



dsPIC30F
Soft Modem Library
User's Guide

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. No representation or warranty is given and no liability is assumed by Microchip Technology Incorporated with respect to the accuracy or use of such information, or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Microchip's products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights.

Trademarks

The Microchip name and logo, the Microchip logo, Accuron, dsPIC, KEELOQ, microID, MPLAB, PIC, PICmicro, PICSTART, PRO MATE, PowerSmart, rPIC, and SmartShunt are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.


Amplab, FilterLab, MXDEV, MXLAB, PICMASTER, SEEVAL, SmartSensor and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Analog-for-the-Digital Age, Application Maestro, dsPICDEM, dsPICDEM.net, dsPICworks, ECAN, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, Migratable Memory, MPASM, MPLIB, MPLINK, MPSIM, PICKit, PICDEM, PICDEM.net, PICLAB, PICTail, PowerCal, PowerInfo, PowerMate, PowerTool, rLAB, rPICDEM, Select Mode, Smart Serial, SmartTel and Total Endurance are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

All other trademarks mentioned herein are property of their respective companies.

© 2004, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

 Printed on recycled paper.

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
== ISO/TS 16949:2002 ==

Microchip received ISO/TS-16949:2002 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona and Mountain View, California in October 2003. The Company's quality system processes and procedures are for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

Table of Contents

Preface	1
Chapter 1. Introduction	
1.1 Introduction	7
1.1 Highlights	7
1.2 Library Overview	7
1.3 Modem Characteristics	9
1.4 Demonstration Programs	10
Chapter 2. Getting Started	
2.1 Introduction	11
2.2 Highlights	11
2.3 Implementing the Soft Modem	11
2.4 Soft Modem Modulation Options	13
2.5 Setting Up Compile Time Options For Your Application	17
2.6 Configuring the Analog Front End	19
2.7 DTMF Generation and Detection Modules	24
2.8 Resource Usage	25
Chapter 3. Soft Modem API Specifications	
3.1 Introduction	27
3.1 Highlights	27
3.2 Overview of Soft Modem Subsystem	27
3.3 Soft Modem Data Pump API Specification	29
3.4 V.42 Protocol Layer API Specifications	48
3.5 AT Command Layer API Specification	63
3.6 Embedded Soft Modem API	71
Chapter 4. DTMF API	
4.1 Introduction	75
4.2 Highlights	75
4.3 Overview	75
4.4 DTMF API Specifications	76
Chapter 5. Soft Modem Demonstrations	
5.1 Introduction	79
5.2 Highlights	79
5.3 Overview and Purpose	79
5.4 Demo Projects	81
5.5 AT Command Modem Demo	93
5.6 Embedded Modem Demo	97

dsPIC30F Soft Modem Library User's Guide

5.7 DTMF Loopback Demo	100
5.8 DTMF PSTN Phone Line Demo	104
Chapter 6. Troubleshooting	
6.1 Introduction	107
6.2 Highlights	107
6.3 Standard Reference Modems	108
6.4 ITU-T V.8 Support	109
6.5 Data Flow Control	109
6.6 V.23 Connection For US Robotics Model 5686E Modem	109
6.7 V.21 Connection for US Robotics Model 5686E Modem	110
6.8 Response Messages for Successful Modem Connections	111
6.9 Response Message for Unsuccessful Modem Connection	113
6.10 Regulatory Compliance Reference Information	114
Appendix A. ITU-T Specifications	
A.1 ITU-T Specifications	115
Appendix B. AT Command Set	
B.1 Introduction	117
B.2 AT Commands	117
Appendix C. Drivers	
C.1 Introduction	121
C.2 DAA/AFE Driver Functions	121
Index	123
Worldwide Sales and Service	126

Preface

INTRODUCTION

This user's guide supports the dsPIC30F Soft Modem Library, which is composed of ITU-T compliant algorithms for V.21/Bell 103, V.22, V.22bis, V.23, V.32, V.32bis and V.42 recommendations. This chapter previews the contents of the manual, tells you how to obtain valuable customer support and recommends useful reference information.

HIGHLIGHTS

Items discussed in this chapter are:

- About This Guide
- Warranty Registration
- Recommended Reading
- The Microchip Web Site
- Development Systems Customer Notification Service
- Customer Support

ABOUT THIS GUIDE

This user's guide describes how to use the dsPIC30F Soft Modem Library. The document is organized as follows:

- **Chapter 1. Introduction** – This chapter describes the basics of the dsPIC30F Soft Modem Library and presents the key features and performance metrics.
- **Chapter 2. Getting Started** – This chapter describes how to implement the ready-to-use soft modem libraries, how to customize modem configurations for user applications, and how to rebuild the libraries. Information is provided for setting options for the soft modem. This chapter also covers the information on building the DTMF generation and detection modules.
- **Chapter 3. Soft Modem API Specifications** – This chapter covers the interface details for the data pump, error control and command layers of the dsPIC30F Soft Modem Library.
- **Chapter 4. DTMF API** – This chapter provides interface details for the DTMF generation and detection modules.
- **Chapter 5. Soft Modem Demonstrations** – This chapter provides “hands-on” experience with four sample applications that demonstrate the major functionality of the dsPIC30F Soft Modem Library.
- **Chapter 6. Troubleshooting** – This chapter deals with specific fallback, data flow and modem connection issues for common reference modems.
- **Appendix A. ITU-T Specifications** – This appendix provides ITU-T reference information.
- **Appendix B. AT Command Set** – This appendix provides information on the AT command set for the dsPIC30F Soft Modem Library.
- **Appendix C. Drivers** – This appendix provides information related to drivers used with the dsPIC30F Soft Modem Library.

dsPIC30F Soft Modem Library User's Guide

Conventions Used in This Guide

This manual uses the following documentation conventions:

TABLE 1: DOCUMENTATION CONVENTIONS

Description	Represents	Examples
Code (Courier font):		
Plain characters	Sample code Filenames and paths	<code>#define START</code> <code>c:\autoexec.bat</code>
Square brackets []	Optional arguments	<code>pic30-as [main.s]</code>
Curly brackets and pipe character: { }	Choice of mutually exclusive arguments; an OR selection	<code>errorlevel {0 1}</code>
Ellipses...	Used to imply (but not show) additional text that is not relevant to the example	<code>list</code> <code>["list_option...",</code> <code>"list_option"]</code>
<i>0xnnnn</i>	A hexadecimal number where <i>n</i> is a hexadecimal digit	<code>0xFFFF</code> , <code>0x007A</code>
Italic characters	A variable argument; it can be either a type of data (in lower case characters) or a specific example (in upper case characters)	<code>pic30-gcc filename</code>
Interface (Arial font):		
Underlined, italic text with right arrow	A menu selection from the menu bar	<u><i>File > Save</i></u>
Bold characters	A window or dialog button to click	OK, Cancel
Characters in angle brackets < >	A key on the keyboard	<Tab>, <Ctrl-C>
Documents (Arial font):		
Italic characters	Referenced books	<i>MPLAB IDE User's Guide</i>

Documentation Updates

All documentation becomes dated, and this document is no exception. Microchip tools are constantly evolving to meet customer needs, some actual dialogs and/or tool descriptions may differ from those in this document. Please refer to our web site to obtain the latest documentation available.

Documentation Numbering Conventions

Documents are numbered with a "DS" number. The number is located on the bottom of each page, in front of the page number. The numbering convention for the DS Number is DSXXXXXA, where:

XXXXX = The document number.

A = The revision level of the document.

WARRANTY REGISTRATION

Please complete the enclosed Warranty Registration Card and mail it promptly. Sending in the Warranty Registration Card entitles user's to receive new product updates. Interim software releases are available on the Microchip web site.

RECOMMENDED READING

This user's guide describes how to use the dsPIC30F Soft Modem. Other useful documents include:

dsPIC30F Family Reference Manual (DS70046)

Consult this document for detailed information on dsPIC30F device operation. This reference manual explains the operation of the dsPIC30F MCU family architecture and peripheral modules but does not cover the specifics of each device. Refer to the appropriate device data sheet for device-specific information.

dsPIC30F Programmer's Reference Manual (DS70030)

This manual is a software developer's reference for the dsPIC30F 16-bit MCU family of devices. It describes the instruction set in detail and also provides general information to assist in developing software for the dsPIC30F MCU family.

dsPIC30F Family Overview (DS70043)

This document provides an overview of the functionality of the dsPIC[®] product family. It helps determine how the dsPIC 16-bit Digital Signal Controller Family fits a specific product application. This document is a supplement to the *dsPIC30F Family Reference Manual*.

dsPIC30F Data Sheet, Motor Control and Power Conversion Family (DS70082)

Consult this document for detailed information on the dsPIC30F Motor Control and Power Conversion devices. Reference information found in this data sheet includes:

- Device memory map
- Device pinout and packaging details
- Device electrical specifications
- List of peripherals included on the device

dsPIC30F Data Sheet, General Purpose and Sensor Families (DS70083)

Consult this document for detailed information on the dsPIC30F Sensor and General Purpose devices. Reference information found in this data sheet includes:

- Device memory map
- Device pinout and packaging details
- Device electrical specifications
- List of peripherals included on the device

dsPIC30F5011, dsPIC30F5013 Data Sheet, High Performance Digital Signal Controllers (DS70116)

This data sheet contains specific information for the dsPIC30F5011/5013 Digital Signal Controller (DSC) devices.

dsPIC30F6011, dsPIC30F6012, dsPIC30F6013, dsPIC30F6014 Data Sheet, High Performance Digital Signal Controllers (DS70117)

This data sheet contains specific information for the dsPIC30F6011/6012/6013/6014 Digital Signal Controller (DSC) devices.

MPLAB[®] ASM30, MPLAB LINK30 and Utilities User's Guide (DS51317)

This document details Microchip Technology's language tools for dsPIC devices based on GNU technology. The language tools discussed are:

- MPLAB ASM30 Assembler
- MPLAB LINK30 Linker
- MPLAB LIB30 Archiver/Librarian
- Other Utilities

dsPIC30F Soft Modem Library User's Guide

MPLAB C30 C Compiler User's Guide and Libraries (DS51284)

This document details the use of Microchip's MPLAB C30 C Compiler for dsPIC devices to develop an application. MPLAB C30 is a GNU-based language tool, based on source code from the Free Software Foundation (FSF). For more information about the FSF, see www.fsf.org.

Other GNU language tools available from Microchip are:

- MPLAB ASM30 Assembler
- MPLAB LINK30 Linker
- MPLAB LIB30 Librarian/Archiver

MPLAB IDE Simulator, Editor User's Guide (DS51025)

Consult this document for more information pertaining to the installation and implementation of the MPLAB Integrated Development Environment (IDE) Software.

To obtain any of these documents, contact the nearest Microchip sales location (see back page) or visit the Microchip web site at www.microchip.com.

Microchip Web Site

The Microchip web site (www.microchip.com) contains a wealth of documentation. Individual data sheets, application notes, tutorials and user's guides are all available for easy download. All documentation is in Adobe® Acrobat® (pdf) format.

THE MICROCHIP WEB SITE

Microchip provides on-line support on the Microchip World Wide Web (WWW) site. The web site is used by Microchip as a means to make files and information easily available to customers. To view the site, you must have access to the Internet and a web browser, such as, Netscape Navigator® or Microsoft® Internet Explorer.

The Microchip web site is available by using your favorite Internet browser to reach:

www.microchip.com

The web site provides a variety of services. Users may download files for the latest development tools, data sheets, application notes, user's guides, articles and sample programs. A variety of information specific to the business of Microchip is also available, including listings of Microchip sales offices, distributors and factory representatives.

Technical Support

- Frequently Asked Questions (FAQ)
- On-line Discussion Groups – conferences for products, development systems, technical information and more
- Microchip Consultant Program Member Listing
- Links to other useful web sites related to Microchip products

Engineer's Toolbox

- Design Tips
- Device Errata

Other Available Information

- Latest Microchip Press Releases
- Listing of seminars and events
- Job Postings

DEVELOPMENT SYSTEMS CUSTOMER NOTIFICATION SERVICE

Microchip started the customer notification service to help our customers keep current on Microchip products with the least amount of effort. Once you subscribe, you will receive e-mail notification whenever we change, update, revise or have errata related to your specified product family or development tool of interest.

Go to the Microchip web site at (www.microchip.com) and click on Customer Change Notification. Follow the instructions to register.

The Development Systems product group categories are:

- Compilers
- Emulators
- In-Circuit Debuggers
- MPLAB IDE
- Programmers

Here is a description of these categories:

Compilers – The latest information on Microchip C compilers and other language tools. These include the MPLAB C17, MPLAB C18 and MPLAB C30 C compilers; MPASM™ and MPLAB ASM30 assemblers; MPLINK™ and MPLAB LINK30 object linkers; MPLIB™ and MPLAB LIB30 object librarians.

Emulators – The latest information on Microchip in-circuit emulators. This includes the MPLAB ICE 2000 and MPLAB ICE 4000.

In-Circuit Debuggers – The latest information on the Microchip in-circuit debugger, MPLAB ICD 2.

MPLAB IDE – The latest information on Microchip MPLAB IDE, the Windows® Integrated Development Environment for development systems tools. This list is focused on the MPLAB IDE, MPLAB SIM and MPLAB SIM30 simulators, MPLAB IDE Project Manager and general editing and debugging features.

Programmers – The latest information on Microchip device programmers. These include the MPLAB PM3 and PRO MATE® II device programmers and PICSTART® Plus development programmer.

dsPIC30F Soft Modem Library User's Guide

CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Corporate Applications Engineer (CAE)
- Hotline

Customers should call their distributor, representative or field application engineer (FAE) for support. Local sales offices are also available to help customers. See the back cover for a list of sales offices and locations.

Corporate Applications Engineers (CAEs) may be contacted at (480) 792-7627.

In addition, there is a Systems Information and Upgrade Line. This line provides system users a list of the latest versions of all of Microchip's development systems software products. Plus, this line provides information on how customers can receive any currently available upgrade kits.

The Hotline Numbers are:

1-800-755-2345 for U.S. and most of Canada.

1-480-792-7302 for the rest of the world.

Chapter 1. Introduction

1.1 INTRODUCTION

The dsPIC30F Soft Modem Library is a collection of algorithms for ITU-T compliant V.21/Bell 103, V.22, V.22bis, V.23, V.32, V.32bis modems and V.42 error correction recommendations. This chapter introduces the dsPIC30F Soft Modem Library and identifies specific capabilities available to applications that run on the dsPIC30F Digital Signal Controller (DSC).

1.1 HIGHLIGHTS

Information in this chapter includes:

- Library Overview
- Modem Characteristics
- Demonstration Programs

1.2 LIBRARY OVERVIEW

The dsPIC30F Soft Modem Library offers PSTN communication capability for the dsPIC30F product family. The library is packaged in two basic software configurations:

- A “free” package offers V.22bis/V.22, V/23 and V.21/Bell 103 algorithms with full source code
- A “for purchase” package offers V.32bis/V.32, V.22bis/V.22, V/23 and V.21/Bell 103 algorithms

Both library versions include V.42 error-correcting protocols for the modems.

The modems are single channel implementations. Both libraries are supported with fallback data rates down to V.21. Each library is provided with an archive that contains all the object code modules required to link to your user application. Hardware component drivers such as UART and Data Converter Interface for Data Access Arrangement (DAA) and Analog Front End (AFE) I/O are provided in assembly source code for linking with the user application.

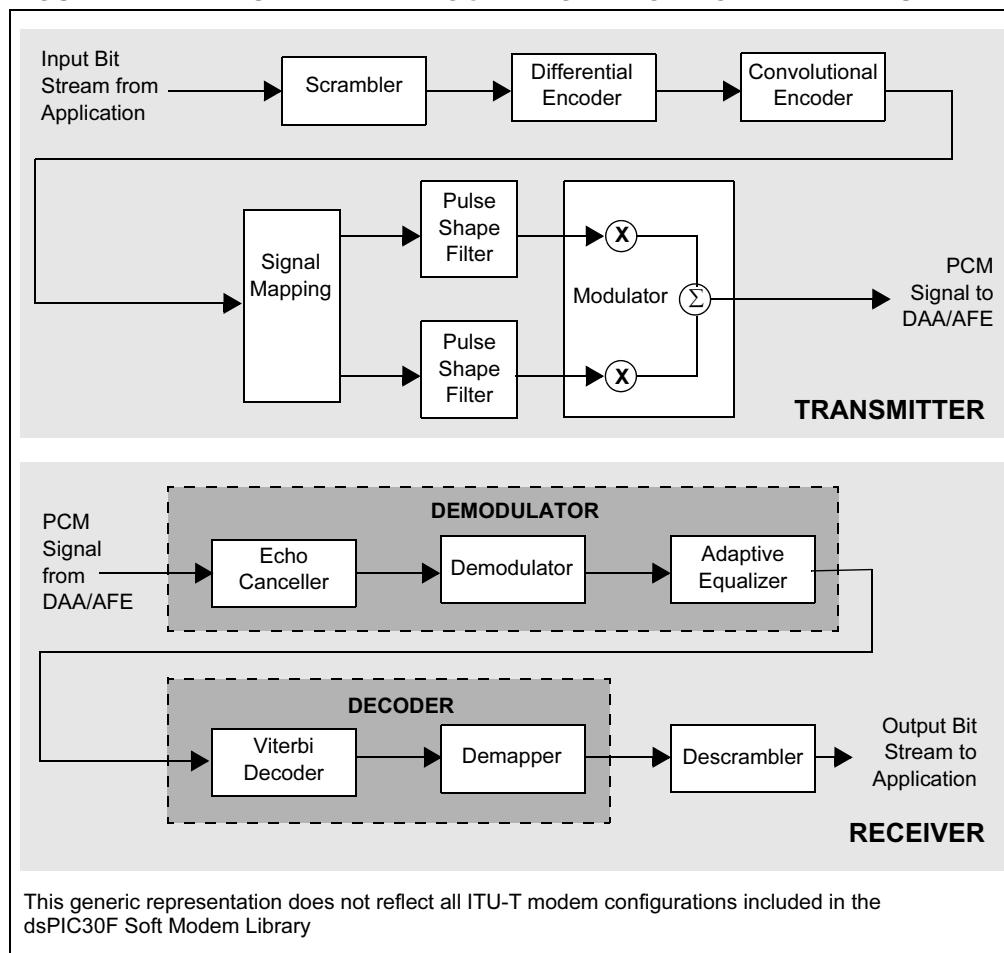
Soft modem functionality is implemented in three functional layers:

- Data Pump
- V.42 Error Control
- AT Command

The Data Pump (DP) layer is the core of the modem. Generally speaking, the major functions performed by the data pump are modulation/demodulation, encoding/decoding, scrambling/descrambling, equalizing, echo cancellation, timing recovery and carrier recovery, as illustrated in Figure 1-1. The dsPIC30F Soft Modem Library includes ITU-T compliant algorithms for all of these modem functions while the dsPIC30F Digital Signal Processor offers a single-chip solution to including modem implementation in your user application.

dsPIC30F Soft Modem Library User's Guide

FIGURE 1-1: SIMPLIFIED BLOCK DIAGRAM OF MODEM DATA PUMP



The V.42 Error-Control layer is a High Level Data Link Control (HDLC) protocol referred to as Link Access Procedure for Modems (LAPM) and defines error-correcting protocols for the modem algorithms. The V.42 Error-Control layer adheres to ITU-T recommendations.

The AT command layer provides you with selected standard AT commands to configure the data pump and V.42 layers. Appendix B lists the AT Commands supported by the dsPIC30F Soft Modem Library.

All data pump modulations are developed in MPLAB ASM30 assembly code yielding optimal code size and execution time. The AT, V.42 and Data Pump APIs are based on MPLAB C30 'C' language.

1.2.1 Soft Modem Library Contents

The free version of the dsPIC30F Soft Modem Library includes:

- V.22bis/V.22 Algorithms
- V.23 Algorithm
- V.21/Bell 103 Algorithms
- V.42 Error Correction
- Data Pump and V.42 API
- AT Command Set

The “for purchase” version of the dsPIC30F Soft Modem Library is offered in object code and includes:

- V.32bis/V.32 Algorithms
- V.22bis/V.22 Algorithms
- V.23 Algorithm
- V.21/Bell 103 Algorithms
- V.42 Error Correction
- Data Pump and V.42 API
- AT Command Set

1.3 MODEM CHARACTERISTICS

The library provides several levels of data throughput and modulation techniques.

- V.21/Bell 103 and V.23 are Frequency Shift Keying (FSK) modems.
- V.32bis/V.32 is a Quadrature Amplitude Modulated (QAM) and Trellis Coded Modulated (TCM) modem.
- V.22bis is a QAM modem.
- V.22 is a Phase Shift Keyed (PSK) modem.
- V.21, V.22, V.22bis, V.32 and V.32bis are all 2-wire, Full Duplex modems.
- V.23 is Full-Duplex when it operates with a 75 bps backwards channel.
- V.22bis includes fallback to V.22, V.23 and V.21 standards.
- V.32bis includes fallback to V.22bis, V.22, V.23 and V.21 standards.

Table 1-1 presents the key parameters for these modems.

dsPIC30F Soft Modem Library User's Guide

TABLE 1-1: SOFT MODEM CHARACTERISTICS

Algorithm ⁽¹⁾	Performance			Program Memory ⁽²⁾ (Kbytes)	Data Memory ⁽²⁾ (Kbytes)	MIPS
	Data Rate (kbps)	Half/Full Duplex	Data Modulation			
V.21/Bell 103	0.3	Full	FSK	13	1.0	4.5
V.22/V.22bis	1.2	Full	PSK/QAM	22	1.7	7.0
	2.4					
V.23	1.2	Half	FSK	15	1.0	4.5
	0.6					
V.32	9.6	Full	QAM/TCM	31	3.2	12
	4.8					
V.32bis	14.4	Full	QAM/TCM	36	3.6	15
	12.0					
	9.6					
	7.2					
	4.8					
V.42	n/a			14	2.0	1.5
DP+V.42 API	n/a			7	1.2	–
AT Command Set	n/a			8	0.15	–

Note 1: Data pump modules, V.21/Bell 103, V.22, V.22bis, V.23, V.32 and V.32bis are implemented in Assembly language. V.42, Data Pump and AT Command APIs are implemented in C language.

2: The program/data memory usage for the V-series data pumps is NOT cumulative, due to the sharing of components internally.

3: Memory size does not account for application which combines data pump, V.42 and AT commands (if required)

1.3.1 Typical Applications

Typical soft modem applications include

- POS terminals
- Set top boxes
- Drop boxes
- Fire panels
- Internet-enabled home security systems
- Internet-connected power, gas and water meters
- Internet-connected vending machines
- Smart appliances
- Industrial monitoring

1.4 DEMONSTRATION PROGRAMS

The dsPIC30F Soft Modem Library is supplied with sample application programs and source code to help you jump start your own solutions. Sample applications that demonstrate.

- AT Command controlled V.22bis soft modem operation
- Embedded V.22bis soft modem operation
- DTMF generation and detection in a loopback connection
- DTMF detection over a PSTN analog phone line

Chapter 2. Getting Started

2.1 INTRODUCTION

This chapter provides basic information to help you implement the soft modem library and DTMF generation and detection modules. A careful review of this section is essential to understanding how to select the correct modem for your application and configure the soft modem build options. The build options require you to select both software and hardware features, which must be properly configured to ensure a reliable connection and proper soft modem operation.

2.2 HIGHLIGHTS

Topics covered in this chapter include:

- Implementing the Soft Modem
- Soft Modem Modulation Options
- Setting Up Compile Time Options For Your Application
- Configuring the Analog Front End
- DTMF Generation and Detection Modules

2.3 IMPLEMENTING THE SOFT MODEM

The dsPIC30F Soft Modem Library provides the utmost flexibility in choosing the specific modem type for your application. To maximize capability you might select the V.22bis (free) or V.32bis (purchased) Integrated Soft Modem, which include both fallback and error correction. To minimize program size you might prefer a single modem type, with or without fallback and error correction. Whatever your need, the library probably has an archive that you can implement “off-the-shelf.”

The ready-to-use configurations included in the dsPIC30F Soft Modem Library are listed in Table 2-1. To implement one of these configurations, select the corresponding archive and support files and add them to your user application.

The soft modem libraries are built around “small code” and “large data” memory models. These memory models place the fewest restrictions on your applications when you incorporate one or more of the modem configurations. *MPLAB C30 C Compiler User's Guide* (DS51284) provides detailed information about these memory models.

The soft modem modules are compiled and linked using Microchip's MPLAB C30 Compiler, MPLAB ASM30 Assembler and MPLAB LINK30 Linker. The integrated development environment (MPLAB IDE) is used to build the soft modem modules.

The soft modem library uses both assembly and C code modules. The assembly modules use both .nbss (uninitialized general near data memory sections) and .bss (uninitialized general data memory sections) data sections. The data variables and structures implemented in the C code modules are built with large data memory models.

dsPIC30F Soft Modem Library User's Guide

TABLE 2-1: LIBRARY SOFT MODEM CONFIGURATIONS

Soft Modem Type	Archive File	Support Files ⁽¹⁾⁽²⁾
V.22bis Integrated V.22bis with fallback to V.23 and V.21/Bell 103 modulations. Includes V.42 error correction	dsPICSM_V22bisE.a	Source Files init.s init_uart.s dci_init.s Si3021_init.s device_config.s dci_isr.s isr_uart1.s traps.s delay.c LCD.c 'C' Files API.C and AT_cmds.C or SMUA_OL.C and SMUA_IL.C ⁽³⁾ Inc Files Si3021config.inc, device_Fcy.inc, dpconfig.inc h Files options.h
V.22bis V.22bis with fallback to V.23 and V.21/Bell 103 modulations. No error correction.	dsPICSM_V22bis.a	
V.22bis Stand Alone V.22bis modulation only. Includes V.42 error correction.	dsPICSM_V22bisSE.a	
V.23 V.23 with fallback to V.21/Bell 103 modulations. No error correction)	dsPICSM_V23.a	
V.32bis Integrated V.32bis with fallback to V.32, V.22bis, V.23 and V.21/Bell 103 modulations. Includes V.42 error correction	dsPICSM_V32bis.a	
V.32 V.32 with fallback to V.22bis, V.23 and V.21/Bell 103 modulations. No error correction.	dsPICSM_V32.a	
V.32bis, V.32 Stand Alone V.32bis with fallback to V.32. Includes V.42 error correction.	dsPICSM_V32bisSE.a	
V.32 Stand Alone V.32 modulation only. Includes V.42 error correction.	dsPICSM_V32SE.a	

- Note 1:** The 'C' source files should be compiled using the "Optimize for size", "small code model" and "large data model" options.
- 2:** Based on whether the library archive supports V.42 error correction, you must enable or disable the DEF_V42 option in the options.h file and DEF_HDLC in the dpconfig.inc file,
- 3:** Use API.C and AT_cmds.c files for applications with an AT command interface. Use SMUA_OL.c and SMUA_IL.c for applications with an embedded interface.

2.4 SOFT MODEM MODULATION OPTIONS

Although the dsPIC30F Soft Modem Library configurations have been set up to integrate directly with your user application, you can implement custom soft modems by using the source files for the underlying soft modem configuration options.

Table 2-2 lists the modulation options that are used for building soft modem modules. These modulation options are enabled in the `DPCONFIG.INC` file as part of the data pump configuration (see **Section 2.5.1 “Data Pump Modulation Options”**).

TABLE 2-2: SOFT MODEM MODULATION OPTIONS

Modulation Option	Description
Integrated Soft Modem	Includes data pump modulations (V.32bis, V.32, V.22bis, V.23 and V.21/Bell 103) along with ITU-T V.42 error control protocol.
DEF_V32bis(1)	Includes only ITU-T V.32bis data pump modulation along with the ITU-T V.42 error control protocol.
DEF_V32(1)	Includes only ITU-T V.32 data pump modulation along with the ITU-T V.42 error control protocol.
DEF_V22	Includes only ITU-T V.22bis data pump modulation along with the ITU-T V.42 error control protocol.
DEF_V23	Includes only ITU-T V.23 data pump modulation (no error control)
DEF_V21	Includes only ITU-T V.21 data pump modulation (no error control)
DEF_B103	Includes only Bell 103 data pump modulation (no error control)
DEF_V42	Includes files required for ITU-T V.42 layer of the soft modem
DEF_AT	Includes files required for AT command layer
DEF_RTOS	Includes files required to enable scheduling of soft modem processing using the CMX Scheduler RTOS
DEF_LCD_MSGS	Includes files required for displaying some of the soft modem state machine messages on the LCD panel on the dsPICDEM.net™ Connectivity Board.

Note 1: Since ITU-T V.32bis supports all the requirements of ITU-T V.32, it is necessary to enable both DEF_V32BIS and DEF_V32 options for ITU-T V.32bis mode.

2.4.1 Configuring Your Own Modem

If you choose to configure your own modem you will need to include the specific source files for all the modem modulations you include in the data pump configuration. For example, if you want a modem that supports V.22bis, V.23, V.21 and V.42, you would select all of the source files for DEF_V22bis, DEF_V23, DEF_V21 and DEF_V42, respectively, along with required support files, and incorporate them into your user application.

Table 2-3 lists the source files that comprise each modem configuration. ‘Y’ indicates that the designated file for that row is required for the modem configuration in that column. ‘N’ indicates that the file is not needed for that configuration.

dsPIC30F Soft Modem Library User's Guide

To create your own modem configuration follow this process:

1. Open a project in MPLAB.
2. To the project, add all of the source files identified in Table 2-3 for each modem configuration.
3. Add all the support files identified in Table 2-1.
4. Add all the source files for your user application.
5. Build the project.

Note: Refer to **Chapter 5. “Soft Modem Demonstrations”** for detailed procedures for interfacing with your user application.

TABLE 2-3: SOURCE FILES FOR MODULATION OPTIONS

Source Files ⁽¹⁾	Integrated Soft Modem	DEF_V32BIS	DEF_V32	DEF_V22	DEF_V23	DEF_V21/DEF_B103	DEF_V42	DEF_AT	DEF_RTOS	DEF_LCD_MSGS
Data Pump Modules										
Agc.s	Y	Y	Y	Y	Y	Y	N	N	N	N
Callprog.s	Y	Y	Y	Y	Y	Y	N	N	N	N
Carrec.s	Y	Y	Y	Y	N	N	N	N	N	N
Data.s	Y	Y	Y	Y	Y	Y	N	N	N	N
Dmctrl.s	Y	Y	Y	Y	Y	Y	N	N	N	N
Ec.s	Y	Y	Y	N	N	N	N	N	N	N
Eqz.s	Y	Y	Y	Y	N	N	NN	N	N	N
Filter.s	Y	Y	Y	Y	Y	Y	N	N	N	N
Hdlc.s	Y	Y	Y	Y	N	N	N	N	N	N
Mdmkern.s	Y	Y	Y	Y	Y	Y	N	N	N	N
Message.s	Y	Y	Y	Y	Y	Y	N	N	N	N
Modemvar.s	Y	Y	Y	Y	Y	Y	N	N	N	N
Psf.s	Y	Y	Y	Y	N	N	N	N	N	N
Sbar.s	Y	Y	Y	Y	N	N	N	N	N	N
Signal.s	Y	Y	Y	Y	Y	Y	N	N	N	N
Sipl.s	Y	Y	Y	Y	Y	Y	N	N	N	N
Tables.s	Y	Y	Y	Y	Y	Y	N	N	N	N
Toned.s	Y	Y	Y	Y	Y	Y	N	N	N	N
V32bis.s	Y	Y	Y	N	N	N	N	N	N	N
V32ec.s	Y	Y	Y	N	N	N	N	N	N	N
V32rr.s	Y	Y	Y	N	N	N	N	N	N	N
Trel.s	Y	Y	N	N	N	N	N	N	N	N
Viterbi.s	Y	Y	N	N	N	N	N	N	N	N
V22bis.s	Y	N	N	Y	N	N	N	N	N	N
V23.s	Y	N	N	N	Y	N	N	N	N	N
V21.s	Y	Y	Y	Y	Y	Y	N	N	N	N
V8.s	Y	Y	Y	Y	Y	Y	N	N	N	N

Note 1: 'C' Source files should be compiled using the “Optimize for size”, small code model and large data model options.

2: For AT command interface use API.C and AT_cmds.c files. For embedded application Interface use SMUA_OL.C and SMUA_IL.C files.

TABLE 2-3: SOURCE FILES FOR MODULATION OPTIONS (CONTINUED)

Source Files ⁽¹⁾	Integrated Soft Modem	DEF_V32BIS	DEF_V32	DEF_V22	DEF_V23	DEF_V21/DEF_B103	DEF_V42	DEF_AT	DEF_RTOS	DEF_LCD_MSGS
V25.s	Y	Y	Y	Y	Y	Y	N	N	N	N
Rtd.c	Y	Y	Y	Y	Y	Y	N	N	N	N
DTMF Generation Modules										
Dtmfgen.s	Y	Y	Y	Y	Y	Y	N	N	N	N
Dtmftables.s	Y	Y	Y	Y	Y	Y	N	N	N	N
Hardware and Target Specific Modules										
Init.s	Y	Y	Y	Y	Y	Y	N	N	N	N
Init_uart.s	Y	Y	Y	Y	Y	Y	N	N	N	N
Dci_init.s	Y	Y	Y	Y	Y	Y	N	N	N	N
Si3021_init.s	Y	Y	Y	Y	Y	Y	N	N	N	N
Dci_isr.s	Y	Y	Y	Y	Y	Y	N	N	N	N
Isr_uart1.s	Y	Y	Y	Y	Y	Y	N	N	N	N
Traps.s	Y	Y	Y	Y	Y	Y	N	N	N	N
V.42 Error-Control Modules										
V42bf.c	Y	Y	Y	Y	N	N	Y	N	N	N
V42if.c	Y	Y	Y	Y	N	N	Y	N	N	N
V42m.c	Y	Y	Y	Y	N	N	Y	N	N	N
V42main.c	Y	Y	Y	Y	N	N	Y	N	N	N
V42n.c	Y	Y	Y	Y	N	N	Y	N	N	N
V42pf.c	Y	Y	Y	Y	N	N	Y	N	N	N
AT Command Layer Module⁽²⁾										
AT_Cmds.c	Y	Y	Y	Y	Y	Y	N	Y	N	N
Sample Application File⁽²⁾										
Api.c or SMUA_OL.c & SMUA_IL.c	Y	Y	Y	Y	Y	Y	Y	Y	N	N
CMX Scheduler RTOS Library Files										
Cmx_inis.o	N	N	N	N	N	N	N	N	Y	N
Csskvla.o	N	N	N	N	N	N	N	N	Y	N
Lbdspic.a	N	N	N	N	N	N	N	N	Y	N
LCD Message Display Specific Files										
Debmsgs.c	N	N	N	N	N	N	N	N	N	Y
Lcdc.c	N	N	N	N	N	N	N	N	N	Y

Note 1: 'C' Source files should be compiled using the "Optimize for size", small code model and large data model options.

2: For AT command interface use API.C and AT_cmds.c files. For embedded application Interface use SMUA_OL.C and SMUA_IL.C files.

2.4.2 Rebuilding the Soft Modem Libraries

A library archive file consists of all the object files for all the modulation options for the modem type represented by the library. These object files are built from source files that represent the modem functions of the data pump modulation options. If you choose to modify a library archive you will need to rebuild the archive file using batch files provided in the soft modem library folder (...dsPIC30F Soft Modem Library\V.22bis Modem_R1.0\lib).

Table 2-4 identifies the resources that need to be duplicated when you rebuild a modem library. The first column identifies the library by modem type. The second column identifies the archive file that needs to be rebuilt. The third column identifies the batch file you will use for that archive. The last column lists the modulation options used for that modem type.

Rebuilding a library archive is a two-step process:

- First you create and build an MPLAB library project
- Then you rebuild the archive file

TABLE 2-4: BATCH FILES FOR REBUILDING MODEM LIBRARIES

Soft Modem Type	Archive File	Batch File	Modulation Options
V.22bis Integrated	dsPICSM_V22bisE.a	dsPICSM_V22bisE.bat	DEF_V22 DEF_V23 DEF_V21_B103 DEF_v42
V.22bis	dsPICSM_V22bis.a	dsPICSM_V22bis.bat	DEF_V22 DEF_V23 DEF_V21_B103
V.22bis Stand Alone	dsPICSM_V22bisSE.a	dsPICSM_V22bisSE.bat	DEF_V22 DEF_V42
V.23	dsPICSM_V23.a	dsPICSM_V23.bat	DEF_V23 DEF_V21_B103
V.32bis Integrated	dsPICSM_V32bis.a	dsPICSM_V32bis.bat	DEF_V32BIS DEF_V32 DEF_V22 DEF_V23 DEF_V21_B103 DEF_V42
V.32	dsPICSM_V32.a	dsPICSM_V32.bat	DEF_V32 DEF_V22 DEF_V23 DEF_V21_B103
V.32bis, V.32 Stand Alone	dsPICSM_V32bisSE.a	dsPICSM_V32bisSE.bat	DEF_V32BIS DEF_V32 DEF_V42
V.32 Stand Alone	dsPICSM_V32SE.a	dsPICSM_V32SE.bat	DEF_V32 DEF_V42

2.4.2.1 CREATE LIBRARY PROJECT

1. Launch MPLAB IDE and create a library project.
2. Include the source files for each of the modulation options (see Table 2-3).
3. Include the support files for the library (see Table 2-1).
4. Build the project.

At this point, all the object files are produced for the archive.

2.4.2.2 CREATE NEW LIBRARY ARCHIVE

1. Select and run the batch file for the library you are rebuilding (see Table 2-4).

Note: These batch files include path dependencies. You must account for the specific location of these files on your system.

2. Validate that the batch file date has been updated and no errors are reported in the build process.

2.5 SETTING UP COMPILE TIME OPTIONS FOR YOUR APPLICATION

The data pump modulation modules are written in dsPIC30F assembly language to achieve optimum performance. V.42 error control protocol and the AT command layer modules are written in 'C' and compiled with the MPLAB C30 compiler.

There are several software build options that must be configured properly. These options are called out in specific include (.inc) and header (.h) support files.

Data pump modulation options are selected from the `DPCONFIG.INC` file. This file is not a native part of the data modem library archive but is required when building the final application.

2.5.1 Data Pump Modulation Options

The data pump modulation protocols are selected with the following options defined in `DPCONFIG.INC` file:

```
.equ DEF_V32BIS,    1    ; Enable(1)/Disable (0), V.32bis modulation
.equ DEF_V32,       1    ; Enable(1)/Disable (0), V.32 modulation
.equ DEF_V22,       1    ; Enable(1)/Disable (0), V.22 modulation
.equ DEF_V23,       1    ; Enable(1)/Disable (0), V.23 modulation
```

Note: For V.32/V.32bis, you cannot enable V.32bis without also using V.32 (i.e., `DEF_V32BIS = 1` and `DEF_V32 = 0` is not supported). However, you can use V.32 without V.32bis (i.e., `DEF_V32BIS = 0` and `DEF_V32 = 1` is supported).

dsPIC30F Soft Modem Library User's Guide

Additional options defined in the file `DPCONFIG.INC` file must be enabled for all the modulation modes:

```
.equ DEF_V21,      1 ; Enable(1)/Disable (0), V.21 modulation
.equ DEF_CALLPROG, 1 ; Enable(1)/Disable (0), Call Progress
                        ; signaling
.equ DEF_V8,       1 : Enable(1)/Disable (0), V.8 Hand Shake
.equ DEF_V25,      1 ; Enable(1)/Disable (0), V.25 (Answer Tone
                        : Generation/Detection)
.equ DEF_B103,     1 ; Enable(1)/Disable (0), Bell 103 modulation
```

The following option defined in the `DPCONFIG.INC` file must be enabled only if ITU-T V.42 protocol is enabled.

```
.equ DEF_HDLC,     1 ; Enable(1) / Disable (0), HDLC protocol
```

2.5.2 V.42 and AT Command Layer Options

ITU-T V.42 Error-Control and AT command layer software modules are defined in the `OPTIONS.H` file. The `OPTIONS.H` file is not a native part of the library archive but is required when building the final application. If you choose not to use a specific option, you must comment it out and then rebuild the soft modem application.

```
#define DEF_V42      // V.42 Error-Control
#define DEF_AT       // AT Command Layer
```

Note: V.23 and V.21/Bell 103 data pump modulations will not establish an error control connection. Therefore, it is not necessary to enable ITU-T V.42 error control protocol for these two data pump modulations.

2.5.3 Miscellaneous Options

To schedule the soft modem with the CMX Scheduler RTOS you enable the `DEF_RTOS` compile time option. The complete soft modem module is considered a single task that is invoked nearly every 5 milliseconds. The codec interrupts, which occur at a constant rate, are used as the time reference for invoking the wake-up event for the Soft Modem task. Once the soft modem task is invoked, it suspends itself until it receives the wakeup event from the codec interrupt handler. At this point the soft modem functionality is performed and suspended again until it receives the next wake up event.

If CMX Scheduler RTOS is used, the following option must be selected in the `OPTIONS.H` and `DCI_ISR.S` files. Disable this option by commenting it out and then rebuilding the soft modem application.

```
#define DEF_RTOS      // Scheduler RTOS
```

The following option is provided to display some of the soft modem state machine messages on the LCD panel of the connectivity board. You can disable of this option by commenting it out and then rebuilding the soft modem application.

```
#define DEF_LCD_MSGS  // Display Messages on LCD
```

Currently, the delay in displaying the characters on the LCD panel is significantly high, hence this option is disabled.

2.6 CONFIGURING THE ANALOG FRONT END

The Analog Front End (AFE) on the dsPICDEM.net Connectivity Development Boards uses the Silicon Lab Si3034 and Si3035 chip sets. These chip sets incorporate an integrated Data Access Arrangement (DAA) that provides a programmable line interface to support global telephone line interface requirements. The Si3034 and Si3035 chip sets consist of the following DAA and AFE ICs:

- Si3034 chip set
 - Si3021 AFE
 - Si3014 DAA
- Si3035 chip set
 - Si3021 AFE
 - Si3012 DAA

The Si3021 is configured as the Master device and uses its internal clock circuitry to generate the serial bit clock and frame sync clock, at a specific sampling rate, for communication with the dsPIC30F. Initially the Si3034/Si3035 AFE is configured for an 8-KHz sampling rate, which is used during the DTMF generation-detection phase. The dsPIC30F is configured as the Slave device.

Table 2-5 calls out the interface connections between the Si3021 AFE and the dsPIC30F device.

TABLE 2-5: AFE CONNECTIONS TO dsPIC30F

Si3021 Pin	dsPIC30F Pin	Function
MCLK		AFE Clock Generation Input
SCLK	CSCK	Bit Clock
SDO	CSDI	Data Output from Si3021
SDI	CSDO	Data Input to Si3021
FSYNC	COFS	Frame Sync to dsPIC30F6014
OFHK	CN9	Hardware OFF HOOK (not used)
RESET	RG8	AFE Device Reset
RGDT/FSD	SS2/CN11	Not used
FC/RGDT	SCK2/CN8	Not used

Various compile time options are provided to configure the Si3034/Si3035 AFE to comply with international telephone interface requirements.

The `Si3021config.inc` file includes the options needed to set up the AFE for the dsPIC30F Soft Modem Library. Review these options carefully to ensure proper modem operation. Refer to the chip set data sheet (provided on the software CD) for details on the different control registers and chip operation.

dsPIC30F Soft Modem Library User's Guide

2.6.1 Gain Control Register Settings (Register 15: TX/RX Gain Control)

The following options are common to both the Si3034 and Si3035 chipsets. The option names indicate the Control Register number and the name of the functional bits in the corresponding register.

2.6.1.1 Si3034/Si3035 – CHIP SET OPTIONS (dsPICDEM.net 1 AND dsPICDEM.net 2 DEVELOPMENT BOARDS)

TABLE 2-6: TX/RX GAIN CONTROL (REGISTER 15)

Option	Description	Default
<code>.equ R15_TXM, n</code>	n = 1, Mutes the transmit signal	0
<code>.equ R15_RXM, n</code>	n = 1, Mutes the receive signal	0
<code>.equ R15_ARX, n</code>	Analog Receive Gain Selection n = 0, 0 dB Gain n = 1, 3 dB Gain n = 2, 6 dB Gain n = 3, 9 dB Gain n = 4, 12 dB Gain	0
<code>.equ R15_ATX, n</code>	Analog Transmit Attenuation selection n = 0, 0 dB attenuation n = 1, 3 dB attenuation n = 2, 6 dB attenuation n = 3, 9 dB attenuation n = 4, 12 dB attenuation	0

2.6.1.2 INTERNATIONAL CONTROL REGISTER SETTING (CONTROL REGISTERS 16, 17 AND 18)

The Si3034/Si3035 chip set supports four configurable International Control Registers. Si3034 is a Global chip set and can be configured for Federal Communications Commission (FCC), Japan Approval Institute for Telecommunications Equipment (JATE) and Common Technical Regulation (CTR_21) or other country specific options. Si3035 is an FCC only chip set.

Therefore two sets of options are provided based on the chip set used on the dsPICDEM.net Development Boards. The dsPICDEM.net 1 board implements the Si3035 chip set and dsPICDEM.net 2 board implements the Si3034 chip set.

2.6.2 Si3035 – FCC Options (dsPICDEM.net™ 1 Development Board)

Table 2-7 provides the single option for the FCC based chip set.

TABLE 2-7: CONTROL REGISTER 16

Option	Description
<code>.equ DEF_Si3034, 0</code> (Global based chip set) <code>.equ DEF_Si3035, 1</code> (FCC and JATE based chip set)	
<code>.equ R16_IIRE, n</code>	IIR or FIR filter selection for transmit and receive filters n = 0, Enable FIR filter n = 1, Enable IIR filter On power up, default FIR filter is enabled

2.6.3 Si3034 – Global based Chip Set Options (dsPICDEM.net™ 2 Board)

The Si3034 chip set can be fully programmed to meet international requirements and is compliant with FCC, CTR-21, JATE and various other country specific Post, Telephone & Telegraph (PTT) options. Table 2-8 lists the options supported in the `Si3021config.inc` file. Review and select these options carefully to ensure optimal performance of the DAA/AFE. After selecting your options you will need to rebuild for these options to be recognized and take affect.

TABLE 2-8: DAA Si3034 CONFIGURATION

Option	Description
<code>.equ DEF_Si3034, 1</code> (Global based chip set) <code>.equ DEF_Si3035, 0</code> (FCC and JATE based chip set)	
<code>.equ R16_OHS, n</code>	On-Hook Speed n = 0 – The Si3034 executes a fast on-hook n = 1 – The Si3034 executes a slow, controlled on-hook
<code>.equ R16_ACT, n</code>	AC Termination Select n = 0, Selects real impedance n = 1, Selects complex impedance
<code>.equ R16_IIRE, n</code>	IIR or FIR Filter Selection for Transmit and Receive Filters n = 0, Enable FIR filter n = 1, Enable IIR filter
<code>.equ R16_DCT, n</code>	DC Termination Select n = 1, Japan Mode n = 2, FCC Mode n = 3, CTR_21 Mode
<code>.equ R16_RZ, n</code>	Ringer Impedance Selection n = 0, Maximum (high) Ringer impedance n = 1, Synthesize ringer impedance
<code>.equ R16_RT, n</code>	Used to satisfy country requirements on ring detection. Signals below the lower level will not generate a ring detection; signals above the upper level are guaranteed to generate a ring detection. n = 0, 11 to 22 Arms n = 1, 17 to 33 Arms
<code>.equ R17_LIM, n</code>	Current Limit n = 0, All other Modes n = 3, CTR_21 mode
<code>.equ R18_FJM, n</code>	Force Japan DC Termination Mode n = 0, Normal Gain n = 1, If FCC DC Termination mode is selected, setting this bit will force the Japan DC Termination mode while allowing for a transmit level of -1dBm.
<code>.equ R18_VOL, n</code>	Line Voltage Adjust Used to adjust the tip and ring voltage. Lowering this voltage will improve margin in low voltage countries. Raising this voltage may improve distortion performance. n = 0, Normal n = 1, -0.125V n = 2, 0.25V n = 3, 0.125V

Table 2-9 presents the country specific register settings, which should be referred to for the settings called out in Table 2-7

dsPIC30F Soft Modem Library User's Guide

TABLE 2-9: COUNTRY SPECIFIC REGISTER SETTINGS⁽¹⁾

Register	16					17	18
Country	OHS	ACT	DCT[1:0]	RZ	RT	LIM[1:0]	VOL[1:0]
Argentina	0	0	2	0	0	0	0
Australia ⁽¹⁾	1	1	1	0	0	0	0
Austria	0	0 or 1	3	0	0	3	0
Bahrain	0	0	2	0	0	0	0
Belgium	0	0 or 1	3	0	0	3	0
Brazil ⁽¹⁾	0	0	1	0	0	0	0
Bulgaria	0	1	3	0	0	3	0
Canada	0	0	2	0	0	0	0
Chile	0	0	2	0	0	0	0
China ⁽¹⁾	0	0	1	0	0	0	0
Columbia	0	0	2	0	0	0	0
Croatia	0	1	3	0	0	3	0
CTR_21 ⁽²⁾	0	0 or 1	3	0	0	3	0
Czech Republic	0	1	3	0	0	3	0
Denmark	0	0 or 1	3	0	0	3	0
Ecuador	0	0	2	0	0	0	0
Egypt ⁽¹⁾	0	0	1	0	0	0	0
El Salvador	0	0	2	0	0	2	2
Finland	0	0 or 1	3	0	0	3	0
France	0	0 or 1	3	0	0	3	0
Germany	0	0 or 1	3	0	0	3	0
Greece	0	0 or 1	3	0	0	3	0
Guam	0	0	2	0	0	0	0
Hong Kong	0	0	2	0	0	0	0
Hungary	0	0	2	0	0	0	0
Iceland	0	0 or 1	3	0	0	3	0
India	0	0	1	0	0	0	0
Indonesia	0	0	2	0	0	0	0
Ireland	0	0 or 1	3	0	0	3	0
Israel	0	0 or 1	3	0	0	3	0
Italy	0	0 or 1	3	0	0	3	0
Japan ⁽¹⁾	0	0	1	0	0	0	0
Jordan ⁽¹⁾	0	0	1	0	0	0	0
Kazakhstan ⁽¹⁾	0	0	1	0	0	0	0
Kuwait	0	0	2	0	0	0	0
Latvia	0	0 or 1	3	0	0	3	0
Lebanon	0	0 or 1	3	0	0	3	0
Luxembourg	0	0 or 1	3	0	0	3	0

Note 1: See DC Termination Section in Si3034 Data Sheet

2: CTR_21 includes the following countries: Austria, Belgium, Denmark, Finland, France, Germany, Greece, Iceland, Ireland, Italy, Luxembourg, Netherlands, Norway, Portugal, Spain, Sweden, Switzerland and the United Kingdom.

3: Supported for loop current > 20mA

TABLE 2-9: COUNTRY SPECIFIC REGISTER SETTINGS⁽¹⁾ (CONTINUED)

Register	16					17	18
Country	OHS	ACT	DCT[1:0]	RZ	RT	LIM[1:0]	VOL[1:0]
Macao	0	0	2	0	0	0	0
Malaysia ^(1, 3)	0	0	1	0	0	3	0
Malta	0	0 or 1	3	0	0	3	0
Mexico	0	0	2	0	0	0	0
Morocco	0	0 or 1	3	0	0	3	0
Netherlands	0	0 or 1	3	0	0	3	0
New Zealand	0	1	2	0	0	0	0
Nigeria	0	0 or 1	3	0	0	3	0
Norway	0	0 or 1	3	0	0	3	0
Oman ⁽¹⁾	0	0	1	0	0	0	0
Pakistan ⁽¹⁾	0	0	1	0	0	0	0
Peru	0	0	2	0	0	0	0
Philippines ⁽¹⁾	0	0	1	0	0	0	0
Poland	0	0	2	1	1	0	0
Portugal	0	0 or 1	3	0	0	3	0
Romania	0	0	2	0	0	0	0
Russia ⁽¹⁾	0	0	1	0	0	0	0
Saudi Arabia	0	0	2	0	0	0	0
Singapore	0	0	2	0	0	0	0
Slovakia	0	0	2	0	0	0	0
Slovenia	0	0	2	1	1	0	0
South Africa	1	0	2	1	0	0	0
Spain	0	0 or 1	3	0	0	3	0
Sweden	0	0 or 1	3	0	0	3	0
Switzerland	0	0 or 1	3	0	0	3	0
Syria ⁽¹⁾	0	0	1	0	0	0	0
Taiwan ⁽¹⁾	0	0	1	0	0	0	0
UAE	0	0	2	0	0	0	0
United Kingdom	0	0 or 1	3	0	0	3	0
USA	0	0	2	0	0	0	0
Yemen	0	0	2	0	0	0	0

Note 1: See DC Termination Section in Si3034 Data Sheet

2: CTR_21 includes the following countries: Austria, Belgium, Denmark, Finland, France, Germany, Greece, Iceland, Ireland, Italy, Luxembourg, Netherlands, Norway, Portugal, Spain, Sweden, Switzerland and the United Kingdom.

3: Supported for loop current > 20mA

2.7 DTMF GENERATION AND DETECTION MODULES

The DTMF generation and detection modules are key in applications such as answering machines, reception for public and private telephone exchanges, telephony and line test equipment and telephone equipment. These modules are described in **Chapter 4. "DTMF API"**.

The DTMF generation and detection modules are written in dsPIC30F assembly language. Their resource footprint is very small. Table 2-10 indicates the source modules required for DTMF generation and DTMF detection.

TABLE 2-10: DTMF BUILD TABLE

Operation Modules	DTMF Generation	DTMF Detection
Dtmfgen.s	Yes	No
Dtmftables.s	Yes	No
Dtmfbuf.s	No	Yes
Dtmffunc.s	No	Yes
Dtmfinit.s	No	Yes
DetTables.s	No	Yes
Basicop.s	No	Yes
DetTables.inc	No	Yes
DetConst.inc	No	Yes

Two demonstration programs implementing these code modules are presented in **Chapter 5. "Soft Modem Demonstrations"**.

2.8 RESOURCE USAGE

Table 2-11 lists the program memory, data memory and cycle requirement for various data pump modulation protocols, V.42 Error-Control protocol, AT command layer and sample application.

TABLE 2-11: SOFT MODEM RESOURCE USAGE

Modulation ⁽¹⁾	Data Rate (bps)/ Baud	Data Memory (Bytes)	Program Memory (Kbytes)	MIPS
V.32bis	14400/2400	3600	36	15
V.32	9600/2400	3200	31	12
V.22bis/V.22	2400,1200/600	1700	22	7
V.23	600,1200/600,1200	1000	15	4.5
V.21/Bell 103	300/300	1000	13	4.5
V.42	na	2000	14	1.5
AT Command Set	na	150	8	-
Sample Application ⁽²⁾	na	1200	7	-

Note 1: Each Modulation includes all lower data modulations (e.g., V.32 modulation also supports V.23, V.22bis, V.22 and V.21). Support for lower data modulations allows the dsPIC30F Soft Modem to fall back to a lower data rate to establish a connection with slower modems.

2: Sample application refers to the SMUA, which manages the functional layers of the soft modem.

Table 2-12 provides the data memory, program memory and MIPS requirement for DTMF generation and detection modules.

TABLE 2-12: DTMF RESOURCE USAGE

Module	Data Memory (Bytes)	Program Memory (Bytes)	MIPS
DTMF Generation	46	484	0.3
DTMF Detection	224	2800	1.2

dsPIC30F Soft Modem Library User's Guide

NOTES:

Chapter 3. Soft Modem API Specifications

3.1 INTRODUCTION

The dsPIC30F Soft Modem is a subsystem that integrates three modem functions: modulation/demodulation, error control and commands. The modem functions are structured into individual software layers:

- Data Pump (DP) Layer
- V.42 Error-Control Layer
- AT Command Layer

These functional layers are integrated by a comprehensive layer manager, described as the Soft Modem User Application, that is provided as an integral part of the soft modem library. This chapter provides API details for each of these layers.

3.1 HIGHLIGHTS

This chapter covers the following topics:

- Overview of Soft Modem Subsystem
- Soft Modem Data Pump API Specification
- V.42 Protocol Layer API Specifications
- AT Command Layer API Specification

3.2 OVERVIEW OF SOFT MODEM SUBSYSTEM

The dsPIC30F Soft Modem can be structured for two basic modem configurations. In one configuration, the modem is controlled by AT Commands via the UART, as shown in Figure 3-1. The other configuration eliminates the AT command layer and links your application data to defined variables embedded in memory, as shown in Figure 3-2.

FIGURE 3-1: AT COMMAND CONTROLLED SOFT MODEM

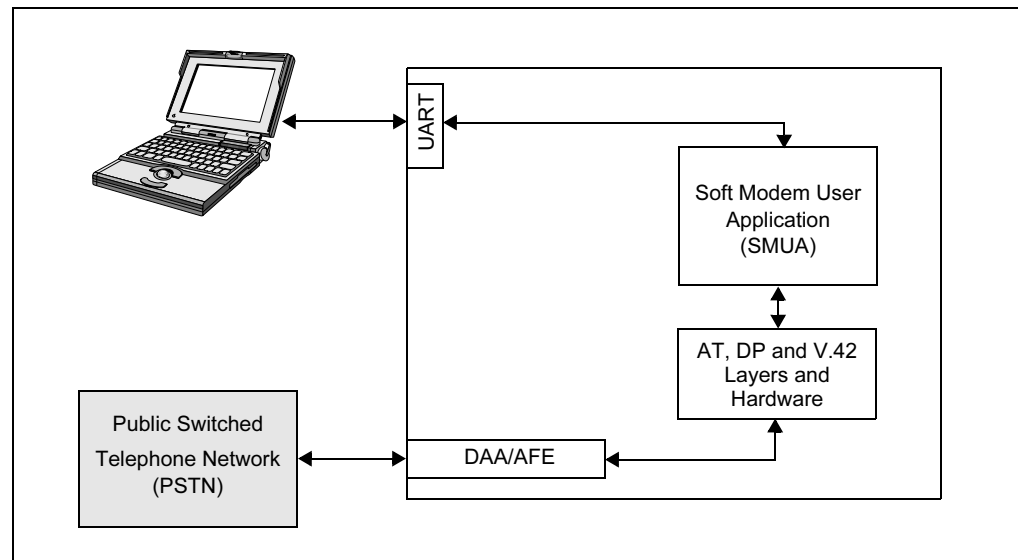


FIGURE 3-2: EMBEDDED SOFT MODEM CONFIGURATION

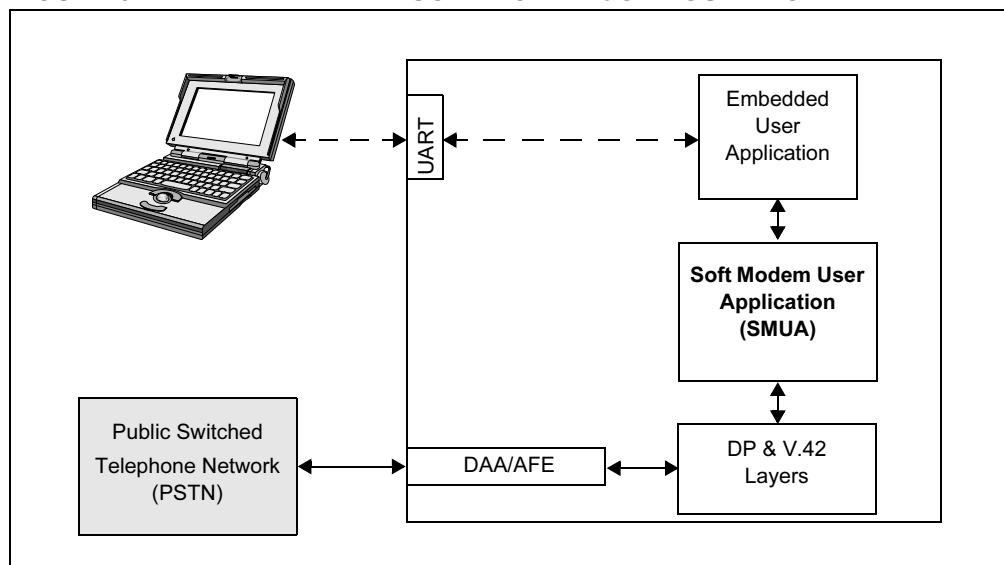
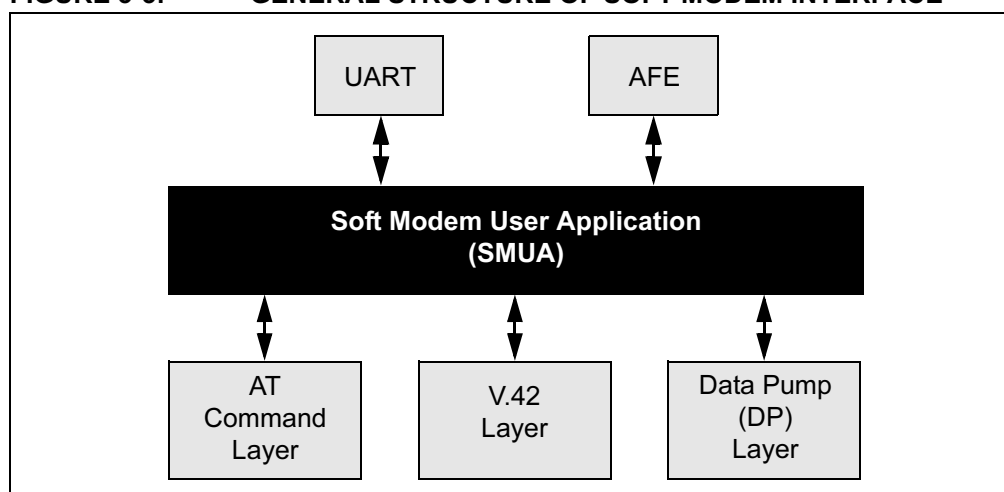


Figure 3-3 focuses more specifically on the general structure of the soft modem interface. The three software layers that comprise the soft modem do not interact directly with each other. They are linked and managed by the Soft Modem User Application (SMUA). The SMUA uses well-defined messages and functions to ensure that the soft modem layers interact appropriately.

Messages that flow between these layers are transparent to the SMUA. The SMUA establishes proper communication between these layers using the API specifications of the different layers.

The SMUA also handles PCM samples that flow between the Data Pump Layer and the AFE as well as the data flow (commands/responses) between the UART and AT Command Layer.

FIGURE 3-3: GENERAL STRUCTURE OF SOFT MODEM INTERFACE

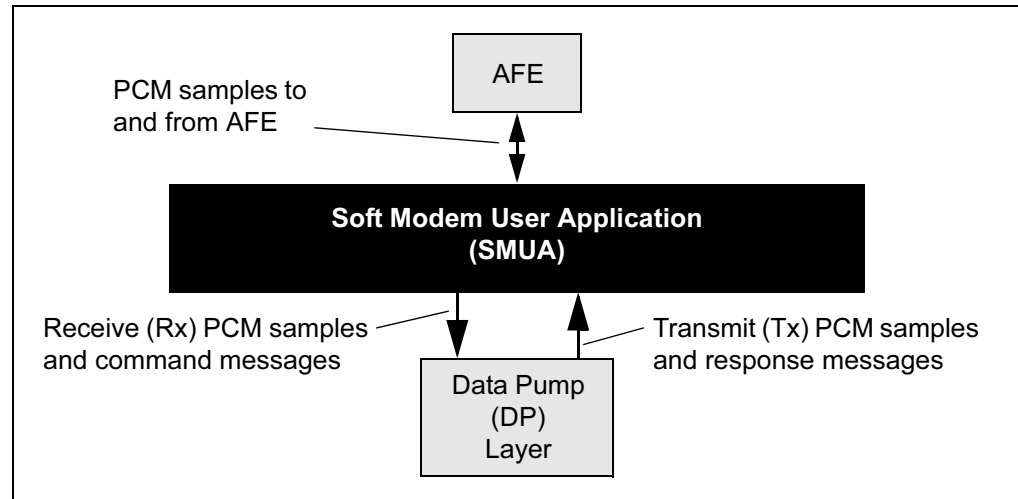


3.3 SOFT MODEM DATA PUMP API SPECIFICATION

The data pump interface layer involves the following entities:

- Interface Data Structures
- Functional Interface
- Messages

FIGURE 3-4: SOFT MODEM DATA PUMP INTERFACE STRUCTURE



3.3.1 Data Pump Interface Data Structures

The data pump interface data structures define messages and the message queue as well as the structure of PCM samples.

3.3.1.1 MESSAGES AND MESSAGE QUEUE DATA STRUCTURE

Information is transferred (shared) between the data pump and the SMUA processes using command or response messages. These messages are generated asynchronously by the data pump or SMUA.

Messages can be queued to facilitate the transfer of multiple messages between the data pump and SMUA. The message queue helps retain messages that do not get completely processed. However, because limited message queue length, it is advisable to empty the queue every time to avoid an overflow and loss of data.

Message queues also help maintain proper sequencing of commands and responses between various components. The Command and Response messages for the data pump consists of three fields:

- Message Identifier
- Length of the Message Parameters
- Message Parameters

dsPIC30F Soft Modem Library User's Guide

A message queue is identified by a structure. In 'C' syntax, the message structure is shown below and is defined in the `API_C.H` interface header file. In the structure called `MESSAGE` there are two fields or elements that represent the Message Identifier and the Length of the Message Parameters. These two elements are implemented for the Command and Response message types.

```
typedef struct {
    unsigned char Id;    // Message ID
    unsigned char Len;   // Message parameters length
} MESSAGE;
```

The following message queue structure consists of four fields or elements that define the message parameters. In 'C' syntax, the message queue structure is shown below and is defined in the `API_C.h` interface header file.

```
typedef struct {
    char *BufPtr;        // Holds the base address of message Queue
    char *Write;         // Write pointer to the queue
    char *Read;          // Read pointer to the queue
    unsigned char Size;  // size of the MESSAGE queue
} MSGQSTRUCT;
```

The SMUA allocates a predefined buffer for each message queue structure.

- One message from the application to the data pump (command message queue)
- One message from the data pump to the application (response message queue)

The message location and length parameters and the read and write access pointers are defined in the message queue structure (illustrated above). The `BufPtr` points to the absolute address where the message queue buffer is located in data memory. The parameter `Size` indicates the length of the message queue buffer.

Note: Note that the message queue is circular. As a result, a message can be left unread in the queue. The recipient (user application or data pump) should make sure that the message is read before the source (data pump or user application) overwrites the message.

The messages are written into the queue using the `Write` pointer and read from the queue using the `Read` pointer. There are no messages in the queue when `Write=Read`. Messages are updated into the message queue in this order: Message ID, Message Length and Message parameters.

3.3.1.2 PCM SAMPLES STRUCTURE

This structure contains pointers to buffers containing modulated samples generated by the data pump instance and the samples required to be demodulated by the data pump instance.

```
typedef struct {
    int FrameLength;    // PCM Sample buffer length
    int *IN;            // Pointer to PCM Receive samples buffer
    int *OUT;           // Pointer to PCM Transmit samples buffer
} PCMDATASTRUCT;
```

The user application will buffer a fixed number of samples and then invoke the data pump. Currently, the data pump processes (demodulation process)/produces (modulation process) 40 samples at each instance.

3.3.2 Data Pump Functional Interface

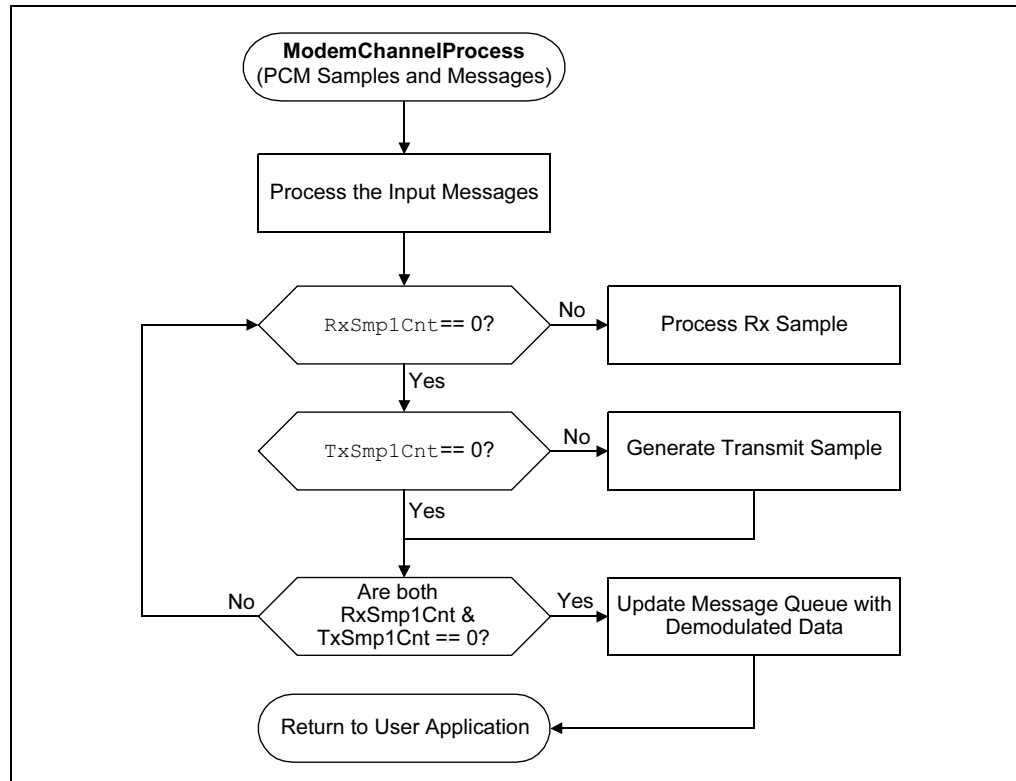
Two functions have been defined for the data pump interface:

- `ModemChannelActivate` opens the data pump channel
- `ModemChannelProcess` modulates and demodulates the data

Figure 3-5 is a simple flow chart that shows the high level functionality of the `ModemChannelProcess` function. An additional function:

- `CheckRingTone` is provided to enable the user to detect the ring tone

FIGURE 3-5: DATA PUMP INTERFACE FLOW



3.3.2.1 FUNCTION DESCRIPTIONS

ModemChannelActivate

Function: Activates/opens the data pump channel. This function has to be called by the SMUA for every new session of the data pump process. (i.e., called once before making a new connection with remote modem). This function initializes the data pump modules and sets up initial conditions. During the call to this function, all the internal variables are set to their default state.

Prototype: `void ModemChannelActivate(PCMDATASTRUCT *
MSGQSTRUCT *,
MSGQSTRUCT *);`

Arguments:

- Pointer to PCM data structure
- Pointer to In Message Queue structure (Commands from application to data pump)
- Pointer to Out Message Queue structure (Responses from data pump to application)

Return Value: None

Example:

```
/* Declare different buffers and structures */  
//PCM samples structure  
PCMDATASTRUCT Pcm;  
//Message Queue for commands sent by the  
//application //to Data pump  
CHAR InMsgQue0[128];  
//Message Queue for responses sent by the  
//data pump to the application  
CHAR OutMsgQue0[128];  
//Structure that maintains the Queue having  
//messages to data pump  
MSGQSTRUCT InMsgQueStr;  
//Structure that maintains the Queue having  
//responses from data pump  
MSGQSTRUCT OutMsgQueStr;  
//pointers to in message queue structure  
MSGQSTRUCT *InMsgQPtr0;  
//pointers to out message queue structure  
MSGQSTRUCT *OutMsgQPtr0;
```

Example:

```
/* Initialize the different pointers */  
Pcm.FrameLength = 40;  
//Pointer to IN and OUT message queues  
InMsgQPtr0 = &InMsgQueStr;  
OutMsgQPtr0 = &OutMsgQueStr;  
//Set the size of the message queue  
InMsgQueStr.Size = 128;  
OutMsgQueStr.Size = 128;  
//Read and write pointers to In and Out  
//message queues  
InMsgQueStr.Read = InMsgQueStr.Write =  
InMsgQueStr.BufPtr = &InMsgQue0[0];  
OutMsgQueStr.Read = OutMsgQueStr.Write =  
OutMsgQueStr.BufPtr = &OutMsgQue0[0];  
/* Activate/open data pump channel */  
ModemChannelActivate(&Pcm[0],  
InMsgQPtr0,OutMsgQPtr0);
```

ModemChannelProcess

Function: Called periodically (after collecting 40 PCM samples, which corresponds to 5 ms at 8 KHz sampling rate) by the SMUA to carry out the modulation/demodulation. PCMDATASTRUCT carries the PCM samples between SMUA and data pump. MSGQSTRUCT carries command and response message queue parameters to/from the data pump.

Prototype: `void ModemChannelProcess(PCMDATASTRUCT *
MSGQSTRUCT *,
MSGQSTRUCT *);`

Arguments:

- Pointer to PCM data structure
- Pointer to In Message Queue structure (Commands from application to data pump)
- Pointer to Out Message Queue structure (Responses from data pump to application)

Return Value: None

Example:

```
/* Invoke DP every 5ms in a loop*/  
while(1)  
{  
    while(PCMSampleCnt < 40); /*PCMSampleCnt is  
                                /incremented by 1 for  
                                /every interrupt in  
CODEC  
                                /ISR */  
    PCMSampleCnt = 0;  
    ModemChannelProcess(&Pcm[0],  
InMsgQPtr0,OutMsgQPtr0);  
}
```

CheckRingTone

Function: Called by the application to check presence of ring signal. Once the data pump enters into either originate or answer mode, this function is invoked by the SMUA every 5 ms. This routine uses SDO output of AFE to check for the presence of ring signal. This function does not give any response message to the application; instead it returns the state of detection for every call to this function.

Prototype: `void CheckRingTone(PCMDATASTRUCT *);`

Arguments: Pointer to PCM data structure

Return Value: 1 – Presence of Ring signal
0 – Absence of Ring signal

Example:

```
Status = CheckRingTone(&Pcm);  
if(Status == 1) RingDetect = 1;  
else RingDetect = 0;
```

3.3.3 Data Pump Command and Response Messages

The data pump can be configured and controlled only by command and response messages. Section 3.3.3.1 describes the command messages that are processed by the data pump. Section 3.3.3.2 describes response messages that are sent by the data pump.

3.3.3.1 COMMAND MESSAGES TO THE DATA PUMP

Command messages to the data pump are defined in Table 3-1.

TABLE 3-1: DATA PUMP COMMAND MESSAGES

Message Definition	Message ID	Description
mDATATODP	0	Data to be modulated
mORIGINATE	1	Enter Answer mode of operation
mANSWER	2	Enter Originate mode of operation
mLOOPBACK	3	Perform local digital loop back
mDATAMODESELECT	5	Configure the Data mode [HDLC-Non-HDLC modes]
mMODESELECT	6	Set up Modulation mode in loop back mode
mRATESELECT	7	This selects the modem connect rate
mHSMODESELECT	8	To select Handshake mode for real time operation
mHANGUP	9	Disconnect/Drop connection with remote modem
mCLPRGSELECT	11	Select the call progress configuration
mSTARTRETRAIN	12	Initiate local retrain
mSELECTRETRAIN	13	Enable/Disable Auto-Retrain
mDETECTDIALTONE	17	Start detection of dial tone
mSETTXLEVEL	19	Program the transmit level in dBm units
mHDLCFRMEND	23	Indicates the end of an HDLC Frame
mCLPRGTONECADENCE	25	Set the tone cadence for different call progress tones.

Following are detailed descriptions of the messages that are sent by the data pump:

mDATATODP

```
Msg_Id = mDATATODP;           //Message ID
Msg_Len = N;                   //Length of the data bytes
Msg_Params[0] = data byte 0;
Msg_Params[N-1] = data byte N-1;
```

This message is used to transfer data to be modulated to the data pump. The data bytes are transferred to the internal data pump buffer for transmission. The internal data pump buffer is the buffer declared as the part of data pump static data memory. The SMUA issues this message for every invocation of the `ModemChannelProcess` function (Section 3.3.2) if the data pump is ready to accept data for modulation. This flow control information is issued by the data pump through response messages `mXOFF` and `mXON`. SMUA starts issuing `mDATATODP` messages only after receiving an `mCONNECT` message from the data pump. Internally the data pump uses a 100-byte buffer to queue the data from the SMUA. The thresholds are set at 25 and 75 for implementing `XON` and `XOFF` conditions, respectively, for bit rates equal to or greater than 1200 bps. Therefore, the maximum message length should be less than $100 - 76 = 24$ data bytes. The minimum message length can be any value greater than or equal to zero. The maximum message length for bit rates less than or equal to 300 bps (V.21 and V.23 modulations) is limited to 1 byte. For these two modulations the SMUA cannot send more than 1 byte per PCM frame.

mORIGINATE

```
Msg_Id = mORIGINATE;
Msg_Len = 0;
```

There are no message parameters for this command. This command is sent at the beginning of the session for starting Call mode handshake procedures. After receiving this message from the SMUA, the data pump waits for an answer tone from the remote modem. When the answer tone is detected, the data pump tries to respond with the appropriate modem signaling. For example, if V.8 `ANSam` is detected, and V.8 is enabled locally, the data pump responds with V.8 signaling. When the connection is established, the data pump responds with the `mCONNECT` message.

mANSWER

```
Msg_Id = mANSWER;
Msg_Len = 0;
```

There are no message parameters for this command. This command is used to enable and initiate Answer mode handshake. After detecting the incoming call, the application should configure the data pump to Answer mode using this command. After receiving this message, the data pump starts sending an answer tone to the line. The data pump tries to connect at the Optimum mode common between the local and the remote modem. After the connection is established and the modem is ready for data transfer, the `mCONNECT` message is sent to the application.

mLOOPBACK

```
Msg_Id = mLOOPBACK;
Msg_Len = 0;
```

There are no message parameters for this command. This command is used to perform local digital loop-back, which may be necessary to test the integrity of the code. When Loop-back mode is selected, the mode and the bit rates are selected according to the settings in `mMODESELECT` and `mDATAMODESELECT` messages.

dsPIC30F Soft Modem Library User's Guide

mDATAMODESELECT

```
Msg_Id      = mDATAMODESELECT;
Msg_Len     = 2;
Msg_Params[0] = 0/1/2/3;
Msg_Params[1] = Number of stop bits // (When configured for
// start-stop mode);
```

This message is used to select the Data Transmission mode as defined below. The message parameter specifies the data format that should be followed by the data pump for both transmission and reception.

Parameter	Mode
0 – DATA_L2MSS	Data is exchanged with start and stop bits, LSB is the first bit in time
1 – DATA_M2L	Data is exchanged without start or stop bits, MSB is the first bit in time
2 – DATA_HDLC	Data is exchanged using HDLC framing
3 – DATA_L2M	Data is exchanged without start or stop bits, LSB is the first bit in time.

It should be noted that in all the modes, the data bit stream is grouped into bytes both for transmission and reception. That is, the application should send the data in 8-bit units, and the data pump will also group the bits into 8-bit units after demodulation.

In the HDLC mode, when there is no data in the transmit buffer, the flag sequence is automatically transmitted. Certain other messages are also provided for controlling the HDLC operation as explained later. In the reception, flags are stripped from the data bit stream and the framed data alone is sent to the user application after de-stuffing.

In the Non-HDLC mode of transmission, when there is a data under run, binary 1 is continuously transmitted until the arrival of the next data byte.

An additional second parameter can be sent with this message. This second parameter selects the number of stop bits when the Start-Stop Bit mode is configured. The valid range is 1-16 bits. This feature can be used for sending Answer Detect Pattern (ADP) and Originate Detect Pattern (ODP) patterns in the V.42 connection establishment phase. ITU-T V.42 specifies these two patterns, which are used during the initial handshake of V.42 connection establishment. These patterns are transmitted in Asynchronous mode with 8-16 stop bits.

mMODESELECT

```
Msg_Id      = mMODESELECT;
Msg_Len     = 1;
Msg_Params[0] = Modulation Mode;
```

This command is used to select the Modulation mode for digital loop back using one of these values:

Value	Modulation Mode
16	V.32bis 14400, 12000, 9600, 7200, 4800 bps
18	V.22/V.22bis 2400, 1200 bps
20	V.23 600, 1200 bps
21	V.21 300 bps
22	Bell 103 300 bps

Soft Modem API Specifications

mRATESELECT

```
Msg_Id      = mRATESELECT;
Msg_Len     = 1;
Msg_Params[0] = Modem Connect Rate index;
```

Using this command the application can select the bit rate at which data transmission can take place. The `Msg_Params[0]` parameter specifies the bit rate based on the parameter value. When this parameter is set to 0, the data pump tries to connect at the highest possible rate that is common between the local and remote modems for the current session.

The parameter values for each Modulation mode are as follows:

Mode	Parameter Value
V.23bis	0 = Adaptive Rates (Usually 14400 bps, TCM) 1 = 12000 bps, TCM 2 = 9600 bps, TCM 3 = 7200 bps, TCM 4 = 4800 bps, Non_TCM
V.32	0 = Adaptive Rates (Usually 9600 Non_TCM) 1 = 4800 Non_TCM
V.22/V.22bis	0 = Adaptive rate (Usually 2400 bps) 1 = Forced to 2400 bps 2 = Forced to 1200 bps
V.21	0 = 300 bps
V.23	For V.23 mode with connect rate set to 0, originating mode transmits at 75 bps and receives at 1200 bps. Answering mode transmits at 1200 bps and receives at 75 bps. 0 = 75TX/1200RX ORIGINATE mode 0 = 1200TX/75RX ANSWER mode 1 = 75TX/600RX ORIGINATE mode 1 = 600TX/75RX ANSWER mode In this mode the connect rate is usually set to 0.

mHSMODESELECT

```
Msg_Id      = mHSMODESELECT;
Msg_Len     = 1;
Msg_Params[0] = Hand Shake Mode;
```

This command is used to select the handshake modes for the current data transmission session. When Auto mode is configured, the data pump tries to establish the connection at the Optimum mode and rate common between both modems.

The Handshake mode can be configured with these parameter values:.

Value	Handshake Mode	Description
0	ENABLE_AUTO	Automatically selects optimum data modulation for both local and remote modem. (Starts with V.8)
1	ENABLE_V8	Enable V.8 Handshake mode
2	ENABLE_V32bis	Enable V.32bis modem handshake
4	ENABLE_V32	Enable V.32 modem handshake
8	ENABLE_V22	Enable V.22 modem handshake
16	ENABLE_V21	Enable V.21 handshake
32	ENABLE_V23	Enable V.23 handshake
64	ENABLE_B103	Enable Bell 103 handshake

dsPIC30F Soft Modem Library User's Guide

When `ENABLE_AUTO` is selected, optimum data modulation is selected. For example if the remote modem supports all the modulation modes and the dsPIC modem supports only V.22bis and V.21, then V.22bis mode will be selected since V.22bis is the Optimum mode supported by both modems.

Only V.32bis is enabled by enabling `ENABLE_V8` and `ENABLE_V.32bis`

Only V.32 is enabled by enabling `ENABLE_V8` and `ENABLE_V.32`

Only V.22bis is enabled by enabling `ENABLE_V8` and `ENABLE_V.22`

Only V.23 is enabled by enabling `ENABLE_V8` and `ENABLE_V.23`

Bell 103 mode should be selected without enabling the V.8 handshake (`ENABLE_V8`)

Modes selected must be included in the data pump software.

mHANGUP

```
Msg_Id          = mHANGUP;
Msg_Len         = 0;
```

This message is used to terminate the active data transmission session. The data pump software returns to the default state immediately as if the channel were freshly opened. Additionally, the `mHANGUPCOMPLETE` message is sent to the application after performing the reset. When in active state of connection, this command abruptly terminates the data pump procedures. This function is useful for terminating the connections on some timeouts and error conditions.

mCLPRGSELECT

```
Msg_Id          = mCLPRGSELECT;
Msg_Len         = 1;
Msg_Params[0]   = Call progress select flag;
```

This message is used to enable or disable the generation/detection of some of the call progress tones. Parameter values are as follows:

Operation	Bit Number	Value
Enable Busy Tone Detection	0	1
Enable Reorder Tone Detection	1	2
Enable Ring-Back Tone Detection	2	4
Enable CNG Tone Transmission	3	8

For example, Busy Tone and Ring-Back tone detection are enabled with the value $1 + 4 = 5$.

mSTARTRETRAIN

```
Msg_Id          = mSTARTRETRAIN;
Msg_Len         = 0;
```

This message is used to force a local retrain. On reception of this message, the data pump will initiate the local retrain.

mSELECTRETRAIN

```
Msg_Id          = mSELECTRETRAIN;
Msg_Len         = 1;
Msg_Params[0]   = 1/0;
```

This command is used to enable the data pump auto retrain configuration. When auto retrain is enabled, the data pump automatically initiates a retrain procedure when the equalizer Mean Square Error (MSE) increases and data transfer cannot continue without a retrain. When auto retrain is disabled, even when the line conditions deteriorate, the data pump doesn't initiate a retrain. Auto retrain is enabled/disabled by sending 1/0, respectively.

mDETECTDIALTONE

```
Msg_Id      = mDETECTDIALTONE;
Msg_Len     = 0;
```

This message is used to start detection of a dial tone. On reception of this message, the data pump starts the detection. The data pump returns the mCLPRGRESP response with mGOTDIALTONE as the parameter to the application to indicate that the dial tone has detected.

mSETTXLEVEL

```
Msg_Id      = mSETTXLEVEL;
Msg_Len     = 1;
Msg_Params[0] = N;
```

This command specifies the transmit level required at the modulator output. The parameter N is the minus dBm transmit level required. For example, if a transmit level of -12 dBm is required, N = 12. N values can range from 0-43.

mHDLCFRAMEEND

```
Msg_Id      = mHDLCFRAMEEND;
Msg_Len     = 0;
```

This message is used only in HDLC Transmission mode and indicates the end of an HDLC frame. This message is especially useful for sending back-to-back HDLC frames without waiting for a frame transmission acknowledge from the data pump. On receiving this message, the data pump automatically sends 16-bit CRC and at least one frame terminating flag before starting the transmission of the next frame.

mCLPRGTONECADENCE

```
Msg_Id      = mCLPRGTONECADENCE;
Msg_Len     = 10;
Msg_Params[0] = Busy tone on time (msec) low byte;
Msg_Params[1] = Busy tone on time (msec) high byte;
Msg_Params[2] = Busy tone off time (msec) low byte;
Msg_Params[3] = Busy tone off time (msec) high byte;
Msg_Params[4] = Reorder tone on time (msec);
Msg_Params[5] = Reorder tone off time (msec);
Msg_Params[6] = Ring back tone on time (msec) low byte;
Msg_Params[7] = Ring back tone on time (msec) high byte;
Msg_Params[8] = Ring back tone off time (msec) low byte;
Msg_Params[9] = Ring back tone off time (msec) high byte;
```

This message is used to set the different call progress tone cadences. The time specified should be in milliseconds.

For example Busy tone cadence (ON Time = 500 ms (0x1F4) and OFF time = 500 ms (0x1F4)) is sent using four message parameters:

```
Msg_Params[0] = 0xF4;
Msg_Params[1] = 0x1;
Msg_Params[2] = 0xF4;
Msg_Params[3] = 0x1;
```

3.3.3.2 RESPONSE MESSAGES FROM THE DATA PUMP

Most of the messages generated by the data pump are in response to commands issued by the SMUA. The data pump generates messages that are similar in structure to those generated by the SMUA. Messages issued by the data pump are enumerated in Table 3-2.

TABLE 3-2: DATA PUMP RESPONSE MESSAGES

Message Definition	Message ID	Description
mDATATOHOST	0	Data demodulated by the data pump
mCONNECT	1	Handshake complete, ready to accept data
mNOCARRIER	2	Data pump detected loss of remote carrier.
mMODEMREADY	3	Data pump is ready to accept commands
mCRCOK	4	HDLC frame received with valid CRC
mCRCERROR	5	HDLC frame failed CRC check
mXON	7	Data transfer can resume
mXOFF	8	Indicates that data transmission is temporarily stopped
mGOTDIALTONE	11	Used as a parameter to the mCLPRGRESP response message to indicate the detection of dial tone
mREMODERETRAIN	16	Indicates the start of modem remote retrain
mLOCALRETRAIN	17	Indicates the start of modem local retrain
mCLPRGRESP	21	Call progress tone detection indication
mSELECTSMPRATE	22	Change the sampling rate of AFE to 7200 Hz
mHANGUPCOMPLETE	23	Response to indicate the completion of HANG UP procedure

mDATATOHOST

```
Msg_Id      = mDATATOHOST;
Msg_Len     = N;
Msg_Params[0] = data byte 0;
Msg_Params[N-1] = data byte N-1;
```

The data pump uses this message to send demodulated data to the application. The length of the message depends on the number of data bytes available during a call (5 msec) to the `ModemChannelProcess` (Section 3.3.2).

Non-HDLC Mode Reception

Data is internally assembled in groups of 8 bits. The assembled byte is saved in the buffer for transmission to the SMUA at the end of processing in `ModemChannelProcess` (Section 3.3.2). When `DATA_L2MSS` is selected, the start and stop bits are deleted from the data bit stream. When Start and Stop Bit mode is not enabled, the data pump transparently passes the assembled bytes to the SMUA. This mode of data transfer is used in any application that uses asynchronous data transfer. For example, if the soft modem is unable to establish connection in V.42 mode, then it falls back to Non-V.42 mode, in which data is transferred in asynchronous (Non-HDLC) mode.

HDLC Mode Reception

In HDLC mode the data message is sent to the SMUA when frame-end is detected or when processing of all the PCM samples in the current time slot is complete. In this mode, when a frame-end sequence is detected, the remaining demodulated data is sent to the SMUA before sending the CRC status message. If the HDLC frame extends to more than one PCM frame, the data is sent to the SMUA only at the end of processing in `ModemChannelProcess`. Hence the SMUA should reject the frames data that it buffers if the CRC error message is sent. The data is assembled in groups of 8 bits with LSB sent first. All the stuffed bits are removed from the received data stream. Since the frame end is detected only after detection of the terminating flag, the demodulated data frame contains the 16 CRC bits also. Hence the last 2 bytes of the received HDLC frame have to be rejected by the SMUA.

mCONNECT

```
Msg_Id = mCONNECT;
Msg_Len = 2;
Modulation Mode = Msg_Params[0];
Connect Rate = Msg_Params[1];
```

This message informs the SMUA that the handshake sequence is complete and data transfer can begin, either transmit or receive or both. This response is issued for `mORIGINATE` and `mANSWER` commands. The two parameters of this message, `Msg_Params[0]` and `Msg_Params[1]` indicate the Modulation mode and the connect rate, respectively, as shown in Table 3-3. The data pump may reissue this message if the data pump retrains and re-establishes the connection with the remote modem.

TABLE 3-3: mCONNECT PARAMETERS

Modulation Mode		Connection Rate	
Msg_Params[0]	Modulation Mode	Msg_Params[1]	Connect Rate
1	V.32bis	0	14400 bps
1	V.32bis	1	12000 bps
1	V.32bis/V.32	2	9600 bps
1	V.32bis	3	7200 bps
1	V.32bis/V.32	4	4800 bps
2	V.22bis	1	2400 bps
2	V.22bis	0	1200 bps
4	V.23	—	75/1200 bps 1200/75 bps
8	V.21	—	300 bps
16	Bell 103	—	300 bps

mNOCARRIER

```
Msg_Id = mNOCARRIER;
Msg_Len = 0;
```

This message has no message parameters and indicates to the SMUA that the data pump has detected the loss of remote carrier.

mMODEMREADY

```
Msg_Id = mMODEMREADY;
Msg_Len = 0;
```

This message is issued to the SMUA to indicate that the data pump is ready to receive messages from the SMUA.

dsPIC30F Soft Modem Library User's Guide

mCRCOK

```
Msg_Id      = mCRCOK;
Msg_Len     = 0;
```

This message is used to indicate to the SMUA that the data pump detected a valid frame and correct CRC in the HDLC mode of reception. This message is not issued in any other data mode. It is to be noted that the actual frame data is 2 bytes less than the data sent to the SMUA. The two CRC bytes at the end of the frame are rejected.

mCRCERROR

```
Msg_Id      = mCRCERROR;
Msg_Len     = 0;
```

This message is used to indicate to the SMUA that the data pump detected an invalid frame and incorrect CRC in the HDLC mode of reception. This message is not issued in any other data mode. The SMUA, on reception of this message, rejects the whole frame data.

mXON

```
Msg_Id      = mXON;
Msg_Len     = 0;
```

The data pump sends this message to resume the data transfer that was temporarily stopped by an mXOFF message. This message is sent by the data pump when the data in its receive buffer is less than 25% of the buffer length.

mXOFF

```
Msg_Id      = mXOFF;
Msg_Len     = 0;
```

This message indicates to the SMUA that the data buffer in the data pump is nearly full and data transfer should be held until an mXON message is sent. The data pump sends this message when the data input buffer is 75% filled.

mREMOTERETRAIN

```
Msg_Id      = mREMOTERETRAIN;
Msg_Len     = 0;
```

This message is used to indicate to the SMUA that the remote modem has initiated a retrain procedure and the data transfer has to be stopped temporarily. The data transfer can resume after getting the mCONNECT message again. The SMUA can start a retrain timeout counter on the reception of this message and terminate the connection if an mCONNECT message is not received within the timeout interval.

mLOCALRETRAIN

```
Msg_Id      = mLOCALRETRAIN;
Msg_Len     = 0;
```

This message is used to indicate to the SMUA that the local modem has initiated a retrain procedure and data transfer should be stopped until re-synchronization is complete. The local retrain procedure is initiated if local auto retrain is enabled. Functionally this message and mREMOTERETRAIN serve the same purpose – both indicate to the SMUA that the data transmission cannot continue until the data pump issues the mCONNECT response.

mCLPRGRES

```
Msg_Id      = mCLPRGRES;  
Msg_Len     = 1;  
Tone_Flag = Msg_Params[0];
```

This message is used to indicate the detection of different call progress tones as indicated by these parameters.

Tone Flag	Tone Detected
11	Dial Tone
12	Busy Tone
13	Reorder Tone
14	Ring back tone

mSELECTSMPRATE

```
Msg_Id      = mSELECTSMPRATE;  
Msg_Len     = 0;
```

The data pump uses this message to change the sampling rate to 7200 Hz. On reception of this message the application should change the sampling rate of the AFE hardware to 7200 Hz. `SetFs7200` and `SetFs8000` functions (see Appendix C) can be used to change the sampling rate to 7200 Hz or 8000 Hz, respectively.

mHANGUPCOMPLETE

```
Msg_Id      = mHANGUPCOMPLETE;  
Msg_Len     = 0;
```

This message is issued by the data pump in response to mHANGUP message. This response indicates to the SMUA that the data pump has executed the hang up procedures.

3.3.4 Data Pump API Implementation Checklist

Use this summary as a checklist to verify your implementation of the SMUA/Data Pump interface.

Data Pump Data Structure Declaration:

1. Declare the message queue data structure (MSGQSTRUCT) and initialize all the members of this data structure (see 3.3.1.1 “Messages and Message Queue Data Structure”).
2. Declare the PCM data structure (PCMDATASTRUCT) and initialize the members of this structure (see 3.3.1.2 “PCM Samples Structure”).

Data Pump Initialization:

1. Invoke the data pump `ModemChannelActivate` initialization function with the correct arguments (see 3.3.2 “Data Pump Functional Interface”).
2. Wait for the mMODEMREADY message from the data pump (see 3.3.3.2 “Response Messages From The Data Pump”).
3. Now the data pump is ready to receive commands and process and generate PCM samples. From here on all the processing is done at the frame boundary.
4. Issue the configuration messages to the data pump (see 3.3.3.1 “Command Messages to the Data Pump”).

Data Pump Processing:

1. Invoke the data pump PCM `ModemChannelProcess` sample processing function at the frame boundary and continue monitoring and processing the responses sent by the data pump (see 3.3.2 “Data Pump Functional Interface”).
2. After receiving the `mCONNECT` message from the data pump, the SMUA sends the data to be modulated to the data pump using `mDATATODP` message considering the data pump flow control messages (see 3.3.3 “Data Pump Command and Response Messages” and 3.3.5 “Data Pump Message Interface Example”).
3. Data pump connection is terminated by issuing the `mHANGUP` message to the data pump (see 3.3.3.1 “Command Messages to the Data Pump”).
4. Data pump connection is reestablished by reconfiguring the data pump for the required mode using the data pump command messages.

3.3.5 Data Pump Message Interface Example

The following example illustrates the flow of messages between the SMUA and the data pump for originate and answer modes. Only basic messages that are required for establishing the connection, transferring data and disconnecting are shown.

TABLE 3-4: ORIGINATE MODE MESSAGE FLOW

SMUA		Soft Modem Data Pump
Wait for the <code>mMODEMREADY</code> message from the data pump		Issue <code>mMODEMREADY</code> message to SMUA to indicate that the data pump is ready to accept commands from SMUA
	←←←	<code>mMODEMREADY</code>
<code>mSETTXLEVEL</code> <code>Msg_Params[0] = 12</code>	→→→	Set the transmit level to -12 dBm
<code>mHSMODESELECT</code> <code>Msg_Params[0] = ENABLE_AUTO</code>	→→→	Enable the optimum data modulation mode
<code>mRATESELECT</code> <code>Msg_Params[0] = 0</code>	→→→	Enable the adaptive rate mode
<code>mMDATAMODESELECT</code> <code>Msg_Params[0] = DATA_L2MSS</code> <code>Msg_Params[1] = 1</code>	→→→	Enable asynchronous (Non-HDLC_ mode with start-stop bits (1 stop bit)
<code>mCLPRGSELECT</code> <code>Msg_Params[0] = 1</code>	→→→	Enable Busy Tone Detection
<code>go_off_hook();</code> <code>mDETECTDIALTONE</code>	→→→	Start the dial tone detection
Wait for the dial tone detection response		Check for the presence of dial tone and return <code>mCLPRGRESP</code> response after detecting the dial tone
	←←←	<code>mCLPRGRESP</code> <code>Msg_Params[0] = mGOTDIALTONE</code>
Start the dial process. Initialize the DTMF generation module to generate DTMF signals for the configured digits. At the end of dial process start the originate mode		

Soft Modem API Specifications

TABLE 3-4: ORIGINATE MODE MESSAGE FLOW (CONTINUED)

SMUA		Soft Modem Data Pump
mORIGINATE	→→→	Start Originate mode handshake
Wait for the mCONNECT message from the data pump		Complete the handshake with the remote modem and issue mCONNECT message at the end of handshake
Data transfer begins	←←←	mCONNECT Msg_Params[0] = Modulation Mode Msg_Params[1] = Connect Rate
Send the data bytes to be modulated to the data pump		Modulate the data bytes received from mDATATODP message.
mDATATODP	→→→	
Msg_Params[0] = TxDataByte1		Send the received demodulated bytes to the SMUA using mDATATOHOST message
Msg_Params[1] = TxDataByte2	←←←	mDATATOHOST
.		Msg_Params[0] = RxDataByte1
.		Msg_Params[1] = RxDataByte2
.		.
Msg_Params[N] = TxDataByteN-1		Msg_Params[N] = RxDataByteN-1
Keep sending the data bytes to be modulated to the data pump		Keep Modulating the data bytes and also keep sending the demodulated data bytes to the SMUA
mDATATODP	→→→	
.	←←←	mDATATOHOST .
.		.
.		
		If the internal data buffers are full, send the mXOFF message to SMUA to request to stop the data.
Stop the data transfer to the DP until the reception of mXON message	←←←	mXOFF
		If the internal data buffers are empty, send the mXON message to SMUA to resume the data transfer.
Resume the data transfer with mDATATODP message	←←←	mXON
Terminate the modem connection with the mHANGUP command		
mHANGUP	→→→	Reset data pump state and issue mHANGUPCOMPLETE response to SMUA
go_on_hook();	←←←	mHANGUPCOMPLETE

dsPIC30F Soft Modem Library User's Guide

TABLE 3-5: ANSWER MODE MESSAGE FLOW

SMUA		Soft Modem Data Pump
Wait for the mMODEMREADY message from the data pump		Issue mMODEMREADY message to SMUA to indicate that the data pump is ready to accept commands from SMUA
	←←←	mMODEMREADY
mSETTXLEVEL	→→→	Set the transmit level to -12dBm
Msg_Params[0] = 12		
mHSMODESELECT	→→→	Enable the optimum data modulation mode
Msg_Params[0] = ENABLE_AUTO		
mRATESELECT	→→→	Enable the adaptive rate mode
Msg_Params[0] = 0		
mMDATAMODESELECT	→→→	Enable asynchronous (Non-HDLC_ mode with start-stop bits (1 stop bit))
Msg_Params[0] = DATA_L2MSS		
Msg_Params[1] = 1		
Wait for the RING.		
Start the Answer mode after RING detection		
go_off_hook()		
mANSWER	→→→	Start Answer mode handshake
Wait for the mCONNECT message from the data pump		Complete the handshake with the remote modem and issue mCONNECT message at the end of handshake
Begin data transfer	←←←	mCONNECT
		Msg_Params[0] = Modulation Mode
		Msg_Params[1] = Connect Rate
Send the data bytes to be modulated to the data pump		
mDATATODP	→→→	Modulate the data bytes received from mDATATODP message.
Msg_Params[0] = TxDataByte1		
Msg_Params[1] = TxDataByte2		
:		
:		
Msg_Params[N] = RxDataByteN-1		
	←←←	mDATATOHOST
		Msg_Params[0] = RxDataByte1
		Msg_Params[1] = RxDataByte2
		:
		:
		Msg_Params[N] = RxDataByteN-1
Keep sending the data bytes to be modulated to the data pump		
mDATATODP	→→→	Keep Modulating the data bytes and sending the demodulated data bytes to the SMUA
	←←←	mDATATOHOST
mDATATODP	→→→	:
:	←←←	mDATATOHOST

TABLE 3-5: ANSWER MODE MESSAGE FLOW (CONTINUED)

SMUA		Soft Modem Data Pump
mDATATODP		If the internal data buffers are full, send the mXOFF message to the SMUA to request to stop the data.
Stop the data transfer to the data pump until mXON message is received.	←←←	mXOFF
		If the internal buffers are empty send the mXON message to the SMUA to resume the data transfer
Resume data transfer with mDATATODP message	←←←	mXON
mDATATODP	→→→	
Terminate the modem connection using mHANGUP command		
mHANGUP	→→→	Reset data pump state and issue mHANGUPCOMPLETE response to SMUA
go_on_hook() ;	←←←	mHANGUPCOMPLETE

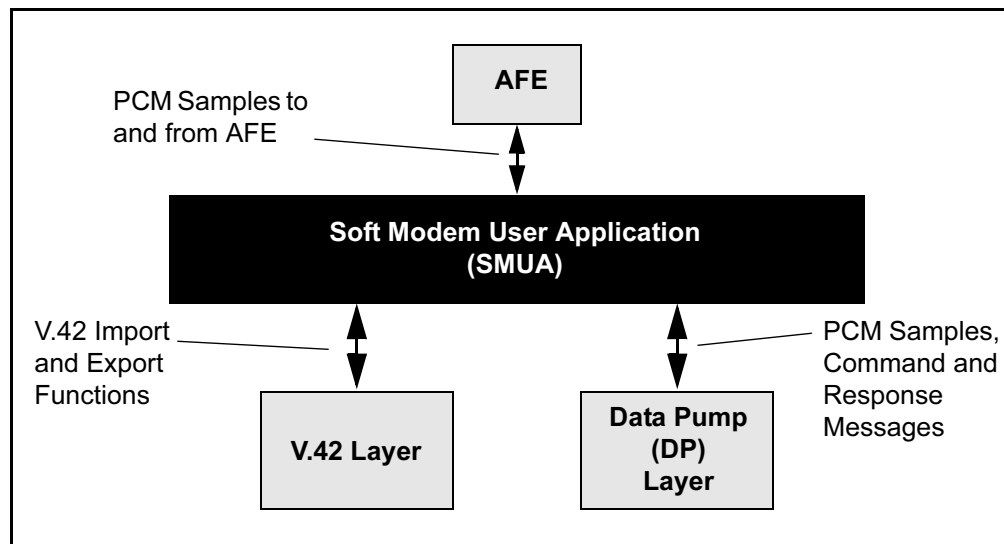
3.4 V.42 PROTOCOL LAYER API SPECIFICATIONS

The dsPIC30F Soft Modem Library uses V.42 for error control. The API includes a set of export functions (defined inside V.42) and import functions (defined inside the SMUA). It is assumed that V.42 gets invoked due to the following events:

- SMUA has data to be sent to the remote modem.
- The underlying data pump has sent a message to the V.42 layer.
- V.42 protocol timer expires.

The interface between V.42 and the data pump consists of messages that are passed by the SMUA transparently. These messages are defined in 3.3.3 “Data Pump Command and Response Messages”). Figure 3-6 shows the relationship of the V.42 layer to the SMUA and data pump.

FIGURE 3-6: V.42 LAYER INTERFACE



3.4.1 V.42 Export Functions

The V.42 layer provides the following interface functions to the SMUA to configure, control and operate the V.42 protocol layer.

- V42GetMemRequirement
- V42InitBaseAddr
- V42OpenChannel
- V42CloseChannel
- V42SendData
- V42ExecuteTimeOut
- V42ConfigParameters
- V42Control
- V42ProcessDPMessage

3.4.1.1 FUNCTION DESCRIPTIONS

V42GetMemRequirement

Function: Returns the size of the static data memory required by the V.42. The SMUA uses this function to get the information on static data memory required by the V.42 layer. The SMUA uses this information to allocate the memory dynamically (Dynamic Memory Allocation).

Prototype: `unsigned long V42GetMemRequirement(void);`

Arguments: None

Return Value: Size of static data memory required by V.42

Example:

```
V42ChnMemSize = V42GetMemRequirement(void);
/* Allocate 'V42ChnMemSize' bytes of static data
/ memory for V.42 layer */
```

V42InitBaseAddr

Function: This function is called by SMUA once at the beginning of the Soft Modem session before the V.42 channel is opened. This function initializes the global pointer `V42Chn0BaseAddress` to the base address of the V.42 data memory block.

Prototype: `void V42InitBaseAddr(void *ChnBasePtr);`

Arguments: * Pointer to V.42 channel structure base address

Return Value: None

Example:

```
/* Pass the base address of the memory of size
'V42ChnMemSize' allocated for V.42 */
V42InitBaseAddr(&V42DataMemBasePtr);
```

V42OpenChannel

Function: SMUA calls this function once at the beginning of the soft modem session. In the dsPIC30F soft modem, V.42 is used as a single channel; therefore, `ChnId` is always zero. Since V.42 protocol layer operation starts only after a connection is established with the data pump, this function is usually called after reception of the `mCONNECT` message from the data pump layer.

This function initializes all V42 variables to their default values. On successful completion of the initialization, returns '1' to the SMUA, otherwise '0' is returned.

Prototype: `unsigned int V42OpenChannel(unsigned int ChnId);`

Arguments: Channel ID (always set to zero)

Return Value: 1 – initialization Succeeded
0 – initialization Failed

Example:

```
Unsigned int ChnId = 0, Status;
Status = V42OpenChannel(ChnId);
```

V42CloseChannel

Function: SMUA calls this function at the end of the modem session to release the memory occupied by the V.42 layer.

Prototype: `void V42CloseChannel(unsigned int ChnId);`

Arguments: * Channel ID

Return Value: None

Example: `Unsigned int ChnId = 0;
V42CloseChannel(ChnId);`

V42SendData

Function: SMUA uses this function to send the data for modulation. V.42 takes attaches the frame header to this data and transfers the framed data to the data pump for modulation. SMUA must call this function once in 5 milliseconds (at PCM frame boundaries, as specified in Section 3.3.2, before invoking the `ModemChannelProcess`.

This routine returns the number of data bytes unread by the underlying V.42 layer. The number of bytes read depends on

- Data size that can be sent to data pump at a time.
- The size of the data that can be buffered in the data pump
- V.42 Window full condition

If the data passed is not completely read by the V.42 layer, the buffer pointer is saved. The remaining data is read in the later stages and the buffer empty situation is indicated to the SMUA with `V42IndicateStatus()` (Section 3.4.2) with `STATUS_ID = V42_READY_FOR_DATA`. SMUA shall send further data only after reception of this indication.

Prototype: `unsigned int V42SendData(unsigned int ChnId, unsigned char *DATA, unsigned int LEN);`

Arguments:

- Channel ID
- Input data buffer pointer
- Length of the buffer

Return Value: Number of data bytes unread from V.42 layer

Example: `unsigned char DataBuffer[128];
/*Send 20 bytes of data to V.42 layer */
Len = V42SendData(0, &DataBuffer[0],20);`

V42ExecuteTimeout

Function: The SMUA calls this function in case of a time out event. The V.42 layer uses an import function (V42SetUpTimer) to set the time out value. SMUA updates the time-out counter, and when the counter becomes zero or negative, executes the time out event by calling this function. This timer is a reference protocol timer and is not referring to any processor timer.

Prototype: void V42ExecuteTimeout(unsigned int ChnId)

Arguments: Channel ID

Return Value: None

Example:

```
/*TimeoutCount is set by the import function
/ V42SetupTimer in 5ms units)
/ TimerStatus is set to active state in the import
/ function V42SetupTimer */
if(TimerStatus == TIMER_ACTIVE)
{
    /* if timeout event is in active state */
    TimeoutCount --; /* Decrement this every
                        /5ms PCM frame boundary */
    If(TimerCounter <= 0)
    {
        //Execute the time out event
        V42ExecuteTimeout(0)
        TimerStatus = TIMER_INACTIVE;
    }
}
```

V42ConfigParameters

Function: SMUA uses this function to configure some V.42 negotiation parameters. When the data pump issues the first `mCONNECT` message, SMUA calls this function with the appropriate configuration parameters. The currently defined Configuration ID's are listed below.

In an active soft modem connection, the data pump issues an `mCONNECT` message after the connection is established with the remote modem. The data pump reissues this message for every connection after local or remote retrain.

Control Parameter	ID	Description
V42DISABLE	2	Disables the V42 protocol. No 'PARAM' associated with this Configuration ID.
DISABLE_ODP_TX	4	Disables the transmission of V.42 Originate Detect Pattern (ODP). When this ID is selected, V.42 directly enters the negotiation phase with transmission of V.42 XID Frame. No 'PARAM' associated with this Configuration ID.
ENABLE_L_TEST	5	This setting allows the V.42 to initiate a request for the loop-back test procedure. No 'PARAM' associated with this Configuration ID.
ENABLE_S_REJECT	6	Enables the single I frame S-REJ handling (receiving only). By default, this feature is disabled. No 'PARAM' associated with this Configuration ID.
TX_WINDOW_SIZE	7	Sets size of the window used by V.42 in the transmit direction. The size is denoted by 'PARAM'. The range allowed is from 1 to 5. The default transmit window size is 5.
RX_WINDOW_SIZE	8	Sets the size of window used by V42 in the receive direction. The size is denoted by 'PARAM'. The range allowed is from 1 to 5. The default receive window size is 5.

Currently the following options in V42 are not supported:

- Multiple I frame S-REJ
- 32-bit FCS
- Transmission of S-REJ frame
- Negotiable frame size (fixed to 128 in both directions)

Prototype: `void V42ConfigParameters(unsigned int ChnId, unsigned int CONFIG_ID, unsigned int *PARAMS)`

Arguments:

- Channel ID
- Configuration ID
- Pointer to Configuration parameters buffer. The maximum length of this buffer is two words.

Return Value: None

Example:

```
/* The transmit window size shall be set to 4
/using this function call */
unsigned int WndSize = 4, ChnId = 0;
V42ConfigParameters(ChnId, TX_WINDOW_SIZE, &WndSize)
```


V42Control

Function: This function is provided by the V.42 layer for the SMUA to control the V.42 connection. When the data pump issues the first `mCONNECT` message to the SMUA after the connection is established with the remote modem, the SMUA calls this function with either `ORIGINATE_MODEM` or `ANSWER_MODEM` control ID, based on the mode of connection. The currently defined Control ID's are as follows:

Control Parameter	ID	Description
<code>START_LTEST</code>	1	On receiving this command, V.42 initiates the loop-back procedure using a pre-defined data sequence. The result of the procedure is indicated by <code>V42IndicateStatus - L_TEST_OK</code> (Section 3.4.2). There are no parameters defined for this Control ID.
<code>V42_HANGUP</code>	2	The SMUA issues this command when it wants to abruptly disconnect the link. The orderly release procedure for V.42 connection is not followed. There are no parameters defined for this Control ID.
<code>ORIGINATE_MODEM</code>	3	This command initiates the V.42 negotiation in Originate mode for the chosen channel. On completing the V.42 handshake, <code>V42IndicateStatus()</code> (Section 3.4.2) is invoked by V.42 to indicate the status. There are no parameters defined for this Control ID.
<code>ANSWER_MODEM</code>	4	This command puts the V.42 channel in the answer mode. The channel awaits the start of negotiation from the originator. There are no parameters defined for this Control ID.
<code>HOST_READY_FOR_DATA</code>	5	The SMUA uses this command to inform V.42 that it is ready to receive the data from V.42. There are no parameters defined for this Control ID.
<code>V42_L_RELEASE</code>	7	This command is used by the SMUA to initiate an orderly release of the error-corrected connection. There are no parameters defined for this Control ID.

Prototype:

```
void V42Control(unsigned int ChnId,  
unsigned int CONTROL_ID,  
unsigned int LEN,  
unsigned int *PARAMS);
```

Arguments:

- Channel ID
- Control ID
- Length of parameters
- Pointer to control parameters buffer

In the above Control Parameter list there are no parameters involved for all the control ID's. However last two parameters (Length and Pointer to buffer) are provided for future use.

Return Value: None

Example:

```
/*Initiate V.42 negotiation in ORIGINATE Mode*/  
unsigned int ChnId = 0, Len = 0, ParamBuffer[1];  
V42Control(ChnId, ORIGINATE_MODEM, LEN,  
&ParamBuffer[0]);
```

V42ProcessDPMessage

Function This function is invoked by the SMUA when the data pump has a message to be transferred to the V.42 layer. This function processes the messages sent by the data pump.

SMUA transfers all the messages sent by the data pump to the V.42 layer after and including the first `mCONNECT` message. SMUA need not process any message after the first `mCONNECT` message; it only passes the messages to the V.42 layer using this function.

Prototype: `void V42ProcessDPMessage(unsigned int ChnId, unsigned char *msg, unsigned char *MsgParams)`

Arguments: Channel ID
Pointer to message structure (MESSAGE)
Pointer to Message parameter buffer.

Return Value: None

Example:

```
unsigned int ChnId = 0
MsgS MESSAGE; //global Message structure declaration
MsgParams[10]; //Local buffer for message parameters
/* If DP issues mDATATOHOST message, SMUA has to call
this function as follows */
MsgS.Id = mDATATOHOST;
MsgS.Len = N;
/*Fill 'N' number of bytes sent by DP through this
message into the MsgParams buffer */
V42ProcessDPMessage(ChnId, (UCHAR *)&MsgS,
&MsgParams[0]);
```

3.4.2 V.42 Import Functions

The SMUA is the communication channel between the data pump and V.42 layers. The SMUA provides these functions for the V.42 layer:

- V42ReceiveData
- V42SetUpTimer
- V42ReleaseTimer
- V42IndicateStatus
- V42SendMessageToDP

3.4.2.1 FUNCTION DESCRIPTIONS

V42ReceiveData

Function: V.42 layer calls this function to transfer the final received data from the remote modem to the SMUA.

The following pseudo-code illustrates the functionality that needs to be implemented by the SMUA.

```
unsigned int V42ReceiveData (unsigned int ChnID,
                             unsigned char *Data,unsigned int LEN)
{
    for(i=0;i< LEN ;i++)
    {
        /*Save the received data in the buffer or
        /transfer these bytes to UART or any upper layer */

        HostRxDataBuf[i++] = *Data++;
    }
    return(LEN-i);
}
```

Prototype: unsigned int V42ReceiveData(unsigned int ChnId,
unsigned char *DATA, unsigned int LEN);

Arguments:

- Channel ID
- Pointer to Data buffer
- Length

Return Value: None

V42SetUpTimer

Function: V.42 layer calls this function to setup a time out value for various V.42 state-machine sequences. The timer count value to be set is passed in the second parameter. The timer count value specifies the time in 5 ms units. The following pseudo-code illustrates the functionality that needs to be implemented by the SMUA.

```
void V42SetupTimer(unsigned int ChnId, unsigned long
Tout_Period)
{
    TimerStatus = TIMER_ACTIVE;
    //Activate the timer
    Timeout = FALSE;
    //No time out has occurred
    TimeOutCount = Tout_Period;
    //Set the time out value.
}
```

During the timer active state, SMUA must decrement the TimeOutCount and should invoke the V42ExecuteTimeOut time out event process function (Section 3.4.1) after this timer count elapses.

```
if(TimerStatus == TIMER_ACTIVE)
{
    TimeOutCount--;
    //Decrement the time out count
    if(!TimeOutCount)
    //Check timer count becomes zero
    {
        Timeout = TRUE;
        //Time out has occurred
        TimerStatus = TIMER_INACTIVE;
        //Timer becomes inactive
        V42ExecuteTimeOut(0);
        //Execute the V.42 export function to
        //process the time out event.
    }
}
```

Prototype: void V42SetupTimer(unsigned int ChnId, unsigned long
TIMER_VALUE);

Arguments:

- Channel ID
- Timer count value in 5 ms unit

Return Value: None

V42ReleaseTimer

Function: V.42 layer calls this function to disable the V.42 event timer. The following pseudo code illustrates the functionality needs to be implemented by the SMUA.

```
void V42ReleaseTimer(unsigned int ChnId)
{
    Timeout          = FALSE;
    TimerStatus      = TIMER_INACTIVE;
    TimeOutCount     = 0;
}
```

This resets the timer and keeps the timer in idle (non-decrementing) state.

Prototype: void V42ReleaseTimer(unsigned int ChnId)

Arguments: Channel ID

Return Value: None

V42IndicateStatus

Function: V.42 layer calls this function to indicate the status of V.42 connection. Status IDs currently defined are:

Status Parameter	ID	Description
V42_CONNECT	2	This response is issued after the local V.42 completes negotiation for an error-corrected link with the remote V42.
ASYNC_CONNECT	4	This response is given to the application if the local V42 fails to negotiate error-corrected connection with the remote V.42.
V42_RETRAIN_CONNECT	18	This response is issued when the V42 state changes from retrain to connect.
DP_RETRAIN_STATUS	14	This response is issued to indicate that data pump and V.42 are in retrain state and cannot accept data. The user application can start resending the data only after receiving the V42_RETRAIN_CONNECT indication.

dsPIC30F Soft Modem Library User's Guide

V42IndicateStatus (Continued)

V42_PROTO_ERROR	5	<p>This indication is produced due the V.42 protocol related exceptions. 'PARAM' can take on any one of the following values:</p> <p>Refer to ITU-T V.42 specification for explanations of these terminologies.</p> <p>N400_RE_TX – Link released after maximum number of retransmissions of a V.42 frame without getting the expected response frame from the remote modem.</p> <p>DM_RESP_WITH_FBIT0 – There is a DM response from remote V.42 with F bit set '0'</p> <p>FRMR_RECD – FRMR frame received from the remote modem during connect state.</p> <p>REJ_RESP_FBIT1 – Received REJ frame with F bit set '1' when V.42 is not in timer recovery condition.</p> <p>DM_RESP_WITH_FBIT1 – There is a DM response from remote V.42 with F bit set '1'</p> <p>NO_ACK_FOR_BREAK – There is no acknowledgement for the break even after maximum number of retransmissions.</p>
DP_LOST_CARRIER	3	<p>This response is issued if the local V42 receives mNOCARRIER from the underlying data pump. The link is abruptly released.</p>
L_TEST_OK	7	<p>This response is issued after the loop-back test procedure is successfully completed. It is SMUA responsibility to handle the situation wherein V42 fails to return this status after it is commanded to initiate the TEST frame.</p>
V42_READY_FOR_DATA	9	<p>This response is used to achieve the flow control for the data from SMUA to the V42 module. This data is passed on to the data pump for modulation after attaching the V.42 frame header information. The SMUA only sends data to V.42 on reception of this status ID.</p>
L_RELEASED	1	<p>This response is issued subsequent to the orderly release procedure initiated due to V42_L_RELEASE given by the SMUA</p> <p>The PARAM can take on the following value:</p> <p>NO_ACK_FOR_DISC – Indicates that local V.42 failed to receive the acknowledgement for the DISC (disconnect frame), even after 4 retransmissions.</p>

V42IndicateStatus (Continued)

REMOTE_V42_DISCONNECT 10 When the remote V.42 initiates an orderly release of the link, this response is given to the application to disconnect the link. No PARAM associated with this call.
The following pseudo code description illustrates the actions to be taken by the SMUA when it receives of above commands.

```
switch (STATUS_ID) {
    case V42_CONNECT:
        /* Indicates connection is established in V.42 LAPM
        /mode and user application starts sending the data
        /to V.42 (This data is passed to DP after attaching
        /V.42 frame header information for modulation) */
        break;
    case ASYNC_CONNECT:
        /* Indicates the connection is established in
        /Non-V.42 mode and user application starts sending
        /the data to V.42 (This data will be passed on to
        /DP for modulation) */
        break;
    case V42_RETRAIN_CONNECT:
        /* During the retrain state V.42 or DP does
        /not accept data for modulation. This indication
        /is issued once DP/V.42 has been successfully
        /retrained and entered into connection state and
        /data transfer begins. */
        break;
    case DP_RETRAIN_STATUS:
        /* This indicates to the user application that
        /DP/V.42 has entered into retrain state and
        /data transfer should stop till reception of
        /V42_RETRAIN_CONNECT status
        break;
    case V42_PROTO_ERROR:
    case DP_LOST_CARRIER:
    case L_RELEASED:
    case REMOTE_V42_DISCONNECT:
        /*User application terminates connection
        /and goes on hook. */
        break;
    case V42_READY_FOR_DATA:
        /*Send Next block of data to V.42 (This data will
        /be passed on to DP after attaching V.42 frame
        /header information for modulation) */
        break;
    default:
        break;
}
```

V42IndicateStatus (Continued)

Prototype: `void V42IndicateStatus(unsigned int ChnId, unsigned int STATUS_ID, unsigned int LEN, unsigned char *PARAM);`

Arguments:

- Channel ID
- Status ID
- Length of status parameter
- Pointer to status parameters buffer

Return Value: None

V42SendMessageToDP

Function: This function is called by V42 whenever it has to send a message to the underlying data pump. It is the soft modem user application's responsibility to route the message to the data pump. It is possible that V42 will dump the entire window of the frames. The user application will have to buffer this data and send to the data pump based on the buffering situation on the data pump side. The data pump provides Xon/Xoff capability so that the data flow across the host and DSP interface can be controlled.

Prototype: `void V42SendMessageToDP(unsigned int ChnId, unsigned int MSG_ID, unsigned int LEN, unsigned char *PARAM);`

Arguments:

- Channel ID
- Message ID
- Message Length
- Pointer to Message buffer

Return Value: None

3.4.3 V.42 API Checklist

Use this summary as a checklist to verify your implementation of the SMUA/V.42 interface.

1. Get the V.42 channel data memory requirement by calling the V.42 export function `V42GetMemRequirement`.
2. Allocate the required amount of memory.
3. Initialize the base address of the channel memory by calling the V.42 export function `V42InitBaseAddr`.
4. Wait for the `mCONNECT` message from the data pump.
5. Open the V.42 channel by executing the V.42 export function `V42OpenChannel`.
6. Start and configure the V.42 protocol layer in either Originate or Answer mode by executing the V.42 export functions `V42Control` and `V42ConfigParameters` and also pass this `mCONNECT` message to the V.42 layer using the V.42 export function `V42ProcessDPMessage`.
7. From now on V.42 layer functions must be invoked in the following events.
 - The underlying data pump has sent a message to the V42 layer
 - V42 protocol timer expires
 - SMUA has data to be sent to the remote modem
8. All the messages received from the data pump are passed on to the V.42 layer using the import function `V42ProcessDPMessage`. All the messages given by the V.42 layer through the import function `V42SendMessageToDP` are passed on to the data pump layer.
9. SMUA layer must implement two export functions, `V42SetupTimer` and `V42ReleaseTimer`, to control the V.42 protocol timer. V.42 layer calls these functions to set up an event timer. The SMUA updates the timer and calls the time-out import function `V42ExecuteTimeOut`.
10. V.42 calls the export function `V42IndicateStatus` to indicate the status of the V.42 layer.
11. Wait for the `V42_CONNECT` status indication from the V.42 layer.
12. If SMUA has data to be sent to the remote modem, SMUA invokes function `V42SendData`. The SMUA invokes this function at the frame boundary. The amount of data per frame depends on the data pump buffer limitation.
13. V.42 layer sends the received data (received from the remote modem) to the SMUA using the import function `V42ReceiveData`.
14. V.42 channel is closed using export function `V42CloseChannel`

3.4.4 Pseudo-Code for Using V.42 API

Following is pseudo code for a typical application that initiates V42 connection, transmits data and disconnects.

```
V42ChnMemsize = V42GetMemRequirement();
V42DataMemBasePtr = malloc(V42ChnMemsize)
V42InitBaseAddr(V42DataMemBasePtr)

V42OpenChannel(ChnId);
Start DP in ORIGINATE mode

LOOP                                /*Wait for mCONNECT message from data pump*/
{
    if(DPMesgID != mCONNECT) continue;

    V42Control(ChnId,ORIGINATE_MODEM)
    V42ProcessDPMessage(mCONNECT,MsgParams);
    Break;
}
LOOP                                /*wait for V42 connect response*/
{
    if (!V42Connect) continue;      /*V42Connect is set in the function
                                     /call V42IndicateStatus with
                                     /status ID V42_CONNECT*/

    /*from this point on all function calls are event based*/
    /*all events are in bold ITALIC letters*/

    if (DATA TO BE SENT)
    {
        if(NO_PENDING_DATA
        /* this flag is managed by the SMUA */
        DATA_SENT =V42SendData(ChnId,&DATA[0],DATA_LEN);

        if(DATA_SENT<DATA_LEN)NO_PENDING_DATA = 0;
        /*PENDING_DATA is set with the function call
        /V42IndicateStatus with STATUS_ID,V42_READY_FOR_DATA*/
    }
    if(TIME_OUT_EVENT)
    {
        V42ExecuteTimeOut();
        /*the time out occurs when the timer count set by
        /V42SetUpTimer function elapses.*/
    }
    if(MESSAGE FROM DP)
    /* If there is a message from DP send it to V.42*/
    {
        V42ProcessDPMessage(ChnId,MSG);
    }

    /* SESSION_COMPLETE is set for the following indications :
    / L_RELEASED
    / REMOTE_V42_DISCONNECT
    / DP_LOST_CARRIER */

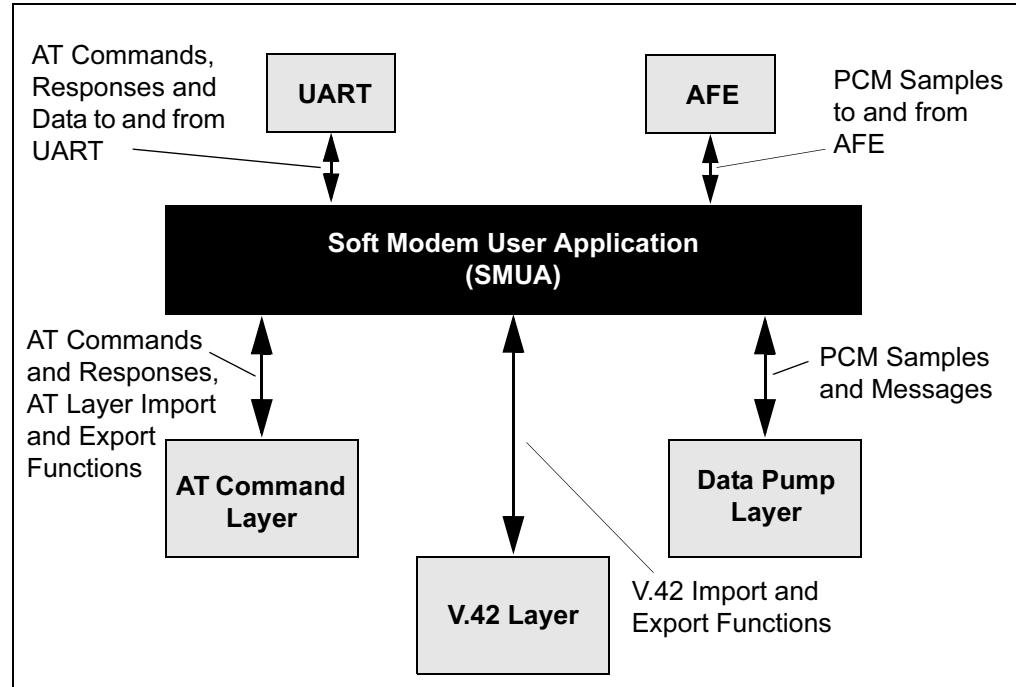
    if(SESSION_COMPLETE) break;
    /*break the loop*/
}
V42CloseChannel(ChnId);
}
```

3.5 AT COMMAND LAYER API SPECIFICATION

The dsPIC30F Soft Modem Library uses a subset of the industry standard AT Command Set (appropriate for embedded applications).

Figure 3-7 illustrates the relationship between the AT Command Layer and the SMUA. The API includes a set of export functions defined inside the AT command layer and import functions defined inside the SMUA.

FIGURE 3-7: AT LAYER INTERFACE



Appendix B describes the AT commands implemented in the dsPIC30F Soft Modem Library.

3.5.1 AT Command Layer Message Queue Structure

The AT command layer interfaces with the data pump and V.42 layers through messages that are passed by the SMUA. The SMUA handles the flow of data to and from the UART. The SMUA routes commands, responses and data between the AT Command Layer and the UART interface. Command data from the SMUA to the AT command layer is transferred through a message queue structure, as described below.

The AT command layer process the commands with import functions and returns the responses to the SMUA. The SMUA transfers the response strings given by the AT command layer to the UART.

```
typedef struct
{
    char *BufPtr;           //Holds the base address of AT message Queue
    char *Write             //Write pointer to the queue
    char *Read;            //Read pointer to the queue
    char Size;             //size of the MESSAGE queue
}
ATMSGQSTRUCT;
```

The SMUA allocates a predefined buffer to the message queue. The `BufPtr` points to the absolute address where the circular command buffer is located and the `Size` parameter indicates the length of the command buffer.

3.5.2 AT Command Layer Export Functions

The AT Command Layer provides these export functions:

- `ATInitBaseAddress`
- `InitATCommands`
- `ProcessATCommands`
- `DPIndicateStatus`

Export functions are provided by the AT command layer to the SMUA to configure, control and operate the AT command layer.

3.5.2.1 FUNCTION DESCRIPTIONS

ATInitBaseAddress

Function: This function initializes the global pointer `ATChnBaseAddress` to the AT command data structure. SMUA calls this function once at the start of the modem session.

Prototype: `void ATInitBaseAddress(void *ChnBaseAddr);`

Arguments: Base address for AT command data structure

Return Value: None

InitATCommands

Function: This function is called by the SMUA once at reset to set different parameters for the AT command layer. AT command layer sets its state variables to default values. The default parameters initialized by the AT command layer will be indicated to the SMUA through AT command layer import functions (Section 3.5.3).

Prototype: `void InitATCommands(void);`

Arguments: None

Return Value: None

Example: `InitATCommands();`

ProcessATCommands

Function: This function is called by the SMUA every frame period (currently 40 samples per frame – equivalent to 5 ms) before the data pump processing (`ModemChannelProcess` (Section 3.3.2) function is invoked. During the command processing mode, this function checks the command buffer and processes any pending commands and sends the responses to the application through the import functions (Section 3.5.3). During the data mode, the command buffer is used to buffer the data. This function checks for the escape sequence in the data buffer and a change of mode is indicated to the SMUA if the escape sequence is found, otherwise no processing is done in this function.

Prototype: `void ProcessATCommands(ATMSGQSTRUCT *, char H_Mode);`

Arguments:

- Pointer to AT message queue structure
- Soft Modem Current mode

Return Value: None

ProcessATCommands (Continued)

Example:

```
ATMSGQSTRUCT   ATInMsgQueStr;
ProcessATCommands (&ATInMsgQueStr,H_Mode);
/H_Mode indicates the current mode of soft-modem
/H_Mode takes the following modes
/*Command mode initial state
/(AT command layer accepts all commands) */
#define OFFLINECMDMODE      0
/*Entering into command mode after the
/connection is established. If AT command layer
/detects an escape sequence in data during the
/data mode, it indicates the SMUA to enter into
/this (command) mode. */
#define ONLINECMDMODE      1
/* SMUA enters this mode after the connection
/ is established with the remote modem */
#define ONLINEDATAMODE      2
/* SMUA enters this mode after dialing or
/answering a call and before the connection is
/established*/
#define ONLINEMODE          3
```

DPIndicateStatus

Function: This function is called by the SMUA to provide the status of the data pump and V.42. The following Status-IDs are defined. It is the SMUA responsibility to invoke this function based on the data pump and V.42 messages as explained in the below table.

Status	ID	Params	Description
DP_CONNECT	1	Connect Rate	The SMUA invoke this function when it receives mCONNECT message from the data pump. The bit rate information is derived from the mCONNECT message parameters.
DP_ASYNC_CONNECT	2	None	The SMUA invokes this function when it receives ASYNC_CONNECT message from the V42 layer
DP_LAPM_CONNECT	3	None	The SMUA invokes this function when it receives V42_CONNECT message from the V42 layer.
DP_LOCAL_RETRAIN	4	None	The SMUA invokes this function when it receives mLOCALRETRAIN message from the data pump.
DP_REMOTE_RETRAIN	5	None	The SMUA invokes this function when it receives mREMOTERE-TRAIN message from the data pump.
DP_CP_STATUS	7	COMRING (173) COMNODIALTONE (191) COMBUSY (182) NOCARRIER (17)	The SMUA invokes this function to indicate the call progress status and no carrier indication of the data pump.
DP_HANGUP	8	None	The application invokes this function when it receives mHANGUPCOMPLETE from the data pump

DPIndicateStatus (Continued)

Prototype:	<code>void DPIndicateStatus(char STATUS_ID, unsigned int *Params);</code>
Arguments:	<ul style="list-style-type: none">• Data pump /V.42 Status ID• Parameters required if any.
Return Value:	None
Example:	<p>If data pump layer indicates to the SMUA the successful data pump connection (mCONNECT message) with the remote modem at V.32 9600 bps The SMUA invokes this function as follows.</p> <pre>unsigned int MsgParams[2]; MsgParams[0] = 9600; DPIndicateStatus(DP_CONNECT,MsgParams);</pre>

3.5.3 AT Command Layer Import Functions

The AT Command Layer provides these import functions:

- ATSendMsgToDP
- ATSendStringToUART
- ATIndicateStatus

3.5.3.1 FUNCTION DESCRIPTIONS

ATSendMsgToDP

Function:	This function is called by the AT command layer whenever it has to send a message to the underlying data pump. It is the SMUA responsibility to route the message to the data pump (see Section 3.5.3). AT command layer uses the same messages as described in Section 4.1.3 and hence SMUA has just route these messages to data pump using message queues as explained in Section 4.1.1
Prototype:	<pre>Void ATSendMsgToDP(char MSG_ID, unsigned char MSG_LEN, unsigned char *Params);</pre>
Arguments:	Message ID (Same as data pump messages) Message Length. Pointer Message Parameters buffer
Return Value:	None

ATSendStringToUART

Function:	This function is called by the AT command layer to transmit a string of characters to the upper layer (currently it is UART). SMUA transfers the AT command response strings to the UART.
Prototype:	<pre>Void ATSendStringToUART(unsigned char *Buffer unsigned int Length);</pre>
Arguments:	<ul style="list-style-type: none">• Character string buffer• Number of characters in the string buffer
Return Value:	None

ATIndicateStatus

Function: This function is implemented by the SMUA to perform actions requested by the AT command layer. This function is called by the AT command layer to indicate the status of the AT command layer and to request some actions from the SMUA. The following Status-IDs are issued by the AT command layer:

Status	ID	Params	Description
DETECT_ESCAPE_SEQUENCE	1	None	Indicates detection of the escape sequence. Correspondingly the SMUA should switch the mode to command mode. (ONLINEDATA-MODE -> ONLINECMDMODE)
GO_ON_HOOK	2	None	Request to change the AFE state to on-hook state.
GO_OFF_HOOK	3	None	Request to change the AFE state to off-hook state.
MODEM_RESET	4	None	Resets the data pump status.
START_ORG_MODE	5	DialStringLen = Params[0]; DialDigits = Params[1] .. [N]	Request to start the data pump in ORIGINATE mode.
START_ANSWER_MODE	6	None	AT command layer issues this to request to start the data pump in ANSWER mode.
MODEM_HANGUP	8	None	Request to hang up the data pump.
SET_H_MODE	9	H_Mode	This command changes the H_Mode.
MODEM_CONFIG	10	See Table 3-6	Passes certain configuration parameters to the data pump.
START_ALB_MODE	11	None	Request to start the data pump in loop back mode
CLPRG_SPEAKER_VOLUME	12	SpeakerVolume = Params[0]	Requests SMUA to change the call progress speaker volume SpeakerVolume = 1 -> Low SpeakerVolume = 2 -> Medium SpeakerVolume = 3 -> High

See Example 3-1 for pseudo-code that illustrates how this functionality is implemented.

Prototype: void ATIndicateStatus(char STATUS_ID, unsigned int *Params);

Arguments:

- AT command Layer Status ID
- Parameters required if any

Return Value: None

**TABLE 3-6: MODEM CONFIGURATION PARAMETERS USED WITH
ATIndicateStatus IMPORT FUNCTION**

No.	Description
0	Transmit Level = Params[0];
1	HandShakeMode = Params[1];
2	HandShakeMode = Params[2];
3	Min_ConnectRate = Params[3];
4	Max_ConnectRate = Params[4];
5	DetectDialTone = Params[5];
6	DetectBusyTone = Params[6];
7	DPConnectTimer = Params[7];
8	V42Enable/Disable= Params[8];
9	Flow Control Type = Params[9];

EXAMPLE 3-1: FUNCTIONALITY NEEDED BY USER APPLICATION

```
switch(STATUS_ID)
{
    case DETECT_ESCAPE_SEQUENCE:
        /* Change the H_Mode from ONLINEDATAMODE to ONLINECMDMODE */
        break;

    case GO_ON_HOOK:
        /* Bring AFE to on-hook state */
        go_on_hook();
        break;

    case GO_OFF_HOOK:
        /* Bring AFE to off-hook state */
        go_off_hook();
        break;

    case MODEM_RESET:
        /* Reset the DP status by invoking the Data Pump
        /initialization function ModemChannelActivate */
        break;

    case START_ALB_MODE:
        /* Start the DP in loop back mode */
        break;

    case START_ANSWER_MODE:
        /* Start the DP in answer mode */
        break;

    case START_ORG_MODE:
        /* Start the DP in originate mode with the dial string
        /indicated in the parameter buffer mode */
        break;

    case MODEM_HANGUP:
        /* Terminate the modem connection by issuing
        /mHANGUP message to DP */
        break;
```

dsPIC30F Soft Modem Library User's Guide

```
case SET_H_MODE:
    /* Set the requested H_Mode mode */
    break;

case MODEM_CONFIG:
    /*Configure the DP with the following parameters */

    /*Set Transmit Level (-dBm) (mSETTXLEVEL) - ATS91*/
    DPTxLevel = (unsigned char)MsgParams[0];

    /*Set the desired soft-modem hand shake mode
    /(mHSMODESELECT) - AT+MS */
    DPHSMODE = (unsigned char)MsgParams[1];

    /*Select the desired connect rate (mRATESELECT) - AT+MS */
    DPCnxRate[0] = MsgParams[2];
    DPCnxRate[1] = MsgParams[3];

    /*Enable/Disable dial tone detection
    /(mDETECTDIALTONE) - ATXn */
    CPDetectDialTone = (unsigned char)MsgParams[4];

    /*Enable/Disable busy tone detection
    /(mCLPRGSELECT) - ATXn */
    CPDetectBusyTone = (unsigned char)MsgParams[5];

    /*Maximum Dial tone detection time = ATS6 */
    CPDialTonePeroid = (unsigned char)MsgParams[6];

    /*Maximum DP connection establishment time - ATS7 */
    DPConnectPeriod = (unsigned char)MsgParams[7];

    /*Enable/Disable V.42 error correction - AT&Qn */
    SMV42Enable = (unsigned char)MsgParams[8];

    /*Select the data flow control type - AT&Kn */
    SMFlowControl = (unsigned char)MsgParams[9];
    break;

case CLPRG_SPEAKER_VOLUME:
    /*Change the volume of the call progress speaker
    /to the required level*/

    default:
        break;
```

3.5.4 AT Layer API Checklist

Following is a summary of the actions/functions that must be implemented by the SMUA to use the services of the AT command layer.

1. Initialize the AT command layer by calling the AT layer export function `InitATCommands` (see Section 3.5.2).
2. Invoke the AT command layer processing function `ProcessATCommands` at frame boundary with the proper arguments (see Section 3.5.2).
3. The SMUA have to invoke the AT command layer export function `DPIndicateStatus` to indicate the data pump status with the ID's as specified (see Section 3.5.2)
4. The SMUA have to provide an import function `ATIndicateStatus` for AT command layer. AT command layer calls this function to request certain operations to be performed by the SMUA. The SMUA have to take appropriate action based on the ID's as declared (see Section 3.5.3).
5. The SMUA layer have to provide an import function `ATSendMsgToDP` for the AT command layer. The AT command layer calls this function to send some data pump messages to the data pump layer. The SMUA have to pass the messages sent by the AT command layer to the data pump layer (see Section 3.5.3).
6. The SMUA layer have to provide an import function for `ATSendStringToUART` for the AT command layer. The AT command layer calls this function to send character strings to UART layer. The SMUA have to pass the string received by the AT command layer to the UART (see Section 3.5.3).

3.6 EMBEDDED SOFT MODEM API

The Embedded Soft Modem configuration eliminates the AT command layer and links the application data to defined variables declared in memory. The embedded configuration typically requires less program and data memory by precluding the resources otherwise required for the AT command layer.

For the embedded configuration the SMUA is split into two layers. An inner layer (SMUA_IL) interacts with the data pump and V.42 layers and deals with most of the necessary communication requirements. An outer layer (SMUA_OL) communicates with the inner layer to provide configurable parameters and pass data in and out.

Splitting the SMUA in this manner typically eliminates the need for dealing with all the interface details of the SMUA. Instead, you only need to deal with the outer layer requirements in relation to your application.

However some embedded applications require you to interact with the inner layer depending on the application requirements (for example, to configure the V.42 layer for different error tests).

dsPIC30F Soft Modem Library User's Guide

3.6.1 Interface Structures

The following structure is defined for SMUA_OL to provide configurable parameters to the SMUA_IL.

```
typedef struct {  
    unsigned char DPTxLevel;  
    unsigned char SpeakerVolume;  
    unsigned char SMMode;  
    unsigned char DialString[20];  
    unsigned char DPMoMode;  
    unsigned int  MaxCnxRate;  
    unsigned char V42Enable;  
}SM_CONFIG;
```

TABLE 3-7: CONFIGURABLE PARAMETERS FOR SM_CONFIG STRUCTURE

Parameter	Description																											
DPTxLevel	Data pump transmit level in terms of -dBm.																											
SpeakerVolume	Indicates the speaker volume (1,2 or 3)																											
SMMode	Indicates the Soft Modem mode of connection 0 - LoopBack 1 - Originate Mode 2 - Answer Mode																											
DialString	Contains the dial digits of the remote modem to dial.																											
DPMoMode	Modulation Mode of the data pump. <table><thead><tr><th>Handshake</th><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>ENABLE_AUTO</td><td>0</td><td>Automatically selects optimum data modulation for both local and remote modems. (Starts with V.8)</td></tr><tr><td>ENABLE_V8</td><td>1</td><td>Enable V.8 handshake mode</td></tr><tr><td>ENABLE_V32BIS</td><td>2</td><td>Enable V.32bis modem handshake</td></tr><tr><td>ENABLE_V32</td><td>4</td><td>Enable V.32 modem handshake</td></tr><tr><td>ENABLE_V22</td><td>8</td><td>Enable V.22 modem handshake</td></tr><tr><td>ENABLE_V21</td><td>16</td><td>Enable V.21 handshake</td></tr><tr><td>ENABLE_V23</td><td>32</td><td>Enable V.23 handshake</td></tr><tr><td>ENABLE_B103</td><td>64</td><td>Enable Bell 103 handshake</td></tr></tbody></table>	Handshake	Value	Description	ENABLE_AUTO	0	Automatically selects optimum data modulation for both local and remote modems. (Starts with V.8)	ENABLE_V8	1	Enable V.8 handshake mode	ENABLE_V32BIS	2	Enable V.32bis modem handshake	ENABLE_V32	4	Enable V.32 modem handshake	ENABLE_V22	8	Enable V.22 modem handshake	ENABLE_V21	16	Enable V.21 handshake	ENABLE_V23	32	Enable V.23 handshake	ENABLE_B103	64	Enable Bell 103 handshake
Handshake	Value	Description																										
ENABLE_AUTO	0	Automatically selects optimum data modulation for both local and remote modems. (Starts with V.8)																										
ENABLE_V8	1	Enable V.8 handshake mode																										
ENABLE_V32BIS	2	Enable V.32bis modem handshake																										
ENABLE_V32	4	Enable V.32 modem handshake																										
ENABLE_V22	8	Enable V.22 modem handshake																										
ENABLE_V21	16	Enable V.21 handshake																										
ENABLE_V23	32	Enable V.23 handshake																										
ENABLE_B103	64	Enable Bell 103 handshake																										
MaxCnxRate	Maximum connect rate in the selected modulation mode. This can take on the values as follows – 75, 300, 1200, 2400, 4800, 7200, 9600, 12000, 14400																											

3.6.2 Interface Functions

The following functions are provided in the `SMUA_IL.c` source file.

- `void Configure_SoftModem(SM_CONFIG *)`
- `unsigned char Start_SoftModem(void)`

These functions are provided in the `SMUA_OL.c` source file as call back functions by the inner layer:

- `Int Get_SM_TransmitData(unsigned char *TxBuffer)`
- `void Put_SM_ReceiveData(unsigned char *RxBuffer, int Len)`
- `void SoftModem_Connect(unsigned char ConnectType, unsigned char ModeMode, unsigned int BitRate)`
- `unsigned char Poll_SMUA_OL_Status(void)`

3.6.2.1 FUNCTION DESCRIPTIONS

void Configure_SoftModem(SM_CONFIG *)

This function must be called by the `SMUA_OL` with desired configurable parameters. After calling this function the `SMUA_IL` is ready to start the soft modem with the configured parameters.

unsigned char Start_SoftModem(void)

This function is used to start the soft modem handshake in the desired mode. This function returns only if either user (`SMUA_OL`) requested for the call termination or the connection termination due to some reason. This function returns under the following conditions. The return value will take one of the following values.

Return Flag	Value
<code>NO_DIAL_TONE</code>	<code>0x1</code>
<code>DETECT_BUSY_TONE</code>	<code>0x2</code>
<code>CONNECTION_TIMEOUT</code>	<code>0x4</code>
<code>HANGUP_COMPLETE</code>	<code>0x8</code>
<code>LOST_REMOTE_CARRIER</code>	<code>0x10</code>
<code>REM_V42_DISCONNECT</code>	<code>0x20</code>
<code>V42_PROTOCOL_ERROR</code>	<code>0x40</code>

User can initiate the call termination using the function `Poll_SMUA_OL_Status`.

Int Get_SM_TransmitData(unsigned char *TxBuffer)

This function has to be provided by the `SMUA_OL` and is called by the `SMUA_IL` to get the data to be transmitted. The data to be transmitted should be updated into `TxBuffer`.

If `SMUA_OL` communicates with the UART for getting the transmit data, then it is `SMUA_OL` responsibility to maintain the data queue (circular) and data flow control with the UART. The user has to copy the data from the above mentioned circular buffer to the `TxBuffer`. Currently the maximum size V.32bis, V.22 and V.23 1200 is 20 bytes and for V.23-75 and V.21 mode it is 1 byte.

void Put_SM_ReceiveData(unsigned char *RxBuffer, int Len)

The SMUA_IL calls this function to provide the data received to the SMUA_OL. The SMUA_OL shall send this data to the UART or buffer the data based on the requirement. User has to maintain the circular buffer and stuff the data from the RxBuffer to the circular buffer to send to the UART.

void SoftModem_Connect(unsigned char ConnectType,unsigned char ModeMode,unsigned int BitRate)

This function is called from the SMUA_IL after the connection establishment to indicate the type/mode of connection. The three values can take on the following values.

ConnectType

- DP_CONNECT (0x1)
- NON_LAPM_CONNECT (0x2)
- LAPM_CONNECT (0x4)
- RETRAIN_CONNECT (0x8)

ModulationMode

- V32BIS_MODE (0x1)
- V22BIS_MODE (0x2)
- V23_MODE (0x4)
- V21_MODE (0x8)
- B103_MODE (0x10)

BitRate

- 75, 300, 1200, 2400, 4800, 7200, 9600, 12000, 14400

unsigned char Poll_SMUA_OL_Status(void)

This function is called by the SMUA_IL every 5 ms (every PCM frame of 40 samples) to check for the call termination request. Returns a flag '1' to terminate the call, otherwise returns '0'. When this function is called, the SMUA_OL can perform the flow control checking for the UART also.

Chapter 4. DTMF API

4.1 INTRODUCTION

This chapter provides API specifications for standalone DTMF Generation and Detection Modules that are included on the dsPIC30F Soft Modem Library CD-ROM.

4.2 HIGHLIGHTS

These topics are covered in this chapter:

- Overview
- DTMF API Specifications

4.3 OVERVIEW

The DTMF generation and detection modules have been developed in accordance with *ITU-T Q.23 Technical features of push-button telephone sets* and *ITU-T Q.24 Multi-frequency push-button signal reception* recommendations. Full source code is provided to fulfill two objectives:

- To help jump start your user applications with plug-in code
- To facilitate customization to satisfy your specific requirements.

To help meet these objectives, two sample application programs running on the dsPIC30F device are examined in **Chapter 5. “Soft Modem Demonstrations”**. These simple programs demonstrate:

- Local DTMF generation and detection in loopback mode
- Remote DTMF detection over a dial-up phone line

The dsPICDEM.net Connectivity Board is required for these demonstration programs. Also the demonstration programs must be built in MPLAB IDE and loaded into the dsPIC30F device.

4.3.1 DTMF Loopback Demonstration

The DTMF loopback demonstration illustrates both DTMF generation and detection modules running on the dsPIC30F6014 device. The generated DTMF signal (a single digit) is sent to the Si3034/Si3035 DAA/AFE telephone interface on the dsPICDEM.net board. The DAA/AFE internally loops the generated digit back to the dsPIC30F6014, where it is interpreted by the DTMF detection module. The dsPIC30F6014 then sends the recognized digit to both the LCD and UART on the board. The LCD displays the digit. The UART send the digit to the attached PC or laptop running Hyper Terminal.

4.3.2 Phone Line Demonstration

The phone line demonstration illustrates DTMF detection from a remote source over a PSTN analog line. The dsPICDEM.net board is dialed from a telephone. The Si3034/Si3035 DAA/AFE telephone interface detects the ring signal and alerts the dsPIC30F6014, which initializes the program. OFF HOOK is simulated by pressing the S! switch on the board. After that, the phone pad is used to generate random digits, which are detected, interpreted and displayed on the LCD.

Procedures for working through these demonstrations are presented in **Section 5.7 “DTMF Loopback Demo”** and **Section 5.8 “DTMF PSTN Phone Line Demo”**.

dsPIC30F Soft Modem Library User's Guide

4.4 DTMF API SPECIFICATIONS

The DTMF generation and detection modules are required for applications such as answering machines, public and private telephone exchanges, telephony and line test equipment and remote control of computer and telephone equipment. Table 4-1 lists the data memory, program memory and MIPS needed to support DTMF generation and detection modules.

TABLE 4-1: DTMF MODULE RESOURCE USAGE

Module	Data Memory* (Bytes)	Program Memory (Bytes)	MIPS
DTMF Generation	46	484	0.3
DTMF Detection	224	2800	1.2

* Uses uninitialized general near data memory

4.4.1 DTMF Generation Module API

The DTMF generation module is implemented with these functions:

- InitDTMFGen
- DTMFGen

4.4.1.1 FUNCTION DESCRIPTIONS

InitDTMFGen

Function: Called by the user application to initialize the DTMF generation module. This function initializes all the parameters for DTMF generation and saves the DTMF digits information in a buffer.

Prototype: void InitDTMFGen(unsigned char *DigitBuf
unsigned char Length);

Arguments:

- Buffer pointer for DTMF digits to be dialed
- Number of digits in the buffer

Return Value: None

Example:

```
unsigned char DTMFString[20];
/* Initialize the DTMF generator
/to dial the number 480 792-7200 */
DTMFString[0] = 4;
DTMFString[1] = 8;
DTMFString[2] = 0;
DTMFString[3] = 7;
DTMFString[4] = 9;
DTMFString[5] = 2;
DTMFString[6] = 7;
DTMFString[7] = 2;
DTMFString[8] = 0;
DTMFString[9] = 0;
InitDTMFGen(&DTMFString[0], 10);
```

DTMFGen

Function: Called periodically by the user application to generate the DTMF samples corresponding to the digits set in the initialization function. This function returns the state of the DTMF generation function. Currently this function produces 40 samples per frame (5 ms @ 8 KHz sampling rate). The following states are defined.

State	ID	Description
DTMFSILENCE	1	DTMF generator is transmitting silence
DTMFSIGNAL	2	DTMF generator is transmitting valid DTMF signal
DTMFEND	4	Indicates the end of DTMF generation

This function should be invoked by the application until it receives DTMFEND state.

Prototype: `char DTMFGen(int *SampleBuffer);`

Arguments: Buffer pointer to put the samples

Return Value: DTMF State

Example:
`int DTMFTxBuffer[40];
State = DTMFGen(&DTMFTxBuffer[0])`

4.4.2 DTMF Detection Module APIs

The DTMF Detection module is implemented with these functions:

- `DTMFInit`
- `DTMFDetection`

4.4.2.1 FUNCTION DESCRIPTIONS

DTMFInit

Function: Initializes the parameters for DTMF detection. The user application must call this function before invoking the `DTMFDetection` function.

Prototype: `void DTMFInit(void);`

Arguments: None

Return Value: None

Example: `DTMFInit();`

DTMFDetection

Function: This function should be invoked by the user application every 10 ms to process the received samples for DTMF detection. Returns the state of the DTMF detection. The user application reads the digit based on the state information returned by the `DTMFDetection` function. Following possible states are returned:

Frame Type	Symbolic Representation	Description
0	<code>VALID_DIGIT_FRAME</code>	A valid DTMF digit just detected.
1	<code>POSSIBLE_DIGIT_FRAME</code>	This signal may be representing a DTMF tone (first detection), yet to complete the final decision.
2	<code>DIGIT_DETECTED</code>	A valid digit is detected.
3	<code>TONE_SHAPE_TEST</code>	This frame indicates the silence period immediately after a DTMF digit (may be separation between two digits).
4	<code>PAUSE_AFTER_DIGIT_FRAME</code>	Pause frame after a DTMF tone.
5	<code>BUFFER_DELAY</code>	No operation frame (the input samples are taken and stored in the system buffer).
6	<code>NOT_A_DIGIT_FRAME</code>	This frame is definitely not a part of a DTMF digit.
-1	<code>ERROR_FRAME</code>	Any error in the frame processing (possibly wrong arguments).

Prototype: `int DTMFDetection(int inputType, int *inSignal, int *Digit);`

- Arguments:**
- Incoming signal is 16-bit left justified or 14-bit right justified.
 '0' indicates 14-bit right-justified signal.
 '1' indicates 16-bit left-justified signal.
 - Pointer to the input samples buffer
 - Pointer to store the detected digit

Return Value: Process State/Type

Example:

```
int inputType = 1;
int inSignal[80];
int Digit, State;
State = DTMFDetection(inputType, &inSignal[0], &Digit);
```

Chapter 5. Soft Modem Demonstrations

5.1 INTRODUCTION

This chapter examines sample application programs that demonstrate key functionality provided in the dsPIC30F Soft Modem Library. These simple demonstrations offer hands-on exposure to the library and provide an informal way to gain implementation experience

These demonstration programs use MPLAB IDE and MPLAB ICD 2 for programming and the dsPICDEM.net development board for program execution.

5.2 HIGHLIGHTS

These topics are covered in this chapter:

- Overview and Purpose
- Demo Projects
- AT Command Modem Demo
- Embedded Modem Demo
- DTMF Loopback Demo
- DTMF PSTN Phone Line Demo

5.3 OVERVIEW AND PURPOSE

The sample programs described in this chapter are an informal way of showing how the dsPIC30F device supports applications that incorporate a soft modem. Although simplistic in scope, these programs are a vehicle for hands-on experience. Working through these demonstrations will increase your comfort level not only with the soft modem library but also with the dsPIC device and its software development environment.

Included are these representative applications:

- AT command controlled soft modem demonstration
- Embedded API soft modem demonstration
- Local DTMF generation and detection in a loopback connection
- DTMF detection over a PSTN analog phone line

After working with these demos you should have:

- Increased your understanding of how the dsPIC30F device interacts with its peripherals
- Gained a further understanding of how to initialize and control dsPIC30F peripherals and associated dsPICDEM.net board functions
- Reduced the learning cycle for getting started with your software development using the dsPIC30F Soft Modem Library.

5.3.1 AT Command Controlled Soft Modem Demonstration

This demonstration illustrates the ability of the dsPIC30F Soft Modem to transmit and receive data over the PSTN. The program implements real-time ITU-T V.22bis Data Pump Modulations. Two hardware configurations are used. In the first configuration two dsPICDEM.net boards use the soft modem. In the second configuration, one board uses the dsPIC30F soft modem while the other board uses a standard reference modem.

5.3.2 Embedded V.22 Soft Modem Demonstration

In this demonstration, soft modem control is embedded in the interface to the user application. This application eliminates the AT command layer and its required hardware and software, and links the user application to defined variables embedded in dsPIC30F memory.

5.3.3 DTMF Loopback Demonstration

The DTMF loopback demonstration illustrates DTMF generation and detection modules running on the dsPIC30F6014 device. The generated DTMF signal (a single digit) is sent to the Si3034/Si3035 DAA/AFE telephone interface on the dsPICDEM.net board. The DAA/AFE internally loops the generated digit back to the dsPIC30F6014, where it is interpreted by the DTMF detection module. The dsPIC30F6014 then sends the recognized digit to both the LCD and UART on the board. The LCD displays the digit. The UART sends the digit to the attached PC or laptop running Hyper Terminal.

5.3.4 DTMF Phone Line Demonstration

The phone line demonstration illustrates DTMF detection from a remote source over a PSTN analog line. The dsPICDEM.net board is dialed from a telephone. The Si3034/Si3035 DAA/AFE telephone interface detects the ring signal and alerts the dsPIC30F6014, which initializes the program. OFF HOOK is simulated by pressing the S! switch on the board. After that, the phone pad is used to generate random digits, which are detected, interpreted and displayed on the LCD.

5.4 DEMO PROJECTS

The dsPICDEM.net Connectivity Board is required for these demonstration programs. The MPLAB ICD 2 is used to build the program, program the dsPIC30F6014 device and examine specific attributes of the demonstrated functionality. You can also use the MPLAB ICD 2 as a debugger if you decide to modify the code.

Full source code is provided for the demonstration programs. If you choose to rebuild the demos and you have not purchased the full MPLAB C30 Compiler, you can download and install the free 60-day trial version from the Microchip web site.

There are four major tasks involved in working with the demonstration programs:

1. Create a demo project and workspace in MPLAB IDE

In this step you open a project, specify the dsPIC30F device to be used, identify and locate the programming tool, add the necessary source and support files and then name the project.

2. Build the program

In this step the source and support files are combined with a linker script file (`p30f6014.gld`) and a header file (`p30f6014.h`) from the MPLAB C30 compiler to build the program.

3. Program the dsPIC30F device

In this step you configure the dsPIC30F device for the program and then program the chip.

4. Run the demo (and, optionally, modify or debug the code)

In this step you explore the specific capabilities being demonstrated, decide if they are appropriate for your application and determine the best way to build this capability into your application.

Before starting, copy both the `DTMF Demo` and `V.22bis Modem_R1.0` folders from the dsPIC30F Soft Modem Library CD to the `C:\` drive.

Note: Files copied from the CD are read only; you will need to change the attributes of files that need to be edited.

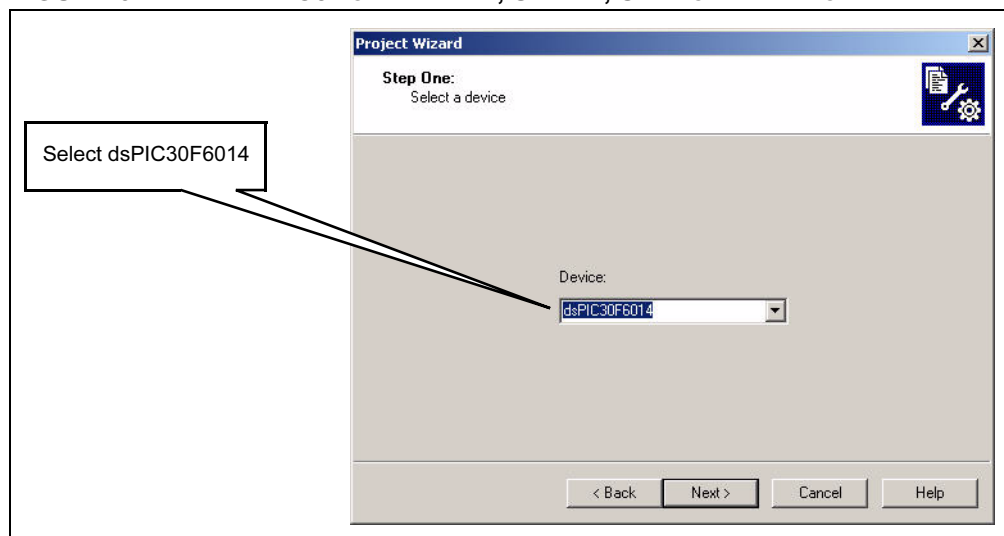
5.4.1 Creating the Project

The first step is to create a project and a workspace in MPLAB IDE. The project contains the source and support files needed to build an application along with their associations to the software development tools being used. It also specifies the device to be programmed and the device configuration best suited for the program. MPLAB IDE contains a Project Wizard to help create new projects.

5.4.1.1 SELECT THE dsPIC30F DEVICE

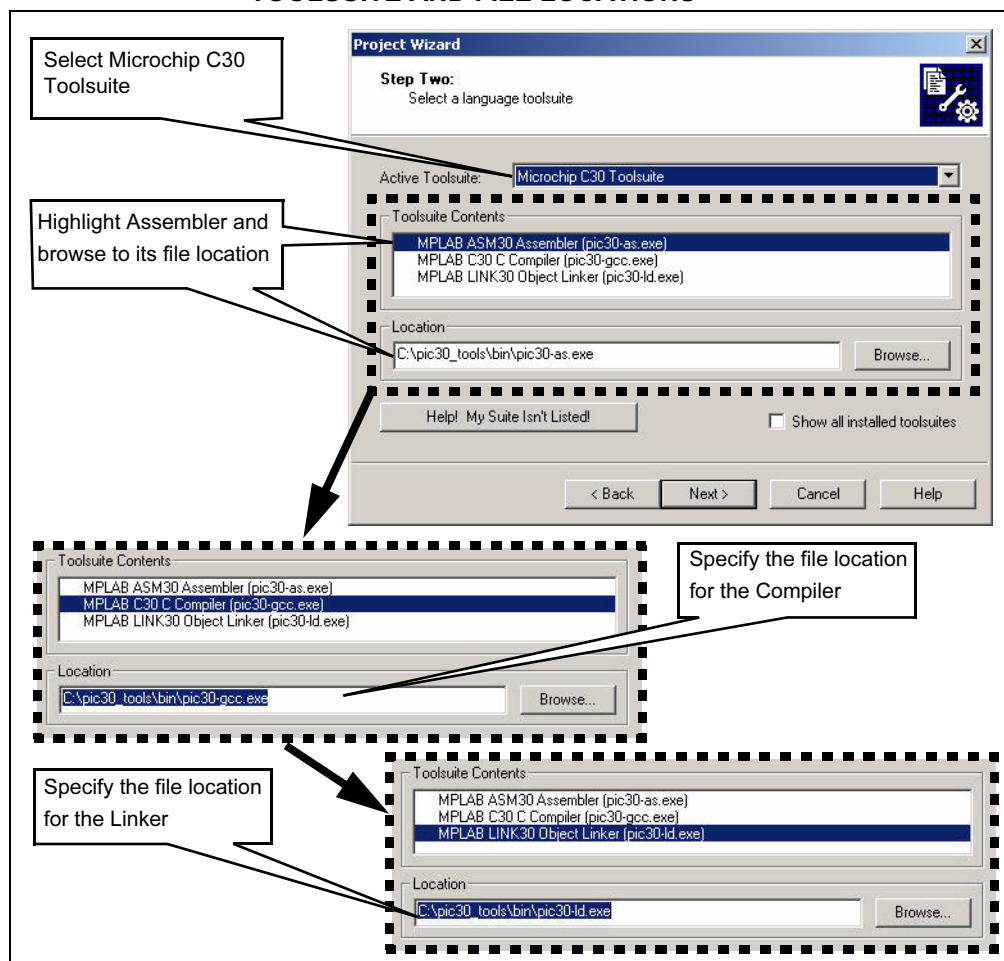
1. Start MPLAB IDE.
2. Close any workspace that might be open (*File>Close Workspace*).
3. From the Project menu, select *Project Wizard*.
4. From the Welcome screen, click the **Next>** to display the Project Wizard Step One dialog (see Figure 5-1).

FIGURE 5-1: PROJECT WIZARD, STEP 1, SELECT A DEVICE



5. Select dsPIC30F6014 as the device and click **Next>**. The Project Wizard Step Two dialog displays (see Figure 5-2).

FIGURE 5-2: PROJECT WIZARD, STEP 2, SELECT LANGUAGE TOOLSUITE AND FILE LOCATIONS



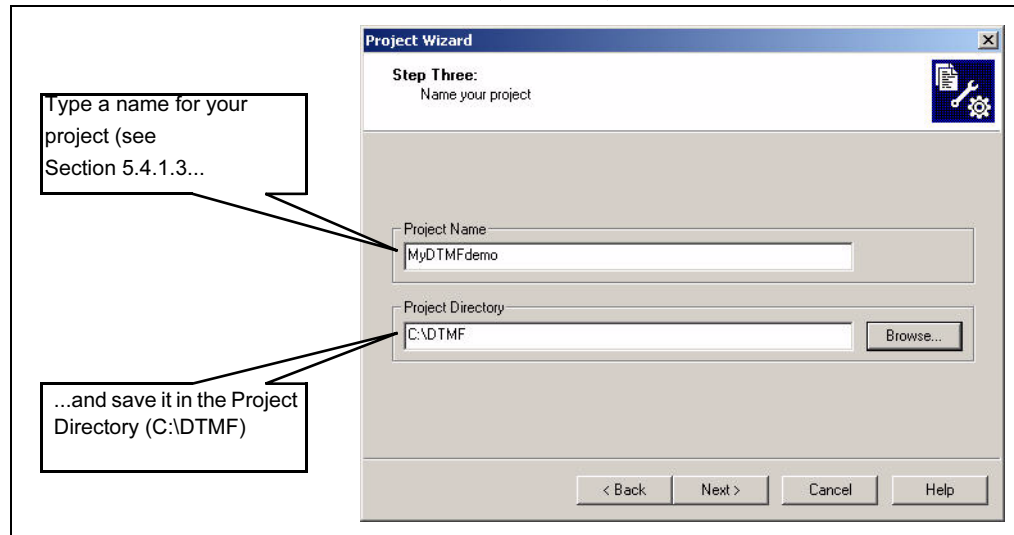
5.4.1.2 SELECT LANGUAGE TOOLSUITE AND FILE LOCATIONS

1. From the **Active Toolsuite** pull-down menu, select Microchip C30 Toolsuite. This toolsuite includes the compiler, assembler and linker that will be used.
2. From **Toolsuite Contents**, select MPLAB ASM 30 Assembler (pic30-as.exe).
3. In the **Location** group, click **Browse...** and navigate to
C:\pic30_tools\Bin\pic30-as.exe

Note: C:\ is the drive implemented for this example. The specific location of the MPLAB C30 compiler may be different on your system. If you haven't purchased the MPLAB C30 compiler, you can download a full-featured 60-day trial version from the Microchip web site (www.microchip.com). Follow the *dsPIC Development Tools and Compilers* link under *dsPIC® Signal Controllers*.

4. From **Toolsuite Contents**, select MPLAB C30 Compiler (pic30-gcc.exe).
5. In the **Location** block, click **Browse...** and navigate to
C:\pic30_tools\Bin\pic30-gcc.exe
6. From **Toolsuite Contents**, select MPLAB LINK30 Object Linker (pic30-ld.exe).
7. In the **Location** block, click **Browse...** and navigate to
C:\pic30_tools\Bin\pic30-ld.exe
8. Click **Next>**. The Project Wizard Step Three dialog displays (see Figure 5-3).

FIGURE 5-3: PROJECT WIZARD, STEP 3, NAME YOUR PROJECT



5.4.1.3 NAME YOUR PROJECT

1. In the **Project Name** text box, type a project name. Use one of these suggested names:

<u>For this demo:</u>	<u>Use this name:</u>
AT Modem	MyATModemDemo
Embedded modem	MyEmbeddedDemo
DTMF Loopback	MyDTMFLoopbackDemo
DTMF Phone	MyDTMFPhoneDemo.
2. Click **Browse...** and navigate to the corresponding folder where you copied the demo files from the dsPIC30F Soft Modem Library CD-ROM.
3. Click **Next>** to continue.

dsPIC30F Soft Modem Library User's Guide

5.4.1.4 ADD FILES TO PROJECT

1. On the Project Wizard Step Four dialog (see Figure 5-4), locate the [Demo] folder and add the files shown in Table 5-1 to the project (right window).

TABLE 5-1: SOURCE AND SUPPORT FILES ADDED FOR DEMO PROGRAMS

AT Modem	Embedded Modem	DTMF Loopback	DTMF Phone
Source agc.s callprog.s carrec.s data.s dmctrl.s eqz.s filter.s hdlc.s mdmkern.s message.s modemvar.s psf.s sbar.s signal.s sipl.s tables.s toned.s v22bis.s v23.s v21.s v8.s v25.s rtd.c dtmfgen.s dtmf tables.s API API.c ATCMD At_Cmds.c V.42 V42bf.c V42if.c V42m.c V42main.c V42n.c V42pf.c AFE dci_init.s dci_isr.s init.s si3021_init.s UART init_uart.s isr_uart1.s h options.h inc device_Fcy.inc Si3021config.inc dpconfig.inc (continued)	Source agc.s callprog.s carrec.s data.s dmctrl.s eqz.s filter.s hdlc.s mdmkern.s message.s modemvar.s psf.s sbar.s signal.s sipl.s tables.s toned.s v22bis.s v23.s v21.s v8.s v25.s rtd.c dtmfgen.s dtmf tables.s API SMUA_OL.c SMUA_IL.c V.42 V42bf.c V42if.c V42m.c V42main.c V42n.c V42pf.c AFE dci_init.s dci_isr.s init.s si3021_init.s UART init_uart.s isr_uart1.s h options.h inc device_Fcy.inc Si3021config.inc dpconfig.inc (continued)	h OPTIONS.H inc device_Fcy.inc si3021config.inc Source API DTMFMain.c DTMFDet Basicop.S DetTables.s Dtmfbuf.s DTMFFUNC.S DTMFIN.S DTMFGen DTMFGEN.s DTMFTables.s Hardware delay.c device_config.s traps.s AFE dci_init.s DCI_ISR.s Init.s si3021_init.s LCD lcd.c Switch INTpin_Init.s INTx_Interrupt.s UART init_uart.s isr_uart1.s	h OPTIONS.H inc device_Fcy.inc si3021config.inc Source API DTMFMain.c DTMFDet Basicop.S DetTables.s Dtmfbuf.s DTMFFUNC.S DTMFIN.S DTMFGen DTMFGEN.s DTMFTables.s Hardware delay.c device_config.s traps.s AFE dci_init.s DCI_ISR.s Init.s si3021_init.s LCD lcd.c Switch INTpin_Init.s INTx_Interrupt.s UART init_uart.s isr_uart1.s

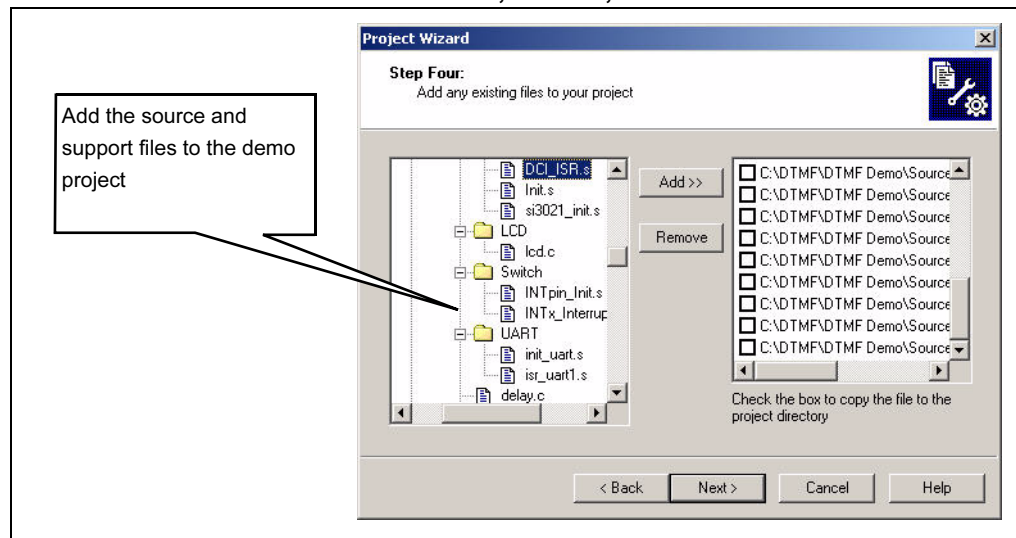
TABLE 5-1: SOURCE AND SUPPORT FILES ADDED FOR DEMO PROGRAMS (CONTINUED)

AT Modem	Embedded Modem	DTMF Loopback	DTMF Phone
hardware delay.c device_config.s traps.s lcd.c V.32bis/V.32 Support ec.s trel.s v32bis.s v32ec.s v32rr.s viterbi.s v32.inc ec.inc trel.inc vit.inc	hardware delay.c device_config.s traps.s lcd.c ADC bin2dec.c display.s init_Adc.s isr_Adc.s Switch INTpin_Init.s INTx_Interrupt.s		

1. Navigate to the C:\pic30_tools\support\gld folder and add file p30f6014.gld to include the linker script file in the project.
2. Navigate to the C:\pic30_tools\support\h folder and add file p30f6014.h to include the header file in the project.

Note: The linker script file and header file locations for your environment may be different. The location will depend on where you installed the C30 compiler.

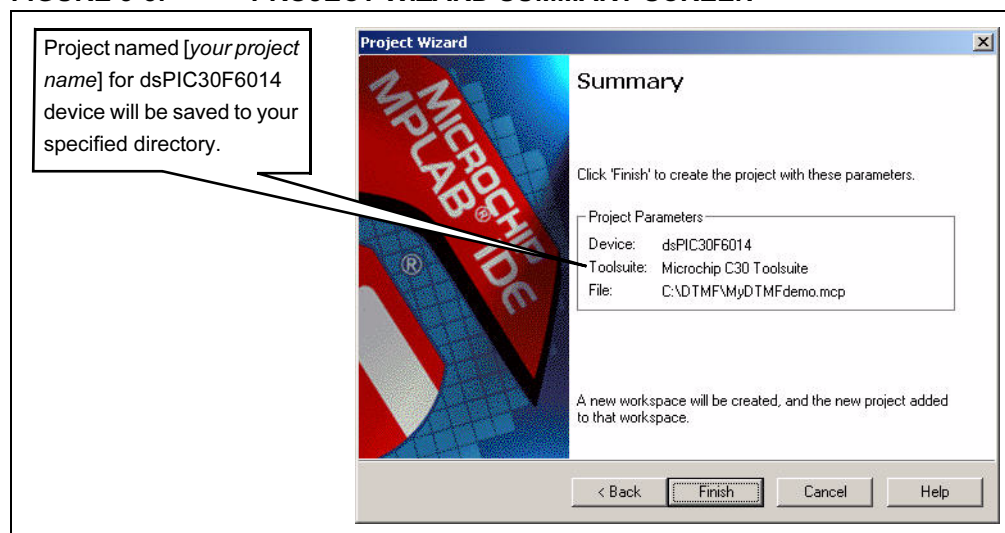
FIGURE 5-4: PROJECT WIZARD, STEP 4, ADD FILES TO PROJECT



dsPIC30F Soft Modem Library User's Guide

3. Click **Next>** to continue. The Project Wizard Summary screen (Figure 5-5) displays the parameters of this demo project.

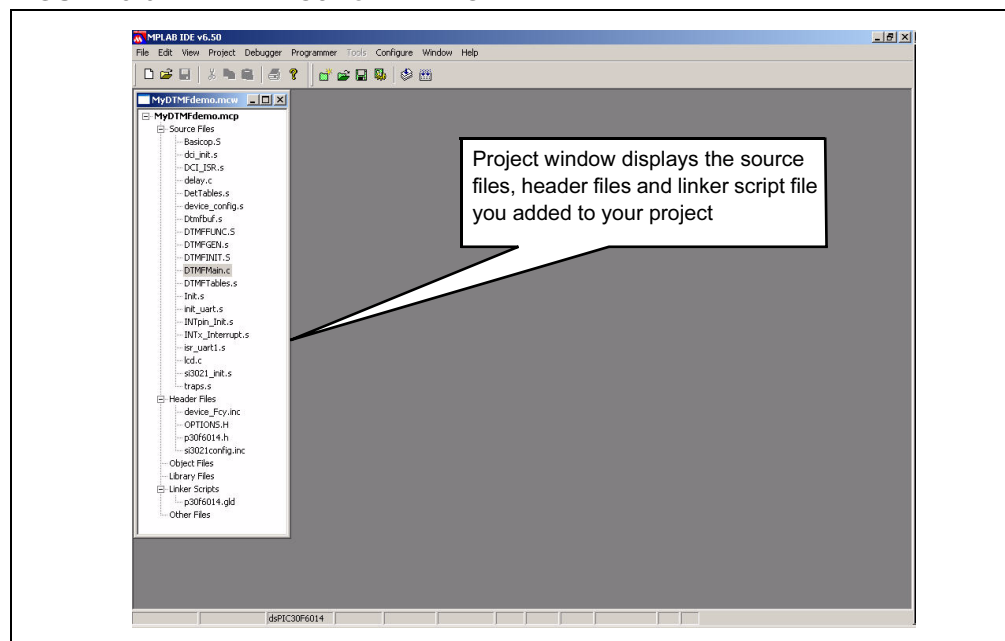
FIGURE 5-5: PROJECT WIZARD SUMMARY SCREEN



4. Click **Finish**.

After the project wizard completes, the MPLAB project window shows the project and all the added files (see Figure 5-6).

FIGURE 5-6: PROJECT WINDOW



At this point a demo project and workspace have been created in MPLAB IDE. Open and display the main demo file:

If you are running this demo:

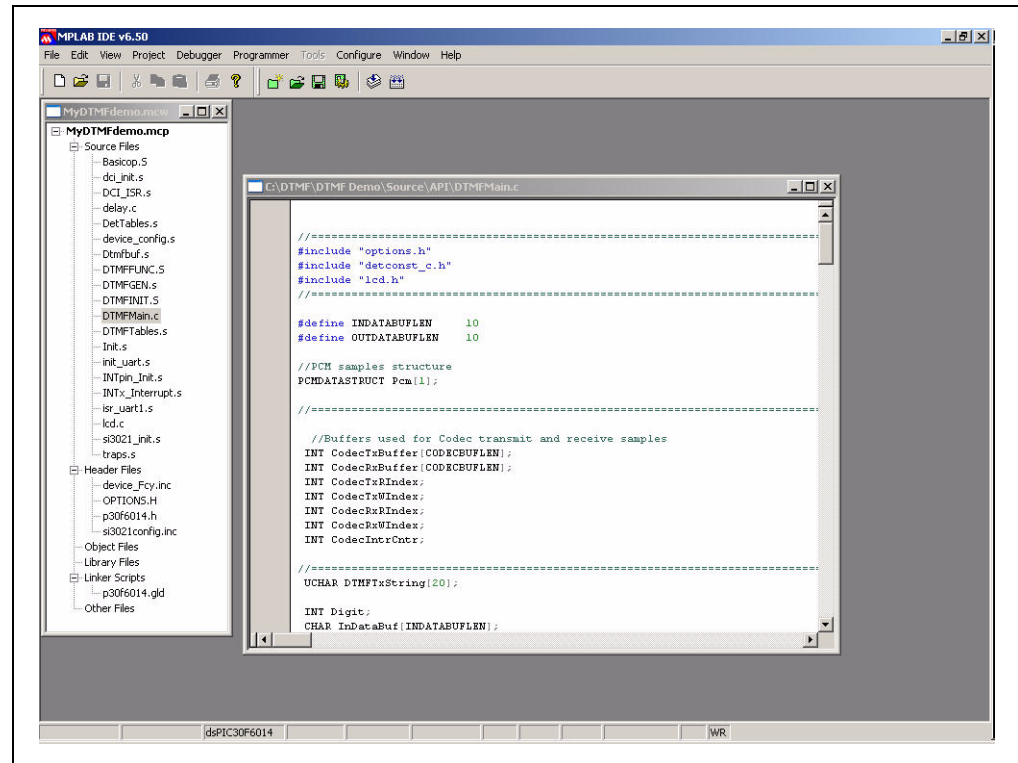
AT Modem
Embedded modem
DTMF Loopback
DTMF Phone

Double-click this file:

API.c
SMUA_OL.c
DTMFmain.c
DTMFmain.c

MPLAB project window should look similar to Figure 5-7.

FIGURE 5-7: MPLAB WORKSPACE



5.4.2 Building the Program

The program is built in two stages – first the source files are compiled into object files, then the object files are linked to build a hex output file that is used to program the dsPIC device. A `cof` output file contains additional information that lets you debug the code at the source code level.

Before building, however, you must specify compiler and linker settings. These settings indicate where to find the 'C' library files and where to reserve space for the extra debug code for the MPLAB ICD 2 (In-Circuit debugger).

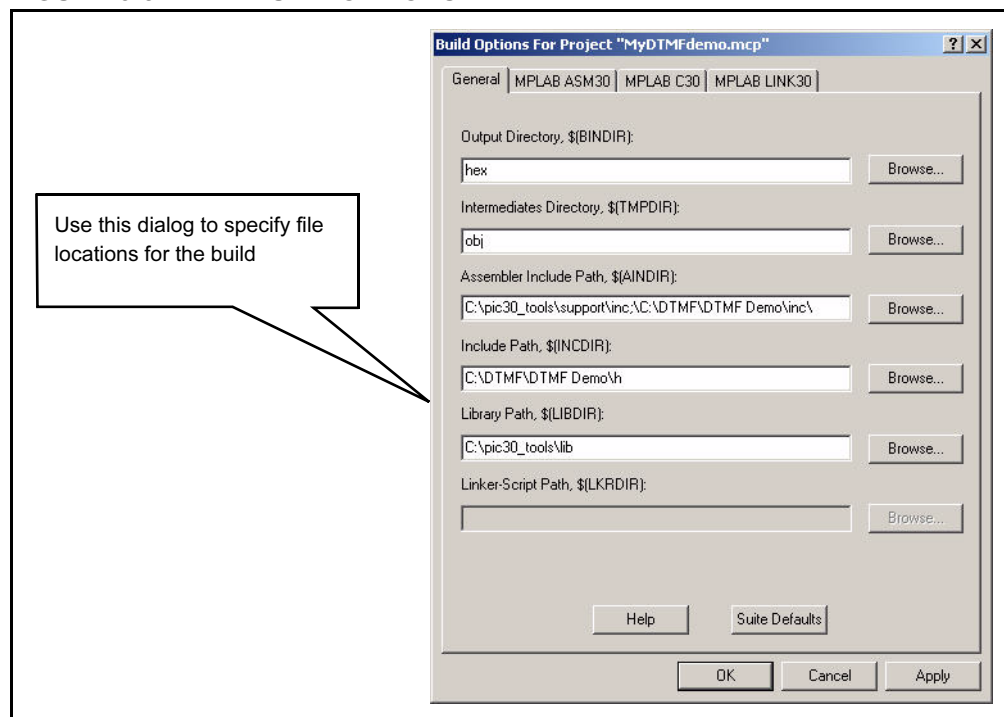
You may also need to modify the settings in the `dpconfig.inc` and `options.h` files to configure specific modem parameters. To ensure the DAA/AFE is properly configured for your country PSTN requirements, you may also need to modify settings in the `Si3021config.inc` file.

5.4.2.1 SET PROJECT BUILD OPTIONS

The demo projects do not explicitly use any libraries, but the 'C' compiler startup library code is always automatically linked into the project. Use the **Project>Build Options>Project** menu to display the Build Options dialog, as shown in Figure 5-8.

dsPIC30F Soft Modem Library User's Guide

FIGURE 5-8: BUILD OPTIONS

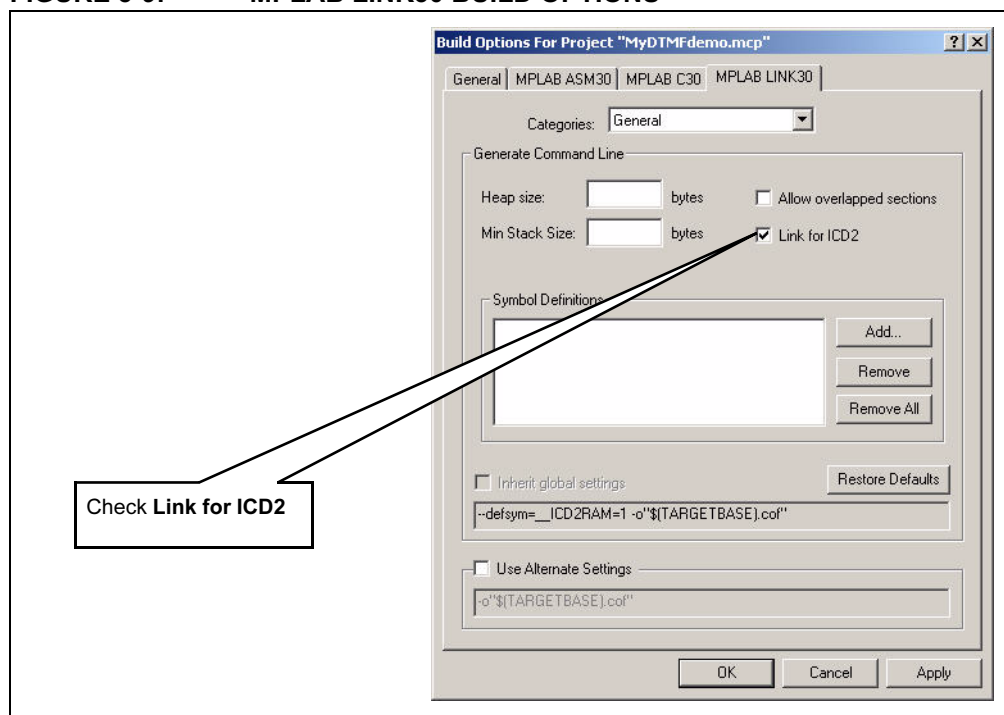


1. Select the Build Options General tab.
2. Set up the file locations as shown in Figure 5-8.

Note: The library path for your environment may be different. The location will depend on where you installed the C30 compiler.

3. Select the MPLAB LINK30 tab to display the linker settings (see Figure 5-9).

FIGURE 5-9: MPLAB LINK30 BUILD OPTIONS



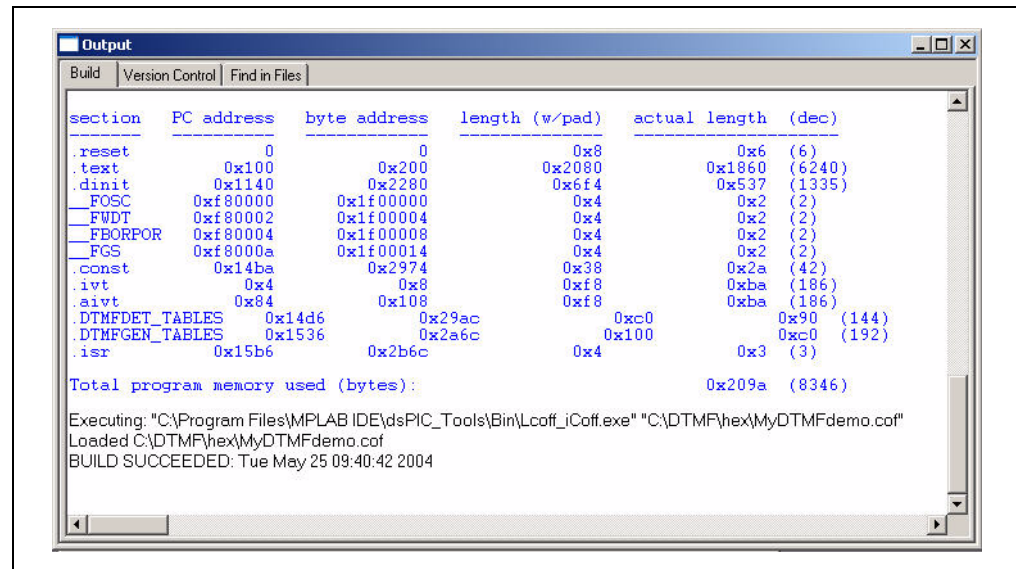
4. Check **Link for ICD2** to tell the linker to reserve space for the debug code used by the ICD 2 In-Circuit Debugger.
5. Click **OK**.

5.4.2.2 BUILD THE PROJECT

At this point the project is ready to be built.

1. From the **Project** menu select **Make**. The Build Output window displays (see Figure 5-10).
2. Observe the progress of the build.
3. When the BUILD SUCCEEDED message displays you are ready to program the device.

FIGURE 5-10: BUILD OUTPUT



If you experience any problems with the build, double check all the steps in this section and ensure that you are using the latest versions of the development tools. The latest upgrades are available on the Microchip Technology Web site:

<http://www.microchip.com>

If there are errors in the source code, you can double-click the error messages in the Output window and MPLAB will point to the offending line in the source code. This should not happen if the files were copied from the dsPIC Soft Modem CD.

5.4.3 Program the Chip

After you build the code you must program the dsPIC30F device. This process begins by configuring the chip to work with the program.

Note: Before proceeding, make sure that the USB driver for the MPLAB ICD2 has been installed on your PC (see the *MPLAB ICD 2 User's Guide* (DS51331) for more details regarding the installation of the USB driver).

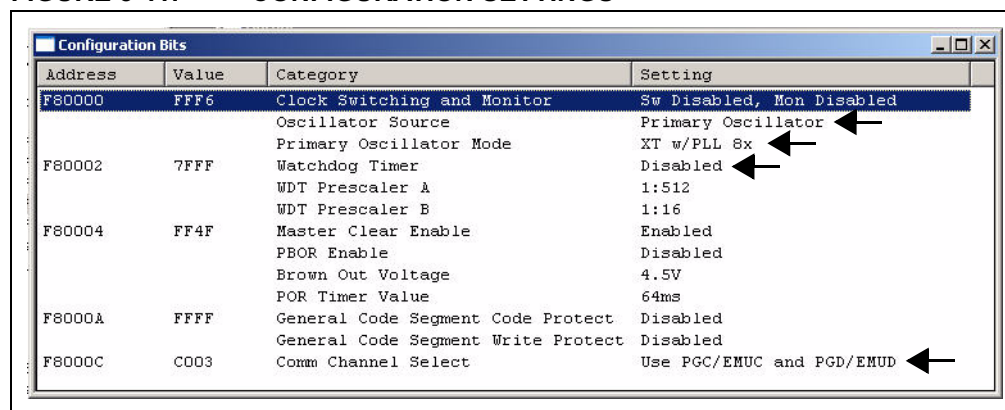
dsPIC30F Soft Modem Library User's Guide

5.4.3.1 CONFIGURE DSPIC30F DEVICE

From the *Configure* menu select *Configuration Bits* to set up the device configuration as shown in Figure 5-11. The settings that will most likely need to change are:

Oscillator Source	Primary Oscillator
Primary Oscillator Mode	XT w/PLL 8x or XT w/PLL 16x (from Device_Fcy.inc)
Watchdog Timer	Disabled
Comm Channel Select	Use PGC/EMUC and PGD/EMUD

FIGURE 5-11: CONFIGURATION SETTINGS



After building the code and setting the configuration bits, the ICD 2 debugger can be used to run and debug the code on the dsPICDEM™.net Demonstration Board. Follow the instructions in one of the next two sections depending on which tool you are using.

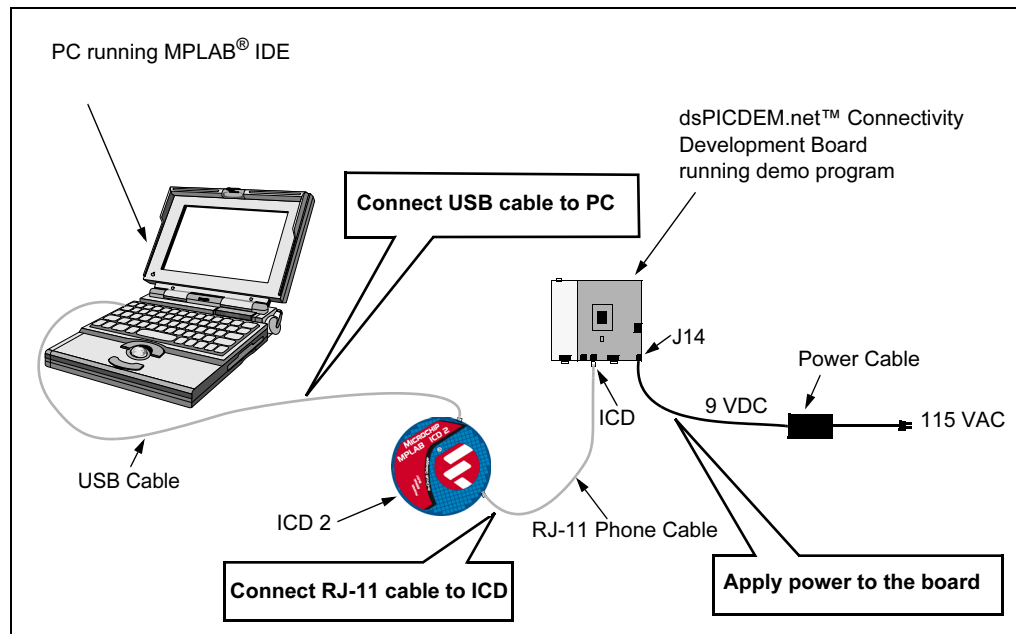
5.4.3.2 ENABLE MPLAB ICD 2 CONNECTION

The MPLAB ICD 2 can be used to program and debug the dsPIC30F6014 device in-circuit on the dsPICDEM.net board.

Note: Before proceeding, make sure that the USB driver for the MPLAB ICD 2 has been installed on your PC (see the *MPLAB ICD 2 User's Guide* (DS51331) for details regarding the installation of the USB driver).

1. Connect the ICD 2 to the PC with a USB cable (see Figure 5-12).
2. Connect the ICD 2 to modular connector labeled ICD on the dsPICDEM.net board with an RJ-11 cable.
3. Apply power to the board.

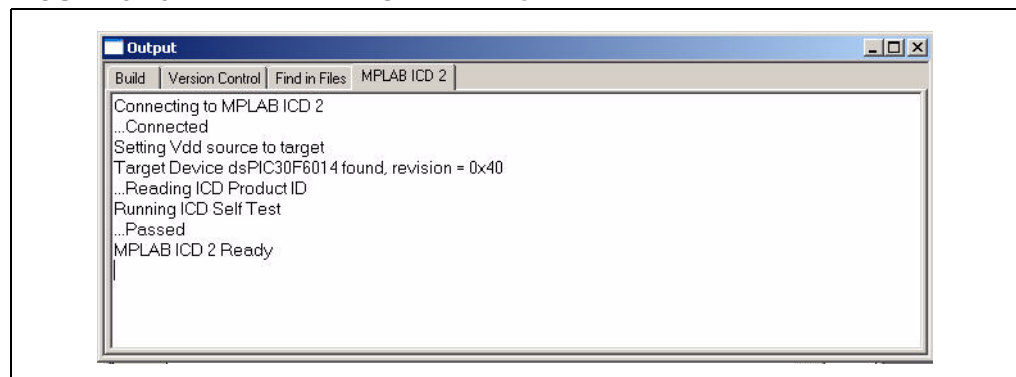
FIGURE 5-12: dsPICDEM.net™ DEVELOPMENT BOARD CONNECTED TO MPLAB ICD 2



4. From the *Debugger>Select Tool* menu, select MPLAB ICD 2 as the debug tool.
5. From the *Debugger* menu, select *Connect*. MPLAB should report that it found the dsPIC30F6014 as shown in Figure 5-13.

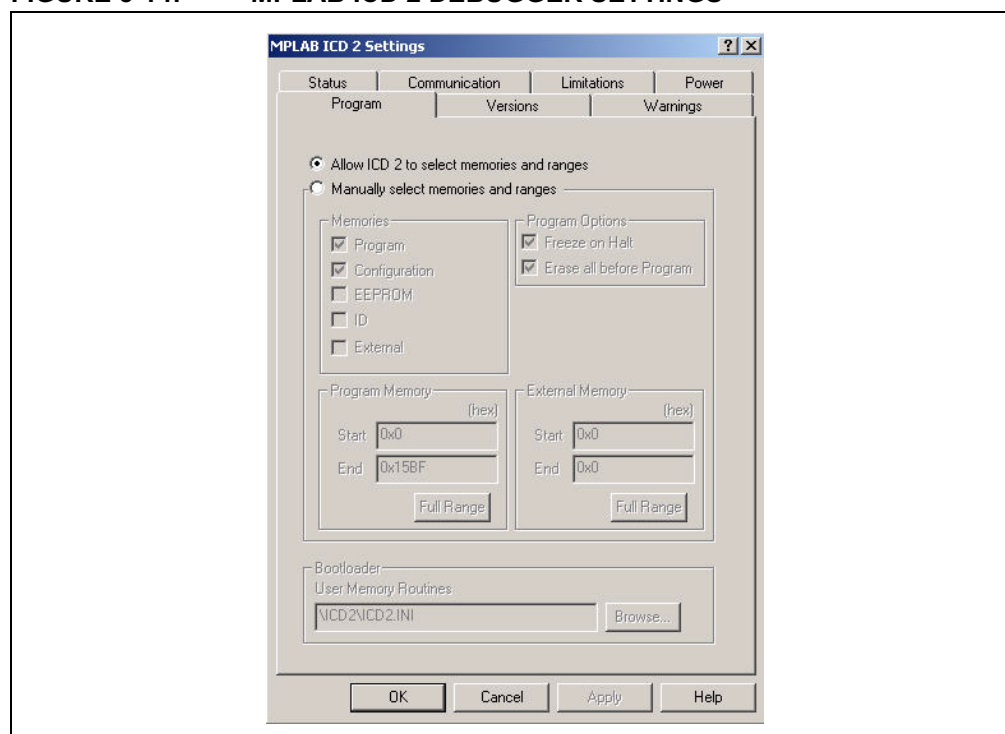
Note: MPLAB may need to download new firmware if this is the first time the MPLAB ICD 2 is being used with a dsPIC30F device. Allow it to do so. If any errors are shown, double-click the error message to get more information.

FIGURE 5-13: ENABLING MPLAB ICD 2



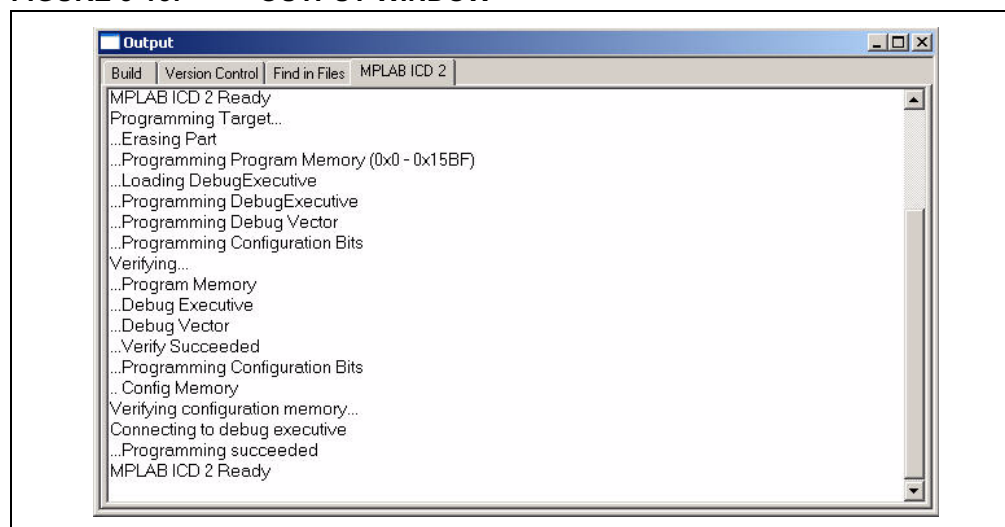
6. From the *Debugger* menu, select *Settings* to display the ICD Debugger settings (see Figure 5-14).
7. On the Program tab, ensure that **Allow ICD 2 to select memories and ranges** is selected. This setting will speed up programming by addressing only a small part of the total program memory.

FIGURE 5-14: MPLAB ICD 2 DEBUGGER SETTINGS



8. Program the part (*Debugger>Program*). The Output window shows the results of the programming cycle as shown in Figure 5-15. The part is now programmed and is ready to run.

FIGURE 5-15: OUTPUT WINDOW



9. Run the code (*Debugger>Run*). The LCD display should display the name of the demo, similar to this:

**DTMF Demo
Revision 1.0**

5.4.4 Run the Demo

After the chip is programmed you can run the demonstration program:

- AT Command Modem Demo (Section 5.5)
- Embedded Modem Demo (Section 5.6)
- DTMF Loopback Demo (Section 5.7)
- DTMF PSTN Phone Line Demo (Section 5.8)

5.4.5 Examine the Code

The MPLAB ICD 2 debugger/programmer can be used to run, halt, and step the code. You can set a breakpoint so that the execution will halt after the code has executed the instruction at the breakpoint. A green arrow points to the next line to be executed. When a breakpoint is reached the MPLAB ICD 2 halts after executing the code at the breakpoint.

The contents of the RAM and registers can be viewed when the processor has been halted. MPLAB ICD 2 uses the following function keys to access the main debugging functions:

<F5>	Halt
<F6>	Reset
<F7>	Single Step
<F9>	Run

There are more functions available by right clicking on a line of source code. The most important of these are Set Breakpoint and Run to Cursor.

Note: When debugging with ICD 2, it is always necessary to reprogram the part with the new code after each build. MPLAB will remind you with a message that states "Program memory has changed since last operation."

5.5 AT COMMAND MODEM DEMO

This demonstration illustrates V.22bis soft modem controlled by AT commands. The demonstration is set up to run either as a pair of dsPIC30F devices running the soft modem and connected end-to-end, or as a dsPIC30F device interoperatively connected to a remote terminal equipped with a standard reference modem.

Figure 5-16 illustrates the set-up configuration for the end-to-end demonstration. This configuration requires the modem software on both dsPICDEM.net boards. One modem operates in Originate mode while the other is in answer mode. The soft modem is controlled by AT commands entered on the Windows Hyper Terminal (or suitable alternative terminal emulator) running on the PC or Laptop. This demonstration requires two analog phone lines, one each for the originate and answer modems.

dsPIC30F Soft Modem Library User's Guide

FIGURE 5-16: END-TO-END DEMONSTRATION SET UP

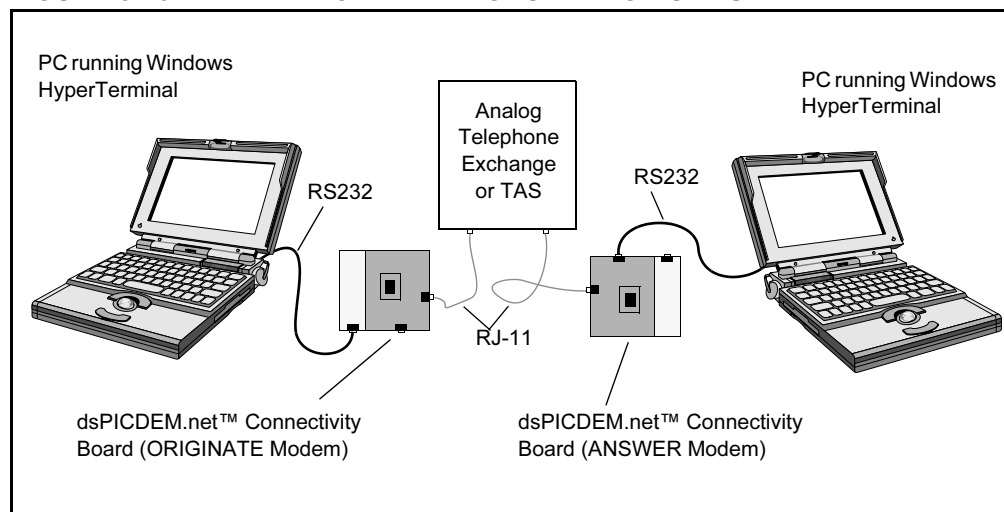
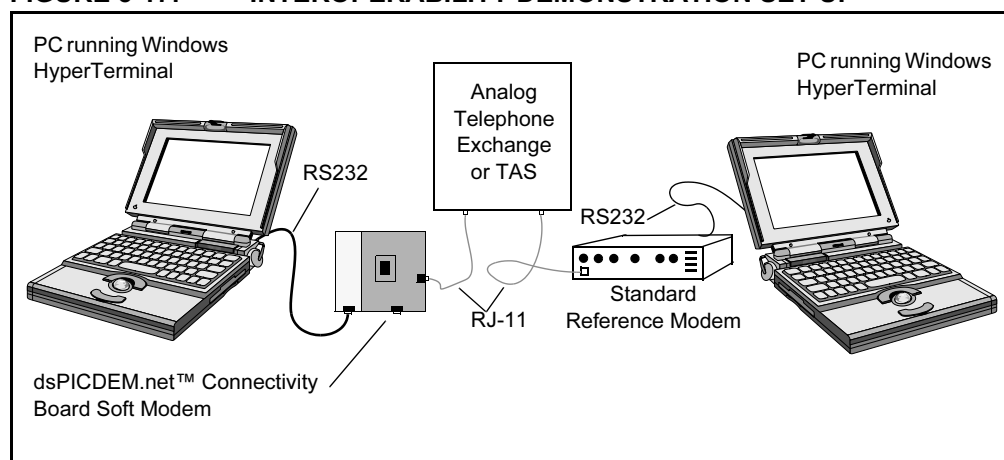


Figure 5-17 illustrates the set-up configuration for the interoperability demonstration. With this configuration, one terminal is the dsPIC30F running the soft modem. The other terminal is a PC or laptop with a standard reference modem.

The soft modem is controlled by AT commands entered on the Hyper Terminal or suitable terminal emulator program running on a PC or laptop. The other terminal is controlled by the terminal emulator on the opposite PC or laptop. This demonstration requires two analog phone lines, one each for the originate and answer modems.

FIGURE 5-17: INTEROPERABILITY DEMONSTRATION SET UP



5.5.1 Getting Started

Follow these steps to set up the AT Command Demo. These procedures assume you are familiar with the process detailed in Section 5.4 for building the program and programming the dsPIC30F6014.

1. Create the AT Command Demo project (see **Section 5.4.1 “Creating the Project”**).
2. Add the required AT Modem files into the project (see Table 5-1).
3. Configure data pump modulations in the `dpconfig.inc` file. Define AT commands and error control options in the `options.h` file (see **Section 2.5 “Setting Up Compile Time Options For Your Application”**).

4. Configure the DAA/AFE hardware using the `Si3021config.inc` file (see **Section 2.6 “Configuring the Analog Front End”**).

5. Select the correct dsPIC device speed in the `Device_Fcy.inc` file.

```
/* Set ONLY ONE of the following equates to a TRUE state */  
.equ  XTx4PLL,    0    ; do not select  
.equ  XTx8PLL,    0  
.equ  XTx16PLL,   1
```

For V.22bis modulations and lower, set `.equ XTx8PLL` to `'1'`.

For V.32bis modulations, set `.equ XTx16PLL` to `'1'`.

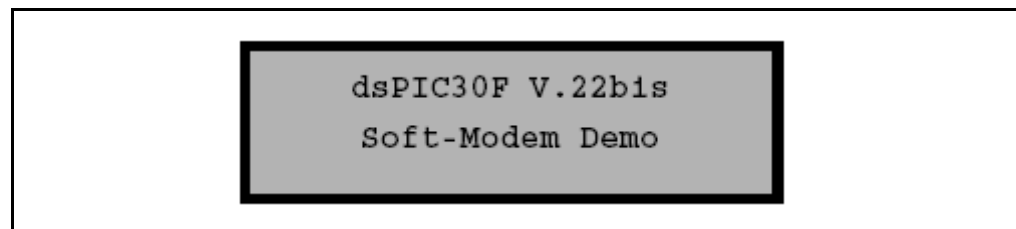
6. Build the project.
7. Download the project (program the chip).

5.5.2 Initialize the Hardware

1. Configure the demonstration for end-to-end operation or interoperability.
2. Make sure the 2x2 shunts on J17 and J19 are inserted in vertical alignment (perpendicular to the LCD display).
3. Apply power to the dsPICDEM.net Board(s) and/or the standard reference modem.
4. Press and release the RESET switch located to the right of the LCD.

The LCD acknowledges that the soft modem is initialized, as shown in Figure 5-18.

FIGURE 5-18: LCD DISPLAY



5.5.3 Configure the Terminal for dsPIC30F Soft Modem

These procedures assume, but do not require, that you are using the Windows Hyper Terminal.

1. Start Hyper Terminal on both PCs. If you are set up for interoperability and are using a standard reference modem on one terminal, consult the modem user manual for specifics on configuring the modem.

2. Configure Hyper Terminal with these settings:

Bits per second = 19200

Data bits = 8

Parity = None

Stop bits = 1

Flow Control = Hardware

Set the Comm Channel appropriately for each PC.

3. Execute these AT commands on each Hyper Terminal and note the response.

Command	Response
AT<ENTER>	OK
AT&F<ENTER>	OK
ATX0	OK
ATS0=1	OK

Refer to Appendix B for descriptions of AT commands supported.

5.5.4 Initiate Communication

At this point, the dsPIC30F Soft Modem is ready to establish a dial connection with another modem.

1. Initiate dialing with the following AT command:

ATD<dial string>

where *<dial string>* indicates the telephone number of the remote soft modem.

By default, 'auto answer after one ring' is enabled. The remote soft modem answers the call after one ring and connects in V.22bis and V.42 (LAPM) Mode.

ITU-T Recommendation V.42 contains a High Level Data Link Control (HDLC) protocol referred to as Link Access Procedure for Modems (LAPM) and defines error-correcting protocols for modems. The following result codes are displayed on both the Hyper Terminals upon a successful connection.

CARRIER 2400

PROTOCOL LAPM

CONNECT 19200

You should hear the call negotiation and training sequence on the call progress speaker. If you are using a standard reference modem on one terminal, the connection message transmitted and displayed on the PC may be slightly different but should present this same basic information (see Table 6-3 and Table 6-4). See **Section 6.8 "Response Messages for Successful Modem Connections"** for examples of a connection messages provided by a US Robotics 5686E modem.

2. With the two modems connected, test data transfer by typing information on one terminal and verifying that the same data is displayed on the other terminal.
3. Test file transfer by creating a brief text file and using following file transfer options on the Hyper Terminal.

Hyper Terminal>Transfer>Send Text File

Hyper Terminal>Transfer>Capture File

4. Repeat the test procedure for the other modulation protocols by executing the following AT commands:

Protocol	Command
V.22bis-2400 bps	AT+MS=2,0,300,2400
V.22bis-1200 bps	AT+MS=2,0,300,1200
V.23 bps	AT+MS=3,0,300,1200
V.21 bps	AT+MS=0,0,300,300
Bell 103-300 bps	AT+MS=1,0,300,300

5. You may need to reset and reconfigure the standard reference modem for enabling these specific data modulations. Consult the standard reference modem user's manual for this information.

An example of configuring a US Robotics modem for V.23 and V.21 is presented in **Section 6.6 "V.23 Connection For US Robotics Model 5686E Modem"** and **Section 6.7 "V.21 Connection for US Robotics Model 5686E Modem"**

5.6 EMBEDDED MODEM DEMO

The embedded soft modem provides all the features and functionality of the AT Command modem less the need for the AT command layer and its required Program Memory and Data Memory resources. The key difference between building the AT command modem and the embedded modem is the implementation of specific API files. The embedded API uses `SMUA_IL.c` and `SMUA_OL.c` files. The AT command modem uses `API.c` and `AT_cmds.c` files.

The demo can be tailored for specific modem algorithms from V.32bis down to V.21. Specifics on configuration of the modem is described in Section 2.5.1. The embedded modem demo uses the same hardware setup as the AT command modem (see Figure 5-16 and Figure 5-17). The embedded modem uses the dsPIC 12-bit ADC, INTx pins and UART peripherals.

5.6.1 Getting Started

Follow these steps to get the embedded demo up and running:

5.6.1.1 CREATE EMBEDDED DEMO PROJECT

1. Create MPLAB IDE project (see 5.4.1 “Creating the Project”).
2. Add the required files into the project (see Table 5-1).
3. Configure modem parameters contained in the `option.h` and `dpconfig.inc` files (see Section 2.5 “Setting Up Compile Time Options For Your Application”).
4. Configure the DAA/AFE hardware using the `Si3021config.inc` file (see Section 2.6 “Configuring the Analog Front End”).

5.6.1.2 SET UP EMBEDDED API

1. Configure the modems for ORIGINATE or ANSWER mode. Locate this ‘C’ statement in the `SMUA_OL.c` file:

```
SM_Config_Params.SMMode = SM_ORIGINATE;
```

To configure a modem for ANSWER mode, change the statement to read:

```
SM_Config_Params.SMMode = SM_ANSWER;
```

2. For the ORIGINATE modem, the DTMF dial string and length must be initialized.

Locate and modify these ‘C’ statements in the `SMUA_OL.c` file:

```
SM_Config_Params.DialString[0] = 8; // Digit string length
SM_Config_Params.DialString[1] = 9; // Digit 1
SM_Config_Params.DialString[2] = 7; // Digit 2
SM_Config_Params.DialString[3] = 9; // Digit 3
SM_Config_Params.DialString[4] = 2; // Digit 4
SM_Config_Params.DialString[5] = 4; // Digit 5
SM_Config_Params.DialString[6] = 0; // Digit 6
SM_Config_Params.DialString[7] = 1; // Digit 7
SM_Config_Params.DialString[8] = 2; // Digit 8
```

Note: These statements are not used in Answer mode and do not need to be modified for the ANSWER modem.

3. Initialize the bit rate and error control settings in the `SMUA_OL.c` file.

Locate the following 'C' define statements:

```
#define SM_TXLEVEL          12
#define SM_SPEAKER_VOLUME  2
#define SM_LOOBACK         0
#define SM_ORIGINATE       1
#define SM_ANSWER          2
#define SM_MODULATION_MODE  ENABLE_AUTO
#define SM_BITRATE         2400
#define SM_V42ENABLE       1
```

If necessary, change the values for the `SM_BITRATE` and `SM_V42ENABLE` defines. For example, if V.42 error control is not required set value for `SM_V42ENABLE` to '0'.

4. Select the correct dsPIC device speed in the `Device_Fcy.inc` file.

```
/* Set ONLY ONE of the following equates to a TRUE state */
.equ  XTx4PLL,    0    ; do not select
.equ  XTx8PLL,    0
.equ  XTx16PLL,   1
```

For V.22bis modulations and lower, set `.equ XTx8PLL to '1'`.

For V.32bis modulations, set `.equ XTx16PLL to '1'`.

5.6.1.3 BUILD THE PROJECT AND PROGRAM THE dsPIC30F DEVICE

1. Build the project.
2. Download the project (program the chip).

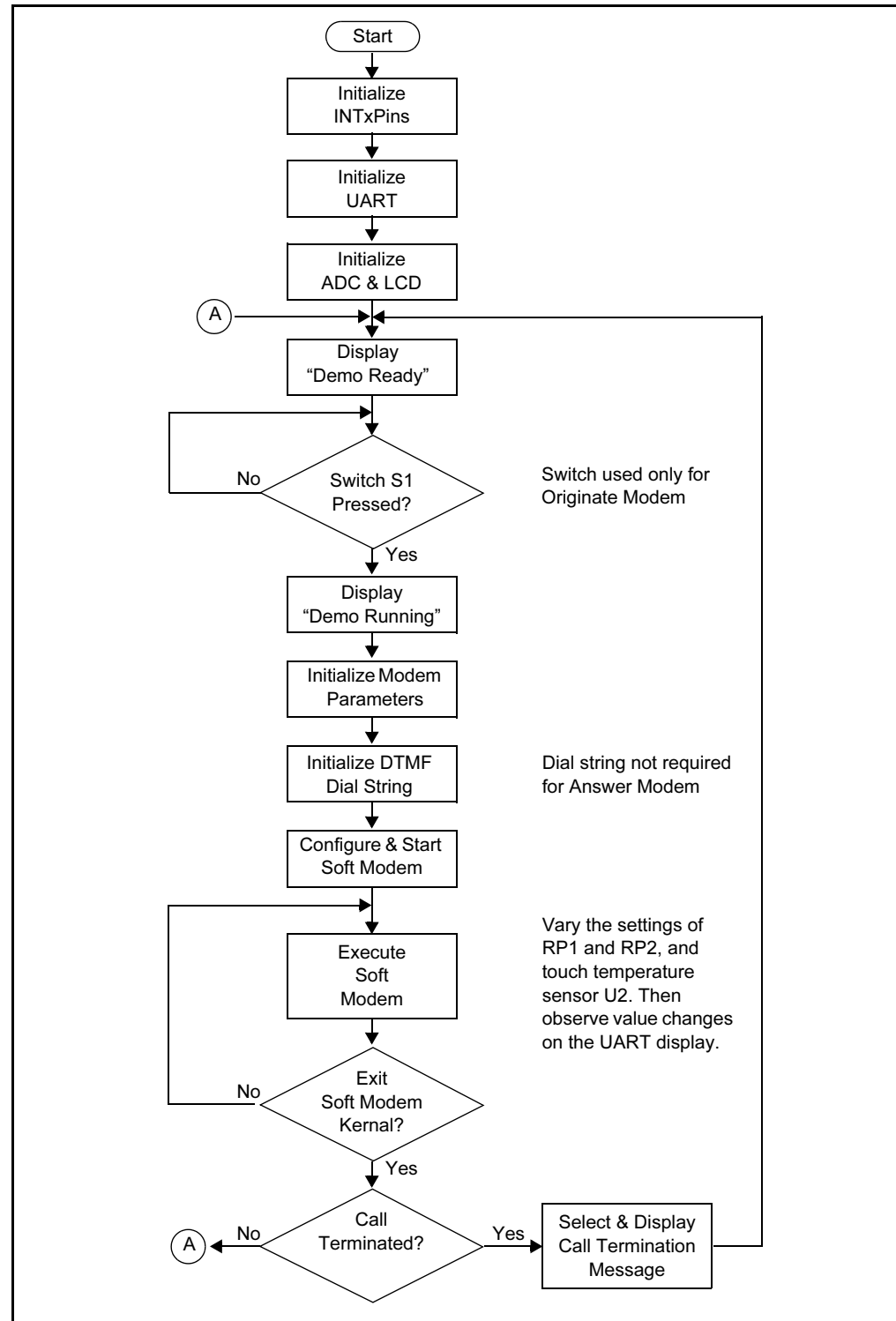
5.6.2 Initialize the Hardware

1. Configure the hardware for end-to-end operation or interoperability.
2. Make sure the 2X2 shunts on J17 and J19 are inserted in vertical alignment (perpendicular to the LCD display).
3. Apply power to the dsPICDEM.net board and/or standard reference modem.
4. Set up the Hyper Terminal connection to the UART for 19200, 8 bits, no parity, 1 stop bit and hardware flow control. This connection allows the PC to be used for monitoring received data and creating text messages to send.
5. Press and release the RESET switch located to the right of the LCD.

5.6.3 Run the Demo

Figure 5-19 is a flow diagram of the embedded modem demo. If the modem is configured for Originate mode, the program will initiate a dial sequence and then enter Full Duplex Communication mode when the connection is made. If the modem is configured for answer mode, it waits for a call and then enters full duplex communication when the connection is made.

FIGURE 5-19: EMBEDDED MODEM DEMO FLOW

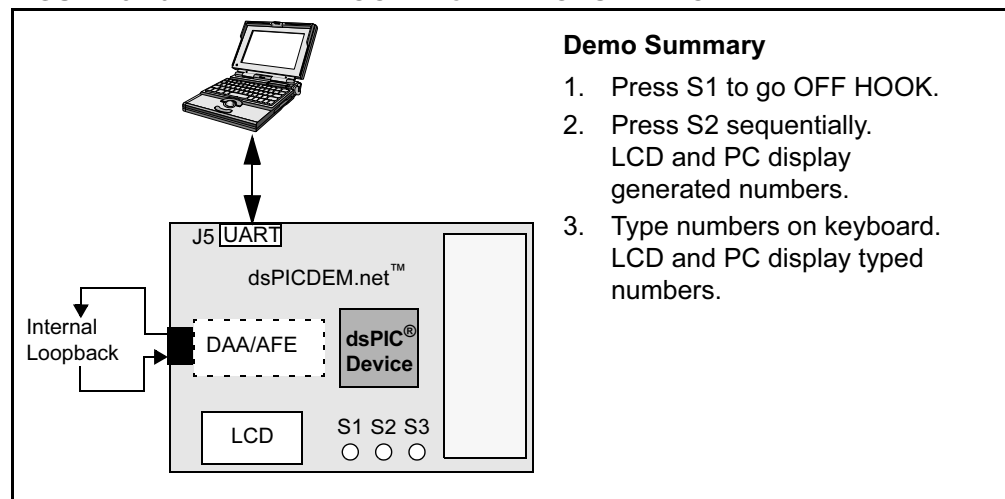


5.7 DTMF LOOPBACK DEMO

The DTMF loopback demonstration uses the dsPIC30F6014 device to both generate and detect the DTMF signal. Two generation methods are used. First, repetitive operation of switch S2 on the dsPICDEM.net development board cycles a program buffer through the numbers 0-9 and the symbols # and *. Second, a PC or laptop connected to the dsPIC30F6014 UART peripheral can be used to externally generate the same characters from the keyboard. Regardless of the source, the DTMF algorithm generates the corresponding DTMF digit and passes it to the DAA/AFE interface on the dsPICDEM.net development board.

The DTMF loopback demonstration is illustrated in Figure 5-20.

FIGURE 5-20: DTMF LOOPBACK DEMONSTRATION



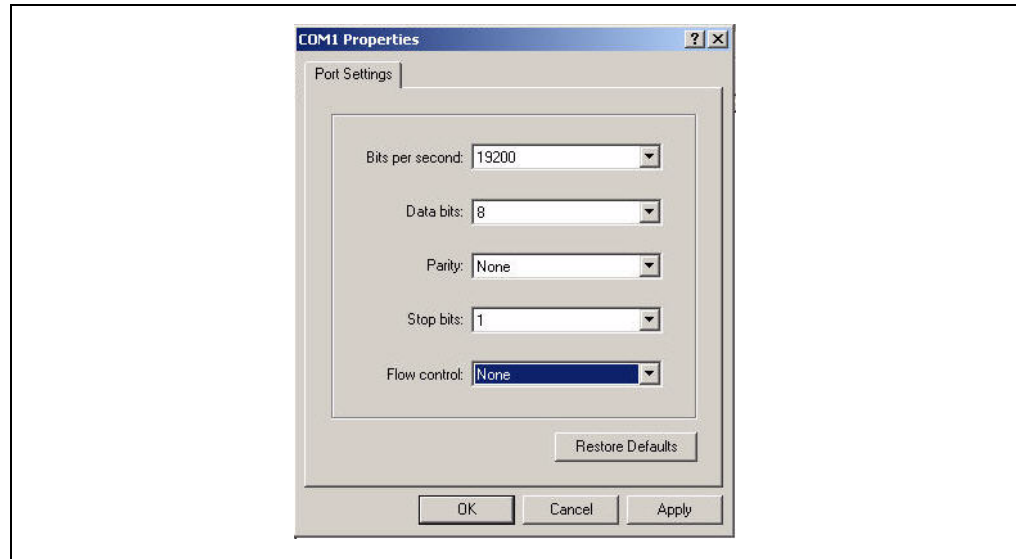
For this demonstration, the DAA/AFE internally loops the signal back to the dsPIC device. The DTMF detection algorithm results in a corresponding character that is displayed on (1) the LCD on the dsPICDEM.net development board and (2) the screen on the PC (via the UART).

5.7.1 Run DTMF Loopback Demo

Follow these steps to run the DTMF loopback demo:

1. Connect the dsPICDEM.net board to your laptop or PC (run an RS-232 cable from J5 on the board to the serial port on the PC).
2. Launch Hyper Terminal on the PC and select the port settings shown in Figure 5-21 .

FIGURE 5-21: HYPER TERMINAL SETUP



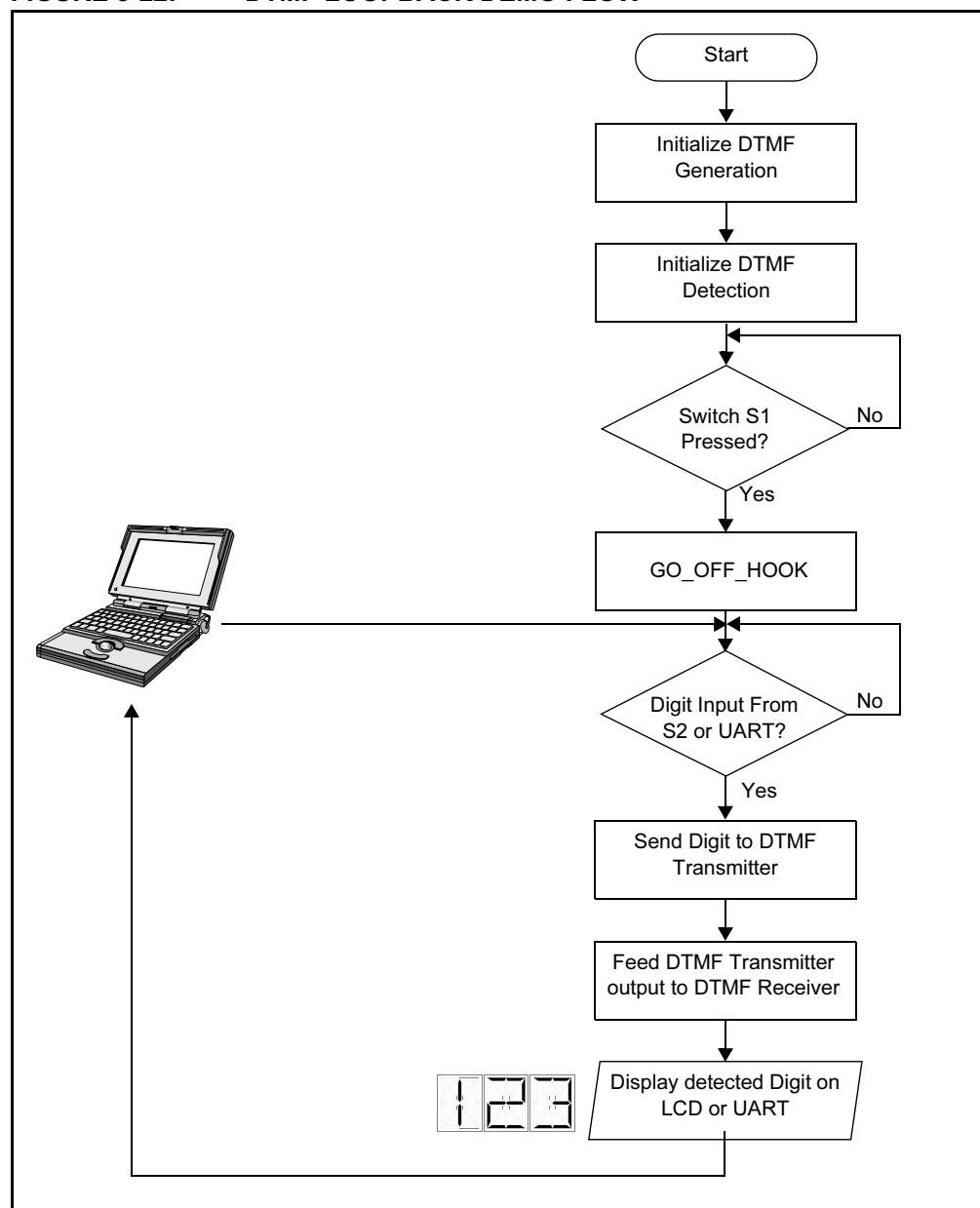
3. From MPLAB IDE, run the project (*Debugger>Run*).

Note: When MPLAB IDE is connected to the dsPICDEM.net board as a debugger, the program must be started, halted and reset from the MPLAB IDE workspace. When MPLAB IDE is connected to the dsPICDEM.net board as a programmer, you disconnect it from the board before running the program from the dsPICDEM.net board.

4. Press S1. The DTMF goes OFF HOOK.
5. Press S2. A digit displays on the LCD. The same digit also displays on Hyper Terminal.
6. Type a digit on Hyper Terminal. That digit displays on both the LCD and Hyper Terminal.
7. Halt the program (*Debugger>Halt*).

Figure 5-22 summarizes the DTMF loopback demo in a flow diagram.

FIGURE 5-22: DTMF LOOPBACK DEMO FLOW



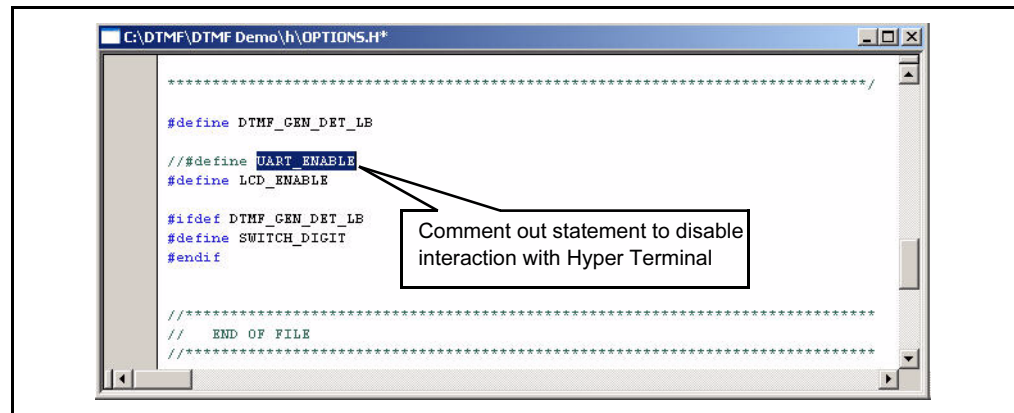
5.7.2 Modify DTMF Loopback Program

The DTMF loopback program, while simplistic in scope, demonstrates important functionality of the dsPIC30F DTMF Generation and Detection Modules. Though the demo focuses on DTMF generation and detection, it also illustrates important interactions with the UART, the DAA/AFE, the LCD and interrupts represented by switch closures.

The other objective of the demo is to give you a jump start toward incorporating this functionality into your user application. Accordingly, the following procedures illustrate techniques for changing the functionality of the DTMF loopback demo program.

1. In MPLAB IDE, open the `options.h` file and disable `#define UART_ENABLE` (see Figure 5-23) .

FIGURE 5-23: EXAMPLE DTMF LOOPBACK DEMO MODIFICATION



2. Rebuild the program (*Debugger>Build All*).

Note: It is always necessary to reprogram the part with the new code after each build. MPLAB will remind you with a message that states “Program memory has changed since last operation.”

3. Reprogram the chip (*Debugger>Program*).
4. Reset the program (*Debugger>Reset*).
5. Rerun the program (*Debugger>Run*).

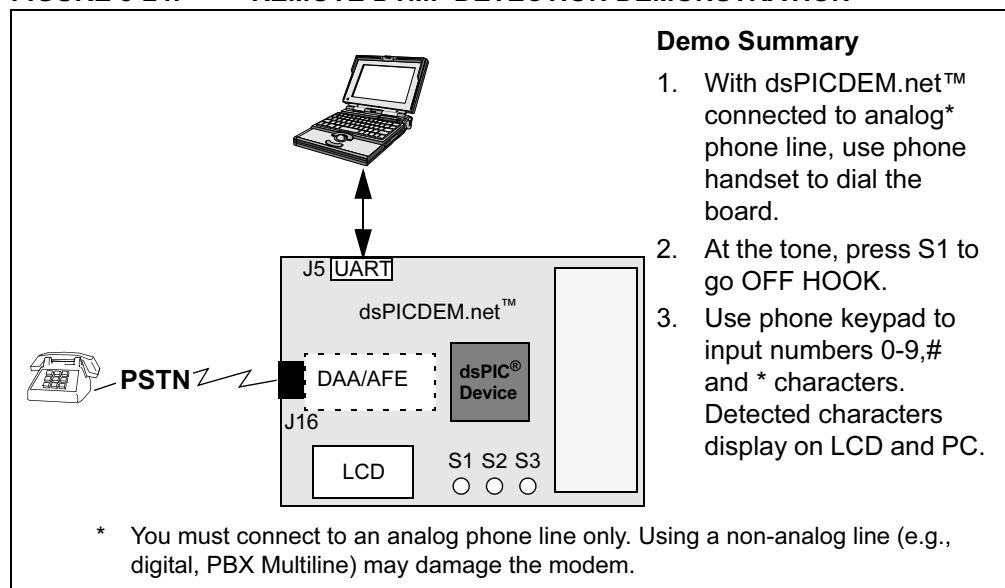
Observe that the program no longer supports the Hyper Terminal.

5.8 DTMF PSTN PHONE LINE DEMO

In this demonstration the DTMF detection module interacts with a remote telephone over the PSTN, as illustrated in Figure 5-24. The telephone handset is used to dial up the dsPICDEM.net board over an analog phone line. The Si3034/Si3035 DAA/AFE interface (on the board) detects the ring and alerts the dsPIC30F6014, which then executes the DTMF detection algorithm. The OFF HOOK function is performed by pressing S1. At that point, the remote telephone is communicating with the DTMF detection module via the DAA/AFE modem circuitry.

Digits can now be generated from the telephone keypad and recognized by the DTMF detection algorithm. The recognized digits are sent to both the LCD and the UART for display. The UART passes the digits to the attached PC running Hyper Terminal.

FIGURE 5-24: REMOTE DTMF DETECTION DEMONSTRATION

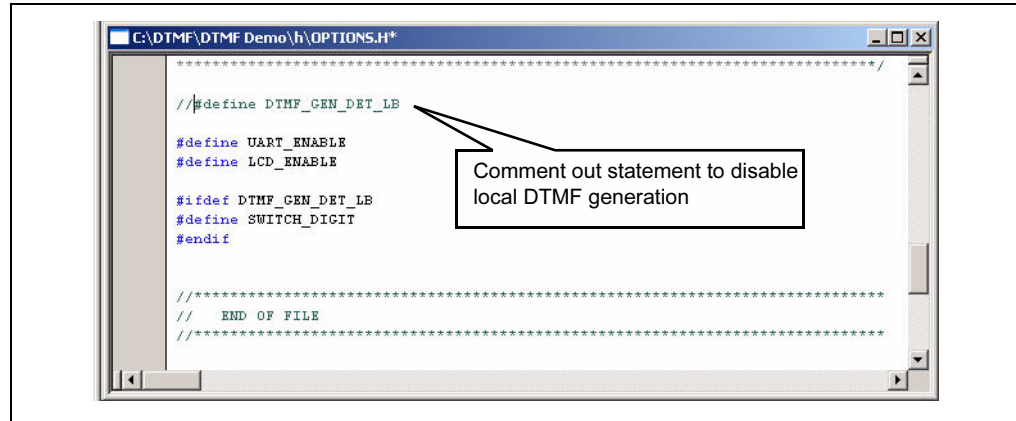


5.8.1 Run DTMF Remote Detection Demonstration

In the local DTMF generation and detection demonstration, the program was assembled and built from ready-to-use source files. The remoted DTMF detection demonstration requires modification of the source files before the demonstration program will work properly.

1. Connect the dsPICDEM.net board (J16) to an analog phone line.
2. In MPLAB IDE, open the `options.h` file and disable `#define DTMF_GEN_DET_LB` (see Figure 5-25).
Be sure that `#define UART_ENABLE` is re-enabled.

FIGURE 5-25: DTMF REMOTE DETECTION DEMO MODIFICATION



3. Rebuild the program (*Debugger>Build All*).

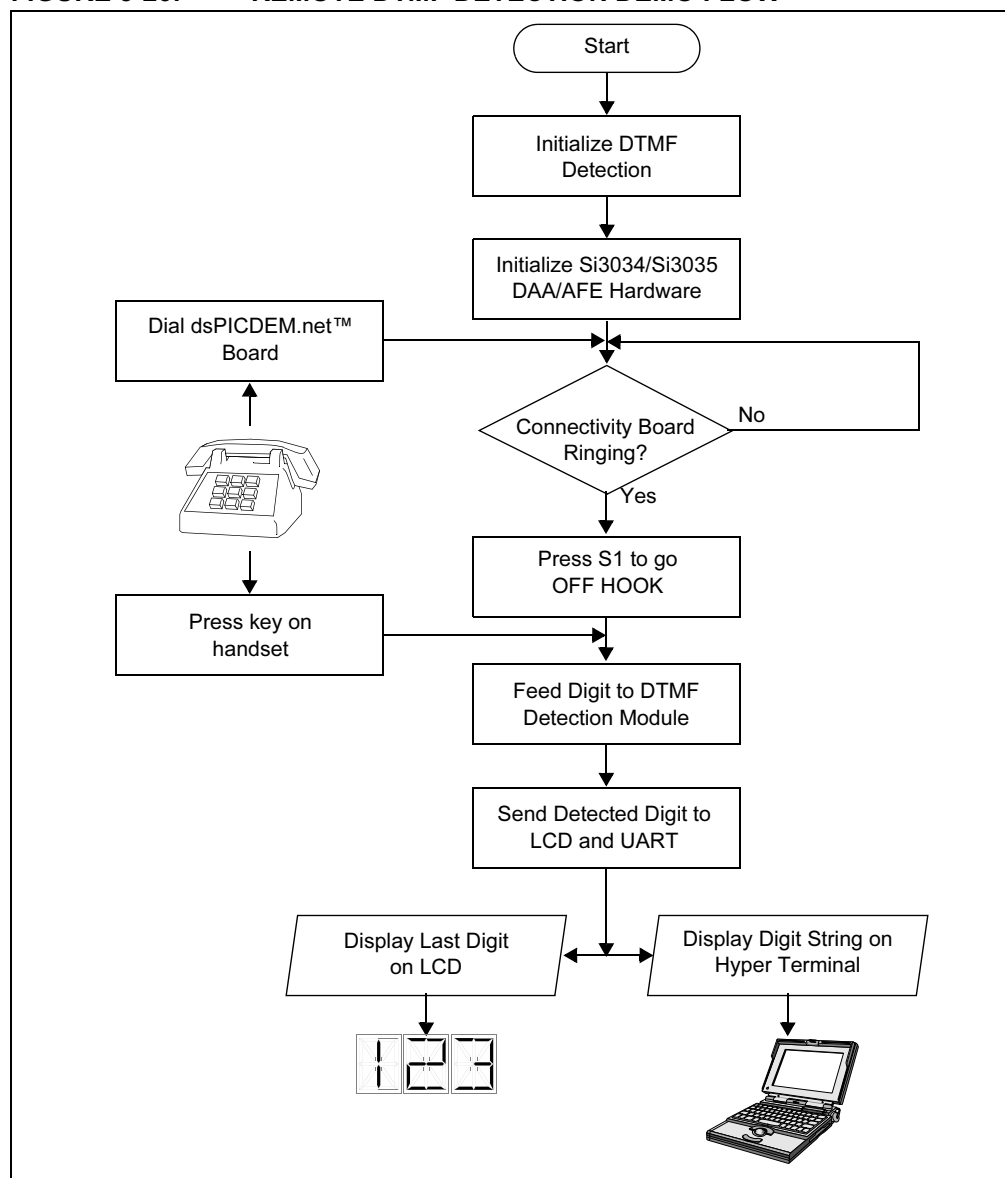
Note: It is always necessary to reprogram the part with the new code after each build. MPLAB will remind you with a message that states “Program memory has changed since last operation.”

4. Reprogram the chip (*Debugger>Program*).
5. Reset the program (*Debugger>Reset*).
6. Run the program (*Debugger>Run*).
7. From the telephone, dial the phone number to which the dsPICDEM.net board is attached. When you hear the ring on the board, press S1 to go OFF HOOK.
8. On the telephone keypad, enter 0-9, * or #. Observe that the last digit displays on the LCD and the series of digits displays on the Hyper Terminal.
9. To end, hang up the phone and press RESET on the board.

dsPIC30F Soft Modem Library User's Guide

Figure 5-26 summarizes the remote DTMF detection demonstration in a flow diagram.

FIGURE 5-26: REMOTE DTMF DETECTION DEMO FLOW



Chapter 6. Troubleshooting

6.1 INTRODUCTION

The dsPIC30F Soft Modem has undergone multiple levels of testing, ranging from bench testing using standard off-the-shelf reference modems to full ITU-T compliance testing using reference modems and PSTN test equipment. The reference modems used in the testing are listed in Table 6-1.

It is important to note that not all reference modems are equal in terms of data pump modulations they support as power-on default. For example, the US Robotics model 5686E external 56K modem does support V.21 and V.23, but only if it is enabled via suitable AT commands. Attempts to connect the dsPIC Soft Modem to this specific modem, for support of V.23 and V.21, will fail until these DP modulations have been enabled in the USR.

Also the US Robotics Model 5686E external 56K modem does not support V.32 9600 bps (NTCM) so the modem will disconnect after the rate exchange sequence during the V.32 hand shake, since this is an invalid speed setting. This section will not address every specification or operational feature for standard reference modems but is intended to be a starting point should you experience some basic connection issues.

Review the information in this chapter in the event your dsPIC30F based soft modem system experiences connection or communication issues.

6.2 HIGHLIGHTS

This chapter covers:

- Standard Reference Modems
- ITU-T V.8 Support
- Data Flow Control
- V.23 Connection For US Robotics Model 5686E Modem
- V.21 Connection for US Robotics Model 5686E Modem
- Response Messages for Successful Modem Connections
- Response Message for Unsuccessful Modem Connection
- Regulatory Compliance Reference Information

dsPIC30F Soft Modem Library User's Guide

6.3 STANDARD REFERENCE MODEMS

During the performance and interoperability testing of the dsPIC30F Soft Modem, the following reference modems have been used in both the originate and answer modes while communicating with the dsPIC Soft Modem running on the dsPICDEM.net™ Connectivity Board. The USR modems, models 5686E and 0839, and Zoom modems are readily available as of this document release.

TABLE 6-1: REFERENCE MODEMS

Reference Modem		Description
Manufacturer	Model	
U.S. Robotics	5686E	External V.90/V.92 56k Fax/Modem
U.S. Robotics	0839	External Sportster 33.6 Fax/Modem
U.S. Robotics	98117203	External Sportster Voice 33.6 Fax/Modem
Zoom	2949	External V.90 Fax/Modem
Zoom	3049	External V.92 Fax/Modem
D-Link	DMF-336/E	External 33.6 Fax/Modem
D-Link	DMF-560/ES	External 56k Data/Fax/Voice Modem
3Com	455630-01	External 56k Fax/Modem

6.3.1 6.2 Library Data Pump Selections

The dsPIC30F Soft Modem Library includes several data pump modulation selections. Each library satisfies a specific modulation specification, as listed in Table 6-2:

TABLE 6-2: SOFT MODEM LIBRARIES

Soft Modem Library	Data Pump Modulations Supported	V.42 Supported?
Without ITU-T V.42 Error Control⁽¹⁾		
dsPICSM_V32bis.a	V.32bis, V.32, V.22bis, V.23 and V.21/Bell 103	No
dsPICSM_V32.a	V.32, V.22bis, V.23 and V.21/Bell 103	No
dsPICSM_V22bis.a	V.22bis, V.23 and V.21/Bell 103	No
dsPICSM_V23.a	V.23 and V.21/Bell 103	No
With ITU-T V.42 Error Control^(1,2)		
dsPICSM_V32bisE.a	V.32bis, V.32, V.22bis, V.23 and V.21/Bell 103	Yes
dsPICSM_V32E.a	V.32, V.22bis, V.23 and V.21/Bell 103	Yes
dsPICSM_V22bisE.a	V.22bis, V.23 and V.21/Bell 103	Yes
Standalone Data Pump Modulations with V.42 Error Control		
dsPICSM_V32bisSE.a	V.32bis	Yes
dsPICSM_V32SE.a	V.32	Yes
dsPICSM_V22bisSE.a	V.22bis	Yes

Note 1: Each Data Pump modulation includes all lower, fallback modulations.

2: V.23 and V.21/Bell 103 do not make V.42 error control connections.

6.4 ITU-T V.8 SUPPORT

Currently dsPIC30F Soft Modem Library only supports falling back to data pump modulations by going through the V.8 handshake. Consequently, the soft modem is not able to establish a fall-back connection in any Data Pump Modulation mode with reference modems that do not support the V.8 handshake.

For example, reference modems such as D-Link and Zoom do not support the V.23 modulation selection through V.8 handshake when they are in the originate mode. Therefore the dsPIC30F soft modem is not able to establish a V.23 connection with these reference modems in answer mode. However in answer mode, these reference modems do support the falling back to V.23 modulations through V.8 handshake. Therefore the dsPIC30F soft modem in Originate mode is able to establish a V.23 connection with these reference modems.

6.5 DATA FLOW CONTROL

The dsPIC30F Soft Modem, by default, uses hardware flow control via the UART for data transfer. UART flow control is provided on pins (RTS & CTS) on the dsPICDEM.net™ connectivity board. A two-position shunt must be installed in the vertical position on jumper J17 and J19. It is also necessary to enable the hardware flow control in Hyper Terminal.

It is also possible to change the Flow Control mode of dsPIC30F Soft Modem to 'Xon/Xoff flow control' by using a specific AT command (AT&Kn) and the corresponding Flow Control mode in Hyper Terminal.

When the Soft Modem Embedded API is implemented the UART is not a base requirement and hence the flow control information described is not applicable.

6.6 V.23 CONNECTION FOR US ROBOTICS MODEL 5686E MODEM

The following is an example for enabling the dsPIC30F Soft Modem for V.23 use with the US Robotics Model 5686E Reference Modem. Recognized AT commands are displayed as 'OK' on the Hyper Terminal. AT commands that are not recognized or supported are displayed as 'ERROR' on the Hyper Terminal.

AT Commands for V.23 Connection on USRobotics Modem:

Command	Function
AT	
AT&F1	Factory Default settings
ATS0=1	Enable auto answer after one ring
ATS27=16	Enable the V.23 fallback

AT Commands for V.23 Connection on dsPIC30F Soft Modem:

Command	Function
AT	
AT&F	dsPIC30F Soft Modem default settings
AT+MS=3,0,300,1200	Force V.23 connection

dsPIC30F Soft Modem Library User's Guide

6.7 V.21 CONNECTION FOR US ROBOTICS MODEL 5686E MODEM

The following is an example for enabling the dsPIC30F Soft Modem for V.21 use with the US Robotics Model 5686E Reference Modem. Recognized AT commands are displayed as 'OK' on the Hyper Terminal. AT commands that are not recognized or supported are displayed as 'ERROR' on the Hyper Terminal.

AT Commands for V.21 Connection on USRobotics Modem:

Command	Function
AT	
AT&F1	Factory Default settings
ATS0=1	Enable auto answer after one ring
ATS27=1	Enable the V.21 fallback

AT Commands for V.21 Connection on dsPIC30F Soft Modem:

Command	Function
AT	
AT&F	dsPIC Soft Modem default settings
AT+MS=0,0,300,300	Force V.21 connection

In most modems, a V.42 LAPM connection is not supported for V.21 and V.23 data pump modulation modes. Consequently, a connection is established in non-V.42 mode (PROTOCOL NONE).

6.8 RESPONSE MESSAGES FOR SUCCESSFUL MODEM CONNECTIONS

Table 6-3 lists connection messages that are displayed on the Hyper Terminal for successful modem connections between the dsPIC30F Soft Modem and the US Robotics Model 5686E modem.

TABLE 6-3: RESPONSE MESSAGES WITH US ROBOTICS MODEM

DP Modulation	dsPIC Soft Modem	US Robotics, Model 5686E
V.32bis 14400 bps	CARRIER 14400 PROTOCOL LAPM CONNECT 19200	CONNECT 14400/ARQ/V32B/LAPM
V.32bis 12000 bps	CARRIER 12000 PROTOCOL LAPM CONNECT 19200	CONNECT 12000/ARQ/V32B/LAPM
V.32bis 9600 bps	CARRIER 9600 PROTOCOL LAPM CONNECT 19200	CONNECT 9600/ARQ/V32B/LAPM
V.32bis 7200 bps	CARRIER 7200 PROTOCOL LAPM CONNECT 19200	CONNECT 7200/ARQ/V32B/LAPM
V.32 9600 bps (NTCM)	n/a	not supported
V.32 4800 bps (NTCM)	CARRIER 4800 PROTOCOL LAPM CONNECT 19200	CONNECT 4800/ARQ/V32/LAPM
V.22bis 2400 bps	CARRIER 2400 PROTOCOL LAPM CONNECT 19200	CONNECT 2400/ARQ/LAPM
V.22 1200 bps	CARRIER 1200 PROTOCOL LAPM CONNECT 19200	CONNECT 1200/ARQ/LAPM
V.23 1200 bps	CARRIER 1200/75 PROTOCOL NONE CONNECT 19200 (dsPIC30F Soft Modem – answer modem)	CONNECT 75/1200/NONE (USRobotics – originate modem)
	CARRIER 75/1200 PROTOCOL NONE CONNECT 19200 (dsPIC30F Soft Modem – originate modem)	CONNECT 1200/75/NONE (USRobotics – answer modem)
V.21 300 bps	CARRIER 300 PROTOCOL NONE CONNECT 19200	CONNECT
Bell 103	CARRIER 300 PROTOCOL NONE CONNECT 19200	CONNECT

The following are connection messages displayed on hyper-terminal for successful modem connections between the dsPIC Soft Modem and the D-Link modem.

dsPIC30F Soft Modem Library User's Guide

TABLE 6-4: RESPONSE MESSAGES WITH D-LINK MODEM

DP Modulation	dsPIC Soft Modem	D-Link DFM-56ES Modem
V.32bis 14400 bps*	CARRIER 14400 PROTOCOL LAPM CONNECT 19200	+MCR: V32B +MRR: 14400, 14400 +ER: LAPM CONNECT 19200
V.32bis 12000 bps*	CARRIER 12000 PROTOCOL LAPM CONNECT 19200	+MCR: V32B +MRR: 12000, 12000 +ER: LAPM CONNECT 19200
V.32bis 9600 bps*	CARRIER 9600 PROTOCOL LAPM CONNECT 19200	+MCR: V32B +MRR: 9600, 9600 +ER: LAPM CONNECT 19200
V.32bis 7200 bps*	CARRIER 7200 PROTOCOL LAPM CONNECT 19200	+MCR: V32B +MRR: 7200, 7200 +ER: LAPM CONNECT 19200
V.32 9600 bps (NTCM)	CARRIER 9600 PROTOCOL LAPM CONNECT 19200	+MCR: V32 +MRR: 9600,9600 +ER: LAPM CONNECT 19200
V.32 4800 bps (NTCM)	CARRIER 4800 PROTOCOL LAPM CONNECT 19200	+MCR: V32 +MRR: 4800,4800 +ER: LAPM CONNECT 19200
V.22bis 2400 bps	CARRIER 2400 PROTOCOL LAPM CONNECT 19200	+MCR: V22B +MRR: 2400,2400 +ER: LAPM CONNECT 19200
V.22 1200 bps	CARRIER 1200 PROTOCOL LAPM CONNECT 19200	+MCR: V22B +MRR: 1200,1200 +ER: LAPM CONNECT 19200
V.23 1200 bps	n/a (dsPIC30F Soft Modem – answer modem)	not supported (D-Link – originate modem)
	CARRIER 75/1200 PROTOCOL NONE CONNECT 19200 (dsPIC30F Soft Modem – originate modem)	+MCR: V23C +MRR: 1200,75 +ER: NONE CONNECT 19200 (D-Link – answer modem)
V.21 300 bps	CARRIER 300 PROTOCOL NONE CONNECT 19200	+MCR: V21 +MRR: 300,300 +ER: NONE CONNECT 19200
Bell 103	CARRIER 300 PROTOCOL NONE CONNECT 19200	+MCR: B103 +MRR: 300,300 +ER: NONE CONNECT 19200

6.9 RESPONSE MESSAGE FOR UNSUCCESSFUL MODEM CONNECTION

Table 6-5 is an example of connection messages displayed on the Hyper Terminal for an unsuccessful V.32 4800bps LAPM connection with the US Robotics Model 5686E modem.

TABLE 6-5: RESPONSE MESSAGES FOR UNSUCCESSFUL CONNECTION USING US ROBOTICS MODEM

Originator	Response Message
dsPIC30F Soft Modem	CARRIER 4800 PROTOCOL NONE CONNECT 19200
USRobotics	CONNECT 4800/ V32/NONE

Table 6-6 is an example of connection messages displayed on the Hyper Terminal for an unsuccessful V.32bis 14400 bps LAPM connection with the D-Link Model DFM-56ES modem.

TABLE 6-6: RESPONSE MESSAGES FOR UNSUCCESSFUL CONNECTION USING D-LINK MODEM

Originator	Response Message
dsPIC30F Soft Modem	CARRIER 14400 PROTOCOL NONE CONNECT 19200
D-Link	+MCR: V32B +MRR: 14400, 14400 +ER: NONE CONNECT 19200

6.10 REGULATORY COMPLIANCE REFERENCE INFORMATION

Every country has telecommunication regulations that prohibit the connection of unapproved telecommunication devices, including modems, to the phone line. Approval by a country's telecommunications regulatory agency may entail hardware and/or firmware modifications to bring your end-system modem into compliance in such areas as radio-frequency interference, pulse dial make/break ratios, redial capabilities and so forth.

The words "approved or compliant for use in country XYZ" mean that the modem has been modified to comply with the telecommunication regulations of that country. However, an American FCC approved modem imported to Germany would not automatically be a legal telecommunications device in Germany. FCC approval in the USA, for example, does not imply BZT approval in Germany.

The Silicon Laboratories chipsets used with the dsPICDEM.net 1 and dsPICDEM.net 2 Connectivity Boards are compliant with Federal Communications Commission (FCC), Japan Approval Institute for Telecommunications Equipment (JATE) and Common Technical Regulation (CTR_21) country-specific Post, Telephone & Telegraph (PTT) specifications. The chipsets can be fully programmed for AC termination, DC termination, ringer impedance and ringer threshold, enabling the devices to meet worldwide telephone line interface requirements. The devices interface directly to the dsPIC30F Data Converter Interface (DCI) Peripheral.

6.10.1 Federal Communications Commission

The Federal Communications Commission (FCC) rules (47 C.F.R. Part 68) governs the direct connection of Terminal Equipment (TE) to the Public Switched Telephone Network (PSTN), and to wireline carrier-owned facilities used to provide private line services.

6.10.2 Industry Canada DOC CS-03

Approval of equipment for use in Canada is relatively straight-forward. Testing and compliance should be in accordance with technical standard CS-03, which are well harmonized with FCC part-68. In many instances it is possible to use FCC reports to obtain approval. For additional information on FCC part-68 and approved test facilities, refer to the following link: www.fcc.gov

6.10.3 CTR-21

CTR-21 is a Common Technical Regulation that defines a harmonized standard for analog access to the PSTN throughout the European Economic Area (EEA) and Switzerland. In the past, analog standards were not harmonized. Manufacturers were required to go to each country and test analog equipment to that country's unique specifications. CTR-21 simplifies this process by enabling manufacturers to go to one test lab and take one compliance test for all EEA member countries. This enables faster product delivery throughout the EEA market.

6.10.4 JATE

JATE is the institute that approves telecommunications equipment for use with Japan's public telephone network. Approved equipment bears a JATE approval mark or number.

For more information on JATE check this web site: www.jate.or.jp/index_e.html

Additional information is available at the Nippon Telegraph and Telephone Corporation web site: www.ntt.co.jp/index_e.html.

Appendix A. ITU-T Specifications

A.1 ITU-T SPECIFICATIONS

Table A-1 lists the ITU-T specifications implemented for the development and testing of the dsPIC Soft Modem and DTMF Generation/Detection modules.

TABLE A-1: ITU-T SPECIFICATIONS

ITU-T Specification	Name of the Document
ITU-T V.32bis	A duplex modem operating at data signalling rates of up to 14400 bit/s for use on the general switched telephone network and on leased point-to-point 2-wire telephone-type circuits.
ITU-T V.32	A family of 2-wire, duplex modems operating at data signalling rates of up to 9600 bit/s for use on the general switched telephone network and on leased telephone-type circuits.
ITU-T V.22bis	2400 bits per second duplex modem using the frequency division technique standardized for use on the general switched telephone network and on point-to-point 2-wire leased telephone-type circuits.
ITU-T V.21	300 bits per second duplex modem standardized for use in the general switched telephone network.
ITU-T V.23	600/1200-baud modem standardized for use in the general switched telephone network.
ITU-T V.8	Procedures for starting sessions of data transmission over the public switched telephone network.
ITU-T V.25	Automatic answering equipment and general procedures for automatic calling equipment on the general switched telephone network including procedures for disabling of echo control devices for both manually and automatically established calls.
ITU-T V.42	Error-correcting procedures for DCEs using asynchronous-to-synchronous conversion.
ITU-T V.56bis	Network transmission model for evaluating modem performance over 2-wire voice grade connections.
ITU-T V.56ter	Test procedure for evaluation of 2-wire 4 kHz voice band duplex modems.
ITU-T Q.23	Technical features of push-button telephone sets.
ITU-T Q.24	Multi-frequency push-button signal reception.

dsPIC30F Soft Modem Library User's Guide

NOTES:

Appendix B. AT Command Set

B.1 INTRODUCTION

This section defines the AT command set implemented on the dsPIC30F Soft Modem.

B.2 AT COMMANDS

The AT commands listed in the table are a subset of the overall AT command set. The AT commands listed have been selected to serve the dsPIC Soft Modem in a range of applications.

TABLE B-1: AT COMMANDS

Command	Description
En	COMMAND ECHO E0 – Inhibits the echoing of commands to the computer E1 – Echoes commands to the computer
Hn	HANG Up and HOOK CONTROL H – Modem hangs up and go on-hook. H0 – Causes the modem to go on-hook H1 – Causes the modem to go off-hook
Ln	Control Speaker Volume L0 – Speaker Turn off L1 – Low L2 – Medium (Default) L3 – High
In	INFORMATION and IDENTIFICATION This command has various options that are used to instruct the modem to provide specific information about itself I0 – Displays the modem controller firmware revision I3 – Same as I0 I4 – Current modem settings I6 – Link diagnostics
Qn	QUIET MODE Determines whether the modem sends result codes and status codes (OK, BUSY, RING, etc) to the terminal. Q0 – Display result codes, user sees command responses (e.g., OK) Q1 – Result codes are suppressed, user does not see responses
Vn	RESULT CODE FORM Selects whether the modem sends long form (in words) or short form (numeric) result codes the terminal. V0 – Numeric result codes V1 – English result codes (e.g., CONNECT, BUSY etc)
A	ANSWER Modem goes off hook, transmits answer tone and waits for a carrier response from the remote modem.

dsPIC30F Soft Modem Library User's Guide

TABLE B-1: AT COMMANDS (CONTINUED)

Command	Description
Dnnn . .	DIAL Dials a telephone number and attempts to connect. The dial command must be the last command on the line. To cancel the Dial command, press any key. To dial 4807927200, type ATD4807927873
T	LOCAL LOOP BACK Modem Starts Local loop back
On	ENTER CONNECT STATE AND RETRAIN O0 – Leave On-line Command mode and return to On-line Data mode. O1 – Issues the retrain command
Wn	NEGOTIATION PROGRESS MESSAGE CONTROL W0 – CONNECT result code reports DTE speed; disable the display of all extended result codes. W1 – CONNECT result code reports DTE speed, enable the display of CARRIER and PROTOCOL extended codes only W2 – CONNECT result code reports DCE speed; disable the display of all the extended result codes.
Xn	DIAL TONE AND BUSY TONE DETECTION X0 – Busy detect and dial tone detect disabled X1 – Busy detect and dial tone detect disabled. X2 – Busy detect disabled and dial tone detect enabled. X3 – Busy detect enabled and dial tone detect disabled. X4 – Busy detect and dial tone detect enabled.
&F	MODEM RESET Restore settings to defaults (Modem reset)
&Kn	MODEM FLOW CONTROL This command determines how the flow control between the computer and the local modem is handled. &K0 – Disable local flow control. &K3 – Enable RTS/CTS (HARDWARE) flow control (default). &K4 – Enable XON/XOFF (SOFTWARE) flow control.
&Qn	COMMUNICATION MODE Q5 – Modem negotiates an Error-corrected link (Default) Q6 – Selects asynchronous operation

TABLE B-1: AT COMMANDS (CONTINUED)

Command	Description																																							
+MS_1, 2, 3, 4,	<p>SELECT MODULATION +MS=<mod> ,<automod>,<minrate>,<maxrate> Where, <mod> – Defines the specified modulation <automod> – Enables Auto mode operation. This mode is always enabled. <minrate> – Sets the lowest modem speed <maxrate> – Sets the highest modem speed</p> <table><tr><th><Mod></th><th>Modulation</th><th>Possible Rates(bps)</th></tr><tr><td>0</td><td>V.21/Bell</td><td>103300</td></tr><tr><td>2</td><td>V.22bis</td><td>2400 or 1200</td></tr><tr><td>3</td><td>V.23</td><td>1200</td></tr><tr><td>9</td><td>V.32</td><td>9600 or 4800</td></tr></table> <p>By default V.32bis-14400 mode is selected and V.22bis/V.23/V.21/Bell 103 modes can be selected by the following commands respectively:</p> <table><tr><th>Modulation Mode</th><th>Command</th></tr><tr><td>V.32bis-14400 bps</td><td>AT+MS=8,0,300,14400</td></tr><tr><td>V.32bis-12000 bps</td><td>AT+MS=8,0,300,12000</td></tr><tr><td>V.32bis-9600 bps</td><td>AT+MS=8,0,300,9600</td></tr><tr><td>V.32bis-7200 bps</td><td>AT+MS=8,0,300,7200</td></tr><tr><td>V.32-9600 bps</td><td>AT+MS=9,0,300,9600</td></tr><tr><td>V.32-4800 bps</td><td>AT+MS=9,0,300,4800</td></tr><tr><td>V.22bis -2400 bps</td><td>AT+MS=2,0,300,2400</td></tr><tr><td>V.22bis-1200 bps</td><td>AT+MS=2,0,300,1200</td></tr><tr><td>V.23 bps</td><td>AT+MS=3,0,300,1200</td></tr><tr><td>V.21 bps</td><td>AT+MS=0,0,300,300</td></tr><tr><td>Bell 103 -300 bps</td><td>AT+MS=1,0,300,300</td></tr></table> <p>Note 1: For V.23, originating mode transmits at 75bps and receives at 1200 bps. Answering mode transmits at 1200bps and receives at 75bps.</p>	<Mod>	Modulation	Possible Rates(bps)	0	V.21/Bell	103300	2	V.22bis	2400 or 1200	3	V.23	1200	9	V.32	9600 or 4800	Modulation Mode	Command	V.32bis-14400 bps	AT+MS=8,0,300,14400	V.32bis-12000 bps	AT+MS=8,0,300,12000	V.32bis-9600 bps	AT+MS=8,0,300,9600	V.32bis-7200 bps	AT+MS=8,0,300,7200	V.32-9600 bps	AT+MS=9,0,300,9600	V.32-4800 bps	AT+MS=9,0,300,4800	V.22bis -2400 bps	AT+MS=2,0,300,2400	V.22bis-1200 bps	AT+MS=2,0,300,1200	V.23 bps	AT+MS=3,0,300,1200	V.21 bps	AT+MS=0,0,300,300	Bell 103 -300 bps	AT+MS=1,0,300,300
<Mod>	Modulation	Possible Rates(bps)																																						
0	V.21/Bell	103300																																						
2	V.22bis	2400 or 1200																																						
3	V.23	1200																																						
9	V.32	9600 or 4800																																						
Modulation Mode	Command																																							
V.32bis-14400 bps	AT+MS=8,0,300,14400																																							
V.32bis-12000 bps	AT+MS=8,0,300,12000																																							
V.32bis-9600 bps	AT+MS=8,0,300,9600																																							
V.32bis-7200 bps	AT+MS=8,0,300,7200																																							
V.32-9600 bps	AT+MS=9,0,300,9600																																							
V.32-4800 bps	AT+MS=9,0,300,4800																																							
V.22bis -2400 bps	AT+MS=2,0,300,2400																																							
V.22bis-1200 bps	AT+MS=2,0,300,1200																																							
V.23 bps	AT+MS=3,0,300,1200																																							
V.21 bps	AT+MS=0,0,300,300																																							
Bell 103 -300 bps	AT+MS=1,0,300,300																																							
<p>S-REGISTERS: These registers are used set some of the simple modem configurations.</p> <p>ATS_n=x This command writes the value x to the specified S-register n. ATS_n? This command displays the value of S-register (n)</p>																																								
S0	<p>RING TO AUTO-ANSWER ON Sets the number of rings required before the modem answers. Auto answer ring count Default Value: S0=1</p>																																							
S1	<p>RING COUNTER Counts and stores the number of rings from an incoming call.</p>																																							
S6	<p>WAIT BEFORE DIALING Sets the number of seconds the modem waits for dial tone before dialing Default Value: S6=1</p>																																							
S7	<p>WAIT FOR CARRIER AFTER DIAL/ANSWER Sets the number of seconds the modem waits for a carrier or answers before returning on-hook and sending a NO CARRIER result code. Default Value: S7=60</p>																																							
S91	<p>TRANSMIT SIGNAL LEVEL This register is used to specify the transmit signal level in –dBm. Default Value: S91=12 (-12 dBm)</p>																																							
+++	<p>ESCAPE SEQUENCE Used to escape to Command mode from the data mode.</p>																																							

dsPIC30F Soft Modem Library User's Guide

NOTES:

Appendix C. Drivers

C.1 INTRODUCTION

This section provides information related to drivers used with the dsPIC30F Soft Modem.

C.2 DAA/AFE DRIVER FUNCTIONS

The dsPIC30F Soft Modem Library implements both 8.0 kHz and 7.2 kHz sampling rates when communicating with the DAA/AFE. The following driver functions are provided to operate the Si3021 with different options specified in the preceeding tables.

Init_Si3021

Function: Used to initialize different parameters of Si3034/Si3035 AFE.
Sets the sampling rate to 8 kHz and also sets different international control registers of this chip set to the selected options.

Prototype: `void Init_Si3021(void)`

Arguments: None

Return Value: None

Example: `Init_Si3021();`

SetFs7200

Function: Used to change the AFE sampling rate to 7.2 kHz.
Initially Si3021 is initialized for 8 KHz sampling rate in `Init_Si3021`.
This function must be called by the application when data pump requests the application to change the sampling rate to 7.2 kHz.

Prototype: `void SetFs7200(void)`

Arguments: None

Return Value: None.

Example: `SetFs7200();`

SetFs8000

Function: Used to change the AFE sampling rate to 8 kHz.
This function must be called by the application to change the AFE sampling rate from 7.2 kHz to 8 kHz.

Prototype: `void SetFs8000(void)`

Arguments: None

Return Value: None.

Example: `SetFs8000();`

go_on_hook

Function: Used to set the AFE to On-Hook state.

Prototype: `void go_on_hook(void)`

Arguments: None

Return Value: None

Example: `go_on_hook();`

go_off_hook

Function: Used to set the AFE to Off-Hook state.

Prototype: `void go_off_hook(INT SpeakerVolume)`

Arguments: Speaker Volume control value for Si3021 control register 6 (DAA Control 2)

Return Value: None.

Example: `SpeakerVolume = 0x03 -> Low
SpeakerVolume = 0x63 -> Medium
SpeakerVolume = 0x00 -> High
go_off_hook(SpeakerVolume);`

speaker_off

Function: Used to mute the transmit and receive path signals for call progress AOUT pin of this chip set. This function is used to turn off the transmit and receive signal to the speaker connected to AOUT pin of the chip set

Prototype: `void speaker_off(void)`

Arguments: None

Return Value: None

Example: `speaker_off();`

Index

A

A/D Converter	
Convert	32, 33, 49, 50, 51, 52, 53, 54, 55, 56, 57, 60, 64, 65, 67, 68, 73, 74, 76, 77, 78, 121, 122
AT Command Layer	
Overview	8
AT Command Layer Options.....	18
AT Command Set	117

B

Batch Files	16
-------------------	----

C

CMX Scheduler RTOS	18
Compile Options	17
ConvertADC	32, 33, 49, 50, 51, 52, 53, 54, 55, 56, 57, 60, 64, 65, 67, 68, 73, 74, 76, 77, 78, 121, 122
Country Settings	22
CTR-21 Compliance.....	21, 114
Customer Notification Service.....	5
Customer Support.....	6

D

DAA/AFE	
chip sets.....	19
Configuration Options	21
Connections to dsPIC30F.....	19
Driver Functions.....	121
Gain Control.....	20
Data Flow Control	109
Data Pump	
Overview	7
Data Pump Modulation Options	17
Documentation	
Numbering Conventions	2
Updates	2
DTMF	
Build Table	24
Generation and Detection Modules	24
Resource Usage	25

F

FCC Compliance.....	21, 114
Free Software Foundation	4
FSK Modems	9
Full Duplex Modems	9

G

GNU Language Tools	4
--------------------------	---

I

Industry Canada DOC CS-03 Compliance.....	114
Internet Address.....	4
ITU-T Specifications.....	115
ITU-T V.8 Support.....	109

J

JATE Compliance	21, 114
-----------------------	---------

L

Language Toolsuite.....	83
Library Contents.....	9

M

Microchip Web Site	4
Miscellaneous Options	18
Modem Characteristics	9
Modem Libraries	16
Archive Files	12
Modulation Options	13
Source Files for.....	14
MPLAB	2
MPLAB IDE User's Guide	4

P

Performance Characteristics.....	10
Project	81
Project Wizard	81
PTT Compliance	21

Q

QAM Modems	9
------------------	---

R

Recommended Reading	3
Resource Usage	
DTMF	25
Soft Modem.....	25

S

Soft Modem Resource Usage	25
Standard Reference Modems	108

T

TCM Modems.....	9
Troubleshooting	
D-Link Modem.....	112
US Robotics Modem	109

V

V.42 Error Control	
Overview	7

dsPIC30F Soft Modem Library User's Guide

V.42 Error Control Layer

 Overview 8

V.42 Options 18

V.8 Handshake..... 109

W

Warranty Registration..... 2

WWW Address..... 4

NOTES:



WORLDWIDE SALES AND SERVICE

AMERICAS

Corporate Office

2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support: 480-792-7627
Web Address: www.microchip.com

Atlanta

3780 Mansell Road, Suite 130
Alpharetta, GA 30022
Tel: 770-640-0034
Fax: 770-640-0307

Boston

2 Lan Drive, Suite 120
Westford, MA 01886
Tel: 978-692-3848
Fax: 978-692-3821

Chicago

333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 630-285-0071
Fax: 630-285-0075

Dallas

4570 Westgrove Drive, Suite 160
Addison, TX 75001
Tel: 972-818-7423
Fax: 972-818-2924

Detroit

Tri-Atria Office Building
32255 Northwestern Highway, Suite 190
Farmington Hills, MI 48334
Tel: 248-538-2250
Fax: 248-538-2260

Kokomo

2767 S. Albright Road
Kokomo, IN 46902
Tel: 765-864-8360
Fax: 765-864-8387

Los Angeles

18201 Von Karman, Suite 1090
Irvine, CA 92612
Tel: 949-263-1888
Fax: 949-263-1338

San Jose

1300 Terra Bella Avenue
Mountain View, CA 94043
Tel: 650-215-1444
Fax: 650-961-0286

Toronto

6285 Northam Drive, Suite 108
Mississauga, Ontario L4V 1X5, Canada
Tel: 905-673-0699
Fax: 905-673-6059

ASIA/PACIFIC

Australia

Suite 22, 41 Rawson Street
Epping 2121, NSW
Australia
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing

Unit 706B
Wan Tai Bei Hai Bldg.
No. 6 Chaoyangmen Bei Str.
Beijing, 100027, China
Tel: 86-10-85282100
Fax: 86-10-85282104

China - Chengdu

Rm. 2401-2402, 24th Floor,
Ming Xing Financial Tower
No. 88 TIDU Street
Chengdu 610016, China
Tel: 86-28-86766200
Fax: 86-28-86766599

China - Fuzhou

Unit 28F, World Trade Plaza
No. 71 Wusi Road
Fuzhou 350001, China
Tel: 86-591-7503506
Fax: 86-591-7503521

China - Hong Kong SAR

Unit 901-6, Tower 2, Metroplaza
223 Hing Fong Road
Kwai Fong, N.T., Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431

China - Shanghai

Room 701, Bldg. B
Far East International Plaza
No. 317 Xian Xia Road
Shanghai, 200051
Tel: 86-21-6275-5700
Fax: 86-21-6275-5060

China - Shenzhen

Rm. 1812, 18/F, Building A, United Plaza
No. 5022 Binhe Road, Futian District
Shenzhen 518033, China
Tel: 86-755-82901380
Fax: 86-755-8295-1393

China - Shunde

Room 401, Hongjian Building, No. 2
Fengxiangnan Road, Ronggui Town, Shunde
District, Foshan City, Guangdong 528303, China
Tel: 86-757-28395507 Fax: 86-757-28395571

China - Qingdao

Rm. B505A, Fullhope Plaza,
No. 12 Hong Kong Central Rd.
Qingdao 266071, China
Tel: 86-532-5027355 Fax: 86-532-5027205

India

Divyasree Chambers
1 Floor, Wing A (A3/A4)
No. 11, O'Shaughnessy Road
Bangalore, 560 025, India
Tel: 91-80-22290061 Fax: 91-80-22290062

Japan

Benex S-1 6F
3-18-20, Shinyokohama
Kohoku-Ku, Yokohama-shi
Kanagawa, 222-0033, Japan
Tel: 81-45-471-6166 Fax: 81-45-471-6122

Korea

168-1, Youngbo Bldg. 3 Floor
Samsung-Dong, Kangnam-Ku
Seoul, Korea 135-882
Tel: 82-2-554-7200 Fax: 82-2-558-5932 or
82-2-558-5934

Singapore

200 Middle Road
#07-02 Prime Centre
Singapore, 188980
Tel: 65-6334-8870 Fax: 65-6334-8850

Taiwan

Kaohsiung Branch
30F - 1 No. 8
Min Chuan 2nd Road
Kaohsiung 806, Taiwan
Tel: 886-7-536-4818
Fax: 886-7-536-4803

Taiwan

Taiwan Branch
11F-3, No. 207
Tung Hua North Road
Taipei, 105, Taiwan
Tel: 886-2-2717-7175 Fax: 886-2-2545-0139

EUROPE

Austria

Durisolstrasse 2
A-4600 Wels
Austria
Tel: 43-7242-2244-399
Fax: 43-7242-2244-393

Denmark

Regus Business Centre
Lautrup høj 1-3
Ballerup DK-2750 Denmark
Tel: 45-4420-9895 Fax: 45-4420-9910

France

Parc d'Activite du Moulin de Massy
43 Rue du Saule Trapu
Batiment A - 1er Etage
91300 Massy, France
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany

Steinheilstrasse 10
D-85737 Ismaning, Germany
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Italy

Via Quasimodo, 12
20025 Legnano (MI)
Milan, Italy
Tel: 39-0331-742611
Fax: 39-0331-466781

Netherlands

Waagenburghtplein 4
NL-5152 JR, Drunen, Netherlands
Tel: 31-416-690399
Fax: 31-416-690340

United Kingdom

505 Eskdale Road
Winnersh Triangle
Wokingham
Berkshire, England RG41 5TU
Tel: 44-118-921-5869
Fax: 44-118-921-5820

05/28/04