56858

Feature Phone Application using Processor Expert™

Targeting User Guide

DSP56850
16-bit Digital Signal Controllers

FP56858UG
Rev. 0
9/2005

freescale.com

freescale™ semiconductor

TABLE OF CONTENTS

About This Document ix

Audience ix
Organization ix
Suggested Reading ix

Preface ix

Conventions x
Definitions, Acronyms, and Abbreviations x
References xi

Chapter 1
Introduction

1.1 Quick Start. 1-1
1.2 Telephony Features Libraries 1-1
1.3 Overview of the Feature Phone Application 1-2
1.3.1 System Block Diagram. 1-2
1.3.1.1 Feature Phone Application Hardware Setup. 1-3
1.3.2 Software Specification 1-4
1.3.2.1 Feature Phone Application 1-5
1.3.2.2 Type 1 and 2 Telephony Features Library 1-5
1.3.2.3 Type 1 and 2 Telephony Parser Library 1-5
1.3.2.4 Generic Echo Canceller Library 1-5
1.3.2.5 Full Duplex Speakerphone Library. 1-6
1.4 Resource Estimates 1-6

Chapter 2
Interfaces

2.1 HOST-to-DSP COMMANDS. 3-1
2.1.1 Speakerphone Control 3-1
2.1.2 DTMF Generation 3-1
2.1.3 Hookswitch Control 3-2
2.1.4 Volume Control 3-2
2.1.5 Read Version Code. 3-2
2.1.6 Digital Switches for 3-Port Routing. 3-3
2.1.7 Call Waiting Deluxe Features 3-3
2.2 DSP-to-HOST COMMANDS. 3-4
2.2.1 Caller ID and Caller ID on Call Waiting Data 3-4
2.2.2 Additional Caller ID Parameters 3-5

2.2.3 FSK Data Timeout Parameters 3-5
2.2.4 Special Signal Detection Events 3-5
2.2.5 Erroneous Occurrences 3-6

Chapter 3
Building the Feature Phone Application

3.1 Building the Feature Phone Application 4-1

Chapter 4
Feature Phone Application

4.1 Feature Phone Application 5-1
4.2 Application Overview 5-1
4.3 Application Software 5-3

Chapter 5
License

5.1 Limited Use License Agreement 6-1

LIST OF TABLES

1-1 Resource Estimate Requirements 1-6
2-1 Speakerphone Control Command 3-1
2-2 DTMF Generation Command 3-1
2-3 Hookswitch Control Command. 3-2
2-4 Allowable Values for Hookswitch and Muting Control Parameter 3-2
2-5 Volume Control Command 3-2
2-6 Read Version Code Commands 3-2
2-7 Configure Digital Switches for 3-Port Routing. 3-3
2-8 Call Waiting Deluxe Commands for the Type 1 and 2 Telephony Features Library 3-3
2-9 Caller ID and Caller ID Call Waiting Data 3-4
2-10 Additional Caller ID Parameters 3-5
2-11 Timeout Parameters. 3-5
2-12 Special Signal Detection Events 3-5

LIST OF FIGURES

1-1 System Block Diagram 1-3
1-2 Software Block Diagram 1-4
1-3 Detailed Signal Flow of the Telephony Libraries 1-4

LIST OF EXAMPLES

4-1 Feature Phone Application Software Code 5-3

Preface

About This Document

This manual describes the Feature Phone Application for Freescale's DSP56858 16-bit Hybrid Controller.

Audience

This document targets software developers implementing communication features for analog telephone lines.

Organization

This guide is arranged in the following sections:

- **Chapter 1, Introduction**—provides a brief overview of this document
- **Chapter 2, Interfaces**—describes the AT Command set used in the Feature Phone Application
- **Chapter 3, Building the Feature Phone Application**—tells how to build the Feature Phone Application
- **Chapter 4, Feature Phone Application**—describes the use of the Feature Phone Application
- **Chapter 5, License**—provides the license required to use this product

Suggested Reading

We recommend that you have a copy of the following references:

- *DSP56800E Reference Manual,* DSP56800ERM
- *DSP5685x User's Manual*, DSP5685xUM
- *Inside CodeWarrior: Core Tools*, Metrowerks Corp.

Conventions

This document uses the following notational conventions:

Typeface, Symbol or Term	Meaning	Examples
`Courier Monospaced Type`	Code examples	`//Process command for line flash`
Italic	Directory names Project names Calls Functions Statements Procedures Routines Arguments File names Applications Variables Directives Code snippets in text	...and contains these core directories: *applications* contains applications software... ...CodeWarrior project, *3des.mcp* is... ...the *pConfig* argument.... ...defined in the C header file, *aec.h*....
Bold	Reference sources Paths Emphasis	...refer to the **Targeting DSP5685x Platform** manual....
ALL CAPITAL LETTERS	# defines/ Defined constants	# define INCLUDE_STACK_CHECK
Brackets [...]	Function keys	...by pressing function key [F7]
Quotation marks, “...”	Returned messages	...the message, “Test Passed” is displayed.... ...if unsuccessful for any reason, it will return “NULL”...
Blue Text	Linkable on-line	...refer to Chapter 7, License....
Number	Any number is considered a positive value, unless preceded by a minus symbol to signify a negative value	3V -10 DES^{-1}

Definitions, Acronyms, and Abbreviations

The following list defines the acronyms and abbreviations used in this document. As this template develops, this list will be generated from the document. As we develop more group resources, these acronyms will be easily defined from a common acronym dictionary. Please note that while the acronyms are in solid caps, terms in the definition should be initial capped ONLY IF they are trademarked names or proper nouns.

ADC Analog-to-Digital Converter

AGC Automatic Gain Control

API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
BPF	Bandpass Filter
CID	Caller ID (Calling Party Name and/or Number Identification)
CPE	Customer Premises (telephony) Equipment
CQ	Call Qualifier
DAA	Data Access Arrangement
DSP	Digital Signal Processor or Digital Signal Processing
DTMF	Dual Tone Multiple Frequency
FSK	Frequency Shift Keying modulation
IDE	Integrated Development Environment
LPF	Low Pass Filter
MDMF	Multiple Data Message Format of GR-30-CORE
MIC	Microphone
MIPS	Million Instructions Per Second
OnCE™	On-Chip Emulation
PC	Personal Computer
PCM	Pulse Code Modulation
PE	Processor Expert
PSTN	Public Switched Telephone Network
SDMF	Single Data Message Format of GR-30-CORE
SRC	Source
VMWI	Visual Message Waiting Indicator

References

The following sources were referenced to produce this book:

1. *DSP56800E Reference Manual*, DSP56800ERM/D
2. *DSP5685x User's Manual*, DSP5685xUM/AD
3. SR-3004, *Testing Guidelines for Analog Type 1, 2, and 3 CPE as Described in SR-INS-002726 (a module of ADSI, FR-12),* Telcordia Technologies, January 1995.
4. GR-30-CORE, *LSSGR: Voiceband Data Transmission Interface Section 6.6 (a module of LSSGR, FR-64),* Telcordia Technologies, December 1998.
5. GR-31-CORE, *LSSGR CLASS^{SM} Feature: Calling Number Delivery (FSD 01-02-1051) (a module of LSSGR, FR-64),* Telcordia Technologies, June 2000.
6. GR-1188-CORE, *LSSGR CLASS^{SM} Feature: Calling Name Delivery Generic Requirements (FSD 01-02-1070) (a module of LSSGR, FR-64),* Telcordia Technologies, December 2000.
7. GR-1401-CORE, *LSSGR CLASS^{SM} Feature: Visual Message Waiting Indicator Generic Requirements (FSD 01-02-2000), (a module of LSSGR, FR-64),* Telcordia Technologies, June 2000.

8. *ITU-T Recommendation V.23*, 600/1200-baud modem standardized for use in the general switched telephone network, November 1988.
9. GR-575-CORE, *LSSGR CLASS^{SM} Feature Calling Identity Delivery on Call Waiting, (FSD 01-02-109), (a module of LSSGR, FR-64),* Telcordia Technologies, June 2000.
10. GR-416-CORE, *CLASS^{SM} Feature Call Waiting Deluxe, (FSD 01-02-1215), (a module of LSSGR, FR-64)* Telcordia Technologies, December 1999.
11. SR-TSV-002476, *CPE Compatibility Considerations For The Voiceband Data Transmission Interface,* Telcordia Technologies, December 1999.
12. SR-NWT-002024, *Customer Premises Equipment Compatibility Considerations for the SPCS-to-CPE Data Transmission Interface,* Telcordia Technologies, December 1999.
13. ANSI/TIA/EIA-470B, *Telephony Instruments with Loop Signaling,*1997.
14. ANSI/TIA/EIA-716, *Telecommunications Telephone Terminal Equipment - Type 1 Caller Identity Equipment Performance Requirements,* 1999.
15. ANSI/TIA/EIA-777, *Telecommunications Telephone Terminal Equipment - Type 2 Caller Identity Equipment Performance Requirements,* 1999.

Chapter 1
Introduction

This document describes the Feature Phone Application. In this document, you will find all the information required to use and maintain the Feature Phone Application.

Freescale provides these algorithms to you under license for use with Freescale's DSP to expedite your application development and reduce the time it takes to bring your own products to market.

Freescale's Feature Phone Application is a licensed software library for use on Freescale 56800E series processors. Please refer to the Software License Agreement in **Chapter 5** for license terms and conditions.

1.1 Quick Start

Processor Expert (PE) help system on the Feature Phone bean can get you started fast. Look for the "targeting" manuals in the PE help system or use the PE search feature.

1.2 Telephony Features Libraries

The Feature Phone Application is one of a set of modules and applications consisting of the following:

- Type 1 Telephony Features Library
- Type 1 and 2 Telephony Features Library
- Type 1 and 2 Telephony Parser Library
- Full Duplex Speakerphone Library
- Generic Echo Canceller Library
- Feature Phone Application

These modules are designed to interoperate to provide all of the software necessary to implement a feature phone with full duplex speakerphone and Type 1 and 2 Caller ID functionality. Using some or all of these modules, several other types of telephony applications are also possible. Each module may also be used independently.

1.3 Overview of the Feature Phone Application

The Feature Phone Application is a hardware and software implementation which includes a single processor architecture design that demonstrates the abilities of the 56800E family of devices. The Feature Phone Application implements the Type 1 and 2 Telephony Features Library. This Library includes support for on-hook GR-30 services:

- Calling Number Delivery
- Calling Name Delivery
- Dialable Directory Number
- Call Qualifier
- Visual Message Waiting Indicator

The library includes support for off-hook GR-30 services:

- Calling Identity Delivery on Call Waiting
- Calling Identity Delivery with Call Waiting Deluxe
- Dialable Directory Number
- Call Qualifier

The Feature Phone Application also includes a full duplex feature phone with solid sound quality, offering the ability to originate and terminate a call in full duplex feature phone mode.

User interface:

- HyperTerminal, to display GR-30 messages
- Keyboard, to enter AT commands
- Microphone and speaker, for human interface

Libraries:

- Feature Phone Application
- Type 1 and 2 Telephony Features Library
- Type 1 and 2 Telephony Parser Library
- Generic Echo Canceller Library
- Full Duplex Speakerphone Library
- Embedded Drivers

Hardware:

- DSP56858EVM
- Telephony Daughter Card (TDC1)
- Amplified microphone
- Amplified speaker

1.3.1 System Block Diagram

Figure 1-1 illustrates a system block diagram of the Feature Phone Application.

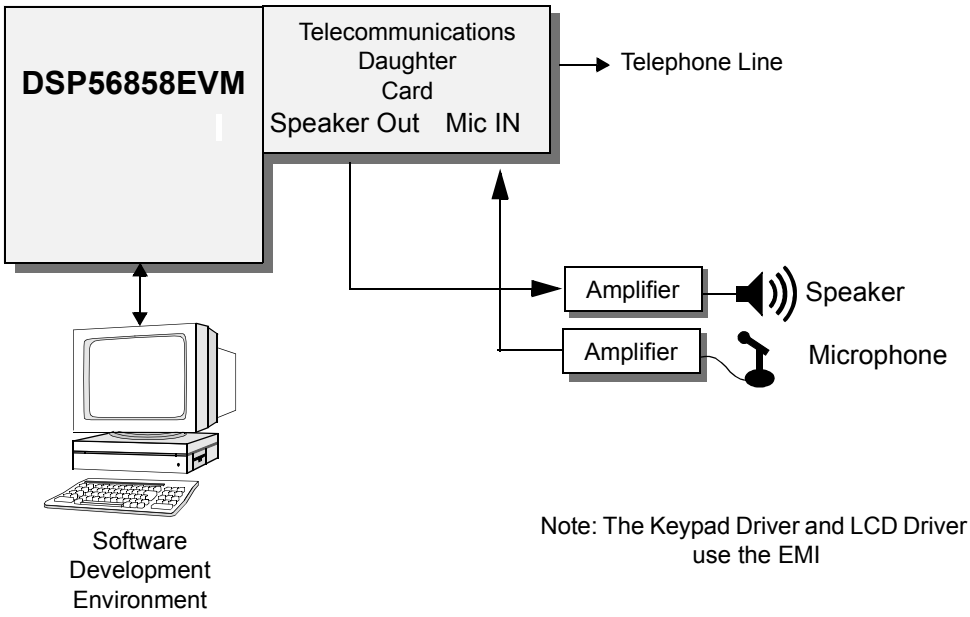

Figure 1-1. System Block Diagram

1.3.1.1 Feature Phone Application Hardware Setup

Do the following to configure the hardware for the Feature Phone Application:

DSP56858 EVM board

1. All jumpers on the SSI0 connector (JG6) must be removed. This bypasses the on-board EVM codec to the TDC1 board codec.

TDC1 board

1. ESSI & Channel Switch 1 (S1) configuration: 1 - on, 2 - off, 3 - off.
2. A mono amplified speaker is plugged into Ch1 Speaker
3. An amplified microphone is attached to the connector.

1.3.2 Software Specification

This section describes the software fuctional requirements of the Feature Phone Application. Software must be developed for two main areas: telephony software and driver software (Board Support). **Figure 1-2** illustrates a block diagram of the software used for the Feature Phone Application. A detailed block diagram of the signal flow of the libraries is shown in **Figure 1-3**.

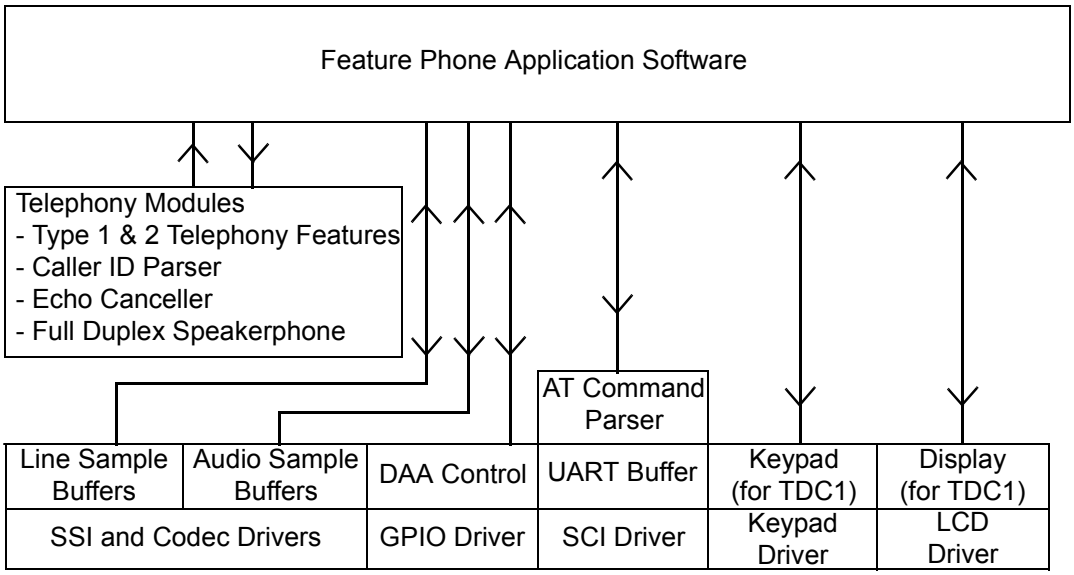

Figure 1-2. Software Block Diagram

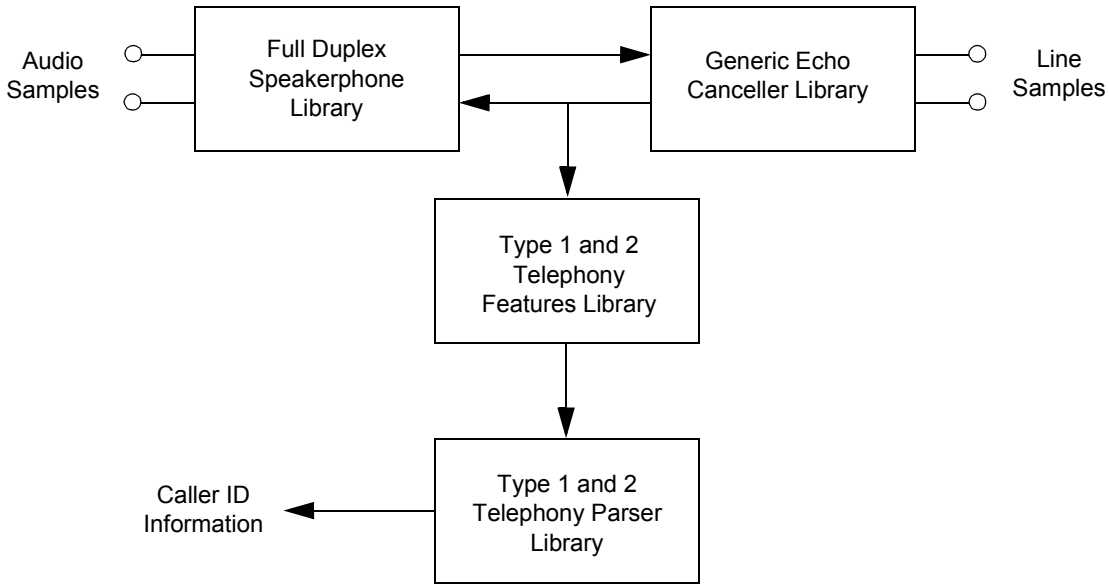

Figure 1-3. Detailed Signal Flow of the Telephony Libraries

1.3.2.1 Feature Phone Application

The Feature Phone Application provides the framework for a complete Feature Phone solution on the DSP56858EVM, using the four Telephone Features Libraries described in the following sections. This software contains the main sample processing loop, which will call other libraries. Also included in this software is initialization and buffer allocation for all of the telephony libraries, as well as initialization routines and buffers for hardware control and drivers. The Feature Phone Application implements the interface between hardware control, software (device drivers, peripheral drivers, etc.) and the telephony libraries. All input commands and data samples passed to the libraries are contained in one of three data structures as input parameters to a function call. These data structures are then modified by the called library to create new output samples as well as telephony events such as Caller ID information. The Feature Phone Application also contains basic local-feature functionality, including a DTMF string dialer, Terminal Interface, and an AT command set parser.

1.3.2.2 Type 1 and 2 Telephony Features Library

This library contains routines which allow for reception of both on-hook (Type 1) and off-hook (Type 2) Caller ID. Functionality includes a Bell 202 Modem Receiver, CPE Alerting Signal (CAS) Detector, CAS Acknowledgment Signal Generator and Handshake Timers, Extension-in-Use Detection Timers, and a complete state machine which allows for simple decoding of 8KHz input samples into Type 1 and 2 Caller ID data bytes using a single function call. The Bell 202 receiver is highly robust and can perform under various signal tolerances by employing techniques such as symbol timing recovery, carrier frequency-shift tracking, and automatic gain control. The software meets or exceeds all signaling specifications for Caller ID Customer Premises Equipment as defined by Telcordia SR-3004. In addition, a general purpose DTMF Generator, Ring Signal Detector, and Ring Tone Generator are included to enable a complete telephony solution.

1.3.2.3 Type 1 and 2 Telephony Parser Library

This library is used with the Type 1 and 2 Telephony Features Library to decode and parse the received Caller ID bytes into the various message and parameter types as defined by the Telcordia Specification SR-3004. The library will parse Caller ID data bytes into Date, Time, Calling Name, Calling Number, Reason for Name or Number Absence, etc. In addition, a checksum calculation and other error detection methods are performed. Single Data Message Format (SDMF) and Multiple Data Message Format (MDMF) are supported.

1.3.2.4 Generic Echo Canceller Library

This library provides a flexible, complete general-purpose echo cancellation system. Two streams of 8KHz samples are used as the input to the library. The first stream is the reference signal to be output to the echo interface. The second stream is an input from the echo interface, which contains both an echo of the reference plus an additive independent input signal. The echo canceller is capable of achieving an echo attenuation of greater than 40dB (under white noise input), while the independent input signal remains unchanged. Advanced adaptive filtering techniques are employed which allow the system to adapt to and perform particularly well in situations where speech is used. The library contains voice activity detection and complete state machine software, so that echo cancellation of a channel can be accomplished using a single simple function call. Tail lengths from 8 to 64ms (in increments of 8ms) may be selected by setting an input parameter to the library.

1.3.2.5 Full Duplex Speakerphone Library

This library contains algorithms which implement a full duplex hands-free telephone using a powered microphone and speaker. The software contains its own complete acoustic echo cancellation system and logarithmic suppression controller and may be used in various types of acoustic echo environments. Since speakerphones typically interface with a telephone line, the library incorporates a simple interface with the Generic Echo Canceller Library which can be used to combat line echo. Together, the libraries provide all of the algorithms and state machines needed for a natural-sounding, full duplex telephone conversation using only two simple function calls. The library also supports other standard features, including digital volume control and system diagnostic software, allowing a developer to "tune" his speakerphone settings for optimal results. Given a specific set of hardware, by simply setting a few parameters in the application software, a speakerphone can be tuned to optimal performance. Tail lengths from 8ms to 64ms (in increments of 8ms) may be selected by setting an input parameter to the library.

1.4 Resource Estimates

Table 1-1 provides an estimate of resources required on the 56858 for the Feature Phone Application, detailing the memory (Program, Table and Data) and MIPs requirements on the 56858 chip.

Table 1-1. Resource Estimate Requirements

Application	Program	Data	MIPS
Type 1 and 2 Telephony Features Library	2.8K	257	9.60
Type 1 and 2 Telephony Parser Library	1.4K	550	N/A (not real time)
Generic Echo Canceller Library	680	580	10.88
Full Duplex Speakerphone Library (with 24ms tail length)	2203	819	12.96
Feature Phone Application[1]	2573	300	3.00
Total	9757	2506	36.40

1. Not Including Processor Expert Drivers

NOTE:

All libraries utilized by the Feature Phone Application can be independently run out of internal or external memory. Due to realtime constraints within the Feature Phone Application, all libraries except the parser library must be run in internal memory.

Chapter 2
Interfaces

2.1 HOST-to-DSP COMMANDS

Communications between DSP and HOST are carried out through an asynchronous serial port using 8 data bits, 1 start bit, 1 stop bit, and no parity bit. The baud rate is 38,400bps. The following sections describe the required AT commands used in the Feature Phone Application, which are transmitted in ASCII as shown followed by a carriage return. Delete characters are invalid. The DSP command interpreter may be reset by sending the "?" character (0x3f).

2.1.1 Speakerphone Control

Table 2-1 describes the Control Speakerphone MUX and enables acoustic echo cancellation when the CPE is off-hook.

Table 2-1. Speakerphone Control Command

Command	Function
AT+VSP = n	n = 0; Disables Speakerphone n = 1; Enables Speakerphone in full duplex mode (default) n = 2; Enables Speakerphone in half duplex mode

2.1.2 DTMF Generation

Table 2-2 describes the AT commands required to generate DTMF signals.

Table 2-2. DTMF Generation Command

Command	Function
AT+VTS = m	Dial DTMF digit(s): AT+VTS = 3 (dials the digit 3) AT+VTS = 34 (dials the digits 3 and 4) AT+VTS = 5551212 (dials the string of digits indicated)

2.1.3 Hookswitch Control

Table 2-3 describes the command required to create the events needed to inform the DSP to take the CPE on or off-hook and to inform the DSP to mute the microphone and/or speaker.

Table 2-3. Hookswitch Control Command

Command	Function
AT+VLS = n	n can be any of the values shown in Table 2-4

Table 2-4. Allowable Values for Hookswitch and Muting Control Parameter

n	Hookswitch	Speaker	Microphone	Remarks
0	On-Hook	Muted or Ringer	Muted	Use when phone is on-hook
7	Off-Hook	Unmuted	Unmuted	Use for normal operations

2.1.4 Volume Control

Table 2-5 describes the commands used to set the speaker volume for the handset or speakerphone. The volume will be set for whichever is currently active. The range boundaries are from -24dB to +24dB.

Table 2-5. Volume Control Command

Command	Function
AT+VGS = m	m is an integer between -24 to 24dB AT+VGS = 18 (sets volume to 18dB) AT+VGS = -20 (sets volume to -20dB) AT+VGS = 4 (sets volume to 4dB)
<	Volume up by 1dB
>	Volume down by 1dB

2.1.5 Read Version Code

Table 2-6 describes the command used to inform the application to respond with the current DSP firmware version code.

Table 2-6. Read Version Code Commands

Command	Function
AT+VTV = <1>	This command informs the Host to send the current DSP firmware version code

2.1.6 Digital Switches for 3-Port Routing

Table 2-7 describes the command required to configure the digital switches which are used for 3-port routing. The six least significant bits of the hexadecimal value *0xnnnn* are used to configure the digital switches for the Full Duplex Speakerphone Library.

Table 2-7. Configure Digital Switches for 3-Port Routing

Command	Function
AT+VDS = nnnn	Bit 0 - Configures Digital Switch D1 (0 = open, 1 = closed) Bit 1 - Configures Digital Switch D2 (0 = open, 1 = closed) Bit 2 - Configures Digital Switch D3 (0 = open, 1 = closed) Bit 3 - Configures Digital Switch D4 (0 = open, 1 = closed) Bit 4 - Configures Digital Switch D5 (0 = open, 1 = closed) Bit 5 - Configures Digital Switch D6 (0 = open, 1 = closed) AT+VDS = 0011; Close Switches 1 and 5, all others open

2.1.7 Call Waiting Deluxe Features

Table 2-8 describes the character code pairs designated for the specific commands used for the Call Waiting Deluxe disposition options. The format should be:

```
<$><code char>
```

For example, <$><0> is transmitted as 0x24 0x30.

Table 2-8. Call Waiting Deluxe Commands for the Type 1 and 2 Telephony Features Library

Command	Function
<$><3>	Call Waiting Deluxe Command - Conference
<$><5>	Call Waiting Deluxe Command - Drop First
<$><6>	Call Waiting Deluxe Command - Hold
<$><7>	Call Waiting Deluxe Command - Drop
<$><8>	Call Waiting Deluxe Command - Announcement
<$><9>	Call Waiting Deluxe Command - Forward
<$><F>	Call Waiting Deluxe Command - Answer or Regular Flash

2.2 DSP-to-HOST COMMANDS

Command Acknowledgement Reports:

- OK - Acknowledges that the DSP has received and processed an event sent by the Host

Event Reports:

- These events are sent in the following format:
  <tag>=<data><CR>

2.2.1 Caller ID and Caller ID on Call Waiting Data

Table 2-9 describes the tags that support the Single Message Data Format and also Reason for Number Absence.

Table 2-9. Caller ID and Caller ID Call Waiting Data

Tag	Description
TIME	TIME = HHMM HH = the hour, 00-23 MM = the minute, 00-59 All numbers are in ASCII decimal, and for numbers less than 10, a leading zero is required Example: TIME = 0803
DATE	DATE = MMDD MM = month, 01-12 DD = day, 01-31 All numbers are in ASCII decimal, and for numbers less than 10, a leading zero is required Example: DATE = 0321
NMBR	NMBR = <number> number = the telephone number of the caller NMBR = P indicates that the calling number is not available since the caller has requested Private service NMBR = O indicates that the calling information is not available because the caller is out of area code Example: NMBR = 7325551212
NAME	NAME = <Listing Name> <Listing Name> is the subscription listing name Example: NAME = John Doe

2.2.2 Additional Caller ID Parameters

Table 2-10 describes tags that are used for additional Caller ID parameters.

Table 2-10. Additional Caller ID Parameters

Tag	Description	
XZRNA	XZRNA = P[1] XZRNA = O[1]	Reason for Name Absence
XZCLQ	XZCLQ = L	Call Qualifier
XZDDN	XZDDN = <number>	Dialable Directory Number
XZRRD	XZRRD = 0 or 1 or 2[2]	Reason for Redirection
XZVMI	XZVMI = 0 or 1	Visual Message Waiting Indicator 0 = OFF 1 = ON

1. “P” indicates that the calling number is not available since the caller has requested Private service
   “O” indicates that the calling information is not available because the caller is out of area code
2. “0” indicates "Call Forward Universal"
   “1” indicates "Call Forward Busy"
   “2” indicates "Call Forward Unanswered"

2.2.3 FSK Data Timeout Parameters

Table 2-11 describes the tag used when data was not sent to the CPE within 500ms of the DTMF Acknowledgement signal.

Table 2-11. Timeout Parameters

Tag	Description	
XZTMO	XZTMO = 1	FSK Data timeout

2.2.4 Special Signal Detection Events

Table 2-12 describes custom tags that are used for special signal detection events.

Table 2-12. Special Signal Detection Events

Tag	Description	
XZCAS	XZCAS = 1	CPE Alerting Signal (CAS) found
XZUSE	XZUSE = 1	CPE Alerting Signal (CAS) found, but no data will follow, since an extension phone is off-hook

2.2.5 Erroneous Occurrences

To indicate to the Host that an error has occurred, the following format is specified:

```
ERRM=<data><CR>
```

As used here, “data” may represent ICLID_202 or Z<text>.

When unknown parameters are found within the message, the MESG tag is specified as follows:

```
MESG=<message type><message length><data><checksum>
```

Chapter 3
Building the Feature Phone Application

3.1 Building the Feature Phone Application

The Feature Phone Application for the 56858 combines all of the components described in **Chapter 1** into one project. The Type 1 and 2 Telephony Features Library, Type 1 and 2 Telephony Parser Library, Generic Echo Canceller Library, and the Full Duplex Speakerphone Library are all required to build this project. These are all involved when the Feature Phone bean is added

Chapter 4
Feature Phone Application

4.1 Feature Phone Application

The Full Duplex Speakerphone application has been provided for use on a DSP56858EVM board. Modifications can be made to suit the application. The application uses codec drivers and serial port drivers that are part of Processor Expert.

4.2 Application Overview

This application will enable the user to create a fully functional Type 1 and 2 Caller ID Feature Phone. When the Type 2 Caller ID CPE detects a CAS signal, it is required to send back a DTMF acknowledgement signal to the Central Office (CO) to indicate that the CPE is ready to receive FSK data. However, if an extension CPE is found to be off-hook following reception of a CAS signal, the CPE should not send back the acknowledgement signal, as the Caller ID data may become corrupted because the extension CPE is off-hook. The Type 1 and 2 Telephony Features Library supports the necessary timing and logic needed for the application to easily check if an extension phone is off-hook, as specified in SR-3004. The Feature Phone Application polls the *ExtUseCheck* element in the *teldefs_SControl* structure and programs the Si3044 DAA accordingly. When *ExtUseCheck* is equal to 1, the application issues a "force on-hook" command by setting the *MODE* bit in register 18 of the Si3044. The "force on-hook" command disables the off-hook counter that is normally enabled when going back off-hook. By setting the *CALD* bit in register 17 of the Si3044, the Si3044 auto-calibration feature is also disabled at this time. When *ExtUseCheck* is equal to 2, the application monitors the line voltage by reading the *LVCS* bits from register 19 on the DAA. If the voltage is greater than a threshold of 13.75 volts (*LVCS* resolution is 2.75 volts/bit), the application sets *NoExtFound* to one. If the voltage is less than the threshold of 13.75 volts, the application sets *NoExtFound* to zero. When *ExtUseCheck* is equal to 3, the application issues a "force off-hook" command to the DAA by clearing the *MODE* bit in register 18. The Si3044 has some special timing requirements which are not supported by the Type 1 and 2 Telephony Features Library, as they are device-specific. For this reason, the application has its own counter, *extcntr*, which is used for timing. After returning to an off-hook state, the *ONHM* bit in register 5 must be set and left enabled for at least 30ms. After 30ms, the *ONHM* is cleared, allowing normal operation.

The application monitors for ring signal from the DAA by polling the *Ring Detect Signal Positive (RDTP)* bit in register 5. Since the *Ring Detector Full Wave Rectifier Enable (RFWE)* bit is cleared in register 18, only positive ring signals are detected. When the *RDTP* bit is set, the application sets the *cidRingPolarity* element in the

teldefs_SControl structure. Otherwise, the *cidRingPolarity* element is cleared. This information is used by the Type 1 and 2 Telephony Features Library to implement Type 1 Caller ID and to generate ringing tones. The library can also support a ring signal detection using the full wave rectifier; however, this is not shown in this application.

The application communicates with the Si3044 DAA and Si3000 Audio Codec on the TDC daughter card using the Embedded Processor Expert drivers. When the Si3044 and Si3000 have received and transmitted 20 samples, the callback function *Tdc1DaaRXISR()* is called. This function then copies 20 new samples to/from the TDC driver buffers and sets a flag, *SamplesReady*. The application polls the *SamplesReady* flag and when it is set, the *FeaturePhoneAppMain()* function is called, which implements the main loop. The main loop of the Feature Phone Application implements feature phone functionality with full duplex speakerphone and support for both Type 1 and Type 2 Caller ID. Since the Type 1 and 2 Telephony Features Library operates on samples in blocks of five, the main loop for the Feature Phone Application calls this function four times at a rate of 1600 calls per second, while the main loop itself is called 400 times per second. The Generic Echo Canceller Library and Full Duplex Speakerphone Library operate on one sample at a time and are therefore called 8000 times per second.

The Feature Phone Application is controlled using the Embedded Serial Communications Interface peripheral drivers. The application software will support a terminal interface (for example, HyperTerminal) on the RS-232 UART connector on the EVM. This interface should be configured for 8 data bits, no parity bits, and 1 stop bit. The baud rate is 38,400bps by default. If a different baud rate is desired, simply change the definition for *SCI0_BAUD_RATE* in the *appconfig.h* file. Auto-baud support is not available using this software, since the SCI peripheral does not allow this functionality to be implemented. The application reads characters from the SCI peripheral's receive buffer and sends characters to the SCI peripheral's transmit buffer. A simple AT-based command set is implemented in the function *processATComm()*, which is called in the Feature Phone Application main loop. The command set offers a user many functions, including the ability to go on-hook; go off-hook; dial DTMF digits; adjust speakerphone volume; and others. A detailed description of the AT command set is given in **Chapter 2**. Additional functions, such as *processDtmfString()*, *goOnhook*, and *goOffhook,* are called by *processATComm()* in the application main loop to implement a DTMF string dialer and DAA hookswitch control.

Prior to running the main loop, the Feature Phone Application must initialize the TDC daughter card and the SCI peripheral. This is done using the interface functions *sciOpen()* and *open()*. The application creates an instance of the Type 1 and 2 Telephony Features Library by calling the function *Type12Create()*. Prior to this call, several elements of the associated *teldefs_SControl* structure are set to the appropriate values. In a similar fashion, an instance of the Generic Echo Canceller is created by calling *gecEchoCancellerCreate()*, and an instance of the Full Duplex Speakerphone is created by calling *fdspkCreate()*. The *teldefs_SControl* element *Line1Control.gecLengthIndex* is set to 1 to configure the Generic Echo Canceller with a filter length of 128, which corresponds to 16ms. The *teldefs_SControl* element *Line1Control.aecLengthIndex* is set to two to configure the Acoustic Echo Canceller (in the Full Duplex Speakerphone module) with a filter length of 192, which corresponds to 24ms. The elements *Line1Control.totalSupressiondB*, *Line1Control.erlFactor*, and *Line1Control.totalSupression* should be set to values which are determined by first running the diagnostic application software (included with the Full Duplex Speakerphone PE bean) using the same speaker and microphone circuit that will be used for the actual Feature Phone Application. Once these values are calculated, the analog gain on the speaker and microphone should not be adjusted and volume control must be changed by modifying the structure element *Line1Control.volumeGaindB* to the desired volume.

4.3 Application Software

This section includes the software code required to run the Feature Phone Application. **Code Example 4-1** includes the code that is to be run on Processor Expert.

Code Example 4-1. Feature Phone Application Software Code

```
#include "port.h"
#include "fcntl.h"
#include "bsp.h"
#include "tdc1.h"
#include "led.h"


#include "teldefs.h"
#include "gec.h"
#include "cid12.h"
#include "fdspk.h"

struct gec_tsData* pgec1Data;
struct cid_tsData* pcid1Data;
struct fdspk_tsData* pfdspk1Data;
struct teldefs_tsControl Line1Control;
struct teldefs_tsSamples  Line1Samples;
struct teldefs_tsParser ParserControl;

volatile UWord16 audio_input[20];
volatile UWord16 audio_output[20];
volatile UWord16 line_input[20];
volatile UWord16 line_output[20];

volatile UWord16 audio_input2[20];
volatile UWord16 audio_output2[20];
volatile UWord16 line_input2[20];
volatile UWord16 line_output2[20];

volatile UWord16 flag = 0;
volatile UWord16 index = 0;
volatile UWord16 saw = 0;
extern volatile UWord16 Tdc1Daa, Tdc1Codec;
tdc1_sRegister Register;

int layer1_dtmf_temp;
int dtmf_string_length;
int dtmf_string_dialer;
int dtmf_string_waiting;
int layer1_dtmf_request;
int layer1_dtmf_digit ;
int layer1_dtmf_duration;
int dtmfevent_request;
```

```
extern int command_string_indx;
extern char command_string_buf[];
int squelch_on = 0;
int squelch_off = 0;

// Serial port functions.
extern int openSerial();
extern int closeSerial();
extern int sendSerial(char WriteChar);
extern int processATComm();
extern void print_string(char *str);
extern void print_integer(int w);
extern void print_char();
extern void CIDMessageParser(teldefs_sParser*, teldefs_sControl*);

void debug_code();
void goOnhook();
void goOffhook();
void processDtmfString(void);
long int callCntr;
long int lastRingCntr;
int callCntrHalf;


void main (void)
{
       int a,result;
       UWord16 Led;
       Word16 samples;
       Word16 i,j;
       int extcntr;
       int ringLowCntr;
       int SquelchOn;
       int SquelchOnCntr;
       int samp = 0;
       bool OffHook = true;
       tdc1_sRegister Register;
       UWord16 Status;

       UWord16 rate[] = {TDC1_3021_SAMPLE_AT_7200,TDC1_3021_SAMPLE_AT_8000,
              TDC1_3021_SAMPLE_AT_8229,TDC1_3021_SAMPLE_AT_8400,
              TDC1_3021_SAMPLE_AT_9000,TDC1_3021_SAMPLE_AT_9600,
              TDC1_3021_SAMPLE_AT_10286};

       result = 0;
       result = openSerial();
       if(result != 0)
              asm (debughlt);

       /* Initialize Type 1 and 2 Telephony Features Library Control Structure*/
       Line1Control.hookSwitch = 0;
       Line1Control.messageDone=0;
       Line1Control.cidByteReady = 0;
       Line1Control.ExtUseCheck=0;
```

```
        Line1Control.NoExtFound=1;
        Line1Control.FrameErrors=0;
        Line1Control.dtmfRequest=0;
        Line1Control.dtmfDigit=1;
        Line1Control.dtmfComplete=0;
        Line1Control.flashCommand = 0;
        Line1Control.cwdCommand = 0;
        Line1Control.disableRinger=0;

/* Create an instance of Type 1 and 2 Telephony Features */
        pcid1Data = Type12Create(&Line1Control);

/* Initialize Type 1 and 2 Telephony Features Parser Control Structure*/
        ParserControl.FskMessageIndex=0;
        ParserControl.FskParserLength=0;

/* Initialize Generic Echo Canceller Library Control Structure */
        Line1Control.handsFreeLayer1 = 1;
        Line1Control.gecLengthIndex = 1;

/* Create an instance of Generic Echo Canceller */
        pgec1Data = gecEchoCancellerCreate(&Line1Control);

/* Initialize Full Duplex Speakerphone Library Control Structure */
        Line1Control.digitalSwitch1 = 1;
        Line1Control.digitalSwitch2 = 0;
        Line1Control.digitalSwitch3 = 0;
        Line1Control.digitalSwitch4 = 0;
        Line1Control.digitalSwitch5 = 1;
        Line1Control.digitalSwitch6 = 0;
        Line1Control.totalSupression = 1036;    /* obtained from diagnostics   */
        Line1Control.aecLengthIndex = 2;
        Line1Control.silenceOffset = 400;       /* silence threshold for line */
        Line1Control.nesSilenceoffset = 300;    /* silence threshold for audio*/
        Line1Control.erlFactor = 2046;          /* obtained from diagnostics   */
        Line1Control.disableAnalysis = 0;
        Line1Control.disableBothEc = 0;
        Line1Control.totalSupressiondB = 30;
        Line1Control.maxVolumeGaindB = 15;
        Line1Control.volumeGaindB = 15;

/* Create an instance of Full Duplex Speakerphone */
        pfdspk1Data = fdspkCreate(&Line1Control);

/* Set up LEDs */
        Led = open(BSP_DEVICE_NAME_LED_0,0);

/* Set up Tdc1 */

/* open both the DAA and the codec */
        Tdc1Daa = open(BSP_DEVICE_NAME_TDC1_DAA_0, O_RDWR | O_NONBLOCK);
        Tdc1Codec = open(BSP_DEVICE_NAME_TDC1_CODEC_0, O_RDWR | O_NONBLOCK);

/* Set the si3000 codec's speaker drive and line drive to normal operation */
```

```
        Register.Register = 1;
        Register.Data = 0x18;
        ioctl(Tdc1Codec,TDC1_DEVICE_WRITE_REG,(int)&Register);

/* Unmute the si3000 codec's line out */
        while (!ioctl(Tdc1Codec,TDC1_DEVICE_MUTE_LINE_OUT,false));

/* Set the si3000 codec's Rx gain to 0dB and unmute the left and right speaker */
        while(!ioctl (Tdc1Codec,TDC1_DEVICE_MUTE_SPEAKERS,false));
        while(!ioctl (Tdc1Codec,TDC1_DEVICE_MUTE_LINE_IN,false));
        while(!ioctl (Tdc1Codec,TDC1_DEVICE_SET_RX_GAIN, TDC1_CODEC_GAIN_FROM_PERCENT(75)));
        while(!ioctl (Tdc1Codec,TDC1_DEVICE_SET_TX_GAIN, TDC1_CODEC_GAIN_FROM_PERCENT(75)));
        while(!ioctl (Tdc1Daa,TDC1_DEVICE_SET_RX_GAIN, TDC1_DAA_RX_GAIN_FROM_PERCENT(0)));
        while(!ioctl (Tdc1Daa,TDC1_DEVICE_SET_TX_GAIN, TDC1_DAA_TX_GAIN_FROM_PERCENT(0)));


/* The following Enable command actually starts tdc1 data transfer */
#ifdef USE_CALLBACK
        ioctl (Tdc1Daa,TDC1_DEVICE_ENABLE,NULL);
        ioctl (Tdc1Codec,TDC1_DEVICE_ENABLE,NULL);
#endif

        while(!ioctl(Tdc1Daa,TDC1_DEVICE_SET_SAMPLE_RATE,rate[1]));

        /* Initialize the DAA in onhook state */
        goOnhook();

        /* enable charge pump */
        Register.Register = 6;
        Register.Data = 0x80;
        while(!ioctl(Tdc1Daa,TDC1_DEVICE_WRITE_REG,(int)&Register));

        // show Si3044 registers
#if 0
        for( i = 1 ; i <= 19 ; i++){
                Register.Register = i;
                Register.Data = 0;
                while(!ioctl(Tdc1Daa,TDC1_DEVICE_READ_REG,(int)&Register));
                        while(!Register.bDataValid);
                        print_integer(i);
                        print_string("  =  ");
                print_integer((int)Register.Data);
        }
#endif
        // initialize extension in use and other application variables
        extcntr = 0;
        ringLowCntr = 100;
        callCntr = 0;
        lastRingCntr = 9601;
        SquelchOn = 0;
        SquelchOnCntr = 0;

        print_string("Motorola DSPD Feature Phone Application Rev 1.001\n\r");
```

```
/***************************** MAIN LOOP ***************************************/
/* This loop is processed every time we receive 20 samples from the Codecs */
/* Code in this loop runs at 400 Hz */
/*******************************************************************************/

       while(1){

       /* Here we copy the 20 samples to/from the buffers used by */
       /* the DAA and Codec callback functions */

              for( i = 0 ; i < 20 ; i++){
                     line_input[i] = line_input2[i];
                     audio_input[i] = audio_input2[i];
                     line_output2[i] = line_output[i];
                     audio_output2[i] = audio_output[i];
              }

       /* Get Binary Ring Signal from DAA */
       if(Line1Control.hookSwitch == 0 && ParserControl.FskMessageIndex == 0){

              Register.Register = TDC1_3021_DAA_CTRL1;
              Register.Data = 0;
              while(!ioctl(Tdc1Daa,TDC1_DEVICE_READ_REG,(int)&Register));
              while(!Register.bDataValid);
              Status = (Register.Data & TDC1_3021_POS_RING_DETECT);

              if(Status){
                     Line1Control.cidRingPolarity = 1;      /* ring on */
                     ioctl(Led, LED_ON, LED_GREEN2);
              }
              else{
                     Line1Control.cidRingPolarity = 0;      /* ring off */
                     ioctl(Led, LED_OFF, LED_GREEN2);
              }

/* According to the Si3044 data sheet, the SQL2 bits should be set at end of a ring signal
and then cleared before the next ring occurs. Not supported the Type 1 and 2 Telephony
Features Library since it is device specific to the Si3044. */

              if(Line1Control.cidRingPolarity == 0){
                     ringLowCntr++;
                     if(ringLowCntr == 0x7fff) ringLowCntr = 67;
              }
              else{
                     ringLowCntr = 0;
              }
              if(ringLowCntr == 30){     /* Set SQL2 bit at end of ring signal */
                     /* Set SQL2 bit */
                     Register.Register = 16;
                     Register.Data = 0x88;
                     while(!ioctl(Tdc1Daa,TDC1_DEVICE_WRITE_REG,(int)&Register));
                     SquelchOn = 1;
                     SquelchOnCntr = 0;
                     lastRingCntr = 0;   /* clear counter for new ring pulse */
```

```
                    if(SquelchOn){
                          SquelchOnCntr++;
                          if(SquelchOnCntr == 400){ /* 1 second after end of ring
                                                       signal */
                          Register.Register = 16; /* clear SQL2 bit before next ring
                                                       signal */
                          Register.Data = 0x08;
                    while(!ioctl(Tdc1Daa,TDC1_DEVICE_WRITE_REG,(int)&Register));
                          SquelchOn = 0;
                          SquelchOnCntr = 0;
                    }
              }
        }

/* Detect extension in use here by reading the LVCS bits from regsiter 19 on DAA */
        if(Line1Control.ExtUseCheck == 2){
              Register.Register = 19;
              Register.Data = 0;
              while(!ioctl(Tdc1Daa,TDC1_DEVICE_READ_REG,(int)&Register));
              while(!Register.bDataValid);
              Status = (Register.Data & 0x00f8) >> 3;
              //print_integer((int)Status);

              if((int)Status > 5){
                    Line1Control.NoExtFound = 1;
              }
              else{
                    Line1Control.NoExtFound = 0;
              }
        }


/* Since we recieve samples in blocks of 20 and Type12CID() operates on blocks of 5 samples
we process the whole loop 4 times. Code in this loop runs at 1600 Hz. */
        index = 0;
        for( j = 0 ; j < 4 ; j++){

              for( i = 0; i < 5 ; i++){
                    line_output[index]   = Line1Samples.audio[i];
                    audio_output[index]  = Line1Samples.line[i];
                    Line1Samples.line[i] =  line_input[index];
                    Line1Samples.audio[i] = audio_input[index];
                    index++;
              }

/* The following code is used to force the speakerphone to be half duplex for the first 15
seconds after going offhook. While this is not a requirement of the fdspk.lib software, this
is often done in speakerphones for stability. */


        if(Line1Control.hookSwitch){
              callCntr++;
              if((lastRingCntr > 9600) && (callCntr < 24000)){
```

```
                if(Line1Control.disableAnalysis == 0){
                        Line1Control.disableAnalysis = 1;
                        callCntrHalf = 1;
                        //print_string("Half\r");
                }
        }
        else if(callCntr == 24000){             /* 15 seconds */
                if(callCntrHalf){
                        Line1Control.disableAnalysis = 0;
                        //print_string("Full\r");
                        callCntrHalf = 0;
                }
        }
}
else lastRingCntr++;

/* Call Gec and Full Duplex Speakerphone */
/* Code in this loop runs at 8000 Hz */
for(i=0;i<5;i++) {
        if (Line1Control.hookSwitch == 1){
                if (Line1Control.handsFreeLayer1 == 1)
                fdspkState(pfdspk1Data,&Line1Control,&Line1Samples);
                }
                gecEchoCanceller(pgec1Data,&Line1Control,&Line1Samples);
        }

/* If Call Progess Tone Detection is required, input the echo cancelled sample,
Line1Samples.leccid[i] to the necessary Freescale Processor Expert Embedded Library. This
functionality is not supported in this application. */


        /* Call Type 1/2 Caller ID Module */
        Type12CID(pcid1Data,&Line1Control,&Line1Samples);

        /* Do extension in use check if necessary */
        if(Line1Control.ExtUseCheck == 1){
                // Go onhook
                Line1Control.NoExtFound = 1;
                extcntr = 0;

                /* Disable auto calibration using CALD bit*/
                Register.Register = 17;
                Register.Data = 0x20;
                while(!ioctl(Tdc1Daa,TDC1_DEVICE_WRITE_REG,(int)&Register));

                 /* set MODE equal to one - force onhook */
                Register.Register = 18;
                Register.Data = 0x04;
                while(!ioctl(Tdc1Daa,TDC1_DEVICE_WRITE_REG,(int)&Register));
                }
                else if(Line1Control.ExtUseCheck == 3){
                        // Go offhook
                        extcntr++;   /* counter for Si3044 specfic timing */
                        if(extcntr == 1){
```

```
                        /* clear MODE to zero - force back offhook */
                        Register.Register = 18;
                        Register.Data = 0x00;
                  while(!ioctl(Tdc1Daa,TDC1_DEVICE_WRITE_REG,(int)&Register));
                  }
                  else if(extcntr == 2){
                        /* set ONHM equal to one */
                        Register.Register = 05;
                        Register.Data = 0x09;

      while(!ioctl(Tdc1Daa,TDC1_DEVICE_WRITE_REG,(int)&Register));
                  }
                  else if(extcntr == 48){
                        /* clear ONHM back to zero */
                        Register.Register = 05;
                        Register.Data = 0x01;
                            while(!ioctl(Tdc1Daa,TDC1_DEVICE_WRITE_REG,(int)&Register));
                  }
                  else if(extcntr == 50){
                        /* Enable auto calibration using CALD bit*/
                        Register.Register = 17;
                        Register.Data = 0x00;

      while(!ioctl(Tdc1Daa,TDC1_DEVICE_WRITE_REG,(int)&Register));
                  }
                  else if(extcntr > 50){
                        Line1Control.ExtUseCheck=0;
                        if(Line1Control.NoExtFound == 0) print_string("XZUSE=1\r");
                        else print_string("XZCAS=1\r");

                  }

            }

            /* Type 1 and 2 Telephony Parser Library */
            CIDMessageParser(&ParserControl, &Line1Control);

/* If Caller ID data is present,transmit the characters on the Serial Port to a Terminal. */
            if(ParserControl.FskParserLength != 0){
                  for( i = 0 ; i < ParserControl.FskParserLength ; i++)
                        sendSerial(ParserControl.FskParserBuffer[i]);
                        ParserControl.FskParserLength=0;
                  }

            /* Do flash command for Call Waiting Deluxe if necessary */
                  if(Line1Control.flashCommand){
                        Line1Control.flashCommand = 0;
                        if(Line1Control.flashPolarity == 1) goOffhook();
                        else goOnhook();
                  }

                  /* Process any pending AT command characters if necessary */
                  processATComm();
                  /* Process the DTMF string dialler if necessary */
```

```
                processDtmfString();


                }      /* end of 20 sample loop */


            while(flag == 0){ /* Wait until another 20 samples are ready */
                  asm( nop );
            }
            flag = 0;

            }
            /* close serial port */
            closeSerial();
            close(Led);
            close(Tdc1Codec);
            close(Tdc1Daa);

}           /* End of Main Loop */


/*********************************************************/
/* Function: goOnhook() */
/* Inputs:   none */
/* Outputs:  none */
/* Description:This functions places the DAA on hook and */
/* enables onhook line monitoring for Caller ID */
/*********************************************************/

void goOnhook(){
      bool OffHook;

      /* set MODE bit */
      Register.Register = TDC1_3021_INTRNL_CTRL3;
      Register.Data = TDC1_3021_MODE_SET;
      while(!ioctl(Tdc1Daa,TDC1_DEVICE_WRITE_REG,(int)&Register));
            /* set ONHM bit */
            Register.Register = TDC1_3021_DAA_CTRL1;
            Register.Data = TDC1_3021_MONITOR_ON_HOOK_MODE;
      while(!ioctl(Tdc1Daa,TDC1_DEVICE_WRITE_REG,(int)&Register));

            OffHook = false;
      while(!ioctl(Tdc1Daa,TDC1_DEVICE_OFF_HOOK,OffHook));

}

/*********************************************************/
/* Function: goOffhook() */
/* Inputs:   none */
/* Outputs:  none */
/* Description: This functions places the DAA on hook and */
/* disables onhook line monitoring for Caller ID */
/*********************************************************/
```

```
void goOffhook(){
       bool OffHook;

       /* clear MODE bit */
       Register.Register = TDC1_3021_INTRNL_CTRL3;
       Register.Data = TDC1_3021_MODE_CLEAR;
       while(!ioctl(Tdc1Daa,TDC1_DEVICE_WRITE_REG,(int)&Register));
       /* clear ONHM bit */
       Register.Register = TDC1_3021_DAA_CTRL1;
       Register.Data =  TDC1_3021_NORMAL_ON_HOOK_MODE;
       while(!ioctl(Tdc1Daa,TDC1_DEVICE_WRITE_REG,(int)&Register));

       OffHook = true;
       while(!ioctl(Tdc1Daa,TDC1_DEVICE_OFF_HOOK,OffHook));

}

/*************************************************************/
/* Function: processDtmfString() */
/* Inputs:   none */
/* Outputs:  none */
/* Description: This functions dials a string of DTMF digits */
/* which are buffered by the AT command set parser. */
/*************************************************************/

void processDtmfString(void){

       if(dtmf_string_dialer == 1){
              if(dtmf_string_waiting == 0){
                     dtmf_string_waiting = 1;
                     layer1_dtmf_request = 1;
                     Line1Control.dtmfRequest=1;
                     Line1Control.dtmfComplete = 0;
                     layer1_dtmf_temp = command_string_buf[command_string_indx];
                     if(layer1_dtmf_temp >= 0x30 && layer1_dtmf_temp <= 0x39)
                     layer1_dtmf_digit = command_string_buf[command_string_indx]-0x30;
                     else if(layer1_dtmf_temp == 0x41) layer1_dtmf_digit = 0x0a;
                     else if(layer1_dtmf_temp == 0x42) layer1_dtmf_digit = 0x0b;
                     else if(layer1_dtmf_temp == 0x43) layer1_dtmf_digit = 0x0c;
                     else if(layer1_dtmf_temp == 0x44) layer1_dtmf_digit = 0x0d;
                     else if(layer1_dtmf_temp == 0x2a) layer1_dtmf_digit = 0x0e;
                     else if(layer1_dtmf_temp == 0x23) layer1_dtmf_digit = 0x0f;
                     else{
                     }
                     Line1Control.dtmfDigit = layer1_dtmf_digit;

                     command_string_indx++;
                     if(command_string_indx == dtmf_string_length){
                            dtmf_string_dialer = 0;
                            dtmf_string_waiting = 0;
                            print_char(0x0d);
                            print_char('O');
                            print_char('K');
                            print_char(0x0d);
```

```
                }
            }
            else if(Line1Control.dtmfComplete == 1){
                dtmf_string_waiting = 0;
                dtmfevent_request = 0;
                Line1Control.dtmfComplete = 0;
            }
        }
}
```

Chapter 5
License

5.1 Limited Use License Agreement

This software is available under a separate license agreement from Freescale Incorporated. Licensing information can be obtained from your Freescale sales representative or authorized distributor.

For additional product information, see ***freescale.com***.

INDEX

A

Acoustic Echo Cancellation System 1-6
ADC x
AGC x
American Standard Code for Information Interchange
    ASCII xi
Analog-to-Digital Converter
    ADC x
ANSI/TIA/EIA-470B, Telephony Instruments with
    Loop Signaling xii
ANSI/TIA/EIA-716, Telecommunications Telphone
    Terminal Equipment - Type 1Caller Identity
    Equipment Performance Requirements xii
ANSI/TIA/EIA-777, Telecommunications Telephone
    Terminal Equipment - Type 2 Caller Identity
    Equipment Performance Requirements xii
API xi
Application Programming Interface
    API xi
ASCII xi
AT Command Set Parser 1-5
Automatic Gain Control
    AGC x

B

Bandpass Filter
    BPF xi
Bell 202 Modem Receiver 1-5
BPF xi

C

Call Qualifier
    CQ xi
Caller ID
    CID xi
CID xi
CPE xi
CQ xi
Customer Premises Equipment
    CPE xi

D

DAA xi
Data Access Arrangement
    DAA xi
Digital Signal Processor
    DSP xi
DSP xi
DSP Hardware 1-2
DSP56800E Reference Manual xi
DSP5685x User's Manual xi
DTMF xi
DTMF String Dialer 1-5
Dual Tone Multiple Frequency
    DTMF xi

E

Echo Interface 1-5

F

Frequency Shift Keying
    FSK xi
FSK xi
Full Duplex Speakerphone Library 1-1

G

General-Purpose Echo Cancellation System 1-5
GR-1188-CORE, LSSGR CLASS Feature
    Calling Name Delivery Generic Requirements xi
GR-1401-CORE, LSSGR CLASS Feature
    Visual Message Waiting Indicator Generic
        Requirements xi
GR-30-CORE, LSSGR
    Voiceband Data Transmission Interface xi
GR-31-CORE, LSSGR CLASS Feature
    Calling Number Delivery xi
GR-416-CORE, CLASSSM Feature Call Waiting
    Deluxe xii
GR-575-CORE, LSSGR CLASSSM Feature Calling
    Identity Delivery on Call Waiting xii

I

IDE xi
Integrated Development Environment
    IDE xi
ITU-T Recommendation V.23 xii

L

Logarithmic Suppression Controller 1-6
Low Pass Filter
    LPF xi
LPF xi

M

Main Sample Processing Loop 1-5
MDMF xi
MIC xi
Microphone
    MIC xi
Million Instructions Per Second
    MIPS xi
MIPS xi

Multiple Data Message Format
MDMF xi

O

OnCE xi
On-Chip Emulation
OnCE xi

P

PC xi
PCM xi
Personal Computer
PC xi
PSTN xi
Public Switched Telephone Network
PSTN xi
Pulse Code Modulation
PCM xi

S

SDMF xi
Single Data Message Format
SDMF xi
Software Functional Requirements 1-4
Source
SRC xi
SR-3004, Testing Guidelines for Analog Type 1, 2, and
3 CPE xi
SRC xi
SR-NWT-002024, Customer Premises Equipment
Compatibility Comsiderations for the
SPCS-to-CPE Data Transmission Interface xii
SR-TSV-002476, CPE Compatibility Considerations
For The Voiceband Data Transmission Interface xii

T

Terminal Interface 1-5

U

User Interface 1-2

V

Visual Message Waiting Indicator
VMWI xi
VMWI xi

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064, Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

FP56858UG
Rev. 0
09/2005