A Low-Cost Soft Modem using the Freescale Digital Signal Controller

Designer Reference Manual

56800E
16-bit Digital Signal Controllers

DRM073
Rev. 0
07/2005

freescale.com

freescale™
semiconductor

A Low-Cost Soft Modem using the Freescale Digital Signal Controller

Designer Reference Manual

To provide the most up-to-date information, the revision of our documents on the World Wide Web will be the most current. Your printed copy may be an earlier revision. To verify that you have the latest information available, refer to http://www.freescale.com

The following revision history table summarizes changes contained in this document. For your convenience, the page number designators have been linked to the appropriate location.

Revision History

Date	Revision Level	Description	Page Number(s)
07/2005	0	Initial release	N/A

TABLE OF CONTENTS

Chapter 4
Conclusion

Appendix A
Schematics

Appendix B
Low-Cost Soft Modem Bill of Materials

Appendix C
Layout and Governmental Certifications

LIST OF FIGURES

About This Document

This manual describes the use of a 56F8300 device in a low-cost soft modem application.

Note:*The 56F8357EVM was used as the test vehicle, but is no longer available. The 56F8367EVM will yield similar results.*

Audience

This manual supports the Low-Cost Soft Modem Reference Design and customers incorporating a soft modem into their products into their products using the Freescale 56F8300 series.

Organization

This User's Manual consists of the following sections:

- **Chapter 1, Introduction,** offers a brief introduction to the soft modem application.
- **Chapter 2, System Specification and Design,** describes the system specification and implementation of the soft modem application.
- **Chapter 3, Processor Expert Bean Utilization in the Soft Modem,** illustrates the properties, methods and events for beans used in the application.
- **Chapter 4, Conclusion,** details the performance of the soft modem application.
- **Appendix A, Schematics** contains the schematics of the low-cost soft modem application
- **Appendix B, Low-Cost Soft Modem Bill of Materials** describes layout and considerations in bringing a soft modem to market.
- **Appendix C, Layout and Governmental Certifications** discusses layout and necessary approvals for products using modem technology.

Conventions

This document uses the following notational conventions:

Typeface, Symbol or Term	Meaning	Examples
`Courier Monospaced Type`	Code examples	`//Process command for line flash`
Italic	Directory names, project names, calls, functions, statements, procedures, routines, arguments, file names, applications, variables, directives, code snippets in text	...and contains these core directories: *applications* contains applications software... ...CodeWarrior project, *3des.mcp* is... ...the *pConfig* argument.... ...defined in the C header file, *aec.h*....
Bold	Reference sources, paths, emphasis	...refer to the **Targeting DSP56F83xx Platform** manual.... ...see: **C:\Program Files\Freescale\help\tutorials**
Blue Text	Linkable on-line	...refer to Chapter 7, License....
Number	Any number is considered a positive value, unless preceded by a minus symbol to signify a negative value	3V -10 DES^{-1}
ALL CAPITAL LETTERS	# defines/ defined constants	# define INCLUDE_STACK_CHECK
Brackets [...]	Function keys	...by pressing function key [F7]
Quotation marks, “...”	Returned messages	...the message, “Test Passed” is displayed.... ...if unsuccessful for any reason, it will return “NULL”...

Definitions, Acronyms, and Abbreviations

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

AGC	Automatic Gain Control
BERT	Bit Error Rate Test
LCMDC	Low-Cost Modem Daughter Card
RX	Receive
SNR	Signal-to-Noise Ratio
TX	Transmit
UUT	Unit Under Test

References

The following sources were used to produce this book; we recommend that you have a copy of these references:

- *56F8300 Peripheral User Manual*, Freescale, MC56F8300UM
- *56F8367 Evaluation Module User Manual,* Freescale, MC56F8367EVMUM
- *TAS Series II Plus Telephone Network Emulator Operations Manual*, 2700-2003, Version 2.40
- *TAS Series II Telephone Network Emulator UCO Option Operations Manual*, 2700-2734, Version 1.20
- *TAS 240 Voiceband Subscriber Loop Emulator Operations Manual*, 2700-2397, Version 1.20
- *Understanding Telephone Electronics*, SAMS, Seventh printing 1987
- *ITU-T Recommendation 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.
- *ITU-T Recommendation V.21*, 300 Bits Per Second Duplex Modem Standardized For Use On The General Switched Telephone Network.
- *ITU-T Recommendation V.25*, Automatic Answering Equipment and General Procedures for Automatic Callling Equipment on the General Switched Telephone Network, Including Procedures for Disabling of Echo Control Devices for Both Manually and Automatically Established Calls
- *Measuring the Peak-to-Average Power of Digitally Modulated Signals*, Charles J. Meyer, Senior Applications Engineer, Boonton Electronics, Application Note AN-50, 1 Apr. ‘93.
- *CH1837A/7F/8A Data Access Arrangement Module Data Sheet, V.34bis High Speed DAA Module*, Cermetek Microelectronics

Chapter 1
Introduction

This note presents the hardware and software design of a low-cost V.21/V.22/V.22bis *soft* modem. The design does not include a traditional telecommunications Pulse Code Modulation Coder/Decoder (PCM codec), but instead *uses the ADC and PWM* of the Freescale 56F8300 series to implement a less complex solution. An optional serial port with AT command set is included as a *test fixture*. Modem performance figures measured on the implementation are included for reference, as are complete implementation details.

Chapter 2
System Specification and Design

This section describes the system used to implement the soft modem, as well as the motivation for such a system. The system specification is followed by a block diagram and discussion of implementation.

2.1 The Soft Modem System Concept

A soft modem is one that can be used to modulate/demodulate data to be sent serially over an analog channel directly, without the need for a serial data path to another entity to supply data or control signals. It includes a simple way to dial phone numbers, detect ringing signals, control the hook relay of the DAA, input and output analog data, connect with remote modems and communicate with them. Complete control of the modem is embedded within the same host computer performing other system functions, such as alarm monitoring or motion control. A soft modem is the enabling ingredient that allows combining the DTE/DCE into one entity.

This differs from traditional communication systems, which are broken into two parts: a DTE, or data terminal equipment, and a DCE, or data communication equipment. Traditionally, these equipments communicated via a 25-pin serial interface comprised of such signals as are described in the V.14, V.25 and V.22bis (and many other) specifications. These signals were used for flow control and complex signaling between the DTE and DCE as to what respective states they might have been in. With the advent of the AT command set in the 1980's, many of these signals fell into disuse and the 25-pin DTE/DCE interface is now more typically found to be a nine-pin interface. The difference is made up in providing a complex set of commands and responses that attempt to keep the DTE and DCE in sync, depicted in **Figure 2-1**.

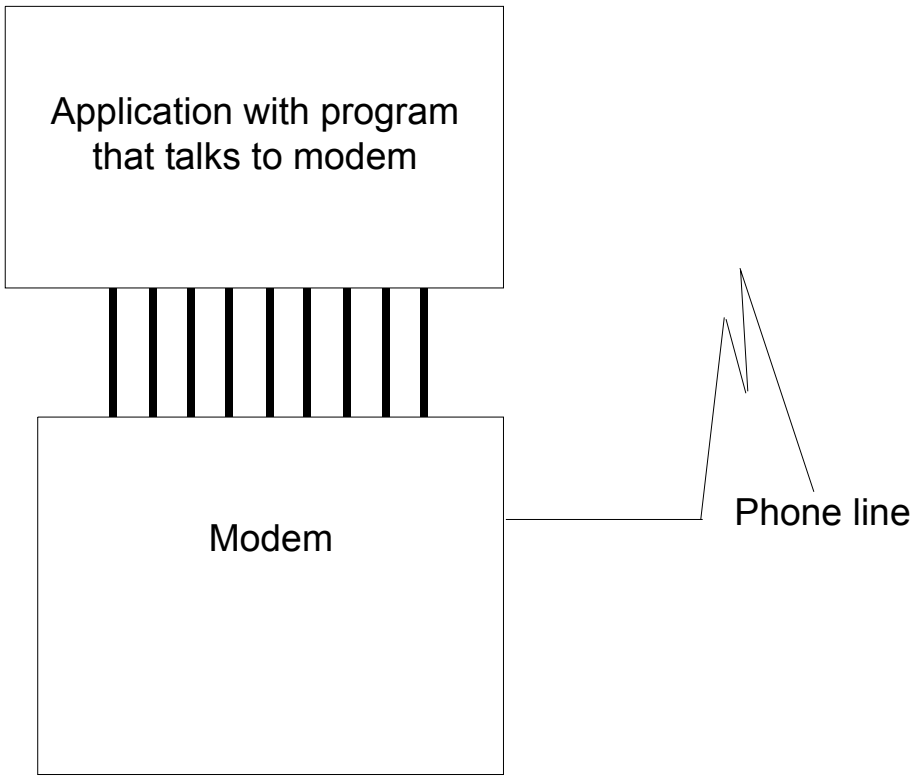


Figure 2-1. Second Millennium Modem Architecture

In the 1990's, this interface all but disappeared when a modem designed only for Windows was developed. This spoofed the serial communications to the DCE, actually performing much of the modem function on the host processor. Only digital signal processing was done on another processor closely coupled to the PC on its bus. The DTE and DCE still were two intelligent processors, but the old serial interface was gone.

When the DTE and DCE were merged into one processor around the turn of the century, the need for such a cable or even bus, vanished, as did the need for a "serial port". This is in fact how modems in PCs are implemented today. The host processor is powerful enough to implement the modem algorithms and does away with the DCE as a separate intelligent device. All that is needed is a way to send and receive analog signals over the phone line and to protect telephone equipment using the DAA. The "serial ports", used only for PC communication software compatability, are complete spoofs.

This same elegant architecture, shown in **Figure 2-2**, can now exist in embedded systems.

The project described here shows the simplicity of such a design, how few resources of the processor it takes, and how well it performs on USA average lines. This design even omits the standard telecommunications codec, instead using a Pulse Width Modulation (PWM) for output and Analog-to-Digital Converter (ADC) for input. Since both of these peripherals are readily available from the many peripherals on one 56F8300 series device, along with more processing power than required from the single core, the design is a true one-chip, one-core system that includes telecommunications ability with room for even more system functionality.

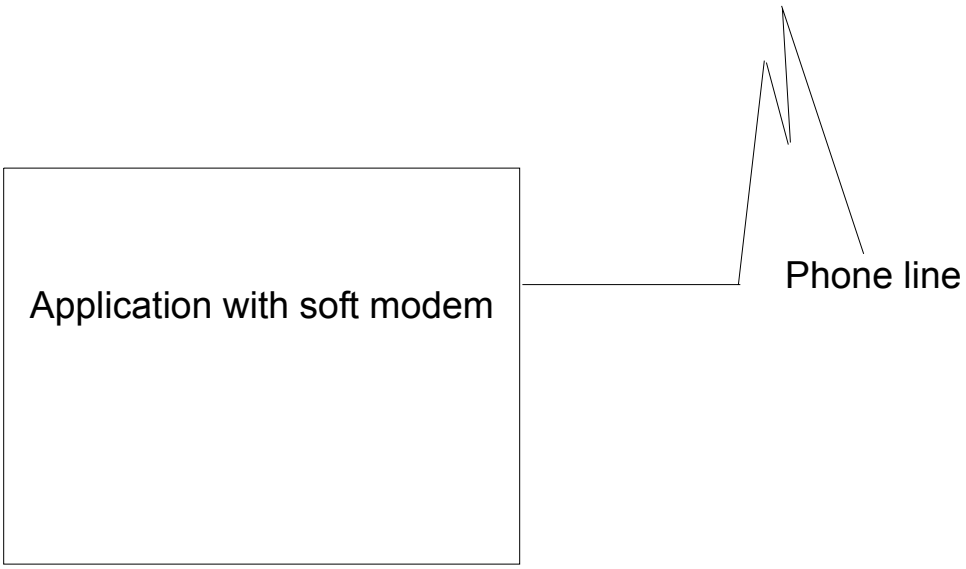


Figure 2-2. Third Millennium Modem Architecture

This approach has many applications in embedded systems. For example, a security system consisting of a single processor charged with monitoring local security sensors communicates directly over phone lines to a central reporting location. This reporting may be for the purpose of conveying the state of security.

Another application would be a portable medical device, such as a heart monitor, that could periodically interface directly to a phone line without the need for an external modem (or costly modem chip on board) of any kind. Without the DCE/DTE split, such a device would be *more reliable* and less expensive, while consuming less power.

While it would be possible to split the design into a DCE and a DTE, this would add cost, not just for the additional processor, but for the two serial ports required to connect them as well as for the software in both parts used for communication between the DTE and DCE, the interrupts and context saving and restoring that these engender. It would also add a potential failure scenario when the DCE and DTE do not observe each other's states in a timely manner.

See the right side of **Figure 2-3** for a block diagram illustrating a low-cost soft modem system utilizing Freescale's 56F8357 controller. The left side of the same figure shows the test harness integrated with the system.

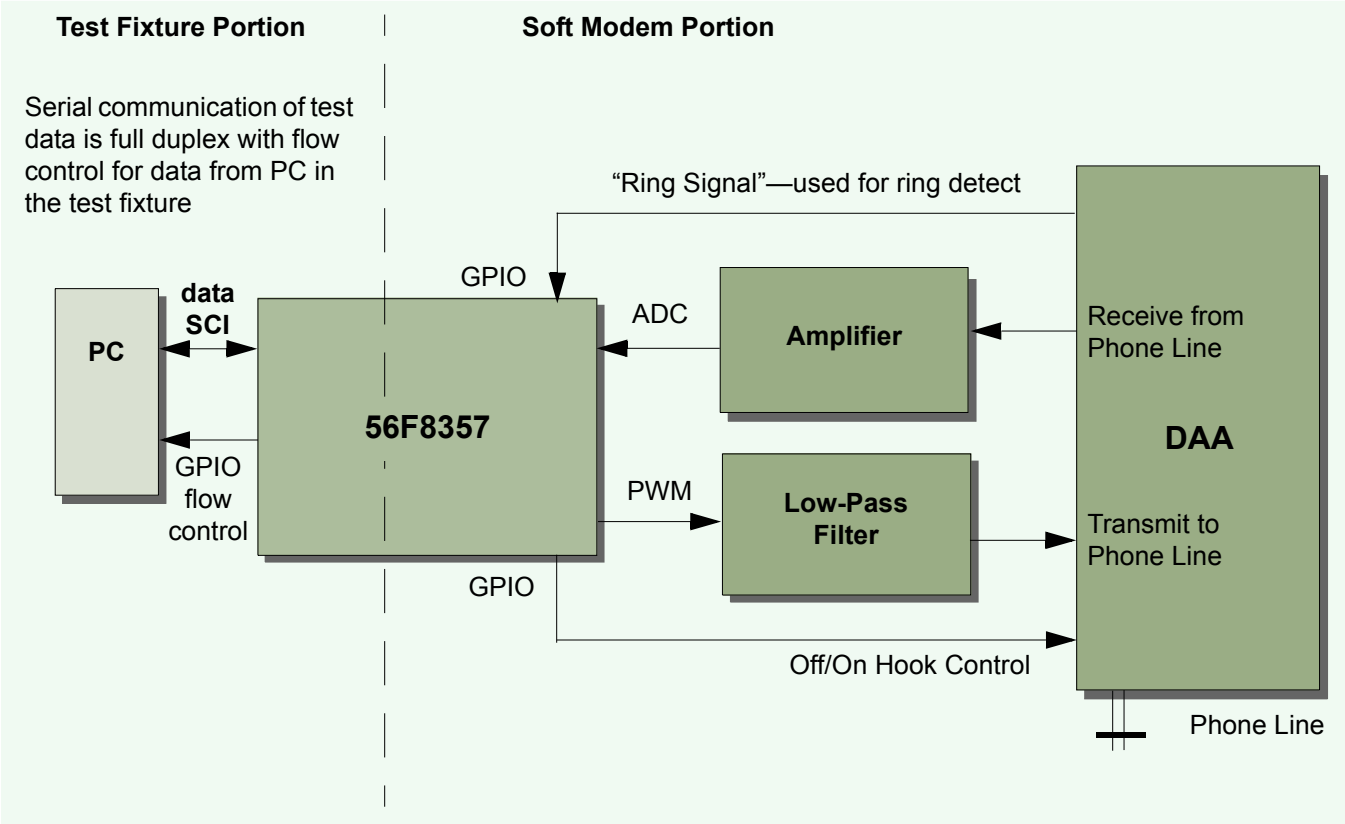


Figure 2-3. Low-Cost Soft Modem System

2.2 Soft Modem Specification and Design

Figure 2-4 shows how the basic modem, DTMF, and call progress detection are implemented. The DAA control signal, ring detect signal, flow control signal (for testing only) and the serial port (for testing only) are not shown.

The modem incorporates portions of the following standards as they apply to a soft modem: V.25, V.21, V.22, V.22bis, all implemented on the host controller. The portions of these standards related to the DTE/DCE interface are simply not required for a soft modem. PWM and ADC peripherals on the DSC pass digital samples at 7200 samples per second (SPS) for the modems and DTMF generation software. A sampling rate of 8000 SPS is used for the call progress software, used to detect dial tone. The smaller the sampling rate, the less work for the processor, and, subsequently, the less power used. The rate of 7200 SPS is ideal for these modems.

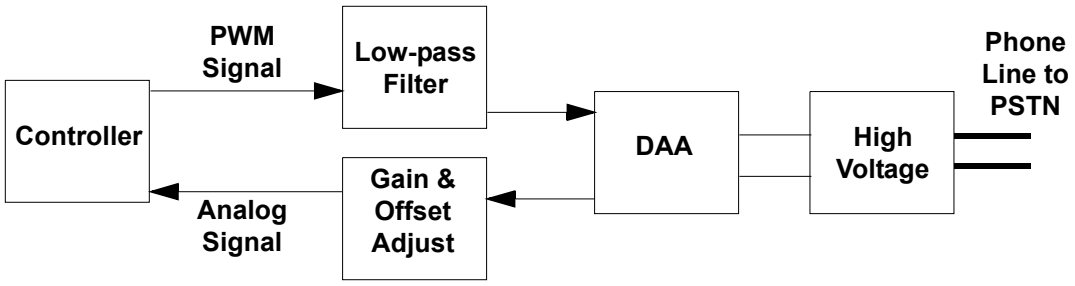


Figure 2-4. Modem Analog Block Design

2.3 Test Harness Specification and Design

The test harness for the soft modem application consists of some of the resources of the 56F8357 and some external test equipment, depicted in **Figure 2-5**. The controller and the telco connection to the controller are the only parts of this figure that are not purely test equipment.

The resources of the 56F8357 device used for testing consist of an asynchronous serial comminication port and associated beans and software. This port is used to support an AT command set which is used only for testing; it does not comprise an essential element of the soft modem design. Data and commands are alternately communicated thru this test channel. Both online and offline commands are supported. Online commands may be issued after the escape sequence puts the test fixture into the online command state. Offline commands may be issued when no connection is in progress.

Online commands supported are:

- *at**o***, which returns to online data state
- *at**h***, which hangs up the phone
- *at**z***, which hangs up the phone and performs a soft reset of the modem
- *+++ escape sequence*, with three second pre- and post-guard times

Offline commands supported are:

- *at**a*** causes the modem to go offhook and answer
  It is not necessary to use this command, since the test fixture will automatically answer two seconds after the first ring
- *at**d***<*string*> sets the number to dial string
  When this string is set to the nonempty condition, the soft modem dials that number in the string and attempts a connection. (A production program interfacing to the soft modem would simply set this string to effect a dialed connection from the soft modem.)
- *at**i*** issues the modem test fixture model and version number
- *at**q1*** puts the AT command set into quiet mode, where it operates without responding
- *at**q0*** undoes the *atq1* command
- *at**z*** performs a soft reset of the soft modem, which hangs up the phone and frees RAM resources

- *at**+0*** puts the modem into V.21 mode, where it will attempt to force all subsequent connections; a modem soft reset is also performed
- *at**+2*** puts the modem into V.22/V.22bis mode, where the modem will attempt to force all subsequent connections; a modem soft reset is also performed

The online data state of the test fixture is attained with the industry-standard escape sequence, consisting of a delay period of three seconds of no traffic on the serial test channel, three “plus” characters, and another three seconds of no activity on the serial test channel.

Connect messages are issued, indicating the line speed. The serial test channel operates at a fixed 2400 baud.

Characters are buffered into the test fixture for transmission and reception to and from the test fixture queues.

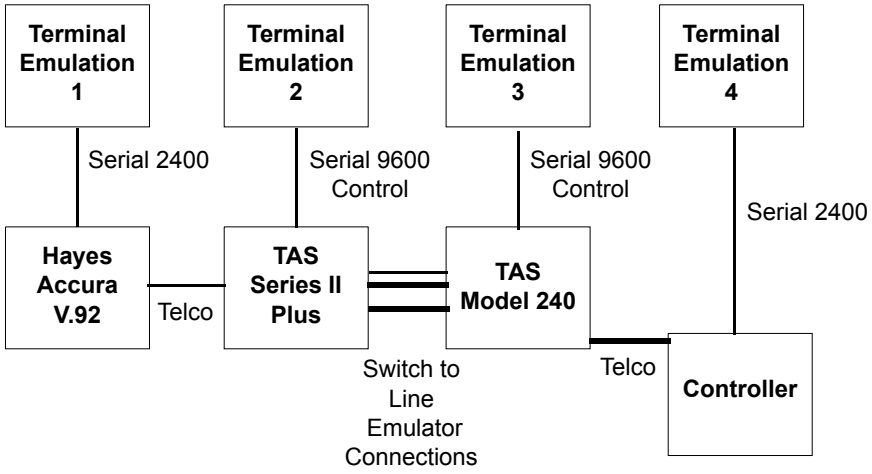


Figure 2-5. Modem Test Set-up

The external test equipment consists of two types, that connected to the RJ11 telco jack of the soft modem, and that connected to a DB9 async serial port on the EVM. The telco jack is connected to equipment that simulates the USA average Public Switched Telephone Network (PSTN) connection to an off-the-shelf modem, a Hayes Accura V.92 modem. This PSTN simulation equipment consists of the TAS Series II Plus unit and the TAS Model 240 unit.

2.3.1 Terminal Emulation One

Terminal Emulation One runs HyperTerminal on a PC at 2400 BPS, 8 bits data, no parity, 1 stop bit on a communication port. It runs a binary file transfer using one of the protocols included with HyperTerminal. It is connected to a Hayes Accura modem, where the AT command set is used to directly control the modem.

2.3.2 Terminal Emulation Two

Terminal Emulation Two is a HyperTerminal session at 9600 baud used to control the TAS Series II Plus unit. This unit is programmed to simulate USA average lines.

2.3.3 Terminal Emulation Three

Terminal Emulation Three controls the TAS Model 240, which for these tests supplies NULL line.

2.3.4 Terminal Emulation Four

Terminal Emulation Four is HyperTerminal on a PC at 2400 BPS 8 bits data, no parity, 1 stop bit, hardware flow control. It is the peer of Terminal Emulation One and runs the other end of the binary file transfer. The active signals in the DB9 include TX data, RX data, and one hardware flow control signal used to signal the PC when it may or may not send data. Flow control in the other direction was found to be unnecessary; due to the abundant resources of the PC, it never became flooded with data from the controller at 2400 BPS.

2.4 Hardware Implementation, Set-up, and Operation

A standard off-the-shelf Freescale 56F8357EVM was used for this project. The Low-Cost Freescale Modem Demonstration Kit comes with a 56F8367EVM and a Low-Cost Modem Daughter Card (LCMDC) in one box. The EVM has a connector for this daughter card. The LCMDC was developed to house the Data Access Arrangement (DAA) as well as conditioning circuits for the single PWM signal from the EVM mother board.

To assemble the modem, snap the daughter card into the connector after making the following revisions to the EVM:

- To wire ring signal for detection by the controller, connect these two signals:
  - PWMA 1 (Pin 2, J7) source is ring indicator from the daughter card
  - Quadrature Decoder 0, PHASEA0 (Pin 1, J15) destination
- To wire flow control of data from serial test device to the controller, connect these two signals and a capacitor:
  - SCI 1 TXD1 (Pin 1, J14) source
  - RTS (Pin 1, J11) destination
  - One end of a 39pF cap to RTS, the other end to ground
- Clip pins 3, 4, 5, 6 of J8 to provide clearance for the DAA high voltage section of the daughter card

The first connection connects the ring signal from the DAA to a counter, enabling ring detection functionality. The second connection connects a GPIO for flow control. To avoid noise-induced glitches on the flow control signal, a small capacitor (39pF) should connect the RTS signal to ground.

The Low-Cost Freescale Modem Demonstration Kit, available for evaluation upon reservation, contains an EVM factory modified as specified previously. The kit also includes documentation, as well as the LCMDC and required cables.

2.4.1 Pulse Width Modulation Telephony Base Band Transmission

The signal to be transmitted over the phone line is developed first as a PWM signal; its duty cycle is the analog of the signal value. This signal is then filtered to produce a real analog signal in the voice band. In the hardware it is done as in the following description.

One of the PWM channels of the 56F8367 outputs a PWM signal on pin 130, Timer D, Channel 1 (TD1). This signal is available on the daughter card via pin 70 of the connector to the EVM. This PWM is associated with the *OutputTimer* bean, shown in **Figures 3-10** through **3-12**. This signal enters from the left of **Figure 2-6**, where it is limited to 4kHz by a 4th order active low-pass filter. The filtered signal is then passed on to the left of **Figure 2-7** for level setting.

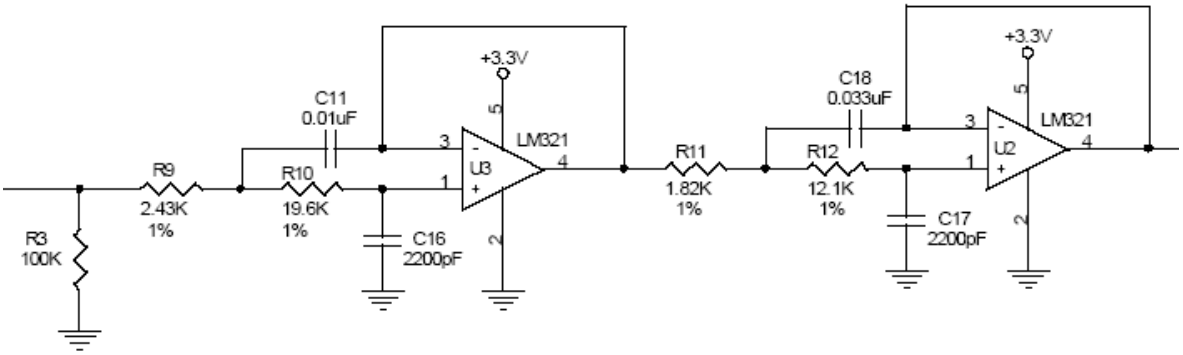


Figure 2-6. TD1 in to Active 4kHz 4th Order Filter to TP7

The ouput, XMIT, from **Figure 2-6** is then passed to the XMIT input of a Cermetek CH 1837A DAA whose output level is set to -9dBm via adjustment of R18. R17 is not populated.

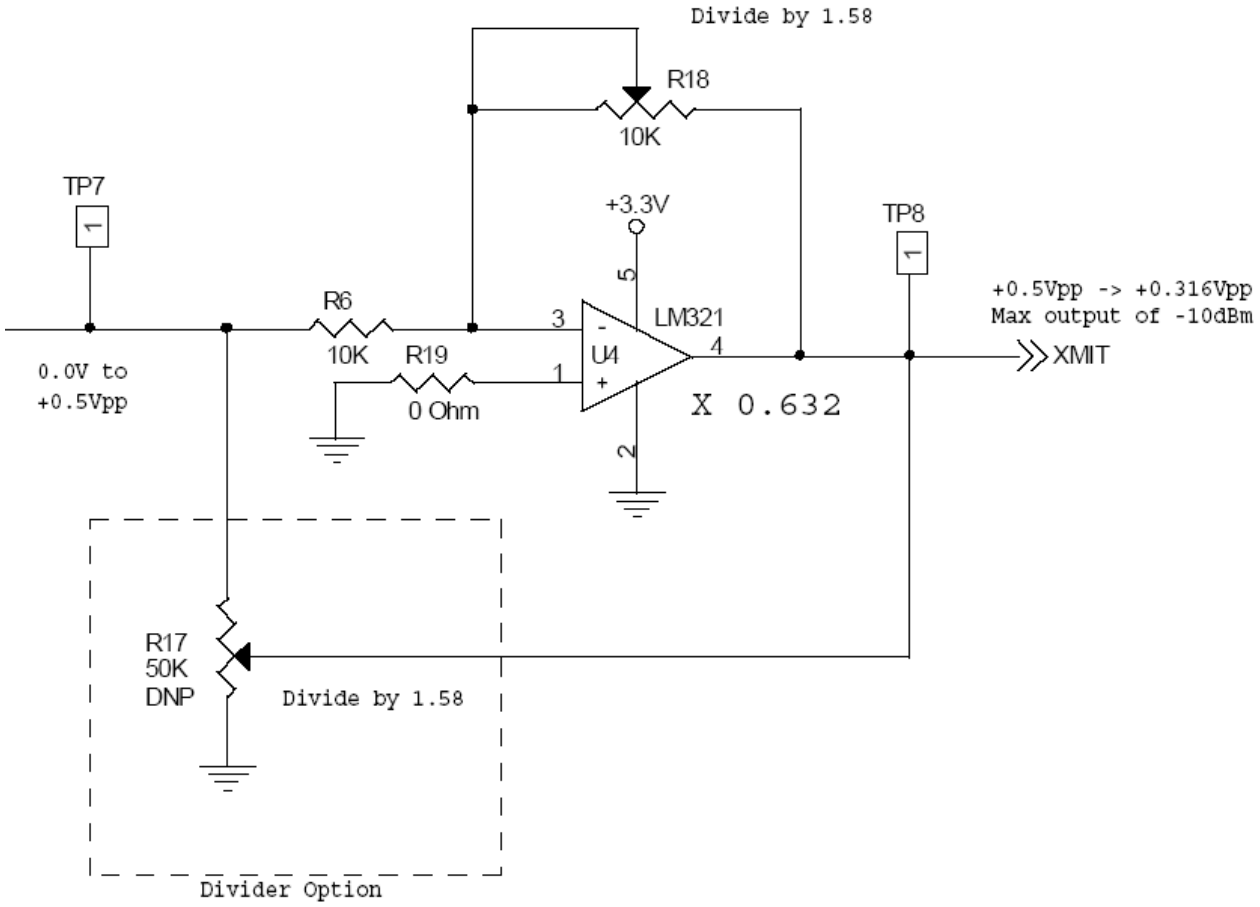


Figure 2-7. TP7 to Level Set to XMIT

2.4.2 Telephony Base Band Reception to Analog-to-Digital Converter

Level adjustment and signal offset calibration are performed with the circuit shown in **Figure 2-8.** The REC signal is from the DAA, the received telephony signal. After level adjustment and offset adjustment, the signal is passed directly to the MC56F8357's ADC, where it is sampled per the bean configuration shown in **Figures 3-7** through **3-9.**

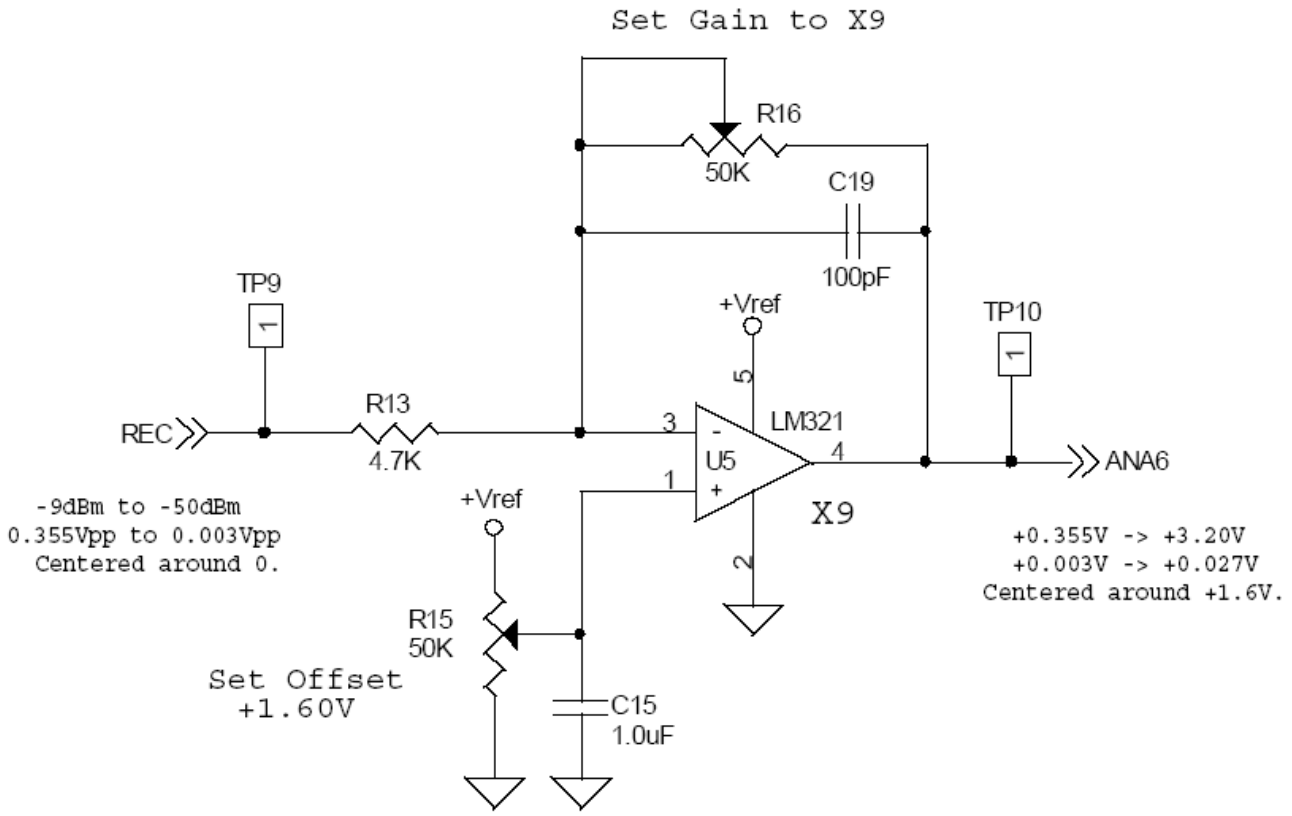


Figure 2-8. REC to Offset and Gain to ANA6

Complete schematics for the Low-Cost Modem Daughter Card (LCMDC) are included in Appendix A.

Note: The LCMDC is available for evaluation from the factory on request.

Chapter 3
Processor Expert Bean Utilization in the Soft Modem

Version 7.1 of Metrowerks CodeWarrior for the Freescale 56F8300 family was used to develop all of the code. Much of the code was generated automatically after GUI selections were made from simple menus presented after the basic system's (software/hardware combinations) building blocks were selected. Processor Expert beans encapsulate these basic system blocks. For the most part, this project was simply the selection and configuration of these beans.

All of the beans are documented in this section so that the modem may be constructed directly from these beans, together with the code which is supplied separately. A bean has properties, methods, and events. Once a bean is added to a project, it must be configured in these three areas. The complete list of properties, methods, and events is illustrated for each bean in the project, along with a commentary on the bean's usage. When Processor Expert is called upon to generate code, it will use the configured beans to generate most of the code required for the soft modem. The user-written code that must be added to complete the project is documented in a separate document; contact your Freescale representative.

Processor Expert simplified the construction of the software by encapsulating peripheral support into beans. For example, there is a bean for an asynchronous serial port, used to form a test channel over which the AT commands were interfaced to the modem. The properties include such items as baud rate, chosen to be 2400 baud. Another property is used to associate the bean with a particular set of pins on the device. Methods are a way to interface the main program to use the peripheral. Code for methods can be drag-and-dropped into the program. Events of the bean facilitate installing user-defined functions to be performed by an Interrupt Service Routine (ISR) used with the bean. Modem data pumps are also added as beans.

The software project is formed by first selecting blank Processor Expert Stationery, then adding and configuring the beans. User-written software programs are added as a final step. Pop-up help on the methods for using the beans along with the drag-and-drop feature makes using the processor simple.

The bean name (given at the time the bean is added to a project by the designer) is followed a colon, then the generic bean type. The soft modem software project consists of the beans depicted in **Figure 3-1**.

The soft modem has been implemented on several members of the DSC family, including the 56F8323, 56F846, 56F8357 (as in this note), and 56F8367 (as shipped with complete project and source code for demonstration/evaluation purposes).

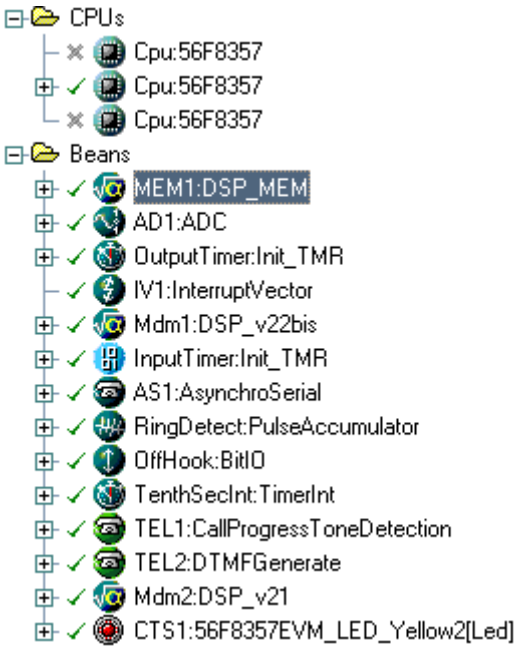


Figure 3-1. Modem Project Bean List

3.1 CPU Bean

There are three CPUs listed in the CPUs folder in **Figure 3-1**. The relevant CPU is checked. This is the one used in the active target, sdm pROM xRAM, or small memory model wherein the constants in data xRAM are initialized from program Flash (pROM). The CPU bean has the default settings. Double clicking on the CPU bean shows its properties, as shown in **Figure 3-3**. It's easy to replace this bean with one for the other EVM's in this family, such as the 56F8367EVM, which contain more system resources.

The pROM xRAM target was chosen to develop this modem because there is plenty of pROM, so the constants that must be used to initialize part of RAM are stored first in pROM. The alternative would be to store them in xROM or to have the debugger run the application out of RAM. In order for the board to run without the debugger, the most intelligent option is to store the constants in pROM, then copy them to xRAM. This allows the entire application to be stored in nonvolatile memory with minimal use of scarce resources, explaining the pROM xRAM target choice.

One of the important tabs of the CPU bean is the *build options* tab. Note the dynamic memory allocation at the bottom of the *build options* list. This allows most of the memory for the soft modem to be deallocated when the soft modem is not in use.

Bean name	Cpu	
CPU type	56F8357	
Oscillator frequency [MHz]	8	8 MHz
Initialization priority	minimal priority	0
Temp sensor	Disabled	
Saturation mode	Disabled	
Initialize shadow registers	no	
Initialize unused I/O pins	no initialization	
Internal peripherals		
SIM module		
Flash security & protection	Disabled	
Peripheral clocks		
Allocated interrupts		
Interrupt SW0	Enabled	
Interrupt	INT_SW0	INT_SW0
Interrupt priority	medium priority	non maskable
Interrupt SW1	Enabled	
Interrupt	INT_SW1	INT_SW1
Interrupt priority	medium priority	non maskable
Interrupt SW2	Enabled	
Interrupt	INT_SW2	INT_SW2
Interrupt priority	medium priority	non maskable
Interrupt SW3	Enabled	
Interrupt	INT_SW3	INT_SW3
Interrupt priority	medium priority	non maskable
Interrupt LP	Enabled	
Interrupt	INT_LP	INT_LP
Interrupt priority	medium priority	1
Interrupt PLL	Enabled	
Interrupt	INT_PLL	INT_PLL
Interrupt priority	medium priority	1
Interrupt LVI	Disabled	
Interrupt Illegal Instruction	Enabled	
Interrupt	INT_Illegal_Instruction	INT_Illegal_Instruction
Interrupt priority	medium priority	non maskable
Interrupt Misalign Access	Disabled	
Interrupt HW Stack Overflow	Enabled	
Interrupt	INT_HWStackOverflow	INT_HWStackOverflow
Interrupt priority	medium priority	non maskable
External bus	Disabled	
Enabled speed modes		
High speed mode	Enabled	
System clock (IP Bus)	60.0	60.0 MHz
PLL clock	Enabled	
PLL clock frequency	120.0	Xtal * 60 / 4
Low speed mode	Disabled	
Slow speed mode	Disabled	

Figure 3-2. Cpu:56F8357 Properties

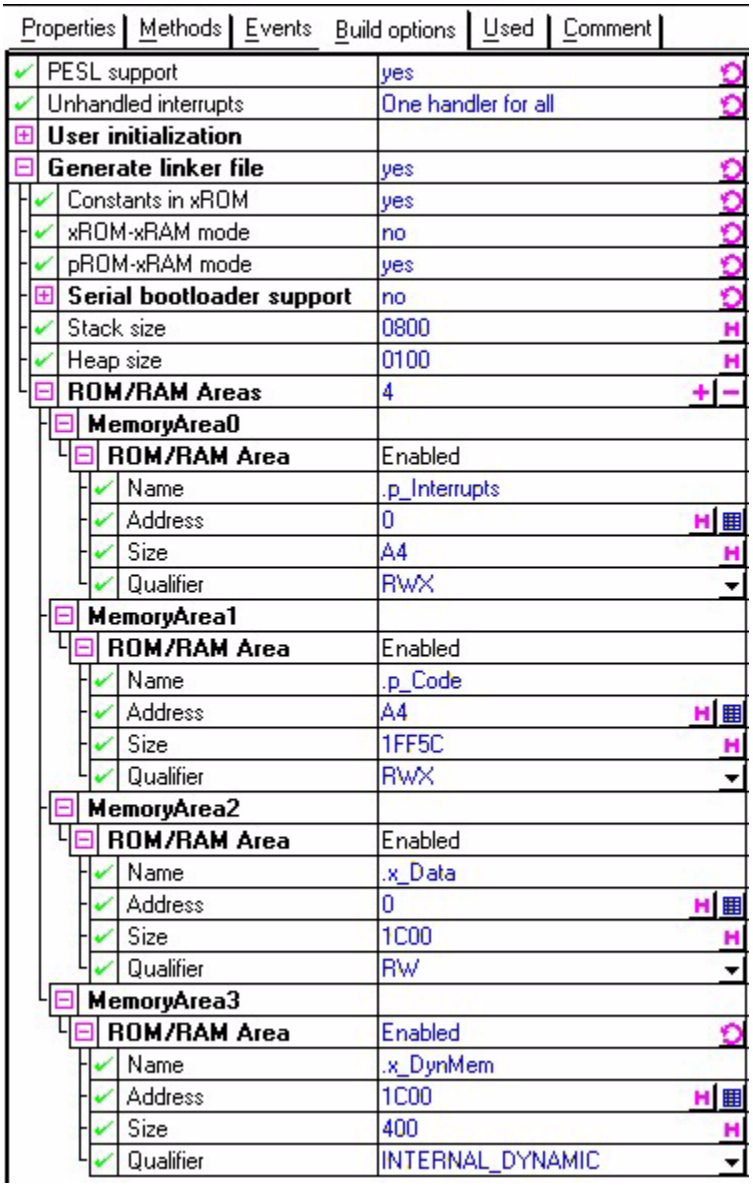


Figure 3-3. Cpu:56F8357 on sdm pROM-xRAM target

3.2 *MEM1:DSP_MEM Bean*

This bean defines internal xRAM for use by the other related beans. Internal memory is much faster than external memory. The most notable feature on this configuration is the *.x_Dynmem* at the bottom of **Figure 3-3**, which must be present in order to provide dynamic memory services for the soft modem. Since the modem is not the only function in a real application, when it is not in operation, it is recommended that it give back all of its RAM memory. This can be most easily accomplished with the help of the dynamic memory services provided by Processor Expert.

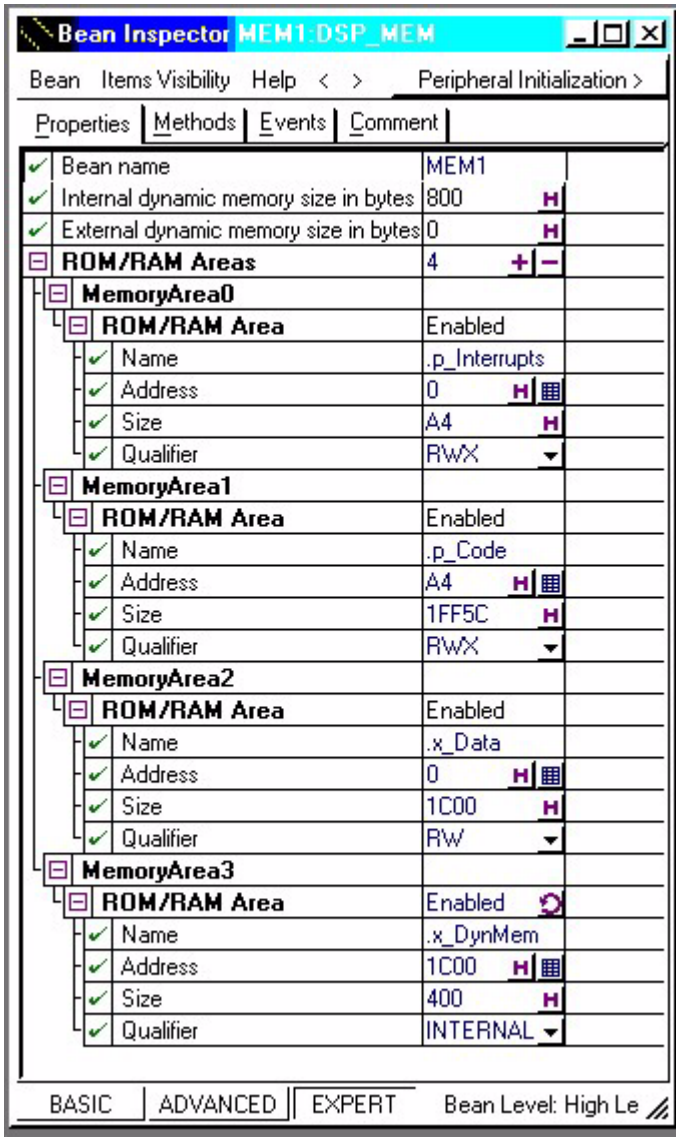


Figure 3-4. *MEM1:DSP_MEM* Properties

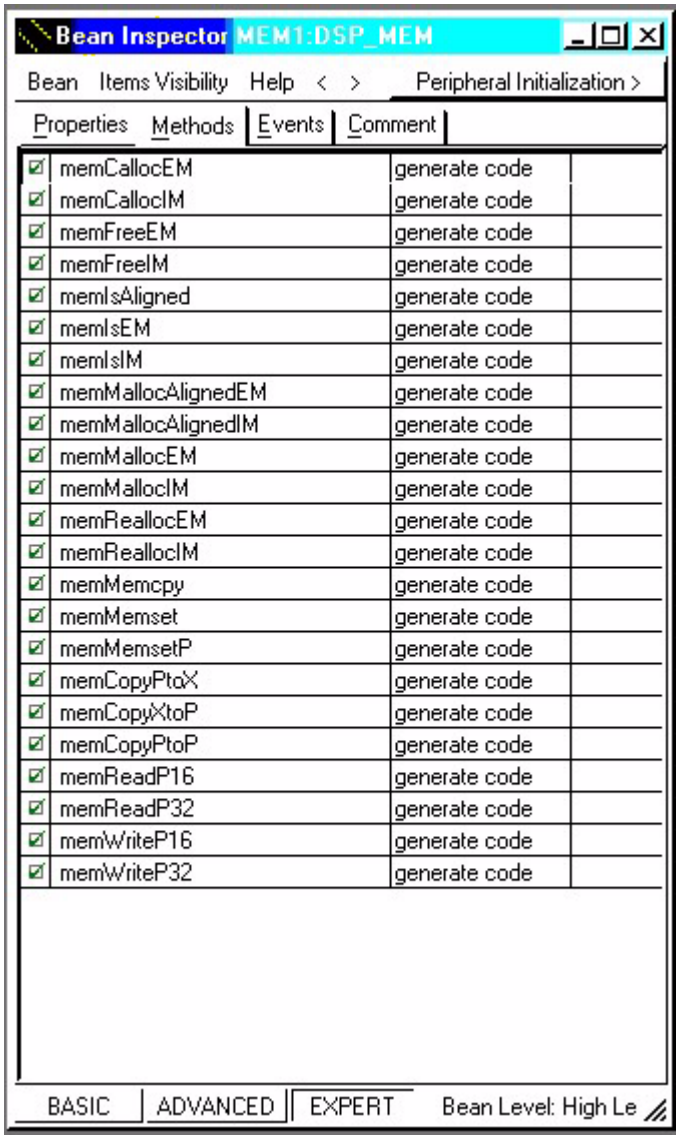


Figure 3-5. *MEM1:DSP_MEM* Methods

These are the various methods available for the algorithms to allocate memory. A major advantage of having so many classes of memory available is to allow the algorithm to specify the arrangement of operands in memory to minimize wait states.

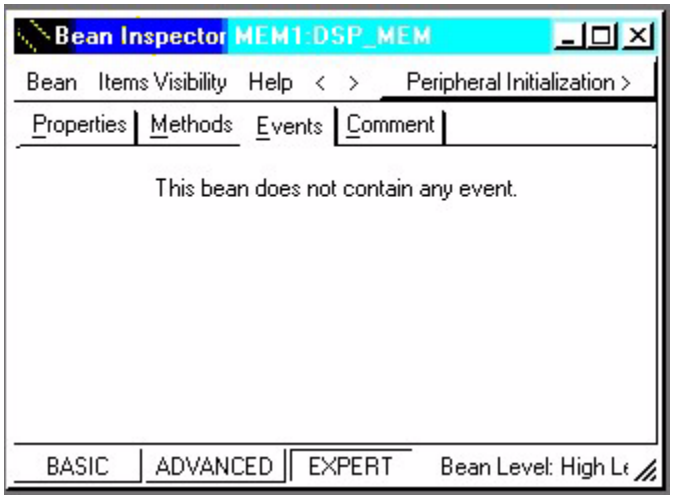


Figure 3-6. *MEM1:DSP_MEM* Events

3.3 The *ADC1:ADC* Bean

The *InputTimer* is used to trigger the ADC conversion. Once it is triggered, it takes eight rapid-fire samples, which are later averaged. The averaging reduces noise contribution to the signal.

Bean name	AD1	
A/D converter	ADCB	ADCB
Sharing	Disabled	
Interrupt service/event	Enabled	
A/D interrupt	INT_ADCB_Complete	INT_ADCB_Complete
A/D interrupt priority	medium priority	1
Interrupt preserve registers	yes	
Interrupt	INT_ADCB_ZC_LE	INT_ADCB_ZC_LE
Interrupt priority	medium priority	1
Interrupt preserve registers	yes	
A/D channels	1	
Channel0		
A/D channel (pin)	ANB6	ANB6
A/D channel (pin) signal		
Mode select	Single Ended	
Queue	Enabled	
Mode	Sequential	
A/D samples	8	
Sample0	Enabled	
Channel	0	
High limit	7FF8	
Low limit	0	
Offset	16376	
Zero crossing	Disabled	
Sample1	Disabled	
Sample2	Disabled	
Sample3	Disabled	
Sample4	Disabled	
Sample5	Disabled	
Sample6	Disabled	
Sample7	Disabled	
A/D prescaler	ADCB_ADCR2	ADCB_ADCR2
A/D resolution	Autoselect	12 bits
Conversion time	8.500 µs	high: 8.217 µs
Internal trigger	Enabled	
Trigger source	InputTimer	
Sync from PWM	no	
Volt. ref. recovery time	100	
Power up delay	13	
Power savings mode	Disabled	
Number of conversions	1	
Initialization		
Enabled in init. code	yes	
Events enabled in init.	yes	
CPU clock/speed selection		
High speed mode	This bean enabled	This bean is enabled
Low speed mode	This bean disabled	This bean is disabled
Slow speed mode	This bean disabled	This bean is disabled

Figure 3-7. *AD1:ADC* Properties

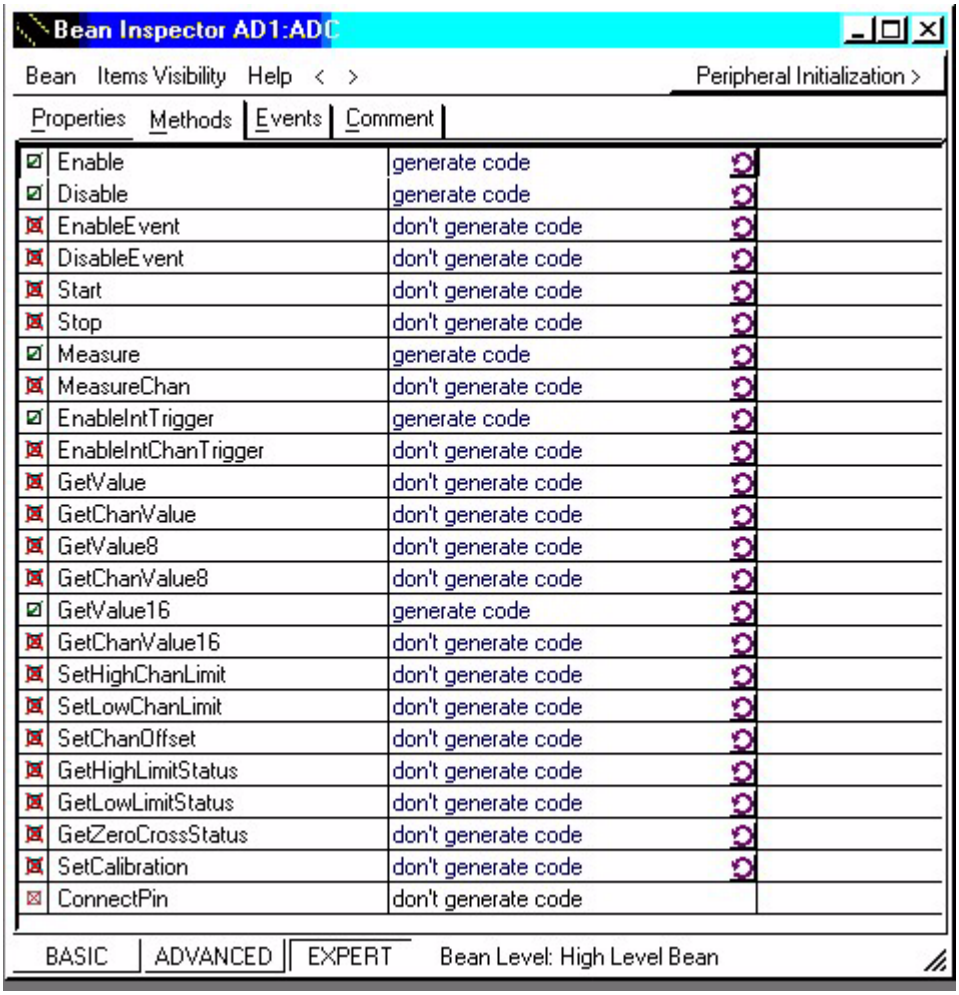


Figure 3-8. *AD1:ADC* Methods

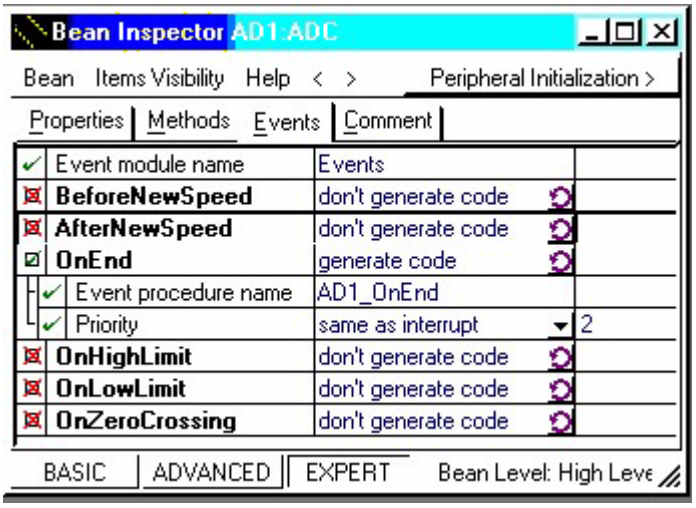


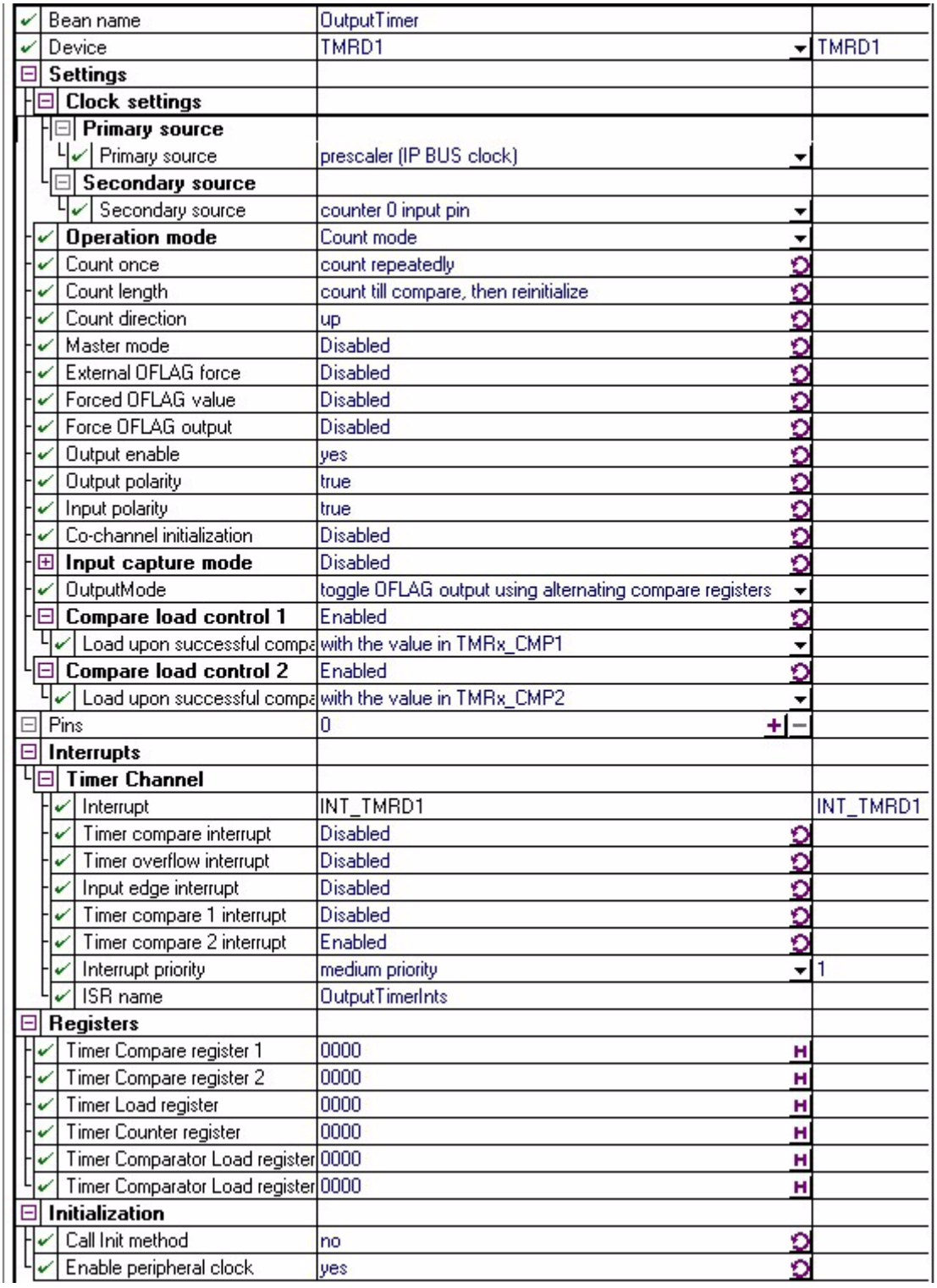
Figure 3-9. *AD1:ADC* Events

3.4 *OutputTimer:Init_TMR* Bean

This bean supplies the PWM signal which is converted to an analog output signal by the low-pass filter, then passed to the DAA and, finally, to the telephone circuit. After the code is generated, the duty cycle is manipulated with PESL commands. The frequency remains constant, as does the “sampling” rate for this output.

Most of the code runs with sampling rates of 7200 samples per second. Only the call progress function runs at 8000 samples per second, but since it does not output, it does not use this bean. Therefore, this bean is always used to output 7200 samples per second in the soft modem.

This implies that the cutoff of the subsequent low-pass filter *could* be 3600Hz, rather than 4000Hz. However, any resultant artifacts are out of the signal band but within the required channel band and should not contribute to noise. Thus, 4000Hz is a good figure to use.



Bean name	OutputTimer	
Device	TMRD1	TMRD1
Settings		
Clock settings		
Primary source		
Primary source	prescaler (IP BUS clock)	
Secondary source		
Secondary source	counter 0 input pin	
Operation mode	Count mode	
Count once	count repeatedly	
Count length	count till compare, then reinitialize	
Count direction	up	
Master mode	Disabled	
External OFLAG force	Disabled	
Forced OFLAG value	Disabled	
Force OFLAG output	Disabled	
Output enable	yes	
Output polarity	true	
Input polarity	true	
Co-channel initialization	Disabled	
Input capture mode	Disabled	
OutputMode	toggle OFLAG output using alternating compare registers	
Compare load control 1	Enabled	
Load upon successful compa	with the value in TMRx_CMP1	
Compare load control 2	Enabled	
Load upon successful compa	with the value in TMRx_CMP2	
Pins	0	
Interrupts		
Timer Channel		
Interrupt	INT_TMRD1	INT_TMRD1
Timer compare interrupt	Disabled	
Timer overflow interrupt	Disabled	
Input edge interrupt	Disabled	
Timer compare 1 interrupt	Disabled	
Timer compare 2 interrupt	Enabled	
Interrupt priority	medium priority	1
ISR name	OutputTimerInts	
Registers		
Timer Compare register 1	0000 H	
Timer Compare register 2	0000 H	
Timer Load register	0000 H	
Timer Counter register	0000 H	
Timer Comparator Load register	0000 H	
Timer Comparator Load register	0000 H	
Initialization		
Call Init method	no	
Enable peripheral clock	yes	

Figure 3-10. *OutputTimer:Init_TMR* Properties

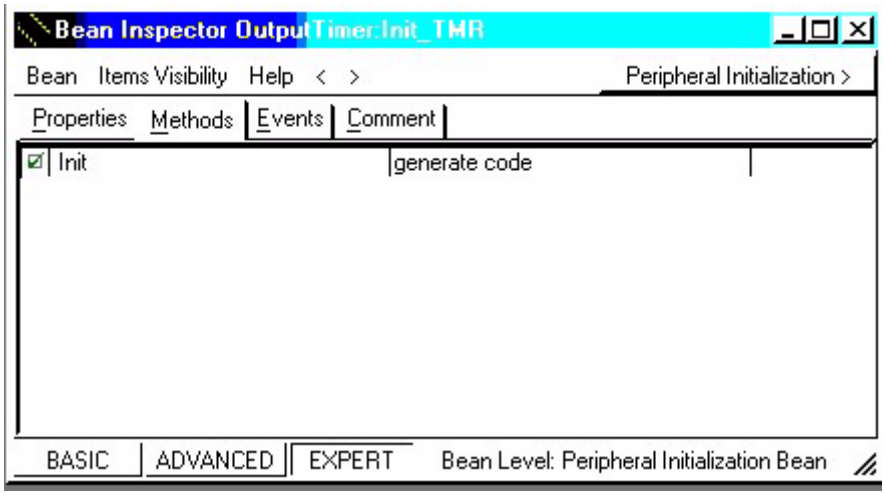


Figure 3-11. *OutputTimer:Init_TMR* Methods

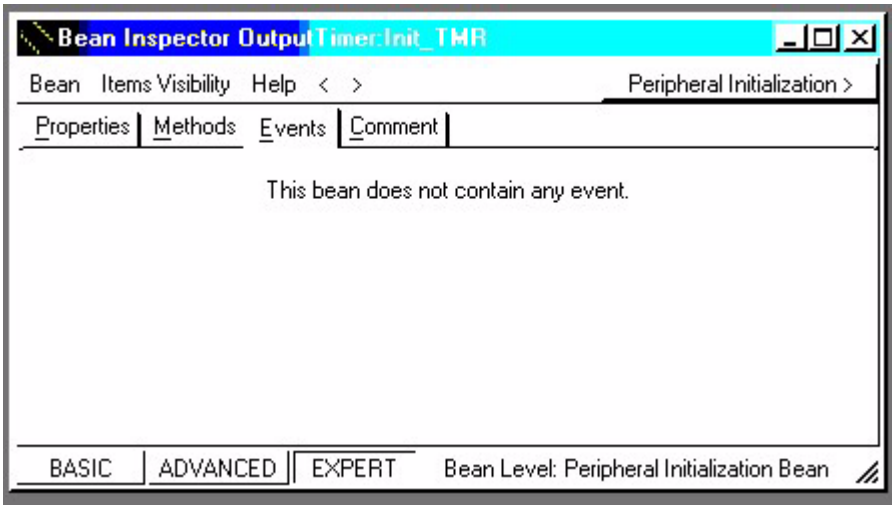


Figure 3-12. *OutputTimer:Init_TMR* Events

3.5 *IV1:InterruptVector* Bean

The misaligned LongWordISR is supplied only to detect a misaligned LongWord interrupt. This does not occur in the project as provided, but is useful to illustrate how to hook an ISR.

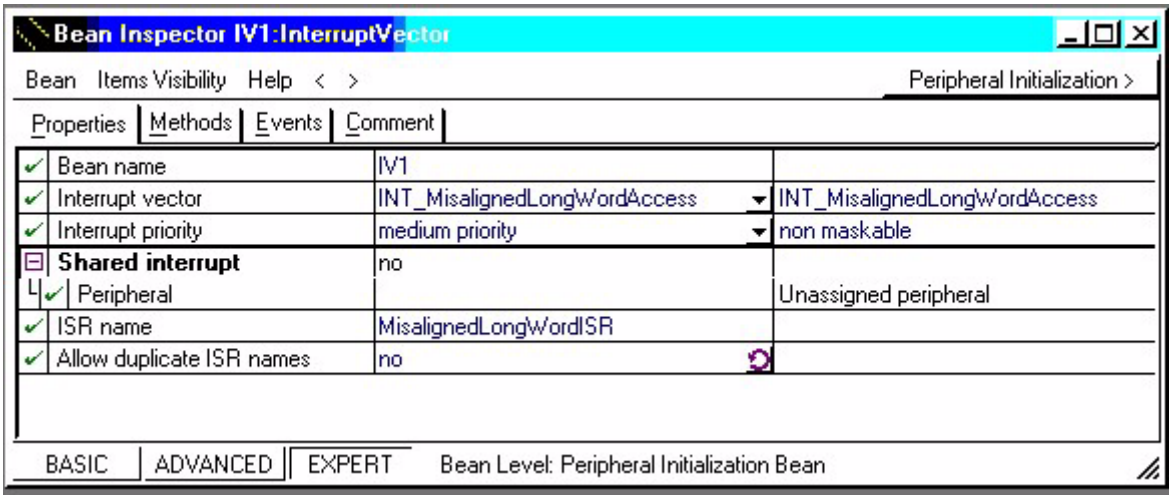


Figure 3-13. *IV1:InterruptVector* Properties

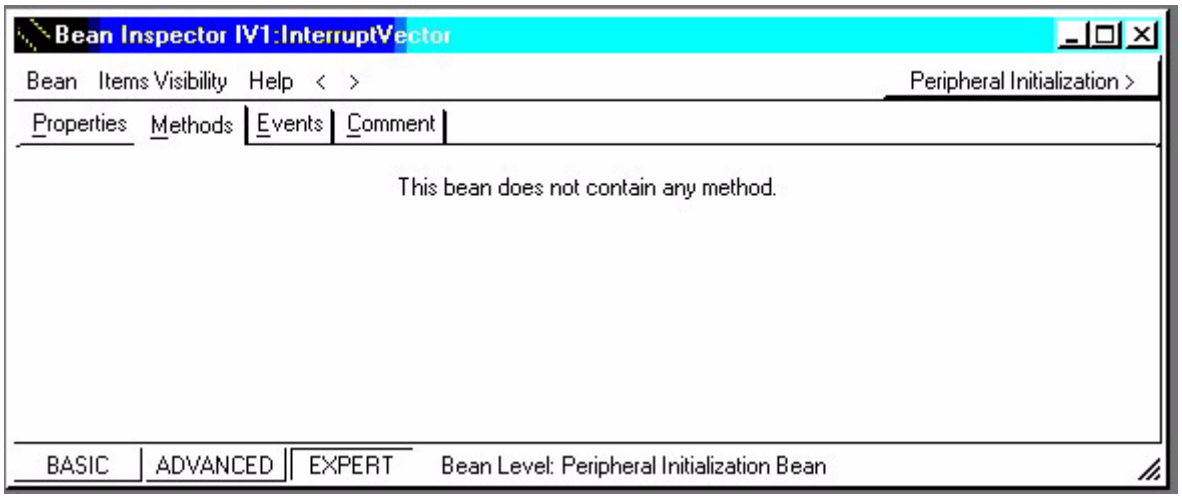


Figure 3-14. *IV1:InterruptVector* Methods

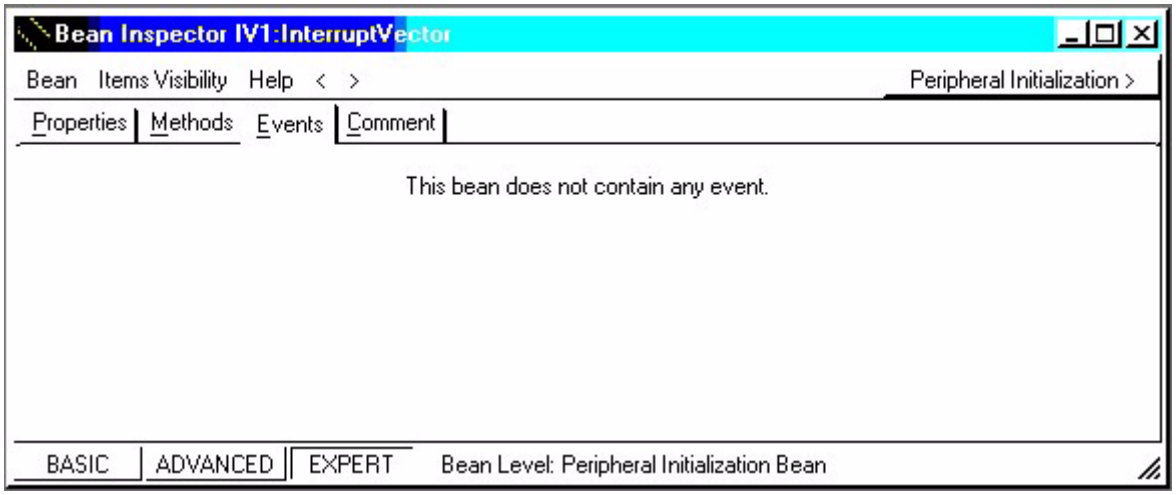


Figure 3-15. *IV1:InterruptVector* Events

This bean does not generate any place to put “event” code in *Events.c*. Instead, it is up to the programmer to locate his ISR code.

3.6 *Mdm1:DSP_v22bis* Bean

This software bean generates modem code for the V22bis/V.22 modem data pump. Most of the code is written in optimized Assembly language. It also generates the API in C. Memory Management makes it possible to set up the modem on a per-call basis. Since V.21 rather than V.22bis may be in use, it is useful to preserve this dynamic property. When a V.21 call is on, the V.22bis call’s resource, its memory, is deallocated, and vice versa. Also, when there is no active call, the modem is not taking RAM resources, because the Memory Management feature is used.

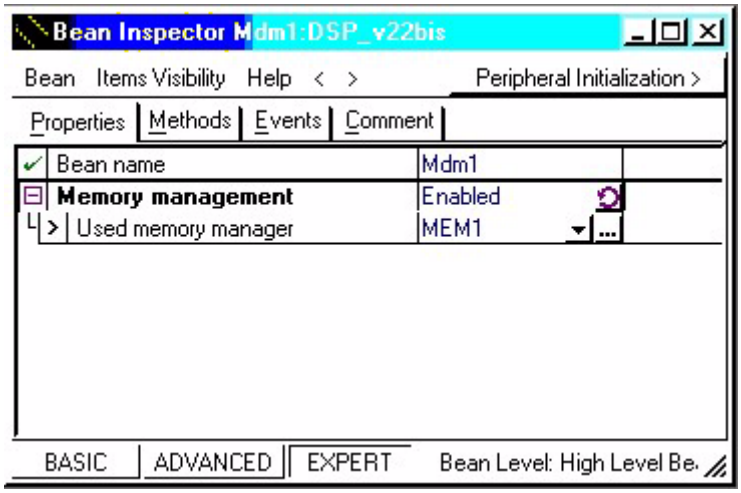


Figure 3-16. *Mdm1:DSP_v22bis* Properties

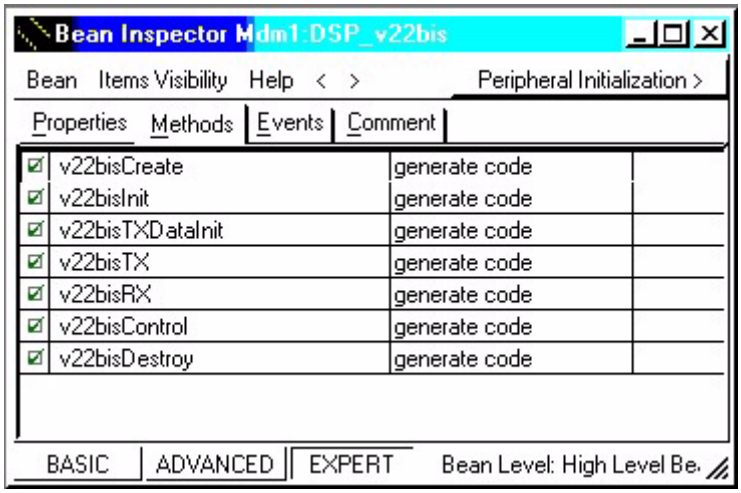


Figure 3-17. *Mdm1:DSP_v22bis* Methods

The *v22bisControl* method is for future expansion of the data pump. Section 3.7 illustrates the use of other methods.

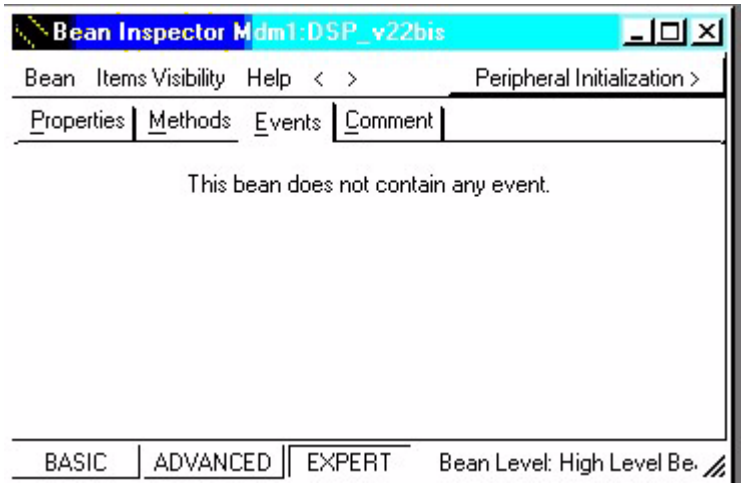
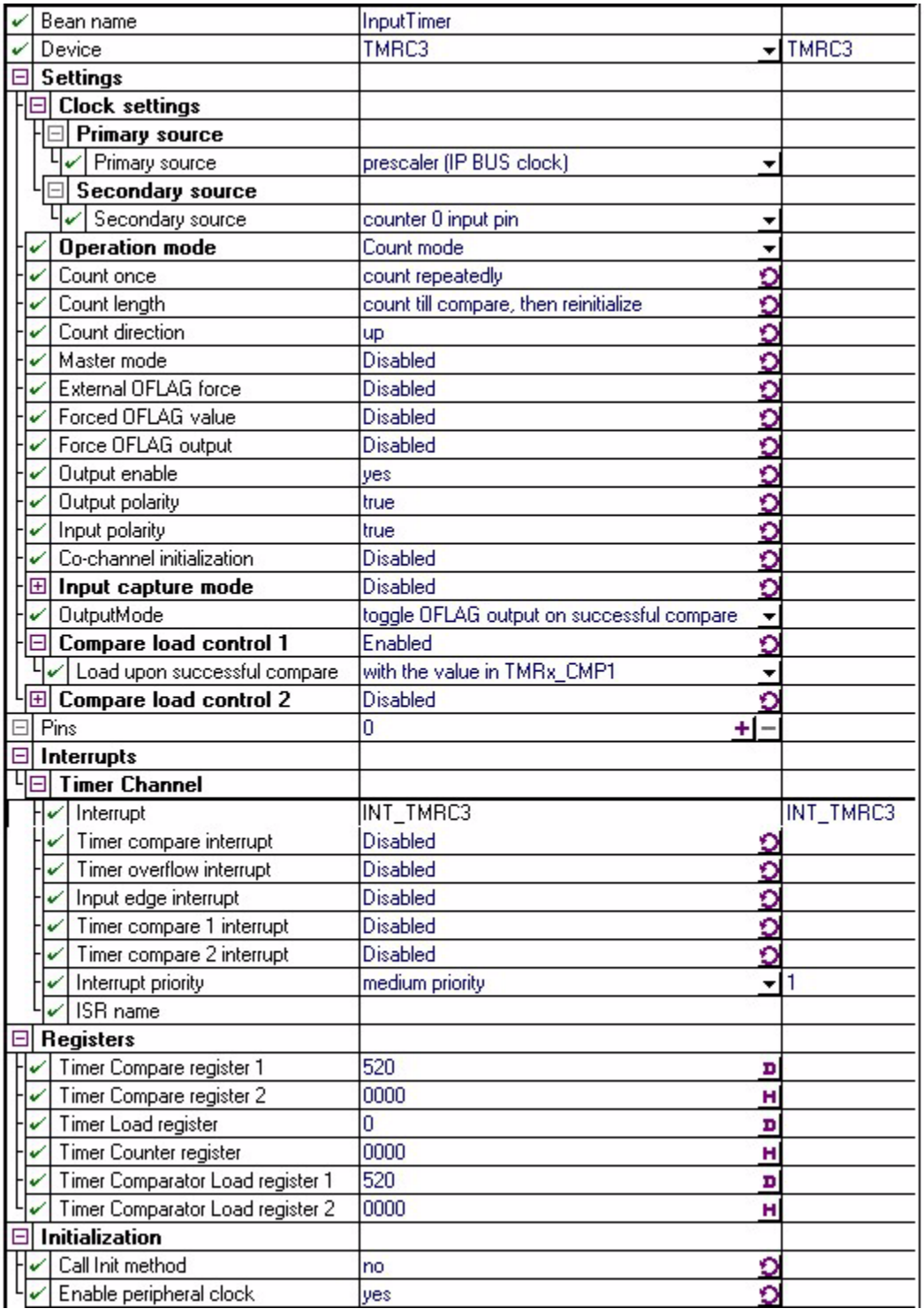


Figure 3-18. Mdm1:DSP_v22bis Events

No events are needed because the modem is called. When it has data or needs data, it calls the application program back at a designated function to deliver or collect data. Since no event is needed, there is no context switch and the code is more efficient.

3.7 *InputTimer:Init_TMR* Bean

This bean is responsible for timing the ADC sample reads and normally operates at 7200 samples per second. Using the PESL commands, the peripheral is modified to sample at 8000 samples per second during call progress. This dynamic sampling rate avoids the use of any sample rate conversion code.



Bean name	InputTimer	
Device	TMRC3	TMRC3
Settings		
Clock settings		
Primary source		
Primary source	prescaler (IP BUS clock)	
Secondary source		
Secondary source	counter 0 input pin	
Operation mode	Count mode	
Count once	count repeatedly	
Count length	count till compare, then reinitialize	
Count direction	up	
Master mode	Disabled	
External OFLAG force	Disabled	
Forced OFLAG value	Disabled	
Force OFLAG output	Disabled	
Output enable	yes	
Output polarity	true	
Input polarity	true	
Co-channel initialization	Disabled	
Input capture mode	Disabled	
OutputMode	toggle OFLAG output on successful compare	
Compare load control 1	Enabled	
Load upon successful compare	with the value in TMRx_CMP1	
Compare load control 2	Disabled	
Pins	0	
Interrupts		
Timer Channel		
Interrupt	INT_TMRC3	INT_TMRC3
Timer compare interrupt	Disabled	
Timer overflow interrupt	Disabled	
Input edge interrupt	Disabled	
Timer compare 1 interrupt	Disabled	
Timer compare 2 interrupt	Disabled	
Interrupt priority	medium priority	1
ISR name		
Registers		
Timer Compare register 1	520	
Timer Compare register 2	0000	
Timer Load register	0	
Timer Counter register	0000	
Timer Comparator Load register 1	520	
Timer Comparator Load register 2	0000	
Initialization		
Call Init method	no	
Enable peripheral clock	yes	

Figure 3-19. *InputTimer:Init_TMR* Properties

To alter the sampling rate without having to stop or reset anything, the Timer Compare Register 1 is changed to a smaller number on the fly, effecting 8000 samples per second with PESL commands during call progress detection. Be sure to enable the PESL commands in the project, since the soft modem uses them. If this is not enabled, the compiler will report no prototype.

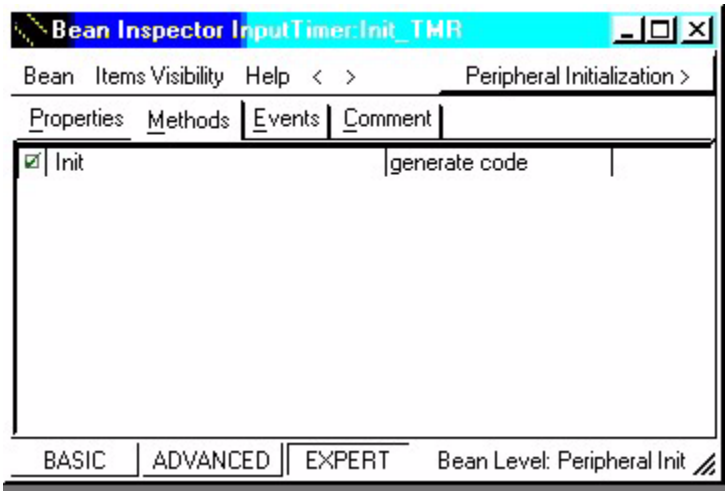


Figure 3-20. *InputTimer:Init_TMR* Methods

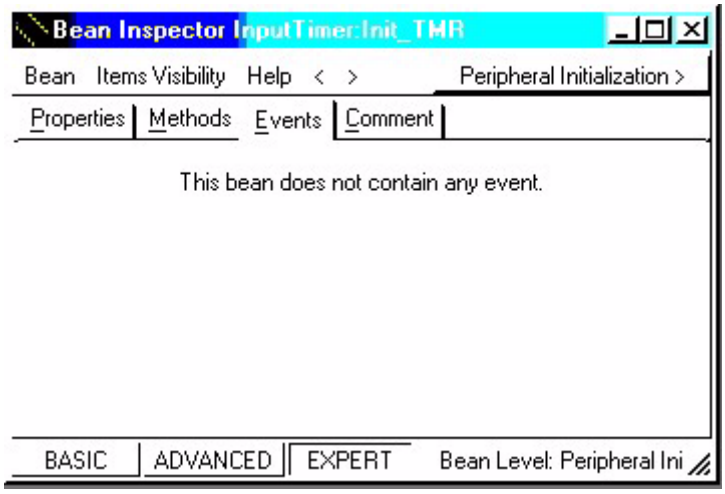
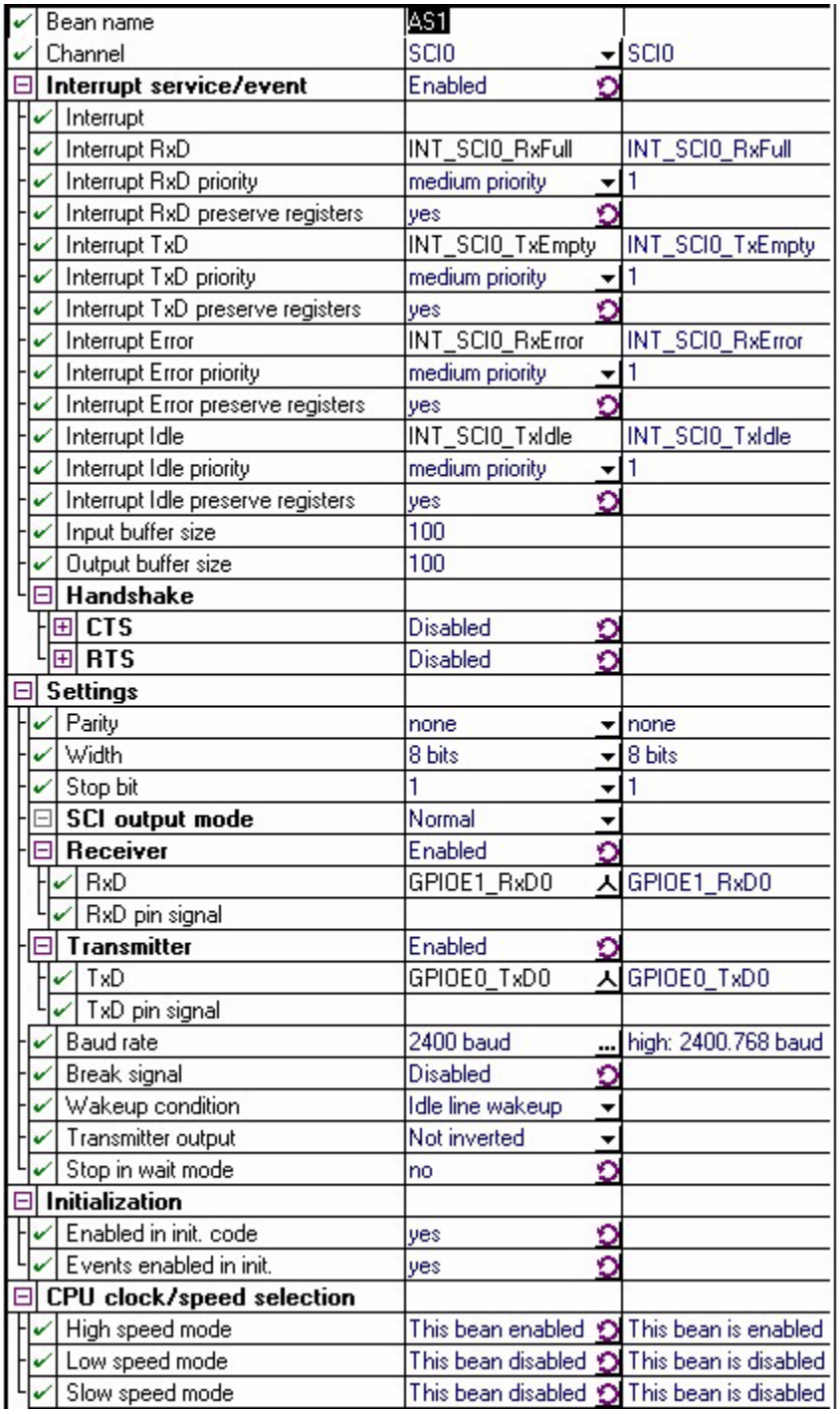


Figure 3-21. *InputTimer:Init_TMR* Events

3.8 *AS1:AsynchroSerial* Bean

This bean was set up for 2400 baud. It actually does not matter what baud rate is used, as long as it is at least 2400 baud. The buffering is done in the modem, and hardware flow control assures that the modem is not overrun with data. Both V.21 and V.22bis use 2400 baud; the rate conversion is done by the modem.



Bean name	AS1	
Channel	SCI0	SCI0
Interrupt service/event	Enabled	
Interrupt		
Interrupt RxD	INT_SCI0_RxFull	INT_SCI0_RxFull
Interrupt RxD priority	medium priority	1
Interrupt RxD preserve registers	yes	
Interrupt TxD	INT_SCI0_TxEmpty	INT_SCI0_TxEmpty
Interrupt TxD priority	medium priority	1
Interrupt TxD preserve registers	yes	
Interrupt Error	INT_SCI0_RxError	INT_SCI0_RxError
Interrupt Error priority	medium priority	1
Interrupt Error preserve registers	yes	
Interrupt Idle	INT_SCI0_TxIdle	INT_SCI0_TxIdle
Interrupt Idle priority	medium priority	1
Interrupt Idle preserve registers	yes	
Input buffer size	100	
Output buffer size	100	
Handshake		
CTS	Disabled	
RTS	Disabled	
Settings		
Parity	none	none
Width	8 bits	8 bits
Stop bit	1	1
SCI output mode	Normal	
Receiver	Enabled	
RxD	GPIOE1_RxD0	GPIOE1_RxD0
RxD pin signal		
Transmitter	Enabled	
TxD	GPIOE0_TxD0	GPIOE0_TxD0
TxD pin signal		
Baud rate	2400 baud	high: 2400.768 baud
Break signal	Disabled	
Wakeup condition	Idle line wakeup	
Transmitter output	Not inverted	
Stop in wait mode	no	
Initialization		
Enabled in init. code	yes	
Events enabled in init.	yes	
CPU clock/speed selection		
High speed mode	This bean enabled	This bean is enabled
Low speed mode	This bean disabled	This bean is disabled
Slow speed mode	This bean disabled	This bean is disabled

Figure 3-22. *AS1:AsynchroSerial* Properties

Notice that block, not character, input/output is used. All interrupts are at the same priority throught the project.

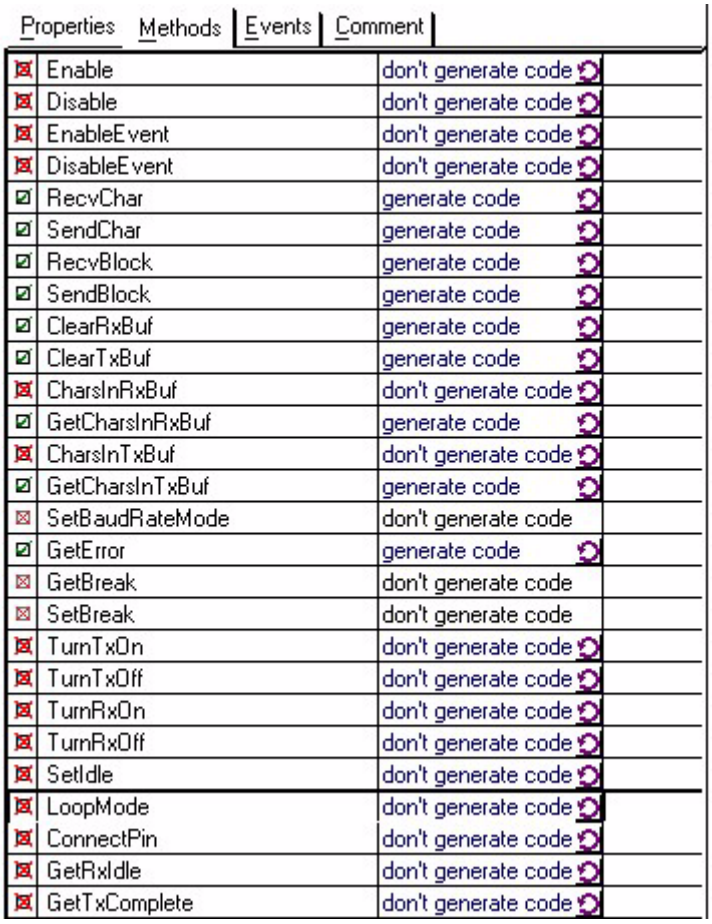


Figure 3-23. *AS1:AsynchroSerial* Methods

Use *SendBlock* to send a block of characters to the serial port and use *RecvBlock* to receive a block of characters.

Simply drag-and-drop these methods into the application code as required.

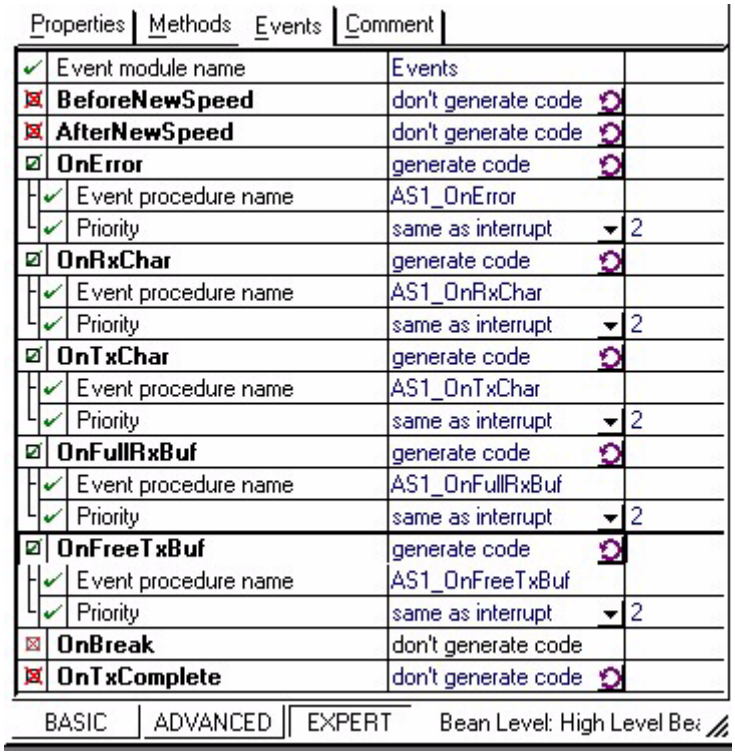


Figure 3-24. *AS1:AsynchroSerial* Events

3.9 *RingDetect:PulseAccumulator* Bean

Ring detect uses the Quadrature Decoder hardware to preprocess the ring signal, reducing the MIP load to process it and filtering out spurious ring signals.

Early telephones used a human-crank-powered generator to generate the high voltage ring signal. Generators and motors are really the same, except that one is driven mechanically and the other electrically. That is why the Quadrature Decoder, normally used to monitor motors, is an ideal peripheral to count and filter the ring pulses.

Note the filtered input property highlighted in **Figure 3-25**, which cuts down on CPU interrupts by a factor of 255. It also gives a very robust ring detection, filtering out events such as “tinkle”, which occurs when an extension on the line is dialed with pulse dialing, or repeatedly lifted and replaced in the cradle. Some ring detectors will falsely trigger on tinkling. Because the quadrature decoder hardware filters out such events, the CPU is not even interrupted for a tinkle.

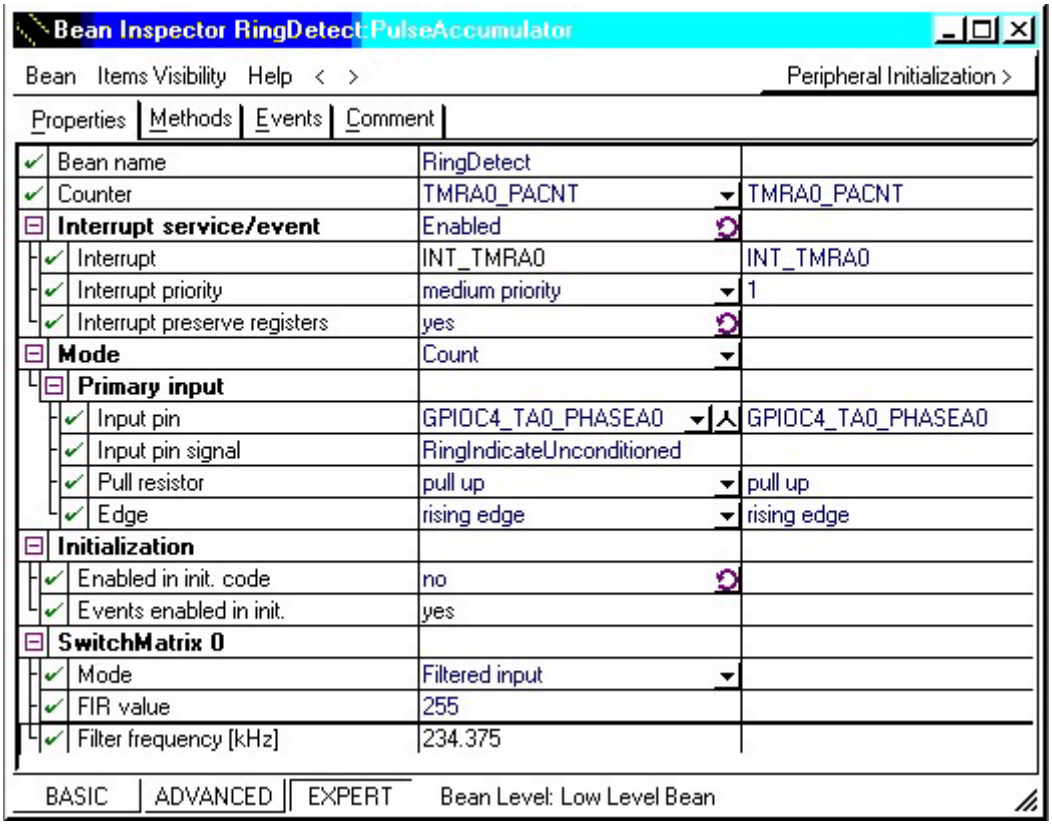


Figure 3-25. *RingDetect:PulseAccumulator* Properties

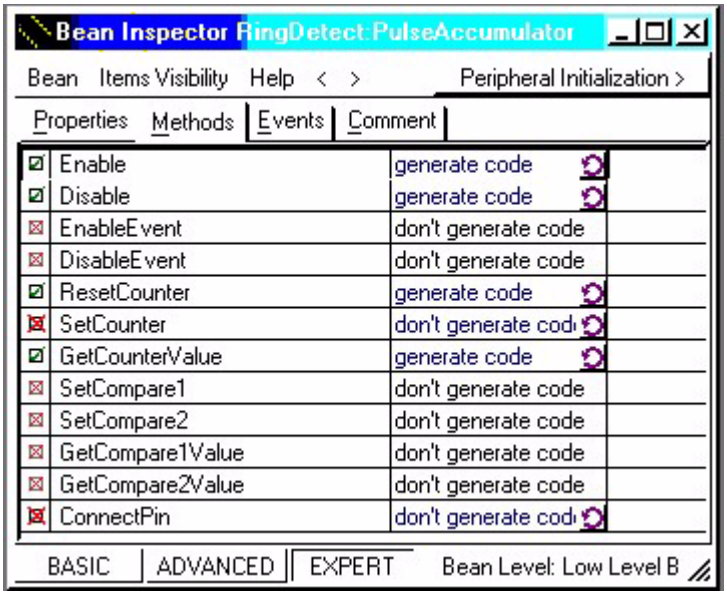


Figure 3-26. *RingDetect:PulseAccumulator* Methods

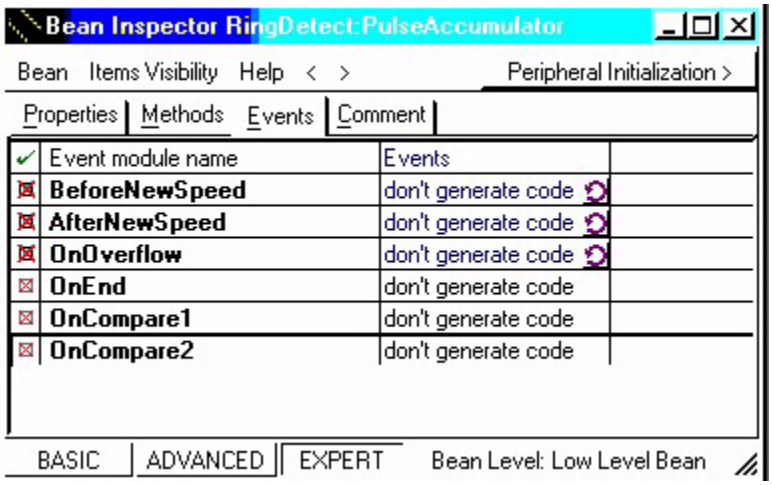


Figure 3-27. *RingDetect:PulseAccumulator* Events

3.10 *OffHook:BitIO* Bean

This *bitIO* is used to control the onhook/offhook status of the DAA. Never take the phone from the onhook to the offhook condition while the phone is ringing, as this may damage the DAA circiuts. The autoanswer feature of the soft modem assures that this will not happen.

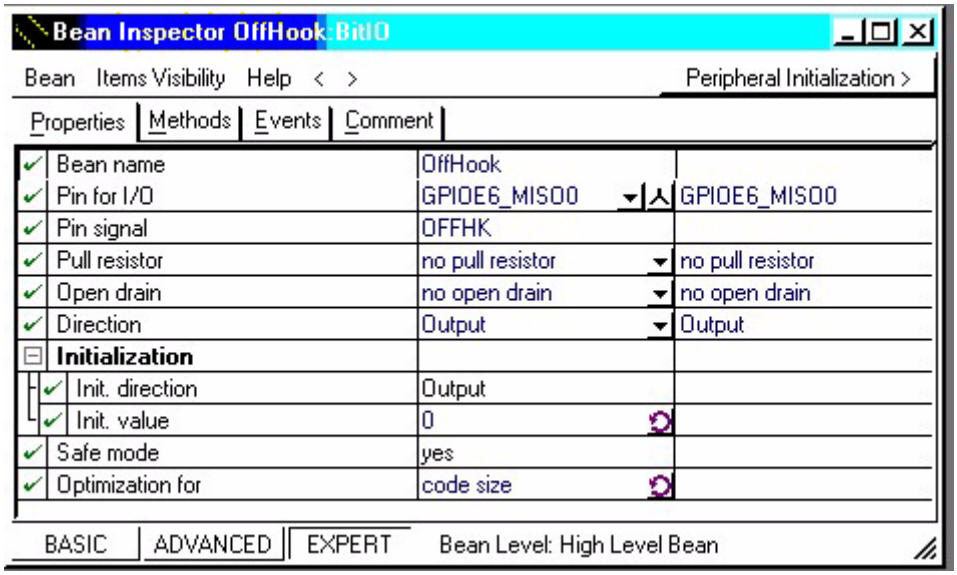


Figure 3-28. *OffHook:BitIO* Properties

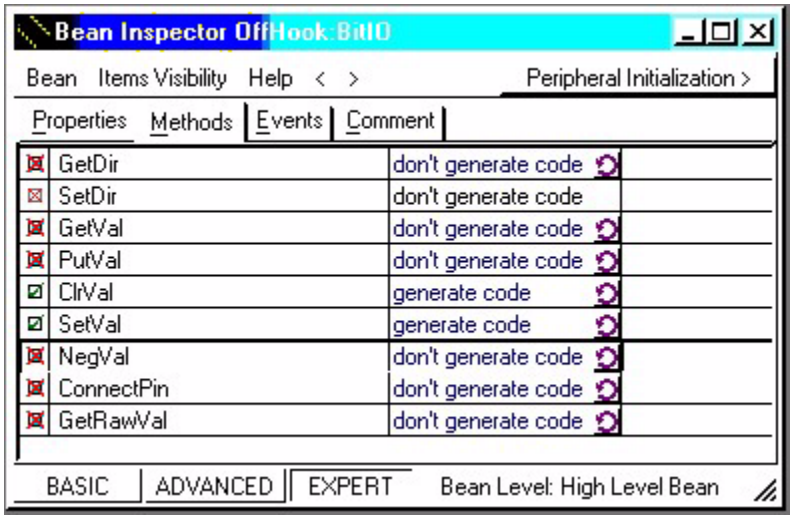


Figure 3-29. *OffHook:BitIO* Methods

ClrVal and *SetVal* are methods used to take the modem on and off the hook. When onhook, the modem is not connected to the phone line, but when offhook, it is connected. This term again goes back to the antique telephone design.

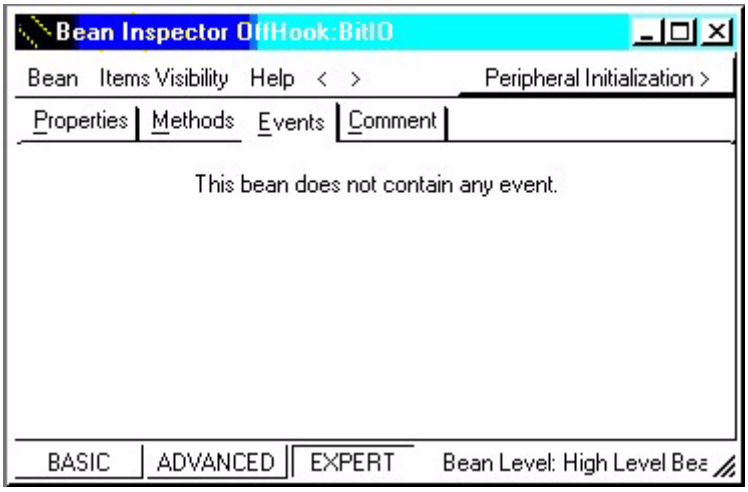


Figure 3-30. *OffHook:BitIO* Events

3.11 *TenthSecInt:TimerInt* Bean

This bean is used to generate an interrupt once every 1/10 second for the ring-detection and AT command set state machines. The AT command set must run a three-second timer when it is online (connected to another modem), to prevent data of '+++' being construed as a request to enter the online commnand state.

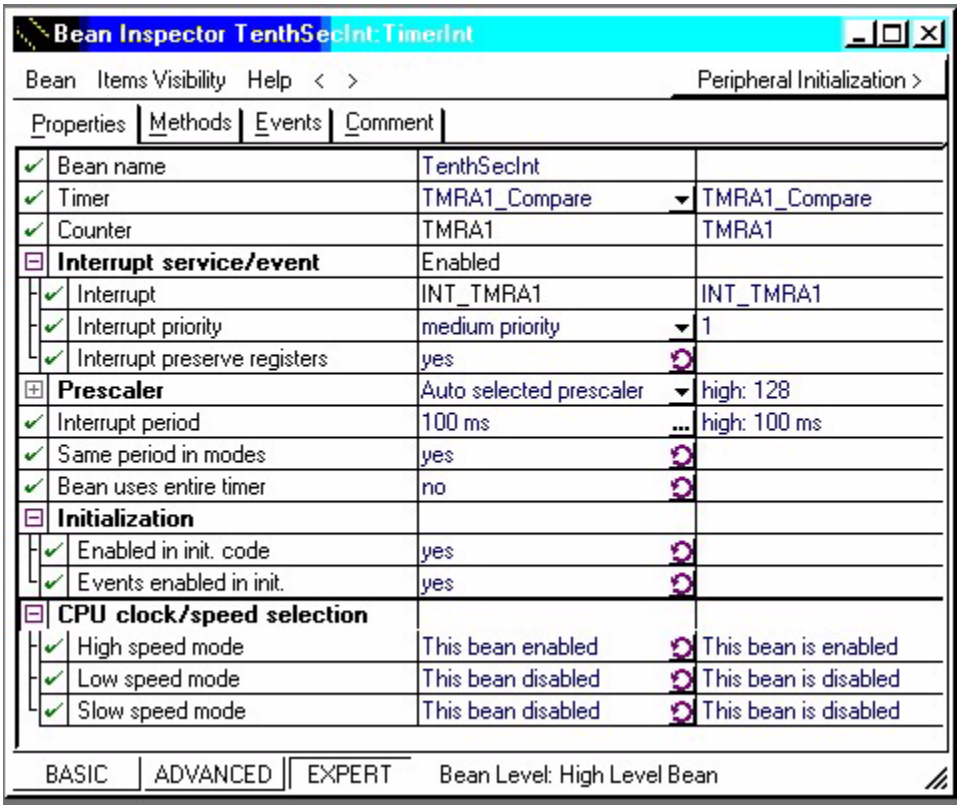


Figure 3-31. *TenthSecInt:TimerInt* Properties

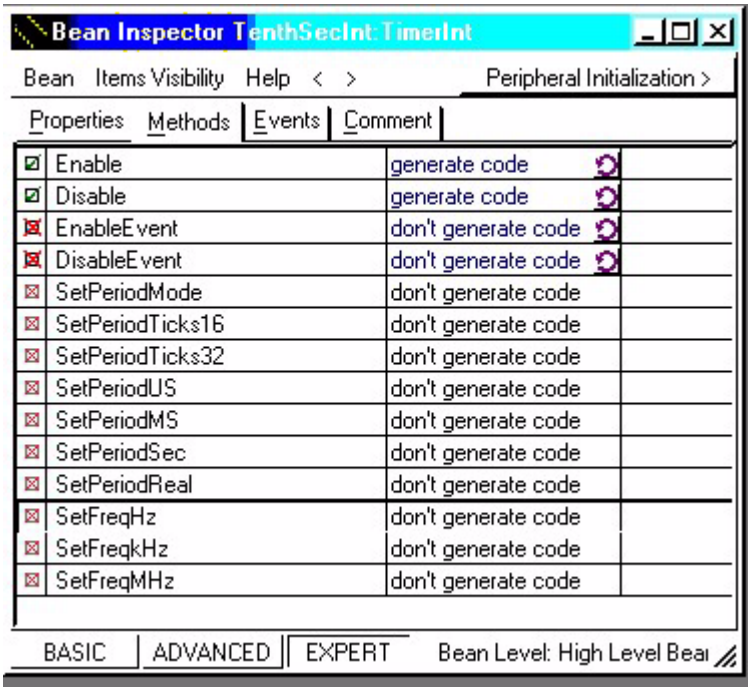


Figure 3-32. *TenthSecInt:TimerInt* Methods

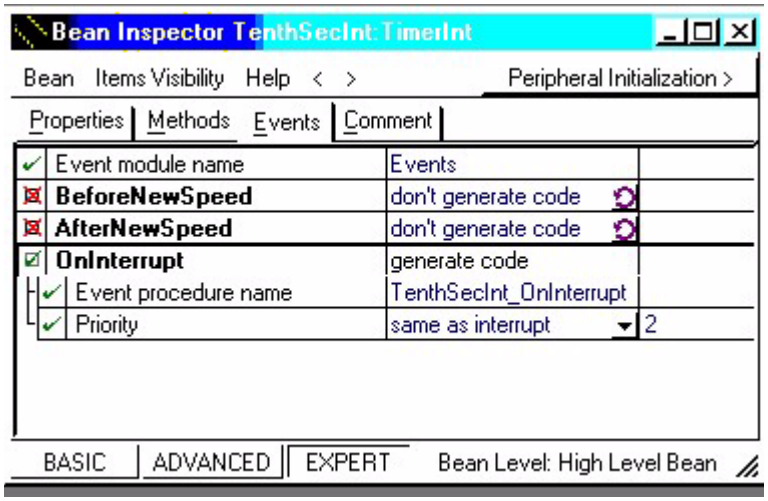


Figure 3-33. *TenthSecInt:TimerInt* Events

3.12 *TEL1:CallProgressToneDetection* Bean

This software bean generates the Call Progress Tone Detection code used to detect dial tone. Note the available methods, shown in **Figure 3-35**. When moused over in the project window, the method's documentation pops up.

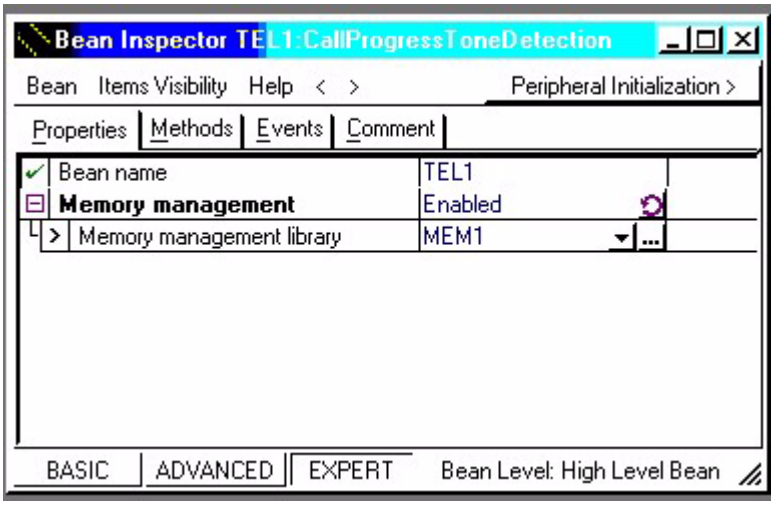


Figure 3-34. *TEL1:CallProgressToneDetection* Properties

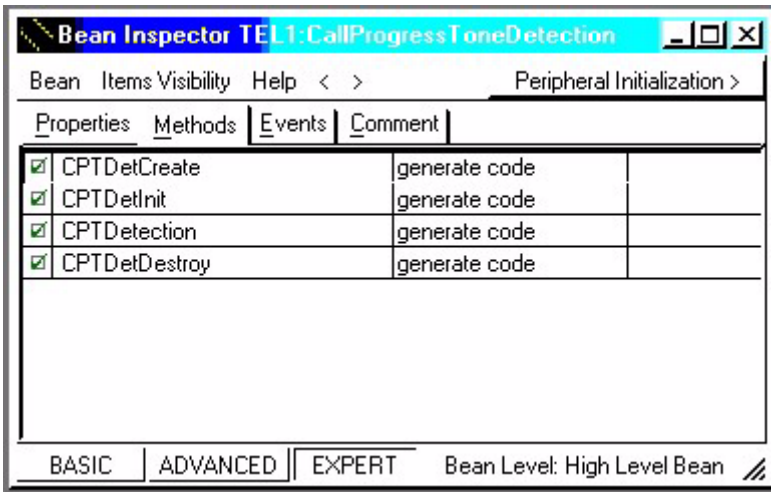


Figure 3-35. *TEL1:CallProgressToneDetection* Methods

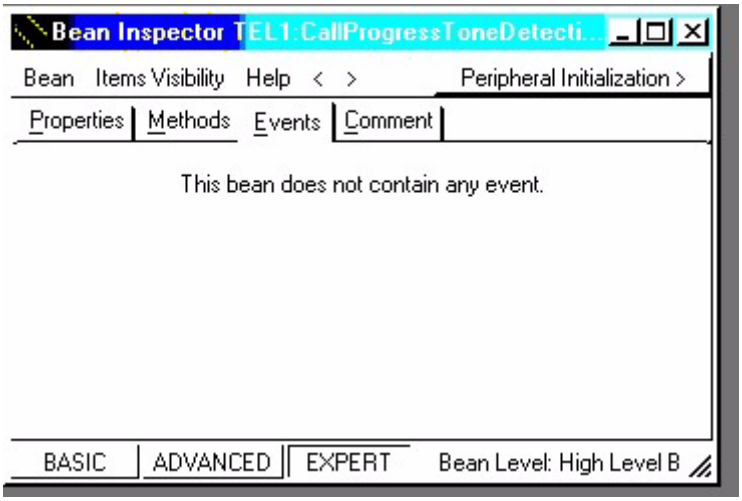


Figure 3-36. *TEL1:CallProgressToneDetection* Events

3.13 *TEL2:DTMFGenerate* Bean

This bean generates the software to generate the samples needed for DTMF dialing and supports both 7200 and 8000 samples per second. The soft modem uses 7200 samples per second, consuming less cycles and energy.

The resources of this bean are only used (RAM-allocated) while the bean is in use, due to the use of Memory Management.

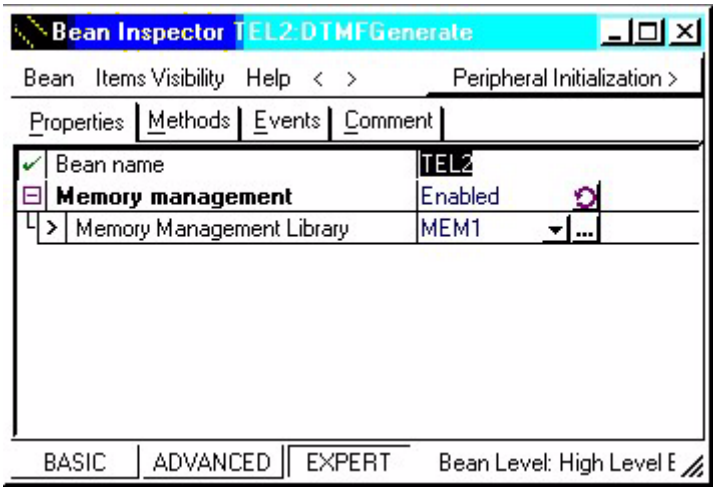


Figure 3-37. *TEL2:DTMFGenerate* Properties

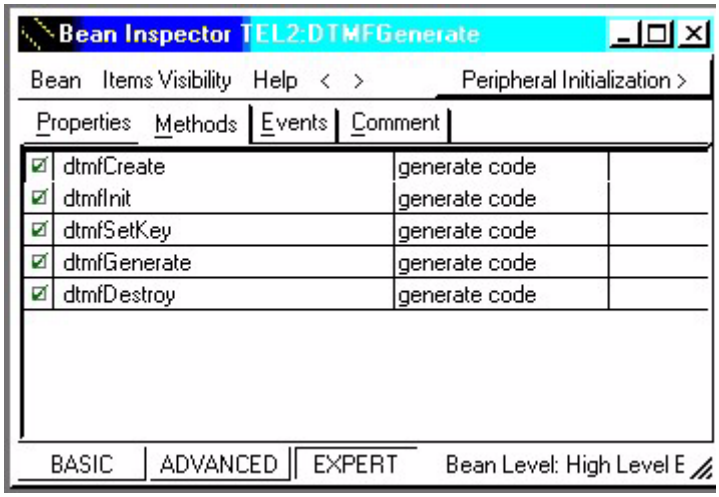


Figure 3-38. *TEL2:DTMFGenerate* Methods

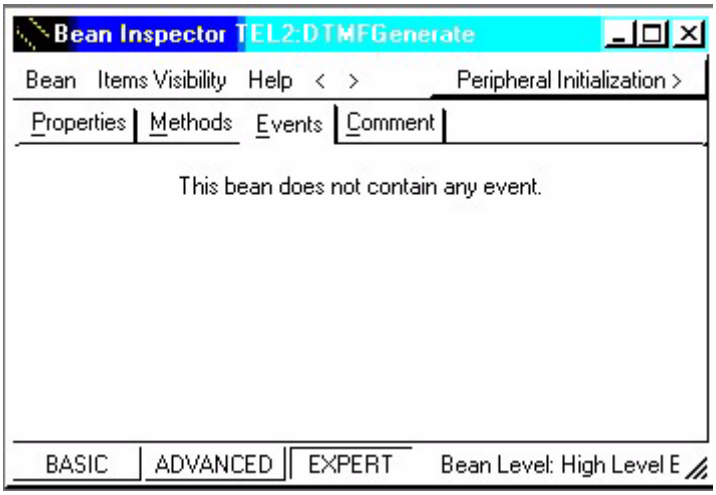


Figure 3-39. *TEL2:DTMFGenerate* Events

3.14 *Mdm2 DSP_v21* Modem Software Bean

One of the two modem beans dropped into this project, *Mdm2* generates the V.21 modem data pump code. This code does not do anything with answer tone or asynchronous characters. In the case of V.21, ANS processing for receive and transmit are both supplied in the applicaton code.

If ANS tone were not needed, V.21 would connect very quickly, allowing very rapid delivery of a real-time message. V.21 does not require training. As soon as the bean is called, it presents carrier and starts exchanging data.

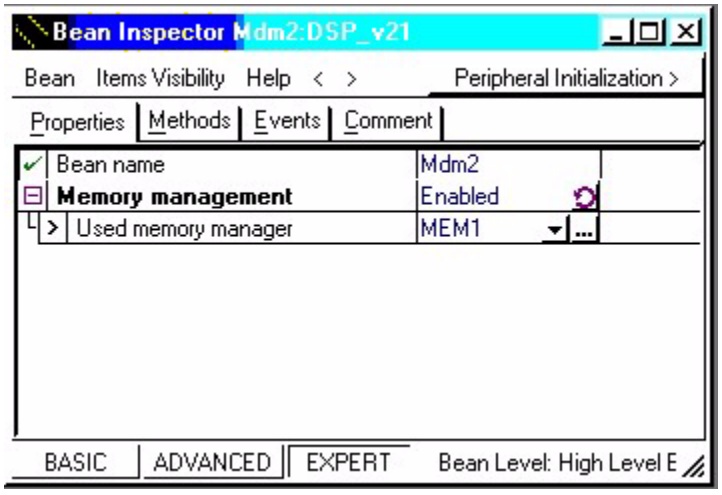


Figure 3-40. *Mdm2:DSP_v21* Properties

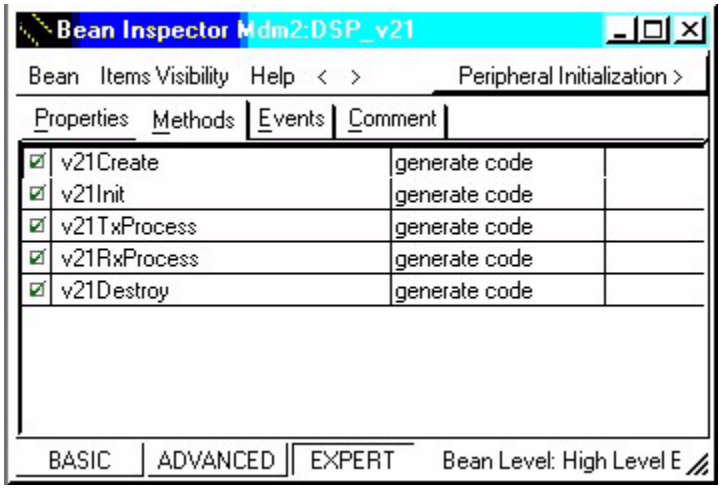
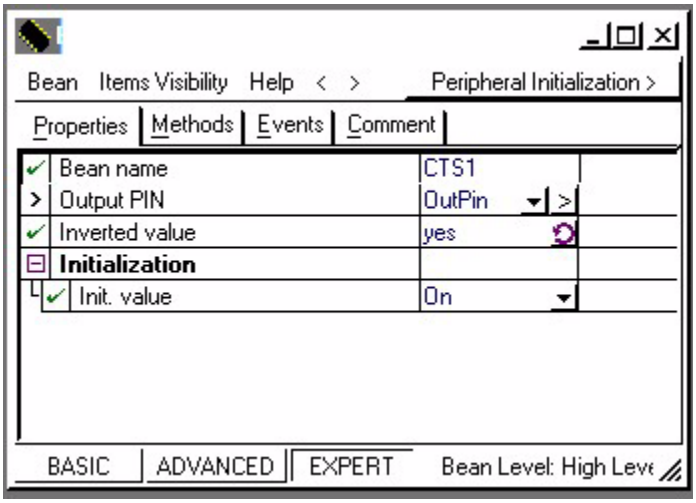


Figure 3-41. *Mdm2:DSP_v21* Methods

3.15 *CTS1:56F8357EVM_LED_Yellow2* Bean

This Clear-to-Send (of hardware flow control application) bean takes a bit IO that drives an LED on the EVM and employs it for the Clear-to-Send signal to the PC serial interface. Without this hardware flow control mechanism, the PC tends to overrun the modem during binary file transfers using XMODEM, or YMODEM, or ZMODEM binary file transfer protocols with HyperTerminal.



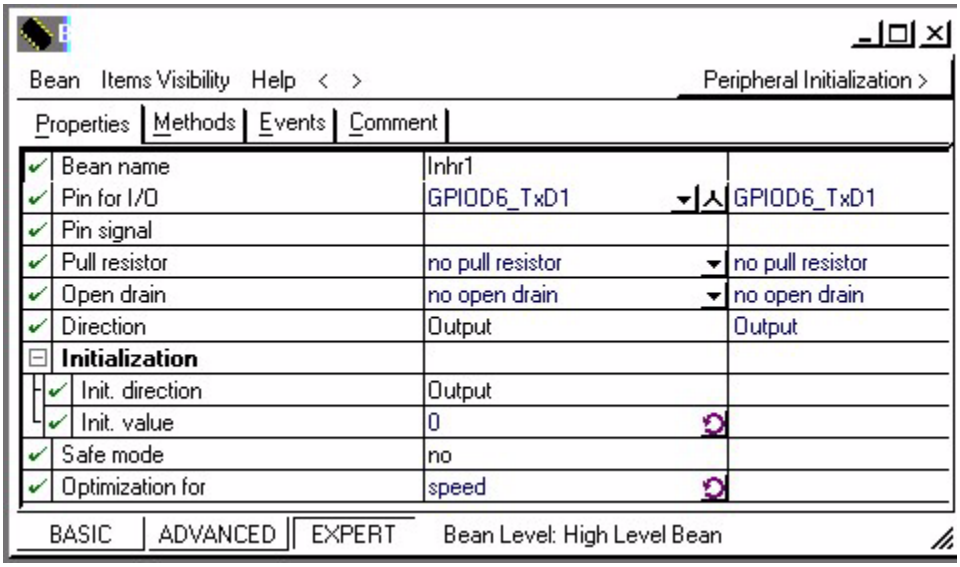


Figure 3-42. *CTS1:56F8357EVM_LED_Yellow2* Properties

Note the right angle bracket on the Output PIN on the top element of **Figure 3-42**, which indicates that this bean encapulates another bean. By clicking on the right angle bracket, a second screen is presented, shown in the lower element of **Figure 3-42** and displays the actual pin used for the Clear-to-Send signal sent to the PC serial port. This bean also toggles an LED so the user can see when flow control is actually being relied on.

This *On* initial value really means that flow is restricted initially.

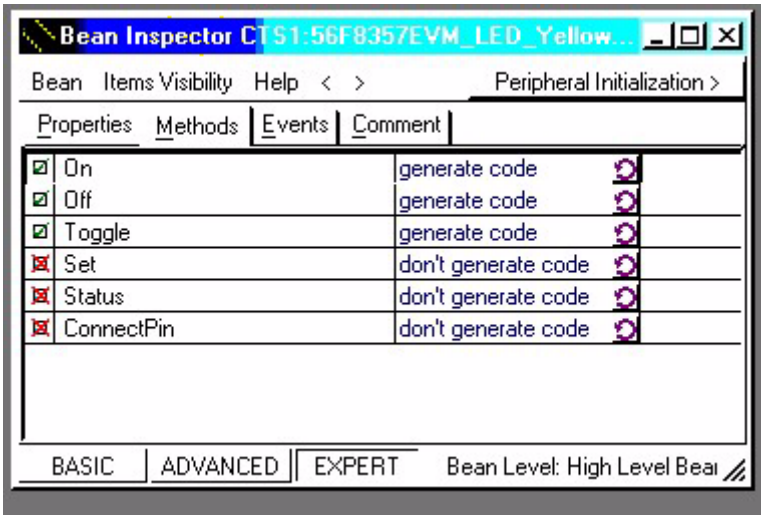


Figure 3-43. *CTS1:56F8357EVM_LED_Yellow2* Methods

The *On* method actually stops the flow from the PC, whereas the *Off* method allows it to resume. This is due to the simple jumper wire connection on the EVM to install the signal. The jumper wire conveys the drive to the RTS pin on the EVM on JG11.

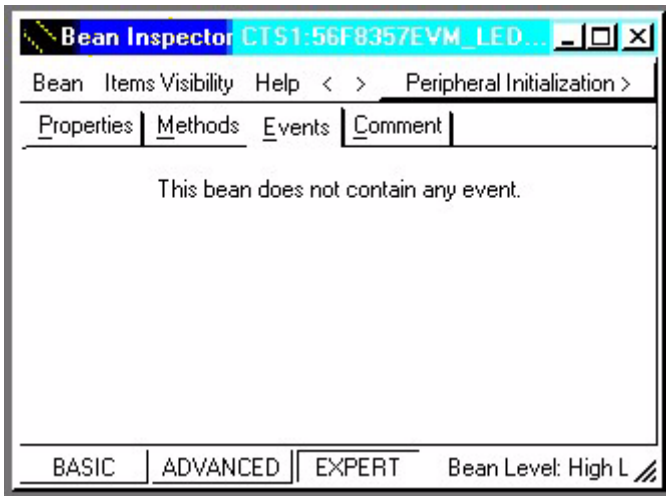


Figure 3-44. *CTS1:56F8357EVM_LED_Yellow2* Events

Chapter 4
Conclusion

The modem is fully capable of error-free operation over average telephone lines in the USA. Limitations to performance are inherent in the limited dynamic range of the AGC. If desired, such limitation may be overcome by the addition of a hardware AGC circuit, since the amplitude variation of a land line telephone circuit over time is not dramatic.

Modem performance curves indicate that the modem meets the performance standards required for V.22bis and V.21 operation.

Transfers of binary files millions of bits long were performed without incident.

4.1 Test Set-up

4.1.1 Routine File Transfer Testing

To set up routine file transfer, the TAS Series II was configured with the script in Example 4-1, which depicts the average USA line.

Code Example 4-1. Routine File Transfer Testing

```
/ad,s03=1,s07=1,c3/
/exch,bal=1/
/file:cseq=usa1/
/io,i-100,l-230,r-100,t-230/
/ad,i3/
/gd,w17,x00,y16,z00/
/rn,l320,s1/
/nl,q520,c500,m0,x1,y1/
/fs,f+1250,m0,s1/
/pj,l0364,f1200,w0,s1/
```

The Hayes modem was simply configured with factory defaults:

```
ATZ
```

For 300 baud testing, the Hayes modem was limited to 300 baud with the following command:

```
ATS37=3
```

which configured the modem for V.21. The Freescale Low-Cost Soft Modem, the unit under test (UUT), was configured for V.21 with the AT+ command.

4.1.2 Bit Error Rate Testing

All Bit Error Rate Test (BERT) runs were performed at 2400 bits per second, V.22bis mode.

For bit error rate tests on the several standard line configurations, white noise was used for the impairment. The contribution of the various lines to the tests are their phase maginitude responses. These results are depicted in **Figures 4-1** through **4-11**.

For the dynamic range test, a null line was used. The null line has a flat phase magnitude response. The purpose of this test is to show the lowest input level to the modem at which it will function on a flat line.

The Hayes equipment was on the A equipment side, the UUT on the B equipment side.

Since the V.22bis contains a built in scrambler, modem BERT perfomance was measured as a function of the number of error characters that were received when no characters were being transmitted. A test frame of 7 minutes, or about 1 million bits, was used for each test. One synchronous bit error results in at least one asychronous character error because it is seen as a start bit. The idle line condition is marking, or 1.

The following parameters were used to obtain SNR BERT raw data, available on request, from which these figures were graphed:

- IR -- the level at the A equipment after passing from the modem transmitter through the Line to the TAS Series II, in dBm
- LR -- the signal level transmitted from the TAS Series 2 into the Line and hence to the modem RX on the A side
- RR -- the level at the B equipment after passing from the modem TX thru the Line t to the TAS Series II
- TR -- the signal level TX from the TAS Series II central office into the Line and hence to the Modem RX on the B side
- RL -- the noise level in dBrn (to convert to dBm, subtract 90); noise was white over 5kHz C-Message
- Dir -- the signal direction affected by noise, which is added as it leaves the TAS Series II into the line.
- I, L, R, and T -- initialized so that the AGC does not have to travel too far during AGC operations with the TAS Series II
- IR, LR, RR, TR -- measured after the AGC; where reported, complete AGCs have been performed

The AGC feature of the TAS Series II was used to align signal levels in the digital portion of the TAS Series II.

In each of the tests shown in **Figures 4-1** through **4-11**, the units used were a Hayes Accura V.92 modem and the UUT, which was the 56F8357EVM with the LCMDC.

4.2 Bit Error Rate Test Results

In each of the lines tested, the line impairment was present in both lines from the central office simulation and to the central office simulation. This resulted in twice the dynamic range requirement imposed by using only one line impairment for one of the legs of the call. As a result, the IEA5 test shows the result of hitting the dynamic range limitation due to the cascaded shaping (magnitude response vs. frequency) of *two* IEA5 lines in **Figure 4-5**.

In the following figures, the four lines are abbreviated in the legend and obtained as follows:

- **HAB --** Hayes modem calling UUT with noise in the A to B transmission path
- **HBA** -- Hayes modem calling UUT with noise in the B to A transmission path
- **AB** -- UUT calling Hayes modem with noise in the A to B transmission path
- **BA** -- UUT calling Hayes modem with noise in the B to A transmission path

In each test, from **Figure 4-1** through **Figure 4-11**, the line specified, such as EIA1, was used for both directions of tranmission.

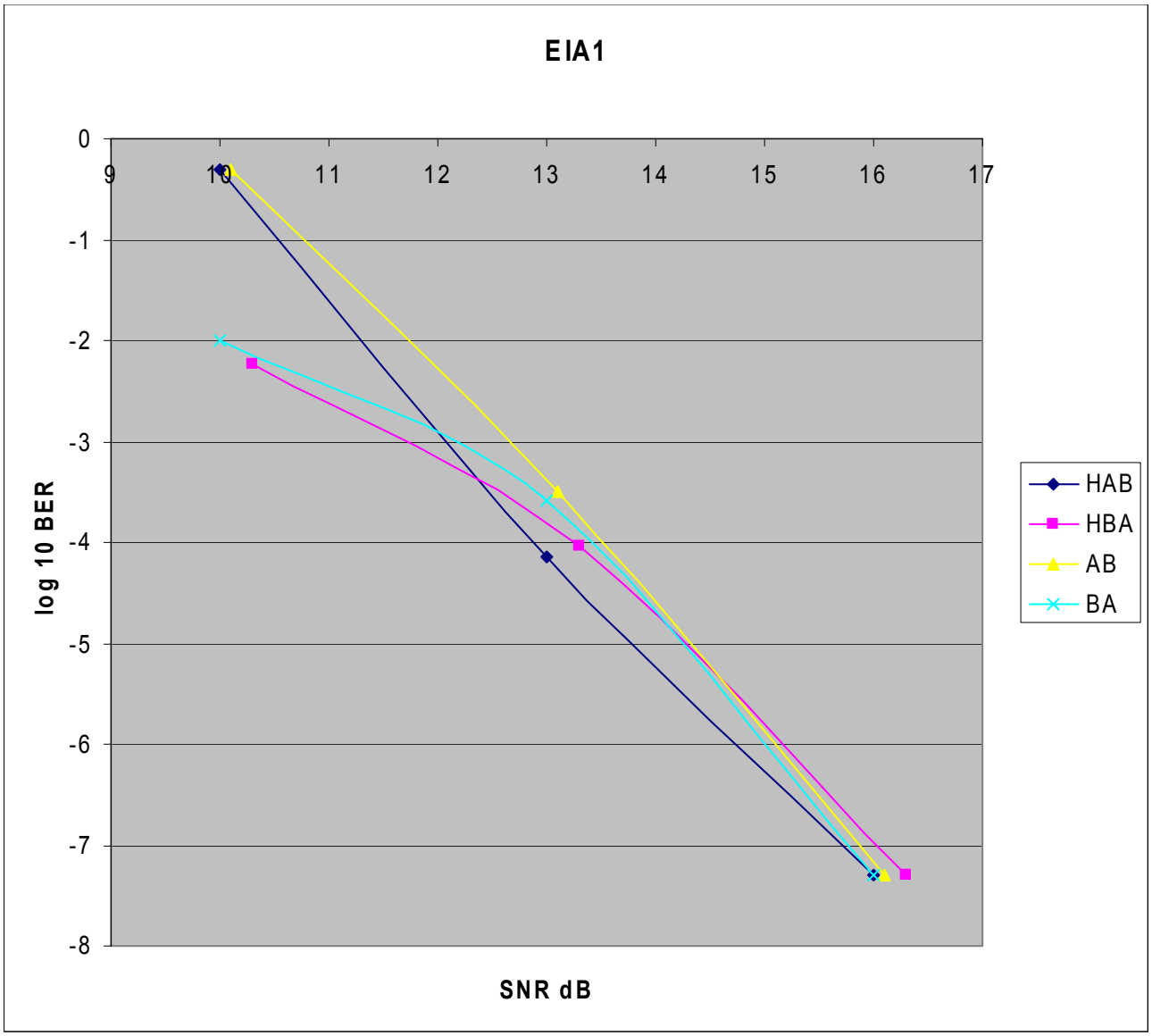


Figure 4-1. EIA1 Line

Figure 4-2 demonstrates that the modem is easily used on EIA2 lines down to a SNR of 11.2. The Hayes modem tested against has nearly 100dB of dynamic range, compared to the ENOB of 8 or 9dB present in the ADC of the soft modem reference design.

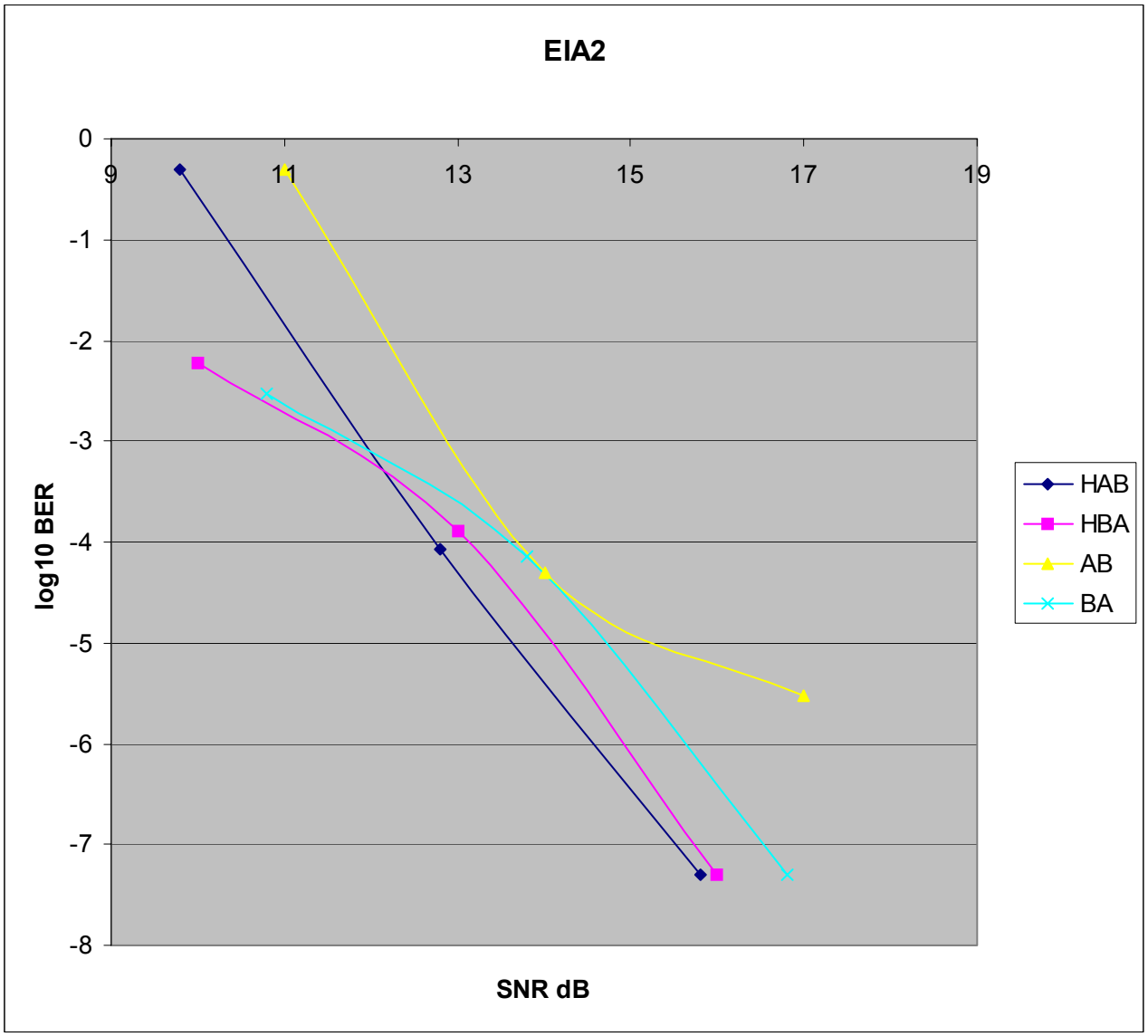


Figure 4-2. EIA2 Line

The pair of EIA3 lines is usable down to 12dB SNR.

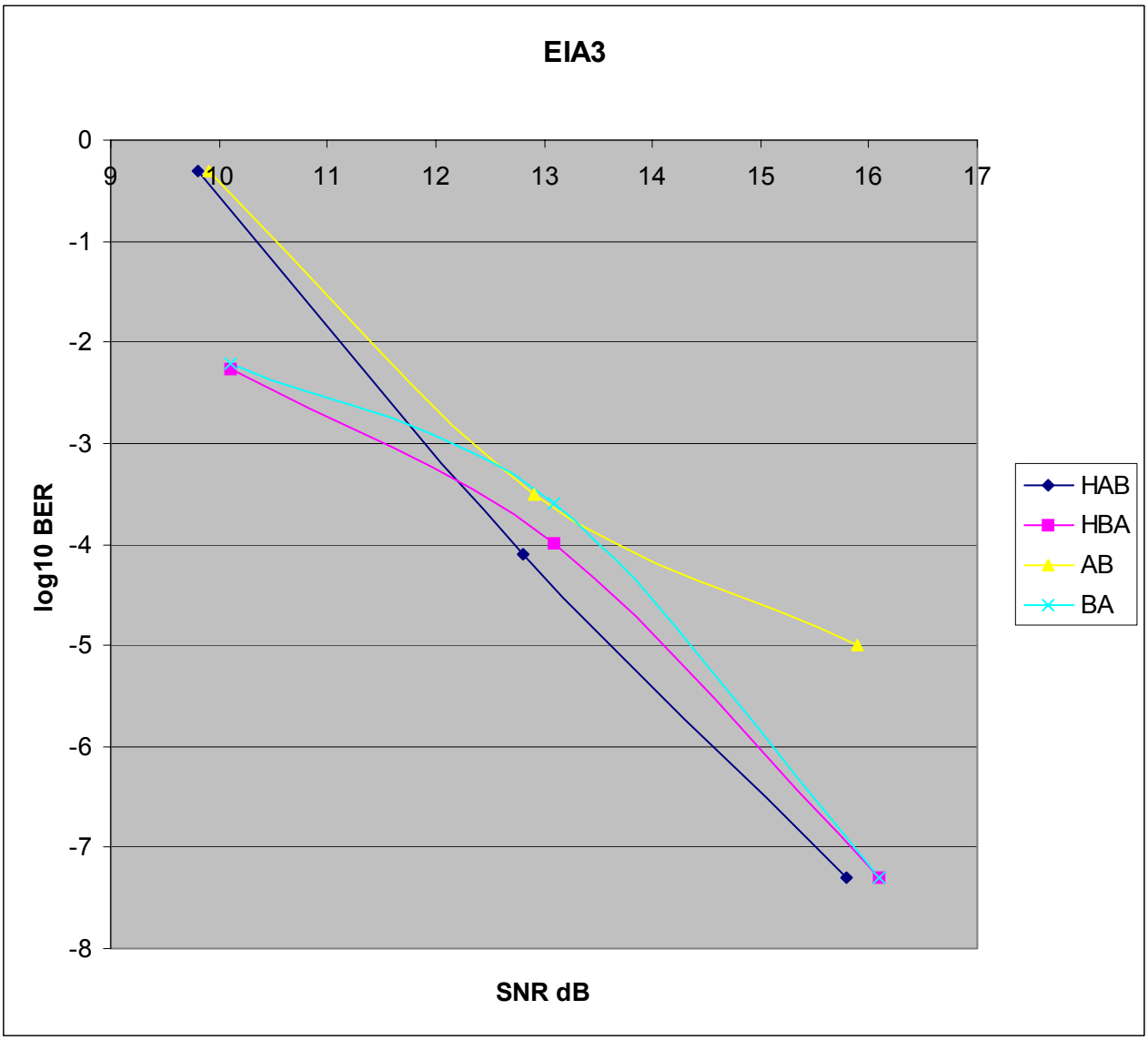


Figure 4-3. EIA3 Line

EIA4 is usable down to 12dB SNR.

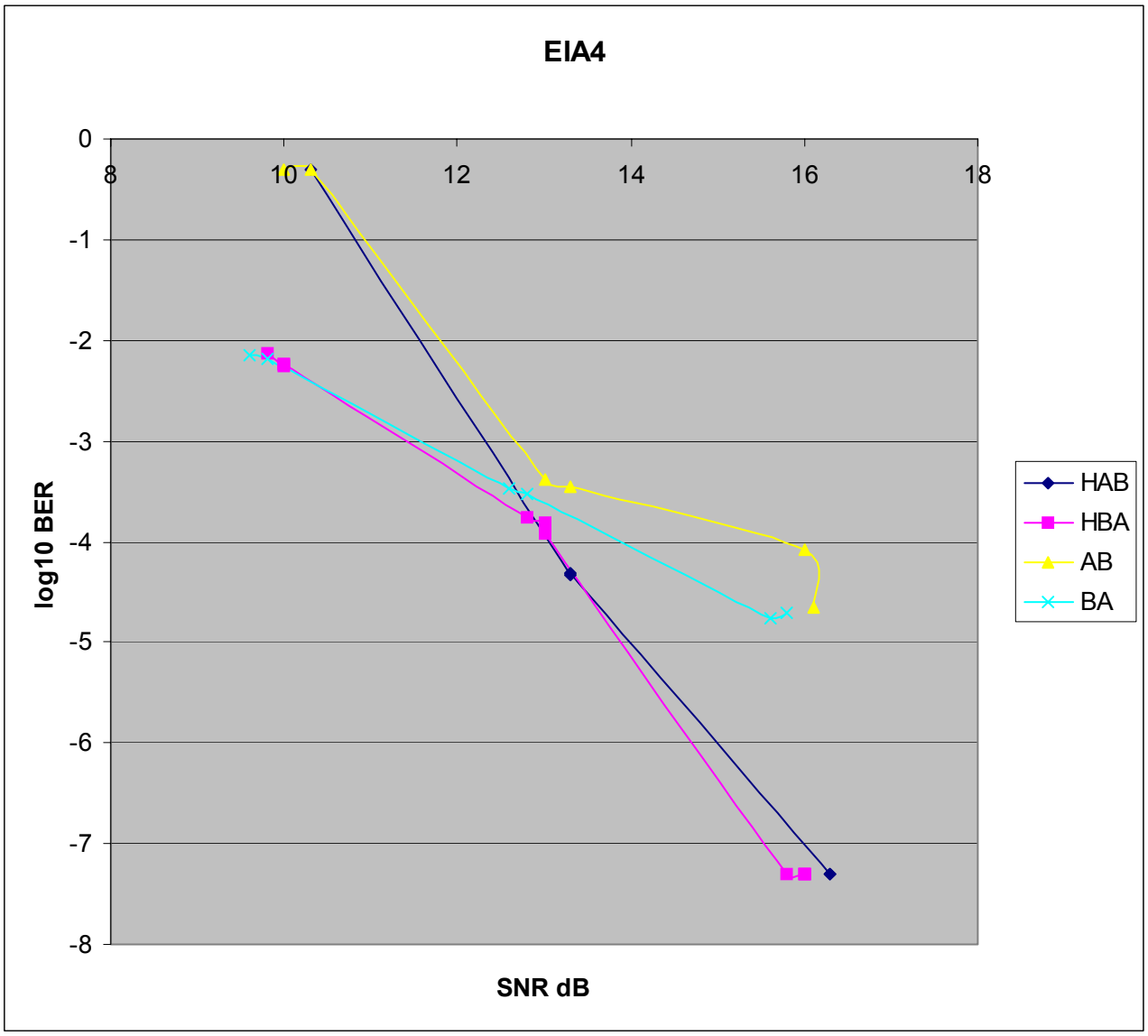


Figure 4-4. EIA4 Line

In the worst case for this modem, the EIA5 line is usable down to 13dB SNR. After that, the spectral shaping of the pair of EIA5 lines used in the test exceeds the dynamic range of the unit under test, so the signal cannot be reconstructed below 13dB SNR.

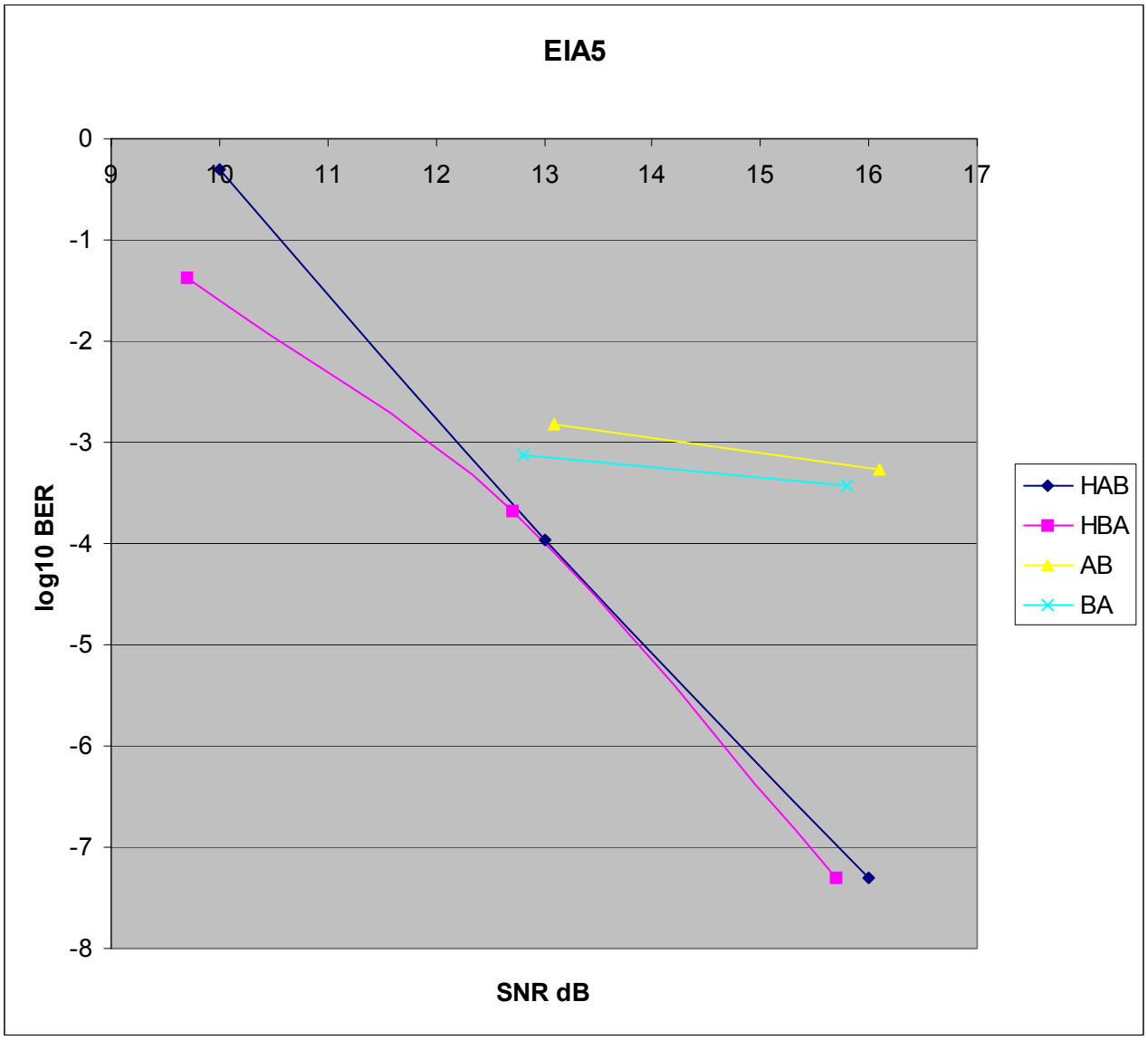


Figure 4-5. EIA5 Line

The EIA6 line is usable down to 12dB SNR.

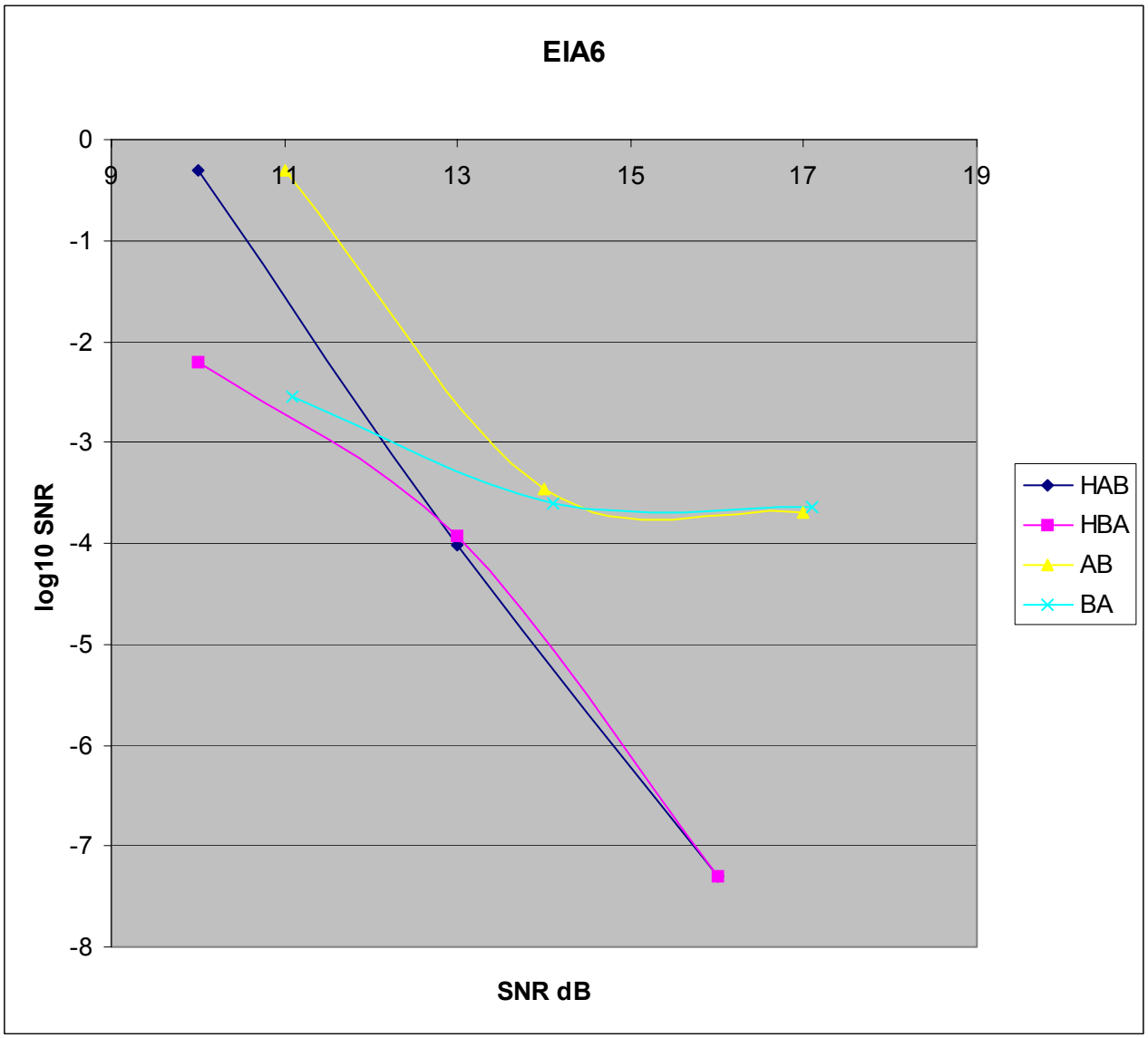


Figure 4-6. EIA6 Line

The EIA7 line is usable to slightly less than 13dB SNR.

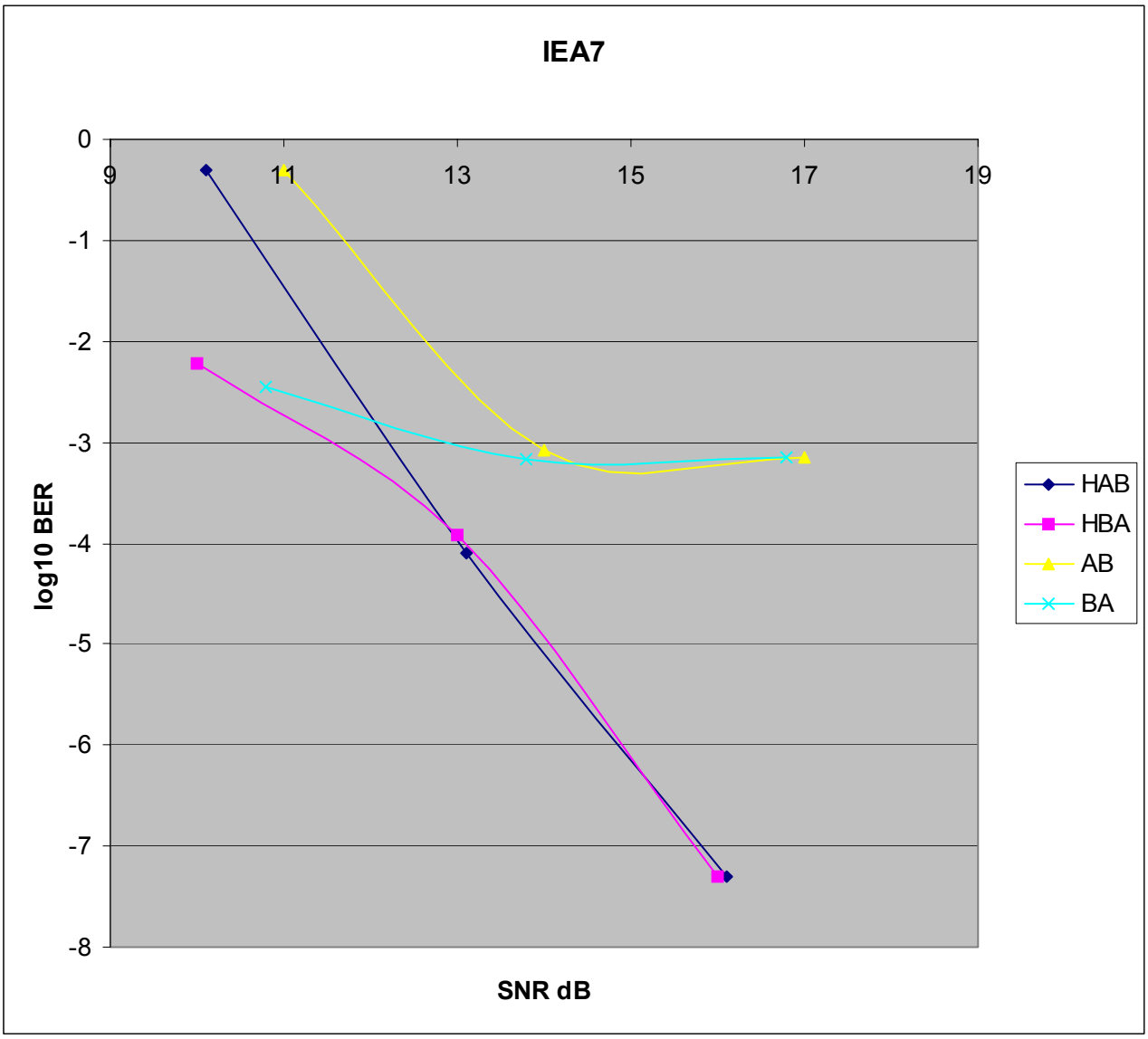


Figure 4-7. EIA7 Line

The ETSI1 line is good down to 11.5dB SNR.

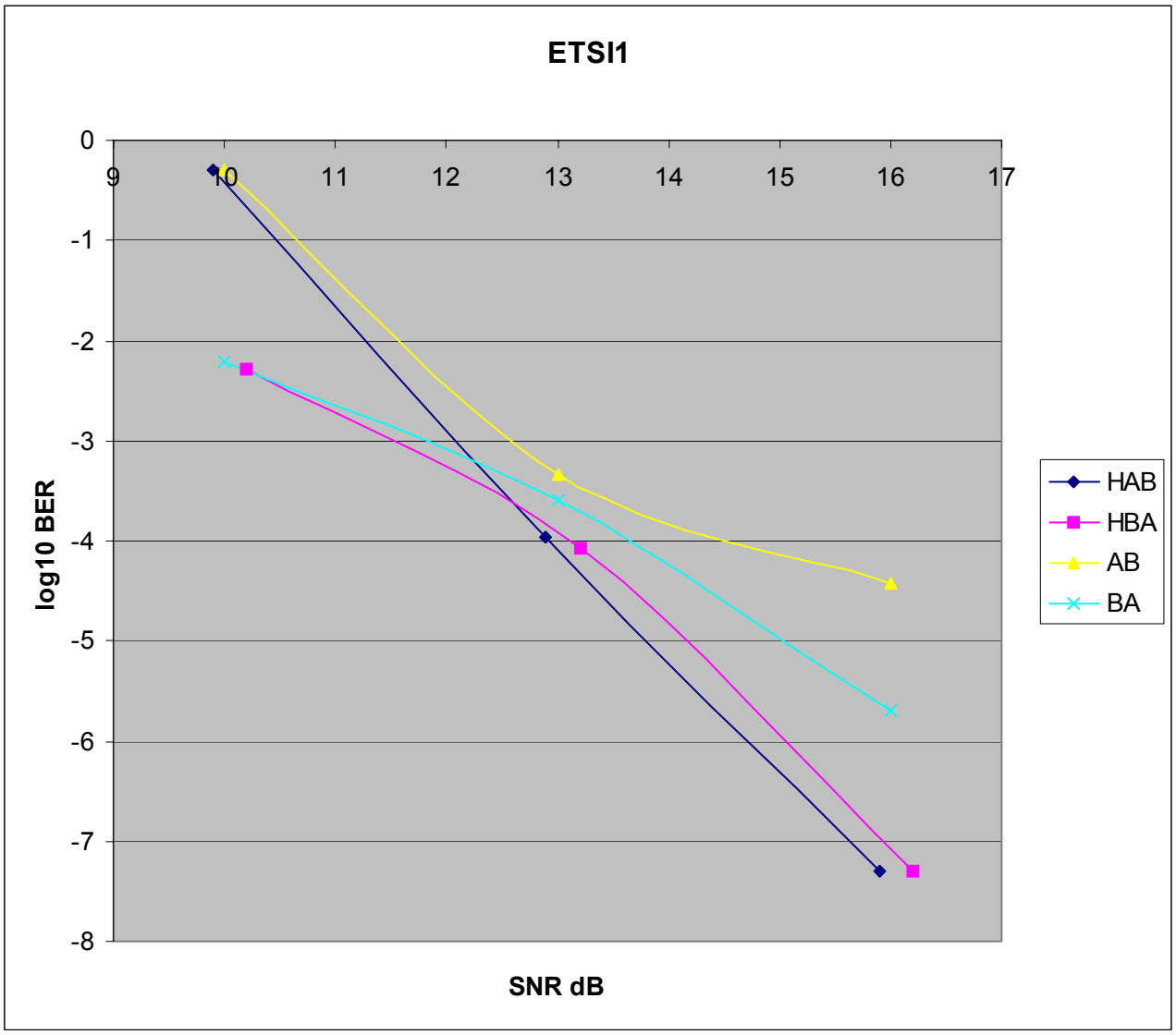


Figure 4-8. ETSI1 Line

The ETSI2 line is good down to 11.4dB SNR.

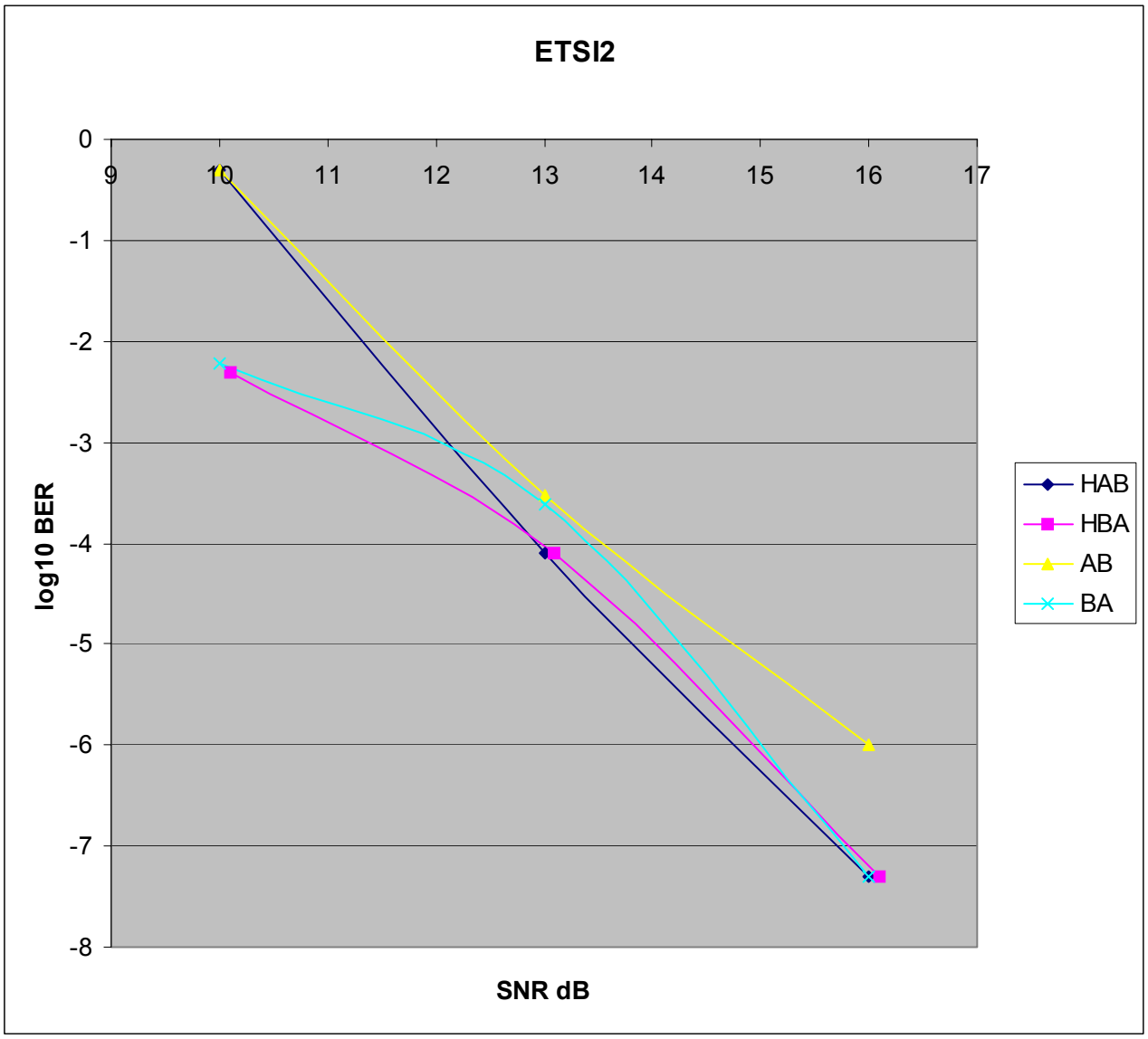


Figure 4-9. ETSI2 Line

The NULL line was good down to 12dB SNR. The region of interest is where the log10 BER is -3.

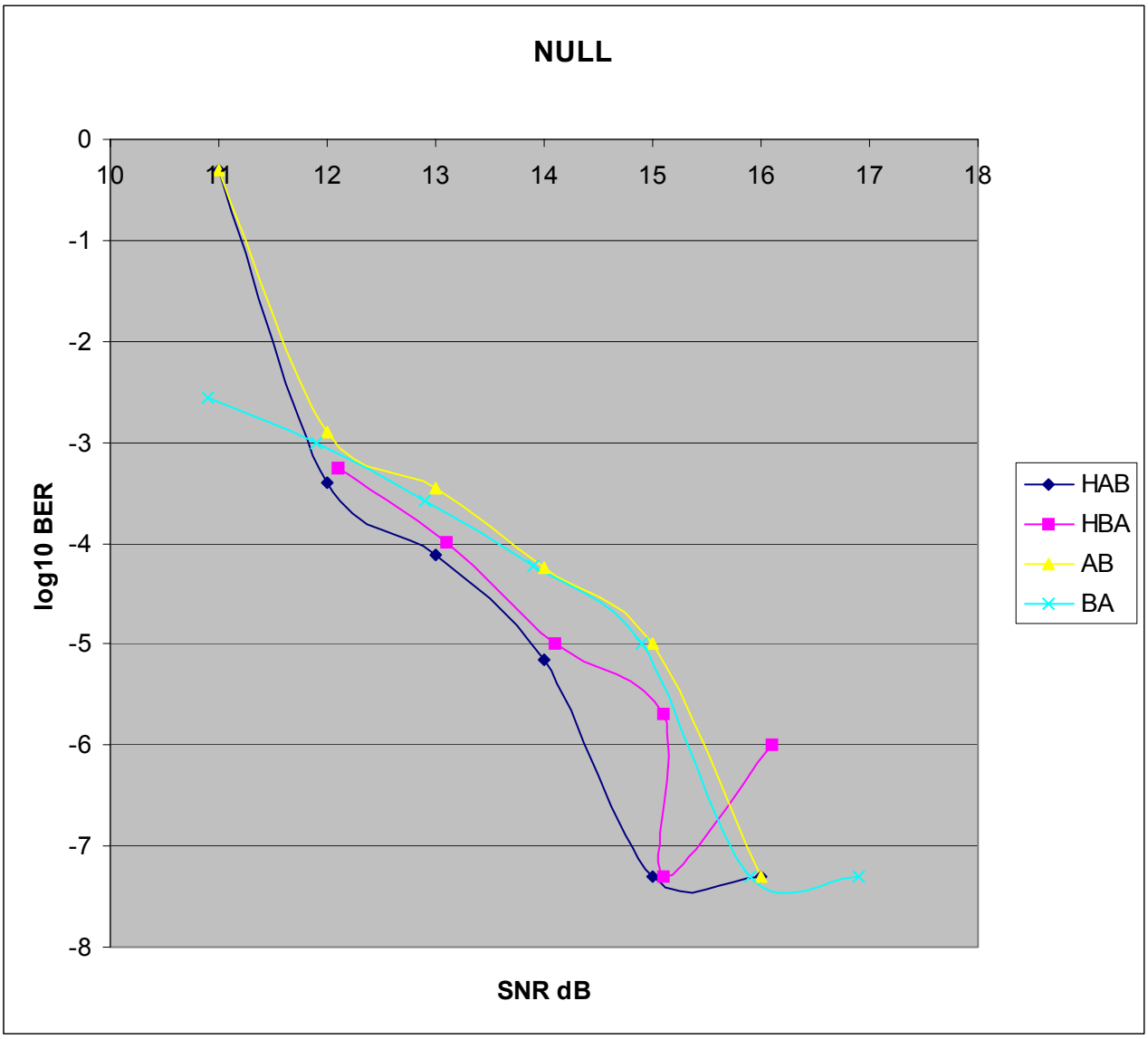


Figure 4-10. Null Line

The attentuation pad (no spectral shaping) was good down to 11.5dB SNR.

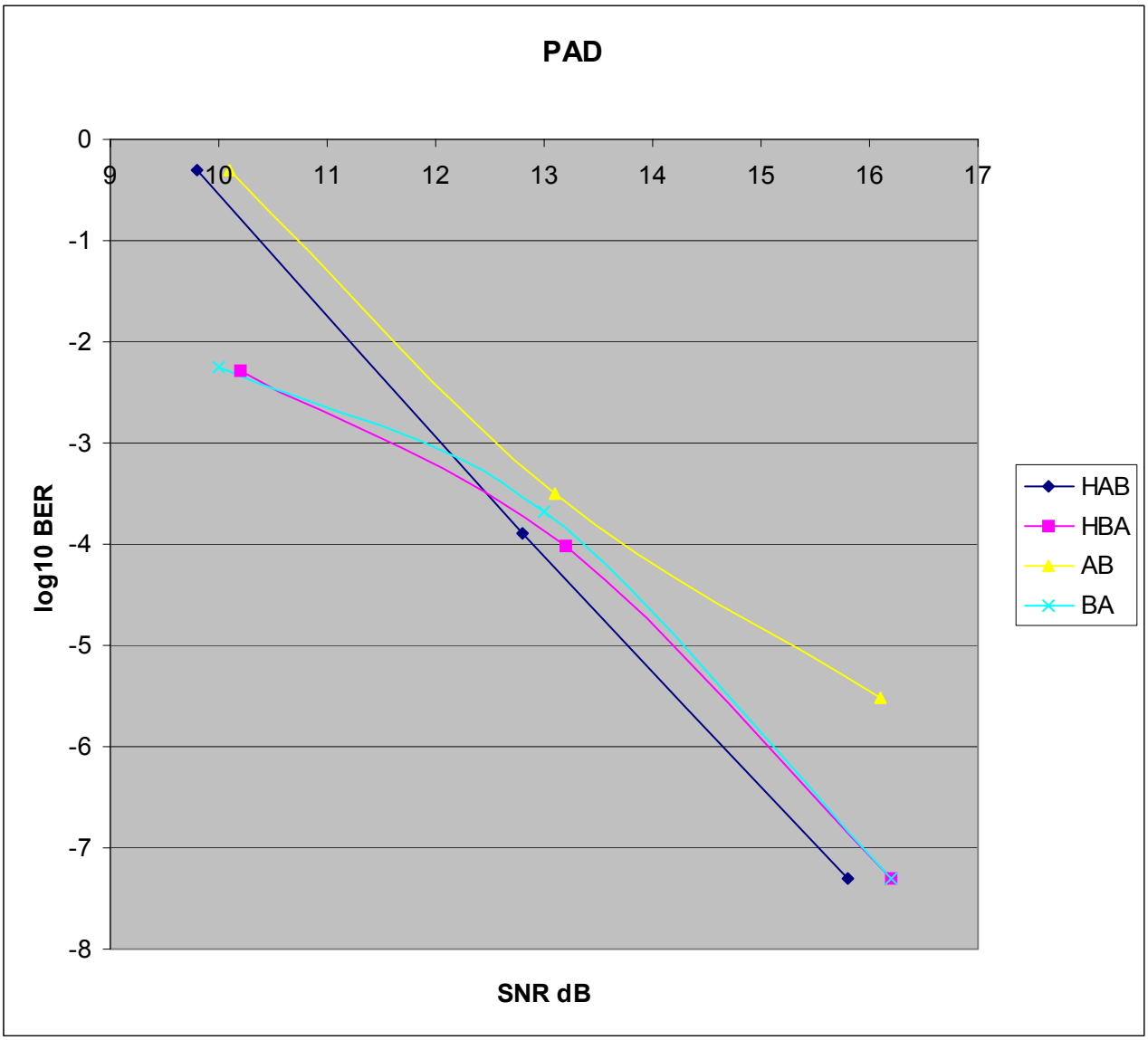


Figure 4-11. EIA2 Line

4.3 Dynamic Range Test Results

Figure 4-12 displays the soft modem's good dynamic range response down to -44dBm.

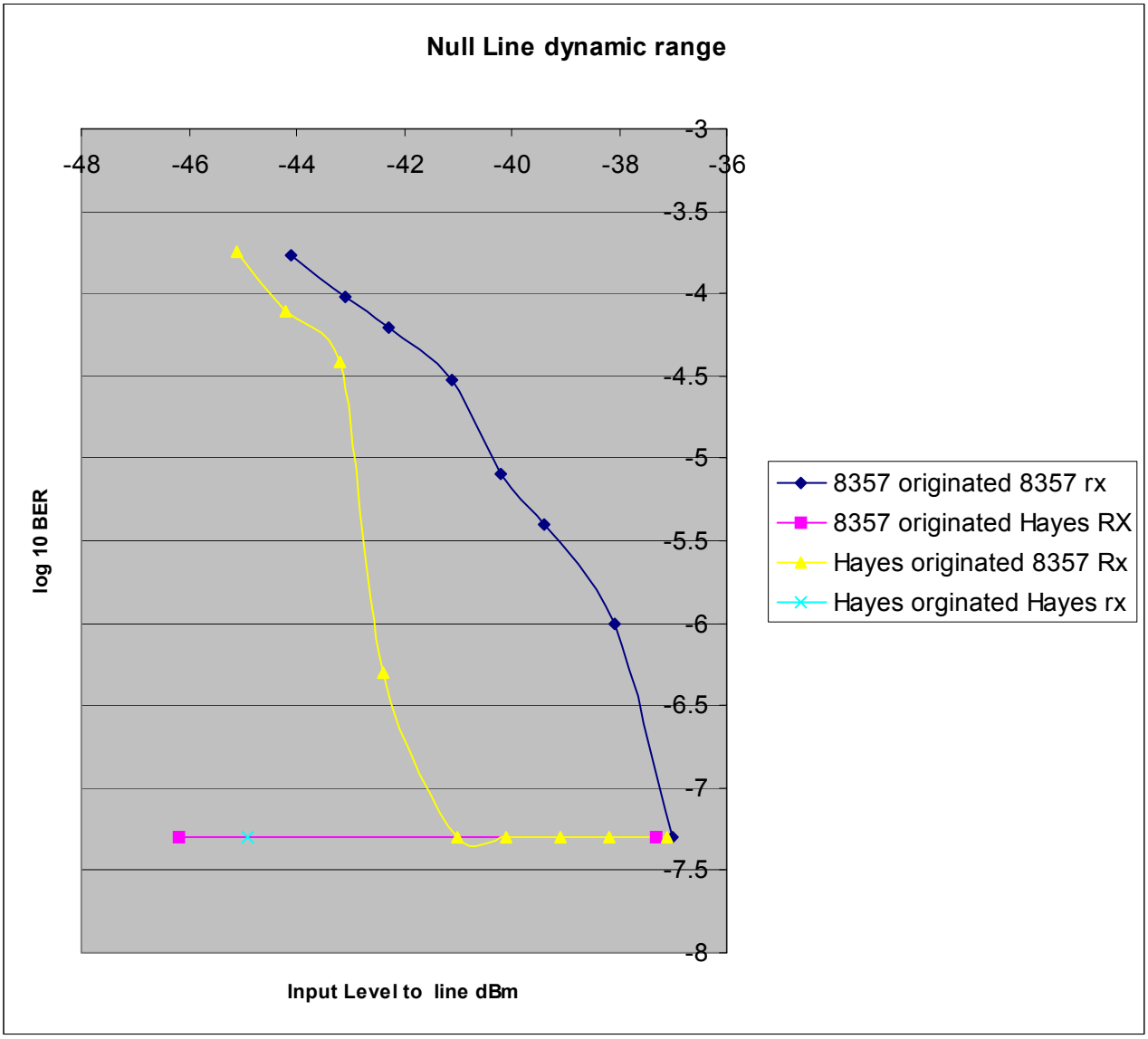


Figure 4-12. Null Line Dynamic Range

4.4 Memory Utilization on the 56F8357

4.4.1 Summary Memory Utilization

```
# Memory map:
  v_addr   p_addr   size     name
  0000F020 0000F020 00000000 .x_Peripherals
  00020000 00020000 00000004 .p_Interruptsboot
  00000000 00000000 000000A4 .p_Interrupts
  000000A4 000000A4 00003CDA .p_Code
  00000000 00000000 0000178C .x_Data
  00001C00 00001C00 00000400 .x_DynMem
  00003D7E 00000000 00000B70 .p_flash_ROM_data
  00002000 00002000 00000070 .x_internal_ROM
  0002F800 0002F800 00000800 .p_internal_RAM
```

4.4.2 Complete Load Map

```
# Link map of F_EntryPoint



# .interrupt_vectorsboot
#>00020000          F_vector_addr (linker command file)
  00020000 00000004 interrupt_vectorsboot.text F_vectboot(Vectors.c)

# .interrupt_vectors
  00000000 000000A4 interrupt_vectors.text F_vect(Vectors.c)

# .Data_xROM
  00002000 00000008 .const.data F@904(modem.c)
  00002008 00000007 .const.data F@905(modem.c)
  0000200F 00000004 .const.data F@906(modem.c)
  00002012 00000008 .const.data F@907(modem.c)
  0000201A 00000008 .const.data F@908(modem.c)
  00002022 00000003 .const.data F@1050(modem.c)
  00002025 00000004 .const.data F@1051(modem.c)
  00002028 0000002B .const.data F@1052(modem.c)
  00002053 00000006 .const.data F@1117(modem.c)
  00002059 00000007 .const.data F@1193(modem.c)
  0000205F 00000009 .const.data F@1194(modem.c)
  00002068 00000009 .const.data F@1195(modem.c)

# .ApplicationCode
#>000000A4          F_Pcode_start_addr (linker command file)
  000000A4 00000030 .text   F_EntryPoint(Cpu.c)
  000000D4 00000007 .text   FCpu_Interrupt(Cpu.c)
  000000DB 00000011 .text   FCpu_PLLInterrupt(Cpu.c)
  000000EC 000000A7 .text   FPE_low_level_init(Cpu.c)
  00000193 00000020 .text   FMEM1_Init(MEM1.c)
  000001B3 0000001D .text   FAD1_InterruptCC(AD1.c)
  000001D0 00000056 .text   FHWEnDi(AD1.c)
  00000226 0000000D .text   FAD1_Enable(AD1.c)
  00000233 0000000C .text   FAD1_Disable(AD1.c)
  0000023F 0000001A .text   FAD1_EnableIntTrigger(AD1.c)
  00000259 00000041 .text   FAD1_Init(AD1.c)
  0000029A 0000002E .text   FOutputTimer_Init(OutputTimer.c)
  000002C8 0000002E .text   FInputTimer_Init(InputTimer.c)
  000002F6 0000000A .text   FHWEnDi(AS1.c)
  00000300 0000004E .text   FAS1_RecvChar(AS1.c)
  0000034E 00000046 .text   FAS1_SendChar(AS1.c)
  00000394 0000001F .text   FAS1_RecvBlock(AS1.c)
  000003B3 00000020 .text   FAS1_SendBlock(AS1.c)
  000003D3 00000003 .text   FAS1_GetCharsInRxBuf(AS1.c)
```

```
000003D6 00000086 .text   FAS1_GetError(AS1.c)
0000045C 00000057 .text   FAS1_InterruptRx(AS1.c)
000004B3 00000054 .text   FAS1_InterruptTx(AS1.c)
00000507 00000039 .text   FAS1_InterruptError(AS1.c)
00000540 0000001F .text   FAS1_Init(AS1.c)
0000055F 0000000F .text   FRingDetect_Enable(RingDetect.c)
0000056E 0000000D .text   FRingDetect_Disable(RingDetect.c)
0000057B 00000006 .text   FRingDetect_ResetCounter(RingDetect.c)
00000581 0000000A .text   FRingDetect_GetCounterValue(RingDetect.c)
0000058B 00000015 .text   FRingDetect_Init(RingDetect.c)
000005A0 00000044 .text   FBitIO__PutVal(OffHook.c)
000005E4 0000002C .text   FSetCV(TenthSecInt.c)
00000610 00000018 .text   FSetPV(TenthSecInt.c)
00000628 00000032 .text   FHWEnDi(TenthSecInt.c)
0000065A 00000030 .text   FTenthSecInt_Init(TenthSecInt.c)
0000068A 00000011 .text   FTenthSecInt_Interrupt(TenthSecInt.c)
0000069B 00000015 .text   Fv22bisCreate(v22bisapi.c)
000006B0 0000002C .text   Fv22bisInit(v22bisapi.c)
000006DC 00000012 .text   Fv22bisTXDataInit(v22bisapi.c)
000006EE 000000AB .text   Fv22bisTX(v22bisapi.c)
00000799 00000014 .text   Fv22bisTransmit(v22bisapi.c)
000007AD 0000008D .text   Fv22bisRX(v22bisapi.c)
0000083A 0000000D .text   Fv22bisDestroy(v22bisapi.c)
00000847 0000007F .text   FTransmitV14(v22bisapi.c)
000008C6 000000B9 .text   FReceiveV14(v22bisapi.c)
0000097F 00000014 .text   FmemMallocEM(mem.c)
00000993 0000001C .text   FmemMallocAlignedEM(mem.c)
000009AF 00000014 .text   FmemFreeEM(mem.c)
000009C3 0000001B .text   FmemIsAligned(mem.c)
000009DE 0000000B .text   FmemProtect(mem.c)
000009E9 0000005F .text   FMergeFree(mem.c)
00000A48 00000064 .text   FSplitBlock(mem.c)
00000AAC 00000055 .text   FSplitBlockRev(mem.c)
00000B01 0000005E .text   FmemInitializePool(mem.c)
00000B5F 000000D3 .text   FmemExtendPool(mem.c)
00000C32 00000038 .text   FmemFree(mem.c)
00000C6A 000000B3 .text   FmemMalloc(mem.c)
00000D1D 000000E3 .text   FmemMallocAligned(mem.c)
00000E00 0000009A .text   FInitialize(mem.c)
00000E9A 0000000C .text   FmemInitialize(mem.c)
00000EA6 0000000E .text   FmemIsIM(memtarget.c)
00000EB4 0000000E .text   FmemIsEM(memtarget.c)
00000EC2 00000041 .text   FAD1_OnEnd(Events.c)
00000F03 00000037 .text   FTenthSecInt_OnInterrupt(Events.c)
00000F3A 0000000F .text   FAS1_OnError(Events.c)
00000F49 00000001 .text   FAS1_OnRxChar(Events.c)
00000F4A 00000001 .text   FAS1_OnTxChar(Events.c)
```

```
00000F4B 00000001 .text   FAS1_OnFullRxBuf(Events.c)
00000F4C 00000001 .text   FAS1_OnFreeTxBuf(Events.c)
00000F4D 00000006 .text   FCpu_OnSwINT0(Events.c)
00000F53 00000006 .text   FCpu_OnSwINT1(Events.c)
00000F59 00000006 .text   FCpu_OnSwINT2(Events.c)
00000F5F 00000006 .text   FCpu_OnReset(Events.c)
00000F65 00000006 .text   FCpu_OnSwINT3(Events.c)
00000F6B 00000006 .text   FCpu_OnSwINTLP(Events.c)
00000F71 00000007 .text   FCpu_OnPLLError(Events.c)
00000F78 00000007 .text   FCpu_OnIllegalOpcode(Events.c)
00000F7F 00000007 .text   FCpu_OnHWStackError(Events.c)
00000F86 0000005F .text   FTEL1_CPTDetCreate(TEL1.c)
00000FE5 00000014 .text   FTEL1_CPTDetInit(TEL1.c)
00000FF9 0000000F .text   FTEL1_CPTDetection(TEL1.c)
00001008 0000002E .text   FTEL1_CPTDetDestroy(TEL1.c)
00001036 00000090 .text   Fv21Destroy(v21_destroy.c)
000010C6 000000F8 .text   Fv21Init(v21_init.c)
000011BE 00000078 .text   FV21_Init_to_Zero(v21_init.c)
00001236 00000064 .text   Fv21Create(v21_create.c)
0000129A 00000589 .text   Fmain(modem.c)
00001823 000001AF .text   FAT_offline(modem.c)
000019D2 00000115 .text   FAT_online(modem.c)
00001AE7 0000000B .text   FCPTDetCallback(modem.c)
00001AF2 00000032 .text   FTxCallbackV21(modem.c)
00001B24 0000006B .text   FTXCallbackRoutine(modem.c)
00001B8F 00000055 .text   FRxCallbackV21(modem.c)
00001BE4 0000003C .text   FRXCallbackRoutine(modem.c)
00001C20 0000000F .text   FInitAnalogRxChannel(modem.c)
00001C2F 00000007 .text   FInitAnalogTxChannel(modem.c)
00001C36 00000005 .text   FInitPoorMansCodec(modem.c)
00001C3B 0000002C .text   FReadAnalogRxData(modem.c)
00001C67 00000071 .text   FWriteAnalogTxData(modem.c)
00001CD8 00000032 .text   FOutputTimerInts(modem.c)
00001D0A 00000003 .text   FMisalignedLongWordISR(modem.c)
00001D0D 000000B8 .text   Fv21TxProcessA(v21_processA.c)
00001DC5 00000133 .text   Fv21RxProcessA(v21_processA.c)
00001EF8 00000007 .text   Fv21TxSamplesAmplifyA(v21_processA.c)
00001EFF 00000052 .text   FgetDemodByteA(v21_processA.c)
00001F51 0000000D .text   Fmemset(MSL C 56800E.lib mem.o          )
00001F5E 000000BA .text   F__fill_mem(MSL C 56800E.lib mem_funcs.o    )
00002018 0000000C .text   F__init_sections(Runtime 56800E.Lib initsections.o )
00002024 0000000F rtlib.text F@DummyFn1(v22bis_tx_b_end.asm)
00002024 00000000 rtlib.text end_tx(v22bis_tx_b_end.asm)
00002024 0000000F rtlib.text rtlib.text(v22bis_tx_b_end.asm)
00002033 000000A5 rtlib.text rtlib.text(v22bis_tx_ctrl.asm)
00002033 000000A5 rtlib.text F@DummyFn1(v22bis_tx_ctrl.asm)
00002033 00000000 rtlib.text TXBAUD(v22bis_tx_ctrl.asm)
```

```
0000203A 00000000 rtlib.text Chk_St_Chg(v22bis_tx_ctrl.asm)
00002063 00000000 rtlib.text next_task(v22bis_tx_ctrl.asm)
00002069 00000000 rtlib.text RETRAIN(v22bis_tx_ctrl.asm)
000020D2 00000000 rtlib.text perf_sti_tx(v22bis_tx_ctrl.asm)
000020D8 00000038 rtlib.text F@DummyFn1(v22bis_tx_enc.asm)
000020D8 00000038 rtlib.text rtlib.text(v22bis_tx_enc.asm)
000020D8 00000000 rtlib.text tx_sbit(v22bis_tx_enc.asm)
000020DE 00000000 rtlib.text tx_enc_1(v22bis_tx_enc.asm)
000020E7 00000000 rtlib.text tx_enc_2(v22bis_tx_enc.asm)
000020ED 00000000 rtlib.text tx_enc_4(v22bis_tx_enc.asm)
00002110 00000000 rtlib.text tx_sil(v22bis_tx_feed.asm)
00002110 00000079 rtlib.text F@DummyFn1(v22bis_tx_feed.asm)
00002110 00000079 rtlib.text rtlib.text(v22bis_tx_feed.asm)
00002117 00000000 rtlib.text tx_a_ton(v22bis_tx_feed.asm)
00002131 00000000 rtlib.text tx_wr4(v22bis_tx_feed.asm)
0000213E 00000000 rtlib.text tx_one_2(v22bis_tx_feed.asm)
00002143 00000000 rtlib.text tx_s1(v22bis_tx_feed.asm)
0000214D 00000000 rtlib.text tx_decide(v22bis_tx_feed.asm)
0000215B 00000000 rtlib.text tx_109(v22bis_tx_feed.asm)
00002167 00000000 rtlib.text tx_112(v22bis_tx_feed.asm)
0000217A 00000000 rtlib.text tx_one_4(v22bis_tx_feed.asm)
0000217F 00000000 rtlib.text dummy(v22bis_tx_feed.asm)
00002181 00000000 rtlib.text tx_in_2(v22bis_tx_feed.asm)
00002183 00000000 rtlib.text tx_in_4(v22bis_tx_feed.asm)
00002185 00000000 rtlib.text ERROR(v22bis_tx_feed.asm)
00002189 00000034 rtlib.text F@DummyFn1(v22bis_tx_fm.asm)
00002189 00000000 rtlib.text tx_fm(v22bis_tx_fm.asm)
00002189 00000034 rtlib.text rtlib.text(v22bis_tx_fm.asm)
000021BD 0000003A rtlib.text F@DummyFn1(v22bis_tx_scr.asm)
000021BD 0000003A rtlib.text rtlib.text(v22bis_tx_scr.asm)
000021BD 00000000 rtlib.text tx_scr_1(v22bis_tx_scr.asm)
000021C1 00000000 rtlib.text tx_scr_2(v22bis_tx_scr.asm)
000021C5 00000000 rtlib.text tx_scr_4(v22bis_tx_scr.asm)
000021C8 00000000 rtlib.text tx_scr(v22bis_tx_scr.asm)
000021F7 000000DF rtlib.text rtlib.text(v22bis_tx_state.asm)
000021F7 000000DF rtlib.text F@DummyFn1(v22bis_tx_state.asm)
000021F7 00000000 rtlib.text tx_I1(v22bis_tx_state.asm)
000021FF 00000000 rtlib.text tx_I2(v22bis_tx_state.asm)
00002215 00000000 rtlib.text tx_I3(v22bis_tx_state.asm)
00002222 00000000 rtlib.text tx_I4(v22bis_tx_state.asm)
00002240 00000000 rtlib.text tx_I5(v22bis_tx_state.asm)
0000226F 00000000 rtlib.text tx_I6_1(v22bis_tx_state.asm)
00002291 00000000 rtlib.text tx_I6_2(v22bis_tx_state.asm)
000022A4 00000000 rtlib.text tx_I7_2(v22bis_tx_state.asm)
000022BA 00000000 rtlib.text tx_I8_2(v22bis_tx_state.asm)
000022D6 000000E6 rtlib.text F@DummyFn1(v22bis_txmdmini.asm)
000022D6 00000000 rtlib.text TX_MDM_INIT(v22bis_txmdmini.asm)
```

```
000022D6 000000E6 rtlib.text rtlib.text(v22bis_txmdmini.asm)
000023BC 00000068 rtlib.text rtlib.text(v22bis.asm)
000023BC 00000000 rtlib.text V22BIS_INIT(v22bis.asm)
000023BC 00000068 rtlib.text F@DummyFn1(v22bis.asm)
0000241E 00000000 rtlib.text V22BIS_TX(v22bis.asm)
0000241E 00000000 rtlib.text V22BIS_TX_DLB(v22bis.asm)
0000241E 00000000 rtlib.text V22BIS_TX_ALB(v22bis.asm)
00002421 00000000 rtlib.text V22BIS_RX_DLB(v22bis.asm)
00002421 00000000 rtlib.text V22BIS_RX_ALB(v22bis.asm)
00002421 00000000 rtlib.text V22BIS_RX(v22bis.asm)
00002424 000001F3 rtlib.text rtlib.text(v22bis_initsr.asm)
00002424 00000000 rtlib.text INIT_SP_COMMON(v22bis_initsr.asm)
00002424 000001F3 rtlib.text F@DummyFn1(v22bis_initsr.asm)
000024A4 00000000 rtlib.text CLR_RAM2(v22bis_initsr.asm)
000024C7 00000000 rtlib.text INIT_BEG_AGC(v22bis_initsr.asm)
000024DD 00000000 rtlib.text INIT_BEG(v22bis_initsr.asm)
00002551 00000000 rtlib.text CLEQ_INIT(v22bis_initsr.asm)
000025E6 00000000 rtlib.text AGC_JAM(v22bis_initsr.asm)
00002617 000000B9 rtlib.text F@DummyFn1(v22bis_rx_baud.asm)
00002617 00000000 rtlib.text RXBAUD(v22bis_rx_baud.asm)
00002617 000000B9 rtlib.text rtlib.text(v22bis_rx_baud.asm)
000026D0 000002DA rtlib.text rtlib.text(v22bis_rx_bchk.asm)
000026D0 00000000 rtlib.text RX_wait(v22bis_rx_bchk.asm)
000026D0 000002DA rtlib.text F@DummyFn1(v22bis_rx_bchk.asm)
000026DC 00000000 rtlib.text RX_cd(v22bis_rx_bchk.asm)
000026EC 00000000 rtlib.text RX_atusb1(v22bis_rx_bchk.asm)
000026F5 00000000 rtlib.text RX_usb1(v22bis_rx_bchk.asm)
0000271C 00000000 rtlib.text RX_endusb1(v22bis_rx_bchk.asm)
00002729 00000000 rtlib.text RX_s1call(v22bis_rx_bchk.asm)
0000275D 00000000 rtlib.text RX_cdrop(v22bis_rx_bchk.asm)
0000277A 00000000 rtlib.text RX_signal(v22bis_rx_bchk.asm)
00002796 00000000 rtlib.text RX_carrierup(v22bis_rx_bchk.asm)
000027A9 00000000 rtlib.text RX_s1ans(v22bis_rx_bchk.asm)
000027E2 00000000 rtlib.text RX_scr12(v22bis_rx_bchk.asm)
00002838 00000000 rtlib.text RX_v22dm(v22bis_rx_bchk.asm)
0000285C 00000000 rtlib.text RX_s1end(v22bis_rx_bchk.asm)
00002868 00000000 rtlib.text RX_wait32bit(v22bis_rx_bchk.asm)
00002884 00000000 rtlib.text RX_waitdm(v22bis_rx_bchk.asm)
0000288D 00000000 rtlib.text RX_wait1sec(v22bis_rx_bchk.asm)
0000289D 00000000 rtlib.text RX_v22bisdm(v22bis_rx_bchk.asm)
000028DA 00000000 rtlib.text RX_retrain(v22bis_rx_bchk.asm)
000028FE 00000000 rtlib.text RX_RETR_REP(v22bis_rx_bchk.asm)
00002909 00000000 rtlib.text RX_RETR_A(v22bis_rx_bchk.asm)
00002923 00000000 rtlib.text RX_NEXT(v22bis_rx_bchk.asm)
00002925 00000000 rtlib.text ENDRX(v22bis_rx_bchk.asm)
00002926 00000000 rtlib.text error(v22bis_rx_bchk.asm)
00002947 00000000 rtlib.text RXDM_CD(v22bis_rx_bchk.asm)
```

```
00002955 00000000 rtlib.text RXDMCDON(v22bis_rx_bchk.asm)
00002985 00000000 rtlib.text RXDMCDOFF(v22bis_rx_bchk.asm)
0000298C 00000000 rtlib.text RXDM_OFF(v22bis_rx_bchk.asm)
0000299E 00000000 rtlib.text CDONOF(v22bis_rx_bchk.asm)
000029AA 0000004C rtlib.text F@DummyFn1(v22bis_rx_bpf.asm)
000029AA 00000000 rtlib.text RXBPF(v22bis_rx_bpf.asm)
000029AA 0000004C rtlib.text rtlib.text(v22bis_rx_bpf.asm)
000029F6 00000021 rtlib.text F@DummyFn1(v22bis_rx_car.asm)
000029F6 00000000 rtlib.text RXCAR(v22bis_rx_car.asm)
000029F6 00000021 rtlib.text rtlib.text(v22bis_rx_car.asm)
00002A17 000000B9 rtlib.text F@DummyFn1(v22bis_rx_cdagc.asm)
00002A17 00000000 rtlib.text RXCDAGC(v22bis_rx_cdagc.asm)
00002A17 000000B9 rtlib.text rtlib.text(v22bis_rx_cdagc.asm)
00002AD0 00000019 rtlib.text rtlib.text(v22bis_rx_ctrl.asm)
00002AD0 00000000 rtlib.text RXBAUDPROC(v22bis_rx_ctrl.asm)
00002AD0 00000019 rtlib.text F@DummyFn1(v22bis_rx_ctrl.asm)
00002AD8 00000000 rtlib.text rx_no_sti(v22bis_rx_ctrl.asm)
00002ADD 00000000 rtlib.text rx_next_task(v22bis_rx_ctrl.asm)
00002AE9 00000031 rtlib.text F@DummyFn1(v22bis_rx_decim.asm)
00002AE9 00000000 rtlib.text RXDECIM(v22bis_rx_decim.asm)
00002AE9 00000031 rtlib.text rtlib.text(v22bis_rx_decim.asm)
00002B1A 0000004A rtlib.text F@DummyFn1(v22bis_rx_demod.asm)
00002B1A 00000000 rtlib.text RXDEMOD(v22bis_rx_demod.asm)
00002B1A 0000004A rtlib.text rtlib.text(v22bis_rx_demod.asm)
00002B64 0000007E rtlib.text rtlib.text(v22bis_rx_difdc.asm)
00002B64 00000000 rtlib.text RXDEC4(v22bis_rx_difdc.asm)
00002B64 0000007E rtlib.text F@DummyFn1(v22bis_rx_difdc.asm)
00002B75 00000000 rtlib.text DECA(v22bis_rx_difdc.asm)
00002B7F 00000000 rtlib.text DECB(v22bis_rx_difdc.asm)
00002B94 00000000 rtlib.text RXDEC16(v22bis_rx_difdc.asm)
00002BBD 00000000 rtlib.text RXDIFDEC(v22bis_rx_difdc.asm)
00002BE2 00000000 rtlib.text RXDESCR2(v22bis_rx_dscr.asm)
00002BE2 0000004C rtlib.text rtlib.text(v22bis_rx_dscr.asm)
00002BE2 0000004C rtlib.text F@DummyFn1(v22bis_rx_dscr.asm)
00002BE6 00000000 rtlib.text RXDESCR4(v22bis_rx_dscr.asm)
00002BEA 00000000 rtlib.text RXDESCR16(v22bis_rx_dscr.asm)
00002BED 00000000 rtlib.text rx_dscr(v22bis_rx_dscr.asm)
00002C28 00000000 rtlib.text dscr_upd(v22bis_rx_dscr.asm)
00002C2E 0000007E rtlib.text F@DummyFn1(v22bis_rx_eqerr.asm)
00002C2E 00000000 rtlib.text RXEQERR(v22bis_rx_eqerr.asm)
00002C2E 0000007E rtlib.text rtlib.text(v22bis_rx_eqerr.asm)
00002CAC 0000005A rtlib.text F@DummyFn1(v22bis_rx_eqfil.asm)
00002CAC 00000000 rtlib.text RXEQFIL(v22bis_rx_eqfil.asm)
00002CAC 0000005A rtlib.text rtlib.text(v22bis_rx_eqfil.asm)
00002D06 0000002C rtlib.text F@DummyFn1(v22bis_rx_equpd.asm)
00002D06 00000000 rtlib.text RXEQUD(v22bis_rx_equpd.asm)
00002D06 0000002C rtlib.text rtlib.text(v22bis_rx_equpd.asm)
```

```
00002D32 0000001B rtlib.text F@DummyFn1(v22bis_rx_int.asm)
00002D32 00000000 rtlib.text RXINTP(v22bis_rx_int.asm)
00002D32 0000001B rtlib.text rtlib.text(v22bis_rx_int.asm)
00002D4D 000002F2 rtlib.text rtlib.text(v22bis_rx_stat.asm)
00002D4D 00000000 rtlib.text rx_CA(v22bis_rx_stat.asm)
00002D4D 000002F2 rtlib.text F@DummyFn1(v22bis_rx_stat.asm)
00002D5E 00000000 rtlib.text rx_CB(v22bis_rx_stat.asm)
00002D69 00000000 rtlib.text rx_CC(v22bis_rx_stat.asm)
00002D89 00000000 rtlib.text rx_CD(v22bis_rx_stat.asm)
00002D91 00000000 rtlib.text rx_CE(v22bis_rx_stat.asm)
00002DAF 00000000 rtlib.text rx_CF(v22bis_rx_stat.asm)
00002DC4 00000000 rtlib.text rx_AA(v22bis_rx_stat.asm)
00002DCF 00000000 rtlib.text rx_AB(v22bis_rx_stat.asm)
00002DDA 00000000 rtlib.text rx_AC(v22bis_rx_stat.asm)
00002DF1 00000000 rtlib.text rx_AC1(v22bis_rx_stat.asm)
00002E07 00000000 rtlib.text rx_AD(v22bis_rx_stat.asm)
00002E25 00000000 rtlib.text rx_G22A(v22bis_rx_stat.asm)
00002E3F 00000000 rtlib.text rx_G22B(v22bis_rx_stat.asm)
00002E61 00000000 rtlib.text rx_G22C(v22bis_rx_stat.asm)
00002E9C 00000000 rtlib.text rx_G22D(v22bis_rx_stat.asm)
00002EEB 00000000 rtlib.text rx_GBisA(v22bis_rx_stat.asm)
00002F0B 00000000 rtlib.text rx_GBisB(v22bis_rx_stat.asm)
00002F16 00000000 rtlib.text rx_GBisC(v22bis_rx_stat.asm)
00002F5D 00000000 rtlib.text rx_GBisD(v22bis_rx_stat.asm)
00002F8C 00000000 rtlib.text rx_GBisE(v22bis_rx_stat.asm)
00002FA8 00000000 rtlib.text rx_GBisF(v22bis_rx_stat.asm)
00002FF2 00000000 rtlib.text rx_GBisG(v22bis_rx_stat.asm)
00003010 00000000 rtlib.text rx_GRetA(v22bis_rx_stat.asm)
0000303F 000000FD rtlib.text F@DummyFn1(v22bis_rx_ton.asm)
0000303F 00000000 rtlib.text RXUSB1(v22bis_rx_ton.asm)
0000303F 000000FD rtlib.text rtlib.text(v22bis_rx_ton.asm)
0000307E 00000000 rtlib.text RXS1(v22bis_rx_ton.asm)
0000311B 00000000 rtlib.text RXTON(v22bis_rx_ton.asm)
0000313C 00000091 rtlib.text F@DummyFn1(v22bis_rxmdmini.asm)
0000313C 00000000 rtlib.text RX_MDM_INIT(v22bis_rxmdmini.asm)
0000313C 00000091 rtlib.text rtlib.text(v22bis_rxmdmini.asm)
000031CD 0000001F rtlib.text F@DummyFn1(v22bis_tondet.asm)
000031CD 00000000 rtlib.text TONEDETECT(v22bis_tondet.asm)
000031CD 0000001F rtlib.text rtlib.text(v22bis_tondet.asm)
000031EC 0000000F rtlib.text F@DummyFn1(v22bis_rxstub.asm)
000031EC 00000000 rtlib.text rx_stub(v22bis_rxstub.asm)
000031EC 0000000F rtlib.text rtlib.text(v22bis_rxstub.asm)
000031FB 00000026 rtlib.text F@DummyFn1(v22bis_txstub.asm)
000031FB 00000000 rtlib.text tx_stub(v22bis_txstub.asm)
000031FB 00000026 rtlib.text rtlib.text(v22bis_txstub.asm)
00003221 000000CE rtlib.text F@DummyFn1(v22bisapi.asm)
00003221 00000000 rtlib.text FINITIALIZE_V22BIS(v22bisapi.asm)
```

```
00003221 000000CE rtlib.text rtlib.text(v22bisapi.asm)
00003256 00000000 rtlib.text FV22BIS_TRANSMIT(v22bisapi.asm)
00003278 00000000 rtlib.text FV22BIS_RECEIVE_SAMPLE(v22bisapi.asm)
000032EF 00000094 rtlib.text F@DummyFn1(v22_v42d.asm)
000032EF 00000000 rtlib.text V42DRV_INIT(v22_v42d.asm)
000032EF 00000094 rtlib.text rtlib.text(v22_v42d.asm)
00003302 00000000 rtlib.text FV42_V22DRV_INIT(v22_v42d.asm)
00003308 00000000 rtlib.text V22_V42DRV(v22_v42d.asm)
00003337 00000000 rtlib.text V42_V22DRV(v22_v42d.asm)
0000333A 00000000 rtlib.text LAPM_MDM_INIT(v22_v42d.asm)
0000333B 00000000 rtlib.text WRITE_NIBBLE(v22_v42d.asm)
0000333D 00000000 rtlib.text READ_NIBBLE(v22_v42d.asm)
0000337E 00000000 rtlib.text End_ReadNibble(v22_v42d.asm)
00003383 00000011 rtlib.text F@DummyFn1(archgetsetsaturationmode.asm)
00003383 00000000 rtlib.text FarchGetSetSaturationMode(archgetsetsaturationmode.asm)
00003383 00000011 rtlib.text rtlib.text(archgetsetsaturationmode.asm)
00003394 0000000B rtlib.text F@DummyFn1(memcpy.asm)
00003394 00000000 rtlib.text FmemMemcpy(memcpy.asm)
00003394 0000000B rtlib.text rtlib.text(memcpy.asm)
0000339F 00000020 rtlib.text F@DummyFn1(memset.asm)
0000339F 00000000 rtlib.text FmemMemset(memset.asm)
0000339F 00000020 rtlib.text rtlib.text(memset.asm)
000033BF 00000049 rtlib.text F@DummyFn1(process_cpt.asm)
000033BF 00000000 rtlib.text FPROCESS_CPT(process_cpt.asm)
000033BF 00000049 rtlib.text rtlib.text(process_cpt.asm)
00003408 0000018C rtlib.text F@DummyFn1(cpsi_api.asm)
00003408 00000000 rtlib.text SILENCE_DETECT_CPT(cpsi_api.asm)
00003408 0000018C rtlib.text rtlib.text(cpsi_api.asm)
0000340B 00000000 rtlib.text SILENCE_DEBOUNCE(cpsi_api.asm)
0000343F 00000000 rtlib.text FCALLPROGRESS_DETECT_INIT(cpsi_api.asm)
0000346F 00000000 rtlib.text CALLPROGRESS_DETECT(cpsi_api.asm)
0000347B 00000000 rtlib.text CALLPROGRESS_DEBOUNCE(cpsi_api.asm)
00003484 00000000 rtlib.text no_cpt(cpsi_api.asm)
00003490 00000000 rtlib.text cpt_on(cpsi_api.asm)
0000349E 00000000 rtlib.text noisy_cpt(cpsi_api.asm)
000034A9 00000000 rtlib.text end_cpt(cpsi_api.asm)
000034B7 00000000 rtlib.text cpt_silence(cpsi_api.asm)
000034C4 00000000 rtlib.text noisy_sil(cpsi_api.asm)
000034D0 00000000 rtlib.text new_cpt(cpsi_api.asm)
000034D9 00000000 rtlib.text check_previous_cpt(cpsi_api.asm)
000034E5 00000000 rtlib.text check_cpt_off(cpsi_api.asm)
000034F2 00000000 rtlib.text reset_cpsi(cpsi_api.asm)
000034F9 00000000 rtlib.text exit_cpt_debounce(cpsi_api.asm)
00003502 00000000 rtlib.text CALLPROGRESS_DECODE(cpsi_api.asm)
00003521 00000000 rtlib.text clear_bursts(cpsi_api.asm)
00003526 00000000 rtlib.text check_group2(cpsi_api.asm)
00003553 00000000 rtlib.text check_group3(cpsi_api.asm)
```

```
00003567 00000000 rtlib.text last_on(cpsi_api.asm)
0000357F 00000000 rtlib.text end_cpt_on(cpsi_api.asm)
00003587 00000000 rtlib.text not_cpt(cpsi_api.asm)
0000358E 00000000 rtlib.text exit_decode(cpsi_api.asm)
00003594 00000044 rtlib.text F@DummyFn1(cpsi_low.asm)
00003594 00000044 rtlib.text rtlib.text(cpsi_low.asm)
00003594 00000000 rtlib.text SIL_DEC_CPT(cpsi_low.asm)
000035A9 00000000 rtlib.text assign1_cpt(cpsi_low.asm)
000035D8 0000003D rtlib.text F@DummyFn1(cpt_api.asm)
000035D8 00000000 rtlib.text PAPI_TONE_DETECT(cpt_api.asm)
000035D8 0000003D rtlib.text rtlib.text(cpt_api.asm)
00003615 00000046 rtlib.text rtlib.text(cpt_buf.asm)
00003615 00000000 rtlib.text GENERATE_ANALYSIS_ARRAY_CPT(cpt_buf.asm)
00003615 00000046 rtlib.text F@DummyFn1(cpt_buf.asm)
00003647 00000000 rtlib.text CALC_SIG_EN_CPT(cpt_buf.asm)
0000365B 0000018C rtlib.text rtlib.text(cpt_low.asm)
0000365B 0000018C rtlib.text F@DummyFn1(cpt_low.asm)
0000365B 00000000 rtlib.text MG_EN(cpt_low.asm)
0000366D 00000000 rtlib.text end_mg_cpt(cpt_low.asm)
0000366F 00000000 rtlib.text NEWNUM_CPT(cpt_low.asm)
00003687 00000000 rtlib.text TST_CPT(cpt_low.asm)
000036A3 00000000 rtlib.text UPDCPT(cpt_low.asm)
000036A5 00000000 rtlib.text l1(cpt_low.asm)
000036BC 00000000 rtlib.text FIND_PK_CPT(cpt_low.asm)
000036BC 00000000 rtlib.text End_upd_subroutine(cpt_low.asm)
000036D8 00000000 rtlib.text LOAD_THRESH_CPT(cpt_low.asm)
00003746 00000000 rtlib.text MAG_CPT(cpt_low.asm)
0000375B 00000000 rtlib.text GROUP_TST_CPT(cpt_low.asm)
00003779 00000000 rtlib.text TWIST_CPT(cpt_low.asm)
00003779 00000000 rtlib.text REL_EN_CPT(cpt_low.asm)
00003792 00000000 rtlib.text REL_MAG_CPT(cpt_low.asm)
000037DA 00000000 rtlib.text FIND_REL_MAG(cpt_low.asm)
000037E7 00000036 rtlib.text rtlib.text(v21_mod.asm)
000037E7 00000000 rtlib.text Fv21ModInit(v21_mod.asm)
000037E7 00000036 rtlib.text F@DummyFn1(v21_mod.asm)
00003806 00000000 rtlib.text Fv21Mod(v21_mod.asm)
0000381D 00000004 rtlib.text F@DummyFn1(v21_agc.asm)
0000381D 00000000 rtlib.text V21_Rxagcgjam(v21_agc.asm)
0000381D 00000004 rtlib.text rtlib.text(v21_agc.asm)
00003821 00000034 rtlib.text F@DummyFn1(v21_cd.asm)
00003821 00000000 rtlib.text V21_RxCd(v21_cd.asm)
00003821 00000034 rtlib.text rtlib.text(v21_cd.asm)
00003855 00000013 rtlib.text F@DummyFn1(v21_rxavg_fil.asm)
00003855 00000000 rtlib.text V21_RxAvg_Filter(v21_rxavg_fil.asm)
00003855 00000013 rtlib.text rtlib.text(v21_rxavg_fil.asm)
00003868 00000085 rtlib.text rtlib.text(v21_rxbchk.asm)
00003868 00000000 rtlib.text V21_Rxcdwait(v21_rxbchk.asm)
```

```
00003868 00000085 rtlib.text F@DummyFn1(v21_rxbchk.asm)
00003880 00000000 rtlib.text V21_Rxwait(v21_rxbchk.asm)
0000389C 00000000 rtlib.text V21_Rxagc(v21_rxbchk.asm)
000038C1 00000000 rtlib.text V21_Rxfirstzc(v21_rxbchk.asm)
000038DA 00000000 rtlib.text V21_Rxdata(v21_rxbchk.asm)
000038ED 00000013 rtlib.text F@DummyFn1(v21_rxdecode.asm)
000038ED 00000000 rtlib.text V21_RxDecode(v21_rxdecode.asm)
000038ED 00000013 rtlib.text rtlib.text(v21_rxdecode.asm)
00003900 0000009B rtlib.text rtlib.text(v21_rxdemod.asm)
00003900 00000000 rtlib.text Fv21RxDemodInit(v21_rxdemod.asm)
00003900 0000009B rtlib.text F@DummyFn1(v21_rxdemod.asm)
00003956 00000000 rtlib.text V21_RxDemod(v21_rxdemod.asm)
0000399B 00000032 rtlib.text F@DummyFn1(v21_rxdiv.asm)
0000399B 00000000 rtlib.text V21_RxDiv(v21_rxdiv.asm)
0000399B 00000032 rtlib.text rtlib.text(v21_rxdiv.asm)
000039CD 00000042 rtlib.text F@DummyFn1(v21_rxfreq_shift.asm)
000039CD 00000000 rtlib.text V21_RxFreq_Shift(v21_rxfreq_shift.asm)
000039CD 00000042 rtlib.text rtlib.text(v21_rxfreq_shift.asm)
00003A0F 00000033 rtlib.text F@DummyFn1(v21_rxlpf.asm)
00003A0F 00000000 rtlib.text V21_RxLpf(v21_rxlpf.asm)
00003A0F 00000033 rtlib.text rtlib.text(v21_rxlpf.asm)
00003A42 00000016 rtlib.text rtlib.text(v21_rxsm.asm)
00003A42 00000000 rtlib.text Fv21Rxctrl(v21_rxsm.asm)
00003A42 00000016 rtlib.text F@DummyFn1(v21_rxsm.asm)
00003A50 00000000 rtlib.text V21_Rx_Nxt_Tsk(v21_rxsm.asm)
00003A58 00000000 rtlib.text V21_Rxcdw_Init(v21_rxsti.asm)
00003A58 0000007B rtlib.text rtlib.text(v21_rxsti.asm)
00003A58 0000007B rtlib.text F@DummyFn1(v21_rxsti.asm)
00003A79 00000000 rtlib.text V21_Rxagc_Init(v21_rxsti.asm)
00003A94 00000000 rtlib.text V21_Rxagcfzc_Init(v21_rxsti.asm)
00003AB6 00000000 rtlib.text V21_Rxfzc_Init(v21_rxsti.asm)
00003AC0 00000000 rtlib.text V21_Rxdat_Init(v21_rxsti.asm)
00003AD3 00000044 rtlib.text F@DummyFn1(v21_rxtimejam.asm)
00003AD3 00000000 rtlib.text V21_RxTimejam(v21_rxtimejam.asm)
00003AD3 00000044 rtlib.text rtlib.text(v21_rxtimejam.asm)
00003B17 0000009F rtlib.text F@DummyFn1(v21_rxtimrec.asm)
00003B17 00000000 rtlib.text V21_RxTimrec(v21_rxtimrec.asm)
00003B17 0000009F rtlib.text rtlib.text(v21_rxtimrec.asm)
00003BB6 00000042 rtlib.text rtlib.text(Runtime 56800E.Lib save_reg.o     )
00003BB6 00000000 rtlib.text INTERRUPT_SAVEALL(Runtime 56800E.Lib save_reg.o     )
00003BD9 00000000 rtlib.text INTERRUPT_RESTOREALL(Runtime 56800E.Lib save_reg.o     )
00003BF8 00000035 rtlib.text F@DummyFn1(Runtime 56800E.Lib artdivrec_s32_0)
00003BF8 00000000 rtlib.text FARTDIVREC_S16(Runtime 56800E.Lib artdivrec_s32_0)
00003BF8 00000000 rtlib.text ARTDIVREC_S16(Runtime 56800E.Lib artdivrec_s32_0)
00003BF8 00000035 rtlib.text rtlib.text(Runtime 56800E.Lib artdivrec_s32_0)
00003BFF 00000000 rtlib.text ARTDIVREC_U16(Runtime 56800E.Lib artdivrec_s32_0)
00003BFF 00000000 rtlib.text FARTDIVREC_U16(Runtime 56800E.Lib artdivrec_s32_0)
```

```
  00003C06 00000000 rtlib.text ARTDIVREC_S32(Runtime 56800E.Lib artdivrec_s32_0)
  00003C06 00000000 rtlib.text FARTDIVREC_S32(Runtime 56800E.Lib artdivrec_s32_0)
  00003C18 00000000 rtlib.text ARTDIVREC_U32(Runtime 56800E.Lib artdivrec_s32_0)
  00003C18 00000000 rtlib.text FARTDIVREC_U32(Runtime 56800E.Lib artdivrec_s32_0)
  00003C2D 00000100 rtlib.text rtlib.text(dtmfgen_3x.lib dtmf_gen.o     )
  00003C2D 00000000 rtlib.text FdtmfGenerate(dtmfgen_3x.lib dtmf_gen.o     )
  00003C74 00000000 rtlib.text FdtmfInit(dtmfgen_3x.lib dtmf_gen.o     )
  00003CBC 00000000 rtlib.text FdtmfSetKey(dtmfgen_3x.lib dtmf_gen.o     )
  00003D2D 00000050 startup.text startup.text(56F83x_init.asm)
  00003D2D 0000004E startup.text Finit_56800_(56F83x_init.asm)
  00003D7B 00000002 startup.text F@DummyFn1(56F83x_init.asm)
#>00003D7D          F_Pbss_start_addr (linker command file)
#>00003D7D          _P_BSS_ADDR (linker command file)
#>00000000          F_Pbss_length (linker command file)
#>00003D7D          F_Pcode_end_addr (linker command file)
#>00003D7E          __pROM_data_start (linker command file)

# .data_in_p_flash_ROM
#>00000000          __xRAM_data_start (linker command file)
  00000000 00000000 rtlib.data cpt_cosval(cpt_dc.asm)
  00000000 00000006 rtlib.data rtlib.data(cpt_dc.asm)
  00000006 00000002 .data   FBitIO_portDsc(Cpu.c)
  00000008 00000001 .data   FpModemRxWrt(modem.c)
  00000009 00000001 .data   FpModemRxRead(modem.c)
  0000000A 00000001 .data   FresTx(modem.c)
  0000000B 00000001 .data   FresRx(modem.c)
  0000000C 00000018 .data   FV25ansTone(modem.c)
  00000024 00000001 .data   FBitCounterRx1(modem.c)
  00000025 00000000 TX_MEM.data MDMCONFIG(v22bis_gmdmmem.asm)
  00000025 00000000 TX_MEM.data FMDMCONFIG(v22bis_gmdmmem.asm)
  00000025 0000000E TX_MEM.data TX_MEM.data(v22bis_gmdmmem.asm)
  00000027 00000000 TX_MEM.data TX_GAIN(v22bis_gmdmmem.asm)
  00000028 00000000 TX_MEM.data MDMSTATUS(v22bis_gmdmmem.asm)
  00000028 00000000 TX_MEM.data FMDMSTATUS(v22bis_gmdmmem.asm)
  00000029 00000000 TX_MEM.data mode_flg(v22bis_gmdmmem.asm)
  0000002A 00000000 TX_MEM.data rx_st_id(v22bis_gmdmmem.asm)
  0000002B 00000000 TX_MEM.data tx_st_id(v22bis_gmdmmem.asm)
  0000002C 00000000 TX_MEM.data flg_107(v22bis_gmdmmem.asm)
  0000002D 00000000 TX_MEM.data flg_112(v22bis_gmdmmem.asm)
  0000002E 00000000 TX_MEM.data flg_109(v22bis_gmdmmem.asm)
  0000002F 00000000 TX_MEM.data flg_104(v22bis_gmdmmem.asm)
  00000030 00000000 TX_MEM.data flg_106(v22bis_gmdmmem.asm)
  00000031 00000000 TX_MEM.data loopback(v22bis_gmdmmem.asm)
  00000032 00000000 TX_MEM.data Fretrain_flag(v22bis_gmdmmem.asm)
  00000032 00000000 TX_MEM.data retrain_flag(v22bis_gmdmmem.asm)
  00000033 00000000 TX_MEM.data txI1ctr(v22bis_txmdmmem.asm)
  00000033 00000000 TX_MEM.data TXMEMB(v22bis_txmdmmem.asm)
```

```
00000033 00000047 TX_MEM.data TX_MEM.data(v22bis_txmdmmem.asm)
00000034 00000000 TX_MEM.data txI2ctr(v22bis_txmdmmem.asm)
00000035 00000000 TX_MEM.data txI3ctr(v22bis_txmdmmem.asm)
00000036 00000000 TX_MEM.data txI4ctr(v22bis_txmdmmem.asm)
00000037 00000000 TX_MEM.data txI51ctr(v22bis_txmdmmem.asm)
00000038 00000000 TX_MEM.data txI52ctr(v22bis_txmdmmem.asm)
00000039 00000000 TX_MEM.data txI61ctr(v22bis_txmdmmem.asm)
0000003A 00000000 TX_MEM.data txI62ctr(v22bis_txmdmmem.asm)
0000003B 00000000 TX_MEM.data txI72ctr(v22bis_txmdmmem.asm)
0000003C 00000000 TX_MEM.data txI82ctr(v22bis_txmdmmem.asm)
0000003D 00000000 TX_MEM.data mdm_flg(v22bis_txmdmmem.asm)
0000003E 00000000 TX_MEM.data gt_flg(v22bis_txmdmmem.asm)
0000003F 00000000 TX_MEM.data ccitt_flg(v22bis_txmdmmem.asm)
00000040 00000000 TX_MEM.data tx_ans_flg(v22bis_txmdmmem.asm)
00000041 00000000 TX_MEM.data tx_rx16(v22bis_txmdmmem.asm)
00000042 00000000 TX_MEM.data atone_ptr(v22bis_txmdmmem.asm)
00000043 00000000 TX_MEM.data tx_data(v22bis_txmdmmem.asm)
00000044 00000000 TX_MEM.data tx_out(v22bis_txmdmmem.asm)
00000044 00000000 TX_MEM.data Ftx_out(v22bis_txmdmmem.asm)
00000050 00000000 TX_MEM.data tx_quad(v22bis_txmdmmem.asm)
00000051 00000000 TX_MEM.data Ival(v22bis_txmdmmem.asm)
00000052 00000000 TX_MEM.data Qval(v22bis_txmdmmem.asm)
00000053 00000000 TX_MEM.data tx_scr_buf(v22bis_txmdmmem.asm)
00000054 00000000 TX_MEM.data tx_scr_buf_1(v22bis_txmdmmem.asm)
00000055 00000000 TX_MEM.data tx_scr_ctr(v22bis_txmdmmem.asm)
00000056 00000000 TX_MEM.data gtamp(v22bis_txmdmmem.asm)
00000057 00000000 TX_MEM.data gtone_ptr(v22bis_txmdmmem.asm)
00000058 00000000 TX_MEM.data tx_fm_buf(v22bis_txmdmmem.asm)
0000005E 00000000 TX_MEM.data tx_fm_coef(v22bis_txmdmmem.asm)
0000005F 00000000 TX_MEM.data tx_fm_gt_offset(v22bis_txmdmmem.asm)
00000060 00000000 TX_MEM.data tx_ctr(v22bis_txmdmmem.asm)
00000061 00000000 TX_MEM.data tx_tmp(v22bis_txmdmmem.asm)
00000062 00000000 TX_MEM.data tmp_flg(v22bis_txmdmmem.asm)
00000063 00000000 TX_MEM.data tx_st_chg(v22bis_txmdmmem.asm)
00000064 00000000 TX_MEM.data TxQ(v22bis_txmdmmem.asm)
00000065 00000000 TX_MEM.data TxQ_1(v22bis_txmdmmem.asm)
00000066 00000000 TX_MEM.data TxQ_2(v22bis_txmdmmem.asm)
00000067 00000000 TX_MEM.data TxQ_3(v22bis_txmdmmem.asm)
00000068 00000000 TX_MEM.data TxQ_4(v22bis_txmdmmem.asm)
00000069 00000000 TX_MEM.data TxQ_5(v22bis_txmdmmem.asm)
0000006A 00000000 TX_MEM.data StQ1(v22bis_txmdmmem.asm)
0000006B 00000000 TX_MEM.data StQ1_1(v22bis_txmdmmem.asm)
0000006C 00000000 TX_MEM.data StQ1_2(v22bis_txmdmmem.asm)
0000006D 00000000 TX_MEM.data StQ1_3(v22bis_txmdmmem.asm)
0000006E 00000000 TX_MEM.data StQ1_4(v22bis_txmdmmem.asm)
0000006F 00000000 TX_MEM.data StQ1_5(v22bis_txmdmmem.asm)
00000070 00000000 TX_MEM.data StQ2(v22bis_txmdmmem.asm)
```

```
00000073 00000000 TX_MEM.data StQ_ptr(v22bis_txmdmmem.asm)
00000074 00000000 TX_MEM.data TxQ_ptr(v22bis_txmdmmem.asm)
00000075 00000000 TX_MEM.data TXMEMSIZE(v22bis_txmdmmem.asm)
00000076 00000000 TX_MEM.data DC_Alpha(v22bis_txmdmmem.asm)
00000077 00000000 TX_MEM.data DC_Tap(v22bis_txmdmmem.asm)
00000078 00000000 TX_MEM.data DC_Tap_Scaled(v22bis_txmdmmem.asm)
00000079 00000000 TX_MEM.data DC_Error(v22bis_txmdmmem.asm)
0000007A 00000000 API.data cnt7(v22bis_rxstub.asm)
0000007A 00000002 API.data API.data(v22bis_rxstub.asm)
00000100 00000000 V22B_PROM.data SIN_TBL(v22bis_mdm_prom.asm)
00000100 000002EC V22B_PROM.data V22B_PROM.data(v22bis_mdm_prom.asm)
00000200 00000000 V22B_PROM.data cos2100(v22bis_mdm_prom.asm)
00000220 00000000 V22B_PROM.data MOD_TBL(v22bis_mdm_prom.asm)
0000022C 00000000 V22B_PROM.data absdat(v22bis_mdm_prom.asm)
0000023C 00000000 V22B_PROM.data tx_quadtab(v22bis_mdm_prom.asm)
0000024C 00000000 V22B_PROM.data tx_IQmap(v22bis_mdm_prom.asm)
0000026C 00000000 V22B_PROM.data IFCOE(v22bis_mdm_prom.asm)
000003EC 00000000 ROM_XMEM.data PAR_2100(v22bis_mdm_xrom.asm)
000003EC 00000153 ROM_XMEM.data ROM_XMEM.data(v22bis_mdm_xrom.asm)
000003EF 00000000 ROM_XMEM.data RXBPF22H(v22bis_mdm_xrom.asm)
0000044F 00000000 ROM_XMEM.data RXBPF22L(v22bis_mdm_xrom.asm)
000004AF 00000000 ROM_XMEM.data tx_fm_coef_low(v22bis_mdm_xrom.asm)
000004F7 00000000 ROM_XMEM.data tx_fm_coef_high(v22bis_mdm_xrom.asm)
00000540 00000000 RX_MEM.data rx_st_chg(v22bis_rxmdmmem.asm)
00000540 00000000 RX_MEM.data RXMEMB(v22bis_rxmdmmem.asm)
00000540 00000268 RX_MEM.data RX_MEM.data(v22bis_rxmdmmem.asm)
00000541 00000000 RX_MEM.data rx_ans_flg(v22bis_rxmdmmem.asm)
00000542 00000000 RX_MEM.data RX_LAPM_EN(v22bis_rxmdmmem.asm)
00000543 00000000 RX_MEM.data TX_LAPM_EN(v22bis_rxmdmmem.asm)
00000544 00000000 RX_MEM.data retctr(v22bis_rxmdmmem.asm)
00000545 00000000 RX_MEM.data rx_ctr(v22bis_rxmdmmem.asm)
00000546 00000000 RX_MEM.data err_ctr(v22bis_rxmdmmem.asm)
00000547 00000000 RX_MEM.data rx_toutctr(v22bis_rxmdmmem.asm)
00000548 00000000 RX_MEM.data RXTH(v22bis_rxmdmmem.asm)
00000549 00000000 RX_MEM.data DEL_2100(v22bis_rxmdmmem.asm)
0000054E 00000000 RX_MEM.data TON2100(v22bis_rxmdmmem.asm)
0000054F 00000000 RX_MEM.data TON150(v22bis_rxmdmmem.asm)
00000550 00000000 RX_MEM.data TONS1(v22bis_rxmdmmem.asm)
00000551 00000000 RX_MEM.data SPEED(v22bis_rxmdmmem.asm)
00000552 00000000 RX_MEM.data USB1PAT(v22bis_rxmdmmem.asm)
00000553 00000000 RX_MEM.data RETRCNT(v22bis_rxmdmmem.asm)
00000554 00000000 RX_MEM.data TRN_LNG(v22bis_rxmdmmem.asm)
00000580 00000000 RX_MEM.data IB(v22bis_rxmdmmem.asm)
000005A2 00000000 RX_MEM.data IBPTR(v22bis_rxmdmmem.asm)
000005A3 00000000 RX_MEM.data RXSB(v22bis_rxmdmmem.asm)
000005C0 00000000 RX_MEM.data RXRB(v22bis_rxmdmmem.asm)
000005F0 00000000 RX_MEM.data RXFPTR(v22bis_rxmdmmem.asm)
```

```
000005F1 00000000 RX_MEM.data DPHASE(v22bis_rxmdmmem.asm)
000005F2 00000000 RX_MEM.data CDP(v22bis_rxmdmmem.asm)
000005F3 00000000 RX_MEM.data DP(v22bis_rxmdmmem.asm)
000005F4 00000000 RX_MEM.data DPHADJ(v22bis_rxmdmmem.asm)
000005F5 00000000 RX_MEM.data BPF_OUT(v22bis_rxmdmmem.asm)
0000060D 00000000 RX_MEM.data BPFOUT_PTR(v22bis_rxmdmmem.asm)
0000060E 00000000 RX_MEM.data RXMPTR(v22bis_rxmdmmem.asm)
0000060F 00000000 RX_MEM.data RXCB2A(v22bis_rxmdmmem.asm)
00000610 00000000 RX_MEM.data RXCB2A_1(v22bis_rxmdmmem.asm)
00000611 00000000 RX_MEM.data RXCB2A_2(v22bis_rxmdmmem.asm)
00000612 00000000 RX_MEM.data RXCB2A_3(v22bis_rxmdmmem.asm)
00000613 00000000 RX_MEM.data RXCB2A_4(v22bis_rxmdmmem.asm)
00000614 00000000 RX_MEM.data RXCB2A_5(v22bis_rxmdmmem.asm)
00000615 00000000 RX_MEM.data RXCB2A_6(v22bis_rxmdmmem.asm)
00000627 00000000 RX_MEM.data RXCBPTR(v22bis_rxmdmmem.asm)
00000628 00000000 RX_MEM.data PREV_ENERGY(v22bis_rxmdmmem.asm)
00000634 00000000 RX_MEM.data PRV_ENPTR(v22bis_rxmdmmem.asm)
00000635 00000000 RX_MEM.data ENBUF_PTR(v22bis_rxmdmmem.asm)
00000636 00000000 RX_MEM.data AGCG(v22bis_rxmdmmem.asm)
00000637 00000000 RX_MEM.data AGCC1(v22bis_rxmdmmem.asm)
00000638 00000000 RX_MEM.data AGCC2(v22bis_rxmdmmem.asm)
00000639 00000000 RX_MEM.data AGCC3(v22bis_rxmdmmem.asm)
0000063A 00000000 RX_MEM.data AGCC4(v22bis_rxmdmmem.asm)
0000063B 00000000 RX_MEM.data AGCLP1(v22bis_rxmdmmem.asm)
0000063C 00000000 RX_MEM.data AGCLP2(v22bis_rxmdmmem.asm)
0000063D 00000000 RX_MEM.data AGCLG(v22bis_rxmdmmem.asm)
0000063E 00000000 RX_MEM.data RXSBAG(v22bis_rxmdmmem.asm)
0000063F 00000000 RX_MEM.data CD1(v22bis_rxmdmmem.asm)
00000640 00000000 RX_MEM.data ENERBUF(v22bis_rxmdmmem.asm)
00000680 00000000 RX_MEM.data RXCB(v22bis_rxmdmmem.asm)
00000681 00000000 RX_MEM.data RXCB_1(v22bis_rxmdmmem.asm)
00000682 00000000 RX_MEM.data RXCB_2(v22bis_rxmdmmem.asm)
00000683 00000000 RX_MEM.data RXCB_3(v22bis_rxmdmmem.asm)
00000684 00000000 RX_MEM.data RXCB_4(v22bis_rxmdmmem.asm)
00000685 00000000 RX_MEM.data RXCB_5(v22bis_rxmdmmem.asm)
00000686 00000000 RX_MEM.data RXCB_6(v22bis_rxmdmmem.asm)
0000069E 00000000 RX_MEM.data RXCBIN_PTR(v22bis_rxmdmmem.asm)
0000069F 00000000 RX_MEM.data RXCBOUT_PTR(v22bis_rxmdmmem.asm)
000006A0 00000000 RX_MEM.data CD_CNT(v22bis_rxmdmmem.asm)
000006A1 00000000 RX_MEM.data LPBAGC(v22bis_rxmdmmem.asm)
000006A2 00000000 RX_MEM.data LPBAGC2(v22bis_rxmdmmem.asm)
000006A3 00000000 RX_MEM.data HPG1(v22bis_rxmdmmem.asm)
000006A4 00000000 RX_MEM.data HPG2(v22bis_rxmdmmem.asm)
000006A5 00000000 RX_MEM.data BLPG1(v22bis_rxmdmmem.asm)
000006A6 00000000 RX_MEM.data BLPG2(v22bis_rxmdmmem.asm)
000006A7 00000000 RX_MEM.data BOFF(v22bis_rxmdmmem.asm)
000006A8 00000000 RX_MEM.data BHPX1(v22bis_rxmdmmem.asm)
```

```
000006A9 00000000 RX_MEM.data BHPY1(v22bis_rxmdmmem.asm)
000006AA 00000000 RX_MEM.data BHPX3(v22bis_rxmdmmem.asm)
000006AB 00000000 RX_MEM.data BHPY3(v22bis_rxmdmmem.asm)
000006AC 00000000 RX_MEM.data BHPE1(v22bis_rxmdmmem.asm)
000006AD 00000000 RX_MEM.data BHPE3(v22bis_rxmdmmem.asm)
000006AE 00000000 RX_MEM.data BACC1(v22bis_rxmdmmem.asm)
000006AF 00000000 RX_MEM.data BACC2(v22bis_rxmdmmem.asm)
000006B0 00000000 RX_MEM.data BLP(v22bis_rxmdmmem.asm)
000006B1 00000000 RX_MEM.data BINTG(v22bis_rxmdmmem.asm)
000006B2 00000000 RX_MEM.data BINTGA(v22bis_rxmdmmem.asm)
000006B3 00000000 RX_MEM.data status(v22bis_rxmdmmem.asm)
000006B4 00000000 RX_MEM.data CARG1(v22bis_rxmdmmem.asm)
000006B5 00000000 RX_MEM.data CARG2(v22bis_rxmdmmem.asm)
000006B6 00000000 RX_MEM.data CARG3(v22bis_rxmdmmem.asm)
000006B7 00000000 RX_MEM.data CARG4(v22bis_rxmdmmem.asm)
000006B8 00000000 RX_MEM.data COFF(v22bis_rxmdmmem.asm)
000006B9 00000000 RX_MEM.data CLP(v22bis_rxmdmmem.asm)
000006BA 00000000 RX_MEM.data RCBUF(v22bis_rxmdmmem.asm)
000006BB 00000000 RX_MEM.data RCBUF_1(v22bis_rxmdmmem.asm)
000006BC 00000000 RX_MEM.data RCBUF_2(v22bis_rxmdmmem.asm)
000006BD 00000000 RX_MEM.data RCBUF_3(v22bis_rxmdmmem.asm)
000006BE 00000000 RX_MEM.data RCBUF_4(v22bis_rxmdmmem.asm)
000006BF 00000000 RX_MEM.data RCBUF_5(v22bis_rxmdmmem.asm)
000006C0 00000000 RX_MEM.data THBUF(v22bis_rxmdmmem.asm)
000006D0 00000000 RX_MEM.data BBUF(v22bis_rxmdmmem.asm)
000006DD 00000000 RX_MEM.data JITTER(v22bis_rxmdmmem.asm)
000006DE 00000000 RX_MEM.data JITG1(v22bis_rxmdmmem.asm)
000006DF 00000000 RX_MEM.data JITG2(v22bis_rxmdmmem.asm)
000006E0 00000000 RX_MEM.data WRPFLG(v22bis_rxmdmmem.asm)
000006E1 00000000 RX_MEM.data ACODE(v22bis_rxmdmmem.asm)
000006E2 00000000 RX_MEM.data EQRT(v22bis_rxmdmmem.asm)
000006E3 00000000 RX_MEM.data EQRT_1(v22bis_rxmdmmem.asm)
000006E4 00000000 RX_MEM.data EQRT_2(v22bis_rxmdmmem.asm)
000006E5 00000000 RX_MEM.data EQRT_3(v22bis_rxmdmmem.asm)
000006E6 00000000 RX_MEM.data EQRT_4(v22bis_rxmdmmem.asm)
000006E7 00000000 RX_MEM.data EQRT_5(v22bis_rxmdmmem.asm)
000006E8 00000000 RX_MEM.data EQRT_6(v22bis_rxmdmmem.asm)
000006E9 00000000 RX_MEM.data EQRT_7(v22bis_rxmdmmem.asm)
000006EA 00000000 RX_MEM.data EQRT_8(v22bis_rxmdmmem.asm)
000006EB 00000000 RX_MEM.data EQRT_9(v22bis_rxmdmmem.asm)
000006EC 00000000 RX_MEM.data EQRT_10(v22bis_rxmdmmem.asm)
000006F1 00000000 RX_MEM.data EQIT(v22bis_rxmdmmem.asm)
00000700 00000000 RX_MEM.data EQRSB(v22bis_rxmdmmem.asm)
0000071E 00000000 RX_MEM.data EQRBIN(v22bis_rxmdmmem.asm)
0000071F 00000000 RX_MEM.data EQIBIN(v22bis_rxmdmmem.asm)
00000720 00000000 RX_MEM.data EQISB(v22bis_rxmdmmem.asm)
0000073E 00000000 RX_MEM.data EQUDSIZ(v22bis_rxmdmmem.asm)
```

```
0000073F 00000000 RX_MEM.data LUPALP(v22bis_rxmdmmem.asm)
00000740 00000000 RX_MEM.data RXSCRD(v22bis_rxmdmmem.asm)
00000741 00000000 RX_MEM.data RXODAT(v22bis_rxmdmmem.asm)
00000742 00000000 RX_MEM.data RXQ(v22bis_rxmdmmem.asm)
00000743 00000000 RX_MEM.data RXQ_1(v22bis_rxmdmmem.asm)
00000744 00000000 RX_MEM.data RXQ_2(v22bis_rxmdmmem.asm)
00000745 00000000 RX_MEM.data RXQ_3(v22bis_rxmdmmem.asm)
00000746 00000000 RX_MEM.data RXQ_4(v22bis_rxmdmmem.asm)
00000747 00000000 RX_MEM.data RXQ_5(v22bis_rxmdmmem.asm)
00000748 00000000 RX_MEM.data RXQ_6(v22bis_rxmdmmem.asm)
00000749 00000000 RX_MEM.data RXQ_7(v22bis_rxmdmmem.asm)
0000074A 00000000 RX_MEM.data RXQ_8(v22bis_rxmdmmem.asm)
0000074B 00000000 RX_MEM.data RXQ_9(v22bis_rxmdmmem.asm)
0000074C 00000000 RX_MEM.data RXQ_10(v22bis_rxmdmmem.asm)
0000074D 00000000 RX_MEM.data RXQ_11(v22bis_rxmdmmem.asm)
0000074E 00000000 RX_MEM.data RXQ_12(v22bis_rxmdmmem.asm)
0000074F 00000000 RX_MEM.data RXQ_13(v22bis_rxmdmmem.asm)
00000750 00000000 RX_MEM.data RXQ_14(v22bis_rxmdmmem.asm)
00000751 00000000 RX_MEM.data RXQ_15(v22bis_rxmdmmem.asm)
0000075B 00000000 RX_MEM.data RxQ_ptr(v22bis_rxmdmmem.asm)
0000075C 00000000 RX_MEM.data Rx_StQC(v22bis_rxmdmmem.asm)
00000762 00000000 RX_MEM.data Rx_StQA(v22bis_rxmdmmem.asm)
00000767 00000000 RX_MEM.data Rx_StQG22(v22bis_rxmdmmem.asm)
0000076C 00000000 RX_MEM.data Rx_StQGBis(v22bis_rxmdmmem.asm)
00000774 00000000 RX_MEM.data Rx_StQ_ptr(v22bis_rxmdmmem.asm)
00000775 00000000 RX_MEM.data RXMASK_MC(v22bis_rxmdmmem.asm)
00000776 00000000 RX_MEM.data RXMASK(v22bis_rxmdmmem.asm)
00000777 00000000 RX_MEM.data TXMASK_MC(v22bis_rxmdmmem.asm)
00000778 00000000 RX_MEM.data TXMASK(v22bis_rxmdmmem.asm)
00000779 00000000 RX_MEM.data RN_BITS_BAUD(v22bis_rxmdmmem.asm)
0000077A 00000000 RX_MEM.data TN_BITS_BAUD(v22bis_rxmdmmem.asm)
0000077B 00000000 RX_MEM.data T401_VALUE(v22bis_rxmdmmem.asm)
0000077C 00000000 RX_MEM.data T401B_VALUE(v22bis_rxmdmmem.asm)
0000077D 00000000 RX_MEM.data T403_VALUE(v22bis_rxmdmmem.asm)
0000077E 00000000 RX_MEM.data LASTDP(v22bis_rxmdmmem.asm)
0000077F 00000000 RX_MEM.data WRAP(v22bis_rxmdmmem.asm)
00000780 00000000 RX_MEM.data BBUFPTR(v22bis_rxmdmmem.asm)
00000781 00000000 RX_MEM.data rx_data(v22bis_rxmdmmem.asm)
00000781 00000000 RX_MEM.data Frx_data(v22bis_rxmdmmem.asm)
00000782 00000000 RX_MEM.data NOISE(v22bis_rxmdmmem.asm)
00000783 00000000 RX_MEM.data RETCNT_RM(v22bis_rxmdmmem.asm)
00000784 00000000 RX_MEM.data speed(v22bis_rxmdmmem.asm)
00000785 00000000 RX_MEM.data ICOEFF(v22bis_rxmdmmem.asm)
00000791 00000000 RX_MEM.data BPF_PTR(v22bis_rxmdmmem.asm)
00000792 00000000 RX_MEM.data temp1(v22bis_rxmdmmem.asm)
00000793 00000000 RX_MEM.data temp2(v22bis_rxmdmmem.asm)
00000794 00000000 RX_MEM.data mod_tbl_offset(v22bis_rxmdmmem.asm)
```

```
  00000795 00000000 RX_MEM.data TRAINING(v22bis_rxmdmmem.asm)
  00000796 00000000 RX_MEM.data IFBANK(v22bis_rxmdmmem.asm)
  00000797 00000000 RX_MEM.data IBCNT(v22bis_rxmdmmem.asm)
  00000798 00000000 RX_MEM.data TXBD_CNT(v22bis_rxmdmmem.asm)
  00000799 00000000 RX_MEM.data TNSUM(v22bis_rxmdmmem.asm)
  0000079A 00000000 RX_MEM.data TNASUM(v22bis_rxmdmmem.asm)
  0000079B 00000000 RX_MEM.data EQX(v22bis_rxmdmmem.asm)
  0000079C 00000000 RX_MEM.data EQY(v22bis_rxmdmmem.asm)
  0000079D 00000000 RX_MEM.data RXDATA(v22bis_rxmdmmem.asm)
  0000079E 00000000 RX_MEM.data DECX(v22bis_rxmdmmem.asm)
  0000079F 00000000 RX_MEM.data DECY(v22bis_rxmdmmem.asm)
  000007A0 00000000 RX_MEM.data DX(v22bis_rxmdmmem.asm)
  000007A1 00000000 RX_MEM.data DY(v22bis_rxmdmmem.asm)
  000007A2 00000000 RX_MEM.data dscr_mask(v22bis_rxmdmmem.asm)
  000007A3 00000000 RX_MEM.data rx_dscr_buff(v22bis_rxmdmmem.asm)
  000007A4 00000000 RX_MEM.data rx_dscr_buff_1(v22bis_rxmdmmem.asm)
  000007A5 00000000 RX_MEM.data dscr_cntr(v22bis_rxmdmmem.asm)
  000007A6 00000000 RX_MEM.data RXMEMSIZE(v22bis_rxmdmmem.asm)
  000007A7 00000000 RX_MEM.data IBPTR_IN(v22bis_rxmdmmem.asm)
  000007A8 0000001D v21_xrom.data v21_xrom.data(v21_rom.asm)
  000007A8 00000000 v21_xrom.data LPF_COEF(v21_rom.asm)
  000007B4 00000000 v21_xrom.data NBY16_TABLE(v21_rom.asm)
  00000800 00000000 v21_xrom1.data SINE_TABLE2(v21_rom.asm)
  00000800 00000090 v21_xrom1.data v21_xrom1.data(v21_rom.asm)
  00000A00 00000000 v21_xrom2.data SINE_TABLE1(v21_rom.asm)
  00000A00 00000168 v21_xrom2.data v21_xrom2.data(v21_rom.asm)
#>00000000          _EX_BIT (linker command file)
#>00000001          _NUM_IM_PARTITIONS (linker command file)
#>00000000          _NUM_EM_PARTITIONS (linker command file)
#>00000B68          FmemEXbit (linker command file)
#>00000B69          FmemNumIMpartitions (linker command file)
#>00000B6A          FmemNumEMpartitions (linker command file)
#>00000B6C          FmemIMpartitionAddr (linker command file)
#>00000B6D          FmemIMpartitionSize (linker command file)
#>00000000          FmemEMpartitionAddr (linker command file)
#>00000B6E          FmemEMpartitionSize (linker command file)
#>00000B70          __xRAM_data_end (linker command file)
#>00000B70          __data_size (linker command file)

# .ApplicationData
#>00000B70          F_Xbss_start_addr (linker command file)
#>00000B70          _START_BSS (linker command file)
  00000B70 00000001 .bss    FSR_lock(Cpu.c)
  00000B71 00000001 .bss    FSR_reg(Cpu.c)
  00000B72 00000001 .bss    FAD1_ModeFlg(AD1.c)
  00000B73 00000001 .bss    FOutFlg(AD1.c)
  00000B73 00000001 .bss    FAD1_EnUser(AD1.c)
```

```
00000B74 00000032 .bss    FOutBuffer(AS1.c)
  00000BA6 00000001 .bss    FOutPtrW(AS1.c)
  00000BA7 00000001 .bss    FOutPtrR(AS1.c)
  00000BA8 00000001 .bss    FOutLen(AS1.c)
  00000BA9 00000032 .bss    FInpBuffer(AS1.c)
  00000BDB 00000001 .bss    FInpPtrW(AS1.c)
  00000BDC 00000001 .bss    FInpPtrR(AS1.c)
  00000BDD 00000001 .bss    FInpLen(AS1.c)
  00000BDE 00000001 .bss    FErrFlag(AS1.c)
  00000BDF 00000001 .bss    FSerFlag(AS1.c)
  00000BE0 00000001 .bss    FEnUser(TenthSecInt.c)
  00000BE1 00000001 .bss    Fv22dibittxcount(v22bisapi.c)
  00000BE2 00000001 .bss    Frxbitcounter(v22bisapi.c)
  00000BE3 00000001 .bss    Fstartbit(v22bisapi.c)
  00000BE4 00000001 .bss    FPartialRxByte(v22bisapi.c)
  00000BE5 00000003 .bss    FNibbles(v22bisapi.c)
  00000BE8 00000001 .bss    FNibbleCount(v22bisapi.c)
00000BE9 00000001 .bss    FNumberBytes(v22bisapi.c)
  00000BEA 00000001 .bss    FBytePtr(v22bisapi.c)
  00000BEB 00000001 .bss    FByteCount(v22bisapi.c)
  00000BEC 00000001 .bss    Fmessageover(v22bisapi.c)
  00000BEE 00000004 .bss    FRXCallback(v22bisapi.c)
  00000BF2 00000004 .bss    FTXCallback(v22bisapi.c)
  00000BF6 00000001 .bss    FbMemInitialized(mem.c)
  00000BF8 00000004 .bss    FEmptyExternalMemoryPool(mem.c)
  00000BFC 00000004 .bss    FEmptyInternalMemoryPool(mem.c)
  00000C00 00000005 .bss    FInitialState(mem.c)
  00000C06 00000028 .bss    FExternalMemoryPool(mem.c)
  00000C2E 00000028 .bss    FInternalMemoryPool(mem.c)
  00000C56 00000001 .bss    Fring_state(Events.c)
  00000C57 00000001 .bss    Ferror_cnt(Events.c)
  00000C58 00000001 .bss    Fring_pulse_count(Events.c)
  00000C5A 00000002 .bss    FCombinedAnalogRxSample(Events.c)
00000C5C 00000001 .bss    FRxSampleCount(Events.c)
  00000C5D 00000001 .bss    FxresetC(Events.c)
  00000C5E 00000001 .bss    Fthe_errors(Events.c)
  00000C5F 00000001 .bss    FCountTime(Events.c)
  00000C60 00000001 .bss    Fis_time_to_shake(modem.c)
  00000C61 00000001 .bss    FPreviousSample(modem.c)
  00000C62 00000001 .bss    FCumCnt(modem.c)
  00000C63 00000001 .bss    FSaveCnt(modem.c)
  00000C64 00000001 .bss    FStartBitRXd(modem.c)
  00000C65 00000001 .bss    FMsgByteRx1(modem.c)
  00000C66 0000000C .bss    FConfigV21(modem.c)
  00000C72 00000001 .bss    FpV21(modem.c)
  00000C73 00000001 .bss    Fstate_of_escape(modem.c)
  00000C74 00000001 .bss    FAT_on_state(modem.c)
```

```
00000C74 00000001 .bss    FAT_off_state(modem.c)
  00000C75 00000015 .bss    Fphone_number(modem.c)
  00000C89 00000001 .bss    Fp_phone_number(modem.c)
  00000C8A 00000001 .bss    Fh_parm(modem.c)
  00000C8C 00000002 .bss    FAC24(modem.c)
  00000C8E 00000002 .bss    FAC12(modem.c)
  00000C90 00000002 .bss    FAC0(modem.c)
  00000C92 00000002 .bss    FACa(modem.c)
  00000C94 00000001 .bss    FLine_Tones(modem.c)
  00000C95 00000001 .bss    FpAnalogTxRead(modem.c)
  00000C96 00000001 .bss    FpAnalogTxWrite(modem.c)
  00000C97 00000080 .bss    FAnalogTxBuffer(modem.c)
  00000D17 00000001 .bss    FpAnalogRxRead(modem.c)
  00000D18 00000001 .bss    FpAnalogRxWrite(modem.c)
  00000D19 00000020 .bss    FAnalogRxBuffer(modem.c)
  00000D39 00000008 .bss    FModemRxBuffer(modem.c)
  00000D41 00000001 .bss    Ftty_in_status(modem.c)
00000D42 00000032 .bss    Fmodem_in(modem.c)
  00000D74 00000032 .bss    Ftty_in(modem.c)
  00000DA6 00000064 .bss    FCodecRxBuffer(modem.c)
  00000E0A 00000064 .bss    FCodecTxBuffer(modem.c)
  00000E6E 00000001 .bss    FAns_Tone_Detect(modem.c)
  00000E6F 00000001 .bss    FAns_Tone_Start(modem.c)
  00000E6F 00000001 .bss    FV21_Mode(modem.c)
  00000E70 00000001 .bss    FAT_q_flag(modem.c)
  00000E70 00000001 .bss    FAT_z_flag(modem.c)
  00000E71 00000001 .bss    Fcall_phone_number(modem.c)
  00000E71 00000001 .bss    FCaller_Modem(modem.c)
  00000E72 00000001 .bss    FConnecting_v21(modem.c)
  00000E72 00000004 .bss    Fv22bis_RXCallback(modem.c)
  00000E76 00000004 .bss    Fv22bis_TXCallback(modem.c)
  00000E7A 00000001 .bss    Frate_negotiated(modem.c)
  00000E7B 00000001 .bss    Fconnection_lost(modem.c)
  00000E7C 00000001 .bss    Fv22bis_connection_established(modem.c)
00000E7D 0000000E API.bss API.bss(v22bis_apimem.asm)
  00000E7D 00000000 API.bss s_ctr(v22bis_apimem.asm)
  00000E7E 00000000 API.bss Tx_Baud_Count(v22bis_apimem.asm)
  00000E7F 00000000 API.bss Rx_Baud_Flg(v22bis_apimem.asm)
  00000E80 00000000 API.bss TIME_CNT(v22bis_apimem.asm)
  00000E81 00000000 API.bss TIME_CNTL(v22bis_apimem.asm)
  00000E82 00000000 API.bss TIME_CNTH(v22bis_apimem.asm)
  00000E83 00000000 API.bss in_data_ptr(v22bis_apimem.asm)
  00000E84 00000000 API.bss txrx_status(v22bis_apimem.asm)
  00000E85 00000000 API.bss WordWrFlg(v22bis_apimem.asm)
  00000E86 00000000 API.bss WordRdFlg(v22bis_apimem.asm)
  00000E87 00000000 API.bss StartCompare(v22bis_apimem.asm)
  00000E88 00000000 API.bss SyncWord_mem(v22bis_apimem.asm)
```

```
00000E89 00000000 API.bss Sync_sent_status(v22bis_apimem.asm)
  00000E8A 00000000 API.bss SyncWord_rx(v22bis_apimem.asm)
#>00000E8B          _END_BSS (linker command file)
#>0000031B          F_Xbss_length (linker command file)
#>00000E8C          _HEAP_ADDR (linker command file)
#>00000100          _HEAP_SIZE (linker command file)
#>00000F8C          _HEAP_END (linker command file)
#>00000800          _min_stack_size (linker command file)
#>00000F8C          _stack_addr (linker command file)
#>0000178C          _stack_end (linker command file)
#>00000E8C          F_heap_addr (linker command file)
#>00000F8C          F_heap_end (linker command file)
#>00000F8C          F_Lstack_addr (linker command file)
#>00000F8C          F_StackAddr (linker command file)
#>0000178B          F_StackEndAddr (linker command file)
#>00000B70          F_Ldata_size (linker command file)
#>00000000          F_Ldata_RAM_addr (linker command file)
#>00003D7E          F_Ldata_ROM_addr (linker command file)
#>00000000          F_xROM_to_xRAM (linker command file)
#>00000001          F_pROM_to_xRAM (linker command file)
#>00000B70          F_start_bss (linker command file)
#>00000E8B          F_end_bss (linker command file)
#>0000178C          __DATA_END (linker command file)
#>0000F020          FArchIO (linker command file)
```

4.5 Core Processor Loading and RTOS

Since the modem uses almost none of the core processor's resources, an RTOS may be used to run several tasks along with the modem task.

4.5.1 Core Processor Load

When the modem is idle, waiting for a call, it consumes almost no MIPS, because the hardware is used to count ring pulses without the help of the core. The core is interrupted only when a significant ring is detected.

The *V.22bis* bean methods with a line rate of 2400 characters per second are called to recieve data for a consumption rate of 6.21 MIPS. The *V.22bis* bean methods for transmission of data consume .94 MIPS, so the total MIPS for V.22bis is only 7.15. This is merely a small fraction of the MIPS available on the 60 MIP controller, just over 10 percent.

The fraction consumed is even less when V.21, with a line rate of 300 characters per second is used. The V.21 receiver takes only 1.2516 MIPS. The transmitter takes even less at .1134 MIPS, or much less than one percent of the MIPS available. The total MIPS for V.21 adds up to only 1.364 MIPS, or about 2 percent of the processor's available MIPS.

4.5.2 Using RTOS to Run the Modem Concurrently with Other Tasks

In order to blend other tasks with the modem, it is advisable to use a real-time operating system (RTOS). A multi-tasking, context-switching RTOS could be used to share the remaining resouces with other tasks, such as alarm monitoring or process control. The modem code itself would be one task, since it is written as one thread.

There are several ready sources for RTOS, including Metrowerks, Unicoi, and Micrium. Details about Micrium's RTOS may be found at: **http://www.ucos-ii.com/**. CodeWarrior includes a demonstration project showing how this RTOS is to be used. Once the RTOS files are obtained from Micrium, they are simply inserted into the code directory, and this demostration project may be run and applied to the modem code.

With the RTOS, the modem code would be made into a task, which would sleep while it is waiting for IO to complete, freeing and unlocking the CPU for other tasks.

4.6 Peripheral Footprint

The total resource utilization is summarized in **Figure 4-13**. This resource load meter is a convient feature of CodeWarrior with Processor Expert. As the project is developed, resource utilization is easily tracked as beans are added.

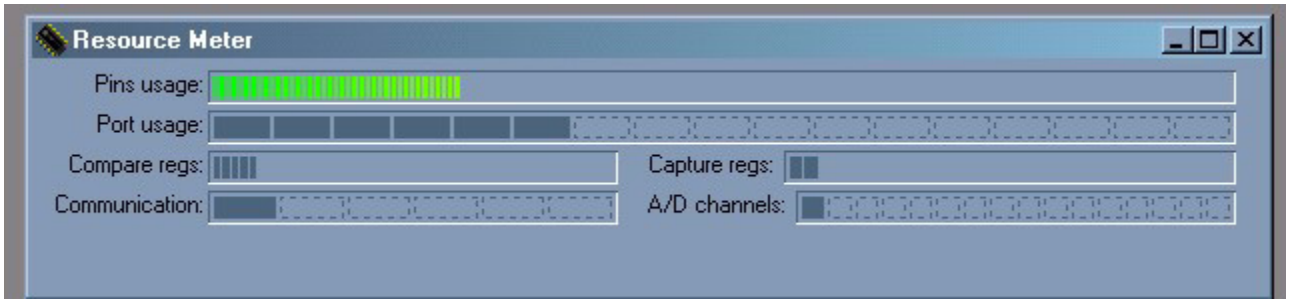


Figure 4-13. Soft Modem Peripheral Footprint

The small percentage of peripherals used by the modem is depicted in **Figure 4-14**. This also shows how the beans are grapically associated with pins on the device. Pins without such associations are free for use in other tasks. Each bean has a unique grapic icon that is easily associated with the pins in **Figure 4-14**.

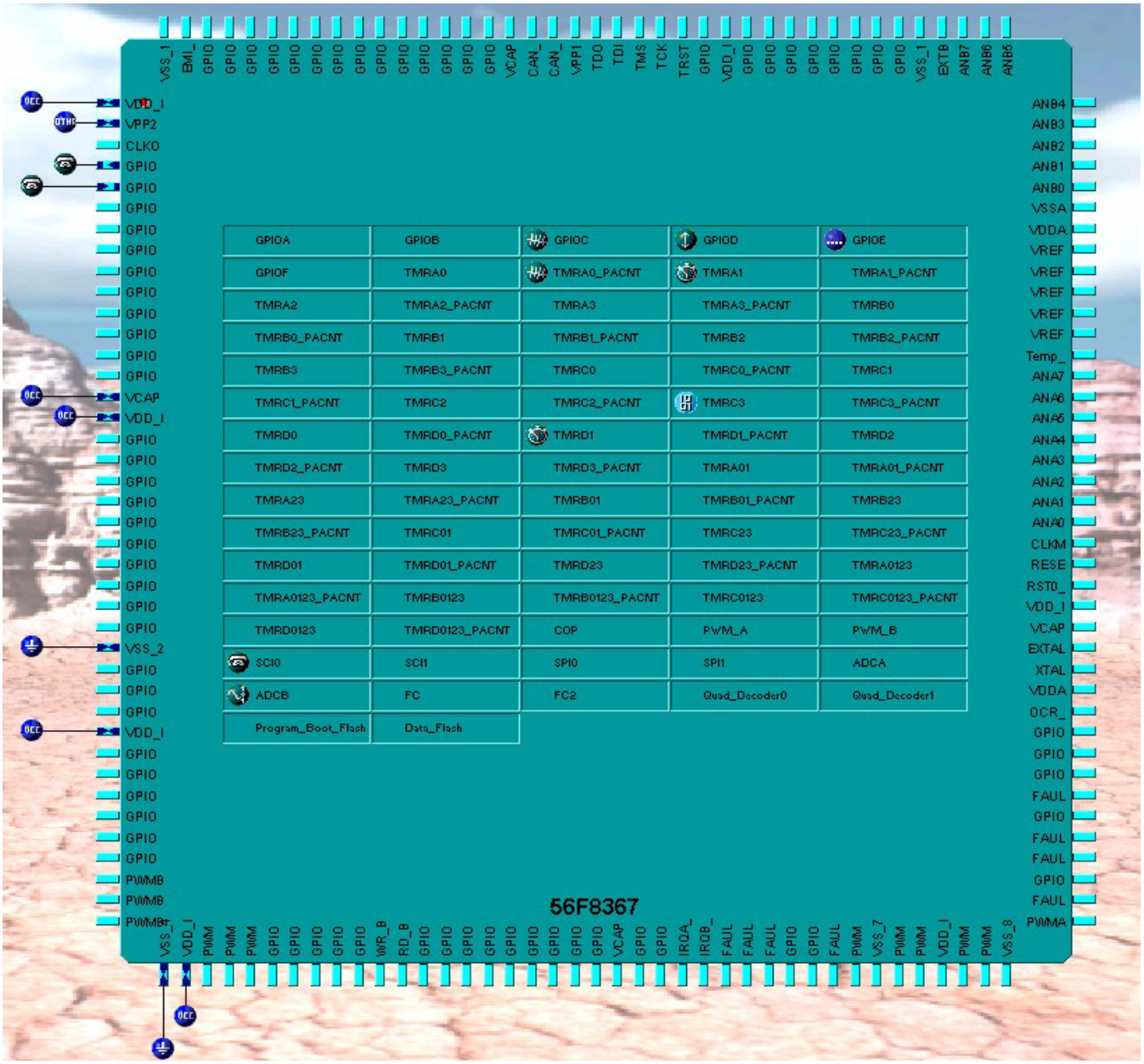


Figure 4-14. Soft Modem Pin Utilization

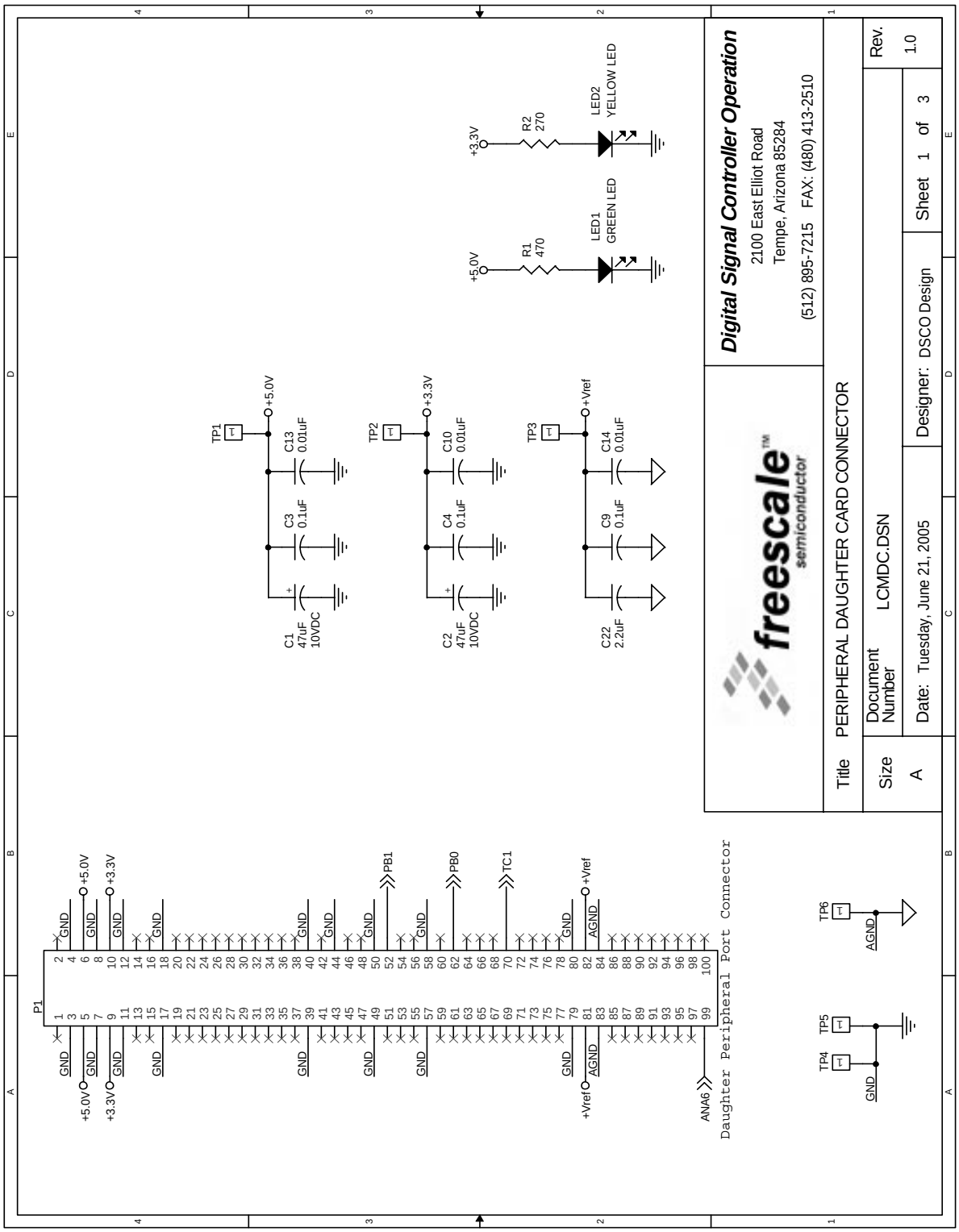
4.7 Conclusions

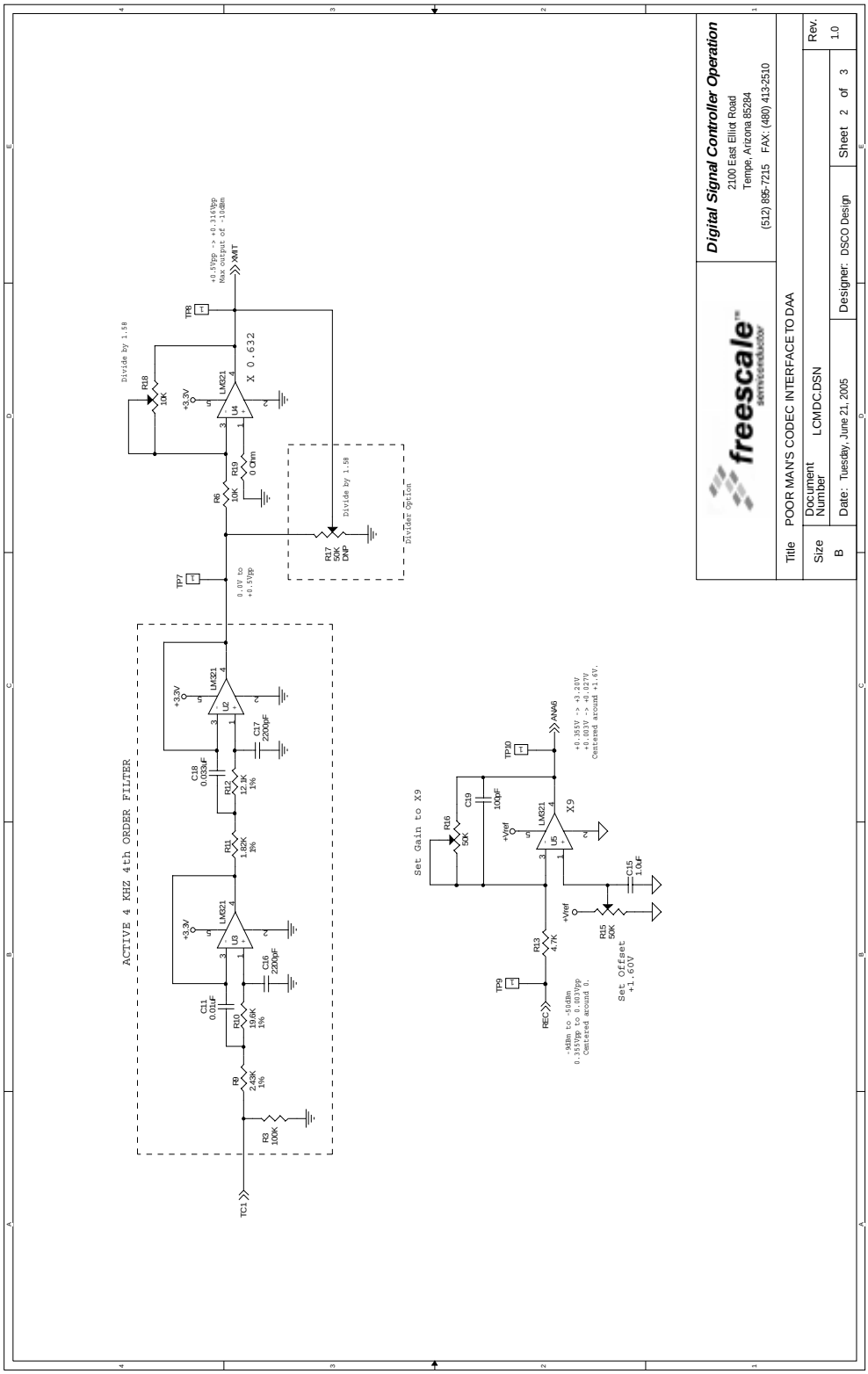
The soft modem developed is suitable for incorporation into commercial products requiring communication over the Public Switched Telephone Network (PSTN) at speeds up to 2400 bits per second. A traditional telephone codec is not required in the design, resulting in a one-chip/one-core system capable of a complete mission, including communications functions.

Both V.21 and V.22bis/V.22 are supported. The V.22bis falls back to V.22 when noise dictates.

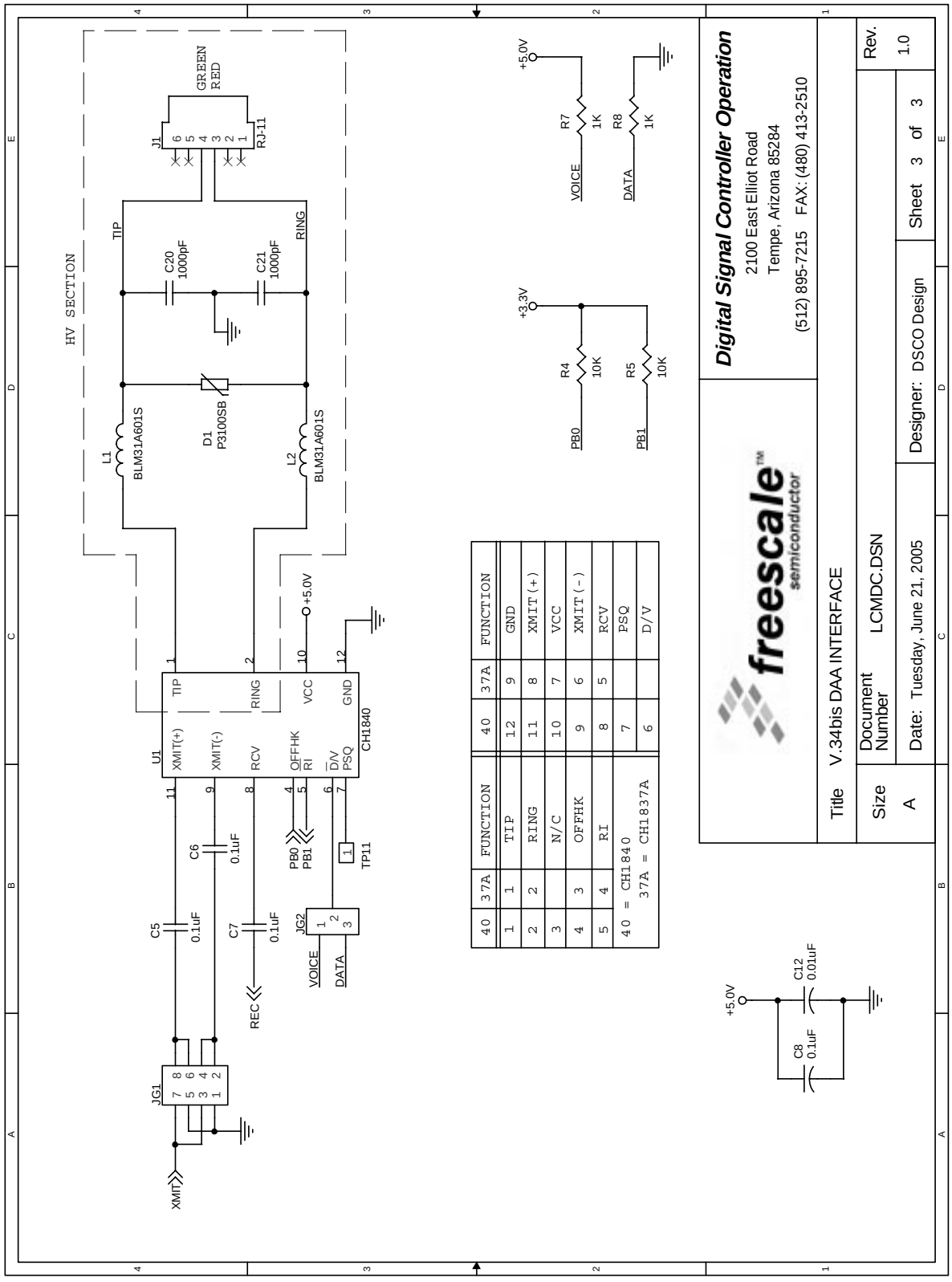
The modem is easily added to projects developed for the Freescale 56F8300 family.

Appendix A
Schematics


Digital Signal Controller Operation
2100 East Elliot Road
Tempe, Arizona 85284
(512) 895-7215 FAX: (480) 413-2510
freescale semiconductor
Title PERIPHERAL DAUGHTER CARD CONNECTOR
Size A
Document Number LCMDC.DSN
Rev. 1.0
Date: Tuesday, June 21, 2005
Designer: DSCO Design
Sheet 1 of 3
P1
Daughter Peripheral Port Connector
LED1 GREEN LED
LED2 YELLOW LED
R1 470
R2 270
C1 47uF 10VDC
C2 47uF 10VDC
C22 2.2uF
C3 0.1uF
C4 0.1uF
C9 0.1uF
C10 0.01uF
C13 0.01uF
C14 0.01uF
TP1
TP2
TP3
TP4
TP5
TP6
+5.0V
+3.3V
+Vref
GND
AGND
PB1
PB0
TC1
ANA6



Schematics, Rev. 0


HV SECTION
TIP
RING
C20 1000pF
C21 1000pF
L1 BLM31A601S
L2 BLM31A601S
D1 P3100SB
J1 RJ-11
GREEN
RED
U1 CH1840
XMIT(+)
XMIT(-)
RCV
OFFHK
RI
D/V
PSQ
VCC
GND
C5 0.1uF
C6 0.1uF
C7 0.1uF
JG1
JG2
XMIT
REC
VOICE
DATA
PB0
PB1
TP11
+5.0V
+3.3V
R4 10K
R5 10K
R7 1K
R8 1K
C8 0.1uF
C12 0.01uF
40 = CH1840
37A = CH1837A
FUNCTION
TIP
RING
N/C
OFFHK
RI
GND
XMIT(+)
XMIT(-)
VCC
RCV
PSQ
D/V
Digital Signal Controller Operation
2100 East Elliot Road
Tempe, Arizona 85284
(512) 895-7215 FAX: (480) 413-2510
freescale semiconductor
Title V.34bis DAA INTERFACE
Size A
Document Number LCMDC.DSN
Date: Tuesday, June 21, 2005
Designer: DSCO Design
Sheet 3 of 3
Rev. 1.0

Appendix B
Low-Cost Soft Modem Bill of Materials

Quantity	Reference	Part	Tolerance	Package
2	C1, C2	47uF	10V DC	ELECT-C
7	C3—C9	0.1uF		805
5	C10—C14	0.01uF		805
1	C15	1.0uF		805
2	C17, C16	2200pF		805
1	C18	0.033uF		805
1	C19	100pF		805
2	C20, C21	1000pF		1812
1	C22	2.2uF		805
1	D1	P3100SB		SOD-6
1	JG1	HEADER 4x2		BERG-4x2
1	JG2	HEADER 3x1		BERG-3x1
1	J1	RJ-11		RJ11
1	LED1	GREEN LED		HPLED
1	LED2	YELLOW LED		HPLED
2	L1, L2	BLM31A601S		1206
1	P1	HRS FX6-100S-0.8SV2		HRS_FX6-100S-0.8SV2
1	R1	470		805
1	R2	270		805
1	R3	100K		805
3	R4—R6	10K		805
2	R7, R8	1K		805
1	R9	2.43K	1%	805
1	R10	19.6K	1%	805

Quantity	Reference	Part	Tolerance	Package
1	R11	1.82K	1%	805
1	R12	12.1K	1%	805
1	R13	4.7K		805
2	R16, R15	50K		CT94W
1	R17	50K	DNP	CT94W
1	R18	10K		CT94W
1	R19	0 Ohm		805
10	TP1—TP10	TP		BERG_1x1
1	TP11	TP		BERG-1x1
1	U1	CH1840		CH180-TH12
4	U2—U5	LM321		LM321_SOT-23-5

Appendix C
Layout and Governmental Certifications

C.1 Design for Performance

Layout considerations for the PWM are easy. Since it is a digital signal, it can tolerate considerable noise, and layout considerations are few. The ADC however, is an analog input to the controller. It should not parallel at close range signals containing clocks or signals that change often. Stripmine-type shielding could be considered. Please refer to Freescale FAQs and application notes relating to optimal use of the ADC. The layout of the EVM may be used as an example, even though it is possible to reduce the noise floor even further. Given the dynamic range available, this is not required to obtain the performance documented in this note.

C.2 Design for Agency Approvals

The final step in bringing a product containing a soft modem to market involves obtaining the approval of and certification by the various government agencies regulating the sale of products that are to be connected to the PSTN. In some countries, the same agency that regulates the post office also regulates the modem product industry. In the United States, FCC part 15 and part 68 should be met and UL approval of any electrical appliance is advised.

The government is concerned with several factors:

- **The ability of the equipment to operate in the presense of radio frequency interference**
  Governments dislike receiving numerous complaints about radio frequency interference. The more a product can be exposed to radio frequency interference without faltering, the better the government's view of the product.
- **The amount of radio frequency interference produced by the product**
  Certain frequencies are used by government agencies; these frequecies are monitored most closely for compliance. For example, 75 megahertz is allocated to aeronautical radionavigation. If a product broadcasts on that frequency, planes could be endangered.
- **The effect of high voltages from tip and ring (including lightning strikes) on the product's viability and safety**
  In the interest of consumer safety and product salability, the product should not be destroyed, and any damage should be limited, when lightning or other high-voltage sources come down the tip and ring from the phone pole. The idea is to limit the damage to the parts of the product called the Data Access Arrangement (DAA), when lightning strikes, or when power lines become tangled with phone lines. The ring voltage itself is considered high voltage and dangerous.
- **The propensity of the product to become a nuisance by repeatedly calling wrong numbers in the middle of the night to private homes**
  With the advent of the FAX machine and computer bulletin board, a disturbing trend began. People were called repeatedly by modems or FAX machines. Some countries require "blacklisting" certain numbers and limiting the number of times other numbers may be called per unit time. In order to comply with this requirement, firmware must be tested by government agencies or their designated agents.

One of the factors simply requires software that cannot automatically and repeatedly dial phone numbers. The radio frequency issues are dealt with by shielding (u metal) and/or minimizing current loop size. Current loop minimization is a layout technique where the open area of a current loop is minimized.

The other factors can be dealt with by selecting a DAA that meets the standards of multiple governments, such as the one selected for this design. Additionally, the area of the product's PC board where this DAA device is mounted requires special layout attention.

The layout should minimize the area of current loops, or enclose them in shielding, and isolate the high voltage section of the DAA on a one- or two-layer section of the board, well fused from the tip and ring, and located at the extreme boundary of the product.

The current loop area is minimized by running the return wire from any circuit next to, or above or below the source wire for the circuit. It is also good to surround these "wires" or traces with ground plane on as many sides as possible.

NOTE: This *reference* design has not been examined by government authorities for compliance, but is only used on equipment that simulates the PSTN. Any products developed to be sold to the public would require the respective government agency approvals prior to sale in the respective country and use on the PSTN.

INDEX

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

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

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

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

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

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

DRM073
Rev. 0
07/2005