Winbond
Electronics Corporation America

ISD-SR3000

Embedded Speech Recognition Processor

Advanced Information

EMBEDDED SPEECH RECOGNITION PROCESSOR FOR COMMAND AND CONTROL APPLICATIONS

The ISD-SR3000 is a complete embedded speech recognition processor. It consists of a speech recognition engine, a speech compression engine, and a recording function. The ISD-SR3000 hardware includes a parallel PISC/DSP core with an optimized instruction set, a flexible CODEC interface, and a serial host controller interface.

The speech recognition engine uses sophisticated Hidden Markov Models (HMMs), which enables recognition of continuous speech and connected digits. An application consists of speaker-independent commands (chosen by the application developer), connected digits and speaker-defined commands. The speaker-defined commands allow users to sore and recognize voicetags that can be used for custom commands or name list management. The speaker-defined commands use the HMMs, providing much more robust performance compared to conventional speaker-dependent commands.

The speaker-independent commands, the audio prompts, and the speaker-defined voicetags are stored in external memory, allowing for maximum application flexibility. Typical storage requirements are 2kB for each predefined command, 2.5kB for each audio prompt, and 3kB for each voicetag, including the model, recording, and data. Application commands are divided into topics (menus), with active vocabulary size governed by the size of the external SRAM. A development system, the ISD-DS3000, is available. The development system includes tools for compiling and sizing commands and prompts, as well as sample C-code for host control program development.

IDEAL EMBEDDED SPEECH APPLICATIONS

- **Accessible appliances**
- **Desktop phones**
- **Home automation**
- **Cordless phones**
- **Information kiosks**
- **Automotive command and control**
- **Cellular car kits**
- **Cellular handsets**
- **Instrumentation control**
- **Internet appliances**

IMPORTANT NOTICE: *This product concept and specifications are preliminary and subject to change without notice. Please contact ISD before using this information in any product design.*

September 2000

ISD · 2727 North First Street, San Jose, CA 95134 · TEL: 408/943-6666 · FAX: 408/544-1787 · http://www.isd.com

ISD-SR3000 PRODUCT FEATURES

Speech Recognition Attributes

- Speech recognition processor optimized for command and control applications
- All speech recognition processing performed on chip
- Supports speaker-independent continuous speech
  - Command vocabulary selected from a large (>100k word) dictionary
  - Number of commands and voicetags determined by external memory availability
  - Multiple topics and finite state grammar supported
  - Connected digit recognition with no domain restrictions

  Supports speaker defined voicetags
  - Phonetic models of user speech input created “on the fly”
  - More robust than typical speaker-dependent word models
  - Used for phone books and customized commands

- Recognition always active
  - Allows for voice activation with keyword command
  - Push-to-talk option for battery applications
- Zero power voicetag storage

- Hidden Markov Models and triphones used to optimize accuracy while maintaining real-time recognition

Recognition Processor Attributes

- Recognition engine optimized for feature extraction and real-time acoustic model search
- Interfaces to µ-Law, A-Law or linear voice CODEC
- Serial interface to host microcontroller
- Single +5V or +3.3V power supply
- Current: 40mA (typical) during active recognition
- Package: 100-pin QFP
- Temperature range: 0 to +70°C

Application Attributes

- **Advanced API enables sophisticated VUI development**
  - **True hands-free control**
  - **Optional activation by voice**
  - **Standard, easy to use interface for voice activated appliances**
  - **Minimizes adaptation time for users**
  - **Accelerated application development by providing standard interface software**

- **Flexible API provides high level commands suitable for a wide variety of applications**

- **Measure accuracy >99% per digit for connected digit strings**

Figure i: Stand-alone Speech Recognition System Diagram

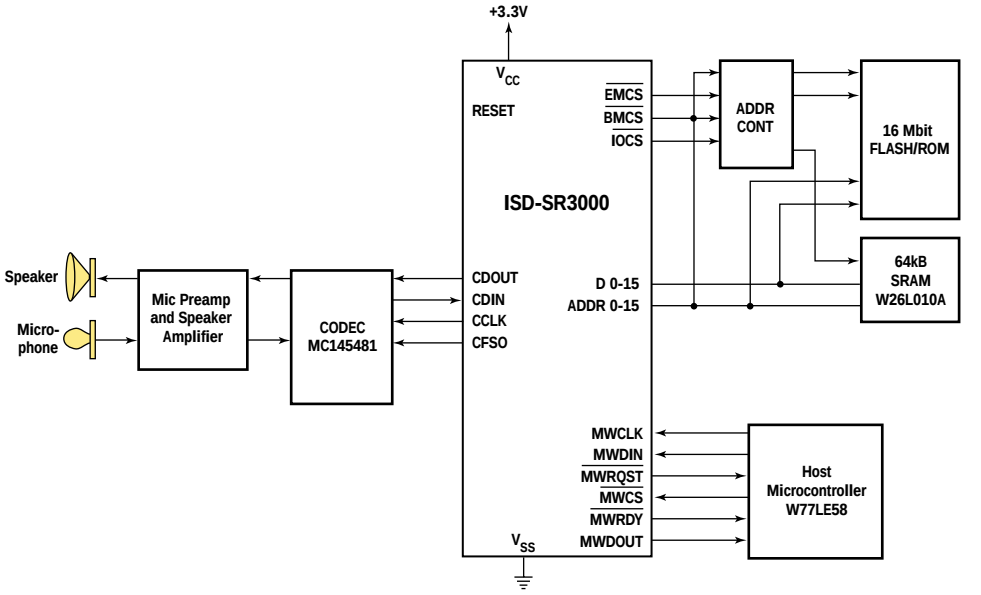


Table of Contents

Chapter 1—Hardware

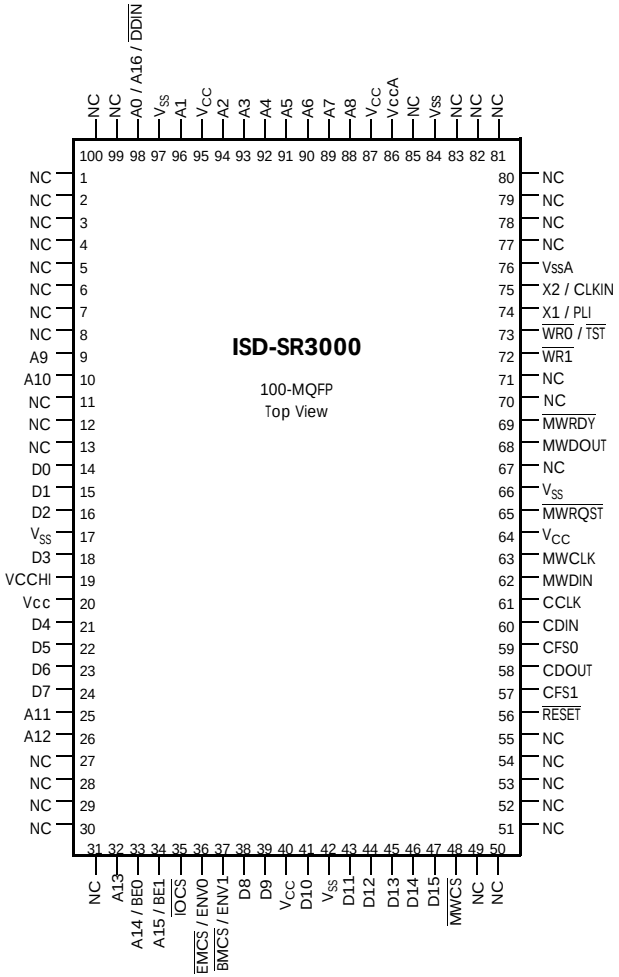
1.1 PIN ASSIGNMENT

The following sections detail the pins of the ISD-SR3000 processor. Slashes separate the names of signals that share the same pin.

1.1.1 PIN-SIGNAL ASSIGNMENT

Table 1-1 shows all the pins and the signals that use them in different configurations. It also shows the type and direction of each signal.

Figure 1-1: 100-MQFP Package Connection Diagram



***Note**: Pins marked NC should not be connected.*

Table 1-1: ISD-SR3000 Pin Signal Assignment

Pin Name	Signal Name	Type	Description
A(0:15)	A(0:16)	Output	Address bits 0 through 16
CCLK	CCLK	I/O	Codec Master/slave Clock
$\overline{\text{BMCS}}$	$\overline{\text{BMCS}}$	Output	Base memory chip select
$\overline{\text{BMCS}}$/ ENV1	BMCS	I/O	Base memory chip select Environment Select
CDIN	CDIN	Input	Data Input from Codec
CDOUT	CDOUT	Output	Data Output to Codec
CFS0	CFS0	I/O	Codec 0 Frame Synchronization
CFS1	CFS1	Output	Codec 1 Frame Synchronization
D(0:7)	D(0:7)	I/O	Data bits 0 through 7
$\overline{\text{EMCS}}$/ ENV0	$\overline{\text{EMCS}}$	Output[2]	Expansion Memory Chip Select
$\overline{\text{EMCS}}$/ ENV0	ENV0	Input[1]	Environment Select
MWCLK	MWCLK	Input[3]	MICROWIRE Clock
$\overline{\text{MWCS}}$	$\overline{\text{MWCS}}$	Input[3]	MICROWIRE Chip Select
MWDIN	MWDIN	Input[3]	MICROWIRE Data Input
MWDOUT	MWDOUT	Output	MICROWIRE DATA Output
$\overline{\text{MWRDY}}$	$\overline{\text{MWRDY}}$	Output	MICROWIRE Ready
$\overline{\text{MWRQST}}$	$\overline{\text{MWRQST}}$	Output	MICROWIRE Request Signal
$\overline{\text{RESET}}$	$\overline{\text{RESET}}$	Input[3]	Reset
$\overline{\text{TST}}$	$\overline{\text{TST}}$	Input	Test pin
V_{CC}	V_{CC}	Power	3.3 V power supply pin
$V_{CC}A$	$V_{CC}A$	Power	3.3 V analog circuitry power supply pin
$V_{CC}HI$	$V_{CC}HI$	Power	5 V power supply pin. Connect to V_{CC} if 3.3 V power supply is used.
V_{SS}	V_{SS}	Power	Ground for on-chip logic and output drivers
$V_{SS}A$	$V_{SS}A$	Power	Ground for on-chip analog circuitry
X1	X1	Oscillator	Crystal Oscillator Interface
X2/CLKIN	X2	Oscillator	Crystal Oscillator Interface

1. *TTL1 output signals provide CMOS levels in the steady state, for small loads.*
2. *Input during reset. CMOS level input.*
3. *Schmitt trigger input.*

1.2 DESCRIPTION

This section provides details of the functional characteristics of the ISD-SR3000 processor. It is divided into the following sections:

- Resetting
- Clocking
- Power-Down Mode
- Power and Grounding
- Memory Interface
- Codec Interface

1.2.1 RESETTING

The $\overline{\text{RESET}}$ pin is used to reset the ISD-SR3000 processor.

On application of power, $\overline{\text{RESET}}$ must be held low for at least t_{pwr} after V_{CC} is stable. This ensures that all on-chip voltages are completely stable before operation. Whenever $\overline{\text{RESET}}$ is applied, it must also remain active for not less than t_{RST}, see Table 1-11 and Table 1-12. During this period, and for 100 ms after, the $\overline{\text{TST}}$ signal must be high. This can be done with a pull-up resistor on the $\overline{\text{TST}}$ pins

The value of $\overline{\text{MWRDY}}$ is undefined during the reset period, and for 100 ms after. The microcontroller should either wait before polling the signal for the first time, or the signal should be pulled high during this period.

Upon reset, the ENV0 and ENV1 input pins are sampled to determine the operating environment. During reset, the $\overline{\text{EMCS}}$/ENV0 and $\overline{\text{BMCS}}$/ENV1 pins are used for the ENV0 and ENV1 inputs signals respectively. An internal pull-up resistor sets ENV0 and ENV1 to 1. An external 5.1k Ω resistor connected to V_{SS} can be used to set them to 0.

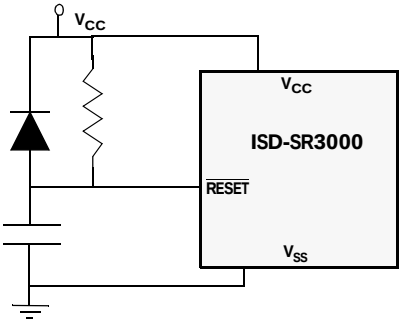
After reset, the same pin is used for $\overline{\text{EMCS}}$.

SYSTEM LOAD ON ENV0

For any load on the ENV0 pin, the voltage should not drop below V_{ENVh}. Therefore, apply a load on the ENV0 pin that ensures the voltage does not go below 2.4V.

Figure 1-2 shows a recommended circuit for generating a reset signal when the power is turned on.

Figure 1-2: Recommended Power-On Reset Circuit



1.2.2 CLOCKING

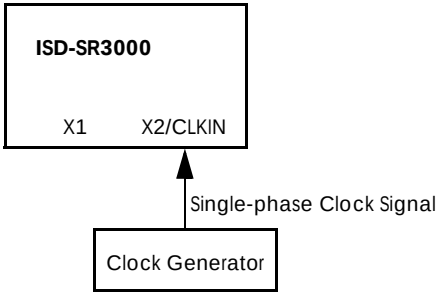
The ISD-SR3000 processor provides an internal oscillator that interacts with an external clock source through the X1 and X2/CLKIN pins. Either an external single-phase clock signal or a crystal oscillator may be used as the clock source.

EXTERNAL SINGLE-PHASE CLOCK SIGNAL

If an external single-phase clock source is used, it should be connected to the CLKIN signal as shown in Figure 1-3, and should conform to the voltage-level requirements for CLKIN stated in "Electrical Characteristics" on page 1-19.

***Note**: the CLKIN signal is not 5V tolerant.*

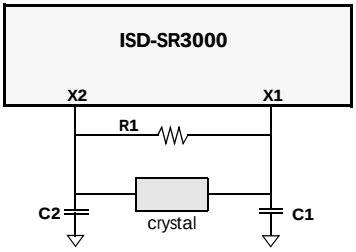
Figure 1-3:External Clock Source



CRYSTAL OSCILLATOR

A crystal oscillator is connected to the on-chip oscillator circuit via the X1 and X2 signals, as shown in Figure 1-4.

Figure 1-4:Connections for an External Crystal Oscillator



Keep stray capacitance and inductance, in the oscillator circuit, as low as possible. The crystal resonator, and the external components, should be as close to the X1 and X2/CLKIN pins as possible, to keep the trace lengths in the printed circuit to an absolute minimum.

You can use crystal oscillators with maximum load capacitance of 20pF, although the oscillation frequency may differ from the crystal's specified value.

The following table lists the components in the crystal oscillator circuit.

Table 1-2: Components of Crystal Oscillator Circuit

Component	Values	Tolerance
Crystal Resonator	4.096MHz	
Resistor R1	10MΩ	5%
Capacitors C1, C2	33pF	20%

1.2.3 POWER-DOWN MODE

Power-down mode is useful during a power failure or in a power-saving model when the power source for the processor is a backup battery or in battery-powered devices, while the processor is in idle mode.

In power-down mode, the clock frequency of the ISD-SR3000 processor is reduced and some of the processor modules are deactivated. As a result, the ISD-SR3000 consumes considerably less power than in normal-power mode. Recognition is not active during power-down mode, so the ISD-SR3000 must return to normal power mode to resume speech recognition.

***Note**: In power-down mode all the chip select signals, CS0 to CS3, are set to 1. To guarantee that there is no current flow from these signals to the Flash devices, the power supply to these devices must not be disconnected*.

The ISD-SR3000 stores voicetags and all memory management information in Flash memory. When Flash memory is used for memory management, power does not need to be maintained to the processor to preserve stored voicetags.

To keep power consumption low during power-down mode, the $\overline{\text{RESET}}$, $\overline{\text{MWCS}}$, MWCLK and MWDIN signals should be held above V_{CC} – 0.5 V or below V_{SS} + 0.5 V.

1.2.4 POWER AND GROUNDING

POWER PIN CONNECTIONS

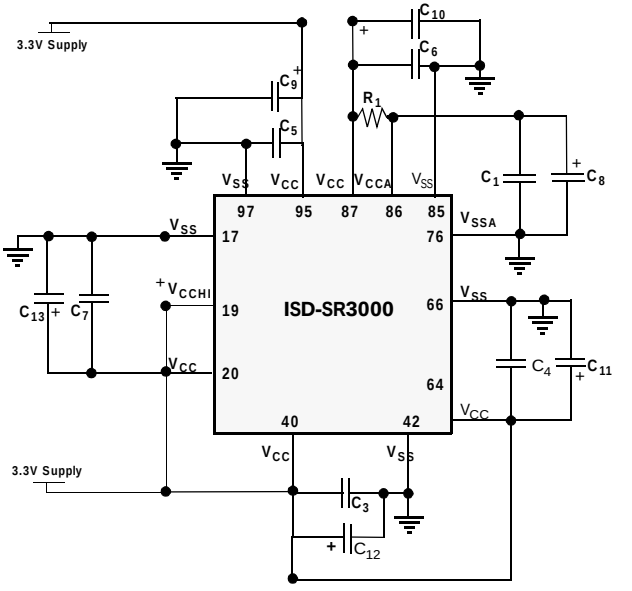
The ISD-SR3000 can operate over two supply voltage ranges 3.3V ±10% and 5V ±10%. The five power supply pins (V_{CC}, V_{SS}, $V_{CC}A$, $V_{SS}A$ and $V_{CC}HI$) must be connected as shown in Figure 1-5 when operating in a 3.3 V environment, and as shown in Figure 1-6 when operating in a 5V environment. Failure to correctly connect the pins may result in damage to the device.

The capacitor and resistor values are given in Table 1-3.

Table 1-3: Components of Supply Circuit

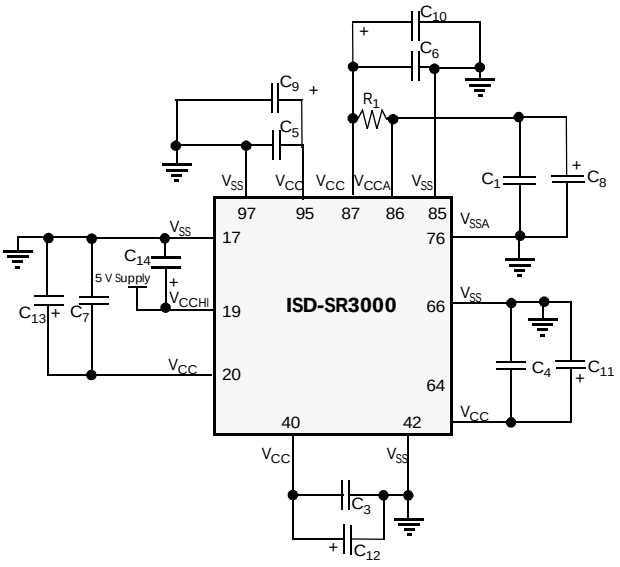
Component	Values	Tolerance
Resistor R1	10Ω	5%
Capacitors C1, C2, C3, C4, C5, C6, C7	0.1µF Ceramic	20%
Capacitors C8, C9, C10, C11, C12, C13, C14	1µF Tantalum	20%

Figure 1-5: 3. 3V Power Connection Diagram



***Note**: all Vcc pins in Figure 1-5 should be connected to a 3.3V supply*

Figure 1-6: 5V Power Connection Diagram



For optimal noise immunity, the power and ground pins should be connected to V_{CC} and the ground planes, respectively, on the printed circuit board. If V_{CC} and the ground planes are not used, single conductors should be run directly from each V_{CC} pin to a power point, and from each GND pin to a ground point. Avoid daisy-chained connections. The ISD-SR3000 does not perform recognition in power-down mode.

When you build a prototype, using wire-wrap or other methods, solder the capacitors directly to the power pins of the ISD-SR3000 processor socket, or as close as possible, with very short leads.

1.2.5 MEMORY INTERFACE

The ISD-SR3000 supports flash or ROM devices for storing voicetags, vocabulary, prompts, and acoustic models, such that power can be removed from the system without any data loss. The ISD-SR3000 supports AMD-compatible parallel flash devices, depending upon the required amount of memory. The recommended flash size is 16Mb, but larger sizes may be required for some applications. Organization can be 1Mb X 16 or 2Mb X 8. The recommended device is Am29LV160D, which is organized as 1Mb X 16, and operates on a 3V power supply.

1.2.6 THE CODEC INTERFACE

The ISD-SR3000 provides an on-chip interface for analog and digital telephony, supporting master and slave CODEC interface modes. In master mode, the ISD-SR3000 controls the operation of the CODEC for use in analog telephony or stand-alone applications. In the slave mode, the ISD-SR3000 CODEC interface is controlled by an external source. This mode is

used in digital telephony (i.e., ISDN or DECT lines). The slave mode is implemented with respect to IOM-2™/CGI specifications.

See Table 1-4 for CODEC options for the ISD-SR3000 (ISD supports compatible CODECS in addition to those listed below).

The CODEC interface supports the following features:

- Master Mode or Slave Mode.
- 8- or 16-bit channel width.
- Long (variable) or short (fixed) frame protocol.
- Single or double bit clock rate.
- Single or dual channel CODECS
- One or two CODECS
- Multiple clock and sample rates.
- One or two frame sync signals

This CODEC interface uses five signals: CDIN, CDOUT, CCLK, CFSO, and CFS1. The CDIN, CDOUT, CCLK, and CFSO pins are connected to the first CODEC. The second CODEC (for speakerphone applications) is connected to CDIN, CDOUT, CCLK, and CFS1 pins. Data is transferred to the CODEC (for speakerphone applications) through the CDOUT output pin. Data is read from the CODEC through the CDIN input pin. The CCLK and CFSO pins are output in Master mode and input in Slave mode. The CFS1 is an output pin.

SHORT FRAME PROTOCOL

When the short frame protocol is configured, eight or sixteen data bits are exchanged with each CODEC in each frame (i.e., the CFSO cycle). Data transfer begins when CFSO is set to 1 for one CCLK cycle. The data is then transmitted, bit by bit, via the CDOUT pin. Concurrently, the received data is shifted in through the CDIN pin. Data is shifted one bit per CCLK cycle. After the last bit has been shifted, CFS1 is set to 1 for one CCLK cycle. Then, the data from the second CODEC is shifted out via CDOUT, concurrently with the inward shift of the data received via CDIN.

LONG FRAME PROTOCOL

When long frame protocol is configured, eight or sixteen data bits are exchanged with each CODEC, as for the short frame protocol. However, for the long frame protocol, data transfer starts by setting CFSO to 1 for eight or sixteen CCLK cycles. Short or long frame protocol is available in both Master and Slave modes.

Table 1-4: Typical Supported CODEC Devices

Manufacturer	CODEC Device Name	Characteristics	Operating Voltage	Conversion Type
National Semiconductor	TP3054	Single CODEC	5V	µ-Law
OKI	MSM7533V	Dual CODEC	5V	µ-Law, A-Law
Macronix	MX93002FC	Dual rail CODEC	5V	µ-Law
Lucent	T7503	Dual CODEC	5V	µ-Law
Motorola	MC145481	Single CODEC	3V	µ-Law

Channel Width

The CODEC interface supports both 8-bit and 16-bit channel width in Master and Slave modes. Figure 1 shows how the CODEC interface signals behave when short frame protocol is configured.

Slave Mode

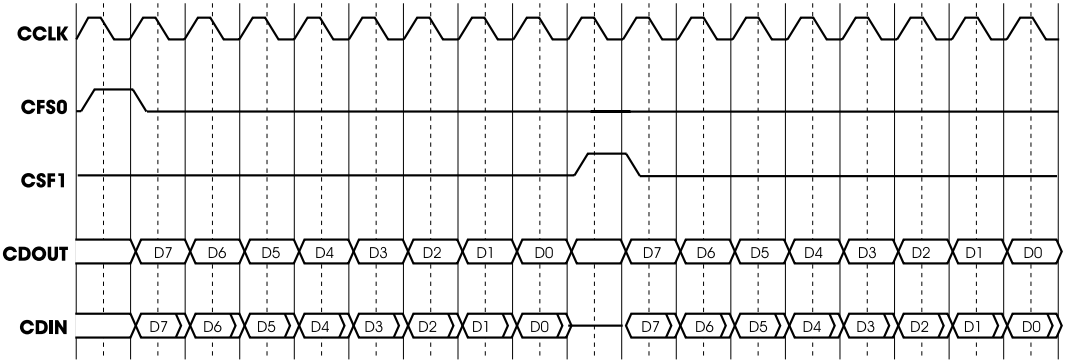
The ISD-SR3000 supports digital telephony applications including DECT and ISDN by providing a Slave mode of operation. In Slave mode operation, the CCLK signal is input to the SR-3000 and controls the frequency of the CODEC interface operation. The CCLK may be any frequency between 500kHz and 4MHz. Both long and short frame protocols are supported with only the CFS1 output signal width affected. The CFSO input signal must be a minimum of one CCLK cycle.

In slave mode, a double clock bit rate feature is available as well. When the CODEC interface is configured to double clock bit rate, the CCLK input signal is divided internally by two and the resulting clock used to control the frequency of the CODEC interface operation.

Table 1-5: Typical CODEC Applications

Application	CODEC Type	No. of Channels	Master/ Slave	Channel Width (No.Bits)	Long/ Short Frame Protocol	Bit Rate	CCLK Freq. (MHz)	Sample Rate (Hz)	No. of Frame Syncs
Analog µ-Law	single	1	Master	8	short or long	1	2.048	8000	1
Linear	single	1	Master	16	short	1	2.048	8000	1

Figure 1-7: Codec Protocol-Short Frame—8-Bit Channel Width



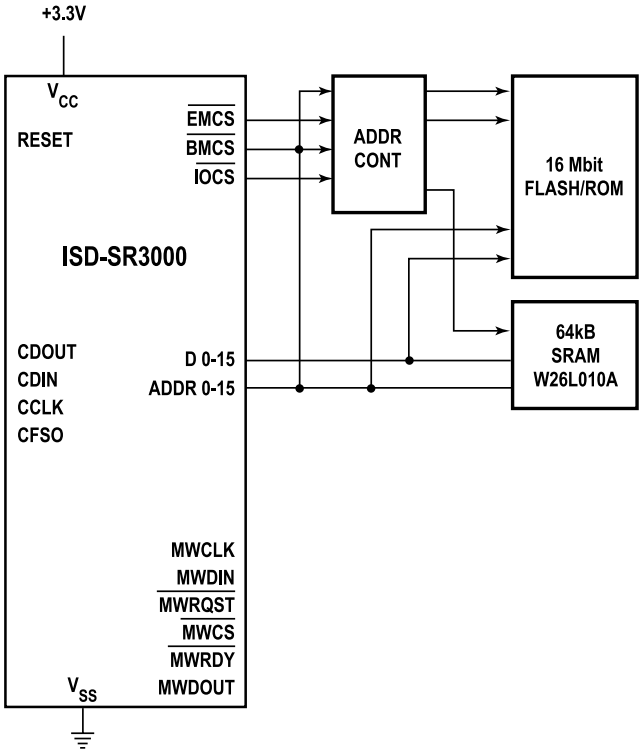
1.2.7 EXPANSION MEMORY ADDRESSING

FUNCTIONAL OVERVIEW

The ISD-SR3000 requires memory subsystems with enhanced physical address space and a paging scheme. The typical memory requirement for the system is 2Mbytes of ROM to store the application database, and an additional 64Kbytes of SRAM to enable the software to perform sophisticated search on the database.

The diagram below describes the connection of the ISD-SR3000 to external memory space of 2Mbytes. This connection is typical for a recognition application. The external ROM holds the acoustic models (1Mbyte). The flash holds all the topics tables (vocabulary), speech prompts, and voicetag recordings that support the specific application (1Mbyte). The SRAM is used by the ISD-SR3000 as extension for the memory and is used to carry out the HMM algorithm (64Kbytes).

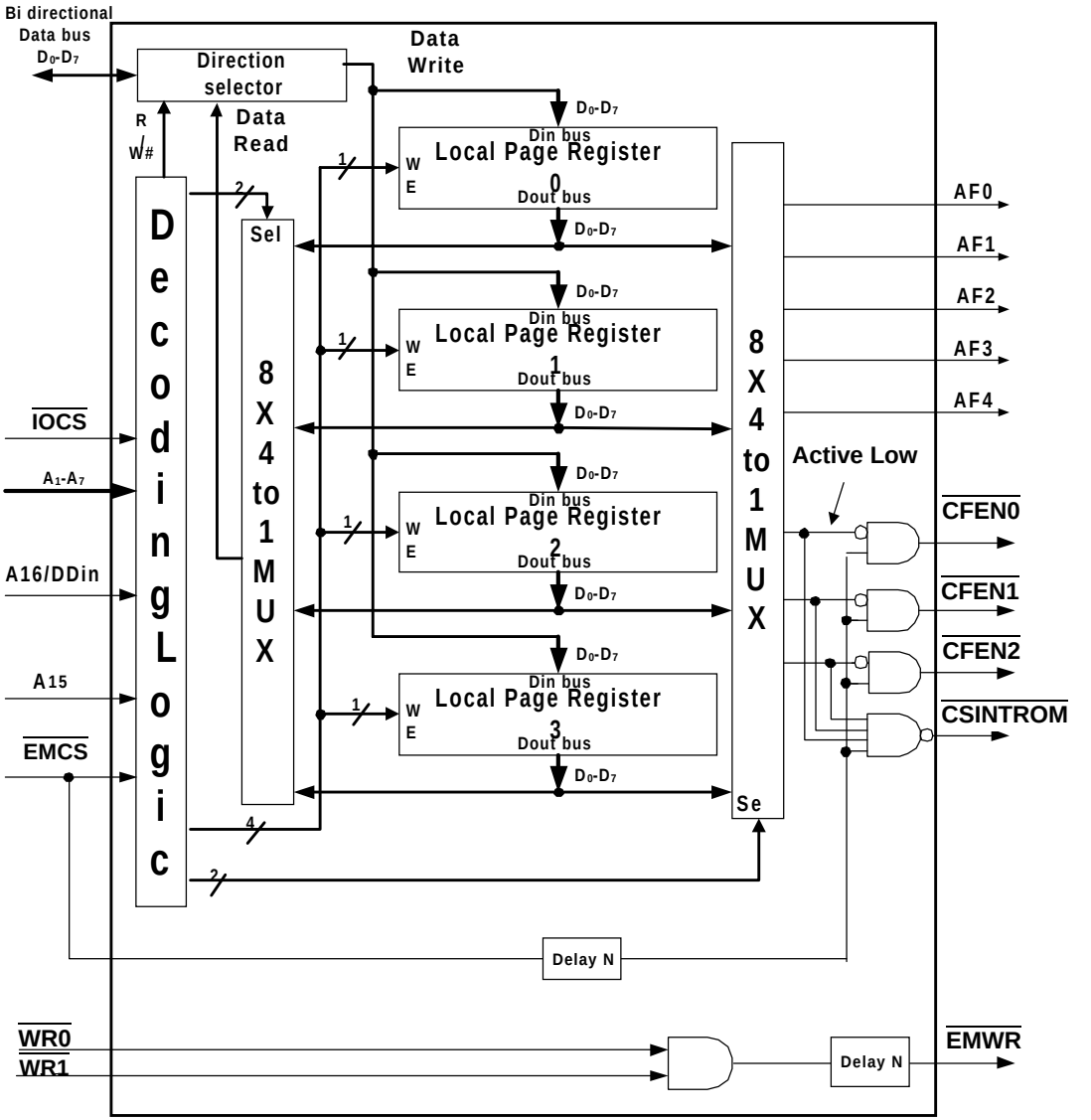
Figure 1-8: ISD-SR3000 connection for external memory space of 2Mbytes



***Note**: If the application does not support user-created voicetags, the Flash can be replaced with a ROM.*

BLOCK DIAGRAM

Figure 1-9: Address Extension Lines Block Diagram



INTERFACE SIGNALS

Table 1-6: ISD-SR3000 Extended Address Lines Interface Signals

Signal Name	Type	Source Unit/Signal	Description
D[7:0]	I/O	ISD-SR3000	Data Bus
A[7:1]	I	ISD-SR3000	Address Bus
$\overline{\text{IOCS}}$	I	ISD-SR3000	I/O Expansion Chip Select to access the I/O registers
$\overline{\text{EMCS}}$	I	ISD-SR3000	Expansion Memory Chip select
DDIN / A16	I	ISD-SR3000	Serve as direction in I/O operation of SR3000; otherwise is bit 16 of address bus
A15	I	ISD-SR3000	Address line
$\overline{\text{WR}}$[0:1]	I	ISD-SR3000	Write signal for external memory
AF[4:0]	O	To external Extended memory	Extended Address bus
$\overline{\text{CFEN}}$	O	To external Extended memory	Extended Memory CS could use for Flash CS
$\overline{\text{CREN}}$	O	To external Extended memory	Extended Memory CS could use for ROM CS
$\overline{\text{CSEN}}$	O	To external Extended memory	Extended Memory CS could use for RAM CS
$\overline{\text{EMWR}}$	O	To external Extended memory	Write signal to external SRAM
$\overline{\text{CSINTROM}}$	O	To internal ROM	Chip select for internal ROM (Added ROM for DSPM)

INTERNAL DESCRIPTION

Detailed Description

The Memory Extension unit contains four page registers to perform memory decoding and control functions, memory mapping and paging.

The page registers are mapped into the I/O Expansion space defined by an active $\overline{\text{IOCS}}$ signal *(*Addresses TBD*)* from the ISD-SR3000. Only address lines A1-A7 are used, while A16 becomes DDIN (data direction signal; low for reads). The registers are both writable and readable.

The software accesses the page registers using store or load operations to the respective I/O ports from the ISD-SR3000. The decoding logic generates the proper signals to write and read the page registers, which are accessed by the ISD-SR3000 when store and load commands are executed.

The paging operation works as follows:

- When the ISD-SR3000 accesses the expansion memory space (indicated by $\overline{\text{EMCS}}$ going active), the decoding logic will select one out of the four page registers, based on the state of A[16:15].
- The contents of the selected page register will be use to generate the proper Chip select and the extended address lines AF[4:0].
- Bits 0 to 4 of the page register connect to AF4 to AF0 respectively.
- Chip Select field is bits 5 to 7 and are used to generate the Chip Select lines
- The chip select signal $\overline{\text{CFEN}}$ will assert low if and only if the value inside the Page register, which is driving the bus, has a logic 1 at the bit 5 and the signal $\overline{\text{EMCS}}$ is low.
- The chip select signals $\overline{\text{CFRN}}$ will assert low if and only if the value inside the Page register, which is driving the bus, has a logic 1 at bit 6 and the signal $\overline{\text{EMCS}}$ is low.
- The chip select signals $\overline{\text{CFSN}}$ will assert low if and only if the value inside the Page register which is driving the bus has a logic 1 at bit 7 and the signal $\overline{\text{EMCS}}$ is low.
- When bits 5, 6, and 7 on the page register are all at logical 0 and the $\overline{\text{EMCS}}$ is asserted low, the signal CSINROM is asserted low.

Table 1-7: Paging Operation

Local Page Register	7	6	5	4	3	2	1	0
	$\overline{\text{CFSN}}$	$\overline{\text{CFRN}}$	$\overline{\text{CFEN}}$	AF[4]	AF[3]	AF[2]	AF[1]	AF[0]

The mapping of the address from the ISD-SR3000 to the expansion memory is:

- Address 0x10000-0x17fff use page register 0.
- Address 0x18000-0x1ffff use page register 1.
- Address 0x20000-0x27fff use page register 2.
- Address 0x28000-0x2ffff use page register 3.

Table 1-8: Address Mapping from ISD-SR3000 to Expansion Memory

	Page Reg #0	Page Reg #1	Page Reg #2	Page Reg #3
A15	0	0	1	1
A16	0	1	0	1

The signal $\overline{\text{EMWR}}$ is write strobe to external memory. The external SRAM chip can support byte or word access. In order to support this mode 3 signals are used: LB to support writing to low byte of the bus, UB to support writing the high byte of the bus, and WR strobe signal. Since the ISD-SR3000 block can supply WR[0-1] signal for the low or high byte of the bus , the WR strobe signal needs to be generated, which is the logical AND operation of the two signals. (for more details, refer to the Winbond W26L010A data sheet).

Note that the block is operational only if the MCFG register of the ISD-SR3000 is programmed in extension memory mode, 64K x 16 RAM.

The following figures illustrate the timing of Extended Memory Read and Write cycles. Note the timing of AF[7:5], which result of the qualification of the respective page register outputs with the delayed EMCS strobe.

Flow Charts

The software on the SR3000 that uses the SR3000 Extended address unit should do the following:

1. Set the MCFG.EMC register to 64kbytes x 16 bits RAM mode. Mode 111 at MCFG.EMC.
2. Set the page register to page xxxxxxxx. For example to the internal ROM use 0x00.
3. To read or write from external memory use load or store operation from the expansion memory. For the internal ROM the address should be 0x10000 – 0x17fff.

Unit Register Space

The following registers are accessed at addresses <0xFBF2 - 0XFBF*> in the SR3000 I/O address space for write operation and addresses <0xFF9e - 0XFFbe> in the SR3000 I/O address space for read operation.

Table 1-9: Unit Register Space

Register Name	Address	Size (bits)	Description	R/W	Reset Value
Local Page register 0	0xFBF2 0xff8e	8	Page register	R/W	0
Local Page register 1	0xFBF4 0xff9e	8	Page register	R/W	0
Local Page register 2	0xFBF8 0xffae	8	Page register	R/W	0
Local Page register 3	0xFBFA 0xffbe	8	Page register	R/W	0

R – read only
W – Write only
R/W read and write
RAC read auto clear for interrupt status registers

Reset Activity

On reset all the registers should be set to their default values 0 and all chip select signals should be inactive.

Power Save Features

On power down, all chip select signals must be high to ensure minimal power consumption by the external peripheral.

Timing Diagrams

Figure 1-10: Read operation to externals memory using the SR3000 extended memory unit

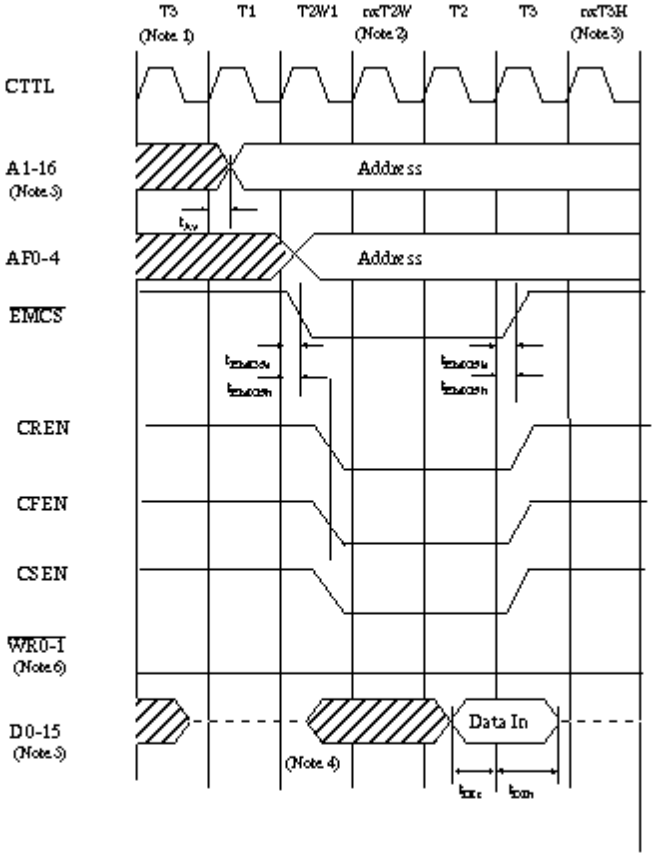
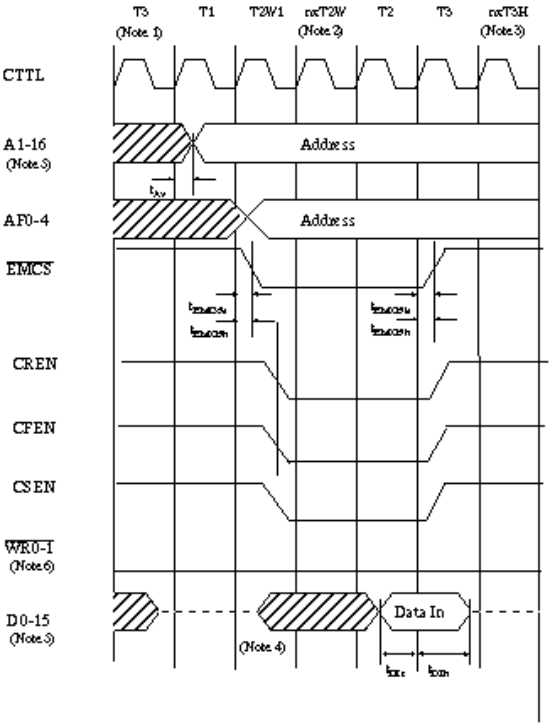


Figure 1-11: Write operation to externals memory using the SR3000 extended memory unit.



SPECIAL SOFTWARE REQUIREMENTS

Note that the block is operational only if the MCFG register of the SR3000 is programmed in extension memory mode, 64K x 16 RAM.

MICRO ARCHITECTURE AND DESIGN CONSIDERATIONS

The design should take into account that the some of signals are connected to external memory units. Hence the timing diagram should comply with external parts. Required wait states will be supported using the EXPANSION MEMORY WAIT STATE mechanism.

1.3 SPECIFICATIONS

1.3.1 ABSOLUTE MAXIMUM RATINGS

Storage temperature	-65°C to +150°C
Temperature under bias	0°C to 70°C
All input or output voltages, with respect to GND	–0.5 V to +6.5 V

1.3.2 ELECTRICAL CHARACTERISTICS

T_A = 0°C to +70°C, V_{CC} = 5V ±10%, GND = 0V

***Note**: Absolute maximum ratings indicate limits beyond which permanent damage may occur. Continuous operation at these limits is not intended; operation should be limited to the conditions specified below.*

Table 1-10: Electrical Characteristics
(All Parameters with Reference to V_{CC} = 3.3V)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
C_X	X1 and X2 capacitance[1]			17.0		pF
I_{CC1}	Active supply current	Normal operation mode, running speech applications[2]		40.0	80.0	mA
I_{CC2}	Standby supply current	Normal operation mode, DSPM idle[2]		30.0		mA
I_{CC3}	Power-down Mode Supply Current	Power-down Mode[2,3]			12	mA
I_L	Input Load Current	$0 V \leq V_{IN} \leq V_{CC}$	–5.0		5.0	µA
I_O (Off)	Output Leakage Current (I/O pins in input mode)	$0 V \leq V_{OUT} \leq V_{CC}$	–5.0		5.0	µA
t_{CASa}	$\overline{CAS}$ Active	After R.E. CTTL, T1 or T2W3			12.0	nsec
t_{CASh}	$\overline{CAS}$ Hold	After R.E. CTTL	0.0			
t_{CASia}	$\overline{CAS}$ Inactive	After R.E. CTTL, T3 or TERF			12.0	nsec
t_{CASLw}	DRAM, PDM, $\overline{CAS}$ Width	At 0.8 V, both edges	600.0			
t_{WRa}	$\overline{WR0}$ Active	After R.E. CTTL, T1			t_{CTp}/ 2 + 2	
t_{WRCSh}	$\overline{WR0}$ Hold after $\overline{EMCS}$[4]	R.E. $\overline{EMCS}$ R.E. to R.E. $\overline{WR0}$	10.0			

Table 1-10: Electrical Characteristics
(All Parameters with Reference to V_{CC} = 3.3V)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
t_{WRh}	$\overline{WR0}$ Hold	After R.E. CTTL	$t_{CTp}/2 - 6$			
t_{WRia}	$\overline{WR0}$ Inactive	After R.E. CTTL, T3			$t_{CTp}/2 + 2$	
V_{ENVh}	ENV0 Input, high voltage		2.0			V
V_{Hh}	CMOS Input with hysteresis, logical 1 input voltage		2.1			V
V_{Hl}	CMOS Input with hysteresis, logical 0 input voltage				0.8	V
V_{Hys}	Hysteresis Loop Width[1]		0.5			V
V_{IH}	TTL Input, logical 1 input voltage		2.0		V_{CC} + 0.5	V
V_{IL}	TTL Input, logical 0 input voltage		–0.5		0.8	V
V_{OH}	Logical 1 TTL, output voltage	I_{OH} = –0.4 mA	2.4			V
V_{OHWC}	$\overline{EMCS}$ Logical 1, output voltage	I_{OH} = –50 µA[5]	V_{CC} – 0.2			V
V_{OL}	Logical 0, TTL output voltage	I_{OL} = 4 mA			0.45	V
		I_{OL} = 50 µA[5]			0.2	V
V_{OLWC}	$\overline{EMCS}$ Logical 0, output voltage	I_{OL} = 50 µA[5]			0.2	V
V_{XH}	CLKIN Input, high voltage	External clock[6]	2.0			V
V_{XL}	CLKIN Input, low voltage	External clock[6]			0.8	V

5. *Guaranteed by design.*

6. *I_{OUT} = 0, T_A = 25°C, V_{CC} = 3.3 V for V_{CC} pins and 3.3 V or 5 V on V_{CCHI} pins, operating from a 4.096 MHz crystal and running from internal memory with Expansion Memory disabled.*

7. *All input signals are tied to 0 (above V_{CC} – 0.5 V or below V_{SS} + 0.5 V), except ENV0, which is tied to V_{CC}.*

8. *Measured in power-down mode. The total current driven, or sourced, by all the ISD-SR3000 processor's output signals is less than 50 µA.*

9. *Guaranteed by design, but not fully tested.*

10. *CLKIN signal is not 5V tolerant.*

1.3.3 SWITCHING CHARACTERISTICS

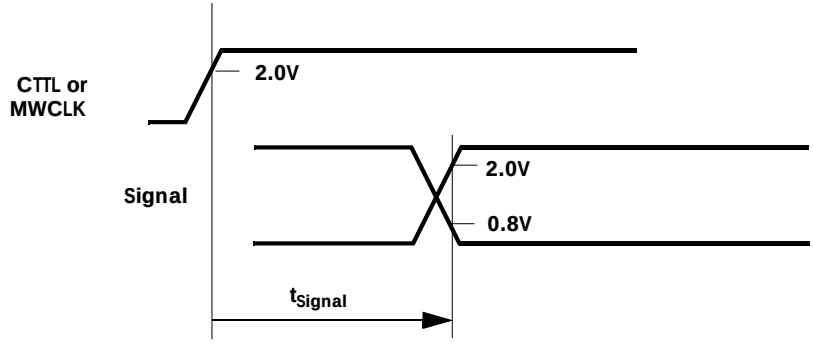
DEFINITIONS

All timing specifications in this section refer to 0.8V or 2.0V on the rising or falling edges of the signals, as illustrated in Figure 1-12 through Figure 1-18, unless specifically stated otherwise.

Maximum times assume capacitive loading of 50pF. CLKIN crystal frequency is 4.096MHz.

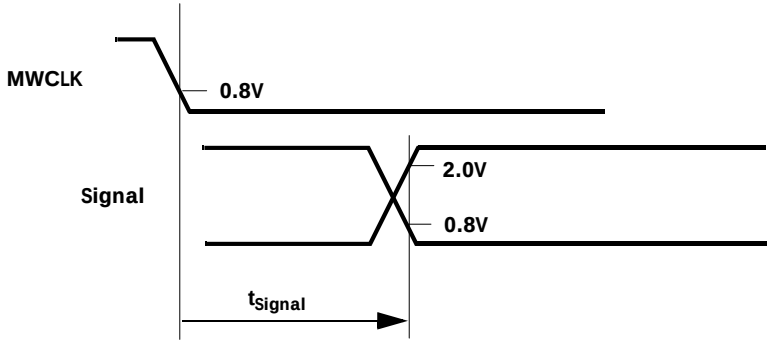
***Note**: CTTL is an internal signal and is used as a reference to explain the timing of other signals. See Figure 1-27*.

Figure 1-12: Synchronous Output Signals (Valid, Active and Inactive)



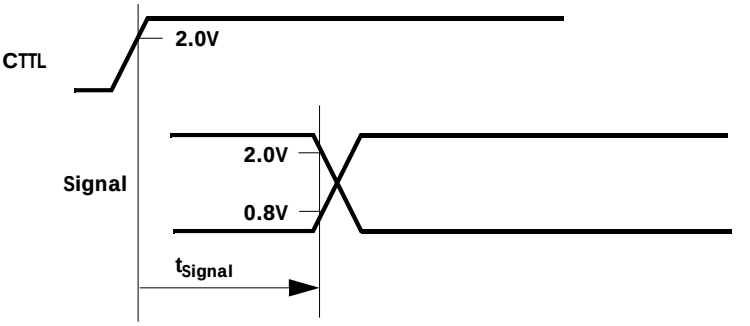
***Note**: Signal valid, active or inactive time, after a rising edge of CTTL or MWCLK.*

Figure 1-13: Synchronous Output Signals (Valid)



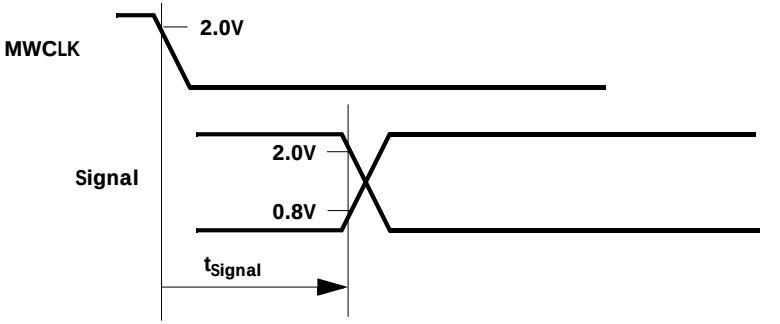
***Note:** Signal valid time, after a falling edge of MWCLK.*

Figure 1-14: Synchronous Output Signals (Hold), after rising edge of CTTL



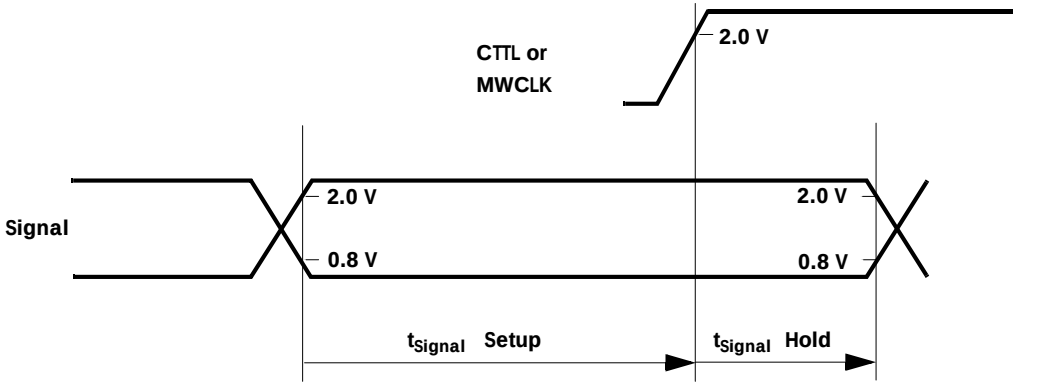
***Note**: Signal hold time, after a rising edge of CTTL.*

Figure 1-15: Synchronous Output Signals (Hold), after falling edge of MWCLK



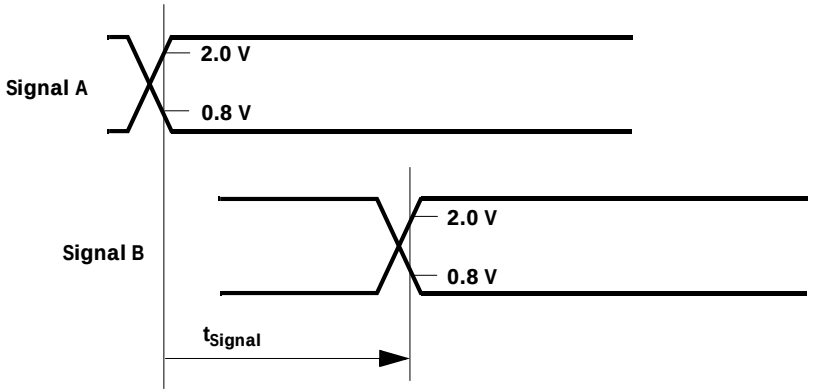
***Note**: Signal hold time, after a falling edge of MWCLK.*

Figure 1-16: Synchronous Input Signals



***Note:** Signal setup time, before a rising edge of CTTL or MWCK, and signal hold time after a rising edge of CTTL or MWCK*

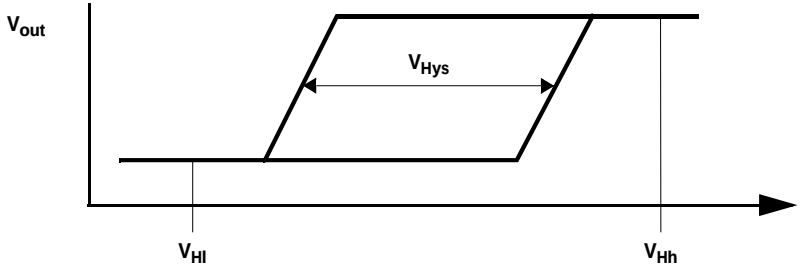
Figure 1-17: Asynchronous Signals



***Note**: Signal B starts after rising or falling edge of signal A.*

The $\overline{\text{RESET}}$ has a Schmitt trigger input buffer. Figure 1-18 shows the input buffer characteristics.

Figure 1-18: Hysteresis Input Characteristics



1.3.4 SYNCHRONOUS TIMING TABLES

In this section, R.E. means Rising Edge and F.E. means Falling Edge.

Table 1-11: Output Signals

Symbol	Figure	Description	Reference Conditions	Min (ns)	Max (ns)
t_{Ah}		Address Hold	After R.E. CTTL	0.0	
t_{Av}		Address Valid	After R.E. CTTL, T1		9.0
t_{CCLKa}		CCLK Active	After R.E. CTTL		12.0
t_{CCLKh}		CCLK Hold	After R.E. CTTL	0.0	
t_{CCLKia}		CCLK Inactive	After R.E. CTTL		12.0
t_{CDOh}		CDOUT Hold	After R.E. CTTL	0.0	
t_{CDOv}		CDOUT Valid	After R.E. CTTL	-2.0[3]	12.0
t_{CTp}		CTTL Clock Period[1]	R.E. CTTL to next R.E. CTTL	30.5	250,000
t_{EMCSa}		$\overline{\text{EMCS}}$ Active	After R.E. CTTL, T2W1		12.0
t_{EMCSh}		$\overline{\text{EMCS}}$ Hold	After R.E. CTTL	0.0	
t_{EMCSia}		$\overline{\text{EMCS}}$ Inactive	After R.E. CTTL T3		12.0
t_{FSa}		CFS0 Active	After R.E. CTTL		25.0
t_{FSh}		CFS0 Hold	After R.E. CTTL	0.0	
t_{FSia}		CFS0 Inactive	After R.E. CTTL		25.0
t_{MMCLKa}		Master MICROWIRE Clock Active	After R.E. CTTL		12.0
t_{MMCLKh}		Master MICROWIRE Clock Hold	After R.E. CTTL	0.0	
$t_{MMCLKia}$		Master MICROWIRE Clock Inactive	After R.E. CTTL		12.0
t_{MMDOh}		Master MICROWIRE Data Out Hold	After R.E. CTTL	0.0	
t_{MMDOv}		Master MICROWIRE Data Out Valid	After R.E. CTTL		12.0
t_{MWDOf}		MICROWIRE Data Float[1]	After R.E. $\overline{\text{MWCS}}$		70.0
t_{MWDOh}		MICROWIRE Data Out Hold[2]	After F.E. MWCLK	0.0	
t_{MWDOnf}		MICROWIRE Data No Float[2]	After F.E. $\overline{\text{MWCS}}$	0.0	70.0
t_{MWDOv}		MICROWIRE Data Out Valid[2]	After F.E. MWCLK		70.0
t_{MWITOp}		MWDIN to MWDOUT	Propagation Time		70.0
t_{MWRDYa}		$\overline{\text{MWRDY}}$ Active	After R.E. of CTTL	0.0	35.0

Table 1-11: Output Signals

Symbol	Figure	Description	Reference Conditions	Min (ns)	Max (ns)
$t_{MWRDYia}$		$\overline{\text{MWRDY}}$ Inactive	After F.E. MWCLK	0.0	70.0
t_{PABCh}		PB and $\overline{\text{MWRQST}}$	After R.E. CTTL	0.0	
t_{PABCv}		PB and $\overline{\text{MWRQST}}$	After R.E. CTTL, T2W1		12.0

1. In normal operation mode, t_{CTp} must be 30.5 ns; in power-down mode, t_{CTp} must be 50,000 ns.
2. Guaranteed by design, but not fully tested.
3. Negative hold times are allowed since they are relative to the internal signal CTTL.

Table 1-12: Input Signals

Symbol	Figure	Description	Reference Conditions	Min (ns)	Max (ns)
t_{CDIh}		CDIN Hold	After R.E. CTTL	0.0	
t_{CDIs}		CDIN Setup	Before R.E. CTTL	25.0	
t_{DIh}		Data in Hold (D0:7)	After R.E. CTTL T1, T3 or TI	0.0	
t_{DIs}		Data in Setup (D0:7)	Before R.E. CTTL T1, T3 or TI	19.0	
t_{MMDINh}		Master MICROWIRE Data In Hold	After R.E. CTTL	0.0	
t_{MMDINs}		Master MICROWIRE Data In Setup	Before R.E. CTTL	11.0	
t_{MWCKh}		MICROWIRE Clock High (slave)	At 2.0 V (both edges)	100.0	
t_{MWCKl}		MICROWIRE Clock Low (slave)	At 0.8 V (both edges)	100.0	
t_{MWCKp}		MICROWIRE Clock Period (slave)[1]	R.E. MWCLK to next R.E. MWCLK	2.5 µs	
t_{MWCLKh}		MWCLK Hold	After $\overline{\text{MWCS}}$ becomes inactive	50.0	
t_{MWCLKs}		MWCLK Setup	Before $\overline{\text{MWCS}}$ becomes active	100.0	
t_{MWCSh}		$\overline{\text{MWCS}}$ Hold	After F.E. MWCLK	75.0	
t_{MWCSs}		$\overline{\text{MWCS}}$ Setup	Before R.E. MWCLK	100.0	
t_{MWDIh}		MWDIN Hold	After R.E. MWCLK	50.0	
t_{MWDIs}		MWDIN Setup	Before R.E. MWCLK	100.0	

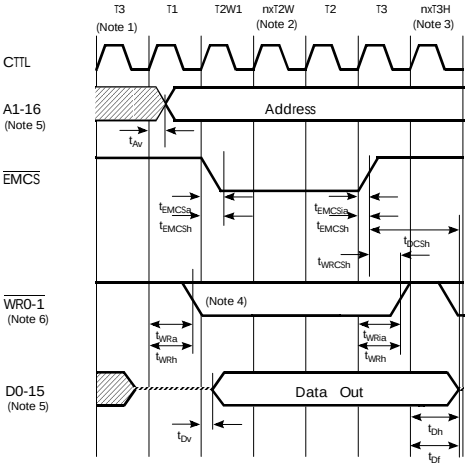
Table 1-12: Input Signals

Symbol	Figure	Description	Reference Conditions	Min (ns)	Max (ns)
t_{PWR}		Power Stable to $\overline{\text{RESET}}$ R.E.[2]	After V_{CC} reaches 4.5 V	30.0 ms	
t_{RSTw}		$\overline{\text{RESET}}$ Pulse Width	At 0.8 V (both edges)	10.0 ms	
t_{Xh}		CLKIN High	At 2.0 V (both edges)	$t_{X1p}/2 - 5$	
t_{Xl}		CLKIN Low	At 0.8 V (both edges)	$t_{X1p}/2 - 5$	
t_{Xp}		CLKIN Clock Period	R.E. CLKIN to next R.E. CLKIN	244.4	

1. Guaranteed by design, but not fully tested in power-down mode.
2. Guaranteed by design, but not fully tested.

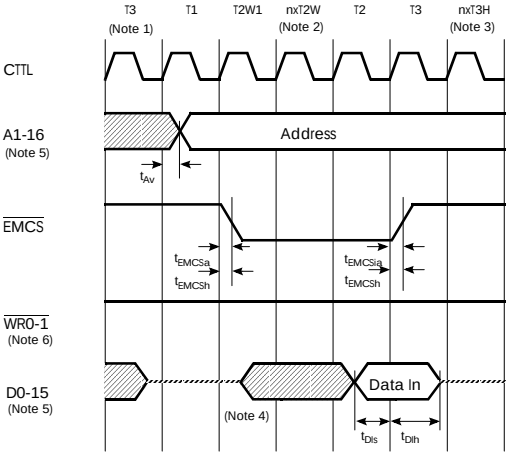
1.3.5 TIMING DIAGRAMS

Figure 1-19: SRAM Write Cycle Timing



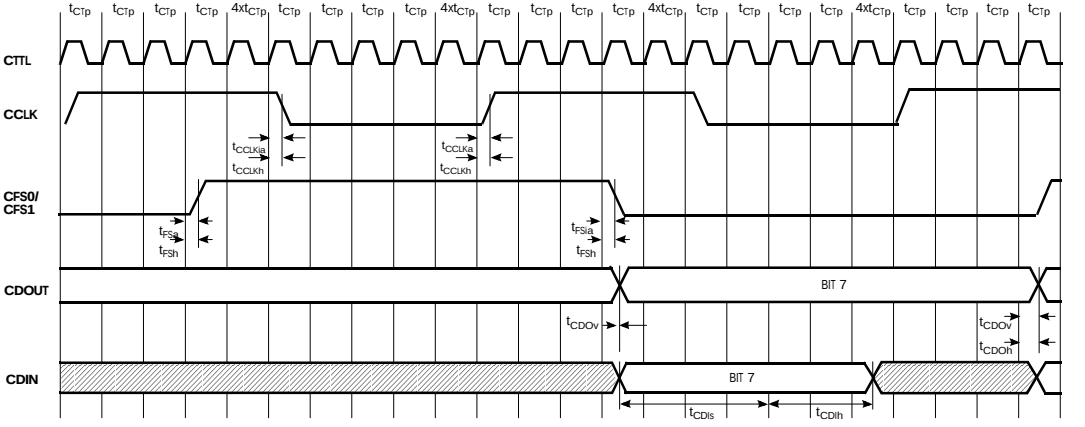
1. *This cycle may be either TI (Idle), T3 or T3H.*
2. *$0 \leq n \leq 7$*
3. *If n = 0, this cycle may be either TI (Idle) or T1 (with a new address) and t_{Df} and t_{Dh} are measured from the R.E. of CTTL in T3.*
4. *If n = 1, this cycle is T3H, the address bus does not change its value and t_{Df} and t_{Dh} are measured from the R.E. of CTTL in T3H.*
5. *Depends on which bytes are written.*
6. *For an 8-bit data bus, address lines are A0-15 and data lines are D0-7*
7. *$\overline{WR1}$ is not available if expansion memory is configured as 8-bit bus RAM.*

Figure 1-20: SRAM Read Cycle



1. *This cycle may be either TI (Idle), T3 or T3H.*
2. *$0 \leq n \leq 7$*
3. *If n = 0, this cycle may be either TI (Idle) or T1 (with a new address). If n = 1, this cycle is T3H, and address bus does not change its value.*
4. *Data can be driven by an external device at T2W1, T2W, T2 and T3.*
5. *For an 8-bit data bus, address lines are A0-15 and data lines are D0-7.*
6. *$\overline{WR0}$ and $\overline{WR1}$ are not available if Expansion Memory is configured as ROM. $\overline{WR1}$ is not available if Expansion Memory is configured as 8-bit bus RAM.*

Figure 1-21: CODEC Short Frame Timing



***Note**: This cycle may be either TI (Idle), T2, T3 or T3H.*

Figure 1-22: CODEC Long Frame Timing

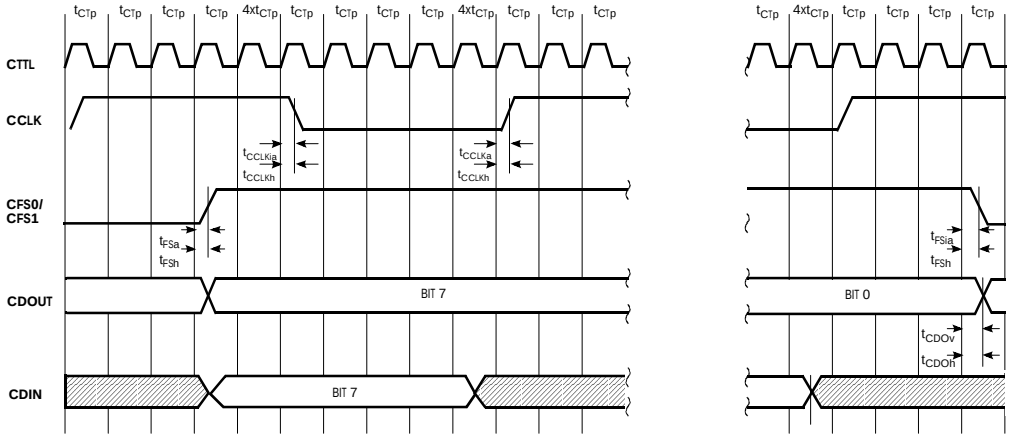


Figure 1-23: Slave CODEC CCLK and CFSO Timing

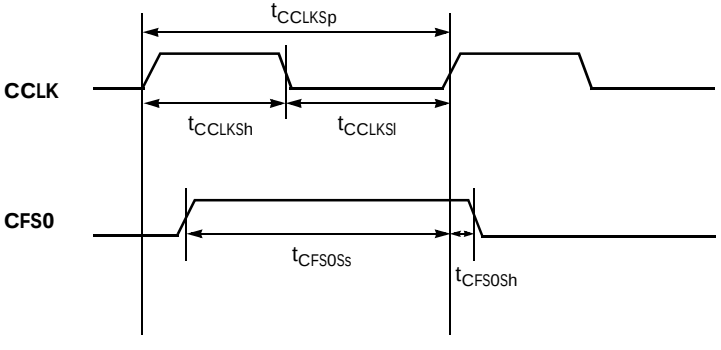


Figure 1-24: MICROWIRE Transaction Timing--Data Transmitted to Output

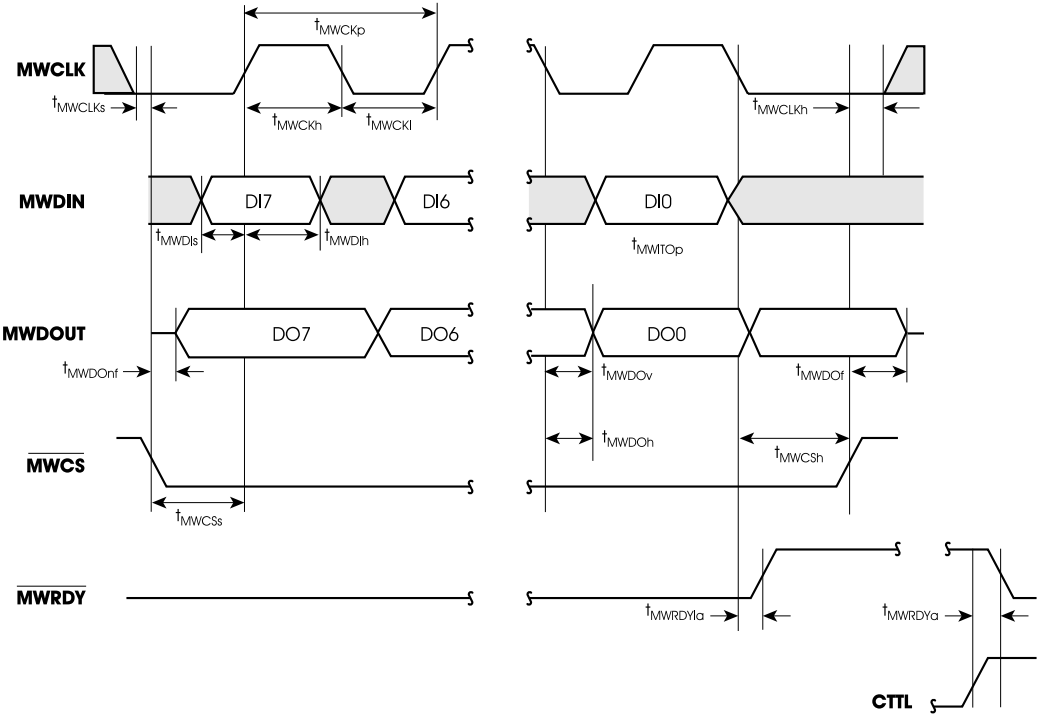


Figure 1-25: MICROWIRE Transaction Timing--Echoed Transmitted to Output

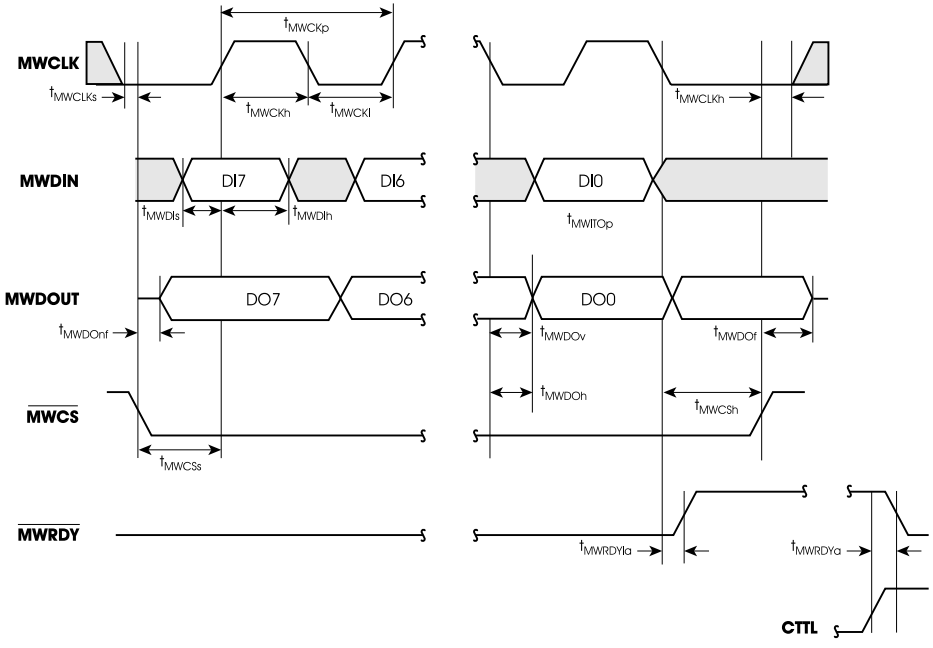


Figure 1-26: Output Signal Timing for $\overline{\text{MWRQST}}$

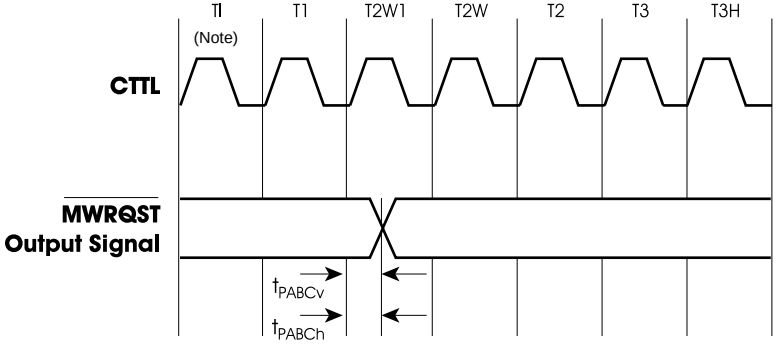


Figure 1-27: CLKIN and CTTL Timing

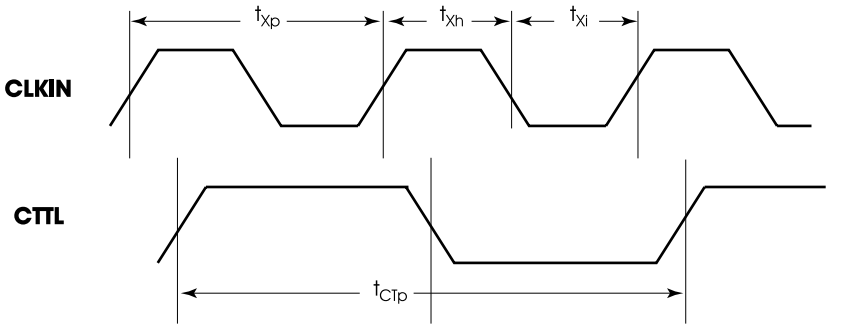


Figure 1-28: Reset Timing When Reset is not at Power-UP

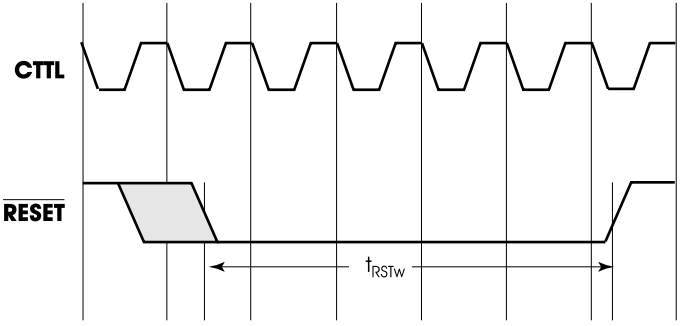


Figure 1-29: Reset Timing when reset is at power-on

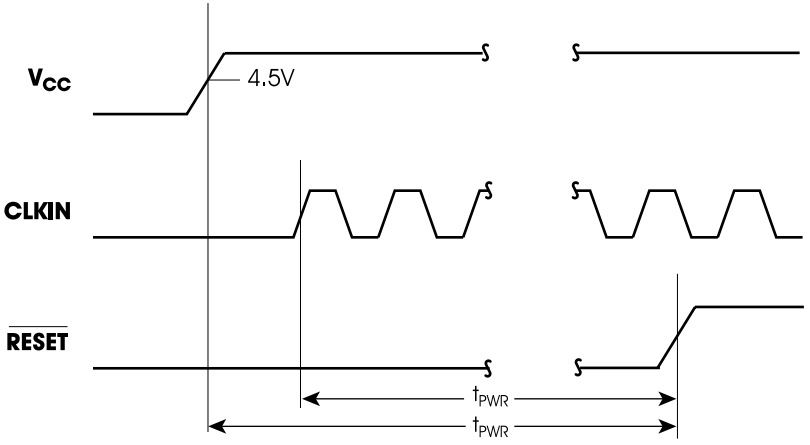
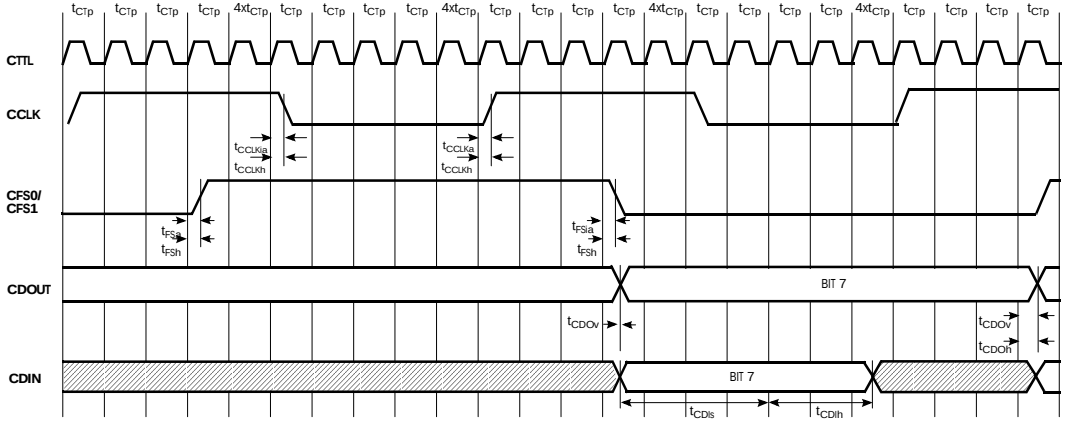
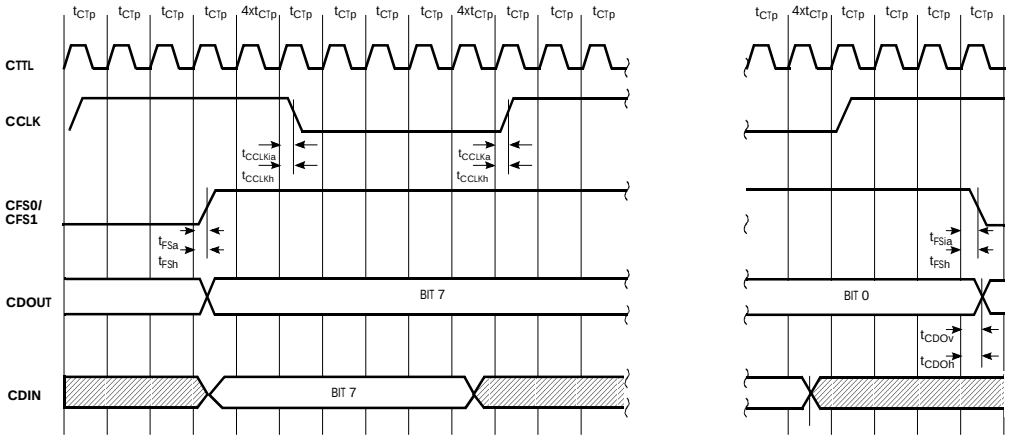


Figure 1-30: Codec Short Frame Timing



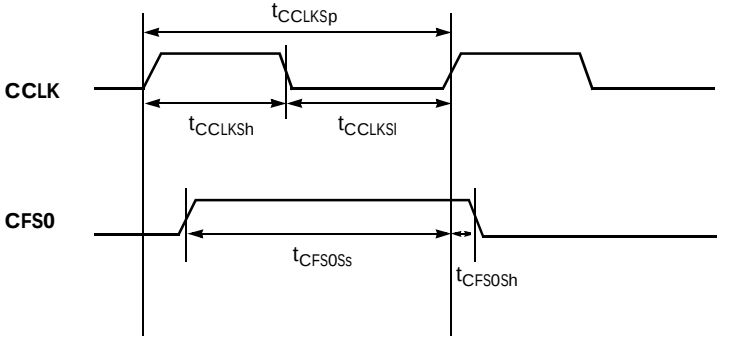
***Note**: The CCLK and CFS0 timing is shown for Master Mode only. For Slave Mode, see Figure 1-32.*

Figure 1-31: Codec Long Frame Timing



***Note**: The CCLK and CFS0 timing is shown for Master Mode only. For Slave Mode, see Figure 1-32.*

Figure 1-32: Slave Codec CCLK and CFS0 Timing



***Note**: For CFS1, CDIN, CDOUT timing, see Figure 1-30 and Figure 1-31.*

Figure 1-33: MICROWIRE Transaction Timing--data transmitted to output

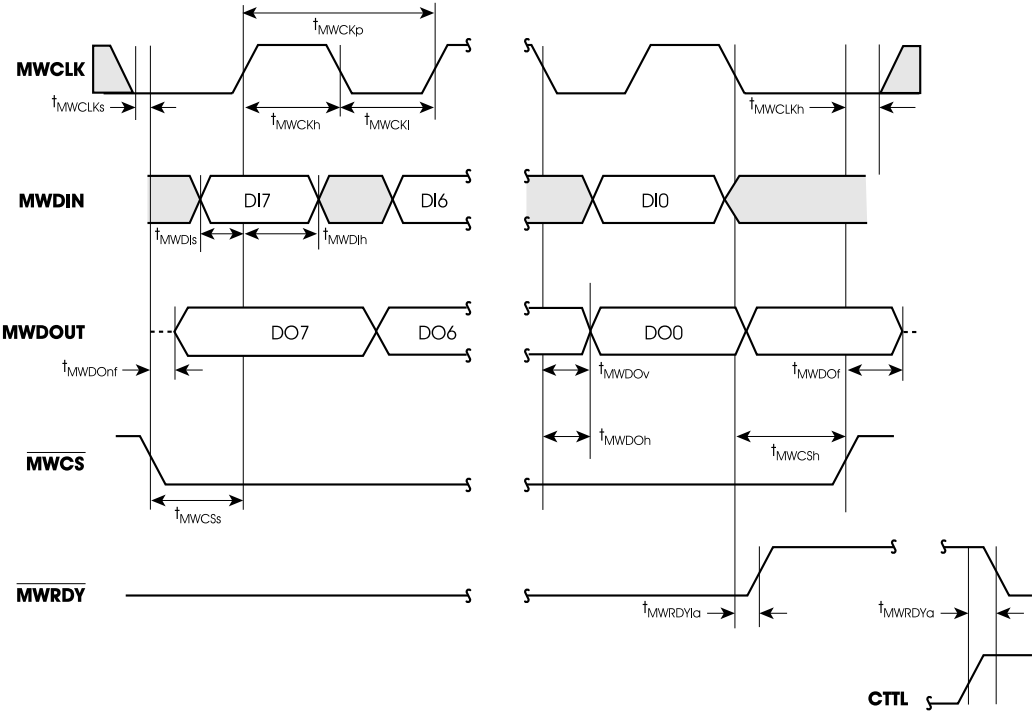
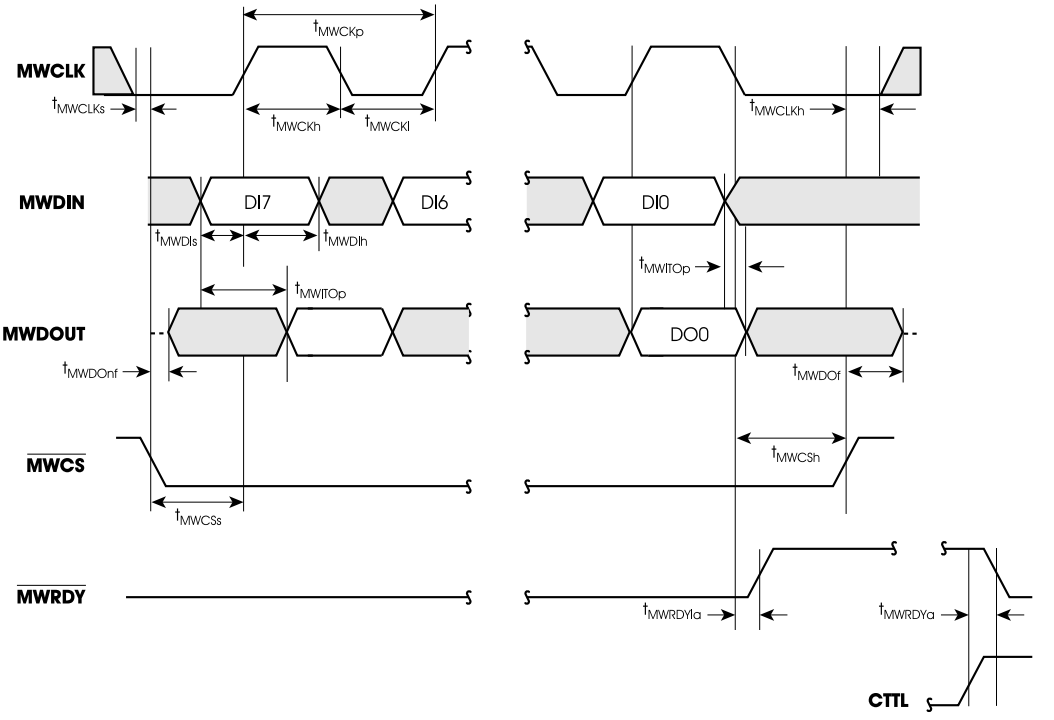


Figure 1-34: MICROWIRE Transaction Timing--data echoed to output



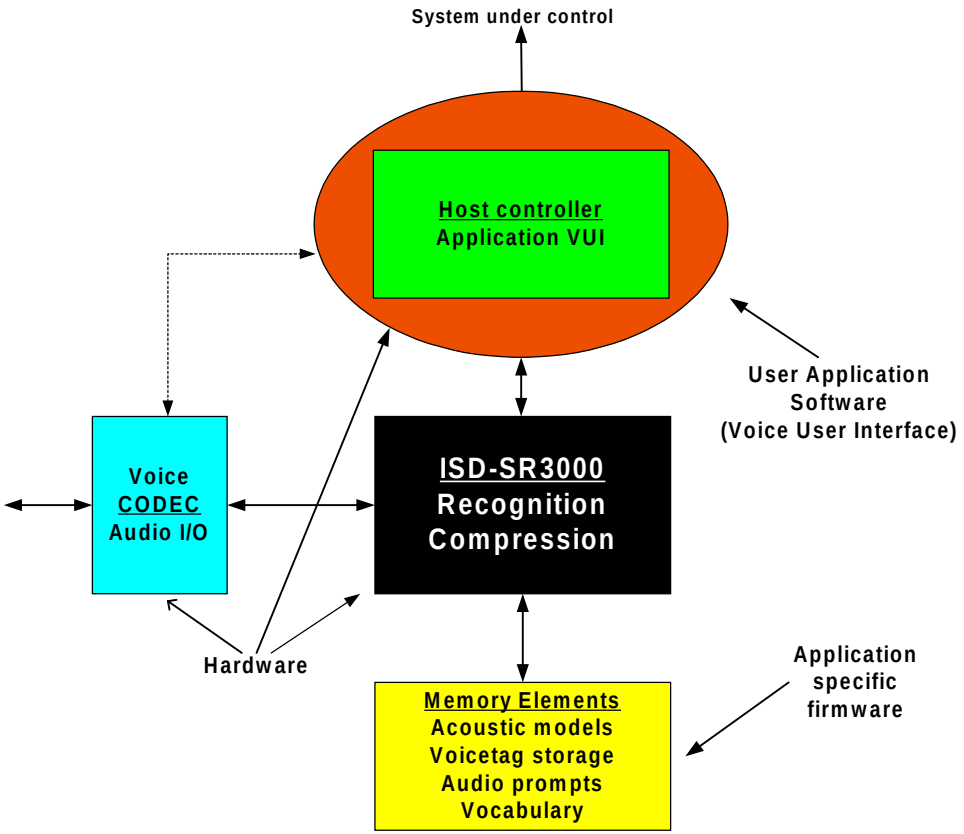
Chapter 2—SOFTWARE

2.1 OVERVIEW

In chapter one, the ISD-SR3000 was described as a hardware component in a system. This system must include a host controller to handle the Voice User Interface layer that controls the ISD-SR3000. The ISD-SR3000 recognizes words from a pre-defined fixed list and returns them to the host controller for handling. It is the host controller's responsibility to decide how to react to a set of recognized words that has been returned by the recognition engine.

The ISD-SR3000 software resides in the on-chip ROM. It includes speech recognition, speech compression, system support functions and a software interface to hardware peripherals. The following sections in this chapter describe, in detail, the ISD-SR3000 interface, operation procedures, software, tools and commands set.

Figure 2-35: Partitioning of Hardware Components and Software Code



2.2 RECOGNITION ENGINE

ISD-SR3000 uses a segmented triphone recognition process. The sampled speech utterance is split into distinct phonetic sounds, the smallest units of speech. Because these phonemes vary in both sound and duration, the processor must be able to determine boundaries between the sounds. The ISD-SR3000 uses Hidden Markov Models to hypothesize boundaries between sounds and to form probabilistic models on each possible combination.

The outputs are then classified by determining matches between the phonetic sounds and the stored phoneme models. The acoustic models for the phonemes are gathered from a large sample of speakers, allowing for a wide variation across accents, dialect, and gender. This allows the recognizer to associate the sound segments with a number of possible phonemes, enabling recognition when words are pronounced differently.

The phonemes are then matched to vocabulary words or phrases using a search routine. The set of phonemes is compared to the vocabulary models for the active topics, and the recognized word is returned. If the phonemes do not match any of the active vocabulary words, nothing is returned. The ISD-SR3000 does not return a score with the word; it either recognizes a word, or it does not.

2.2.1 TYPES OF RECOGNITION

The ISD-SR3000 is capable of both speaker-independent and speaker defined recognition. The recognition engine is continuous, allowing for multiple word commands and connected digits. However, there must be recognized silence before and after valid utterances. The length of the silence is programmed into the host controller, and may be as small as 100ms. The commands and digits are speaker-independent, with models constructed from a large corpus of speakers. The speaker-defined voicetags and commands are partially speaker-dependent. However, they are constructed by creating acoustic models “on-the-fly” from the phoneme base. This means only one training pass is required for entering the voicetags, and recognition is possible with some variation in the way the name is spoken. The first pass is used to create the phoneme model, and a second pass is used for recognition confirmation.

2.2.2 GRAMMAR

A grammar is used to define the structure of the commands. The ISD-SR3000 is designed to work with multiple topics or a finite-state grammar. This type of grammar is designed to limit perplexity (the number of possible branches during recognition) by pre-defining the number of allowable words at a given state. For example, a prompt that requires a “yes” or “no” response has a perplexity of two. Greater perplexities increase the chances for substitution errors. During recognition, a limited number of topics are active. Topics are groups of words that are active at a given time. For example, in a voice dialing application, digit topics are active after the user issues the “dial” command. No other topics are open (except the global topics such as “cancel” or “help”) so that the recognizer is only trying to recognize digits. This type of grammar and active topics inherently increases recognition accuracy.

Figure 2-36: Topic and Grammar Organization

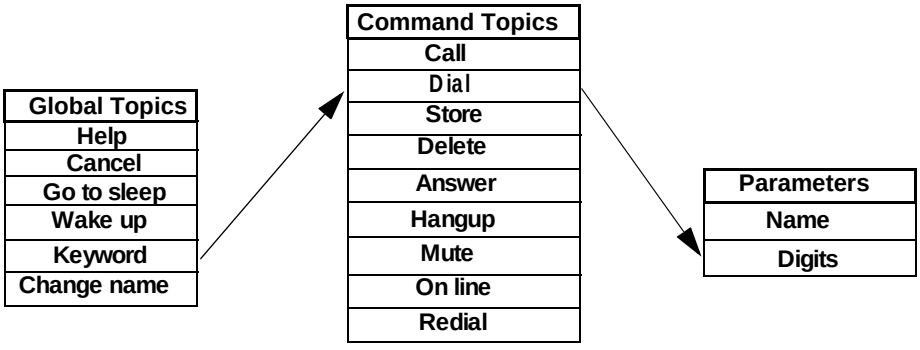


Figure 2-36 shows an example of organizing the commands into menu structure format. From this example, it can be seen how topics are linked, and how only specific topics are active. This is a voice dialing command set furnished as ISD's sample application. Independent VUIs and vocabulary can be developed, but it is necessary to follow the grammar syntax as shown here.

2.2.3 VOCABULARY

A vocabulary defines the following characteristics of the ISD-SR3000:

- Speaker-independent command words and digits for which ISD-SR3000 responds
- Topics under which the commands and digits are organized
- Mapping of tokens to the vocabulary

ISD-SR3000 is designed to work with an application specific vocabulary set. The total vocabulary size is determined by available external memory. When the processor recognizes the commands, tokens (values) are returned to the host controller. These tokens represent spoken words. The host controller maintains a lookup table of the available words and their corresponding token numbers. The host controller can use the tokens to accomplish tasks, such as generating DTMF for dialing a phone number.

ISD supplies recommended vocabulary sets as part of the VUI for specific applications. The vocabulary sets have been carefully selected to ensure high recognition (avoiding words that may be phonetically confused) and effective user utility. The accuracy specifications for ISD-SR3000 are based on the ISD provided commands. It is possible to create custom vocabulary sets for specific applications. Contact ISD for information about vocabulary development tools. The vocabulary can be stored either in external ROM or Flash memory.

2.2.4 LANGUAGE

ISD-SR3000 uses a set of acoustic models designed to recognize a given language. Currently, the supported languages are American English and German. Additional languages require different acoustic models. Contact ISD for availability of additional languages.

2.2.5 ADDITIONAL COMPONENTS

MESSAGE

A message is a compressed recording stored in the ISD-SR3000 memory. This message can be compressed, when using the R (Record Message) command, at 4.7, 6.7 or 8.7kbits/s.

NOISE TOKENS

Each topic can be supplemented with additional tokens that will absorb the 'out of vocabulary' words. These tokens can be added upon user request with ISD's development tools. These tokens are not assigned an index number like the other words in the topic table. Once the engine recognizes a word as a noise token, the ISD-SR3000 returns the noise topic identifier (0xFD) instead of the current topic.

CEPSTRA VALUES

The cepstra coefficients are responsible for normalizing the voice input channel. The ISD-SR3000 contains 15 cepstra values starting from index 0.

VOICE TAG

A voice tag is a recorded message of a user-added word. This message can be compressed, when using the RR (Record for Reco) command, at 4.7, 6.7 or 8.7kbits/s.

2.3 INITIALIZATION

After the system is powered up, it is the host controller's responsibility to initialize the ISD-SR3000. This initialization includes: resetting the chip using the RESET signal, configuring the hardware environment using the CFG command (CODEC and memory types and parameters, etc.), and issuing the INIT and TUNE commands. The TUNE and CFG commands are required only if the system settings are different from the ISD-SR3000's default values.

Refer to the section ISD-SR3000 Initialization on page 2-68 for an example illustrating the system initialization procedure for the ISD-SR3000.

2.4 RECO ENGINE MANAGEMENT

After the system has been initialized (including setting the vocabulary for the prompts) it is the host controller's responsibility to initialize the recognition engine. This includes loading of topics, enabling topics and activating the recognition engine. It is recommended that you load all the topics if the system memory size enables it, rather then loading and unloading a topic each time one needs to be enabled.

Once the recognition engine is initialized, the host controller should wait for the EV_RECO_QUEUE bit in the status register to be set. When the ISD-SR3000 sets this bit, it automatically asserts the MWRQST line as an indication for the host controller to read this register (by issuing the GSW command). When the host controller has identified this bit, it should start retrieving the recognized words (noted as topic number and token number) from the ISD-SR3000 recognition queue buffer using the GNR command. It is the host controller's responsibility to interpret the incoming words into grammar commands using predefined tables. (These tables are created by the ISD development tool). When the host controller interprets a valid command, it can then respond by executing commands such as stopping the recognition engine, playing a prompt or a tone (only after the recognition engine is disabled), changing menus and topics, adding or deleting user voice tags (a new acoustic word), operating an external device, etc. The recognition process is half-duplex. When the ISD-SR3000 is playing back audio, recognition is not active.

Refer to the section Example Operation Procedures on page 2-68 for examples illustrating the normal operation procedure for the ISD-SR3000 engine and the adding of a voice tag.

2.5 HOST CONTROLLER INTERFACE

MICROWIRE/PLUS™ is a synchronous serial communication protocol that minimizes the number of connections, and thus the cost, of communicating with peripherals.

The ISD-SR3000 MICROWIRE interface implements the MICROWIRE/PLUS interface in slave mode, with an additional ready signal. It enables a host controller to interface efficiently with the ISD-SR3000 processor application.

The host controller is the protocol master and provides the clock for the protocol. The ISD-SR3000 processor supports clock rates of up to 400kHz. This transfer rate refers to the bit transfer. The actual throughput is slower due to byte processing by the ISD-SR3000 processor and the host controller.

Communication is handled in bursts of eight bits (one byte). In each burst the ISD-SR3000 processor is able to receive and transmit eight bits of data. After eight bits have been transferred, an internal interrupt is issued for the ISD-SR3000 processor to process the byte, or to prepare another byte for sending. In parallel, the ISD-SR3000 processor sets MWRDY to 1, to signal the host controller that it is busy with the byte processing. Another byte can be transferred only when the MWRDY signal is cleared to 0 by the ISD-SR3000 processor. When the ISD-SR3000 processor transmits data, it expects to receive the value 0xAA before each transmitted byte. The ISD-SR3000 processor reports any status change by clearing the MWRQST signal to 0.

If processor command's parameter is larger than one byte, the host controller transmits the Most Significant Byte (MSB) first. If a return value is larger than one byte, the ISD-SR3000 processor transmits the MSB first.

SIGNAL DESCRIPTION

The following signals are used for the interface protocol. Input and output are relative to the ISD-SR3000.

2.5.1 INPUT SIGNALS

MWDIN

MICROWIRE Data In. Used for input only, for transferring data from the host controller to the ISD-SR3000.

MWCLK

This signal serves as the synchronization clock during communication. One bit of data is transferred on every clock cycle. The input data is available on MWDIN, and is latched on the clock's rising edge. The transmitted data is output on MWDOUT, on the clock's falling edge. The signal should remain low when switching $\overline{\text{MWCS}}$.

$\overline{\text{MWCS}}$

MICROWIRE Chip Select. The $\overline{\text{MWCS}}$ signal is cleared to 0, to indicate that the ISD-SR3000 is being accessed. Setting $\overline{\text{MWCS}}$ to 1 causes the ISD-SR3000 to start driving MWDOUT with bit 7 of the transmitted value. Setting the $\overline{\text{MWCS}}$ signal resets the transfer-bit counter of the protocol. Thus, the $\overline{\text{MWCS}}$ signal is used to synchronize the ISD-SR3000 and the host controller to recognize the beginning of the byte transfer.

To prevent false detection of access to the ISD-SR3000, due to spikes on the MWCLK signal, use this chip select signal to toggle the MWCLK input signal (only when the ISD-SR3000 is open for communication.)

2.5.2 OUTPUT SIGNALS

MWDOUT

MICROWIRE Data Out. Used for output only, for transferring data from the ISD-SR3000 to the host controller. When the ISD-SR3000 receives data, it is echoed back to the host controller on this signal, unless the received data is 0xAA. In this case, the ISD-SR3000 echoes a command's return value.

$\overline{\text{MWRDY}}$

MICROWIRE Ready. When active (0), this signal indicates that the ISD-SR3000 is ready to transfer (receive or transmit) another byte of data.

This signal is set to 1 by the ISD-SR3000 after each byte transfer has been completed. It remains 1 while the ISD-SR3000 is busy reading the byte, writing the next byte or executing the received command (after the last parameter has been received). $\overline{\text{MWRDY}}$ is cleared to 0 after reset. For proper operation after a hardware reset, this signal should be pulled up.

$\overline{\text{MWRQST}}$

MICROWIRE Request. When active (0), this signal indicates that new status information is available. $\overline{\text{MWRQST}}$ is deactivated (set to 1), after the ISD-SR3000 receives a GSW (Get Status Word) command from the host controller. After reset, this signal becomes active (0) to indicate that a reset occurred. $\overline{\text{MWRQST}}$, unlike all the signals of the communication protocol, is an asynchronous line that is controlled by the ISD-SR3000 firmware.

2.5.3 SIGNAL USE IN THE INTERFACE PROTOCOL

After reset, both $\overline{\text{MWRQST}}$ and $\overline{\text{MWRDY}}$ are cleared to 0.

The $\overline{\text{MWRQST}}$ signal is activated to indicate that a reset occurred. The EV_RESET bit in the status register is used to indicate a reset condition.

The GSW command should be issued after reset to verify that the EV_RESET event occurred, and to deactivate the $\overline{\text{MWRQST}}$ signal.

While the $\overline{\text{MWCS}}$ signal is active (0), the ISD-SR3000 reads data from MWDIN on every rising edge of MWCLK. The ISD-SR3000 also writes every bit back to MWDOUT. This bit is either the same bit, which was read from MWDIN (in this case it is written back as a synchronization echo after some propagation delay), or it is a bit of a value the ISD-SR3000 transmits to the host controller (in this case it is written on every falling edge of the clock).

When a command has more than one parameter/return-value, the parameters/return-values are transmitted in the order of appearance. If a parameter/return-value is more than one byte long, the bytes are transmitted from the most significant to the least significant.

The $\overline{\text{MWRDY}}$ signal is used as follows:

1. **Active (0) $\overline{\text{MWRDY}}$ signals the host controller that the last eight bits of data transferred to/from the voice module were accepted and processed (see below).**
2. **The $\overline{\text{MWRDY}}$ signal is deactivated (set to 1 by the ISD-SR3000) after 8-bits of data were transferred to/from the ISD-SR3000. The bit is set following the falling edge of the eighth MWCLK clock-cycle.**
3. **The $\overline{\text{MWRDY}}$ signal is activated (cleared to 0) by the ISD-SR3000 when it is ready to receive the first parameter byte (if there are any parameters) and so on until the last byte of parameters is transferred. An active $\overline{\text{MWRDY}}$ signal after the last byte of parameters indicates that the command was parsed and (if possible) executed. If that command has a return value, the host controller must read the value before issuing a new command.**
4. **When a return value is transmitted, the $\overline{\text{MWRDY}}$ signal is deactivated after every byte, and activated again when the ISD-SR3000 is ready to send another byte or receive a new command.**

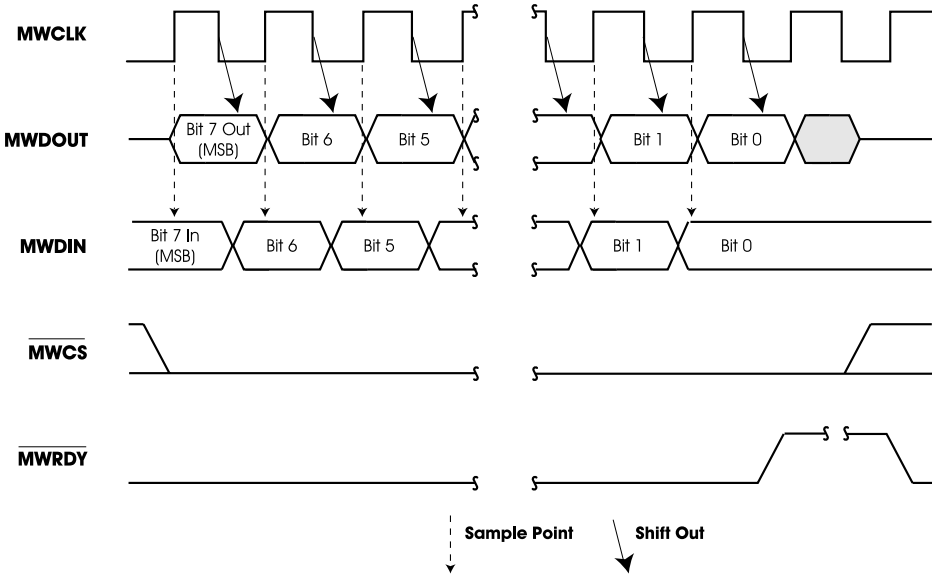
The $\overline{\text{MWRDY}}$ signal is activated (cleared to 0) after reset, and after a protocol time-out.

The $\overline{\text{MWRQST}}$ signal is used as follows:

1. The $\overline{\text{MWRQST}}$ signal is activated (cleared to 0), when the status word is changed.
2. The $\overline{\text{MWRQST}}$ signal remains active (0), until the ISD-SR3000 receives a GSW command.

Figure 2-37 illustrates the sequence of activities during a MICROWIRE data transfer.

Figure 2-37: Sequence of Activities During a MICROWIRE Byte Transfer



2.5.4 INTERFACE PROTOCOL TIME-OUTS

Depending on the ISD-SR3000's state, if more than 100 milliseconds elapse between the assertion of the $\overline{\text{MWRDY}}$ signal of the 8th bit of the next byte pertaining to the same command transaction, a time-out event occurs, and the ISD-SR3000 responds as follows:

1. Sets the error bit in the status word to 1.
2. Sets the EV_TIMEOUT bit in the error word to 1.
3. Activates the $\overline{\text{MWRQST}}$ signal (clears it to 0).
4. Activates the $\overline{\text{MWRDY}}$ signal (clears it to 0).
5. Waits for a new command. (After a time-out occurs, i.e. the host controller received $\overline{\text{MWRQST}}$ during the command transfer, or result reception, the host controller must wait at least four milliseconds before issuing the next command.)

2.5.5 ECHO MECHANISM

The ISD-SR3000 echoes back to the host controller all the bits received by the ISD-SR3000. Upon detection of an error in the echo, the host controller should stop the protocol clock, which eventually causes a time-out error (i.e., ERR_TIMEOUT bit is set in the error word).

Note: *When a command has a return value, the ISD-SR3000 transmits bytes of the return value instead of the echo value.*

The ISD-SR3000 transmits a byte as an echo when it receives the value 0xAA from the host controller. Upon detection of an error, the ISD-SR3000 activates the MWRQST signal and sets the ERR_COMM bit in the error word.

2.6 MEMORY INTERFACE

The ISD-SR3000 memory is externally stored on a ROM or a FLASH device. These devices are addressed through an Expansion Memory mechanism containing four registers that yields to a 20 bit address bus, A(0:19), and 16 bit of data bus, D(0:15). For a detailed description and references, please refer to section 1.2.7.

2.7 CODEC INTERFACE

2.7.1 SUPPORTED FUNCTIONALITY

The ISD-SR3000 processor supports two operational modes: the master mode and the slave mode. In master mode, the ISD-SR3000 provides the clock and the synchronization signals. It supports a list of single channel and dual channel CODECs, as listed in Table 1-4. In slave mode, the control signals are provided by an external source.

The CODEC interface is designed to exchange data in short frame format as well as in long frame format. The channel width may be either 8 bits (u-Law format or A-Law format), or 16 bits (linear format). In slave mode the clock may be divided by two, if required (two bit rate clock mode). The ISD-SR3000 processor supports up to 2 voice channels.

See "THE CODEC INTERFACE" on page 1-7 for a detailed description of the supported CODEC devices and the hardware connectivity.

Use the CFG command to define the CODEC mode (master or slave), the data frame format (short or long), the channel width (8 bits or 16 bits), the clock bit rate (single or dual) and the number and type of CODEC (one or two, single channel or dual channel). **See "SR3000 processor commands--quick reference table" on page 2-16.**

2.8 SPEECH OUTPUT (AUDIO PROMPTS)

Speech output is the technology that is used to create messages or prompts out of predefined words and phrases stored in a vocabulary.

There are two kinds of predefined messages: fixed messages (e.g., help information, indication prompts etc.) and programmable messages (e.g., time-and-day stamp, or the "You have n entries in your phone book" announcement).

A vocabulary includes a set of predefined words and phrases required to construct messages in any language. Applications can support more than one language by using a separate vocabulary for each language.

2.8.1 INTERNATIONAL VOCABULARY SUPPORT (IVS)

IVS is a mechanism by which the ISD-SR3000 processor utilizes several vocabularies stored on an external storage device. IVS enables the ISD-SR3000 to synthesize messages with the same meaning, but in different languages, from separate vocabularies.

Among IVS features:

- Multiple vocabularies stored on a single storage device.
- Plug-and-play. The same host controller code is used for all languages.
- Synthesized and recorded messages use the same voice compression algorithm to achieve equal quality.
- Argumented sentences. (For example: *You have <n> messages*.)
- Auto-synthesized time-and-day stamp (driven by the ISD-SR3000 processor's clock).
- Support for various language and sentence structures:
  - One versus many. (For example: You *have one message* versus *You have two messages*.)
  - None versus many. (For example: *You have no messages* versus *You have two messages*.)
  - Number synthesis (English—*Eighty* versus French—*Quatre-vingt*).
  - Word order (English—*Twenty-one* versus German—*Einundzwanzig*).
- Days of the week (Monday through Sunday versus Sunday through Saturday).

2.8.2 VOCABULARY DESIGN

There are several issues, sometimes conflicting, which must be addressed when designing a vocabulary.

VOCABULARY CONTENT

If memory space is not an issue, the vocabulary could contain all the required sentences, each recorded separately.

If memory space is a concern, the vocabulary must be compact; it should contain the minimum set of words and phrases required to synthesize all the sentences. The least memory is used when phrases and words that are common to more than one sentence are recorded only once, and the IVS tool is used to synthesize sentences out of them.

A good combination of sentence quality and memory space is achieved if you take the "compact" approach, and extend it to solve pronunciation problems. For example, the word twenty is pronounced differently when used in the sentences *You have twenty names* and *You have twenty-two names*. To solve this problem, words that are pronounced differently should be recorded more than once, each in the correct pronunciation.

VOCABULARY RECORDING

When recording vocabulary words, there is a compromise between space and quality. The words should be recorded and saved in a compressed form, and you should use the best voice compression for that purpose. However, lower compression rates do affect the voice quality.

Another issue to consider is the difference in voice quality between synthesized and recorded audio (e.g. between stored audio prompts and recorded names). It is more pleasant for the human ear to hear both messages with the same sound quality.

VOCABULARY ACCESS

Sometimes compactness and high quality are not enough. There should be a simple and flexible interface to access the vocabulary elements. Not just the vocabulary, but the code to access the vocabulary should be compact.

When designing for a multi-lingual environment, there are even more issues to consider. Each vocabulary should be able to handle language-specific structures and designed in a cooperative way with the other vocabularies so that the code to access each vocabulary is the same. When you use the command to synthesize the sentence *Monday. 12:30 P.M.,* you should not care in what language the message is played back.

2.8.3 IVS VOCABULARY COMPONENTS

This section describes the basic concept of an IVS vocabulary, its components, and the relationships between them.

BASIC CONCEPTS

An IVS vocabulary consists of words, sentences, and special codes that control the behavior of the algorithm which ISD-SR3000 processor uses to synthesize sentences.

WORD TABLE

The words are the basic units in the vocabulary. Create synthesized sentences by combining words in the vocabulary. Each word in the vocabulary is given an index which identifies it in the word table.

Note that, depending on the language structures and sentences synthesized, you may need to record some words more than once in the vocabulary. For example, if you synthesize the sen-

tences: *you have twenty names* **and** *you have twenty-five names*, **the word** *twenty* **is pronounced differently. In this example,** *twenty* **should be defined as two different words.**

NUMBER TABLES

The number tables allow you to treat numbers differently depending on the context.

Example 1: The number 1 can be announced as *one* **as in** *name number one* **or as first as in** *first name*.

Example 2: The number 0 can be announced as *no* **as in** *you have no names* **or as** *oh* **as in** *monday, eight oh five A.M*.

A separate number table is required for each particular type of use. The number table contains the indices of the words in the vocabulary that are used to synthesize the number. Up to nine number tables can be included in a vocabulary.

SENTENCE TABLE

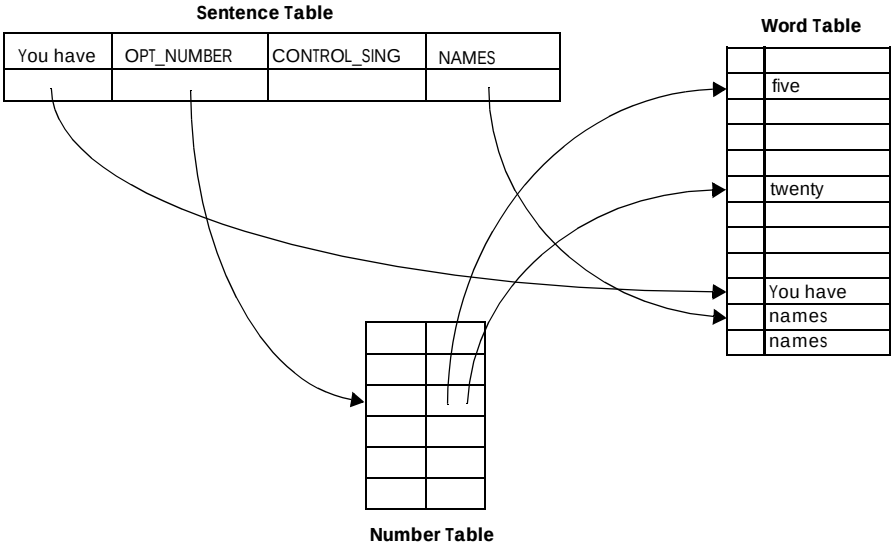
The sentence table describes the predefined sentences in the vocabulary. The purpose of this table is to make the host controller that drives the ISD-SR3000 processor independent of the language being synthesized. For example, if the Flash and/or ROM memory contains vocabularies in various languages, and the first sentence in each vocabulary means you have n messages, the host controller switches languages by issuing the following command to ISD-SR3000 processor:

SPT <storage_media>, <vocabulary_id> -Select a new vocabulary

The host controller software is thus independent of the grammar of the language in use. The sentences consist of words, which are represented by their indices in the vocabulary.

The following figure shows the interrelationship between the three types of tables.

Figure 2-38: The Interrelationship between the Word, the Number, and the Sentence Tables



CONTROL AND OPTION CODES

The list of word indices alone cannot provide the entire range of sentences that the ISD-SR3000 processor is able to synthesize. IVS control and option codes send special instructions to control the speech synthesis algorithm's behavior in the processor.

For example, if the sentence should announce the time of day, the ISD-SR3000 processor should be able to substitute the current day and time in the sentence. These control words do not represent recorded words, rather they instruct the processor to take special actions.

2.9 THE STATE MACHINE

The ISD-SR3000 processor functions as a state machine. It changes state either in response to a command sent by the host controller, after execution of a command is completed, or as a result of an internal event (e.g. memory full or power failure). The ISD-SR3000 processor states are listed below.

RESET

The ISD-SR3000 processor is initialized to this state after a full hardware reset by the RESET signal.

IDLE
This is the state from which most commands are executed. As soon as a command and all its parameters are received, the ISD-SR3000 processor starts executing the command.

PLAY
In this state, a prompt is played.

SYNTHESIS
In this state, an individual word or sentence is synthesized from a vocabulary.

RECORD
In this state, a user's speech is recorded and stored.

RECO
In this state, speech recognition is active.

TONE_GENERATE
In this mode, the ISD-SR3000 synthesizes Tones or Dual Tones to the output.

2.9.1 COMMAND EXECUTION

An ISD-SR3000 command is represented by an 8-bit opcode. Some commands have parameters, and some commands return values to the host controller. Commands are either synchronous or asynchronous.

2.9.2 SYNCHRONOUS COMMANDS

A synchronous command must complete execution before the host controller can send a new command

A synchronous command sequence starts when the host controller sends an 8-bit opcode to the ISD-SR3000 processor, followed by the command's parameters (if any).

The ISD-SR3000 processor executes the command and, if required, transmits a return value to the host controller. Upon completion, the ISD-SR3000 processor notifies the host controller that it is ready to accept a new command by asserting the MWRDY signal.

2.9.3 ASYNCHRONOUS COMMANDS

An asynchronous command runs in the background. During execution of an asynchronous command, other commands can be executed according to the source state in the command table.

2.9.4 ISD-SR3000 STATUS AND REGISTERS

STATUS WORD

The 16-bit Status Word indicates events that occur during normal operation. The ISD-SR3000 processor asserts the MWRQST signal to indicate a change in the Status Word. This signal remains asserted until the ISD-SR3000 processor receives a GSW command. The status word is cleared during reset, and upon successful execution of the GSW command.

ERROR WORD

The 16-bit Error Word indicates errors that occurred during execution of a command. If an error is detected, the command is not processed, the EV_ERROR bit in the Status Word is set to 1, and the MWRQST signal is asserted. The error bits will remain active in the error register until the register is read (using the GEW command).

RECO ERROR WORD

This 16-bit Reco Error Word indicates errors that have occurred in the reco engine. If an error is detected, the ERR_RECO bit in the Status Word is set to 1, and the MWRQST signal is asserted. The Reco Error Word will remain in the Reco Error register until the register is read (using the GRE command).

ERROR HANDLING

When the host controller detects that the MWRQST signal has been asserted, the host controller should issue the GSW (Get Status Word) command, which de-asserts the MWRQST signal. Then the host controller should test the EV_ERROR and the ERR_RECO bits in the data (the Status Word contents) returned by the GSW command. If the EV_ERROR bit is set, the host controller should issue the GEW (Get Error Word) command to read the Error Word for details of the error. If the ERR_RECO is set, the host controller should issue the GRE (Get Reco Error) command to read the Reco Error Word for details of the error.

2.10 SR3000 PROCESSOR COMMANDS--QUICK REFERENCE TABLE

Table 2-13: Reco Commands

Command		Description	Opcode Hex	Source State	Result State	Command Parameters		Return Value	
Name	S/A					Description	Bytes	Description	Bytes
AAW	S	Add Acoustic Word	49	Idle	No change	Topic_ID List_ID	1+1	Token Id	1
CEPG	S	Get Cepstra Values	4B	Idle, Reco	No change	Cepstra Index	2	Cepstra Value	4
CEPS	S	Set (Restore) Cepstra Values	4C	Idle, Reco	No change	Cepst_Index Cepst_Value	2 + 4	None	-
DAW	S	Delete Acoustic Word in a list	4A	Idle	No change	Topic_ID List_ID Token ID	1+1+1	None	-
DAWL	S	Delete All Acoustic Words in a List	4F	Idle	No change	Topic_ID List_ID	1+1	None	-
GRE	S	Get RECO Error code	56	All States		None	-	Error word	2
GNR	S	Get Next Recognition	41	Idle, Reco	No change	None	-	Topic ID Token ID	1+1
GNWL	S	Get Number of Added Acoustic Words in a list	4E	Idle, Reco	No change	Topic_ID List_ID	1+1	Numer Of Words	2
GPC	S	Gets the phoneme count	54	Idle, Reco	No change	None	-	Num of Phonemes	2
GRV	S	Get RECO Program Version	52	Reset, Idle	No change	None	-	Reco Version	2
GVQ	S	Get Num of arguments in the VQ buffer	51	Reco, Idle	No change	None	-	Number of VQ	2
RE	A	Recognition Enable	40	Idle	RECO	None	-	None	-
ROL	S	Recognize Offline	50	Idle	No change	None	-	None	-
RR	A	Record Message for Reco	57	Idle	RECORD	Compression Rate	1	None	-
SRI	S	Stop Reco Immediately	5A	Reco	Idle	None	-	None	-

Table 2-13: Reco Commands

Command		Description	Opcode Hex	Source State	Result State	Command Parameters		Return Value	
Name	S/A					Description	Bytes	Description	Bytes
TOPD	S	Topic Disable	44	Idle, Reco	No change	Topic_ID	1	None	-
TOPE	S	Topic Enable	43	Idle, Reco	No change	Topic_ID	1	None	-
TOPL	S	Topic Load	5B	Idle	No change	Topic_ID	1	None	-
TOPS	S	Topic Save	4D	Idle	No change	Topic_ID	1	None	-
TOPU	S	Topic Unload	5C	Idle	No change	Topic_ID	1	None	-

Note: *For the column labeled S/A, S = Synchronous command and A = Asynchronous command.*

Table 2-14: IVS and Message (Voice Tags) Commands

Command		Description	Opcode Hex	Source State	Result State	Command Parameters		Return Value	
Name	S/A					Description	Bytes	Description	Bytes
CP	S	Check Prompts	2B	Idle	No change	None	-	Test Results	1
DM	S	Delete Message	0A	Idle	No change	None	-	None	
DMS	S	Delete Messages	0B	Idle	No change	Tag_Ref, Tag_Mask	2+2	None	-
GML	S	Get Message Length	19	Idle	No change	None	-	Message Length	2
GMS	S	Get Memory Status	12	Idle	No change	Type	1	Remaining memory blocks	2
GMT	S	Get Message Tag	04	Idle	No change	None	-	Message tag	2
GNM	S	Get Number of Messages	11	Idle	No change	Tag_Ref, Tag_Mask	2+2	Num of messages	2
GTM	S	Get Tagged Message	09	Idle	No change	Tag_Ref, Tag_Mask, Dir	2+2+1	Massage found	1
INFG	S	Tag data information get	3E	Idle	No change	Offset Num of bytes	1+1	byte1...byte n	Num of bytes
INFS	S	Tag data information set	3D	Idle	No change	Offset, Num of bytes byte1...byten	1+1+N	None	-

Table 2-14: IVS and Message (Voice Tags) Commands

Command		Description	Opcode Hex	Source State	Result State	Command Parameters		Return Value	
Name	S/A					Description	Bytes	Description	Bytes
PM	A	Play Message	03	Idle	PLAY	None	-	None	-
R	A	Record Message	0C	Idle	RECORD	Tag (message Tag), Compression_ rate	2 + 1	None	-
S	S	Stop	0	All States except reset	IDLE	None	-	None	-
SAS	A	Say Argumented Sentence	1E	Idle	SYNTHESIS	Sentence_n, Mask ref	1 + 2	None	-
SMT	S	Set Message Tag	05	Idle	No change	Tag_Ref	2	None	-
SO	A	Say One Word	07	Idle	SYNTHESIS	Word_number	1	None	-
SPT	S	Set Prompts Type	20	Idle	No change	Type, Id	1 + 1	None	-
SS	A	Say Sentence	1F	Idle	SYNTHESIS	Sentence_n	1	None	-
SW	A	Say Words	21	Idle	SYNTHESIS	N, word1...wordn	1 + N	None	-

Note: *For the column labeled S/A, S = Synchronous command and A = Asynchronous command.*

Table 2-15: General Commands

Command		Description	Opcode Hex	Source State	Result State	Command Parameters		Return Value	
Name	S/A					Description	Bytes	Description	Bytes
CCIO	S	Configure Codec I/O	34	Idle	No change	Config_value	1	None	-
CFG	S	Configure ISD-SR3000	01	Reset	No change	Config_value	3	None	-
GEW	S	Get Error Word	1B	All States	No change	None	-	Error word	2
GHV	S	Get Hardware Version	02	Reset, Idle	No change	None	-	Version	1
GSV	S	Get System Version	53	Reset, Idle	No change	None	-	Version	2
GSW	S	Get Status Word	14	All States	No change	None	-	Status Word	2
GT	A	Generate Tone	0D	Idle	TONE_ GENERA TE	Tone (single Tone or DTMF)	1	None	-

Table 2-15: General Commands

Command		Description	Opcode Hex	Source State	Result State	Command Parameters		Return Value	
Name	S/A					Description	Bytes	Description	Bytes
GTD	S	Get Time and Day	0E	Idle	No change	Time_day_ option	1	Time and day	2
GTUNE	S	Get Tunable Parameter	06	Idle, Reset	No change	Index	1	Parameter _ value	2
INIT	S	Initialize System	13	Reset	Idle	None	-	None	-
PDM	S	Go To Power-Down Mode	1A	Idle	No change	None	-	None	-
SETD	S	Set Time and Day	0F	Idle	No change	Time_and_da y	2	None	-
TUNE	S	Tune Parameters	15	Reset, Idle	No change	Index, Parameter_ value	1 + 2	None	-
VC	S	Volume Control	28	Idle, Play, Synthesis, Tone Generate, Reco, Record	No change	vol_level (increment/ decrement)	1	None	-

Note: *For the column labeled S/A, S = Synchronous command and A = Asynchronous command.*

2.11 COMMAND DESCRIPTION

The commands are listed in alphabetical order. All command opcodes are one byte in length. All opcodes, parameters and examples are shown using hex values for 8 bit and larger quantities, and binary values for bit values, unless otherwise noted.

Each command description includes an example application of the command. The examples display the opcode issued by the host controller and the response returned by the ISD-SR3000 processor. For commands that require a return value from the ISD-SR3000 processor, the start of the return value is indicated by a thick vertical line. When a return value is required, the host controller must pass value AA (hex) to the ISD-SR3000 engine as a place-holder for each byte to be returned.

Page Location for Command Descriptions

AAW Add Acoustic Word

Opcode: 0x49

Syntax: AAW topic_id list_id [Token_ID]

Type: Synchronous

Description: Adds a new dynamic word (user recorded word) to list list_id in topic topic_id

Parameters: The topic_id, list_id and token_id parameters are one byte each.

Example: Add a new acoustic word to list 0 in topic 3. The new word was added and the token id is 05.

Source	Byte Sequence			
Code	AAW 0300			
Host Controller	49	03	00	AA
ISD-SR3000	49	03	00	05

CCIO Configure Codec I/O

Opcode: 0x34

Syntax: CCIO config_value

Type: Synchronous

Description: Configures the voice sample paths in various states. It should be used to change the default ISD-SR3000 processor configuration.

Parameters: The config_value parameter is one byte and is encoded as follows.

BIT	DESCRIPTION
0	Loopback control 0: Loopback disabled (default) 1: Loopback enabled. In the RECORD state, the input samples are echoed back unchanged (i.e., no volume control) to the codec. This is useful for debugging the analog and codec circuitry.
1	Input control 0: Input from line codec. 1: Input from microphone (default).
3 - 2	Output Control Bits 3 2 Settings 0 0 Automatic configuration. When in RECO/RECORD mode output disabled to the 2 codecs. In IDLE mode, outputs to both (default). 0 1 Output to codec 0 only 1 0 Output to codec 1 only 1 1 Output to both codecs.
4 - 7	Reserved. Must be set to 0.

Example: Configure the codec to have loop back enabled.

Source	Byte Sequence	
Code	CCIO 01	
Host Controller	34	01
ISD-SR3000	34	01

CEPG **Get Cepstra Values**

Opcode: **0x4B**

Syntax: **CEPG cepstra_index [cepstra_value]**

Type: **Synchronous**

Description: **Normalizes the speech environment as it comes in from the microphone. This parameter is automatically updated by the ISD-SR3000. The host controller should save the values (via the CEPS command) in the event the settings are required again.**

Parameters: **The cepstra_index is 2 bytes and the returned parameter cepstra_value is 4 bytes (MSB first).**

Example: Get the cepstra values (0300A184) of Cepstra index 2.

Source	Byte Sequence						
Code	CEPG 0002 AAAA AAAA						
Host Controller	4b	00	02	AA	AA	AA	AA
ISD-SR3000	4b	00	02	03	00	A1	84

CEPS **Set (Restore) Cepstra Values**

Opcode: **0x4C**

Syntax: **CEPS cepstra_index cepstra_value**

Type: **Synchronous**

Description: **Restore the value of cepstra cepstra_index by the value cepstra_value.**

Parameters: **The cepstra_index is a two byte parameter and the cepstra_value is a four byte parameter (MSB first).**

Example: Set the cepstra values for Cepstra index 4.

Source	Byte Sequence						
Code	CEPS 0004 03002433						
Host Controller	4c	00	04	03	00	24	33
ISD-SR3000	4c	00	04	03	00	24	33

CFG **Configure ISD-SR3000**

Opcode: **0x01**

Syntax: **CFG config_value**

Type: **Synchronous**

Description: **Configures the ISD-SR3000 processor for various hardware environments. It should be used to change the default ISD-SR3000 processor configuration.**

Parameters: **The config_value parameter is a 24-bit word and is encoded as follows.**

BIT	DESCRIPTION (the sign "X" refers to a don't care)
3 - 0	Memory type Bits 3 2 1 0 Settings 0 0 0 0 No Memory 1 0 0 0 AMD's AM29F800/AM29F160 Flash (default) All other Reserved
15 - 4	Reserved.
16	Clock bit rate 0: One bit rate clock (default) 1: Two bit rate clock.
17	CODEC configuration 0: Short-frame format (default) 1: Long-frame format (guaranteed by design, but not tested).
19 - 18	CODEC type Bits Bits 19 18 Settings 0 0 16-bit linear 0 1 µ-Law (default) 1 0 A-Law 1 1 Reserved

20	CODEC Interface Mode 0: Master CODEC interface (default) 1: Slave CODEC interface.
22 - 21	CODEC type Bits 22 21 Settings 0 0 One single CODEC 0 1 Two single CODECs (default) 1 0 One dual CODEC 1 1 Reserved
23	Reserved. Must be set to 0.

Example: Configure the ISD-SR3000 processor to work with:

- **One single linear CODEC that supports short frame format**
- **One AMD AM29F800**
- **Master Mode**

Source	Byte Sequence			
Code	CFG 01 00 00 08			
Host Controller	01	00	00	08
ISD-SR3000	01	00	00	08

CP	**Check Prompts**
Opcode:	**0x2B**
Syntax:	**CP [return_value]**
Type:	**Synchronous**
Description:	**Checks (checksum) if the IVS data of the currently selected vocabulary was correctly programmed to the ROM or Flash device. If the vocabulary data is correct, the return value is 1. Otherwise the return value is 0. If the current vocabulary is undefined, ERR_INVALID is reported.**
Parameters:	**return_value is one byte long**

Example: Check the prompt to be in order (returned value is 1).

Source	Byte Sequence	
Code	CP	
Host Controller	2B	AA
ISD-SR3000	2B	01

DAW Delete Acoustic Word in list

Opcode: 0x4A

Syntax: DAW topic_id list_id token_id

Type: Synchronous

Description: Deletes the identifying token token_id of a specific word in list list_id of topic topic_id. This command doesn't erase the recorded message associated with the token token_id (see the DM command for details on deleting a message).

Parameters: topic_id, list_id and token_id are one byte long.

Example: Delete acoustic word 9 in list 1 of topic 3.

Source	Byte Sequence			
Code	DAW 03 01 09			
Host Controller	4a	03	01	09
ISD-SR3000	4a	03	01	09

DAWL Delete All Acoustic Words in List

Opcode: 0x4F

Syntax: DAWL topic_id list_id

Type: Synchronous

Description: This command erases all the tokens from topic topic_id that are associated with list list_id (each parameter is one byte long).

Parameters:

Example: Delete all acoustic words in list 1 of topic 5.

Source	Byte Sequence		
Code	DAWL 05 01		
Host Controller	4f	05	01
ISD-SR3000	4f	05	01

DM **Delete Message (voicetag)**

Opcode: **0x0A**

Syntax: **DM**

Type: **Synchronous**

Description: **(Messages are synonymous with voicetags).**
Deletes the current message. Deleting a message clears its message tag. Deleting the current message does not cause a different message to become current. The current message remains undefined until you issue a new command; for example, issue the GTM command to skip to the next message. The memory space released by the deleted message is immediately available for recording new messages.

Parameters:

Example: Delete the current message.

Source	Byte Sequence	
Code	DM	
Host Controller	0A	
ISD-SR3000	0A	

DMS **Delete Messages (voicetags)**

Opcode: **0x0B**

Syntax: **DMS Tag_Ref Tag_Mask**

Type: **Synchronous**

Description: **Deletes all messages whose message tags match the tag_ref parameter. Only bits set in tag_mask are compared, i.e. a match is considered successful if:**

message tag **and** *tag_mask* **=** *tag_ref* **and** *tag_mask*
where and is a bitwise AND operation

After the command completes execution, the current message is undefined. Use the GTM command to select a message to be the current message.

The memory space released by the deleted message is immediately available for recording new messages.

Parameters: **Tag_Ref Tag_Mask are each two bytes long**

Example: Deleting all the messages associated with topic 8:

Source	Byte Sequence				
Code	DMS 0800 FF00				
Host Controller	0B	08	00	FF	00
ISD-SR3000	0B	08	00	FF	00

GEW Returns the 16-bit error word

Opcode: 0x1B

Syntax: GEW [error_word]

Type: Synchronous

Description: Indicates errors that occurred during execution of the last command. If an error is detected, the command is not processed; the EV_ERROR bit in the status word is set to 1, and the MWRQST signal is activated (driven low).

The GEW command reads the error word. All bits in the error word are cleared to zero during reset and after execution of each GEW command.

If errors ERR_COMMAND or ERR_PARAM occur during the execution of a command that has a return value, the return value is undefined. The host controller must still read the return value to ensure proper synchronization.

Parameters: The 16-bits of the error word are as follows:

BIT	DESCRIPTION
0	ERR_ROL 0: No error 1: Recording too short
1	ERR_OPCODE 0: No opcode errors 1: Illegal opcode. The ISD-SR3000 processor does not recognize the opcode.
2	ERR_COMMAND 0: No command errors 1: Illegal command sequence. The command is not legal in the current state.
3	ERR_PARAM 0: No parameter errors 1: Illegal parameter. The value of the parameter is out of range or is not appropriate for the command.
4	0 or 1: Bit 4 is reserved and should be disregarded
5	ERR_COMM 0: No communications error 1: host controller MICROWIRE communication error
6	ERR_TIMEOUT 0: No timeout error 1: Time out error. Depending on the ISD-SR3000's processor's state, more than 100 milliseconds elapsed between the arrival of two consecutive bytes (for commands that have parameters).

7	ERR_INVALID 0: No context error 1: Command can not be performed in current context.
8-9	0 or 1: Bits 8 and 9 are reserved and should be disregarded.
10	ERR_VQ_OVERFLOW: VQ buffer is full (frames will be missed).
11	ERR_FLASH_MAP: error in directory page of Flash.
12	ERR_Q_FULL: queue of recognized words is full (clear using GNR command).
13-15	Bits 13 to 15 are reserved and should be disregarded.

Example: ISD-SR3000's error word is repeated and it returns the code for a communications error.

Source	Byte Sequence		
Code	GEW		
Host Controller	1B	AA	AA
ISD-SR3000	1B	00	20

GHV Get Hardware Version; returns a one byte value indicating the ISD-SR3000 hardware version

Opcode: 02

Syntax: GHV [Hw_Version]

Type: Synchronous

Description: Returns the hardware version

Parameters: The single byte, Hw_Version, return values are as follows:

BIT	DESCRIPTION
0 - 7	Magic number, which specifies the ISD-SR3000 processor's hardware version

Example: Get the ISD-SR3000 processor's magic number. The ISD-SR3000 responds that it is version 1.

Source	Byte Sequence	
Code	GHV	
Host Controller	02	AA
ISD-SR3000	02	01

GML	**Get Message Length**
Opcode:	**0x19**
Syntax:	**GML [message_length]**
Type:	**Synchronous**
Description:	**Returns the length of the current voicetag in multiples of 4 Kbytes (blocks).** **The returned value includes the voicetag directory information (64 bytes for the first block and 32 bytes for every other block), the voicetag data, and the entire last block of the voicetag, even if the voicetag occupies only a portion of the last block. Since a memory block includes 2048 bytes, the returned length may be bigger than the actual voicetag length by up to 2047 bytes. The minimum length of a voicetag is one block. If the current voicetag is undefined, the error ERR_INVALID is generated. See the Glossary for a complete description of this error.**
Parameters:	**message_length is 2 bytes long**

Example: Get the length of the current voicetag. The ISD-SR3000 responds: 4. (The voicetag occupies 8192 = 4 x 2048).

Source	Byte Sequence		
Code	GML		
Host Controller	19	AA	AA
ISD-SR3000	19	00	04

GMS **Get Memory Status**

Opcode: 0x12

Syntax: GMS type [mem_status]

Type: Synchronous

Description: Returns the total number of remaining memory blocks as a type 16bit unsigned integer. The parameter type must be 0. The estimated remaining recording time may be calculated as follows:

Time = (Num_of_blocks x 2048 x 8) / (Compression_rate x 1000)

Parameters:

Example: A command is issued to return the number of remaining memory blocks. The ISD-SR3000 responds: 40 blocks.

Source	Byte Sequence			
Code	GMS 00 AA AA			
Host Controller	12	00	AA	AA
ISD-SR3000	12	00	00	28

GMT **Get Message Tag**

Opcode: 0x04

Syntax: GMT [Tag_Ref]

Type: Synchronous

Description: Returns the 16-bit tag associated with the current voicetag. If the current voicetag is undefined, ERR_INVALID is reported.

Parameters: Tag_Ref is 2 bytes long

Example: Get the current message tag.

Source	Byte Sequence			
Code	GMT AA AA			
Host Controller	04	AA	AA	
ISD-SR3000	04	00	0E	

GNM **Get Number of Messages**

Opcode: **0x11**

Syntax: **GNM Tag_Ref Tag_Mask [Num_Of_Messages] which are 2 bytes each**

Type: **Synchronous**

Description: **Returns the number of voicetags whose message tags match the** *tag_ref* **parameters. Only bits set in** *tag_mask* **are compared. A match is considered successful if:**

message tag **and** *tag_mask* **=** *tag_ref* **and** *tag_mask*
where "and" is a bitwise AND operation.

The *tag_ref* **and** *tag_mask* **parameters are each two bytes. The return value is also two bytes long.**

If *tag_mask* **= 0, the total number of all existing messages is returned, regardless of the** *tag_ref* **value.**

Parameters: **Tag_Ref, Tag_Mask and Num_Of_Messages are all two bytes each**

Example: Get the number of voicetags which have bit 0 cleared, and bit 1 set in their message tags. The ISD-SR3000 processor responds there are five voicetags that satisfy the request.

Source	Byte Sequence						
Code	GNM FF FE 00 03 AA AA						
Host Controller	11	FF	FE	00	03	AA	AA
ISD-SR3000	11	FF	FE	00	03	00	05

GNR **Get Next Recognition**

Opcode: **0x41**

Syntax: **GNR [topic_id token_id]**

Type: **Synchronous**

Description: **Gets the token for the next word recognized. This command should be issued by the host controller after it is interrupted by ISD-SR3000, following ISD-SR3000's recognition of a word (as evidenced by the EV_RECO_QUEUE bit being set in the Status Word - see the GSW command). If the GNR command is issued when the EV_RECO_QUEUE bit is not set, the GNR command will execute but the data returned will not be valid.**

Parameters: **The GNR command returns 2 bytes, defined as follows:**

BYTE	DESCRIPTION
0	The Topic number in which the recognition event occurred. The ISD-SR3000 contains the following special topic indicators: 0XFD - a pause was found 0XFE - a noise was found
1	The Token number that recognition event matches. The special token number zero hex (00h) means a pause was found.

Example: The next recognition event is requested from ISD-SR3000. ISD-SR3000's response indicates:

- **this recognition event is from Topic 01**
- **the Token recognized is Token number 03**

Source	Byte Sequence		
Code	GNR		
Host Controller	0x41	AA	AA
ISD-SR3000	0x41	01	03

GNWL Get Number Added Acoustic Words

Opcode: 0x4E

Syntax: GNWL topic_id list_id [num_of_words]

Type: Synchronous

Description: Returns the number of acoustic words, num_of_words, that exist in list list_id in the topic of topic_id.

Parameters: topic_id and list_id are one byte each. num_of_words is two bytes.

Example: Get number of added acoustic words (returned four) for list 1 in topic 2.

Source	Byte Sequence				
Code	GNWL 0201 AAAA				
Host Controller	4e	02	01	AA	AA
ISD-SR3000	4e	02	01	00	04

GPC Gets Phoneme Count

Opcode: 0x54

Syntax: GPC [num_of_phonems]

Type: Synchronous

Description: Indicates the number of phonemes, num_of_phonems (two bytes long). This number is associated with the last processed recorded word (using the ROL command).

Parameters: num_of_phonems is two bytes

Example: Get the phoneme count (returned nine).

Source	Byte Sequence		
Code	GPC AA AA		
Host Controller	54	AA	AA
ISD-SR3000	54	00	09

GRE Get RECO Error code

Opcode: 0x56

Syntax: GRE [Reco_Error]

Type: Synchronous

Description: Returns the reco engine error word Reco_Error, which contains the errors that occurred in the recognition engine.

Parameters: Reco_Error is two bytes long.

Value (in Hex)	DESCRIPTION
0x9E4B	DWL error; bad phoneme on word. This indicates an unknown phoneme was returned from the ROL command.
0x9E4C	DWL error; too many senones. You have reached the number of senones limitation.
0x9E4D	DWL error; too many words. You have reached the limitation (50) number of words in one list.
0x9E56	DWL error; unknown word. This occurs when deleting an unknown word from a dynamic list (using the DAW command).
0x9E57	DWL error; unknown list. The list you have defined does not exist in the topic.
0xD120	Bad Reco environment. This might occur when trying to issue RE or ROL in a noisy environment.
0xD8F1	SR Error; This error will appear after running the reco engine using the RE command when no topic was enabled.

Example: Get the Reco_Error register values (0x9E57) indicating that an unknown list was issued.

Source	Byte Sequence		
Code	GRE AA AA		
Host Controller	56	AA	AA
ISD-SR3000	56	9E	57

GRV	**Get RECO program version**
Opcode:	**0x52**
Syntax:	**GVR [reco_program_version]**
Type:	**Synchronous**
Description:	**Returns the version of the recognition engine.**
Parameters:	**reco_program_version is two bytes long.**

Example: Get the reco revision (returned 0123).

Source	Byte Sequence		
Code	GRV AA AA		
Host Controller	52	AA	AA
ISD-SR3000	52	01	23

GSV	**Get System Program Version**
Opcode:	**0x53**
Syntax:	**GSV [system_program_version]**
Type:	**Synchronous**
Description:	**This command returns the ISD-SR3000 system firmware version**
Parameters:	**system_program_version is two bytes long**

Example: Get the ISD-SR3000 system version (returned 0300).

Source	Byte Sequence		
Code	GSV AA AA		
Host Controller	52	AA	AA
ISD-SR3000	52	03	00

GSW	**Get Status Word**
Opcode:	**0x14**
Syntax:	**GSW [status_word]**
Type:	**Synchronous**
Description:	**The ISD-SR3000 processor has a 16-bit status word to indicate events that occur during normal operation. The ISD-SR3000 processor asserts the MWRQST signal (driven low), to indicate a change in the status word. This signal remains active until the ISD-SR3000 processor receives a GSW command. The status word is cleared during reset and upon successful execution of the GSW command.**
Parameters:	**The GSW command returns a 16-bit status word, defined as follows:**

BIT	DESCRIPTION
5	EV_NORMAL_END 0: When this bit is zero, it means either: a) no command is underway, or b) a command is being processed but has not yet completed, or c) a command completed but had an error (as indicated by a '1' in the EV_ERROR bit) 1: Normal completion of an operation, e.g., end of playing of a prompt.
6	EV_MEMFUL 0: Memory is not full (when using the R command) 1: Memory is full
7	EV_ERROR 0: No error detected 1: Error detected in the last command. The host controller must issue the GEW command to return the error code and clear the error condition.
10	EV_MEMLOW 0: The ROL memory is not full (when using the RR command) 1: The ROL memory is full (when using the RR command)
11	EV_RECO_QUEUE 0: There are no arguments in the ISD-SR3000 recognition queue. 1: Reco has one or more recognition arguments in its queue. Use command GNR to retrieve items from the queue.
14	EV_RESET 0: Normally, this bit changes to 0 after performing the INIT command. 1: When the ISD-SR3000 processor completes its power-up sequence and enters the RESET state, this bit is set to 1, and the MWRQST signal is activated (driven low). Normally, this bit changes to 0 after performing the INIT command. If this bit is set during normal operation of the ISD-SR3000 processor, it indicates an internal ISD-SR3000 processor error. The host controller can recover from such an error by re-initializing the system.
15	ERR_RECO 0: No reco error 1: Reco error has occurred. The reco register contains the detailed information. (Use the GRE command to read the reco error register.)
All other bits	0 or 1: BIts are reserved and should be disregarded. These bits may return any mix if 0 and1.

Example: ISD-SR3000's error word is requested and returns code 0x20, indicating normal command completion (EV_NORMAL_END) status**.**

Source	Byte Sequence		
Code	GSW		
Host Controller	14	AA	AA
ISD-SR3000	14	00	20

GT	Generate Tone
Opcode:	0x0D
Syntax:	GT Tone
Type:	Asynchronous
Description:	Generates the tone specified by the 1-byte tone parameter. The ISD-SR3000 state changes to TONE_GENERATE. The tone generation continues until an S command is received. A DTMF or a single frequency tone may be generated as shown below.
Parameters:	Tone is one byte long. To generate a DTMF or a single frequency code, encode the bits as follows:

BIT	DESCRIPTION
0	1 Specify a DTMF
4-1	DTMF code, where the DTMF code is encoded as follows: **Value (Hex) DTMF Digit** 0 to 9 0 to 9 A A B * C # D B E C F D
5-7	0

BIT	DESCRIPTION
0	0 Specify a single tone
5-1	3-30 The value in bits 1-5 is multiplied by 100 to generate the required frequency (300Hz-3000Hz).
7-6	0

***Note**: The ISD-SR3000 processor does not check for the validity of the tone specification. Invalid specification yields unpredictable results.*

Example: ISD-SR3000 generates a single-frequency 1600Hz tone.

Source	Byte Sequence	
Code	GT 20	
Host Controller	0D	20
ISD-SR3000	0D	20

GTD	**Get Time and Day**
Opcode:	**0x0E**
Syntax:	**GTD [time_day_option]**
Type:	**Synchronous**
Description:	**Returns the time and day as a 2-byte value [time_day_value]. time_day_option must be set to: 0 This will get the system time and day.**
Parameters:	**The time_day_option is one byte long. The time_day_value is two bytes long and is encoded as follows:**

BIT	DESCRIPTION
2-0	Day of the week (1 through 7)
7-3	Hour of the day (0 through 23)
13-8	Minute of the hour (0 through 59)
15-14	00: The time was not set. 11: The time was set, i.e., the SETD (Set Time and Day) command was executed.

Note: *If the current message is undefined, and time_day_option is 1, an ERR_INVALID error is reported.*

Example: Get the current system time-and-day stamp. The ISD-SR3000 processor responds that the system date is the first day of the week and the time is 5:40 A.M.

Source	Byte Sequence			
Code	GTD 00 AA AA			
Host Controller	0E	00	AA	AA
ISD-SR3000	0E	00	E8	29

GTM Get Tagged Message

Opcode: 0x09

Syntax: GTM Tag_Ref Tag_Mask Dir [Result]

Type: Synchronous

Description: Selects the current message, according to instructions in Dir, to be the first, nth next or nth previous message, which complies with the equation:

message tag and *tag_mask* = *tag_ref* and *tag_mask*
where “and” is a bitwise AND operation.

Dir is one of the following:

0: Selects the first (oldest) message.
-64: Selects the last (newest) message.
n: Selects the nth next message starting from the current message.
–n: Selects the nth previous message starting from the current message.

To select the nth message with a given tag to be the current message, you must first select the first message that complies with the above equation, and then issue another GTM command with n – 1 as a parameter, to skip to the nth message.

The result value Result reports 1 when the message pointer is set. If the message pointer is not set, Result reports 0.

Parameters: Tag_Ref (2 bytes), Tag_Mask (2 bytes), Dir (1 byte) and Result (1 byte)

***Note**: If a message is found, it becomes the current message and 1 (TRUE) is returned. If no message is found, the current message remains unchanged and 0 (FALSE) is returned.*

If Dir is not 0, and the current message is undefined, the return value is unpredictable. After the command execution, the current message may either remain undefined or change to any existing message. The only exception is when the GTM command is executed just after the DM command. (See the DM command for further details.)

Example: Assuming the user arranged the message tag as an indication number received from the AAW command (using the upper byte to notate the topic number and the lower byte as the token number), the following will set the message pointer to the message with Tag_Ref = [0x02 0x06], *n*oting a previously added voice tag message associated with acoustic word 6 in topic 2. The results show that the pointer is set to the desired message.

Source	**Byte Sequence**						
Code	GTM 0206 FFFF 00 AA						
Host Controller	09	02	06	FF	FF	00	AA
ISD-SR3000	09	02	06	FF	FF	00	01

GTUNE **Get Tunable Parameter**

Opcode: **0x06**

Syntax: **GTUNE Index [parameter_value]**

Type: **Synchronous**

Description: **Gets the value of the tunable parameter identified by Index as the value, parameter_value. This command may be used to read and identify the parameter value that was set to tune the ISD-SR3000.**
If Index does not point to a valid tunable parameter, ERR_PARAM is set in the error word.
The GTUNE command may be used in IDLE state or RESET state. If TUNE command was not used to set the tunable parameters, then the GTUNE command will read the default parameter value.

Parameters: **Index is one byte long and parameter_value is 2 bytes long**

Example: Get the value of the return noise tokens flag (index number 0x53). The results indicate it to be enabled.

Source	Byte Sequence			
Code	GTUNE 53			
Host Controller	06	53	AA	AA
ISD-SR3000	06	53	00	01

INFG **Tag Data Information Get**

Opcode: **0x3E**

Syntax: **INFG data_offsett Num_Of_Bytes [$byte_1$..$byte_n$]**

Type: **Synchronous**

Description: **Retrieves the data information (Num_Of_Bytes bytes long) stored in the tag header, located at offset *data_offsett*, of the current message.**

Parameters: **data_offsett and Num_Of_Bytes are one byte long. The maximum number of bytes that can be stored in a message header is 74. This means that: 74>(data_offsett + Num_Of_Bytes).**

Note: the maximum number of bytes allowed to be written in one transaction is 32 bytes.

Example: Get the two byte data, '0123', located at offset '0' from the header of the current message.

Source	Byte Sequence				
Code	INFG 00 2 AA AA				
Host Controller	3E	00	2	AA	AA
ISD-SR3000	3E	00	2	01	23

INFS **Tag Data Information Set**

Opcode: **0x3D**

Syntax: **INFS data_offsett Num_Of_Bytes $byte_1$..$byte_n$**

Type: **Synchronous**

Description: **Resets the data information in the tag header, starting from offset *data_offsett* of the current message, for length of Num_Of_Bytes bytes.**

Note: the data in the Flash memory is set to FF and, by writing to it, the data is changed to 0. After resetting a bit in the Flash, you cannot set it to 1. (The memory is set back to FF after the erasing procedure).

Parameters: **data_offsett and Num_Of_Bytes are one byte long. The maximum number of bytes that can be stored in a message header is 74. This means: 74>(data_offsett + Num_Of_Bytes).**

Note: the maximum number of bytes allowed to be written in one transaction is 32 bytes.

Example: Set the two byte data, '0123', starting at offset '0' from the header of the current message.

Source	Byte Sequence				
Code	INFS 00 02 01 23				
Host Controller	3D	00	02	01	23
ISD-SR3000	3D	00	02	01	23

INIT Initialize System

Opcode: 0x13

Syntax: INIT

Type: Synchronous

Description: Execute this command after the ISD-SR3000 processor has been configured (see the CFG command). INIT Performs a soft reset of the ISD-SR3000 processor, which includes:

- Initializing the message directory information.
- Setting the volume level that is controlled by the VC commands, to 0.
- Switching to IDLE state.

Note: Messages are not deleted. To delete the messages, use the DM and DMS commands.

The current message is undefined after INIT execution.

The tunable parameters are not affected by this command. They are set to their default values only during RESET.

Parameters:

Example: ISD-SR3000 is initialized.

Source	Byte Sequence
Code	INIT
Host Controller	13
ISD-SR3000	13

PDM Go to Power-down mode

Opcode: 0x1A

Syntax: PDM

Type: Synchronous

Description: Switches the ISD-SR3000 processor to power-down mode. Sending any command while in power-down mode returns the ISD-SR3000 processor to normal operation mode. If an event report is pending (i.e., MWRQST is active) and it is not processed by the host controller prior to issuing the PDM command, the event is lost.

Parameters:

Example: This command tells ISD-SR3000 to go into power-down mode.

Source	Byte Sequence
Code	PDM
Host Controller	1A
ISD-SR3000	1A

PM Play Message

Opcode: 0x03

Syntax: PM

Type: Asynchronous

Description: Begins playback of the current message. The ISD-SR3000 processor state changes to PLAY. When playback is complete, the ISD-SR3000 processor sets the EV_NORMAL_END bit in the status word, and activates (clears to 0) the MWRQST signal. The state then changes to IDLE. Playback can be stopped with the S command. If the current message is undefined, ERR_INVALID is reported.

Parameters:

Example: Play the current message.

Source	Byte Sequence
Code	PM
Host Controller	03
ISD-SR3000	03

R	**Record Message**
Opcode:	**0x0C**
Syntax:	**R Tag_Ref Compression_Rate**
Type:	**Asynchronous**
Description:	**Records a new voicetag with message tag [Tag_Ref] and compression rate Compression_Rate. The ISD-SR3000 processor state changes to RECORD. The R command continues execution until stopped by the S command. If the memory becomes full, recording stops and EV_MEMFULL is set in the status word. The compression rate may be defined as 0 for PCM recording or either 1,2,3 for compression rates 4.7 Kbits/s, 6.7 Kbits/s, 8.7 Kbits/s respectively. See "VCD" for a description of the compression algorithm.**
Parameters:	**Tag_Ref is two bytes long and Compression_Rate is one byte long.**

Example: Record a new voicetag with a message tag 0x2001.

Source	Byte Sequence			
Code	R 20 01 03			
Host Controller	0C	20	01	03
ISD-SR3000	0C	20	01	03

RE Enable Recognition

Opcode: 0x40

Syntax: RE

Type: Asynchronous

Description: Turns the ISD-SR3000 recognition engine on, thus allowing it to start listening for command keywords.

Parameters:

Example: This command tells ISD-SR3000 to start the recognition engine.

Source	Byte Sequence
Code	RE
Host Controller	40
ISD-SR3000	40

ROL Recognize Offline

Opcode: 0x50

Syntax: ROL

Type: Synchronous

Description: Takes the last RR-recorded voicetag and produces a phonetic representation of the voicetag. Once this command is finished (indication of normal end in the status register), the controller can check the number of phonemes (GPC) and then add the phonetic representation to a list in a topic using the AAW command. Note: This command execution duration is several seconds long and depends on the recording time.

Parameters:

Example: Initiate the ROL command.

Source	Byte Sequence
Code	ROL
Host Controller	50
ISD-SR3000	50

RR **Record for Reco**

Opcode: 0x57

Syntax: RR Compretion_Rate

Type: Asynchronous

Description: Records a new voicetag with a message tag for recognition. The ISD-SR3000 processor state changes to RECORD. The RR command continues execution until stopped by the S command. The recording buffer is limited to a maximum of 8 seconds of A-Law or µ-Law, or 4 seconds of Linear speech. If the recording reaches these limits, the MEMORY_LOW bit is set (this buffer is erased by the ROL command). The result of the record is stored in a special buffer and can be used later for Recognition off-line. (See ROL command). Hence, after issuing the RR command, you must immediately call the ROL. You may never send two consecutive commands of RR or DM, and DMS may never follow the RR command. If any of these command violations occur, the ERR_COMMAND bit is set in the error register.
If the memory becomes full, recording stops and EV_MEMFULL is set in the status word. The compression rate may be defined as 0 for PCM recording or either 1,2,3 for compression rates 4.7 Kbits/s, 6.7 Kbits/s, 8.7 Kbits/s respectively.

Parameters: Compretion_Rate is one byte long.

Example: Record a new voicetag at compression rate of 8.7 Kbits/s (recommended).

Source	Byte Sequence	
Code	RR 03	
Host Controller	57	03
ISD-SR3000	57	03

S	**Stop**
Opcode:	**0x00**
Syntax:	**S**
Type:	**Synchronous**
Description:	**Stops execution of the current asynchronous command and switches the ISD-SR3000 processor to the IDLE state.**
Parameters:	

***Note**: After recognition is enabled, the Stop command can take up to one second until the ready is active.*

Example: Stop the current activity and return the ISD-SR3000 processor to the IDLE state.

Source	Byte Sequence
Code	S
Host Controller	00
ISD-SR3000	00

SAS	**Say Argumented Sentence**
Opcode:	**0x1E**
Syntax:	**SAS sentence_n tag_ref**
Type:	**Asynchronous**
Description:	**Announces sentence number sentence_n of the currently selected vocabulary and passes arg to it. When playing is complete, the ISD-SR3000 processor sets the EV_NORMAL_END bit in the status word and activates the MWRQST signal. If the current vocabulary is undefined, ERR_INVALID is reported. Tag_Ref can be replaced by an argument in the event the sentence was built differently by the IVS tool (See Chapter 3 for a detailed description of all the available arguments.)**
Parameters:	**sentence_n topic is one byte long and Tag_Ref (or Argument) is two bytes long**

Example: Announce the first sentence in the sentence table of the currently selected vocabulary with '3' as the actual parameter topic and '6' as the token ID.

Source	Byte Sequence			
Code	SAS 0306			
Host Controller	1E	00	03	06
ISD-SR3000	1E	00	03	06

SETD **Set Time and Day**

Opcode: **0x0F**

Syntax: **SETD time_and_day**

Type: **Synchronous**

Description: **Sets the system time and day as specified by the 2-bytes time_and_day parameter.**

Parameters: **The time_and_day parameter is encoded as follows:**

BIT	DESCRIPTION
2-0	Day of the week (1 through 7)
7-3	Hour of the day (0 through 23)
13-8	Minute of the hour (0 through 59)
15-14	Must be set to 1.

***Note**: If time_and_day value is not valid, ERR_PARAM is set in the error word.*

Example: Set time and day to Monday 1:30AM (where Monday is the first day of the week.)

Source	Byte Sequence	
Code	SETD DE09	
Host Controller	0F	DE 09
ISD-SR3000	0F	DE 09

SMT **Set Message Tag**

Opcode: **0x05**

Syntax: **SMT Tag_Ref**

Type: **Synchronous**

Description: **Sets the tag of the current message. To change the message tag, you should first get the tag using the GMT command, read the tag, modify it and write it back.**

Parameters: **Tag_Ref is two bytes long.**

***Note**: Message tag bits can only be cleared. Message tag bits are set only when a message is first created. If the current message is undefined, ERR_INVALID is reported.*

Example: Change bit 3 settings in the current pointed message. Note that the ISD-SR3000 processor ignores bits in the tag that are set to 1- only bit 3 is modified in the message tag.

Source	Byte Sequence		
Code	SMT FF F7		
Host Controller	05	FF	F7
ISD-SR3000	05	FF	F7

SO	**Say One Word**
Opcode:	**0x07**
Syntax:	**SO word_number**
Type:	**Asynchronous**
Description:	**Plays the word number word_number in the current vocabulary. When playback of the selected word has been completed, the ISD-SR3000 processor sets the EV_NORMAL_END bit in the status word and activates the MWRQST signal. If word_number is not defined in the current vocabulary, or if it is an IVS control or option code, ERR_PARAM is set in the error word. If the current vocabulary is undefined, ERR_INVALID is reported.**
Parameters:	**word_number is one byte long. word_number may be any value from 0 through the index of the last word in the vocabulary.**

Example: This command tells ISD-SR3000 to announce the first word in the word table of the currently selected vocabulary.

Source	Byte Sequence	
Code	SO 00	
Host Controller	07	00
ISD-SR3000	07	00

SPT	**Set Prompts Type**
Opcode:	**0x20**
Syntax:	**SPT type id**
Type:	**Synchronous**
Description:	**Selects the vocabulary table to be used for voice synthesis. The vocabulary type is set according to the type parameter:** **0: For compatibility only.** **1: External vocabulary in ROM.** **2: External vocabulary in Flash.** **3-7: Reserved.** **The host controller is responsible for selecting the current vocabulary, using the SPT command, before using any of the play commands, e.g.- SAS, SO, SS, SW. Each external vocabulary table has a unique id that is part of the vocabulary internal header (See the *IVS User's Guide* for more details). If type is 1 or 2, the ISD-SR3000 processor searches for the one byte id parameter in each vocabulary table header until a match is found. If the id parameter does not point to a valid IVS vocabulary, ERR_PARAM is set in the error word.**
Parameters:	**type and id are one byte long.**

Example: Select the vocabulary with vocabulary-id 3, which resides on a Flash as the current vocabulary.

Source	Byte Sequence		
Code	SPT 02 02		
Host Controller	20	02	03
ISD-SR3000	20	02	03

SRI	**Stop Recognition Immediately**
Opcode:	**0x5A**
Syntax:	**SRI**
Type:	**Synchronous**
Description:	**Stops recognition immediately by flushing all internal buffers. It is the same as the regular S command in all other aspects.**
Parameters:	

Example: Stop recognition process and immediately return the ISD-SR3000 processor to the IDLE state .

Source	Byte Sequence
Code	SRI
Host Controller	5A
ISD-SR3000	5A

SS	Say Sentence
Opcode:	1F hex
Syntax:	SS sentence_n
Type:	Asynchronous
Description:	Say sentence number sentence_n of the currently selected vocabulary. If the sentence has an argument, 0 is passed as the value for this argument. When playing has been completed, the ISD-SR3000 processor sets the EV_NORMAL_END bit in the status word and activates the MWRQST signal. If sentence_n is not defined in the current vocabulary, ERR_PARAM is set in the error word. If the current vocabulary is undefined, ERR_INVALID is reported.
Parameters:	sentence_n is one byte long

Example: This command tells ISD-SR3000 to announce the first sentence in the sentence table of the currently selected vocabulary.

Source	Byte Sequence	
Code	SS 00	
Host Controller	1F	00
ISD-SR3000	1F	00

SW Say Words

Opcode: 0x21

Syntax: SW n word1...wordn

Type: Asynchronous

Description: Plays n words, indexed by word1 to wordn. On completion, the EV_NORMAL_END bit in the status word is set and the MWRQST signal goes low. If one of the words is not defined in the current vocabulary, or if it is an IVS control or option code, or if n > 20, ERR_PARAM is reported. If the current vocabulary is undefined, ERR_INVALID is reported.

Parameters: The parameters n, word1 through wordn, are each one byte

Example: Announce, twice, the first word in the word table of the currently selected vocabulary.

Source	Byte Sequence			
Code	SW 02 00 00			
Host Controller	21	02	00	00
ISD-SR3000	21	02	00	00

TOPD Topics Disable

Opcode: 0x44

Syntax: TOPD topic_id

Type: Synchronous

Description: Disables selected topics. If the topic number does not exist, this command sets bit ERR_PARAM in the Error Word (see the GEW command on page 2-30). topic_id identifies the topic to be disabled. The special value FF (hex) disables all topics.

Parameters: topic_id is one byte

Example: ISD-SR3000 is instructed to disable Topic number 10 (hex).

Source	Byte Sequence	
Code	TOPD 10	
Host Controller	0x44	10
ISD-SR3000	0x44	10

TOPE Topic Enable

Opcode: 0x43

Syntax: TOPE topic_id

Type: Synchronous

Description: Enables selected topics. If the topic number does not exist, this command sets bit ERR_PARAM in the Error Word (see the GEW command on page 2-30). The value topic_id identifies the topic to be enabled. The special value FF (hex) enables all topics.

Parameters: topic_id is one byte long

Example: ISD-SR3000 is instructed to enable Topic number 05 (hex).

Source	Byte Sequence	
Code	TOPE 05	
Host Controller	43	05
ISD-SR3000	43	05

TOPL	**Topic Load**
Opcode:	**0x5B**
Syntax:	**TOPL topic_id**
Type:	**Synchronous**
Description:	**Loads topic *topic_id* from external ROM/FLASH to the RAM. NOTE: A topic must be loaded to the ISD-SR3000 RAM before it can be enabled with the TOPE command.**
Parameters:	**topic_id is one byte long**

Example: Load topic 8 from the external non volatile memory.

Source	Byte Sequence	
Code	TOPL 08	
Host Controller	5b	08
ISD-SR3000	5b	08

TOPS	**Topic Save**
Opcode:	**0x4D**
Syntax:	**TOPS topic_id**
Type:	**Synchronous**
Description:	**Saves topic *topic_id* into the external non volatile memory. It is strongly recommended to use this command for a topic with a dynamic list after adding a new word (using the AAW command)**
Parameters:	**topic_id is one byte long**

Example: Save topic 8 to the external non volatile memory.

Source	Byte Sequence	
Code	TOPS 08	
Host Controller	4d	08
ISD-SR3000	4d	08

TOPU	**Topic Unload**
Opcode:	**0x5C**
Syntax:	**TOPU topic_id**
Type:	**Synchronous**
Description:	**Unloads topic topic_id from the RAM. NOTE: The host controller must load a topic before enabling it.**
Parameters:	**topic_id is one byte long.**

Example: Load topic 8 from the external non volatile memory.

Source	Byte Sequence	
Code	TOPU 08	
Host Controller	5c	08
ISD-SR3000	5c	08

TUNE	**Tune**
Opcode:	**0x15**
Syntax:	**TUNE index parameter_value**
Type:	**Synchronous**
Description:	**Sets the value of the tunable parameter identified by index to the value, parameter_value. This command may be used to tune the DSP algorithms or change other parameters. If you do not use TUNE, the ISD-SR3000 processor uses default values. If index does not point to a valid tunable parameter, ERR_PARAM is set in the error word.**
Parameters:	**index is one byte long and parameter_value is 2-byte long**

***Note**: The tunable parameters are assigned with their default values on application of power. The INIT command does not affect these parameters.*

Example: Change the ISD-SR3000 to enable the return of noise tokens.

Source	Byte Sequence			
Code	TUNE 53 0001			
Host Controller	15	53	00	01
ISD-SR3000	15	53	00	01

VC **Volume Control**

Opcode: 0x28

Syntax: VC vol_level

Type: Synchronous

Description: Controls the energy level of all the voice outputs. The resolution is ±3 dB. The actual output level is composed of the tunable VCD_PLAY_LEVEL variable, plus the VOL_LEVEL. For example, if the tunable variable VCD_LEVEL is 6, and vol_level is -2, then the output level equals VCD_PLAY_LEVEL + VOL_LEVEL = 4.

Parameters: vol_level is one byte long

Example: ISD-SR3000 is instructed to set the vol_level to +5 (hex).

Source	Byte Sequence	
Code	VC 05	
Host Controller	28	05
ISD-SR3000	28	05

2.12 TUNABLE PARAMETERS

The tunable parameters are arguments that control some of the ISD-SR3000's functionality. These parameters can be read using the GTUNE command or changed using the TUNE command.

Table 2-16: Tunable Parameters Page Reference by Parameter Name (in Alphabetical Order)

Parameter Name	Index Number	Page Number in this Chapter
Channel 0 Delay: CFRD0	72	page 2-66
Channel 1 Delay: CFRD1	73	page 2-66
Channel 2 Delay: CFRD2	74	page 2-66
Data Valid Delay: CFET	76	page 2-66
DTMF Generation: DTMF_GEN_TWIST_LEVEL	26	page 2-65
Frame Synch Delay: CFSD	75	page 2-66
Noise Token Enable: Noise_Report_Flag	83	page 2-66
Tone Generation: TONE_GEN_LEVEL	15	page 2-65
VCD Playback and Voice Synthesis: VCD_PLAY_LEVEL	20	page 2-65
Dynamic Word Usage: WITH_DYNAMIC_WORDS	88	page 2-66

Table 2-17: Tunable Parameters LIsted by Index Number

Index	Parameter Name	Description	Default
15	Tone Generation: TONE_GEN_LEVEL	Controls the energy level at which DTMF and other tones are generated. Each unit represents 3 dB. The default level is the reference level. For example, if you set this parameter to 4, the energy level is 6 dB less than the default level. The actual output level is the sum of TONE_GEN_LEVEL and the vol_level parameter, controlled by the VC command. The tones are distorted when the level is set too high. Legal values: 0 ≤ TONE_GEN_LEVEL + vol_level ≤12.	6
20	VCD Playback and Voice Synthesis: VCD_PLAY_LEVEL	Controls the energy during playback and external voice synthesis. Each unit represents 3 dB. The default level is the reference level. For example, if you set this parameter to 4, the energy level is 6 dB less than the default level. The actual output level is the sum of VCD_PLAY_LEVEL and the vol_level parameter (controlled by the VC command). Speech is distorted when the level is set too high. Legal values: 0 ≤ VCD_PLAY_LEVEL + vol_level ≤ 12.	6
26	DTMF Generation: DTMF_GEN_TWIST_LEVEL	A one-byte value that controls the twist level of a DTMF tone, generated by the GT command, by controlling the energy level of each of the two tones (low frequency and high frequency) composing the DTMF tone. The Least Significant Nibble (LSN) controls the low tone and the Most Significant Nibble (MSN) controls the high tone. The energy level of each tone, as measured at the output of a TP3054 codec (before the DAA) connected to the ISD-SR3000 processor, is summarized in the following table: Nibble Value — Tone Energy (dB-Volts) 0 — 0 1 — –17.8 2 — –14.3 3 — –12.9 4 — –12.4 5 — –12.0 6 — –11.9 7 — –11.85 8 - 15 — –11.85 The volume of the generated DTMF tone during measurements was 6. (TONE_GEN_LEVEL+vol_level = 6). Note: the vol_level parameter is controlled by the VC command. For the default level, the high tone is –14.3 dBV and the low tone is –12.4 dBV, which gives a DTMF twist level of 1.9 dB. The energy level of a single generated tone is the level of the low tone.	66

Index	Parameter Name	Description	Default
72	Channel 0 Delay: CFRD0	The delay of codec channel 0 from Frame Synch 0 (CFS0) to start of valid data. Legal values: 0 to 255	1
73	Channel 1 Delay: CFRD1	The delay of codec channel 1 from Frame Synch 0 (CFS0) to start of valid data. Legal values: 0 to 255	9
74	Channel 2 Delay: CFRD2	The delay of codec channel 2 from Frame Synch 0 (CFS0) to start of valid data. Legal values: 0 to 255	17
75	Frame Synch Delay: CFSD	The delay of Frame Synch 1 (CFS1) from Frame Synch 0 (CFS0). Legal values: 0 to 255	16
76	Data Valid Delay: CFET	The delay between Frame Synch 0 (CFS0) to end of valid data of all channels. Legal values: 0 to 255	25
83	Noise Token Enable: Noise_Report_Flag	When set to one, this tunable enables the return of noise tokens when they have identified by the ISD-SR3000.	0
88	Dynamic Word Usage: WITH_DYNAMIC_WORDS	This tunable is used when working with applications that do not support dynamic words. ***Note**: without changing this tunable, the application will require the XYZ table in the Flash, which is unnecessary for non-dynamic-word applications. This table is in use when extracting the phonemes from a recorded voice tag.*	1

2.13 EXAMPLE OPERATION PROCEDURES

2.13.1 ISD-SR3000 INITIALIZATION

- Reset the ISD-SR3000 processor by activating the RESET signal.
- Issue a CFG (Configure ISD-SR3000) command to change the configuration according to your environment.
- Issue an INIT (Initialize System) command to initialize the ISD-SR3000 firmware.
- Issue a TUNE (Tune Parameter) command to change some of the ISD-SR3000 tunable parameters (only if you must change their default values. Refer to the Tunable Parameters table on page 2-64.)

2.13.2 NORMAL OPERATION PROCEDURE

The following application contains grammar and prompts as follows:

Grammar (tokens are automatically indexed, in alphabetical order, by the compiler(s)

<Topic 1>

Commands:

HANG UP, MUTE, ANSWER PHONE, STORE NAME, CALL NAME

0- ANSWER

1- CALL

2- HANG

3- MUTE

4- NAME

5- PHONE

6- STORE

7- UP

<Topic 2>

0- NO

1- YES

<Topic 3>

0- EIGHT

1- FIVE

2- FOUR

3- NINE

4- OH

5- ONE

6- SEVEN

7- SIX

8- THREE

9- TWO

A- ZERO

<Topic 4>

0- Dynamic List

Prompts (by their hexidecimal values)-

0 to 9- Zero, One, ..., Nine

A- Go Ahead

B- Muted

C- Good By

D- Please Say The Name You Want To Store

E- Please Repeat The Name

F- Please Say The Telephone Number

10- Is This Correct

To initialize the recognition engine, the host controller will send the following commands to load topics 1, 2, 3 and 4 to the RAM, and enable topic 1:

```
TOPL 1
TOPE 1
TOPL 2
TOPL 3
TOPL 4
```

Once all the desired topics are loaded and enabled, the host controller should enable the recognition:

```
RE
```

At this point the recognition engine starts processing the incoming data from the system CODEC. When the user pronounces the command "Answer Phone", the MWRQST signal goes low and the host controller should perform the following commands:

Note: *return values are marked with brackets.*

```
GSW [08 00]
```

The status word returned 08 00 indicating that a recognition result is in the recognition buffer. The host controller should retrieve the words until the buffer is empty (noted by FF FF) as follows:

```
GNR [01 00]
GNR [01 05]
GNR [FD 00]
GNR [FF FF]
```

Using this internal look up table, the host controller interprets the words to be ANSWER PHONE and responds by playing a prompt "Go Ahead".

Note*: the host controller must first stop the recognition engine before playing a prompt; it is recommended that you use the immediate stop command (SRI) for the quickest completion of the stop command.*

```
SRI
SW 1 A
```

At this point the host controller should wait for the EV_NORMAL_END bit to be set in the status register (0x20), indicating that the prompt finished playing (The host controller will get the status register after the MWRQST line is asserted)

GSW [00 20]

The host controller can now enable the recognition engine and wait for the next command. The following set of commands will demonstrate a similar procedure, but this time the user will say "HANG UP" and the prompt will be "Good Bye":

```
RE
```

(Wait for $\overline{\text{MWRQST}}$ to go low)

Note: Every time a required token is entered into the reco queue, the EV_RECO_QUEUE is set, causing the $\overline{\text{MWRQST}}$ to go low. This means that a fast reading of the reco queue can cause or result in several assertions of the $\overline{\text{MWRQST}}$.

```
GSW [08 00]
GNR [01 02]
GNR [01 07]
GNR [FD 00]
GNR [FF FF]
SRI
SW 1 B
```

(Wait for $\overline{\text{MWRQST}}$ to go low)

```
GSW [00 20]
RE
```

(Wait for $\overline{\text{MWRQST}}$ to go low)

2.13.3 ADD VOICE TAG PROCEDURE

In this section we will describe the procedure for adding a new user word, the voicetag. This example assumes that the recognition engine is enabled, the user said "STORE NAME" and the ISD-SR3000 asserted the $\overline{\text{MWRQST}}$ signal:

```
GSW [08 00]
GNR [01 06]
GNR [01 04]
GNR [FD 00]
GNR [FF FF]
SRI
SW 1 D
```

(Wait for $\overline{\text{MWRQST}}$ to go low)

```
GSW [00 20]
```

These first few commands are similar to the previously described procedure for extracting words from the ISD-SR3000. After the host controller identifies the "STORE NAME" command, it should execute the Record for Reco command:

```
RR
```

The host controller should wait for 4 seconds before stopping the recording (using the Stop command). Then the host controller should disable all of the topics, load and enable only topic 0 (the phonemes topic) and run the ROL command for extracting the phoneme sequence from the recording:

```
S
TOPD FF
TOPL 0
TOPE 0
ROL
```

(Wait for $\overline{\text{MWRQST}}$ to go low)

At this stage the host controller can retrieve the number of phonemes extracted from the recording for evaluating whether to continue with the procedure or prompt the user to record

again. The more phonemes the voice tag has, the better the received recognition results. (A recommended threshold number is seven phonemes). Assuming the host controller received 12 phonemes and continued, it then should add the phoneme information of the voice tag to a dynamic list in a topic (as topic 4 with dynamic list number 0) using the AAW command. The return value would be a token number that identifies the new word.

```
GPC [00 0A]
AAW 04 00 [01]
```

After the host controller added the new word to a topic, it will now conduct a verification procedure by disabling all topics and enabling the one with the new word. It will then prompt the user to repeat the name.

```
TOPD FF
TOPE 4
SW 1 E
```

(Wait for $\overline{\text{MWRQST}}$ to go low)

```
GSW [00 20]
RE
```

(Wait for $\overline{\text{MWRQST}}$ to go low)

```
GSW [08 00]
GNR [04 01]
GNR [FF FF]
```

After the host controller confirms the returned token to be the same as the AAW command previously returned, it should set the tag of the recorded message to correspond with the token and topic so that in the future it can easily play this message again.

```
SRI
SMT 04 01
```

The host controller can now continue to prompt the user for information to be associated with the new word, in our case a telephone number. After the host controller plays the appropriate prompt, it disables all topics and enables the digit topic.

```
SRI
SW 1 5
```

(Wait for $\overline{\text{MWRQST}}$ to go low)

```
GSW [00 20]
TOPD FF
TOPE 3
RE
```

(Wait for $\overline{\text{MWRQST}}$ to go low)

```
GSW [08 00]
GNR [03 05]
```

```
GNR [03 09]
GNR [03 08]
GNR [03 02]
GNR [FF FF]
```

The host controller will interpret this number sequence to be "1234". The host controller can confirm its findings with the user by playing back the numbers to the user and prompting the user with a yes/no question: "Is this correct" (prompt number 10). The host controller then enables the yes/no topic and waits for an answer.

```
SW 5 1 2 3 4 10
```

(Wait for $\overline{\text{MWRQST}}$ to go low)

```
GSW [00 20]
TOPD FF
TOPE 2
RE
```

(Wait for $\overline{\text{MWRQST}}$ to go low)

```
GSW [08 00]
GNR [02 01]
GNR [FF FF]
```

The host controller received a "Yes" answer and must now either store the information in its memory or attach it to the recording of the voice tag by using the INFS command as follows:

```
INFS 0 4 1 2 3 4
```

This command will store the information on the message information header starting from offset 0 and can later be extracted by the host controller using the INFG command.

Now that a new word has been added to topic 4, the host controller must ask the ISD-SR3000 to save this topic (TOPS command) back into its non volatile memory so that the next time the system loads, the updated topic will be loaded.

```
TOPS 4
```

This finalizes the procedure for adding a user voice tag to the ISD-SR3000, and the host controller can continue to the next state in its state machine. If the next state is returning to the main menu, the following commands should be assigned:

```
TOPD FF
TOPE 1
RE
```

Chapter 3—INTERNATIONAL VOCABULARY SUPPORT (IVS) and the IVS Tool

3.1 INTERNATIONAL VOCABULARY SUPPORT (IVS)

IVS is a mechanism by which the ISD-SR3000 processor utilizes several vocabularies stored on an external storage device. IVS enables the ISD-SR3000 to synthesize messages with the same meaning, but in different languages, from separate vocabularies.

3.2 IVS FEATURES

- **Multiple vocabularies stored on a single storage device.**
- **Plug-and-play. The same Host Controller code is used for all languages.**
- **Synthesized and recorded messages use the same voice compression algorithm to achieve equal quality.**
- **Argumented sentences. (For example:** *You have <n> messages*.**)**
- **Auto-synthesized time-and-day stamp (driven by the ISD-SR3000 processor's clock).**
- **Support for various language and sentence structures:**
  - **One versus many. (For example: You** *have one message* **versus** *You have two messages*.**)**
  - **None versus many. (For example:** *You have no messages* **versus** *You have two messages*.**)**
  - **Number synthesis (English**—*Eighty* **versus French—***Quatre-vingt***).**
  - **Word order (English—***Twenty-one* **versus German—***Einundzwanzig***).**
  - **Days of the week (Monday through Sunday versus Sunday through Saturday).**

3.3 THE IVS TOOL

The IVS tool includes two utilities:

1. **The DOS-based IVS Compiler**
2. **IVSTOOL for Windows. A Windows 95/98/2000 utility**

The tools help create vocabularies for the ISD-SR3000 processor. They take you from designing the vocabulary structure, through defining the vocabulary sentences, to recording the vocabulary words.

IVS COMPILER

The IVS compiler runs on MS-DOS (version 5.0 or later) and enables you to insert your own vocabulary, (i.e., basic words and data used to create numbers and sentences, as directories and files in MS-DOS). The IVS compiler then outputs a binary file containing that vocabulary. In turn, this information can be burned into an EPROM or Flash memory to be used by the ISD-SR3000 software.

IVS VOICE COMPRESSION

Each IVS vocabulary can be compiled with either the 4.7 Kbit/s, the 6.7 Kbit/s or the 8.7 Kbit/s voice compression algorithm, or in PCM format. Define the bit rate before compilation. The ISD-SR3000 processor automatically selects the required voice decompression algorithm when the SV command chooses the active vocabulary.

GRAPHICAL USER INTERFACE (GUI)

The IVS package includes a Windows utility to assist the vocabulary designer to synthesize sentences. With this utility, you can record words, compose sentences and listen to words and sentences in the specific compression rate quality selected.

3.3.1 HOW TO USE THE IVS TOOL WITH THE ISD-SR3000 PROCESSOR

The IVS tool creates IVS vocabularies, and stores them as a binary file. This file is burnt into a ROM device or programmed into a Flash memory device. The ISD-SR3000 processor SPT (Set Prompt Type) command is used to select the required vocabulary. The SW (Say Words), SO, SS (Say Sentence) and SAS (Say Argumented Sentence) commands are used to synthesize the required word or sentence. The typical vocabulary-creation process using the IVS Tool software is as follows:

1. Design the vocabulary.
2. Create the vocabulary files (as described in detail below). Use IVS TOOL for Windows to simplify this process.
3. Record the words using any standard PC sound card and sound editing software, that can create.wav files.
4. Run the IVS compiler to compress the.wav files, and compile them and the vocabulary tables into an IVS vocabulary file.
5. Repeat steps 1 to 4 to create a separate IVS vocabulary for each language that you want to use. Note that each language file must have a different ID number. To specify an ID Number, go to the "Vocabulary ID" option in the Vocabulary/Build Options menu.
6. Exit IVS Tool and burn the IVS vocabulary files into a ROM (or Flash memory) device.

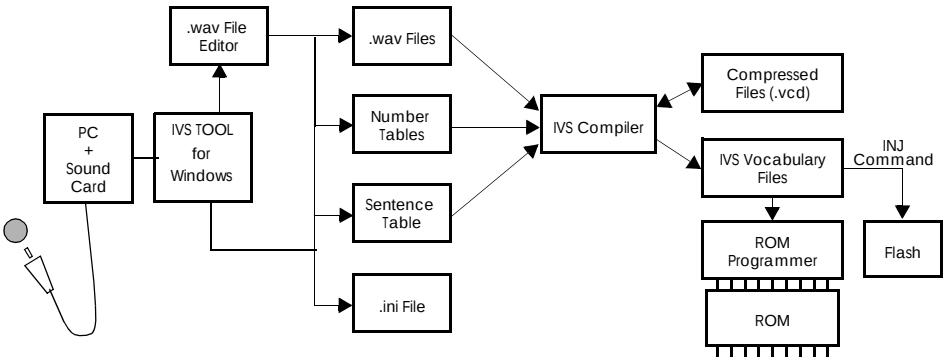
   Note: to concatenate vocabularies with multiple languages into one file, complete the following steps.

- **Load each vocabulary into flash. You must load the vocabularies in consecutive blocks on the flash and the order in which you load the vocabularies dictates the order in which they are available on the flash. For example, the first vocabulary you load becomes vocabulary #1 on the flash.**
- **The two files are now combined into a single file.**
- **To switch between the two languages in real time operation, use the SPT command. (See page 2-56 for details.) The SPT *type id* command switches between specified vocabulary IDs.**

Once the vocabulary is in place, the speech synthesis commands of the ISD-SR3000 processor can be used to synthesize sentences.

Figure 3-39 shows the vocabulary-creation process for a single table on a ROM or Flash memory device.

Figure 3-39: Creation of an IVS Vocabulary



Chapter 4—Glossary

This glossary contains the following information

4.1 ERROR CODES AND EXPLANATIONS

ERR_PARAM (bit 3 in error word) occurrences:

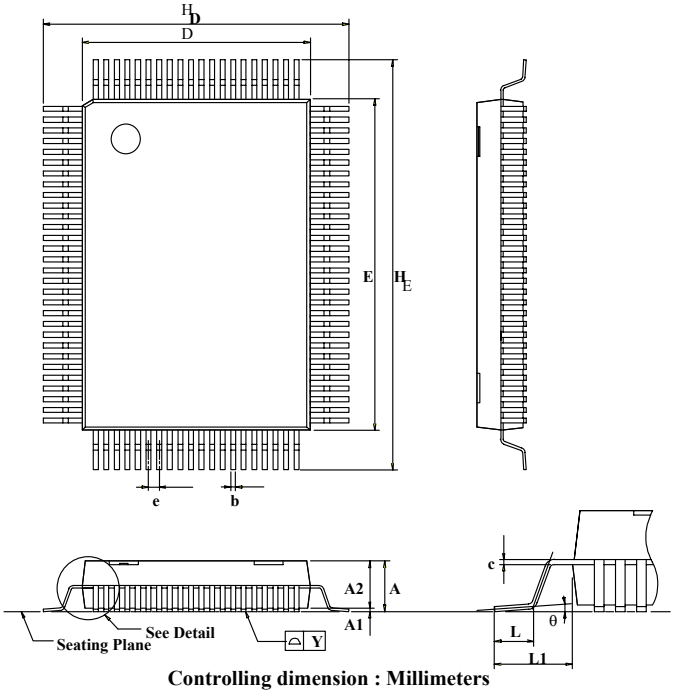
COMMAND	PARAMETER	LEGAL VALUE
RR (record for reco)	compression_type	0 - 3
RECORD	compression_type	0 - 3
SO (Say One Word)	word index	up to number of words in IVS
SW (Say Words)	word index	up to number of words in IVS
	number of words	up to MAX_SENT_LEN (currently 20)
SAS (Say Argumented Sentence)	sentence number	up to number of sentences in IVS
SS (Say Sentence)	sentence number	up to number of sentences in IVS
SETD (Set Date)	minute	0 - 59
	hour	0 - 23
	day	1 - 7
TUNE	tune_index	up to number of tunables (currently 86)
SPT (Set Prompt Type)	ivs_type	IVS_external (1)
All Topics Command: TOPS/ TOPD/TOPE/UTOP/LTOP	topic index	up to Max Topics all except LTOP also check that the topic is already loaded

ERR_INVALID (bit 7 in error word)
This invalid error occurs in the following conditions:

- **while trying to use a command that refers to the current message when no message is declared current. These commands include: PLAY, CMT, DM, GML, GMT, INFG, INFS.**
- **when IVS is not set**
- **when using GTD (Get Time and Date) before setting a date**
- **when the RR command is not immediately followed by the ROL command. For example: if RR is called and followed by TOPS, RR, DMS or DM, then the invalid error is generated.**

Chapter 5—Device Physical Dimensions

Figure: 5-1:100L PQFP (14x20x2.75mm footprint 3.2mm) IQC



Controlling dimension : Millimeters

Symbol	Dimension in inch			Dimension in mm		
	Min	Nom	Max	Min	Nom	Max
A	—	—	0.130	—	—	3.30
A1	0.010	0.014	0.018	0.25	0.35	0.45
A2	0.101	0.107	0.113	2.57	2.72	2.87
b	0.008	0.012	0.016	0.20	0.30	0.40
c	0.004	0.006	0.008	0.10	0.15	0.20
D	0.547	0.551	0.555	13.90	14.00	14.10
E	0.783	0.787	0.791	19.90	20.00	20.10
e	0.020	0.026	0.032	0.50	0.65	0.80
H_D	0.669	0.677	0.685	17.00	17.20	17.40
H_E	0.905	0.913	0.921	23.00	23.20	23.40
L	0.025	0.031	0.037	0.65	0.80	0.95
L1	—	0.063	—	—	1.60	—
Y	—	—	0.003	—	—	0.08
θ	0°	—	7 °	0°	—	7 °