bits are set by the MCU to program the ACG2 frequency divider. 0000b = divide by 1, 0001b = divide by 2, ... 1111b = divide by 16
3:0	–	Reserved	Reserved for future use.

A.5.3.10 Adaptive Clock Generator1 Divider Control Register (ACG1DCTL – Address FFE2h)

Bit	7	6	5	4	3	2	1	0
Mnemonic	DIVM3	DIVM2	DIVM1	DIVM0	–	DIV12	DIV11	DIV10
Type	R/W	R/W	R/W	R/W	R	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION
7:0	DIVM(3:0)	Divide by M value	The divide by M control bits are set by the MCU to program the ACG1 frequency divider. 0000b = divide by 1, 0001b = divide by 2, ... 1111b = divide by 16
2:0	DIVI(2:0)	Divide by I value	The divide by I control bits are set by the MCU to program the MCKLKI divider. 000b = divide by 1, 001b = divide by 2, ..., 111b = divide by 8

A.5.3.11 Adaptive Clock Generator Control Register (ACGCTL – Address FFE1h)

Bit	7	6	5	4	3	2	1	0
Mnemonic	MCLKO2EN	MCLKO2EN	–	MCLKO1S1	MCLKO1S0	DIVEN	MCLKO2S1	MCLKO2S0
Type	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION
7	MCLKO2EN	MCLKO2 Output enable	This bit is set to 1 by the MCU to enable the MCLKO2 signal to be an output from the TAS1020 device. If the MCLKO2 signal is not being used, then the MCU can clear this bit to 0 to disable the output.
6	MCLKO1EN	MCLKO1 Output enable	This bit is set to 1 by the MCU to enable the MCLKO1 signal to be an output from the TAS1020 device. If the MCLKO1 signal is not being used, then the MCU can clear this bit to 0 to disable the output.
4	MCLKO1S1	MCLKO1 Clock select	This bit in conjunction with MCLKO1S0, selects the MCLKO1. Refer to ACG block diagram. MCLKO1S1 MCLKO1S0 MCLKO1 0 0 acg_clk (after %M) x 1 mclki (after %I) 1 0 acg2_clk(after %M)
3	MCLKO1S0	MCLKO1 Clock select	Refer to the description above.
2	DIVEN	Divider Enable	The Divider Enable bit is set to 1 by the MCU to enable the Divide-by-I and Divide-by-M circuits.
1	MCLKO2	MCLKO1 Clock select	This bit in conjunction with MCLKO2S0, selects the MCLKO2. Refer to ACG block diagram. MCLKO2S1 MCLKO2S0 MCLKO2 0 0 acg_clk (after %M) x 1 mclki (after %I) 1 0 acg2_clk(after %M)
0	MCLKO2S0	MCLKO2 Clock select	Refer to the description above.

A.5.4 Codec Port Interface Registers

This section describes the memory-mapped registers used for the codec port interface control and operation. The codec port interface has a set of ten registers. Note that the four codec port interface configuration registers can only be written to by the MCU if the codec port enable bit (CPTEN) in the global control register is a 0.

A.5.4.1 Codec Port Interface Configuration Register 1 (CPTCNF1 – Address FFE0h)

The codoec port interface configuration register 1 is used to store various control bits for the codec port interface operation.

Bit	7	6	5	4	3	2	1	0
Mnemonic	NTSL4	NTSL3	NTSL2	NTSL1	NTSL0	MODE2	MODE1	MODE0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION
7:3	NTSL(4:0)	Number of time slots	The number of time slots bits are set by the MCU to program the number of time slots per audio frame. 00000b = 1 time slot per frame, 00001b = 2 time slots per frame, …, 01101 = 14 time slots per frame
2:0	MODE(2:0)	Mode select	The mode select bits are set by the MCU to program the codec port interface mode of operation. In addition to selecting the desired mode of operation, the MCU must also program the other configuration registers to obtain the correct serial interface format. 000b = mode 0 - General-purpose mode 001b = mode 1 - AIC mode 010b = mode 2 - AC ’97 1.X mode 011b = mode 3 - AC ’97 2.X mode 100b = mode 4 - I^2S mode – 1 OUT and 2 IN at same frequency 101b = mode 5 - I^2S mode – 1 OUT and 1 IN at different frequencies

A.5.4.2 Codec Port Interface Configuration Register 2 (CPTCNF2 – Address FFDFh)

The codec port interface configuration register 2 is used to store various control bits for the codec port interface operation.

Bit	7	6	5	4	3	2	1	0
Mnemonic	TSL0L1	TSL0L0	BPTSL2	BPTSL1	BPTSL0	TSLL2	TSLL1	TSLL0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION
7:6	TSL0L(1:0)	Time slot 0 length	The time slot 0 Length bits are set by the MCU to program the number of serial clock (CSCLK) cycles for time slot 0. 00b = CSCLK cycles for time slot 0 same as other time slots 01b = 8 CSCLK cycles for time slot 0 10b = 16 CSCLK cycles for time slot 0 11b = 32 CSCLK cycles for time slot 0
5:3	BPTSL(2:0)	Data bits per time slot	The data bits per time slot bits are set by the MCU to program the number of data bits per audio time slot. Note that this value in not used for the secondary communication address and data time slots. 000b = 8 data bits per time slot 001b = 16 data bits per time slot 010b = 18 data bits per time slot 011b = 20 data bits per time slot 100b = 24 data bits per time slot 101b = 32 data bits per time slot 110b = reserved 111b = reserved
2:0	TSLL(2:0)	Time slot length	The time slot length bits are set by the MCU to program the number of serial clock (CSCLK) cycles for all time slots except time slot 0. 000b = 8 CSCLK cycles per time slot 001b = 16 CSCLK cycles per time slot 010b = 18 CSCLK cycles per time slot 011b = 20 CSCLK cycles per time slot 100b = 24 CSCLK cycles per time slot 101b = 32 CSCLK cycles per time slot 110b = reserved 111b = reserved

A.5.4.3 Codec port interface configuration register 3 (CPTCNF3 – Address FFDEh)

The codec port interface configuration register 3 is used to store various control bits for the codec port interface operation.

Bit	7	6	5	4	3	2	1	0
Mnemonic	DDLY	TRSEN	CSCLKP	CSYNCP	CSYNCL	BYOR	CSCLKD	CSYNCD
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	1	1	0	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION
7	DDLY	Data delay	The data delay bit is set to a 1 by the MCU to program a one CSCLK cycle delay of the serial data output and input signals in reference to the leading edge of the CSYNC signal. The MCU must clear this bit to a 0 for no delay between these signals.
6	TRSEN	3-State enable	The 3-state enable bit is set to a 1 by the MCU to program the hardware to 3-state the serial data output signal for the time slots during the audio frame that are not valid. The MCU must clear this bit to a 0 to program the hardware to use zero-padding for the serial data output signal for time slots during the audio frame that are not valid.
5	CSCLKP	CSCLK polarity	The CSCLK polarity bit is used by the MCU to program the clock edge used for the codec port interface frame sync (CSYNC) output signal, codec port interface serial data output (CDATO) signal and codec port interface serial data Input (CDATI) signal. When this bit is set to a 1, the CSYNC signal will be generated with the negative edge of the codec port interface serial clock (CSCLK) signal. Also, when this bit is set to a 1, the CDATO signal is generated with the negative edge of the CSCLK signal and the CDATI signal is sampled with the positive edge of the CSCLK signal. When this bit is cleared to a 0, the CSYNC signal is generated with the positive edge of the CSCLK signal. Also, when this bit is cleared to a 0, the CDATO signal is generated with the positive edge of the CSCLK signal and the CDATI signal is sampled with the negative edge of the CSCLK signal.
4	CSYNCP	CSYNC polarity	The CSYNC polarity bit is set to a 1 by the MCU to program the polarity of the codec port interface frame sync (CSYNC) output signal to be active high. The MCU must clear this bit to a 0 to program the polarity of the CSYNC output signal to be active low.
3	CSYNCL	CSYNC length	The CSYNC length bit is set to a 1 by the MCU to program the length of the codec port interface frame sync (CSYNC) output signal to be the same number of CSCLK cycles as time slot 0. The MCU must clear this bit to a 0 to program the length of the CSYNC output signal to be one CSCLK cycle.
2	BYOR	Byte order	The byte order bit is used by the MCU to program the byte order for the data moved by the DMA between the USB end-point buffer and the codec port interface. When this bit is set to a 1, the byte order of each audio sample is reversed when the data is moved to/from the USB end-point buffer. When this bit is cleared to a 0, the byte order of the each audio sample is unchanged.
1	CSCLKD	CSCLK direction	The CSCLK direction bit is set to a 1 by the MCU to program the direction of the codec port interface serial clock (CSCLK) signal as an input to the TAS1020 device. The MCU must clear this bit to a 0 to program the direction of the CSCLK signal as an output from the TAS1020 device.
0	CSYNCD	CSYNC direction	The CSYNC direction bit is set to a 1 by the MCU to program the direction of the codec port interface frame sync (CSYNC) signal as an input to the TAS1020 device. The MCU must clear this bit to a 0 to program the direction of the CSYNC signal as an output from the TAS1020 device.

A.5.4.4 Codec Port Interface Configuration Register 4 (CPTCNF4 – Address FFDDh)

The codec port interface configuration register 4 is used to store various control bits for the codec port interface operation.

Bit	7	6	5	4	3	2	1	0
Mnemonic	ATSL3	ATSL2	ATSL1	ATSL0	CLKS	DIVB2	DIVB1	DIVB0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION
7:4	ATSL(3:0)	Command/status address/data time slot	The command/status address/data time slot bits are set by the MCU to program the time slots to be used for the secondary communication address and data values. For the AC '97 modes of operation, this value must be set to 0001b which results in time slot 1 being used for the address and time slot 2 being used for the data. For the AIC and general-purpose modes of operation, the same time slot is used for both address and data. For the AIC mode of operation, for example, this value must be set to 0111b which results in time slot 7 being used for both the address and data. 0000b = time slot 0, 0001b = time slot 1, …, 1111b = time slot 15
3	CptBlk	C-port bulk mode	This bit is used when C-port is in Mode 0. In this mode, MCU or DMA can send ISO or BULK data to the C-port. If this bit is cleared to 0, the C-port sync/clocks are free running once C-port is enabled. If this bit is set to 1, DMA controls the C-port sync/clocks. The sync/clocks are active only when valid data is present in a codec frame.
2:0	DIVB(2:0)	Divide by B value	The divide by B control bits are set by the MCU to program the CSCLK signal divider. 000b = disabled 001b = divide by 2 010b = divide by 3 011b = divide by 4 100b = divide by 5 101b = divide by 6 110b = divide by 7 111b = divide by 8

A.5.4.5 Codec Port Interface Control and Status Register (CPTCTL – Address FFDCh)

The codec port interface control and status register contains various control and status bits used for the codec port interface operation.

Bit	7	6	5	4	3	2	1	0
Mnemonic	RXF	RXIE	TXE	TXIE	—	CID1	CID0	CRST
Type	R	R/W	R	R/W	R	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION
7	RXF	Receive data register full	The receive data register full bit is set to a 1 by hardware when a new data value has been received into the receive data register from the codec device. This bit is read only and is cleared to a 0 by hardware when the MCU reads the new value from the receive data register. Note that when the MCU writes to the interrupt vector register, the codec port interface receive data register full interrupt is cleared but this status bit is not cleared at that time.
6	RXIE	Receive interrupt enable	The receive interrupt enable bit is set to a 1 by the MCU to enable the C-port receive data register full interrupt.
5	TXE	Transmit data register empty	The transmit data register empty bit is set to a 1 by hardware when the data value in the transmit data register has been sent to the codec device. This bit is read only and is cleared to a 0 by hardware when a new data byte is written to the transmit data register by the MCU. Note that when the MCU writes to the interrupt vector register, the codec port interface transmit data register empty interrupt is cleared but this status bit is not cleared at that time.
4	TXIE	Transmit interrupt enable	The transmit interrupt enable bit is set to a 1 by the MCU to enable the codec port interface transmit data register empty interrupt.
3	—	Reserved	Reserved for future use
2:1	CID(1:0)	Codec ID	The codec ID bits are used by the MCU to select between the primary codec device and the secondary codec device for secondary communication in the AC ’97 modes of operation. When the bits are cleared to ’00’, the primary codec device is selected. When the bits are set to ’01’, ’10’ or ’11’, the secondary codec device is selected. Note that when only a primary codec device is connected to the TAS1020, the bits remain cleared to ’00’.
0	CRST	Codec reset	The codec reset bit is used by the MCU to control the codec port interface reset ($\overline{\text{CRESET}}$) output signal from the TAS1020 device. When this bit is set to a 1, the $\overline{\text{CRESET}}$ signal is a high. When this bit is cleared to a 0, the $\overline{\text{CRESET}}$ signal is active low. At power up this bit is cleared to a 0, which means the $\overline{\text{CRESET}}$ output signal is active low and remains active low until the MCU sets this bit to a 1. Note that this output signal is not used in the I^2S modes of operation.

A.5.4.6 Codec Port Interface Address Register (CPTADR –@ Address FFDBh)

The codec port interface address register contains the read/write control bit and address bits used for secondary communication between the TAS1020 MCU and the codec device. For write transactions to the codec, the 8-bit value in this register is sent to the codec in the designated time slot and appropriate bit locations. Note that for the different modes of operation, the number of address bits and the bit location of the read/write bit is different. For example, the AC ’97 modes require 7 address bits and the bit location of the read/write bit to be the most significant bit. The AIC mode only requires 4 address bits and the bit location of the read/write bit to be bit 13 of the 16-bits in the time slot. The MCU must load the read/write and address bits to the correct bit locations within this register for the different modes of operation. Shown below are the read/write control bit and address bits for the AC ’97 mode of operation.

Bit	7	6	5	4	3	2	1	0
Mnemonic	R/W	A6	A5	A4	A3	A2	A1	A0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION
7	R/W	Command/status read/write control	The command/status read/write control bit value is set by the MCU to program the type of secondary communication transaction to be done. This bit must be set to a 1 by the MCU for a read transaction and cleared to a 0 by the MCU for a write transaction.
6:0	A(6:0)	Command/status address	The command/status address value is set by the MCU to program the codec device control/status register address to be accessed during the read or write transaction. The command/status address value is updated by hardware with the control/status register address value received from the codec device for read transactions.

A.5.4.7 Codec Port Interface Data Register (Low Byte) (CPTDATL – Address FFDAh)

The codec port interface data register (low byte) contains the least significant byte of the 16-bit command or status data value used for secondary communication between the TAS1020 MCU and the codec device. Note that for general-purpose mode or AIC mode only an 8-bit data value is used for secondary communication.

Bit	7	6	5	4	3	2	1	0
Mnemonic	D7	D6	D5	D4	D3	D2	D1	D0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION
7:0	D(7:0)	Command/status data	The command/status data value is set by the MCU with the command data to be transmitted to the codec device for write transactions. The command/status data value is updated by hardware with the status data received from the codec device for read transactions.

A.5.4.8 Codec Port Interface Data Register (High Byte) (CPTDATH – Address FFD9h)

The codec port interface data register (high byte) contains the most significant byte of the 16-bit command or status data value used for secondary communication between the TAS1020 MCU and the codec device. This register is not used for general-purpose mode or AIC mode since these modes only support an 8-bit data value for secondary communication.

Bit	7	6	5	4	3	2	1	0
Mnemonic	D15	D14	D13	D12	D11	D10	D9	D8
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION
7:0	D(15:8)	Command/status data	The command/status data value is set by the MCU with the command data to be transmitted to the codec device for write transactions. The command/status data value is updated by hardware with the status data received from the codec device for read transactions.

A.5.4.9 Codec Port Interface Valid Time Slots Register (Low Byte) (CPTVSLL – Address FFD8h)

The codec port interface valid time slots register (low byte) contains the control bits used to specify which time slots in the audio frame contain valid data. This register is only used in the AC ’97 modes of operation.

Bit	7	6	5	4	3	2	1	0
Mnemonic	VTSL8	VTSL9	VTSL10	VTSL11	VTSL12	—	—	—
Type	R/W	R/W	R/W	R/W	R/W	R	R	R
Default	0	0	0	0	0	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION
7:3	VTSL(8:12)	Valid time slot	The valid time slot bits are set to a 1 by the MCU to define which time slots in the audio frame contain valid data. The MCU must clear to a 0 the bits corresponding to time slots that do not contain valid data. Note that bits 7 to 3 of this register correspond to time slots 8 to 12.
2:0	—	Reserved	Reserved for future use

A.5.4.10 Codec Port Interface Valid Time Slots Register (High Byte) (CPTVSLH – Address FFD7h)

The codec port interface valid time slots register (high byte) contains the control bits used to specify which time slots in the audio frame contain valid data. In addition the valid frame, primary codec ready and secondary codec ready bits are contained in this register. This register is only used in the AC '97 modes of operation.

Bit	7	6	5	4	3	2	1	0
Mnemonic	VF	PCRDY	SCRDY	VTSL3	VTSL4	VTSL5	VTSL6	VTSL7
Type	R/W	R	R	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION
7	VF	Valid frame	The valid frame bit is set to a 1 by the MCU to indicate that the current audio frame contains at least one time slot with valid data. The MCU must clear this bit to a 0 to indicate that the current audio frame does not contain any time slots with valid data.
6	PCRDY	Primary codec ready	The primary codec ready bit is updated by hardware each audio frame based on the value of bit 15 in time slot 0 of the incoming serial data from the primary codec. This bit is set to a 1 to indicate the primary codec is ready for operation.
5	SCRDY	Secondary codec ready	The secondary codec ready bit is updated by hardware each audio frame based on the value of bit 15 in time slot 0 of the incoming serial data from the secondary codec. This bit is set to a 1 to indicate the secondary codec is ready for operation. Note that this bit is only used if a secondary codec is connected to the TAS1020 device.
4:0	VTSL(3:7)	Valid time slot	The valid time slot bits are set to a 1 by the MCU to define which time slots in the audio frame contain valid data. The MCU must clear to a 0 the bits corresponding to time slots that do not contain valid data. Note that bits 4 to 0 of this register correspond to time slots 3 to 7.

A.5.4.11 Codec Port Receive Interface Configuration Register 2 (CPTRXCNF2 – Address FFD6h)

The codec port receive interface configuration register2 is only used in I^2S Mode 5.

Bit	7	6	5	4	3	2	1	0
Mnemonic	–	–	BPTSL2	BPTSL1	BPTSL0	TSLL2	TSLL1	TSSL0
Type	R	R	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION
5:3	BPTSL(2:0)	Data bits per time slot.	The data bits per time slot bits are set by the MCU to program the number of data bits per audio time slot. Note that this value in not used for the secondary communication address and data time slots. 000b = 8 data bits per time slot 001b = 16 data bits per time slot 010b = 18 data bits per time slot 011b = 20 data bits per time slot 100b = 24 data bits per time slot 101b = 32 data bits per time slot 110b = reserved
2:0	TSSL(2:0)	Time slot length	The time slot length bits are set by the MCU to program the number of serial clock (CSCLK2) cycles for all time slots except time slot 0. 000b = 8 CSCLK cycles per time slot 001b = 16 CSCLK cycles per time slot 010b = 18 CSCLK cycles per time slot 011b = 20 CSCLK cycles per time slot 100b = 24 CSCLK cycles per time slot 101b = 32 CSCLK cycles per time slot 110b = reserved 111b= reserved

A.5.4.12 Codec Port Receive Interface Configuration Register 3 (CPTRXCNF3 – Address FFD5h)

The codec port receive interface configuration register3 is only used in I^2S Mode 5.

Bit	7	6	5	4	3	2	1	0
Mnemonic	DDLY	TRSEN	CSCLKP	CSYNCP	CSYNCL	BYOR	CSCLKD	CSYNCD
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	1	1	0	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION
7	DDLY	Data delay	The data delay bit is set to 1 by the MCU to program a one CSCLK2 cycle delay of the serial data output and input signals in reference to the leading edge of the CSYNC2 signal. The MCU must clear this bit to a 0 for no delay between these signals.
6	TRSEN	Tri-state enable	The 3-state enable bit is set to a 1 by the MCU to program the hardware to 3-state the serial data output signal for time slots during the audio frame that are not valid. The MCU must clear this bit to a 0 to program the hardware to use zero-padding for the serial data output signal for time slots during the audio frame that are not valid.
5	CSCLKP	CSCLK polarity	The CSCLK2 polarity bit is used by the MCU to program the clock edge used for the codec Port Interface Frame Sync (CSYNC2) output signal and codec Port Interface Serial Data Input (CDAT() signal. When this bit is set to a 1, the CSYNC2 signal is generated with the negative edge of the codec Port Interface Serial Clock (CSCLK2) signal. Also, when this bit is set a 1, the CDATI signal is sampled with the positive edge of the CLSCLK2 signal. When this bit is cleared to 0, the CDATI signal is sampled with the negative edge of the CSCLK2 signal.
4	CSYNCP	CSYNC polarity	The CSCLK2 polarity bit is set to a 1 by the MCU to program the polarity of the codec Port Interface Frame Sync (CSYNC2) output signal to be active high. The MCU must clear this bit to a 0 to program the polarity of the CSYNC2 output signal to be active low.
3	CSYNCL	CSYNC length	The CSCLK2 polarity bit is set to a 1 by the MCU to program the polarity of the codec Port Interface Frame Sync (CSYNC2) output signal to be the same number of CSCLK cycles as time slot 0. The MCU must clear this bit to a 0 to program the length of the CSYNC2 output signal to be one CSCLK2 cycle.
2	BYOR	Byte order	The byte order bit is used by the MCU to program the byte order for the data moved by the DMA between the USB end-point buffer and the codec port interface. When this bit is set to a 1, the byte order of each audio sample is reversed when the data is moved to/from the USB end-point buffer. When this bit is cleared to a 0, the byte order of the each audio sample is unchanged.
1	CSCLKD	CSCLK direction	The CSCLK2 direction bit is set to a 1 by the MCU to program the direction of the codec port interface serial clock (CSCLK2) signal as an input of the TAS1020 device. The MCU must clear this bit to a 0 to program the direction of the CSCLK signal as an output from the TAS1020 device.
0	CSYNCD	CSYNC direction	The CSCLK2 direction bit is set to a 1 by the MCU to program the direction of the codec port interface frame sync (CSYNC2) signal as an input of the TAS1020 device. The MCU must clear this bit to a 0 to program the direction of the CSYNC2 signal as an output from the TAS1020 device.

A.5.4.13 Codec Port Receive Interface Configuration Register 4 (CPTRXCNF4 – Address FFD4h)

The codec port receive interface configuration register4 is only used in I2S Mode 5.

Bit	7	6	5	4	3	2	1	0
Mnemonic	–	–	–	–	–	DIVB2	DIVB1	DIVB0
Type	R	R	R	R	R	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION
2:0	DIVB(2:0)	Divide by B value	The divide by B control bits are set by the MCU to program the CSCLK2 signal divider. 000b = disabled 001b = divide by 2 010b = divide by 3 011b = divide by 4 100b = divide by 5 101b = divide by 6 110b = divide by 7 111b = divide by 8

A.5.5 P3 Mask Register

Mask register for P3 to enable the wake-up function for these pins when the device is in low-power mode.

A.5.5.1 P3 Mask Register (P3MSK–Address FFCAh)

Bit	7	6	5	4	3	2	1	0
Mnemonic	P3MSK7	P3MSK6	P3MSK5	P3MSK4	P3MSK3	P3MSK2	P3MSK1	P3MSK0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION
7:0	P3MSK(7:0)		0 = Unmasked 1 = Masked

A.5.6 I2C Interface Registers

This section describes the memory-mapped registers used for the I2C Interface control and operation. The I2C interface has a set of four registers. See section 2.2.17 for the operation details of the I2C Interface.

A.5.6.1 I2C Interface Address Register (I2CADR – Address FFC3h)

The I2C interface address register contains the 7-bit I2C slave device address and the read/write transaction control bit.

Bit	7	6	5	4	3	2	1	0
Mnemonic	A6	A5	A4	A3	A2	A1	A0	RW
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION
7:1	A(6:0)	Address	The address bit values are set by the MCU to program the 7-bit I2C slave address of the device to be accessed. Each I2C slave device must have a unique address on the I2C bus. This address is used to identify the device on the bus to be accessed and is not the internal memory address to be accessed within the device.
0	RW	Read/w3rite control	The read/write control bit value is set by the MCU to program the type of I2C transaction to be done. This bit must be set to a 1 by the MCU for a read transaction and cleared to a 0 by the MCU for a write transaction.

A.5.6.2 I2C Interface Receive Data Register (I2CDATI – Address FFC2h)

The I2C interface receive data register contains the most recent data byte received from the slave device.

Bit	7	6	5	4	3	2	1	0
Mnemonic	RXD7	RXD6	RXD5	RDXD4	RXD3	RXD2	RXD1	RXD0
Type	R	R	R	R	R	R	R	R
Default	0	0	0	0	0	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION
7	RXD(7:0)	Receive data	The receive data byte value is updated by hardware for each data byte received from the I2C slave device.

A.5.6.3 I2C Interface Transmit Data Register (I2CDATO – Address FFC1h)

The I2C interface transmit data register contains the next address or data byte to be transmitted to the slave device in accordance with the protocol. Note that for both read and write transactions, the internal register or memory address of the slave device being accessed must be transmitted to the slave device.

Bit	7	6	5	4	3	2	1	0
Mnemonic	TXD7	TXD6	TXD5	TXD4	TXD3	TXD2	TXD1	TXD0
Type	W	W	W	W	W	W	W	W
Default	0	0	0	0	0	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION
7:0	TXD(7:0)	Transmit data	The transmit data byte value is set by the MCU for each address or data byte to be transmitted to the I2C slave device.

A.5.6.4 I2C Interface Control and Status Register (I2CCTL – Address FFC0h)

The I2C interface control and status register contains various control and status bits used for the I2C interface operation.

Bit	7	6	5	4	3	2	1	0
Mnemonic	RXF	RXIE	ERR	FRQ	TXE	TXIE	STPRD	STPWR
Type	R	R/W	R/W	R/W	R	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION
7	RXF	Receive data register full	The receive data register full bit is set to a 1 by hardware when a new data byte has been received into the receive data register from the slave device. This bit is read only and is cleared to a 0 by hardware when the MCU reads the new byte from the receive data register. Note that when the MCU writes to the interrupt vector register, the I2C receive data register full interrupt is cleared but this status bit is not cleared at that time.
6	RXIE	Receive interrupt enable	The receive interrupt enable bit is set to a 1 by the MCU to enable the I2C receive data register full interrupt.
5	ERR	Error condition	The error condition bit is set to a 1 by hardware when the slave device does not respond. This bit is read/write and can only be cleared by the MCU.
4	FRQ	Frequency select	The frequency select bit is used by the MCU to program the I2C serial clock (SCL) output signal frequency. A value of 0 sets the SCL frequency to 100 kHz and a value of 1 sets the SCL frequency to 400 kHz.
3	TXE	Transmit data register empty	The transmit data register empty bit is set to a 1 by hardware when the data byte in the transmit data register has been sent to the slave device. This bit is read only and is cleared to a 0 by hardware when a new data byte is written to the transmit data register by the MCU. Note that when the MCU writes to the interrupt vector register, the I2C transmit data register empty interrupt is cleared but this status bit is not cleared at that time.
2	TXIE	Transmit interrupt enable	The transmit interrupt enable bit is set to a 1 by the MCU to enable the I2C transmit data register empty interrupt.
1	STPRD	Stop – read transaction	The stop read transaction bit is set to a 1 by the MCU to enable the hardware to generate a stop condition on the I2C bus after the next data byte from the slave device is received into the receive data register. The MCU must clear this bit to a 0 after the read transaction has concluded.
0	STPWR	Stop – write transaction	The stop write transaction bit is set to a 1 by the MCU to enable the hardware to generate a stop condition on the I2C bus after the data byte in the transmit data register is sent to the slave device. The MCU must clear this bit to a 0 after the write transaction has concluded.

A.5.7 Miscellaneous Registers

This section describes the memory-mapped registers used for the control and operation of miscellaneous functions in the TAS1020 device. The registers include the USB out end-point interrupt register, the USB in end-point interrupt register, the interrupt vector register, the global control register, and the memory configuration register.

A.5.7.1 USB Out End-Point Interrupt Register (OEPINT – Address FFB4h)

The USB out end-point interrupt register contains the interrupt pending status bits for the USB out end-points. These bits do not apply to the USB isochronous end-points. Also, these bits are read only by the MCU and are used for diagnostic purposes only.

Bit	7	6	5	4	3	2	1	0
Mnemonic	OEPI7	OEPI6	OEPI5	OEPI4	OEPI3	OEPI2	OEPI1	OEPI0
Type	R	R	R	R	R	R	R	R
Default	0	0	0	0	0	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION
7:0	OEPI(7:0)	Out end-point interrupt	The out end-point interrupt status bit for a particular USB out end-point is set to a 1 by the UBM when a successful completion of a transaction occurs to that out end-point. When a bit is set, an interrupt to the MCU will be generated and the corresponding interrupt vector will result. The status bit is cleared when the MCU writes to the interrupt vector register. These bits do not apply to isochronous out end-points.

A.5.7.2 USB In End-Point Interrupt Register (IEPINT – Address FFB3h)

The USB in end-point interrupt register contains the interrupt pending status bits for the USB in end-points. These bits do not apply to the USB isochronous end-points. Also, these bits are read only by the MCU and are used for diagnostic purposes only.

Bit	7	6	5	4	3	2	1	0
Mnemonic	IEPI7	IEPI6	IEPI5	IEPI4	IEPI3	IEPI2	IEPI1	IEPI0
Type	R	R	R	R	R	R	R	R
Default	0	0	0	0	0	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION
7:0	IEPI(7:0)	In end-point interrupt	The in end-point interrupt status bit for a particular USB in end-point is set to a 1 by the UBM when a successful completion of a transaction occurs to that in end-point. When a bit is set, an interrupt to the MCU is generated and the corresponding interrupt vector will result. The status bit will be cleared when the MCU writes to the interrupt vector register. These bits do not apply to isochronous in end-points.

A.5.7.3 Interrupt Vector Register (VECINT – Address FFB2H)

The interrupt vector register contains a 6-bit vector value that identifies the interrupt source for the INT0 input to the MCU. All of the TAS1020 internal interrupt sources and the external interrupt input to the device are ORed together to generate the internal INT0 signal to the MCU. When there is not an interrupt pending, the interrupt vector value will be set to 24h. To clear any interrupt and update the interrupt vector value to the next pending interrupt, the MCU should simply write any value to this register. The interrupt priority is fixed in order, ranging from vector value 1Fh with the highest priority to vector value 00h with the lowest priority. An exception to this priority is the control endpoint EP0 which has top priority.

Bit	7	6	5	4	3	2	1	0
Mnemonic	—	—	IVEC5	IVEC4	IVEC3	IVEC2	IVEC1	IVEC0
Type	R	R	R	R	R	R	R	R
Default	0	0	0	0	0	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION	
7	—	Reserved	Reserved for future use	
6	—	Reserved	Reserved for future use	
5:0	IVEC(5:0)	Interrupt vector	00h = USB out end-point 0 01h = USB out end-point 1 02h = USB out end-point 2 03h = USB out end-point 3 04h = USB out end-point 4 05h = USB out end-point 5 06h = USB out end-point 6 07h = USB out end-point 7 08h = USB in end-point 0 09h = USB in end-point 1 0Ah = USB in end-point 2 0Bh = USB in end-point 3 0Ch = USB in end-point 4 0Dh = USB in end-point 5 0Eh = USB in end-point 6 0Fh = USB in end-point 7	10h = USB setup stage transaction over-write 11h = Reserved 12h = USB setup stage transaction 13h = USB pseudo start-of-frame 14h = USB start-of-frame 15h = USB function resume 16h = USB function suspend 17h = USB function reset 18h = C-port receive data register full 19h = C-port transmit data register empty 1Ah = Reserved 1Bh = Reserved 1Ch = I^2C receive data register full 1Dh = I^2C transmit data register empty 1Eh = Reserved 1Fh = External interrupt input
			20h = DMA Ch.0 interrupt 21h = DMA Ch.1 interrupt 22h - 23h = Reserved	24h = No interrupt pending 25h – 3Fh = Reserved

A.5.7.4 Global Control Register (GLOBCTL – Address FFB1h)

The global control register contains various global control bits for the TAS1020 device.

Bit	7	6	5	4	3	2	1	0
Mnemonic	MCUCLK	XINTEN	P1PUDIS	VREN	RESET	LPWR	P3PUDIS	CPTEN
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION
7	MCUCLK	MCU clock select	The MCU clock select bit is used by the MCU to program the clock frequency to be used for the MCU operation. 0b = 12 MHz and 1b = 24 MHz POR (Power On Reset) value is 0 (12 MHz). Setting this bit to 1 will change MCU clock frequency to 24 MHz. But, once set, only cleared by master reset.
6	XINTEN	External interrupt enable	The external interrupt enable bit is set to a 1 by the MCU to enable the use of the external interrupt input to the TAS1020 device.
5	P1PUDIS	Pull-up resistor disable	If set to 1, disables on-chip pullup resistors on P1 GPIO pins.
4	VREN	VREN	Memory-mapped GPIO pin.
3	RESET	RESET	Memory-mapped GPIO pin.
2	LPWR	Low power mode	The low power mode disable bit is used by the MCU to put the TAS1020 into a semi-low power state. When this bit is cleared to a 0, all USB functional blocks are powered-down. For normal operation, the MCU must set this bit to a 1.
1	P3PUDIS	Pull-up resistor disable	If set to 1, disables on-chip pullup resistors on P3 GPIO pins.
0	CPTEN	Codec port enable	The codec port enable bit is set to a 1 by the MCU to enable the operation of the codec port interface. Note that the codec port interface configuration registers must be fully programmed before this bit is set by the MCU.

A.5.7.5 Memory Configuration Register (MEMCFG – Address FFB0h)

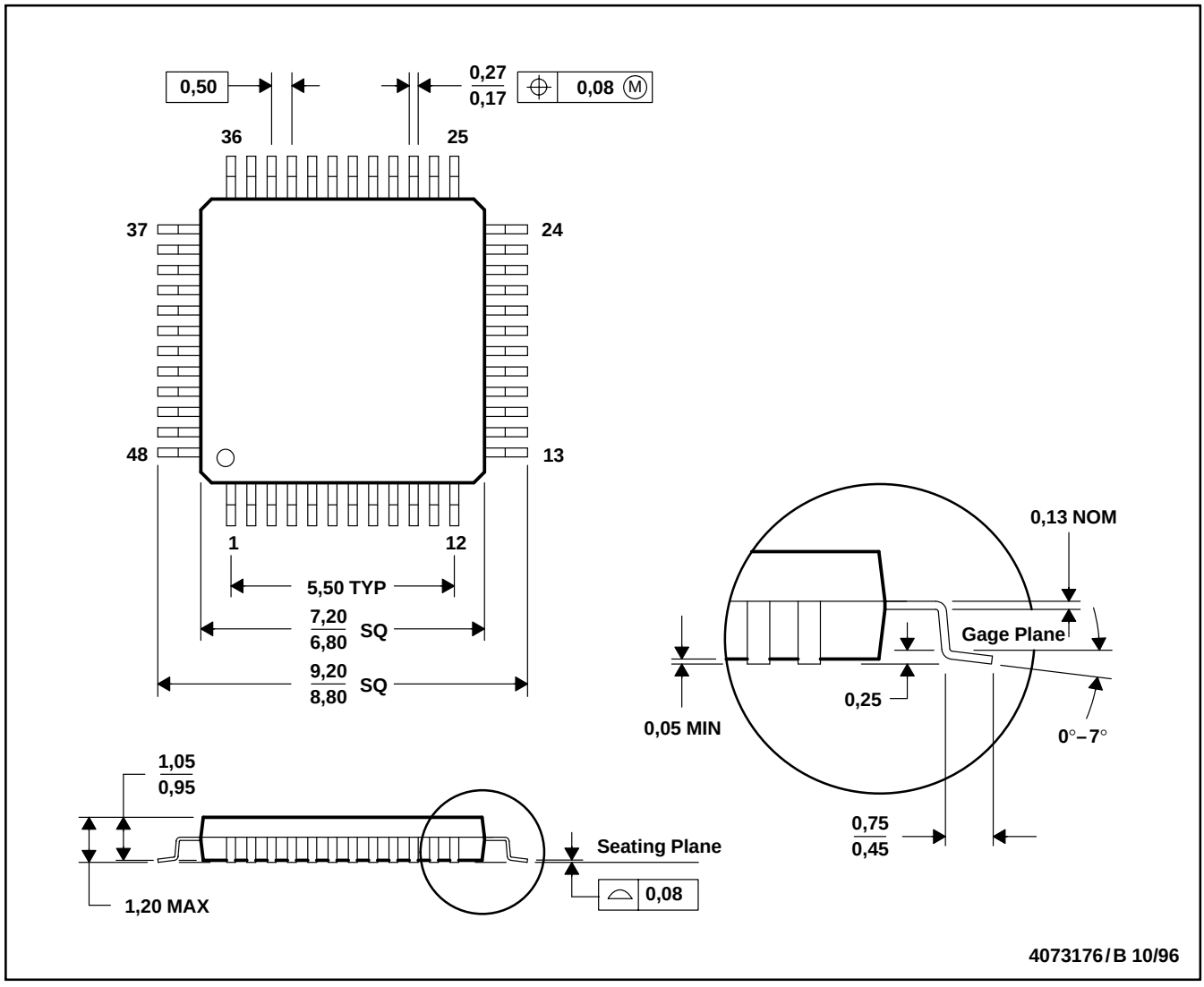
The memory configuration register contains various bits pertaining to the memory configuration of the TAS1020 device.

Bit	7	6	5	4	3	2	1	0
Mnemonic	MEMTYP	CODESZ1	CODESZ0	REV3	REV2	REV1	REV0	SDW
Type	R	R	R	R	R	R	R	R/W
Default	1	0	1	0	0	0	0	0

BIT	MNEMONIC	NAME	DESCRIPTION
7	MEMTYP	Code memory type	The code memory type bit identifies if the type of memory used for the application program code space is ROM or RAM. For the TAS1020, an 8K byte RAM is used and this bit is tied to 1.
6:5	CODESZ	Code space size	The code space size bits identify the size of the application program code memory space. For the TAS1020, an 8K byte RAM is used and these bits are tied to 01b. 00b = 4K bytes, 01b = 8K bytes, 10b = 16K bytes, 11b = 32K bytes
4:1	REV	IC revision	The IC revision bits identify the revision of the IC. 0000b = Rev. -, 0001b = Rev. A, …, 1111b = Rev. F
0	SDW	Shadow the boot ROM	The shadow the boot ROM bit is set to a 1 by the MCU to switch the MCU memory configuration from boot loader mode to normal operating mode. This must occur after completion of the download of the application program code by the boot ROM. Reference SDW protection bit in USBCTL register.

Appendix B
Mechanical Data

PFB (S-PQFP-G48) **PLASTIC QUAD FLATPACK**



NOTES: A. All linear dimensions are in millimeters.
B. This drawing is subject to change without notice.
C. Falls within JEDEC MS-026

PACKAGING INFORMATION

Orderable Device	Status (1)	Package Type	Package Drawing	Pins	Package Qty	Eco Plan (2)	Lead/Ball Finish	MSL Peak Temp (3)
TAS1020PFB	OBSOLETE	TQFP	PFB	48		TBD	Call TI	Call TI

(1) The marketing status values are defined as follows:
ACTIVE: Product device recommended for new designs.
LIFEBUY: TI has announced that the device will be discontinued, and a lifetime-buy period is in effect.
NRND: Not recommended for new designs. Device is in production to support existing customers, but TI does not recommend using this part in a new design.
PREVIEW: Device has been announced but is not in production. Samples may or may not be available.
OBSOLETE: TI has discontinued the production of the device.

(2) Eco Plan - The planned eco-friendly classification: Pb-Free (RoHS) or Green (RoHS & no Sb/Br) - please check http://www.ti.com/productcontent for the latest availability information and additional product content details.
TBD: The Pb-Free/Green conversion plan has not been defined.
Pb-Free (RoHS): TI's terms "Lead-Free" or "Pb-Free" mean semiconductor products that are compatible with the current RoHS requirements for all 6 substances, including the requirement that lead not exceed 0.1% by weight in homogeneous materials. Where designed to be soldered at high temperatures, TI Pb-Free products are suitable for use in specified lead-free processes.
Green (RoHS & no Sb/Br): TI defines "Green" to mean Pb-Free (RoHS compatible), and free of Bromine (Br) and Antimony (Sb) based flame retardants (Br or Sb do not exceed 0.1% by weight in homogeneous material)

(3) MSL, Peak Temp. -- The Moisture Sensitivity Level rating according to the JEDEC industry standard classifications, and peak solder temperature.