National Semiconductor

March 1996

NSAM266SA CompactSPEECH™ Digital Speech Processor with Serial Flash Interface

General Description

The NSAM266SA is a member of National Semiconductor's CompactSPEECH Digital Speech Processor family. This processor provides Digital Answering Machine (DAM) functionality to embedded systems.

The CompactSPEECH interfaces with National Semiconductor's NM29A040 and NM29A080 Serial Flash memory devices to provide a cost-effective solution for DAM and Cordless DAM (CDAM) applications.

The CompactSPEECH processor integrates the functions of a traditional Digital Signal Processing (DSP) chip and the CR16A, a 16-bit general-purpose RISC core implementation of the CompactRISC™ architecture. It contains system support functions such as Interrupt Control Units, Codec interface, MICROWIRE™ interfaces to a microcontroller and Serial Flash, WATCHDOG™ timer, and a Clock Generator.

The CompactSPEECH processor operates as a slave peripheral that is controlled by an external microcontroller via a serial MICROWIRE interface. In a typical DAM environment, the microcontroller controls the analog circuits, buttons and display, and activates the CompactSPEECH by sending it commands. The CompactSPEECH processor executes the commands and returns status information to the microcontroller.

The CompactSPEECH firmware implements voice compression and decompression, tone detection and generation, message storage management, speech synthesis for time-and-day stamp, and supports user-defined voice prompts in various languages.

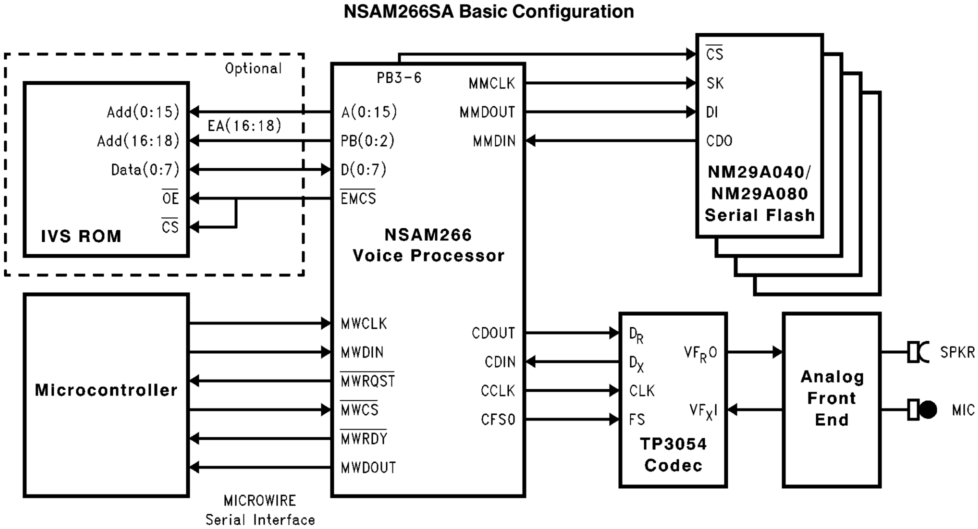
The CompactSPEECH implements echo-cancellation techniques to support high-quality DTMF tone detection during message playback.

The CompactSPEECH can synthesize messages in various languages via the International Vocabulary Support (IVS) mechanism. The NSAM266SA can store vocabularies on either Serial Flash, or Expansion ROM memories. DAM manufacturers can thus create machines that "speak" in different languages, simply by using other vocabularies. For more details about IVS, refer to the *IVS User's Manual.*

1.0 Hardware

1.1 BLOCK DIAGRAMS

NSAM266SA Basic Configuration



TL/EE/12584–1

 RRD-B30M46/Printed in U. S. A. http://www.national.com

Features

- Designed around the CR16A, a 16-bit general-purpose RISC core implementation of the CompactRISC architecture
- 20.48 MHz operation
- On-chip DSP Module (DSPM) for high-speed DSP operations
- On-chip codec clock generation and interface
- Power-down mode
- Selectable speech compression rate of 5.2 kbit/s or 7.3 kbit/s with silence compression
- Up to 16 minutes recording on a 4-Mbit Serial Flash (more than 1 hour total recording time on four devices)
- The number of messages that can be stored is limited only by memory size
- MICROWIRE slave interface to an external microcontroller
- MICROWIRE master interface to Serial Flash memory devices
- Storage and management of messages
- Programmable message tag for message categorization, e.g., Mailboxes, InComing Messages (ICM), OutGoing Messages (OGM)
- Skip forward or backward during message playback
- Digital volume control
- Variable speed playback
- Supports external vocabularies, using Serial Flash or expansion ROM
- Multi-lingual speech synthesis using International Vocabulary Support (IVS)
- Vocabularies available in: English, Japanese, Mandarin, German, French and Spanish
- DTMF generation and detection
- DTMF detection during OutGoing Message playback
- Single tone generation
- Telephone line functions, including busy and dial tone detection
- Call screening (input signal echoed to codec output)
- Real-time clock
- Direct access to message memory
- Supports long-frame and short-frame codecs
- Supports up to four 4-Mbit, or two 8-Mbit, Serial Flash devices
- Supports prerecorded IVS and OGM on Serial Flash
- Available in 68-pin PLCC and 100-pin PQFP packages

Table of Contents

1.0 HARDWARE

1.1 Block Diagrams

1.2 Pin Assignment

1.2.1 Pin—Signal Assignment

1.2.2 Pin Assignment in the 68-PLCC Package

1.2.3 Pin Assignment in the 100-PQFP Package

1.3 Functional Description

1.3.1 Resetting

1.3.2 Clocking

1.3.3 Power-down Mode

1.3.4 Power and Grounding

1.3.5 Memory Interface

1.3.6 Codec Interface

1.4 Specifications

1.4.1 Absolute Maximum Ratings

1.4.2 Electrical Characteristics

1.4.3 Switching Characteristics

1.4.4 Synchronous Timing Tables

1.4.5 Timing Diagrams

2.0 SOFTWARE

2.1 Overview

2.1.1 DSP-based Algorithms

2.1.2 System Support

2.1.3 Peripherals Support

2.2 CompactSPEECH Commands—Quick Reference Table

2.3 The State Machine

2.4 Command Execution

2.5 Tunable Parameters

2.6 Messages

2.6.1 Message Tag

2.7 Speech Compression

2.8 Tone and No-Energy Detection

2.9 Speech Synthesis

2.9.1 Explanation of Terms

2.9.2 Vocabulary Design

2.9.3 IVS Vocabulary Components

2.9.4 The IVS Tool

2.9.5 How to Use the IVS Tool With the CompactSPEECH

2.10 Initialization

2.11 Microwire Serial Interface

2.12 Signal Description

2.12.1 Signal Use in the Interface Protocol

2.12.2 Interface Protocol Error Handling

2.13 The Master Microwire Interface

2.13.1 Master MICROWIRE Data Transfer

2.14 Command Description

APPENDIX A

SCHEMATIC DIAGRAMS

1.0 Hardware (Continued)

1.2 PIN ASSIGNMENT

The following sections detail the pins of the NSAM266SA processor. Slashes separate the names of signals that share the same pin.

1.2.1 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.

TABLE 1-1. CompactSPEECH Pin—Signal Assignment

Pin Name	Type	Signal Name	I/O
A(0:15)	TTL	A(0:15)	Output
CCLK	TTL	CCLK	Output
CDIN	TTL	CDIN	Input
CDOUT	TTL	CDOUT	Output
CFS0	TTL	CFS0	Output
D(0:7)	TTL	D(0:7)	I/O
$\overline{\text{MWCS}}$	TTL (Note A)	$\overline{\text{MWCS}}$	Input
$\overline{\text{TST}}$	TTL	$\overline{\text{TST}}$	Input
$\overline{\text{MWRDY}}$	TTL	$\overline{\text{MWRDY}}$	I/O
$\overline{\text{MWRQST}}$	TTL	$\overline{\text{MWRQST}}$	I/O
MWDOUT	TTL	MWDOUT	Output
PB(0:2) (Note B)	TTL	EA(16:18)	Output
PB(3:6) (Note C)	TTL	CS(0:3)	Output
$\overline{\text{EMCS}}$/ ENV0	TTL1 (Note D) CMOS (Note E)	$\overline{\text{EMCS}}$ ENV0	Output Input
MWCLK	TTL	MWCLK	Input
MWDIN	TTL	MWDIN	Input
MMCLK	TTL1 (Note D)	MMCLK	Output
MMDIN	TTL	MMDIN	Input
MMDOUT	TTL1 (Note D)	MMDOUT	Output
CFS0	CMOS	CFS0	Output
$\overline{\text{RESET}}$	Schmitt (Note A)	$\overline{\text{RESET}}$	Input
V_{CC}	Power	V_{CC}	
V_{SS}	Power	V_{SS}	
X1	XTAL	X1	OSC
X2/CLKIN	XTAL TTL	X2 CLKIN	OSC Input

Note A: Schmitt trigger input.

Note B: Virtual address lines for IVS ROM.

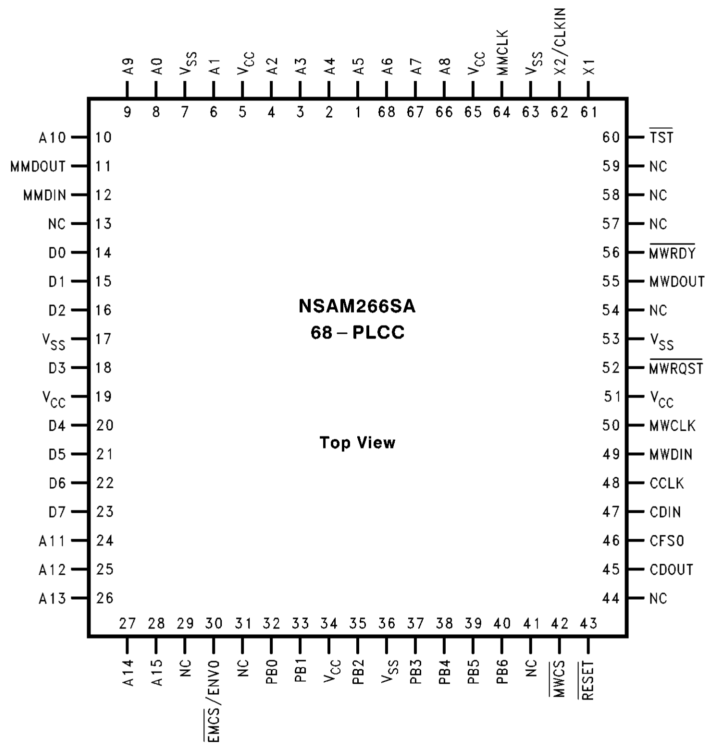
Note C: Chip select lines for Serial Flash devices.

Note D: TTL1 output signals provide CMOS levels in the steady state, for small loads.

Note E: Input during reset, CMOS level input.

1.0 Hardware (Continued)

1.2.2 Pin Assignment in the 68-PLCC Package

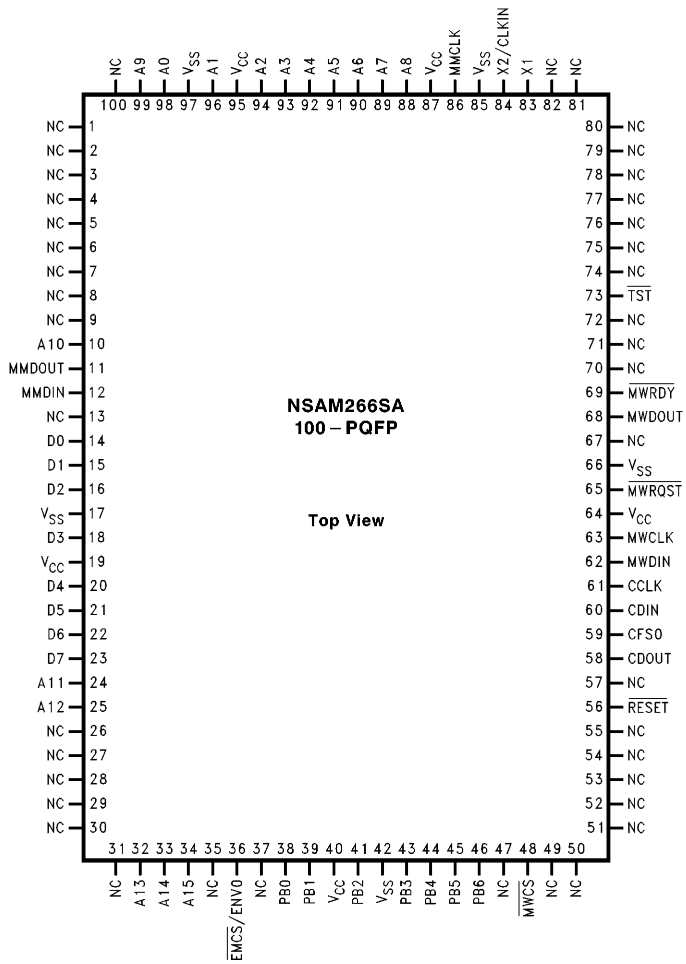


TL/EE/12584–3

Note: Pins marked NC should not be connected.

FIGURE 1-1. 68-PLCC Package Connection Diagram

1.2.3 Pin Assignment in the 100-PQFP Package



TL/EE/12584–4

Note: Pins marked NC should not be connected.

FIGURE 1-2. 100-PQFP Package Connection Diagram

1.0 Hardware (Continued)

1.3 FUNCTIONAL DESCRIPTION

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

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

1.3.1 Resetting

The $\overline{\text{RESET}}$ pin is used to reset the CompactSPEECH 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}. During this period, and for 100 μs after, the $\overline{\text{TST}}$ signal must be high. This can be done with a pull-up resistor on the $\overline{\text{TST}}$ pin.

The value of $\overline{\text{MWRDY}}$ is undefined during the reset period, and for 100 μs 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 signal is sampled to determine the operating environment. During reset, the $\overline{\text{EMCS}}$/ENV0 pin is used for the ENV0 input signals. An internal pull-up resistor sets ENV0 to 1.

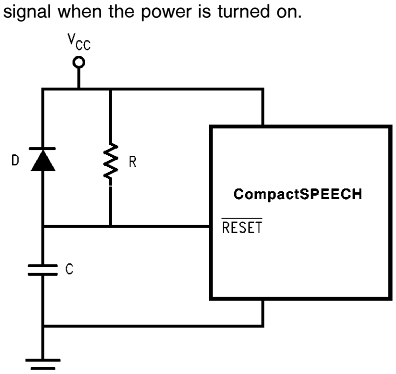
After reset, the same pin is used for $\overline{\text{EMCS}}$.

System Load on ENV0

For any load on the ENV0 pin, the voltage should not drop below V_{ENVh}.

If the load on the ENV0 pin causes the current to exceed 10 μA, use an external pull-up resistor to keep the pin at 1.

Figure 1-3 shows a recommended circuit for generating a reset signal when the power is turned on.



TL/EE/12584–5

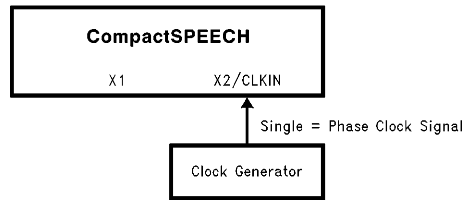
FIGURE 1-3. Recommended Power-On Reset Circuit

1.3.2 Clocking

The CompactSPEECH 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 1-4*, and should conform to the voltage-level requirements for CLKIN stated in Section 1.4.2.

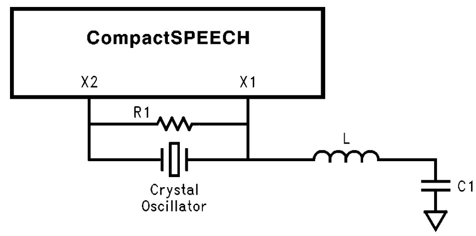


TL/EE/12584–6

FIGURE 1-4. 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 1-5*.



TL/EE/12584–7

FIGURE 1-5. 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 resonators with maximum load capacitance of 20 pF, although the oscillation frequency may differ from the crystal's specified value.

Table 1-2 lists the components in the crystal oscillator circuit.

TABLE 1-2. Crystal Oscillator Component List

Component	Parameters	Values	Tolerance
Crystal Resonator	Resonance Frequency	40.96 MHz	N/A
	Third Overtone	Parallel	
	Type	AT-Cut	
	Maximum Serial Resistance	50Ω	
	Maximum Shunt Capacitance	7 pF	
	Maximum Load Capacitance	12 pF	
Resistor R1		10 MΩ	5%
Capacitor C1		1000 pF	20%
Inductor L		3.9 μH	10%

1.0 Hardware (Continued)

1.3.3 Power-Down Mode

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

In power-down mode, the clock frequency of the CompactSPEECH is reduced, and some of the processor modules are deactivated. As a result, the CompactSPEECH consumes much less power than in normal-power mode ($<$ 1.5 mA). Although the CompactSPEECH does not perform all its usual functions in power-down mode, it still keeps stored messages and maintains the time of day.

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 Serial Flash devices, the power supply to these devices must not be disconnected.

The CompactSPEECH stores messages, and all memory management information, in flash memory. Thus, there is no need to maintain the power to the processor to preserve stored messages. If the microcontroller's real-time clock (and *not* the CompactSPEECH's real-time clock) is used to maintain the time and day, neither the flash nor the CompactSPEECH require battery backup during power failure. In this case, when returning to normal mode, the microcontroller should perform the initialization sequence, as described in Section 2.10, and use the SETD command to set the time and day.

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

The PDM (Go To Power-down Mode) command switches the CompactSPEECH to power-down mode. (For an explanation of the CompactSPEECH commands, see Section 2.14.) It may only be issued when the CompactSPEECH is in the **IDLE** state. (For an explanation of the CompactSPEECH states, see Section 2.3.) If it is necessary to switch to power-down mode from any other state, the controller must first issue an S command to switch the CompactSPEECH to the **IDLE** state, and then issue the PDM command. Sending any command while in power-down mode resets the CompactSPEECH detectors, and returns the CompactSPEECH to normal operation mode.

1.3.4 Power and Grounding

The CompactSPEECH processor requires a single 5V power supply, applied to the V_{CC} pins.

The grounding connections are made on the GND pins.

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.

Use decoupling capacitors to keep the noise level to a minimum. Attach standard 0.1 μF ceramic capacitors to the V_{CC} and GND pins, as close as possible to the CompactSPEECH.

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

1.3.5 Memory Interface

Serial Flash Interface

The CompactSPEECH supports up to four NM29A040 4-Mbit, or up to two NM29A080 8-Mbit, serial flash memory devices for storing messages.

NM29A040

The NM29A040 is organized as 128 blocks of 128 pages, each containing 32 bytes. A block is the smallest unit that can be erased, and is 4 kbytes in size.

Not all 128 blocks are available for recording. Up to 10 blocks may contain bad bits, and one block is write-once and holds the locations of these unusable blocks.

For further information about the NM29A040, see the *NM29A040 Datasheet*.

NM29A080

The NM29A080 is organized as 256 blocks of 128 pages, each containing 32 bytes. A block is the smallest unit that can be erased, and is 4 kbytes in size.

Not all 256 blocks are available for recording. Up to 20 blocks may contain bad bits, and two blocks are write-once and hold the locations of these unusable blocks.

For further information about the NM29A080, see the *NM29A080 Datasheet*.

Message Organization and Recording Time

A CompactSPEECH message uses at least one block. The number of messages that can be stored on one NM29A040 device is 117–127, and on one NM29A080 device is 234 to 254 depending on the number of bad blocks. The maximum recording time depends on four factors:

- — The basic compression rate (5.2 kbit/s or 7.3 kbit/s)
- — The amount of silence in the recorded speech
- — The number of unusable blocks
- — The number of recorded messages. (The basic memory allocation unit for a message is a 4 kbyte block which means that half a block in average is not used per recorded message)

Assuming a single message is recorded in all the available memory space of a 4 Mbit device with all blocks usable, the maximum recording time using 5.2 kbit/s compression is as follows:

TABLE 1-3. Recording Time on 4 Mbit Device

Amount of Silence	Total Record Time
0	13 min 9 sec
10	14 min 25 sec
15	15 min 7 sec
20	15 min 47 sec
25	16 min 25 sec

Serial Flash Endurance

The serial flash may be erased up to 100,000 times. To reduce the effect of this limitation, the memory manager utilizes the serial flash's blocks evenly, i.e., each block is erased more or less the same number of times, to ensure that all blocks have the same lifetime.

Consider the following extensive usage of all the NM29A040's blocks:

1. Record 15 minutes of messages (until the memory is full).
2. Playback 15 minutes (all the recorded messages).
3. Delete all messages.

Assuming a NM29A040 device is used in this manner 24 times a day, its expected lifetime is:

Flash Lifetime = 100,000/(24 * 365) = 11.4 years

Thus the NM29A040 device will last for over ten years, even when used for six hours of recording per day.

Note, that if an NM29A080 device is used, then, under the same conditions, it will last for more than 20 years.

ROM Interface

IVS vocabularies can be stored in either serial flash and/or ROM. The CompactSPEECH supports IVS ROM devices through Expansion Memory. Up to 64 kbytes (64k x 8) of Expansion Memory are supported directly. Nevertheless, the CompactSPEECH uses bits of the on-chip port (PB) to further extend the 64 kbytes address space up to 0.5 Mbytes address space.

ROM is connected to the CompactSPEECH using the data bus, D(0:7), the address bus, A(0:15), the extended address signals, EA(16:18), and Expansion Memory Chip Select, $\overline{\text{EMCS}}$, controls. The number of extended address pins to use may vary, depending on the size and configuration of the ROM.

Reading from Expansion Memory

An Expansion Memory read bus-cycle starts at T1, when the data bus is in TRI-STATE®, and the address is driven on the address bus. $\overline{\text{EMCS}}$ is asserted (cleared to 0) on a T2W1 cycle. This cycle is followed by three T2W cycles and one T2 cycle. The CompactSPEECH samples data at the end of the T2 cycle.

The transaction is terminated at T3, when $\overline{\text{EMCS}}$ becomes inactive (set to 1). The address remains valid until T3 is complete. A T3H cycle is added after the T3 cycle. The address remains valid until the end of T3H.

1.3.6 Codec Interface

The CompactSPEECH provides an on-chip interface to a serial codec. This interface supports codec operation in long- or short-frame formats. The format is selected with the CFG command.

The codec interface uses four signals—CDIN, CDOUT, CCLK and CFS0.

Data is transferred to the codec through the CDOUT pin. Data is read from the codec through the CDIN pin.

Data transfer between the CompactSPEECH and the serial codec starts by the CompactSPEECH asserting (setting to 1) the CFS0 frame synchronization signal. After one clock cycle, the CompactSPEECH de-asserts (clears to 0) CFS0, data from the CompactSPEECH is sent to the codec through CDOUT, and simultaneously data from the codec is sent to the CompactSPEECH through CDIN.

Short Frame Protocol

When short frame protocol is configured, eight data bits are exchanged with each codec in each frame, i.e., CFS0 cycle.

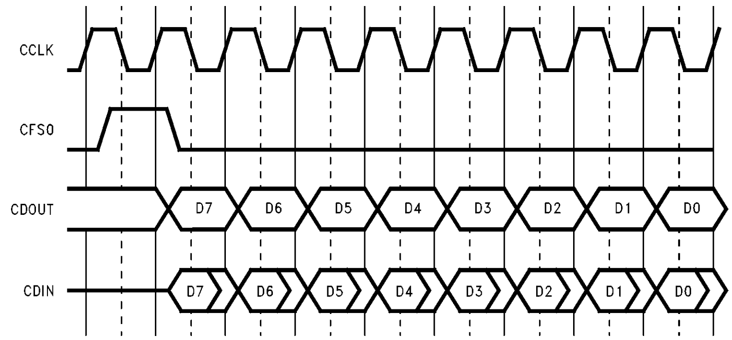
Data transfer starts when CFS0 is set to 1 for one CCLK cycle. The data is then transmitted, bit-by-bit, via the CDOUT output pin. Concurrently, the received data is shifted in via the CDIN input pin. Data is shifted one bit in each CCLK cycle.

Figure 1-6 shows how the codec interface signals behave when short frame protocol is configured.

Long Frame Protocol

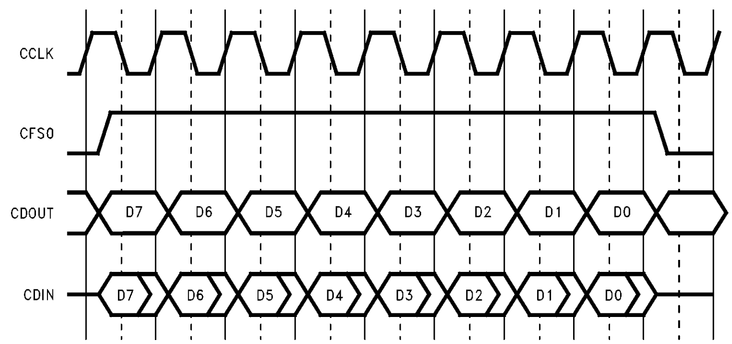
When long frame protocol is configured, eight data bits are exhanged with each codec, as with the short frame protocol. However, for the long frame protocol, data transfer starts by setting CFS0 to 1 for eight CCLK cycles. Simultaneously, the data for the first codec is shifted out bit-by-bit, via the CDOUT output pin, as in short frame protocol. Concurrently, the received data is shifted in through the CDIN input. The data is shifted one bit in each CCLK cycle.

Figure 1-7 shows how the codec interface signals behave when long frame protocol is configured.



TL/EE/12584–8

FIGURE 1-6. Codec Protocol—Short Frame



TL/EE/12584–9

FIGURE 1-7. Codec Protocol—Long Frame

1.0 Hardware (Continued)

1.4 SPECIFICATIONS

1.4.1 Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Storage Temperature	−65°C to +150°C
Temperature under Bias	0°C to +70°C
All Input or Output Voltages, with Respect to GND	−0.5V to +6.5V

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 those conditions specified below.

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

Symbol	Parameter	Conditions	Min	Typ	Max	Units
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_{XH}	CLKIN Input, High Voltage	External Clock	2.0			V
V_{XL}	CLKIN Input, Low Voltage	External Clock			0.8	V
V_{ENVh}	ENV0 High Level, Input Voltage		3.6			V
V_{Hh}	CMOS Input with Hysteresis, Logical 1 Input Voltage		3.6			V
V_{Hl}	CMOS Input with Hysteresis, Logical 0 Input Voltage				1.1	V
V_{Hys}	Hysteresis Loop Width (Note A)		0.5			V
V_{OH}	Logical 1 TTL, Output Voltage	I_{OH} = −0.4 mA	2.4			V
V_{OHWC}	MMCLK, MMDOUT and $\overline{EMCS}$ Logical 1, Output Voltage	I_{OH} = −0.4 mA	2.4			V
		I_{OH} = −50 μA (Note B)	V_{CC} − 0.2			V
V_{OL}	Logical 0, TTL Output Voltage	I_{OL} = 4 mA			0.45	V
		I_{OL} = 50 μA (Note B)			0.2	V
V_{OLWC}	MMCLK, MMDOUT and $\overline{EMCS}$ Logical 0, Output Voltage	I_{OL} = 4.0 mA			0.45	V
		I_{OL} = 50 μA (Note B)			0.2	V
I_L	Input Load Current (Note C)	0V ≤ V_{IN} ≤ V_{CC}	−5.0		5.0	μA
I_O (Off)	Output Leakage Current (I/O Pins in Input Mode) (Note C)	0V ≤ V_{OUT} ≤ V_{CC}	−5.0		5.0	μA
I_{CC1}	Active Supply Current	Normal Operation Mode, Running Speech Applications (Note D)		65	80	mA
I_{CC2}	Standby Supply Current	Normal Operation Mode, DSPM Idle (Note D)		40		mA
I_{CC3}	Power-Down Mode Supply Current	Power-Down Mode (Notes D and E)			1.5	mA
C_X	X1 and X2 Capacitance (Note A)			17		pF

Note A: Guaranteed by design.

Note B: Measured in power-down mode. The total current driven, or sourced, by all the CompactSPEECH's output signals is < 50 μA.

Note C: Maximum 20 μA for all pins together.

Note D: I_{OUT} = 0, T_A = 25°C, V_{CC} = 5V, operating from a 40.96 MHz crystal, and running from internal memory with Expansion Memory disabled.

Note E: All input signals are tied to 1 or 0 (above V_{CC} − 0.5 or below V_{SS} +0.5V).

1.4.3 Switching Characteristics

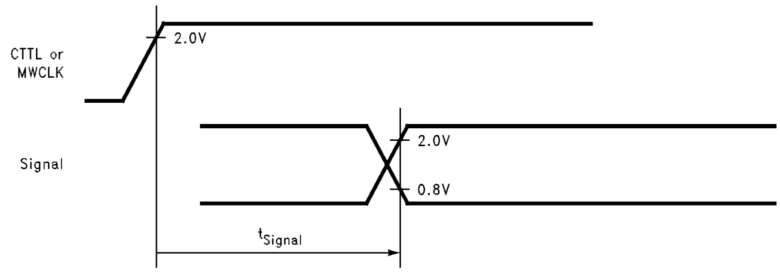
Definitions

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 *Figures 1-8* through *1-14,* unless specifically stated otherwise.

Maximum times assume capacitive loading of 50 pF.

CLKIN crystal frequency is 40.96 MHz.

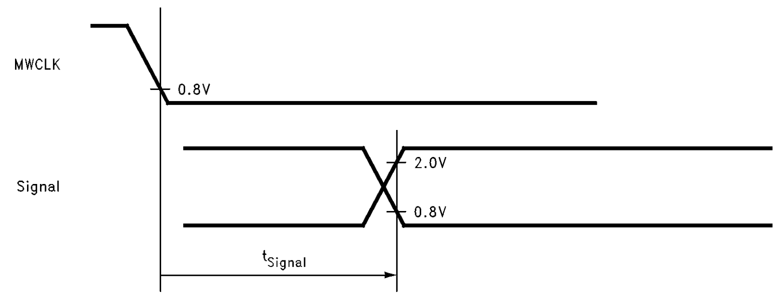
Note: CTTL is an internal signal and is used as a reference to explain the timing of other signals. See *Figure 1-22*.



TL/EE/12584–10

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

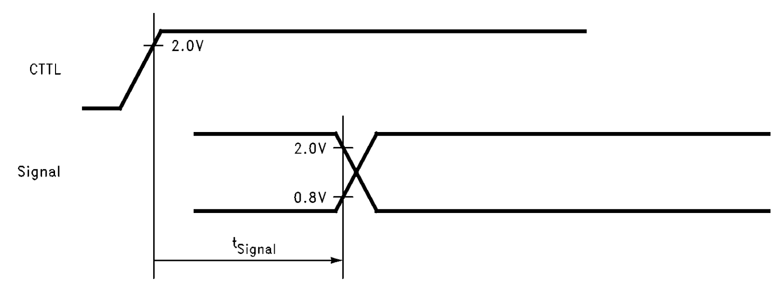
FIGURE 1-8. Synchronous Output Signals (Valid, Active and Inactive)



TL/EE/12584–11

Signal valid time, after a falling edge of MWCLK.

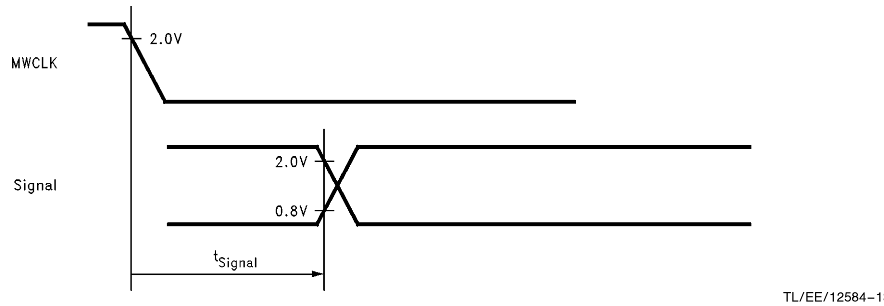
FIGURE 1-9. Synchronous Output Signals (Valid)



TL/EE/12584–12

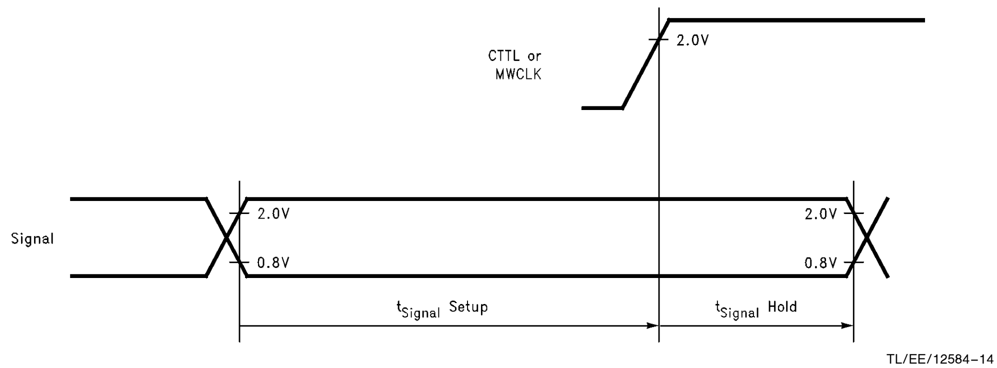
Signal hold time, after a rising edge of CTTL.

FIGURE 1-10. Synchronous Output Signals (Hold)



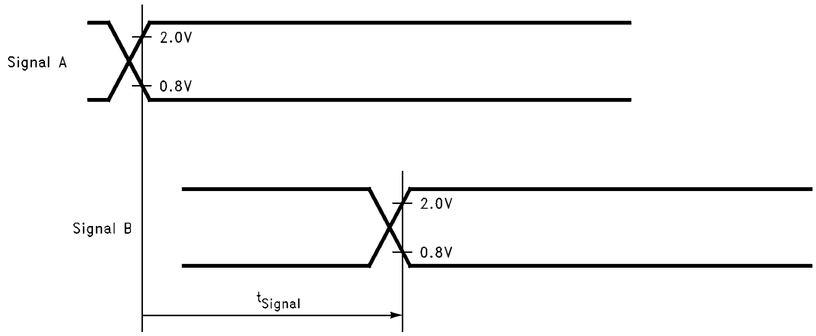
Signal hold time, after a falling edge of MWCLK.

FIGURE 1-11. Synchronous Output Signals (Hold)



Signal setup time, before a rising edge of CTTL or MWCLK, and signal hold time after a rising edge of CTTL or MWCLK.

FIGURE 1-12. Synchronous Input Signals

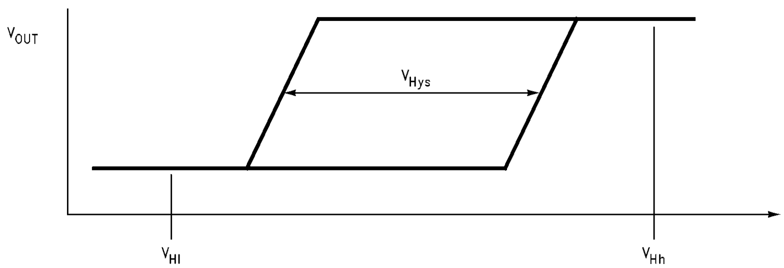


TL/EE/12584–15

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

FIGURE 1-13. Asynchronous Signals

The $\overline{\text{RESET}}$ signal has a Schmitt trigger input buffer. *Figure 1-14* shows the characteristics of the input buffer.



TL/EE/12584–16

FIGURE 1-14. Hysteresis Input Characteristics

1.0 Hardware (Continued)

1.4.4 Synchronous Timing Tables

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

OUTPUT SIGNALS

Symbol	Figure	Description	Reference Conditions	Min (ns)	Max (ns)
t_{Ah}	1-17	Address Hold	After R.E. CTTL	0.0	
t_{Av}	1-17	Address Valid	After R.E. CTTL, T1		12.0
t_{CCLKa}	1-15	CCLK Active	After R.E. CTTL		12.0
t_{CCLKh}	1-15	CCLK Hold	After R.E. CTTL	0.0	
t_{CCLKia}	1-15	CCLK Inactive	After R.E. CTTL		12.0
t_{CDOh}	1-15	CDOUT Hold	After R.E. CTTL	0.0	
t_{CDOv}	1-15	CDOUT Valid	After R.E. CTTL		12.0
t_{CTp}	1-22	CTTL Clock Period (Note A)	R.E. CTTL to next R.E. CTTL	48.8	50,000
t_{EMCSa}	1-17	$\overline{\text{EMCS}}$ Active	After R.E. CTTL, T2W1		12.0
t_{EMCSh}	1-17	$\overline{\text{EMCS}}$ Hold	After R.E. CTTL	0.0	
t_{EMCSia}	1-17	$\overline{\text{EMCS}}$ Inactive	After R.E. CTTL, T3		12.0
t_{FSa}	1-15	CFS0 Active	After R.E. CTTL		25.0
t_{FSh}	1-15	CFS0 Hold	After R.E. CTTL	0.0	
t_{FSia}	1-15	CFS0 Inactive	After R.E. CTTL		25.0
t_{MMCLKa}	1-20	Master MICROWIRE Clock Active	After R.E. CTTL		12.0
t_{MMCLKh}	1-20	Master MICROWIRE Clock Hold	After R.E. CTTL	0.0	
$t_{MMCLKia}$	1-20	Master MICROWIRE Clock Inactive	After R.E. CTTL		12.0
t_{MMDOh}	1-20	Master MICROWIRE Data Out Hold	After R.E. CTTL	0.0	
t_{MMDOv}	1-20	Master MICROWIRE Data Out Valid	After R.E. CTTL		12.0
t_{MWDOf}	1-18	MICROWIRE Data Float (Note B)	After R.E. $\overline{\text{MWCS}}$		70.0
t_{MWDOh}	1-18	MICROWIRE Data Out Hold (Note B)	After F.E. MWCK	0.0	
t_{MWDOnf}	1-18	MICROWIRE Data No Float (Note B)	After F.E. $\overline{\text{MWCS}}$	0.0	70.0
t_{MWDOv}	1-18	MICROWIRE Data Out Valid (Note B)	After F.E. MWCK		70.0
t_{MWITOp}	1-19	MWDIN to MWDOUT	Propagation Time		70.0
t_{MWRDYa}	1-18	$\overline{\text{MWRDY}}$ Active	After R.E. of CTTL	0.0	35.0
$t_{MWRDYia}$	1-18	$\overline{\text{MWRDY}}$ Inactive	After F.E. MWCLK	0.0	70.0
t_{PABCh}	1-21	PB and $\overline{\text{MWRQST}}$	After R.E. CTTL	0.0	
t_{PABCv}	1-21	PB and $\overline{\text{MWRQST}}$	After R.E. CTTL, T2W1		12.0

Note A: In normal operation mode t_{CTp} must be 48.8 ns; in power-down mode, t_{CTp} must be 50,000 ns.

Note B: Guaranteed by design, but not fully tested.

1.0 Hardware (Continued)

INPUT SIGNALS

Symbol	Figure	Description	Reference Conditions	Min (ns)
t_{CDIh}	1-15	CDIN Hold	After R.E. CTTL	0.0
t_{CDIs}	1-15	CDIN Setup	Before R.E. CTTL	11.0
t_{DIh}	1-17	Data in Hold (D0:7)	After R.E. CTTL T1, T3 or TI	0.0
t_{DIs}	1-17	Data in Setup (D0:7)	Before R.E. CTTL T1, T3 or TI	15.0
t_{MMDINh}	1-20	Master MICROWIRE Data In Hold	After R.E. CTTL	0.0
t_{MMDINs}	1-20	Master MICROWIRE Data In Setup	Before R.E. CTTL	11.0
t_{MWCKh}	1-18	MICROWIRE Clock High (Slave)	At 2.0V (Both Edges)	100.0
t_{MWCKl}	1-18	MICROWIRE Clock Low (Slave)	At 0.8V (Both Edges)	100.0
t_{MWCKp}	1-18	MICROWIRE Clock Period (Slave) (Note A)	R.E. MWCLK to next R.E. MWCLK	2.5 μs
t_{MWCLKh}	1-18	MWCLK Hold	After $\overline{\text{MWCS}}$ becomes Inactive	50.0
t_{MWCLKs}	1-18	MWCLK Setup	Before $\overline{\text{MWCS}}$ becomes Active	100.0
t_{MWCSh}	1-18	$\overline{\text{MWCS}}$ Hold	After F.E. MWCLK	50.0
t_{MWCSs}	1-18	$\overline{\text{MWCS}}$ Setup	Before R.E. MWCLK	100.0
t_{MWDIh}	1-18	MWDIN Hold	After R.E. MWCLK	50.0
t_{MWDIs}	1-18	MWDIN Setup	Before R.E. MWCLK	100.0
t_{PWR}	1-24	Power Stable to $\overline{\text{RESET}}$ R.E. (Note B)	After V_{CC} reaches 4.5V	30.0 ms
t_{RSTw}	1-23	$\overline{\text{RESET}}$ Pulse Width	At 0.8V (Both Edges)	10.0 ms
t_{Xh}	1-22	CLKIN High	At 2.0V (Both Edges)	$t_{X1p}/2 - 5$
t_{Xl}	1-22	CLKIN Low	At 0.8V (Both Edges)	$t_{X1p}/2 - 5$
t_{Xp}	1-22	CLKIN Clock Period	R.E. CLKIN to next R.E. CLKIN	24.4

Note A: Guaranteed by design, but not fully tested in power-down mode.

Note B: Guaranteed by design, but not fully tested.

1.4.5 Timing Diagrams

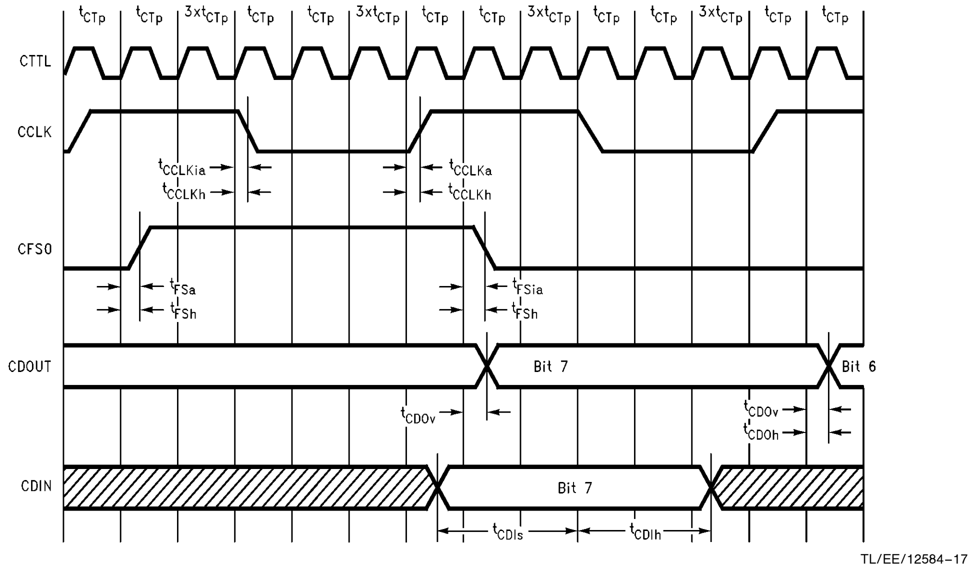


FIGURE 1-15. Codec Short Frame Timing

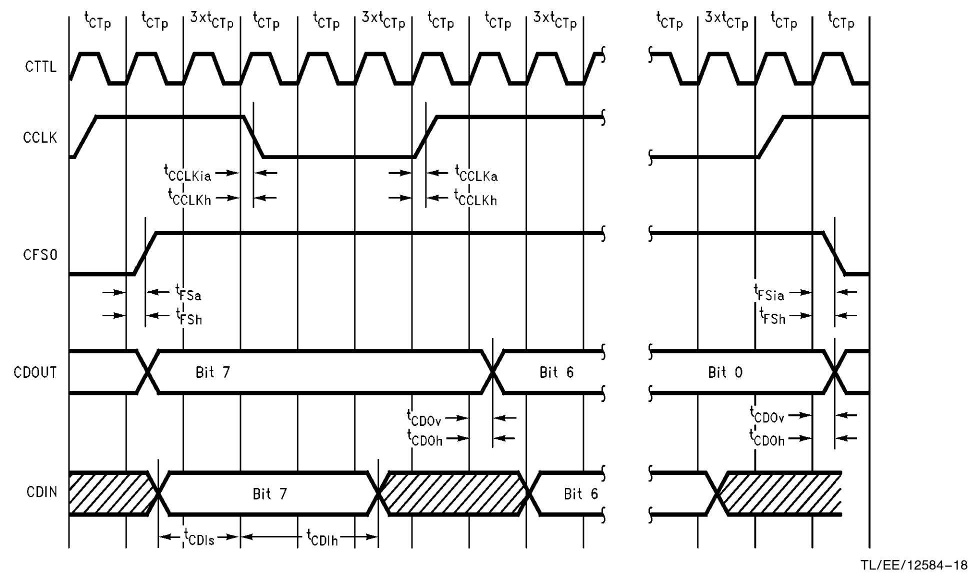
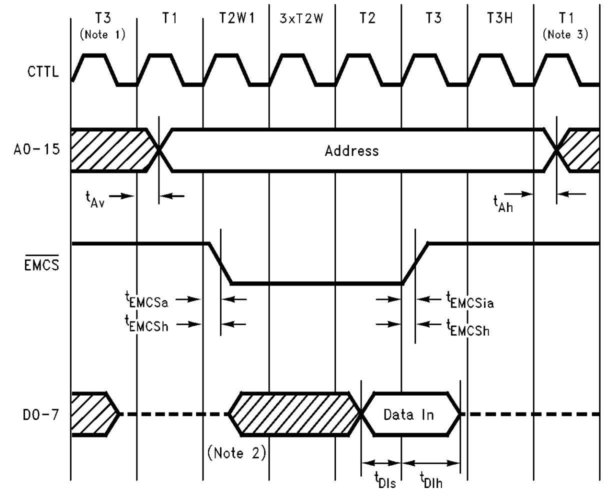


FIGURE 1-16. Codec Long Frame Timing



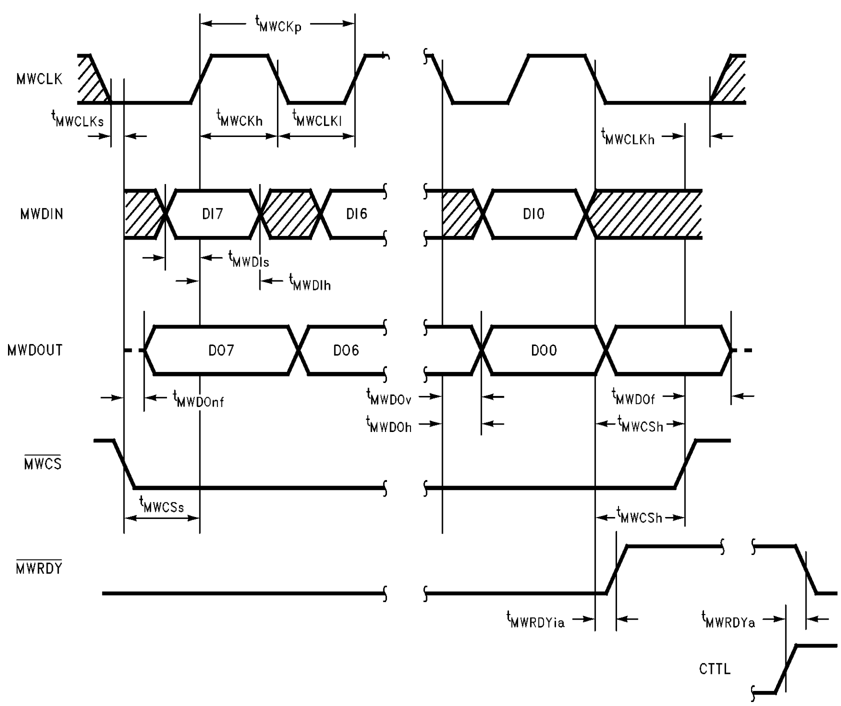
TL/EE/12584-19

Note 1: This cycle may be either TI (Idle), T3 or T3H.

Note 2: Data can be driven by an external device at T2W1, T2W, T2 and T3.

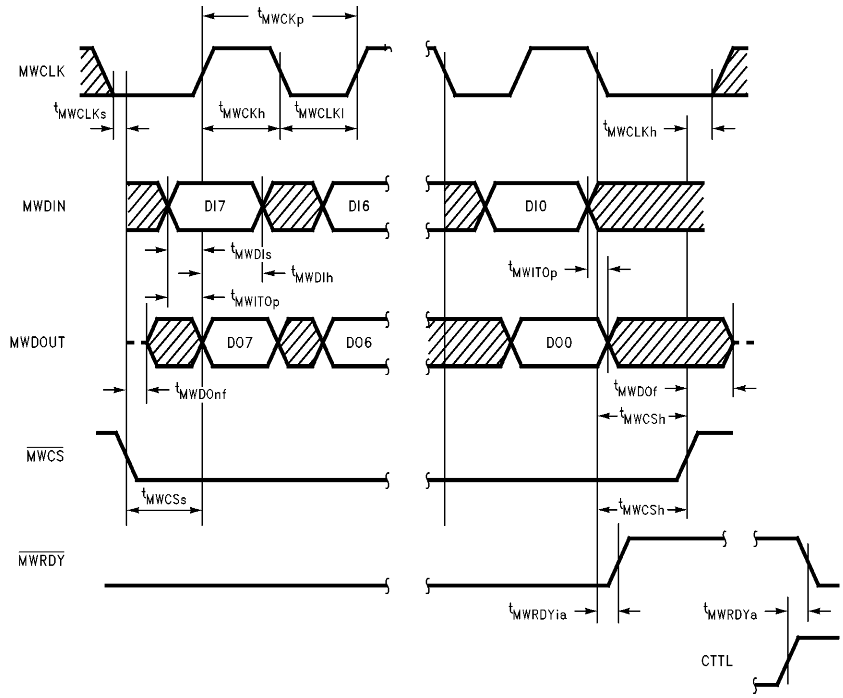
Note 3: This cycle may be either TI (Idle) or T1.

FIGURE 1-17. ROM Read Cycle Timing



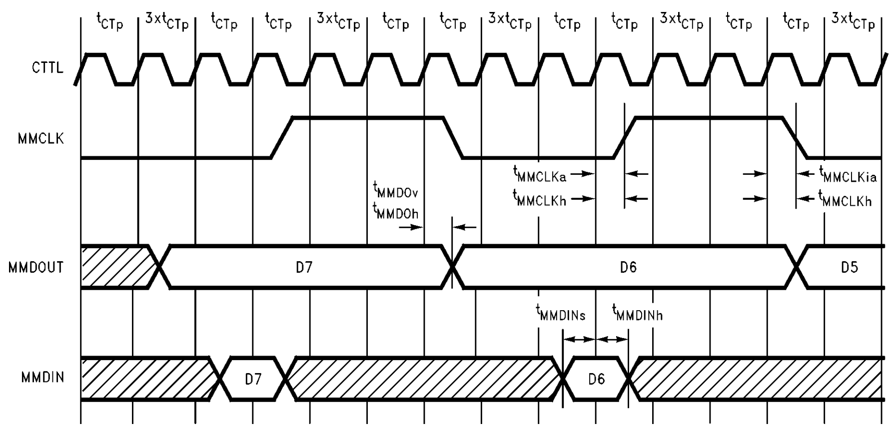
TL/EE/12584-20

FIGURE 1-18. MICROWIRE Transaction Timing—Data Transmitted to Output



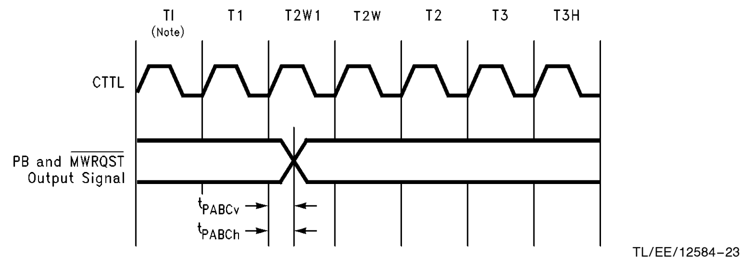
TL/EE/12584–21

FIGURE 1-19. MICROWIRE Transaction Timing—Data Echoed to Output



TL/EE/12584–22

FIGURE 1-20. Master MICROWIRE Timing



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

FIGURE 1-21. Output Signal Timing for Port PB and $\overline{\text{MWRQST}}$

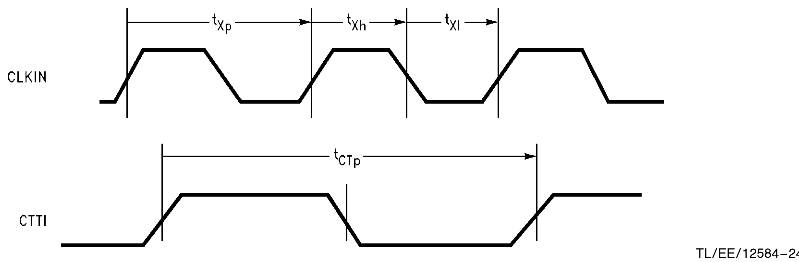


FIGURE 1-22. CTTL and CLKIN Timing

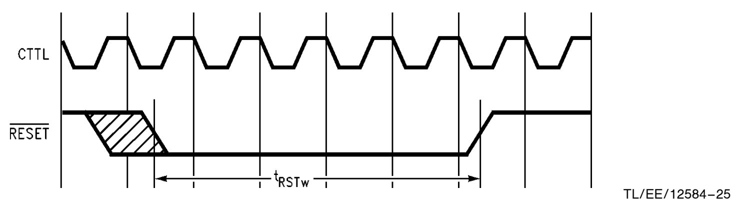


FIGURE 1-23. Reset Timing When Reset is not at Power-Up

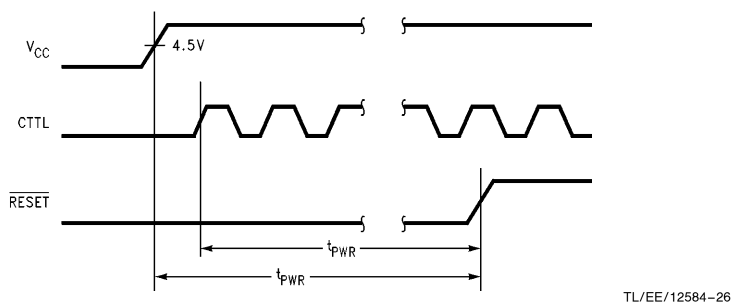


FIGURE 1-24. Reset Timing When Reset is at Power-Up

2.0 Software

2.1 OVERVIEW

The CompactSPEECH software resides in the on-chip ROM. It includes DSP-based algorithms, system support functions and a software interface to hardware peripherals.

2.1.1 DSP-based Algorithms

- Speech compression and decompression
- DTMF detector with echo canceler
- Energy-based busy and dial-tone detector
- Digital volume control

2.1.2 System Support

- Command interface to an external microcontroller
- Memory and message manager
- IVS support
- Tone generator
- Real-time clock handler
- Power-down mode support

2.1.3 Peripherals Support

- Serial flash interface (Master MICROWIRE handler)
- Microcontroller interface (Slave MICROWIRE handler)
- Codec interface

The following sections describe the CompactSPEECH software in detail.

2.0 Software (Continued)

2.2 CompactSPEECH COMMANDS—QUICK REFERENCE TABLE

Command		Description	Opcode Hex	Source State	Result State	Command Parameters		Return Value	
Name	S/A					Description	Bytes	Description	Bytes
AMAP**	S	Check and Map ARAM	06			Action__number	1	Test Result	1
CCIO	S	Configure Codec I/O	34	RESET, IDLE	No Change	Config—value	1	None	
CFG	S	Configure CompactSPEECH	01	RESET	RESET	Config__value	2	None	
CMT	S	Cut Message Tail	26	IDLE	IDLE	Length of Time	2	None	
CVOC	S	Check Vocabulary	2B	IDLE	IDLE	None		Test Result	1
DM	S	Delete Message	0A	IDLE	IDLE	None		None	
DMS	S	Delete Messages	0B	IDLE	IDLE	Tag__ref, Tag__mask	2+2	None	
FR**	A	Free Memory	08			None		None	
GCFG	S	Get Configuration Value	02	RESET, IDLE	No Change	None		Version, Config__value	2
GEW	S	Get Error Word	1B	All States	No Change	None		Error Word	2
GI	S	Get Information Item	25	PLAY, RECORD, SYNTHESIS, TONE__GENERATE, IDLE	No Change	Index	1	Value	2
GL	S	Get Length	19	IDLE	IDLE	None		Message Length	2
GMS	S	Get Memory Status	12	IDLE	IDLE	Type	1	Recording Time Left	2
GMT	S	Get Message Tag	04	IDLE	IDLE	None		Message Tag	2
GNM	S	Get Number of Messages	11	IDLE	IDLE	Tag__ref, Tag__mask	2+2	Number of Messages	2
GSW	S	Get Status Word	14	All States	No Change	None		Status Word	2
GT	A	Generate Tone	0D	IDLE	TONE__GENERATE	Tone or DTMF	1	None	
GTD	S	Get Time and Day	0E	IDLE	IDLE	Time/Day Option	1	Time/Day	2
GTM	S	Get Tagged Message	09	IDLE	IDLE	Tag__ref, Tag__Mask, Dir	2+2+1	Message Found	1
INIT	S	Initialize System	13	RESET, IDLE	IDLE	None		None	
INJ	S	Inject IVS Data	29	RESET, IDLE	No Change	N, byte1 . . . byten	4+n	None	
MR	S	Memory Reset	2A	IDLE	IDLE	None		None	
P	A	Playback	03	IDLE	PLAY	None		None	
PA	S	Pause	1C	PLAY, RECORD, SYNTHESIS, TONE__GENERATE, IDLE*	No Change	None		None	
PDM	S	Go to Power-Down Mode	1A	IDLE	IDLE	None		None	

2.0 Software (Continued)

2.2 CompactSPEECH COMMANDS—QUICK REFERENCE TABLE (Continued)

Command		Description	Opcode Hex	Source State	Result State	Command Parameters		Return Value	
Name	S/A					Description	Bytes	Description	Bytes
R	A	Record Message	0C	IDLE	RECORD	Message Tag	2	None	
RDET	S	Reset Detectors	2C	IDLE	No Change	Detectors Reset Mask	1	None	
RES	S	Resume	1D	PLAY, RECORD, SYNTHESIS, TONE__GENERATE, IDLE*	No Change	None		None	
RRAM	S	Read RAM	18	IDLE, MEMORY__READ	MEMORY__READ	None		Data	32
S	S	Stop	00	All States but RESET	IDLE	None		None	
SAS	A	Say Argumented Sentence	1E	IDLE	SYNTHESIS	Sentence__n arg1	1+1	None	
SB	S	Skip Backward	23	PLAY, IDLE*	No Change	Length of Time	2	None	
SDET	S	Set Detectors Mask	10	IDLE	No Change	Detectors Mask	1	None	
SE	S	Skip to End of Message	24	PLAY, IDLE*	No Change	None		None	
SETD	S	Set Time and Day	0F	IDLE	No Change	Time/Day	2	None	
SF	S	Skip Forward	22	PLAY, IDLE*	No Change	Length of Time	2	None	
SMT	S	Set Message Tag	05	IDLE	IDLE	Message Tag	2	None	
SO	A	Say One Word	07	IDLE	SYNTHESIS	Word Number	1	None	
SPS	S	Set Playback Speed	16	PLAY, SYNTHESIS, IDLE	No Change	Speed Value	1	None	
SS	A	Say Sentence	1F	IDLE	SYNTHESIS	Sentence__n	1	None	
SV	S	Set Vocabulary Type	20	IDLE	IDLE	Mode, Id	1+1	None	
SW	A	Say Words	21	IDLE	SYNTHESIS	N, word1 . . . wordn	1+n	None	
TUNE	S	Tune	15	IDLE	IDLE	Index, Value	1+2	None	
VC	S	Volume Control	28	PLAY, SYNTHESIS, IDLE, TONE__GENERATE	No Change	Increment/Decrement	1	None	
WRAM	S	Write RAM	17	IDLE, MEMORY__WRITE	MEMORY__WRITE	Message Tag, Data	2+32	None	

*Command is valid in **IDLE** state; but has no effect.

**This command exists for compatibility reasons only, and will be obsoleted in future revisions of CompactSPEECH.

S Synchronous command.

A Asynchronous command.

2.3 THE STATE MACHINE

The CompactSPEECH functions as a state machine. It changes state either in response to a command sent by the microcontroller, after execution of the command is completed, or as a result of an internal event (e.g., memory full or power failure).

The CompactSPEECH may be in one of the following states:

RESET

The CompactSPEECH is initialized to this state after a full hardware reset by the $\overline{\text{RESET}}$ signal (see Section 1-6). CompactSPEECH detectors (VOX, call progress tones and DTMF tones) are not active. In all other states, the detectors are active. (See the SDET and RDET commands for further details).

IDLE

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

PLAY

In this state a message is decompressed, and played back.

RECORD

In this state a message is compressed, and recorded into the message memory.

SYNTHESIS

An individual word or a sentence is synthesized from an external vocabulary.

TONE_GENERATE

The CompactSPEECH generates single or DTMF tones.

MEMORY_READ

The CompactSPEECH reads a 32-byte block from the message memory and sends it to the external microcontroller.

MEMORY_WRITE

The CompactSPEECH accepts a 32-byte block from the external microcontroller and writes it to the message memory.

After receiving an asynchronous command, (see Section 2.4) such as P (Playback), R (Record), SW (Say Words) or GT (Generate Tone), the CompactSPEECH switches to the appropriate state and executes the command until it is completed, or an S (Stop) or PA (Pause) command is received from the microcontroller.

When an asynchronous command execution is completed, the EV_NORMAL_END event is set, and the CompactSPEECH switches to the **IDLE** state.

Section 2.2 provides a table which shows all the CompactSPEECH commands, the source states in which these commands are valid, and the result states which the CompactSPEECH enters as a result of the command.

2.4 COMMAND EXECUTION

A CompactSPEECH command is represented by an 8-bit opcode. Some commands have parameters, and some have return values. Commands are either synchronous or asynchronous.

SYNCHRONOUS COMMANDS

A synchronous command must complete execution before the microcontroller can send a new command (e.g., GMS, GEW).

A command sequence starts when the microcontroller sends an 8-bit opcode to the CompactSPEECH, followed by the command's parameters (if any).

The CompactSPEECH executes the command and, if required, transmits a return value to the microcontroller. Upon completion, the CompactSPEECH notifies the microcontroller that it is ready to accept a new command.

ASYNCHRONOUS COMMANDS

An asynchronous command starts execution in the background and notifies the microcontroller, which can send more commands while the current command is still running (e.g., R, P).

STATUS WORD

The 16-bit status word indicates events that occur during normal operation. The CompactSPEECH activates the $\overline{\text{MWRQST}}$ signal, to indicate a change in the status word. This signal remains active until the CompactSPEECH receives a GSW command.

ERROR WORD

The 16-bit error word 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 $\overline{\text{MWRQST}}$ signal is activated.

ERROR HANDLING

When the microcontroller detects that the $\overline{\text{MWRQST}}$ signal is active, it should issue the GSW (Get Status Word) command, which deactivates the $\overline{\text{MWRQST}}$ signal. Then, it should test the EV_ERROR bit in the status word, and, if it is set, send the GEW (Get Error Word) command to read the error word for details of the error.

For a detailed description of each of the CompactSPEECH commands, see Section 2.14.

2.5 TUNABLE PARAMETERS

The CompactSPEECH processor can be adjusted to your system's requirements. For this purpose the CompactSPEECH supports a set of tunable parameters, which are set to their default values after reset and can be later modified with the TUNE command. By tuning these parameters, you can control various aspects of the CompactSPEECH's operation, such as silence compression, tone detection, no-energy detection, etc.

Table 2-2 describes all the tunable parameters in detail. Section 2.14 describes the TUNE command.

2.6 MESSAGES

The CompactSPEECH message manager supports a wide range of applications, which require different levels of DAM functionality.

The message-organization scheme, and the message tag, support advanced memory-organization features such as multiple OutGoing Messages (OGMs), mailboxes, and the ability to distinguish between InComing Messages (ICMs) and OGMs.

A message is the basic unit on which most of the CompactSPEECH commands operate. A CompactSPEECH message, stored on a flash device, can be regarded as a computer file stored on a mass-storage device.

A message is created with either the R or the WRAM (Write Memory) command.

When a message is created, it is assigned a time-and-day stamp and a message tag which can be read by the microcontroller.

The R command takes voice samples from the codec, compresses them, and stores them in the message memory.

When a message is created with the WRAM command the data to be recorded is provided by the microcontroller and not via the codec. The data is transferred directly to the message memory. It is not compressed by the CompactSPEECH voice compression algorithm.

The WRAM command, together with the RRAM (Read Memory) command which enables the microcontroller to read data from the CompactSPEECH, can be used to store data other than compressed voice in the message memory e.g., a telephone directory.

A message can be played back (P command) and deleted (DM command). Redundant data (e.g., trailing tones or silence) can be removed from the message tail with the CMT (Cut Message Tail) command.

The PA (Pause) and RES (Resume) commands, respectively, temporarily suspend the P and R commands, and then allow them to resume execution from where they were suspended.

CURRENT MESSAGE

Most message handling commands, e.g., P, DM, RRAM, operate on the current message. The GTM (Get Tagged Message) command selects the current message.

Deleting the current message does not cause a different message to become current. The current message is undefined. If, however, you issue the GTM command to skip to the next message, the first message that is newer than the just deleted message is selected as the current message.

2.6.1 Message Tag

Each message has a 2-byte message tag which you can use to categorize messages, and implement such features as OutGoing Messages, mailboxes, and different handling of old and new messages.

The most significant bit (bit 15) of the message tag is used to indicate the speech compression rate. The microcontroller should program it before recording (“1” for 4.8 kbit/s, “0” for 6.6 kbit/s). The CompactSPEECH reads the bit before message playback to select the appropriate decompression algorithm.

The GMT (Get Message Tag) and SMT commands may be used to handle message tags.

Note: Message tag bits can only be cleared. Message tag bits are set only when a message is first created.

This limitation is inherent in flash memories, which only allow bits to be changed from 1 to 0 (changing bits from 0 to 1 requires a special erasure procedure, see Section 1.3.5). However, the main reason for updating an existing tag is to mark a message as old, and this can be done by using one of the bits as a new/old indicator, setting it to 1 when a message is first created, and clearing it when necessary.

2.7 SPEECH COMPRESSION

The CompactSPEECH implements two speech compression algorithms. One algorithm, with 5.2 kbit/s compression rate, enables up to 14–16 minutes of recording on a 4-Mbit device, while the other uses a 7.3 kbit/s compression rate to support 10–12 minutes of recording. Both compression rates assume 10% silence.

Before recording each message, the microcontroller can select one of the two algorithms by programming bit 15 of the message tag. During message playback the CompactSPEECH reads this bit and selects the appropriate speech decompression algorithm.

IVS vocabularies can be prepared in either of the two compression formats using the IVS tool. All the messages in a single vocabulary must be recorded using the same algorithm. (See the *IVS User's Manual* for further details). During speech synthesis, the CompactSPEECH automatically selects the appropriate speech decompression algorithm.

2.8 TONE AND NO-ENERGY DETECTION

The CompactSPEECH detects DTMF, busy, and dial tones, and no-energy (VOX). This enables remote control operations and call progress. Detection is active throughout the operation of the CompactSPEECH. Detection can be configured using the SDET (Set Detectors Mask) command, which controls the reporting of the occurrence of tones, and the RDET (Reset Detectors) command which resets the detectors.

DTMF

DTMF detection may be reported at the starting point, ending point, or both. The report is made through the status word (for further details, see GSW command in Section 2.14).

The DTMF detector performance, as measured on the line input using an NSV-AM265-DAA board, is summarized below (see Table 2-1).

ECHO CANCELLATION

Echo cancellation is a technique used to improve the performance of DTMF tone detection during speech synthesis, tone generation, and OGM playback. For echo cancellation to work properly, AGC must not be active in parallel. Thus, to take advantage of echo cancellation, the microcontroller must control the AGC, i.e., disable the AGC during **PLAY, SYNTHESIS** and **TONE_GENERATE** states and enable it again afterwards. If AGC cannot be disabled, do not use echo cancellation. The microcontroller should use the CFG command to activate/deactivate echo cancellation. (For further details, see Section 2.14.)

Echo cancellation applies only to DTMF tones. Busy and dial-tone detection is not affected by this technique. This implies that the performance of the busy and dial-tone detector during message playback depends on the message being played.

TABLE 2-1. DTMF Detector Performance

	PLAY	RECORD/IDLE
Detection Sensitivity (Note A)	Performance Depends on Message Being Played (Note B)	−40 dBm
Accepted DTMF Length	> 50 ms	> 40 ms
Frequency Tolerance	±1.5%	±1.5%
S/N Ratio	12 dB	12 dB
Minimum Spacing (Note C)	> 50 ms	> 45 ms
Normal Twist	8 dB	8 dB
Reverse Twist (Note D)	4 dB or 8 dB	4 dB or 8 dB

Note A: Performance depends on the DAA design.

Note B: Performance with echo canceler is 10 dB better than without echo canceler. For a silent message, Detection Sensivitiy is −34 dBm with echo canceler.

Note C: If the interval between two consecutive DTMF tones is ≤ 20 ms, the two are detected as one long DTMF tone.
If the interval between two consecutive DTMF tones is between 20 ms and 45 ms, separate detection is unpredictable.

Note D: Determined by the DTMF__REV__TWIST tunable parameter value.

OTHER DETECTORS

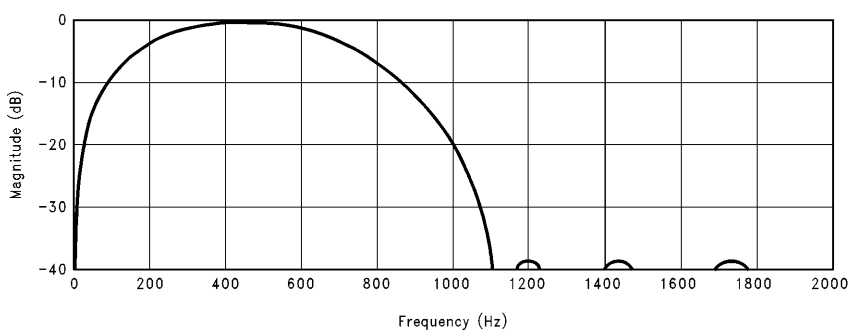
Detection of busy and dial tones, and no-energy is controlled by tunable parameters. You should tune these parameters to fit your hardware. For more information see the TUNE command in Section 2.14.

Dial and busy tone detectors work with a band-pass filter that llmits the frequency range in which tones can be detected to 0 Hz–1100 Hz. Its frequency response is illustrated in *Figure 2-1* and the busy tone cadences in *Figure 2-2*.

TONE GENERATION

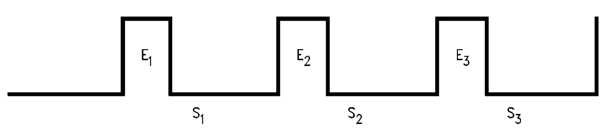
The CompactSPEECH can generate DTMF tones and single-frequency tones from 300 Hz to 3000 Hz in increments of 100 Hz. CompactSPEECH tone generation conforms to the EIA-470-RS standard. Note, however, that you may have to change the value of some tunable parameters in order to meet the standard specifications since the energy level of generated tones depends on the analog circuits being used.

- Tune the DTMF__TWIST__LEVEL parameter to control the twist level of the generated DTMF tones.
- Use the VC command, and tune the TONE__GENERATION__LEVEL parameter, to control the energy level at which these tones are generated.
- Use the GT command to specify the DTMF tones, and the frequency at which single tones are generated.



TL/EE/12584–27

FIGURE 2-1. Busy and Dial-Tone Band-Pass Filter Frequency Response



TL/EE/12584–28

$[E_1 - E_3] < 90$ ms $[S_1 - S_3] < 90$ ms $90 < E_i < 1650$ ms $65 < S_i < 1250$ ms

FIGURE 2-2. Busy-Tone Detector—Cadence Specification

2.9 SPEECH SYNTHESIS

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

There are two kinds of predefined messages: fixed messages (e.g., voice menus in a voice-mail system) and programmable messages (e.g., time and day stamp, or the *You have n messages* announcement in a DAM).

A vocabulary includes a set of predefined words and phrases, needed to synthesize messages in any language. Applications which support more than one language require a separate vocabulary for each language.

2.9.1 International Vocabulary Support (IVS)

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

Among IVS features:

- Multiple vocabularies are stored on a single storage device.
- Plug-and-play. The same microcontroller 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 CompactSPEECH's clock).
- Support for various language and sentence structures:
  - — One versus many (for example: *You have one* ***message*** vs. You have two **messages**).
  - — None versus many (for example: *You have* ***no*** *messages* vs. You have **two** messages).
  - — Number synthesis (English–*Eighty* vs. French–*Quatre-vingt*).
  - — Word order (English–*Twenty one* vs. German–*Einundzwanzig*).
  - — Days of the week (Monday through Sunday vs. Sunday through Saturday).

2.9.2 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 messages* and *You have twenty two messages*. 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. On one hand, the words should be recorded and saved in a compressed form, and you would like to use the best voice compression for that purpose. On the other hand, the higher the compression rate, the worse the voice quality.

Another issue to consider is the difference in voice quality between synthesized and recorded messages (e.g., between time-and-day stamp and incoming messages (ICMs) in a DAM environment). It is more pleasant to the human ear to hear them both in the same 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 only the vocabulary, but also the code to access it should be compact.

When designing for a multi-lingual environment, there are 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 PM*, you should not care in what language it is going to be played back.

2.9.3 IVS Vocabulary Components

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

The basic concepts An IVS vocabulary consists of words, sentences, and special codes that control the behavior of the algorithm which CompactSPEECH uses to synthesize sentences.

The word table The words are the basic units in the vocabulary. You 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 that you wish to synthesize, you may need to record some words more than once in the vocabulary. For example, if you synthesize the sentences: *you have twenty messages* and *you have twenty five messages, the word twenty* is pronounced differently. They should, therefore, be defined as two different words.

The 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 *message number one* or as *first* as in *first message*.

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

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.

The sentence table The sentence table describes the predefined sentences in the vocabulary. The purpose of this table is to make the microcontroller that drives the CompactSPEECH independent of the language being synthesized.

For example, if the serial flash and/or ROM contain vocabularies in various languages, and the first sentence in each vocabulary means *you have n messages*, the microcontroller switches languages by issuing the following command to CompactSPEECH:

SV <*storage_media*>, <*vocabulary_id*>

Select a new vocabulary

The microcontroller 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.

Sentence 0 All sentences but one are user defined. The CompactSPEECH treats the first sentence in the sentence table, i.e., sentence 0, in a special way to support time & day announcement. It assumes that the sentence is designed for system and message time & day announcement and has one argument which is interpreted as follows:

0 - System time will be announced

1 - The time & day of the current message will be announced.

Example 1: When the microcontroller sends the command:

SAS O, *0*

The system time & day is announced.

Example 2: When the microcontroller sends the command:

SAS O, *1*

The current message time & day stamp is announced.

Figure 2-3 shows the interrelationship between the three types of tables:

Control and option codes The list of word indices alone cannot provide the entire range of sentences that the CompactSPEECH can synthesize. IVS control and option codes are used as special instructions that control the behavior of the speech synthesis algorithm in the CompactSPEECH.

For example, if the sentence should announce the time of day, the CompactSPEECH 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 CompactSPEECH to take special actions.

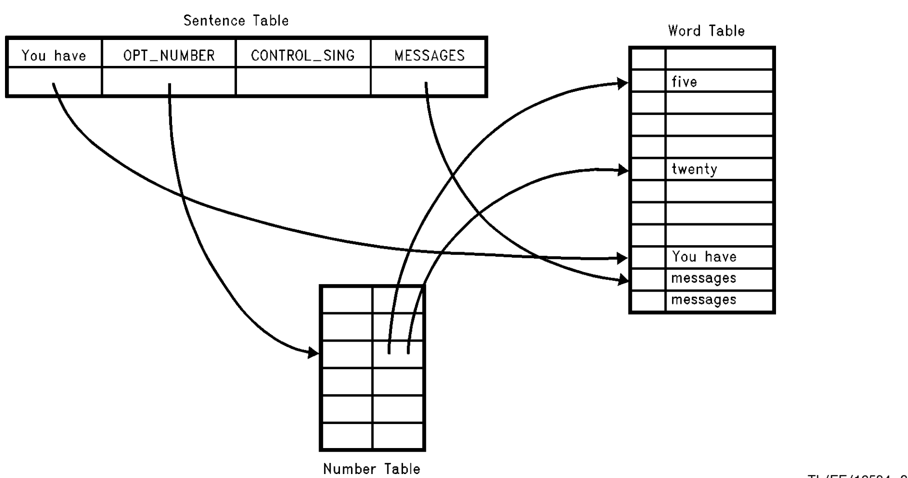


FIGURE 2-3. Relationship of IVS Tables

2.0 Software (Continued)

2.9.4 The IVS Tool

The IVS tool includes two utilities:

— The DOS-based IVS Compiler

— IVSTOOL for Windows. A Windows 3.1 based utility.

The tools allow you to create vocabularies for the CompactSPEECH processor. They take you all the way from designing the vocabulary structure, through defining the vocabulary sentences, and recording the vocabulary words.

The IVS Compiler The IVS compiler runs on MS-DOS (version 5.0 or later). It allows 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. This information can be burned into an EPROM or serial flash for use by the CompactSPEECH software.

Voice Compression Each IVS vocabulary can be compiled using either 5.2 kbit/s or 7.3 kbit/s voice compression algorithm. The user defines the compression rate before compilation. The CompactSPEECH automatically selects the required voice decompression algorithm when the SV command is used to select the active vocabulary.

The Graphical User Interface (GUI) The IVS package includes a Windows utility that assists the vocabulary designer to synthesize sentences. With this utility, you can both compose sentences and listen to them.

2.9.5 How to Use the IVS Tool With the CompactSPEECH

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 serial flash device using the INJ command. The CompactSPEECH SV command is used to select the required vocabulary. The SW, SO, SS and SAS commands are used to synthesize the required word or sentence. The typical vocabulary-creation process is as follows:

1. Design the vocabulary.
2. Create the vocabulary files. Use IVSTOOL for Windows 3.1 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.
6. Burn the IVS vocabulary files into a ROM (or serial flash) device. Use the INJ (Inject IVS) command to program the data into a serial flash device.
7. Once the vocabulary is in place, the speech synthesis commands of the CompactSPEECH can be used to synthesize sentences.

Figure 2-4 shows the vocabulary-creation process for a single table on a ROM or serial flash device.

2.10 INITIALIZATION

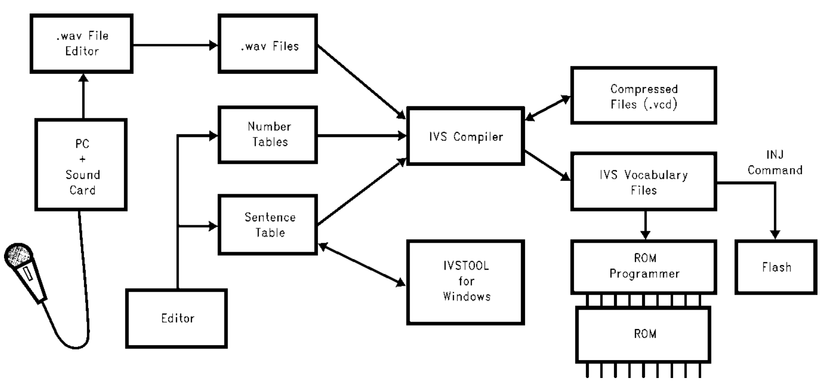
Use the following procedures to initialize the CompactSPEECH processor:

NORMAL INITIALIZATION

1. Reset the CompactSPEECH by activating the $\overline{\text{RESET}}$ signal. (See Section 1.3.1.)
2. Issue a CFG (Configure CompactSPEECH) command to change the configuration according to your environment.
3. Issue an INIT (Initialize System) command to initialize the CompactSPEECH firmware.
4. Issue a series of TUNE commands to adjust the CompactSPEECH to the requirements of your system.

2.11 MICROWIRE SERIAL INTERFACE

MICROWIRE/PLUS™ is a synchronous serial communication protocol, originally implemented in National Semiconductor's COPS™ microcontrollers and HPC™ families of microcontrollers to minimize the number of connections, and thus the cost, of communicating with peripherals.



TL/EE/12584–38

FIGURE 2-4. Creation of an IVS Vocabulary

The CompactSPEECH MICROWIRE interface implements the MICROWIRE/PLUS interface in slave mode, with an additional ready signal. It enables a microcontroller to interface efficiently with the CompactSPEECH application.

The microcontroller is the protocol master, and provides the clock for the protocol. The CompactSPEECH supports clock rates of up to 400 kHz. This transfer rate refers to the bit transfer; the actual throughput is slower due to byte processing by the CompactSPEECH and the microcontroller.

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

If a parameter of a CompactSPEECH command is bigger than one byte, the microcontroller should transmit the Most Significant Byte (MSB) first. If a return value is bigger than one byte, the CompactSPEECH transmits the MSB first.

2.12 SIGNAL DESCRIPTION

The following signals are used for the interface protocol. Input and output are relative to the CompactSPEECH.

INPUT SIGNALS

MWDIN

MICROWIRE Data In. Used for input only, for transferring data from the microcontroller to the CompactSPEECH.

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 rising edge. The transmitted data is output on MWDOUT on the clock 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 CompactSPEECH is being accessed. Setting $\overline{\text{MWCS}}$ to 1 causes the CompactSPEECH 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, so the signal can be used to synchronize between the CompactSPEECH and the microcontroller.

To prevent false detection of access to the CompactSPEECH due to spikes on the MWCLK signal, use this chip select signal, and toggle the MWCLK input signal only when the CompactSPEECH is accessed.

OUTPUT SIGNALS

MWDOUT

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

$\overline{\text{MWRDY}}$

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

This signal is set to 1 by the CompactSPEECH after each byte transfer has been completed. It remains 1, while the CompactSPEECH 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 CompactSPEECH receives a GSW (Get Status Word) command from the microcontroller. After reset, this signal is 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 CompactSPEECH firmware.

2.12.1 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 CompactSPEECH reads data from MWDIN on every rising edge of MWCLK. CompactSPEECH 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 CompactSPEECH transmits to the microcontroller (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 microcontroller 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 CompactSPEECH) after 8-bits of data were transferred to/from the CompactSPEECH. 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 CompactSPEECH when it is ready to receive the first parameter byte (if there are any parameters) and so on till the last byte of parameters is transferred. An active 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 microcontroller must read the value before issuing a new command.

4. When a return value is transmitted, the MWRDY signal is deactivated after every byte, and activated again when the CompactSPEECH is ready to send another byte, or to receive a new command.

   The $\overline{MWRDY}$ signal is activated (cleared to 0) after reset, and after a protocol time-out. (See Section 2.12.2.)

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

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

Figure 2-5 illustrates the sequence of activities during a MICROWIRE data transfer.

2.12.2 Interface Protocol Error Handling

Interface Protocol Time-Outs

Depending on the CompactSPEECH's state, if more than 20 ms–30 ms elapse between two consecutive byte transmissions, or two byte receptions, within the same command or return value, after the $\overline{MWRDY}$ signal is asserted, a time-out event occurs, and the CompactSPEECH 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 $\overline{MWRQST}$ signal (clears it to 0).
4. Activates the $\overline{MWRDY}$ signal (clears it to 0).
5. Waits for a new command. (After a time-out occurs, the microcontroller must wait at least four milliseconds before issuing the next command.)

Echo Mechanism

The CompactSPEECH echoes back to the microcontroller all the bits received by the CompactSPEECH. Upon detection of an error in the echo, the microcontroller 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 CompactSPEECH transmits bytes of the return value instead of the echo value.

The CompactSPEECH transmits a byte as an echo when it receives the value 0xAA from the microprocessor. Upon detection of an error the CompactSPEECH activates the $\overline{MWRQST}$ signal, and sets the ERR_COMM bit in the error word.

2.13 THE MASTER MICROWIRE INTERFACE

The CompactSPEECH's Master MICROWIRE controller implements the MICROWIRE/PLUS interface in master mode. It enables the CompactSPEECH to control flash devices. Several devices may share the Master MICROWIRE channel. This can be implemented by connecting device selection signals to general purpose output ports.

2.13.1 Master MICROWIRE Data Transfer

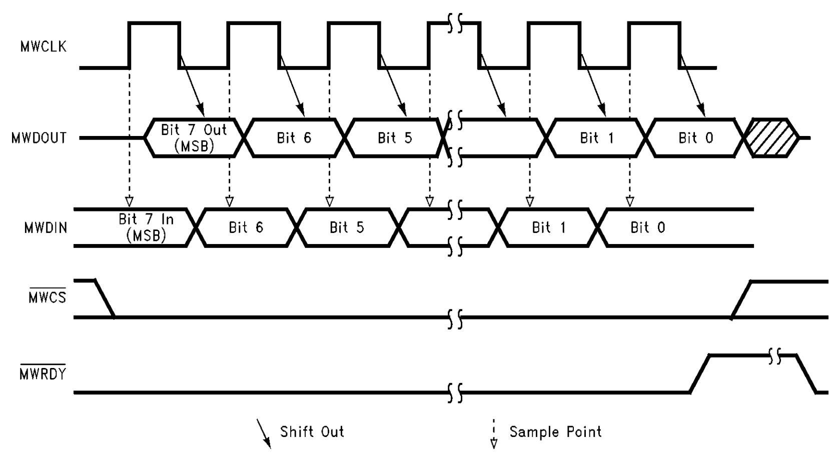
The Signals

The Master MICROWIRE controller's signals are the Master MICROWIRE serial CLocK (MMCLK), the Master MICROWIRE serial Data OUT (MMDOUT) signal and the Master MICROWIRE serial Data In (MMDIN) signal.

The Master MICROWIRE controller can handle up to four flash devices. The CompactSPEECH uses the signals, CS0–CS3, as required for the number of devices in use, as device chip-select signals.

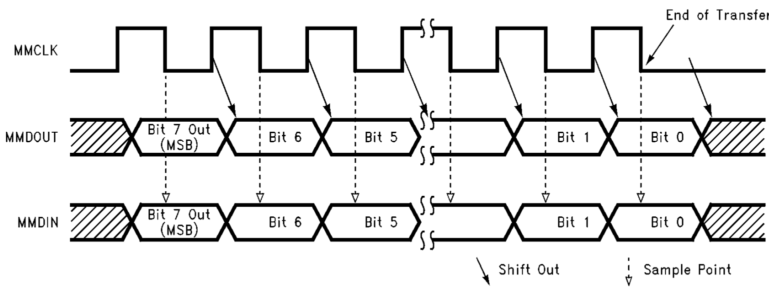
The Clock for Master MICROWIRE Data Transfer

Before data can be transferred, the transfer rate must be determined and set. The rate of data transfer on the Master MICROWIRE is determined by the Master MICROWIRE serial CLocK (MMCLK) signal. This rate is the same as the Codec CLocK (CCLK) signal. As long as the Master MICROWIRE is transferring data, the codec interface must be enabled and its sampling rate should not be changed.



TL/EE/12584–29

FIGURE 2-5. Sequence of Activities during a MICROWIRE Byte Transfer



TL/EE/12584–30

FIGURE 2-6. Master MICROWIRE Data Transfer

2.14 COMMAND DESCRIPTION

The commands are listed in alphabetical order.

The execution time for all commands, when specified, includes the time required for the microcontroller to retrieve the return value, where appropriate.

The execution time does not include the protocol timing overhead, i.e., the execution times are measured from the moment that the command is detected as valid until the command is fully executed.

Note: Each command description includes an example application of the command. The examples show the opcode issued by the microcontroller, and the response returned by the CompactSPEECH. For commands which require a return value from the CompactSPEECH, the start of the return value is indicated by a thick vertical line.

CCIO **Configure Codec I/O *config-value***

Configures the voice samples paths in various states. It should be used to change the default CompactSPEECH configuration. The *config-value* parameter is encoded as follows:

Bit 0 Loopback control.

0: Loopback disable (default)

1: Loopback enabled. During RECORD state, the input samples are echoed back unchanged (i.e., no volume control) to the codec.

Bits 1–7 Reserved

Example

CCIO 3401			
Byte sequence:	Microcontroller	34	01
	CompactSPEECH	34	01
Description:	Enable Loopback.		

CFG **Configure CompactSPEECH *config_value***

Configures the CompactSPEECH in various hardware environments. It should be used to change the default CompactSPEECH configuration.

The *config_value* parameter is encoded as follows:

Bit 0 Codec configuration.

0: short-frame format (default)

1: long-frame format. (Guaranteed by design, but not tested.)

Bit 1 Reserved.

Bit 2 Echo cancellation control.

0: Echo cancellation off (default)

1: Echo cancellation is on during playback.

Echo cancellation improves the performance of DTMF detection during playback. Echo cancellation can be turned on only with a system that can disable AGC during playback. A system with AGC that cannot be controlled (i.e., enabled/disabled) by the microcontroller must not turn on this bit.

Bit 3 Reserved—must be cleared to 0.

Bits 4–5 Reserved—must be set to 10.

Bits 6–7 Reserved—must be cleared to 00.

Bits 8–10 Number of installed flash devices.

Valid range 1 .. 4 flash devices.

Default is 1.

Bits 11–15 Reserved—Must be cleared to 0.

Note: The CompactSPEECH automatically detects the type of flash device in use, i.e., NM29A040 or NM29A080.

Example

CFG 0324				
Byte sequence:	Microcontroller	01	03	24
	CompactSPEECH	01	03	24
Description:	Configure the CompactSPEECH to work with: Codec that supports short-frame format. Three, NM29A040, flash devices. Echo cancellation on.			

CMT **Cut Message Tail *time_length***

Cut *time_length* units, each of 10 ms duration, off the end of the current message. The maximum value of *time_length* is 6550. Cut-time accuracy is ±0.14 sec.

Note: If *time_length* is longer than the total duration of the message, the EV_NORMAL_END event is set in the status word, and the message becomes empty (i.e., message length is 0), but is not deleted. Use the DM (Delete Message), or DMS (Delete Messages), command to delete the message.

Example

CMT 02BC				
Byte sequence:	Microcontroller	26	02	BC
	CompactSPEECH	26	02	BC
Description:	Cut the last seven seconds of the current message.			

CVOC **Check Vocabulary**

Checks (checksum) if the IVS data 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.

Example

CVOC			
Byte sequence:	Microcontroller	2B	AA
	CompactSPEECH	2B	01
Description:	Check the current vocabulary. The CompactSPEECH responds that the vocabulary is OK.		

DM **Delete Message**

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 is undefined. If, for example, you issue the GTM command to skip to the next message, the first message that is newer than the just deleted message is selected as the current message.

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

Example

DM		
Byte sequence:	Microcontroller	0A
	CompactSPEECH	0A
Description:	Delete current message.	

DMS **Delete Messages *tag_ref tag_mask***

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_task* = *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.

Example

DMS FFC2 003F						
Byte sequence:	Microcontroller	0B	FF	C2	00	3F
	CompactSPEECH	0B	FF	C2	00	3F
Description:	Delete all old incoming messages from mailbox Number 2 in a system where the message tag is encoded as follows: Bits 0–2: mailbox ID 8 mailboxes indexed: 0 to 7 Bit 3: new/old message indicator 0: Message is old 1: Message is new Bits 4–5: message type 00: ICM/memo 01: OGM 10: Call transfer message Bits 6–15: not used **Note:** The description of the tag is an example only. All bits of the tag are user-definable.					

GCFG **Get Configuration Value**

Returns a sequence of two bytes with the following information:

Bits 0–7 Magic number, which specifies the CompactSPEECH firmware version.

Bits 8–9 Memory type.

00: Reserved

01: Reserved

10: Serial Flash

11: Reserved

The command should be used together with the CFG and INIT commands during CompactSPEECH initialization. See the CFG command for more details, and an example of a typical initialization sequence.

Example

GCFG				
Byte sequence:	Microcontroller	02	AA	AA
	CompactSPEECH	02	02	01
Description:	Get the CompactSPEECH magic number. The CompactSPEECH responds that it is Version 1, with Serial Flash.			

GEW **Get Error Word**

Returns the 2-byte error word.

THE ERROR WORD

The 16-bit error word 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 $\overline{MWRQST}$ signal is activated (set to low).

The GEW command reads the error word. The error word is cleared during reset and after execution of the 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 microcontroller must still read the return value, to ensure proper synchronization.

15–9	8	7	6	5	4	3	2	1	0
Res	**Res**	**ERR__INVALID**	**ERR__TIMEOUT**	**ERR__COMM**	**Res**	**ERR__PARAM**	**ERR__COMMAND**	**ERR__OPCODE**	**Res**

The bits of the error word are used as follows:

ERR__OPCODE

Illegal opcode. The command opcode is not recognized by the CompactSPEECH.

ERR__COMMAND

Illegal command sequence. The command is not legal in the current state.

ERR__PARAM

Illegal parameter. The value of the parameter is out of range, or is not appropriate for the command.

ERR__COMM

Microcontroller MICROWIRE communication error.

ERR__TIMEOUT

Time-out error. Depending on the CompactSPEECH's state, more than 20 ms to 30 ms elapsed between the arrival of two consecutive bytes (for commands that have parameters).

ERR__INVALID

Command cannot be performed in current context.

Example

GEW				
Byte sequence:	Microcontroller	1B	AA	AA
	CompactSPEECH	1B	00	02
Description:	Get the CompactSPEECH error word (typically sent after GSW when EV__ERROR is reported in the status word). The CompactSPEECH responds: **ERR__OPCODE:**			

GI **Get Information *item***

Returns the 16-bit value specified by *item* from one of the internal registers of the CompactSPEECH.

item may be one of the following:

0: The duration of the last detected DTMF tone, in 10 ms units. The return value is meaningful only if DTMF detection is enabled, and the status word shows that a DTMF tone was detected.

1: The duration of the last detected busy tone in 10 ms units.

2: The energy level of the samples in the last 10 ms.

3: The energy level of the samples, in the last 10 ms, that are in the frequency range described in *Figure 2-1.* The return value is meaningful only if one of the tone detectors is enabled (bits 0,1 of the detectors mask; see the description of SDET command).

The return value is unpredictable for any other value of *item*.

Example

GI 0					
Byte sequence:	Microcontroller	25	00	AA	AA
	CompactSPEECH	25	00	00	06
Description:	Get the duration of the last detected DTMF tone. The CompactSPEECH responds: **60 ms**				

GL **Get Length**

Returns the length of the current message in multiples of 32 bytes.

The returned value includes the message directory information (64 bytes for the first block and 32 bytes for every other block), message data, and the entire last block of the message, even if the message occupies only a portion of the last block. Since a flash block includes 4096 bytes, the returned length may be bigger than the actual message length by up to 4095 bytes.

The minimum length of a message is one block, i.e., an empty message occupies 4 kbytes (the message length is: 4096/32 = 128).

Example

GL				
Byte sequence:	Microcontroller	19	AA	AA
	CompactSPEECH	19	02	00
Description:	Get the length of the current message. The CompactSPEECH responds: **512** i.e., the message occupies 16384 (512 * 32) bytes			

GMS **Get Memory Status *type***

Returns the estimated total remaining recording time in seconds as a 16-bit unsigned integer. This estimate assumes 5.2 kbit/s with no silence compression: a real recording may be longer, according to the amount of silence detected and compressed.

The return value is dependent on the value of the *type* parameter as follows:

0: The remaining recording time is returned.

1: Returns 0. (For compatibility only.)

2: Same as 0. (For compatibility only.)

The return value is unpredictable for any other value of *type*.

Example

GMS 0					
Byte sequence:	Microcontroller	12	00	AA	AA
	CompactSPEECH	12	00	01	40
Description:	Return the remaining recording time. The CompactSPEECH responds: **320 seconds**				

GMT **Get Message Tag**

Returns the 16-bit tag associated with the current message. If the current message is undefined, ERR_VALID is reported.

Example

GMT				
Byte sequence:	Microcontroller	04	AA	AA
	CompactSPEECH	04	00	0E
Description:	Get the current message tag. In a system where the message tag is encoded as described in the DMS command, the CompactSPEECH return value indicates that the message is a new ICM in mailbox Number 6.			

GNM **Get Number of Messages *tag_ref tag_mask***

Returns the number of 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.

The *tag_ref* and *tag_mask* parameters are each two bytes; the return value is also 2-bytes long.

For example, if *tag_ref* = 42_{16}, and *tag_mask* = $3F_{16}$, the number of existing old messages whose user-defined tag is 2 is returned. See Section 2.6.1 for a description of message-tag encoding. If *tag_mask* = 0, the total number of all existing messages is returned, regardless of the *tag_ref* value.

Example

GNM FFFE 0003								
Byte sequence:	Microcontroller	11	FF	FE	00	03	AA	AA
	CompactSPEECH	11	FF	FE	00	03	00	05
Description:	Get the number of messages which have bit 0 cleared, and bit 1 set, in their message tags. CompactSPEECH responds that there are five messages which satisfy the request.							

GSW **Get Status Word**

Returns the 2-byte status word.

THE STATUS WORD

The CompactSPEECH processor has a 16-bit status word to indicate events that occur during normal operation. The CompactSPEECH asserts the $\overline{\text{MWRQST}}$ signal (clears to 0), to indicate a change in the status word. This signal remains active until the CompactSPEECH receives a GSW command.

The status word is cleared during reset, and by the GSW command.

15	14	13	1211	10	9	8	7	6	5	4	30
EV__DTMF	EV__RESET	EV__VOX	Res	Res	EV__DIALTONE	EV__BUSY	EV__ERROR	EV__MEMFULL	EV__NORMAL__END	EV__DTMF__END	EV__DTMF__DIGIT

The bits in the status word are used as follows:

EV__DTMF__DIGIT

DTMF digit. A value indicating a detected DTMF digit. (See the description of DTMF code in the GT command.)

EV__DTMF__END

1 = Ended detection of a DTMF tone. The detected digit is held in EV__DTMF__DIGIT.

EV__NORMAL__END

1 = Normal completion of operation, e.g., end of message playback.

EV__MEMFULL

1 = Memory is full.

EV__ERROR

1 = Error detected in the last command. You must issue the GEW command to return the error code and clear the error condition.

EV__BUSY

1 = Busy tone detected. Use this indicator for call progress and line disconnection.

EV__DIALTONE

1 = Dial tone detected. Use this indicator for call progress and line disconnection.

EV__VOX

1 = a period of silence (no energy) was detected on the telephone line during recording. (See VOX__TIME__COUNT in Table 2-2.)

EV__RESET

When the CompactSPEECH completes its power-up—sequence and enters the **RESET** state, this bit is set to 1, and the $\overline{\text{MWRQST}}$ signal is activated (cleared to 0).

Normally, this bit changes to 0 after performing the INIT command. If this bit is set during normal operation of the CompactSPEECH, it indicates an internal CompactSPEECH error. The microcontroller can recover from such an error by re-initializing the system.

EV__DTMF

1 = Started detection of a DTMF tone.

Example

GSW				
Byte sequence:	Microcontroller	14	AA	AA
	CompactSPEECH	14	00	40
Description:	Get the CompactSPEECH Status Word (typically sent after the $\overline{\text{MMRQST}}$ signal is asserted by the CompactSPEECH which indicates a change in the status word). The CompactSPEECH responds that the memory is full.			

GT **Generate Tone *tone***

Generates the tone specified by the 1-byte *tone* parameter, until an S command is received.

Specify the tone by setting the bits of *tone* as follows:

Bit 0 1

Bits 1–4 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

Bits 5–7 0

To generate a single-frequency tone encode the bits as follows:

Bit 0 0

Bits 1–5 3–30

The value in bits 1–5 is multiplied by 100 to generate the required frequency (300 Hz–3000 Hz).

Bits 6, 7 0

The CompactSPEECH does not check for the validity of the tone specification. Invalid specification yields unpredictable results.

Example

GT 0D20			
Byte sequence:	Microcontroller	0D	20
	CompactSPEECH	0D	20
Description:	Generate a single-frequency 1600 Hz tone.		

GTD **Get Time and Day *time__day__option***

Returns the time and day as a 2-byte value. *time__day__option* may be one of the following:

0: Get the system time and day.

1: Get the current message time-and-day stamp.

Any other *time__day__option* returns the time-and-day stamp of the current message.

Time of day is encoded as follows:

Bits 0–2 Day of the week (1 through 7).

Bits 3–7 Hour of the day (0 through 23).

Bits 8–13 Minute of the hour (0 through 59).

Bits 14–15 00: The time was not set before the current message was recorded.

11: The time was set, i.e., the SETD (Set Time of Day) command was executed.

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

Example

GTD 1					
Byte sequence:	Microcontroller	0E	01	AA	AA
	CompactSPEECH	0E	01	E8	29
Description:	Get the current message time-and-day stamp. The CompactSPEECH responds that the message was created on the first day of the week at 5:40 AM. The return value also indicates that the SETD command was used to set the system time and day before the message was recorded. **Note:** If the SAS command is used to announce the time-and-day stamp, ''Monday'' is announced as the first day of the week. For an external vocabulary, the announcement depends on the vocabulary definition. (See the *IVS User's Manual* for more details.)				

GTM **Get Tagged Message *tag__ref tag__mask dir***

Selects the current message, according to instructions in *dir,* to be the first, n[th] next or n[th] 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.

128: Selects the last (newest) message.

n: Selects the n[th] next message starting from the current message.

-n: Selects the n[th] previous message starting from the current message.

Note: 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.

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 description for further detail.)

To access the nth message, when n > 127, a sequence of GTM commands is required.

2.0 Software (Continued)

Example

GTM FFCE 003F 0									
Byte sequence:	Microcontroller	09	FF	CE	00	3F	00	AA
	CompactSPEECH	09	FF	CE	00	3F	00	01
Description:	Select the oldest of the new ICMs, in mailbox number 6, to be the current message. For a system where the message tag is encoded as described in the example for the DMS command. The CompactSPEECH return value indicates that there is such a message. The following pseudo-code demonstrates how to play all new ICMs. The messages are marked after being played. In mailbox number 6:							

```
Return_val = GTM (FFCE, 003F, 1)
While (ReturnVal == TRUE)
Begin
P() /* Play */
Message_tag = GMT() /* get message tag */
SMT(FFF7) /* Mark the message as "old' */
GTM(FFCE, 003F, 1) /* get next with same tag */
End
```

INIT **Initialize System**

Execute this command after the CompactSPEECH has been configured (see CFG and GCFG commands).

Performs a soft reset of the CompactSPEECH as follows:

- Initializes the message directory information. Messages are not deleted. To delete the messages, use the DM and DMS commands.
- Sets the detectors mask to 0.
- Sets the volume level that is controlled by the VC command, to 0.
- Sets the playback speed to normal (0).
- Switches to the **IDLE** state.
- Activates (clears to 0) the $\overline{\text{MWRDY}}$ signal.
- Initializes the tone detectors.

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.

Example

INIT		
Byte sequence:	Microcontroller	13
	CompactSPEECH	13
Description:	Initialize the CompactSPEECH.	

INJ **Inject IVS data *n byte*$_1$ . . . *byte*$_n$**

Injects vocabulary data of size *n* bytes to good flash blocks.

This command programs flash devices, on a production line, with IVS vocabulary data. It is optimized for speed; all CompactSPEECH detectors are suspended during execution of the command. Use the CVOC command to check whether programming was successful.

If there is not enough memory space for the vocabulary data, ERR_PARAM is set in the error word, and execution stops.

Flash blocks that include IVS data cannot be used for recording, even if only one byte of the block contains IVS data (e.g., if the vocabulary size is 4k + 100 bytes, two blocks of the flash are not available for message recording).

Example

INJ 128 *Data*							
Byte sequence:	Microcontroller	29	00	00	00	80	Vocabulary Data
	CompactSPEECH	29	00	00	00	80	Echo of Data
Description:	Inject 128 bytes of vocabulary data.						

MR **Memory Reset**

Erases all good flash blocks and initializes the CompactSPEECH (i.e., does exactly what the INIT command does). Bad blocks, and blocks which are used for IVS vocabularies, are not erased.

Note: The command erases all messages and should be used with care.

Example

MR		
Byte sequence:	Microcontroller	2A
	CompactSPEECH	2A
Description:	Erase all Serial Flash blocks.	

P **Playback**

Begins playback of the current message. The CompactSPEECH state changes to **PLAY.** When playback is complete, the CompactSPEECH sets the EV_NORMAL_END bit in the status word, and activates (clears to 0) the $\overline{\text{MWRQST}}$ signal. Playback can be paused with the PA command, and can be resumed later with the RES command.

If the current message is undefined, ERR_INVALID is reported.

Example

P		
Byte sequence:	Microcontroller	03
	CompactSPEECH	03
Description:	Play the current message.	

PA **Pause**

Suspends the execution of the current R, P, GT, SO, SW, SS or SAS command. The PA command does not change the state of the CompactSPEECH; execution can be resumed with the RES command.

Note: DTMF and tone detectors remain active during Pause.

Example

PA		
Byte sequence:	Microcontroller	1C
	CompactSPEECH	1C
Description:	Suspend playback of current message.	

PDM **Go To Power-down Mode**

Switches the CompactSPEECH to power-down mode (see Section 1.3.3 for details). Sending any command while in power-down mode resets the CompactSPEECH detectors, and returns the CompactSPEECH to normal operation mode.

Example

PDM		
Byte sequence:	Microcontroller	1A
	CompactSPEECH	1A
Description:	Put the CompactSPEECH in power-down mode.	

R **Record *tag***

Records a new message with message tag *tag.* The CompactSPEECH state changes to **RECORD**. The R command continues execution until stopped by the S command. Recording can be paused with the PA command, and can be resumed later with the RES command.

If the memory becomes full, recording stops and EV_MEMFULL is set in the status word.

Note: A time-and-day stamp is automatically attached to each message. Before using the R command for the first time, use the SETD command. Failure to do so results in undefined values for the time-and-day stamp.

Example

R 000E				
Byte sequence:	Microcontroller	0C	00	0E
	CompactSPEECH	0C	00	0E
Description:	Record a new ICM in mailbox Number 6 in a system where the message tag is encoded as described in the example of the DMS command.			

RDET **Reset Detectors *detectors_reset_mask***

Resets the CompactSPEECH tone and energy detectors according to the value of the *detectors_reset_mask* parameter. A bit set to 1 in the mask, resets the corresponding detector. A bit cleared to 0 is ignored.

The 1-byte *detectors_reset_mask* is encoded as follows:

Bit 0 Reset the busy and dial tone detectors.

Bits 1–4 Reserved. Must be cleared to 0.

Bit 5 Reset the no energy (VOX) detector.

Bit 6 Reset the DTMF detector.

Bit 7 Reserved. Must be cleared to 0.

Example

RDET 20			
Byte sequence:	Microcontroller	2C	20
	CompactSPEECH	2C	20
Description:	Reset the VOX detector.		

RES **Resume**

Resumes the activity that was suspended by the PA, SF or SB commands.

Example

RES		
Byte sequence:	Microcontroller	1D
	CompactSPEECH	1D
Description:	Resume playback which was suspended by either the PA, SF or SB command.	

RRAM **Read Memory**

Returns 32 bytes from the current message. The first RRAM command returns the first 32 bytes of the current message. Subsequent RRAM commands return the next following 32 bytes from the message until the end of the message. The command sequence can be stopped by the S command.

Note 1: Trying to read beyond the end of the message sets the EV_NORMAL_END event, and the CompactSPEECH switches to the IDLE state. In this case, the return value is undefined and should be ignored.

Note 2: When using WRAM and RRAM to write and read messages of arbitrary length, the microcontroller is responsible for marking the actual end of the message (e.g., with a delimiter string).

The next RRAM command after the end of the message is reached, starts again from the beginning of the current message.

Note 3: If the current message is undefined, ERR_INVALID is reported.

Example

RMEM *Data*					
Byte sequence:	Microcontroller	18	AA	AA	. . .
	CompactSPEECH	18	32 bytes of data		
Description:	Read 32 bytes from the current message memory.				

S **Stop**

Stops execution of the current command and switches the CompactSpeech to the **IDLE** state. S may be used to stop the execution of WRAM, RRAM and all asynchronous commands.

Example

S				
Byte sequence:	Microcontroller	00		
	CompactSPEECH	00		
Description:	Stop current activity (e.g., playback, recording) and put the CompactSPEECH in **IDLE** state.			

SAS **Say Argumented Sentence** ***sentence_n arg***

Announces sentence number *sentence_n* of the currently selected vocabulary, and passes *arg* to it. *sentence_n* and *arg* are each 1-byte long.

When playing is complete, the CompactSPEECH sets the EV_NORMAL_END bit in the status word, and activates the $\overline{\text{MWRQST}}$ signal.

If the current vocabulary is undetermined, ERR_INVALID is reported.

Example

SAS 00 03				
Byte sequence:	Microcontroller	1E	00	03
	CompactSPEECH	1E	00	03
Description:	Announce the first sentence in the sentence table of the currently selected vocabulary with "3" as the actual parameter.			

SB **Skip Backward** ***time_length***

Skips backward in the current message *time_length* units, each of 0.2s duration, and causes message playback to pause. *time_length* is a 2-byte parameter that can have any value up to 320, i.e., 64s. The skip accuracy is 5%. This command is meaningful only in the **PLAY** state. The RES command must be issued to continue playback.

If the beginning of the message is detected, during execution of the SB command, execution is terminated, the EV_NORMAL_END bit in the status register is set, the $\overline{\text{MWRQST}}$ signal is activated, and the CompactSPEECH switches to the **IDLE** state.

If *time_length* is greater than 320, ERR_PARAM is set in the error word.

Playback speed does not affect the behavior of this command.

Example

SB 19				
Byte sequence:	Microcontroller	23	00	19
	CompactSPEECH	23	00	19
Description:	Skip back five seconds from the current position in the message being played.			

SDET **Set Detectors Mask** ***detectors_mask***

Controls the reporting of detection for tones and VOX according to the value of the *detectors_mask* parameter. A bit set to 1 in the mask, enables the reporting of the corresponding detector. A bit cleared to 0 disables the reporting.

Disabling reporting of a detector does not stop or reset the detector.

The 1-byte *detectors_mask* is encoded as follows:

Bit 0 Report detection of a busy tone.

Bit 1 Report detection of a dial tone.

Bits 2–4 Reserved. Must be cleared to 0.

Bit 5 Report detection of no energy (VOX) on the line. (The VOX attributes are specified with the tunable parameters VOX_TIME_COUNT and VOX_ENERGY_LEVEL.)

Bit 6 Report the ending of a detected DTMF.

Bit 7 Report the start of a detected DTMF (up to 40 ms after detection start).

Example

SDET A3			
Byte sequence:	Microcontroller	10	A3
	CompactSPEECH	10	A3
Description:	Set reporting of all CompactSPEECH detectors, except for end-of-DTMF.		

SE **Skip to End of Message**

This command is valid only in the **PLAY** state. When invoked, playback is suspended (as for the PA command), and a jump to the end of the message is performed. Playback remains suspended after the jump.

Example

SE		
Byte sequence:	Microcontroller	24
	CompactSPEECH	24
Description:	Skip to end of current message.	

SETD **Set Time and Day** ***time_and_day***

Sets the system time and day as specified by bits 0–13 in the 2-byte *time_and_day* parameter. The *time_and_day* parameter is encoded as follows:

Bits 0–2 Day of the week (1 through 7).

Bits 3–7 Hour of the day (0 through 23).

Bits 8–13 Minute of the hour (0 through 59).

Bits 14–15 These bits must be set to 1.

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

Example

SETD 0E09				
Byte sequence:	Microcontroller	0F	0E	09
	CompactSPEECH	0F	0E	09
Description:	Set time and day to Monday 1:30 AM.			

SF **Skip Forward *time__length***

Skips forward in the current message *time__length* units, each of 0.2s duration, and causes message playback to pause. *time__length* is a 2-byte parameter that can have any value up to 320, i.e., 64s. The skip accuracy is 5%. This command is meaningful only in the **PLAY** state. The RES command must be issued to continue playback.

If the end of the message is detected during execution of SF, execution of the command is terminated, the EV__NORMAL__END bit in the status register is set, the $\overline{\text{MWRQST}}$ signal is activated and the CompactSPEECH switches to the **IDLE** state.

If *time__length* is greater than 320, ERR__PARAM is set in the error word.

Playback speed does not affect the behavior of this command.

Example

SF 19				
Byte sequence:	Microcontroller	22	00	19
	CompactSPEECH	22	00	19
Description:	Skip forward five seconds from the current position in the message being played.			

SMT **Set Message Tag *message__tag***

Sets the tag of the current message. The 2-byte *message__tag* can be used to implement mailbox functions by including the mailbox number in the tag, or to handle old and new messages differently by using one bit in the tag to mark the message as old or new. See Section 2.6.1.

To change the tag of a message, we recommend that you read the message tag, modify it, and write it back.

Note 1: Message tag bits can only be cleared. Message tag bits are set only when a message is first created.

Note 2: If the current message is undefined, ERR__INVALID is reported.

Note 3: Bit 15 of the message tag is used to select the voice compression algorithm and should not be modified after recording.

Example

SMT FF F7				
Byte sequence:	Microcontroller	05	FF	F7
	CompactSPEECH	05	FF	F7
Description:	Mark the current message as old in a system where the message tag is encoded as described in the example of the DMS command. Note that the CompactSPEECH ignores bits in the tag which are set to 1; only bit 3 is modified in the message tag.			

SO **Say One Word *word__number***

Plays the word number *word__number* in the current vocabulary. The 1-byte *word__number* may be any value from 0 through the index of the last word in the vocabulary.

When playback of the selected word has been completed, the CompactSPEECH sets the EV__NORMAL__END bit in the status word, and activates the $\overline{\text{MWRQST}}$ 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.

Example

SO 00			
Byte sequence:	Microcontroller	07	00
	CompactSPEECH	07	00
Description:	Announce the first word in the word table of the currently selected vocabulary.		

SPS **Set Playback Speed *speed***

Sets the speed of message playback as specified by *speed*. The new speed applies to all recorded messages and synthesized messages (only if synthesized using external voice synthesis), until changed by another SPS command. If this command is issued while the CompactSPEECH is in the **PLAY** state, the speed also changes for the message currently being played.

speed may be one of 13 values, from -6 to $+6$. A value of 0 represents normal speed.

Note that a negative *speed* value represents an increase in speed, a positive value represents a decrease in speed.

The change in speed is approximate, and depends on the recorded data.

If *speed* is not in the -6 to $+6$ range, ERR__PARAM is set in the error word.

Example

SPS FB			
Byte sequence:	Microcontroller	16	FB
	CompactSPEECH	16	FB
Description:	Set playback speed to -5.		

SS **Say Sentence *sentence__n***

Say sentence number *sentence__n* of the currently selected vocabulary. *sentence__n* is 1-byte long.

If the sentence has an argument, 0 is passed as the value for this argument.

When playing has been completed, the CompactSPEECH sets the EV__NORMAL__END bit in the status word, and activates the $\overline{\text{MWRQST}}$ 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.

Example

SS 00			
Byte sequence:	Microcontroller	1F	00
	CompactSPEECH	1F	00
Description:	Announce the first sentence in the sentence table of the currently selected vocabulary.		

SV **Set Vocabulary Type *type id***

Selects the vocabulary table to be used for voice synthesis. The vocabulary type is set according to the 1-byte *type* parameter:

0: For compatibility only

1: External vocabulary in ROM

2: External vocabulary in Serial Flash

All others: Reserved

The host is responsible to select the current vocabulary with SV before using an SO, SW, SS or SAS command.

Each external vocabulary table has a unique *id* which is part of the vocabulary internal header (See the *IVS User's Manual* for more details). If *type* is 1 or 2, the CompactSPEECH 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.

Example

SV 02 03				
Byte sequence:	Microcontroller	20	02	03
	CompactSPEECH	20	02	03
Description:	Select the vocabulary with vocabulary-id 3, which resides on Serial Flash, as the current vocabulary.			

SW **Say Words *n word$_1$ . . . word$_n$***

Plays *n* words, indexed by *word$_1$* to *word$_n$*. On completion, the EV__NORMAL__END bit in the status word is set, and the $\overline{\text{MWRQST}}$ 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, ERR__PARAM is set in the error word.

If the current vocabulary is undefined, ERR__INVALID is reported.

Example

SW 02 00 00					
Byte sequence:	Microcontroller	21	02	00	00
	CompactSPEECH	21	02	00	00
Description:	Announce the first word, in the word table of the currently selected vocabulary, twice.				

TUNE **Tune *index parameter__value***

Sets the value of the tunable parameter identified by *index* (one byte) to the 2-byte value, *parameter__value.* This command may be used to tune the DSP algorithms to a specific Data Access Arrangement (DAA) interface, or to change other parameters. If you do not use TUNE, the CompactSPEECH uses default values.

If *index* does not point to a valid tunable parameter, ERR__PARAM is set in the error word.

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

Table 2-2 describes the tunable parameters, their index numbers and their default values.

2.0 Software (Continued)

TABLE 2-2. Tunable Parameters

Index	Parameter Name	Description	Default
0–3	Reserved		—
4	__SIL__THRESHOLD	Prevents speech from being interpreted as silence. The silence detection algorithm has an adaptive threshold, which is changed according to the noise level. This parameter is, therefore, only the initial threshold level. Legal values: 9216 to 13824 in 512 (6 dB) steps.	11264
5	__SIL__THRESHOLD__STEP	Defines the adaptive threshold changes step. If this threshold is too low, the threshold converges too slowly. If it is too high, silence detection is too sensitive to any noise. Legal values: 3 to 48.	12
6	__SIL__BURST__THRESHOLD	The minimum time period for speech detection during silence. As this threshold increases, the time period interpreted as silence increases. If this threshold is too low, a burst of noise is detected as speech. If it is too high, words may be partially cut off. Legal values: 1 to 3.	2
7	__SIL__HANG__THRESHOLD	The minimum time period for silence detection, during speech. As this threshold increases, the time period interpreted as silence decreases. If this threshold is too low, words may be partially cut off. If it is too high, silence is detected. Legal values: 8 to 31.	15
8	__SIL__ENABLE	Silence compression control. 0 turns silence compression off.	1
9	__ENERGY__FACTOR	Determines the energy level used to synthesize silence. For the default value, the energy levels of the synthesized silence and the recorded silence are the same. If you divide (multiply) the default value by two, the synthesized silence is 6 dB less (more) than the level of the recorded silence. Legal values: 1024 to 16384.	8192
10	VOX__ENERGY__THRESHOLD	This constant determines the minimum energy level at which voice is detected. Below this level, it is interpreted as silence. If you divide (multiply) the value by 2 you get 3 dB decrease (increase) in the threshold. Legal values: 0 to 65535.	12
11	Reserved		
12	VOX__TIME__COUNT	This constant, in units of 10 ms, determines the period of silence before the CompactSPEECH reports silence. The accuracy of the constant is ±10 ms. Legal values: 0 to 65535.	700
13–15	Reserved		

TABLE 2-2. Tunable Parameters (Continued)

Index	Parameter Name	Description	Default
16	TONE__GENERATION__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__GENERATION__LEVEL and the VOL__LEVEL variable, controlled by the VC command. The tones are distorted when the level is set too high. The valid range is: 0 ≤ TONE__GENERATION__LEVEL + VOL__LEVEL ≤ 12.	6
17	Reserved		
18	TONE__TIME__COUNT	Controls the duration of a tone before it is reported as a dial tone, in 10 ms units. The accuracy of the constant is ±10 ms. Legal values: 0 to 65535.	700
19	TONE__ON__ENERGY__THRESHOLD	Minimum energy level at which busy and dial tones are detected as ON (after 700 Hz filtering). If you divide (multiply) the value by 2 you get a 3 dB decrease (increase) in the threshold. Legal values: 0 to 65535.	160
20	TONE__OFF__ENERGY__THRESHOLD	Maximum energy level at which busy and dial tones are detected as OFF (after 700 Hz filtering). If you divide (multiply) the value by 2 you get a 3 dB decrease (increase) in the threshold. Legal values: 0 to 65535.	110
21	VCD__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__LEVEL and the VOL__LEVEL variable, controlled by the VC command. Speech is distorted when the level is set too high. The valid range is: 0 ≤ VCD__LEVEL + VOL__LEVEL ≤ 12.	6
22	VOX__TOLERANCE__TIME	Controls the maximum energy period, in 10 ms units, that does NOT reset the vox detector. Legal values: 0 to 255.	3
23	MIN__BUSY__DETECT__TIME	Minimum time period for busy detection, in 10 ms units. The accuracy of the constant is ±10 ms. Legal values: 0 to 65535.	600
24	ECHO__DELAY	The near-echo delay in samples. The sampling rate is 8000 Hz (i.e., 125 μs per sample). Legal values: 0 to 16.	4
25	Reserved		
26	DTMF__REV__TWIST	Controls the reverse twist level at which CompactSPEECH detects DTMF tones. While the normal twist is set at 8 dB, the reverse twist can be either 8 dB (default) or 4 dB (If this parameter is set to 1).	0

TABLE 2-2. Tunable Parameters (Continued)

Index	Parameter Name	Description	Default
27	DTMF__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 CompactSPEECH 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 meaurements was 6 (TONE__GENERATION__LEVEL + VOL__LEVEL = 6). 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
28	Reserved		
29	Reserved		

Example

TUNE 23 700					
Byte sequence:	Microcontroller	15	17	02	BC
	CompactSPEECH	15	17	02	BC
Description:	Set the minimum period for busy detection to seven seconds.				

VC — **Volume Control *vol__level***

Controls the energy level of all the output generators (playback, tone generation, and voice synthesis), with one command. The resolution is ±3 dB.

The actual output level is composed of the tunable level variable, plus the *vol__level.* The valid range for the actual output level of each output generator is defined in Table 2-2.

For example, if the tunable variable VCD__LEVEL is 6, and *vol__level* is −2, then the output level equals VCD__LEVEL + *vol__level* = 4.

Example

VC 04			
Byte sequence:	Microcontroller	28	04
	CompactSPEECH	28	04
Description:	Set the volume level to VCD__LEVEL + 4.		

WRAM — **Write Memory *tag, data***

This command creates a new message with a message tag *tag.* The following 32 bytes of data *data* are stored as the new message data in the message memory.

The WRAM command switches the CompactSPEECH to the **MEMORY__WRITE** state. As long as it remains in this state, each subsequent WMEM command appends new message data to the end of the previous data. The CompactSPEECH remains in the **MEMORY__WRITE** state until an S command is issued. Note that, while the CompactSPEECH is in the **MEMORY__WRITE** state, *tag* is ignored.

If the memory becomes full, recording stops and EV__MEMFULL is set in the status word.

Example

WMEM 1 *32 bytes*				
Byte sequence:	Microcontroller	17	01	32 bytes of data to write
	CompactSPEECH	17	01	echo 32 bytes of data
Description:	Create a message with tag = 01, and write 32 bytes in the message memory.			

Appendix A

SCHEMATIC DIAGRAMS

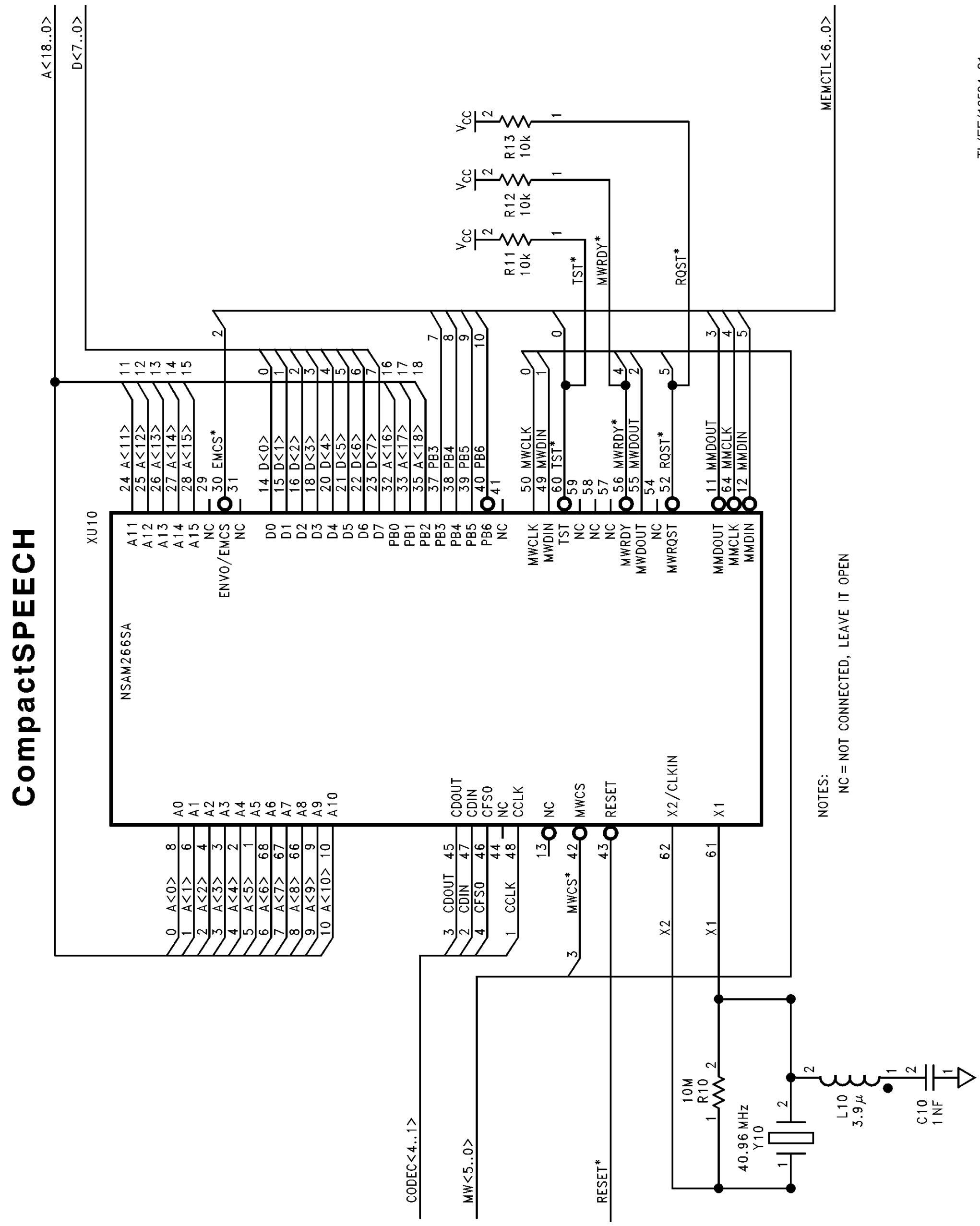
The following schematic diagrams are extracted from a CompactSPEECH demo unit, based on the NSV-AM266-SPAF board, designed by National Semiconductor.

This demo includes three basic clusters:

- COP888EEG Microcontroller.
- CompactSPEECH cluster, including a TP3054 codec and an NSAM266SA controlling a Serial Flash device.
- User interface that includes one 16-digit LCD, and a 16-key (4 x 4) keypad.

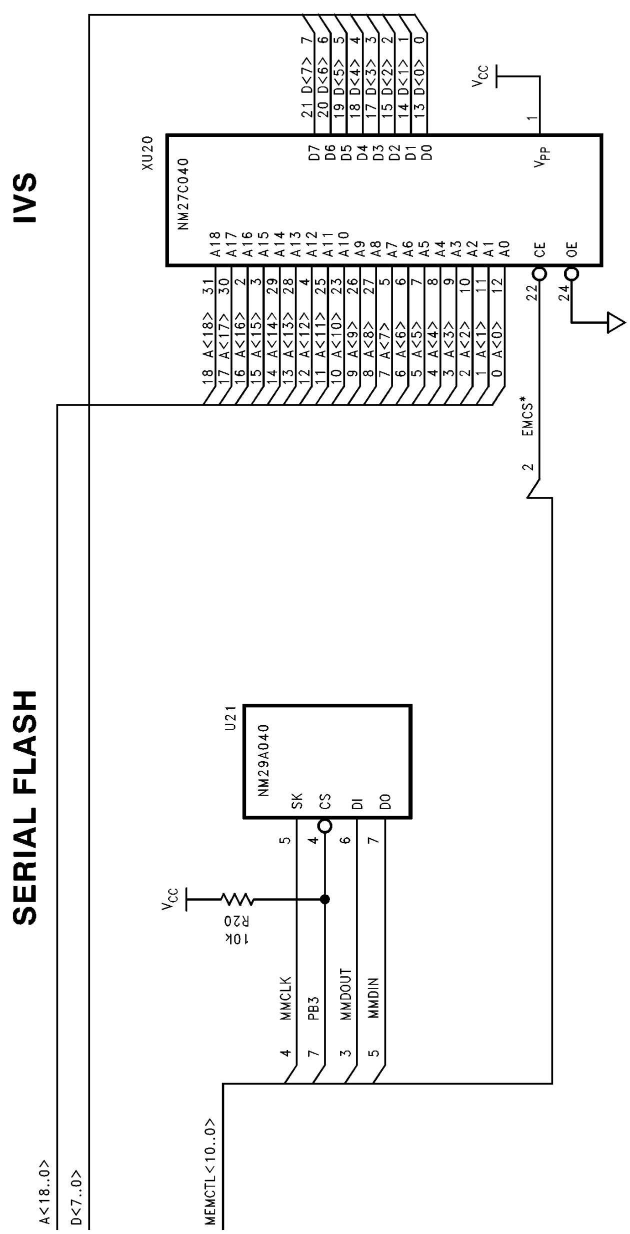
For more details about the demo please refer to the *NS Digital Answering Machine Demo Operating Instructions.*

Note: If IVS resides in serial flash, and not in ROM, the address- and data-line connections are not required, and the layout is much simpler.

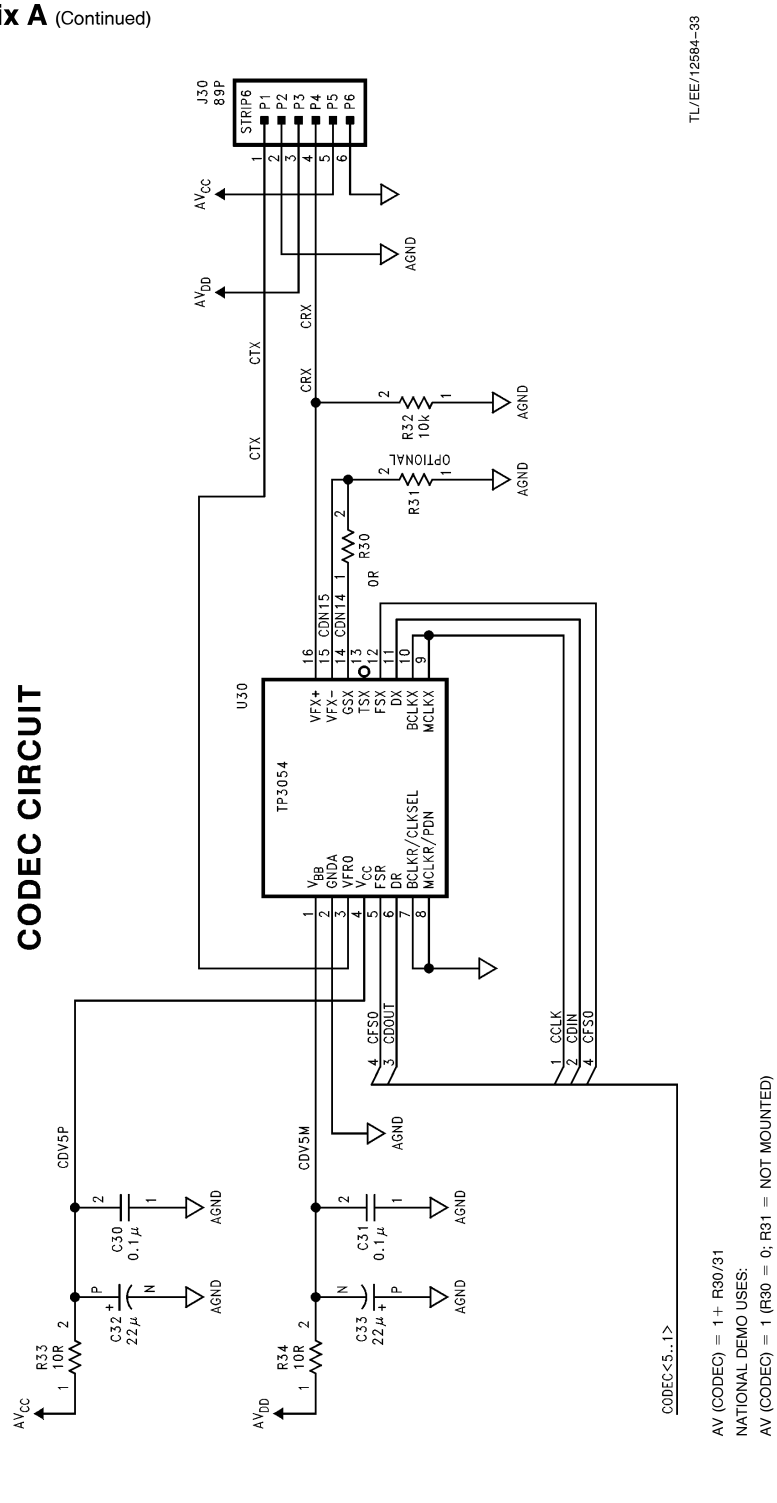


TL/EE/12584-31

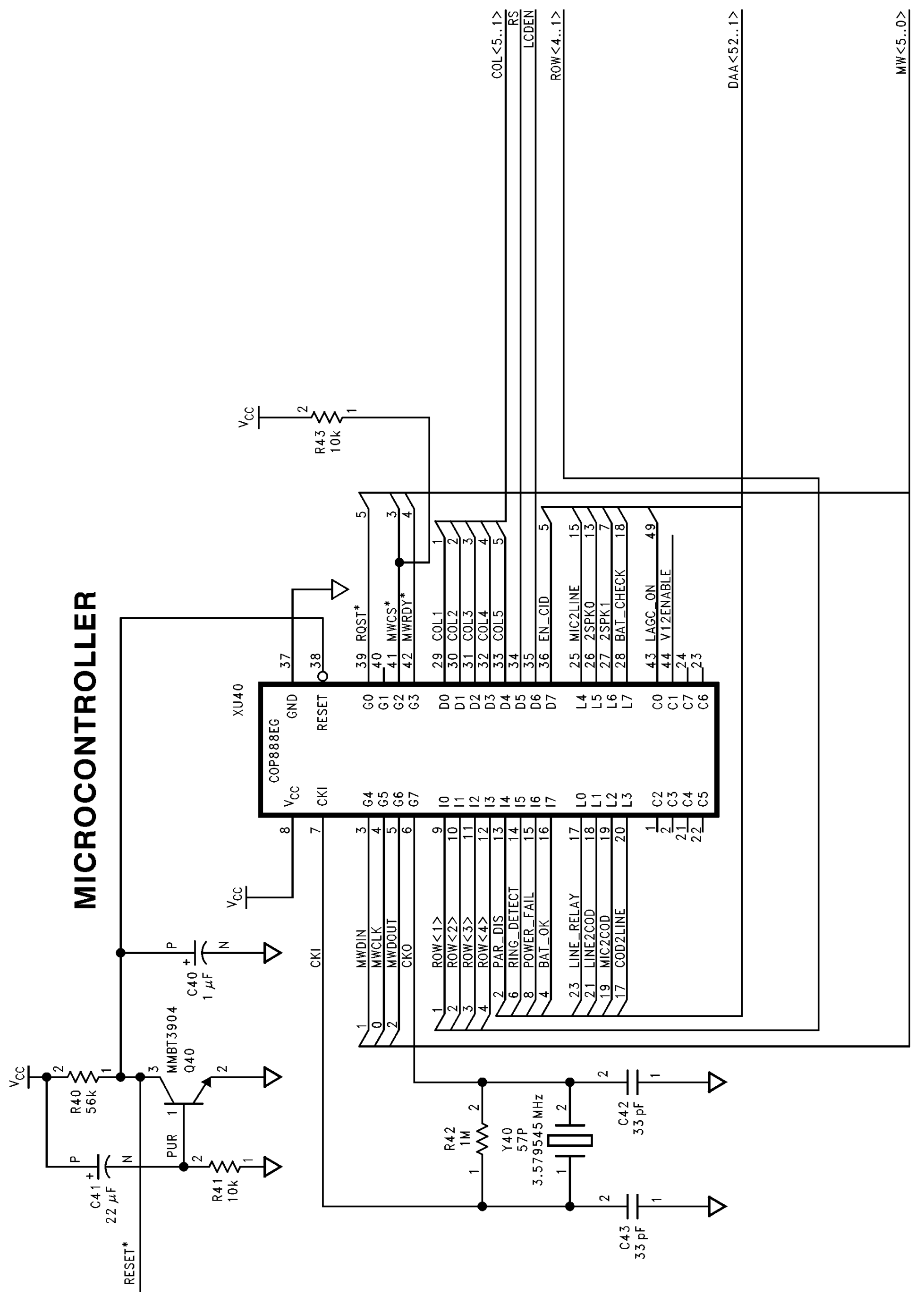
Appendix A (Continued)



TL/EE/12584–32

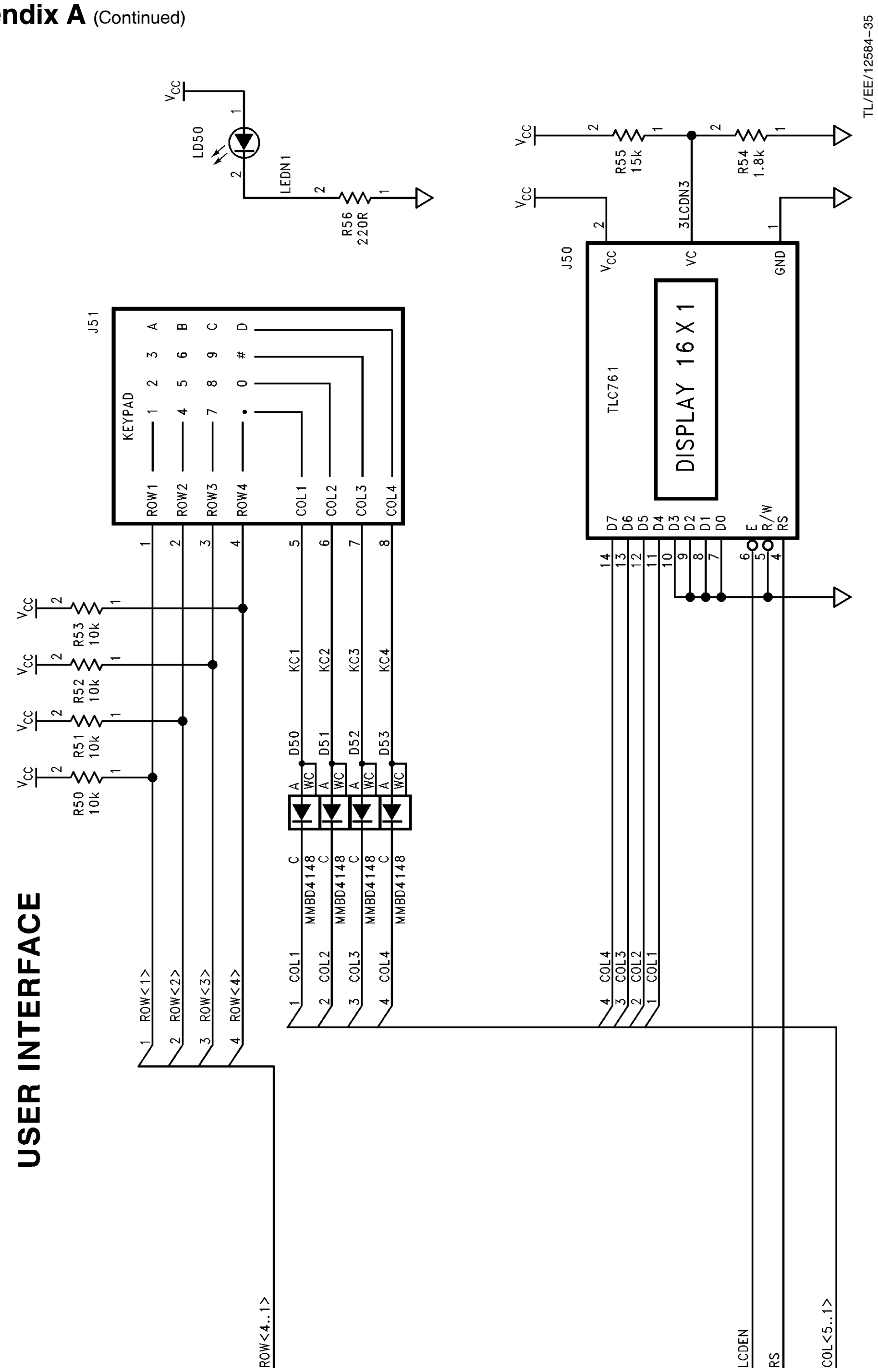

CODEC CIRCUIT
J30
89P
STRIP6
P1
P2
P3
P4
P5
P6
AVCC
AVDD
AGND
CTX
CRX
R32
10k
R31
OPTIONAL
R30
0R
CDN15
CDN14
U30
TP3054
VFX+
VFX−
GSX
TSX
FSX
DX
BCLKX
MCLKX
VBB
GNDA
VFRO
VCC
FSR
DR
BCLKR/CLKSEL
MCLKR/PDN
CFS0
CDOUT
CCLK
CDIN
CDV5P
CDV5M
C30
0.1µ
C31
0.1µ
C32
22µ
C33
22µ
R33
10R
R34
10R
CODEC<5..1>
AV (CODEC) = 1 + R30/31
NATIONAL DEMO USES:
AV (CODEC) = 1 (R30 = 0; R31 = NOT MOUNTED)
TL/EE/12584-33

Appendix A (Continued)

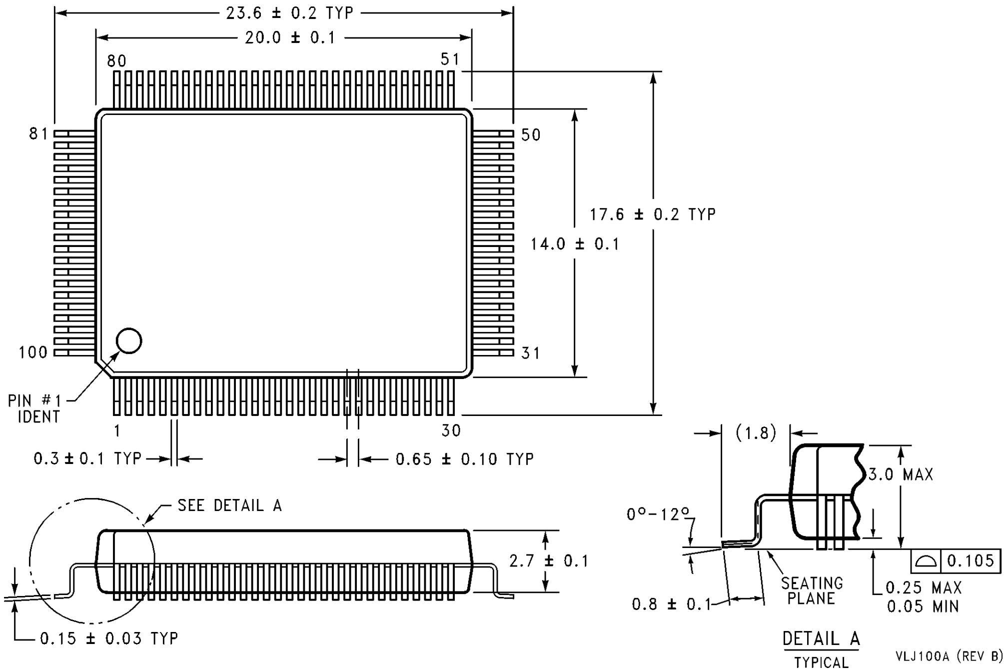


TL/EE/12584-34

Appendix A (Continued)

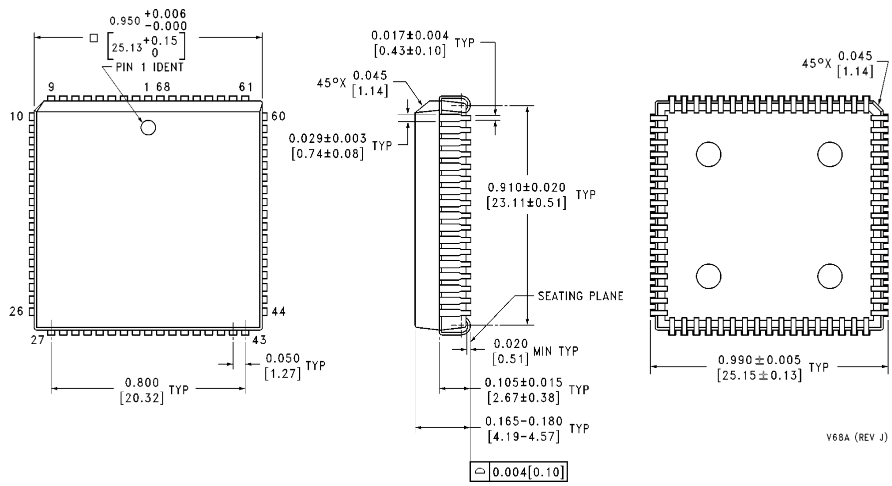


Physical Dimensions inches (millimeters)



100-Pin Molded Plastic Quad Flatpak (EIAJ)
Order Number NSAM266SAA/VLJ
NS Package Number VLJ100A

Physical Dimensions inches (millimeters) (Continued)



68-Pin Plastic Leaded Chip Carrier (V)
Order Number NSAM266SAA/V
NS Package Number V68A

National Semiconductor Corporation
1111 West Bardin Road
Arlington, TX 76017
Tel: 1(800) 272-9959
Fax: 1(800) 737-7018

National Semiconductor Europe
Fax: +49 (0) 180-530 85 86
Email: europe.support@nsc.com
Deutsch Tel: +49 (0) 180-530 85 85
English Tel: +49 (0) 180-532 78 32
Français Tel: +49 (0) 180-532 93 58
Italiano Tel: +49 (0) 180-534 16 80

National Semiconductor Hong Kong Ltd.
13th Floor, Straight Block,
Ocean Centre, 5 Canton Rd.
Tsimshatsui, Kowloon
Hong Kong
Tel: (852) 2737-1600
Fax: (852) 2736-9960

National Semiconductor Japan Ltd.
Tel: 81-043-299-2308
Fax: 81-043-299-2408

http://www.national.com