599 Menlo Drive, Suite 100
Rocklin, California 95765, USA
Office: (916) 624-8333
Fax: (916) 624-8003

General: info@parallaxinc.com
Technical: stamptech@parallaxinc.com
Web Site: www.parallaxinc.com
Educational: www.stampsinclass.com

Stamp Modem AppMod (#29116)

Cermetek CH1786 2400 BPS Modem Board

Figure #1: Stamp Modem on BOE Rev. B
Stamp Modem mounted on the Parallax Board of Education (#28150).[1]

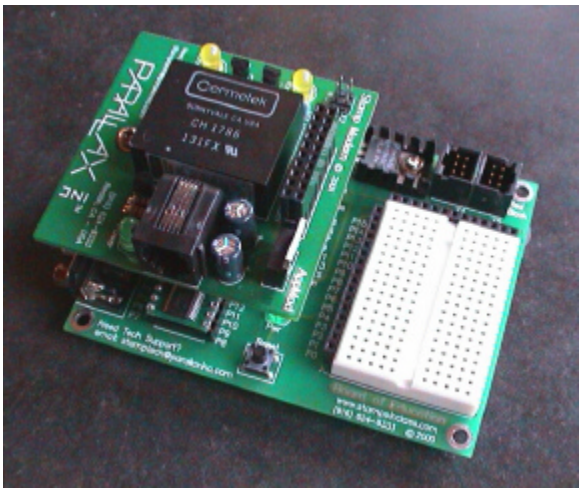
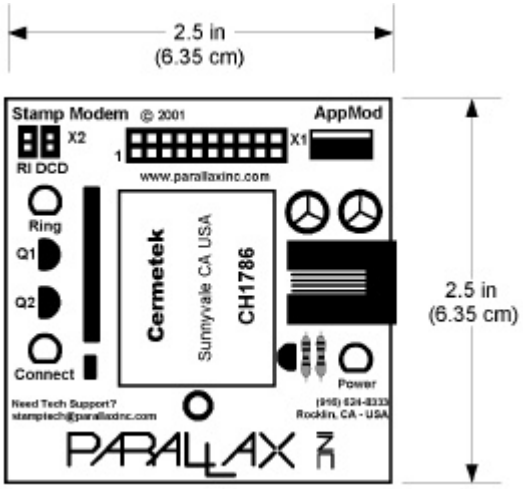


Figure #2: Dimensions
Stamp Modem board is 2.5" x 2.5".



Features

The Stamp Modem provides a functional and easy-to-use modem interface to a BASIC Stamp. The Stamp Modem is based on the Cermetek CH1786, an FCC part 68 pre-approved modem with a V.22 bis full "AT" command set. Stamp Modem is connected to Parallax boards on the 2x10 "AppMod" connector and secured with a screw and standoff. Power, ground and BASIC Stamp I/O connections are conveniently connected to the BASIC Stamp through this header.

The Stamp Modem could be utilized in several ways:

- PC modem with HyperTerminal[2] calls Stamp Modem to transmit/receive data
- Stamp Modem calls Stamp Modem so BASIC Stamps may communicate with each other
- Stamp Modem calls PC modem to transmit/receive data

[1] This AppMod is also compatible with the Super Carrier (#27130) and the BASIC Stamp Activity Board (#27905).
[2] Other terminal emulation programs (i.e., Procomm) are also suitable for this project.

Packing List

The Stamp Modem AppMod (#29116) package should include the following parts (the source code included in this documentation is only available for download from www.parallaxinc.com/stampmodem):

- Documentation (these pages)
- Stamp Modem AppMod with (2) mini jumpers installed
- 2 x 10 female-male connector
- 3/4" male-female hex standoff, 4/40 x 3/8" screw and nut

Optionally, you could have also purchased the Stamp Modem Plus Pack (#29117), a small assortment of components used in these examples. The Stamp Modem Plus Pack includes:

- (1) red and (1) yellow LED
- (2) 470 ohm resistors
- (1) Dallas Semiconductor 1620 Digital Thermometer
- (1) 1K ohm resistor
- (1) 0.1 uF capacitor
- (1) pack of 10 jumper wires

Hardware Description

Refer to the diagram in Figure 2 and the schematic in Figure 3. The Stamp Modem has the following hardware components:

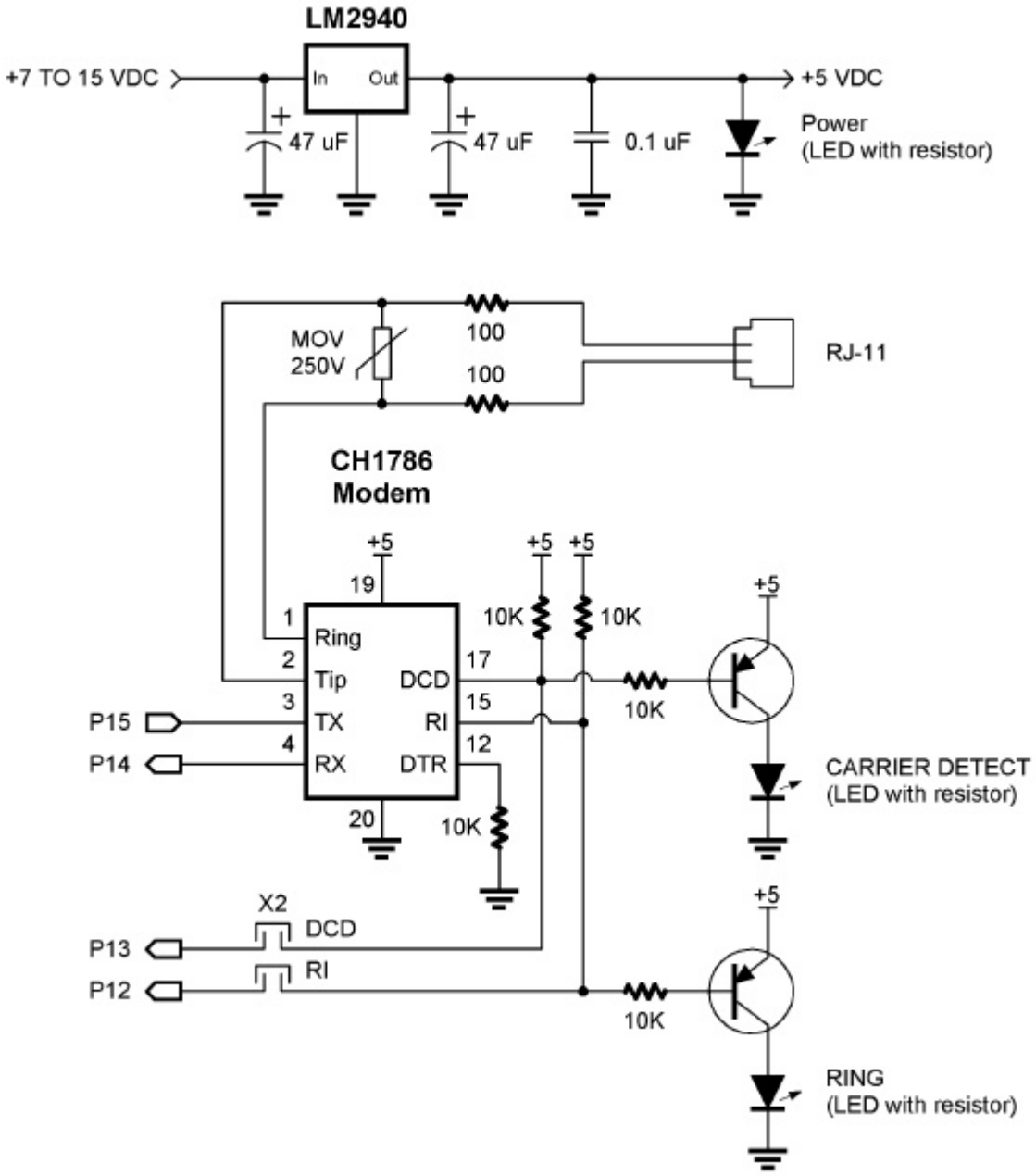
- Cermetek Modem CH1786 embedded modem. Datasheet is available at www.cermetek.com.
- X1 AppMod Connector This bus is used to interface the Stamp Modem to Parallax Boards. The pinout for the connector is printed on each Parallax board.
- X2 Jumper Block The two mini jumpers are used to connect/disconnect the Ring Indicator (RI) and Data Carrier Detect (DCD) modem signals. When the RI jumper is installed, the RI signal is connected to I/O pin 12. When the DCD jumper is installed, the DCD signal is connected to I/O pin 13. These features may be used as necessary or not used to save two BASIC Stamp I/O pins for other purposes.
- Power LED This LED illuminates when power is applied to the Stamp Modem.
- Ring LED Illuminates when the phone line connected to the modem rings.
- Connect LED Illuminates when the modem goes off hook (answers) and synchronizes with the remote modem.

The Cermetek CH1786 modular modem, like most modems, has a number of configuration registers inside the modem to configure its functionality. These configuration registers are user changeable. Please refer to the CH1786 datasheet for more information.

Schematic Explanation

The 250 V metal oxide varistor protects the modem from high voltages that could be introduced on the RJ11 connected telephone line through surges or lighting. The two 10 Ohm resistors (R6 & R7) are for over-current protection.

Figure 3: Stamp Modem Schematic



There are five (5) modem control lines: RX, TX, DTR, DCD, and RI. The Stamp Modem uses four (4) of the control lines:

- Data Terminal Ready (DTR) is a control line to the modem; when low indicates to the modem that the remote serial device is ready to communicate. This input to the modem is tied low to enable it all the time and is not available for BASIC Stamp control.
- Ring Indicator (RI) This signal pulses low when the telephone line connected to the modem rings. The LED will flash when an incoming call rings the modem line. This signal can optionally be read through BASIC Stamp I/O pin 12 when the RI jumper is installed on the X2 jumper block.

- Data Carrier Detect (DCD) This signal goes low after the modem answers the call (goes OFF HOOK) and connects with the calling modem. The LED will light when this signal goes low (CONNECTED). This signal is connected through jumper pins to the Stamp I/O pin 15. This signal is normally used to indicate to the Stamp that the modem has connected to a remote modem. This signal is used in the example program presented in this document.
- Transmit Data (TX) is the serial output line from the BASIC Stamp to the modem. This line carries serial data from the Stamp to the modem for communications with the modem, or data output to the remote connected modem.
- Receive Data (RX) is the serial input line from the modem to the BASIC Stamp I/O pin 14. This line carries serial data from the modem to the BASIC Stamp. This serial data may be modem responses to commands or serial data received from a remote connected modem.

Setting Up

Follow these steps to setup the Stamp Modem on a Board of Education Rev. B:

1. Verify that no power is supplied to the Board of Education.
2. Plug the Stamp Modem into the X1 connector. Carefully align the pins.
3. Using the standoff, screw and nut, secure the Stamp Modem to the Board of Education.

Two examples are shown on the following pages. Source code for each of these projects is available for download from www.parallaxinc.com/stampmodem. Note: These examples assume you have at least some working knowledge of HyperTerminal or other terminal software.

Federal Communications Commission Notice

FCC Registration
All products in the CH1786 family are registered with the FCC (Federal Communications Commission) under Part 68. To maintain the validity of the registration, you must serve notice to the end user of the products of several restrictions the FCC places on the modem and its use. In addition to restriction notification, the FCC requires that Cermetek make all repairs to all products in the CH1786 family. If repairs are necessary after installation of the CH1786 in the end product and the end product has been delivered to the end user, the end product must be returned to the end product supplier where the CH1786 can be removed and then forward to Cermetek for repair.

FCC Part 68
The Modem AppMod is designed to be used on standard device telephone lines. It connects to the telephone line by means of a standard jack called the USOC RJ-11C (or USOC RJ45S). Connection to telephone-company-provided coin service (central office implemented systems) is prohibited. Connection to party lines service is subject to state tariffs.

Changes in Attestation Procedure for Plugs and Jacks
Parallax, Inc. attests that the network interface plugs or jacks used on this equipment comply with and will continue to comply with the mechanical requirements specified in Part 58, sub-part F, specifically the dimensions, tolerances and metallic plating requirements. The compliance of these connectors will be assured by purchase specifications and incoming inspection. Documentation of such specifications and/ or inspections will be provided to the FCC within 30 days of their request for the same.

Example #1: Controlling LEDs

In this example you will use a PC modem and a terminal program (such as HyperTerminal)[3] to dial the Stamp Modem. When connected, you will toggle two LEDs on and off. This is a simple program intended to demonstrate very basic use of the Stamp Modem.

Follow these steps to setup the Stamp Modem on a Board of Education:

1. Build the circuit shown in Figures 4 and 5 on your Board of Education using two LEDs and two 470 ohm resistors.
2. Connect 6-12 VDC to the Board of Education
3. Connect a serial cable to the Board of Education and your PC's serial port.
4. Connect a phone line to the Stamp Modem's RJ-11 jack.
5. Load CH1786LEDs.BS2 into the BASIC Stamp.
6. Open a HyperTerminal Window. Configure for 8-N-1 and 2400 bps.
7. Dial the Stamp Modem's telephone number from HyperTerminal (Figure 6).

Figure 4: Example #1 Circuit Pictorial

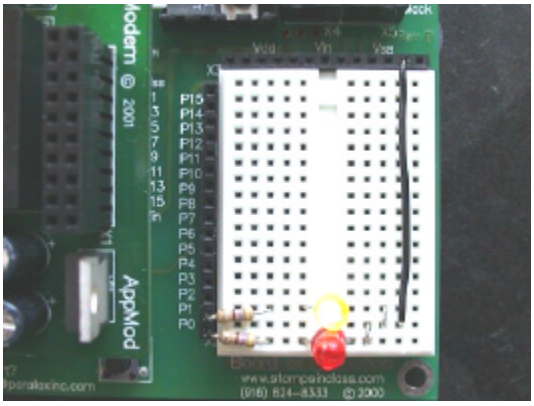


Figure 5: Example #1 Schematic

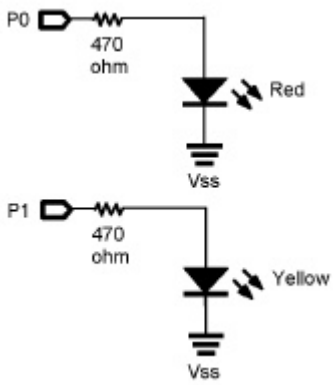
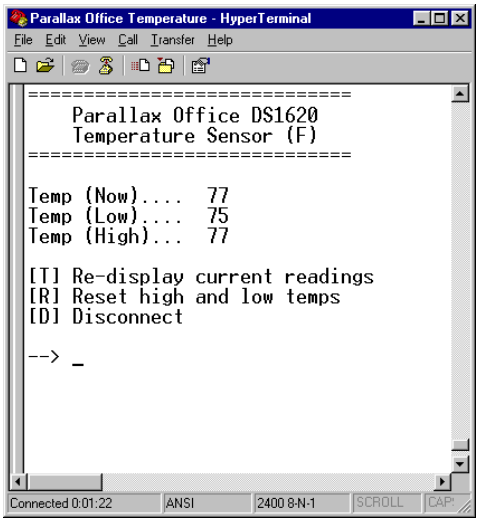


Figure 6: HyperTerminal Connected to Stamp Modem



[3] HyperTerminal Private Edition is available for free download from www.downloads.com.

```
'{$STAMP BS2}

' -----[ Title ]-----------------------------------------------------------
'
' File..... HyperTerminal to Stamp Modem (CH1786LEDs.BS2)
' Purpose... Cermetek CH1786 Demo Program Demonstrates PC to Stamp Modem
' Author.... Parallax, Inc.


' -----[ Program Description ]---------------------------------------------
'
' Stamp Modem answers an incoming call from a PC modem. Two LEDs on the
' Board of Education / Stamp Modem board are controlled using a terminal
' program such as HyperTerminal.


' -----[ I/O Definitions ]-------------------------------------------------
'
Red_LED      CON    0                       ' red LED through 470 ohm
Yellow_LED   CON    1                       ' yellow LED through 470 ohm
RI_          VAR    In12                    ' ring indicator
DCD_         VAR    In13                    ' carrier detect
RX1          CON    14                      ' CH1786 "Rx" pin
TX1          CON    15                      ' CH1786 "Tx" pin


' -----[ Constants ]-------------------------------------------------------
'
T2400        CON    396                     ' 2400 baud for modem: 8N
No           CON    1
Yes          CON    0
LF           CON    10                      ' line feed character
FF           CON    12                      ' clear screen


' -----[ Variables ]-------------------------------------------------------
'
InByte var byte                             ' input character from modem


' -----[ EEPROM Data ]-----------------------------------------------------
'


' -----[ Initialization ]--------------------------------------------------
'
dirs = %100000000000011


' -----[ Main Code ]-------------------------------------------------------
'

Main:
  GOSUB Initialize_Modem                    ' initialize modem
  GOSUB Wait_DCD                            ' carrier detect routine
  GOSUB Communicate
  END


' -----[ Subroutines ]-----------------------------------------------------
'
```

```
Initialize_Modem:
  'DEBUG "Initializing Modem",cr
  PAUSE 1000                                  ' allow modem to power up
  SEROUT TX1, T2400,["AT", CR]                ' setup modem for speed
  SERIN  RX1, T2400, 2500, Error, [WAIT ("OK")]
  PAUSE 250

  SEROUT TX1, T2400,["ATS0=2", CR]            ' answer on second ring.
  SERIN  RX1, T2400, 2500, Error, [WAIT ("OK")]
  PAUSE 250

  SEROUT TX1, T2400,["ATS7=50", CR]           ' max carrier detect is 50 secs
  SERIN  RX1, T2400, 2500, Error, [WAIT ("OK")]
  PAUSE 250

  SEROUT TX1, T2400,["ATS2=43", CR]           ' escape character to "+"
  SERIN  RX1, T2400, 2500, Error, [WAIT ("OK")]
  PAUSE 250

  SEROUT TX1, T2400,["AT&C1", CR]             ' enable DCD
  SERIN  RX1, T2400, 2500, Error, [WAIT ("OK")]
  PAUSE 250

  PAUSE 600
  SEROUT TX1, T2400,["AT&D0", CR]             ' disable DTR
  SERIN  RX1, T2400, 2500, Error, [WAIT ("OK")]
  PAUSE 200
  RETURN

Wait_DCD:                                     ' wait for carrier detect
  'DEBUG "Waiting for carrier detect",cr
  PAUSE 10
  IF DCD_ = YES THEN Connect                 ' wait for carrier
  GOTO Wait_DCD

Connect:
  PAUSE 30000
  RETURN

Communicate:
  'DEBUG "Sending data to HypterTerminal",cr
  SEROUT TX1, T2400, [FF]
  SEROUT TX1, T2400, ["=========================", CR, LF]
  SEROUT TX1, T2400, ["CH1786 LED Demo Program   ", CR, LF]
  SEROUT TX1, T2400, ["=========================", CR, LF]
  SEROUT TX1, T2400, ["Command options:          ", CR, LF]
  SEROUT TX1, T2400, [" [R] Red LED toggle       ", CR, LF]
  SEROUT TX1, T2400, [" [Y] Yellow LED toggle    ", CR, LF]
  SEROUT TX1, T2400, [" [E] End the call         ", CR, LF]
  SEROUT TX1, T2400, [CR, LF]
  SEROUT TX1, T2400, [" Red LED status    = ", bin1 in0,CR, LF]
  SEROUT TX1, T2400, [" Yellow LED status = ", bin1 in1,CR, LF]
  SEROUT TX1, T2400, [CR, LF]
  SEROUT TX1, T2400, ["Enter command ==> "]
  PAUSE 500
  SERIN RX1,T2400,[InByte]                    ' get byte
  IF InByte = "E" THEN Hang_up
  IF InByte = "R" THEN Toggle_red
  IF InByte = "Y" THEN Toggle_yellow
  GOTO Communicate

Hang_up:                                      ' hang up modem
  'DEBUG "Hanging up modem",cr
```

```
  OUTL=%0000                              ' LEDs off
  SEROUT TX1, T2400,[CR,LF,LF,"Disconnect requested", CR,LF]
  PAUSE 2000
  SEROUT TX1, T2400, ["+++"]
  PAUSE 2000
  SEROUT TX1, T2400, 10, ["ATH0", CR]
RETURN

Toggle_red:
  TOGGLE 0
  GOTO Communicate

Toggle_yellow:
  TOGGLE 1
  GOTO Communicate

Error:
  'DEBUG "Communication error!",cr
END
```

Example #2: Temperature Measurement

In this example you'll connect to a remote temperature monitor that displays current, high and low temperatures recorded while the program has been running. The program example is from the Nuts and Volts' Stamp Applications column #61 "Calling All Stamps". This article is available for download from www.parallaxinc.com/stampmodem and could be read while reviewing the example source code.

Follow these steps to setup the Stamp Modem on a Board of Education:

1. Build the circuit shown in Figures 7 and 8 on your Board of Education using a Dallas Semiconductor 1620 Digital Thermometer, 1K ohm resistor and 0.1 uf capacitor (optional).
2. Connect 6-12 VDC to the Board of Education
3. Connect a serial cable to the Board of Education and your PC's serial port.
4. Connect a phone line to the Stamp Modem's RJ-11 jack.
5. Load CH1786Temperature.BS2 into the BASIC Stamp.
6. Open a HyperTerminal Window. Configure for 8-N-1 and 2400 bps.
7. Dial the Stamp Modem's phone number from HyperTerminal (Figure 9).

Figure 7: Example #2 Circuit Pictorial

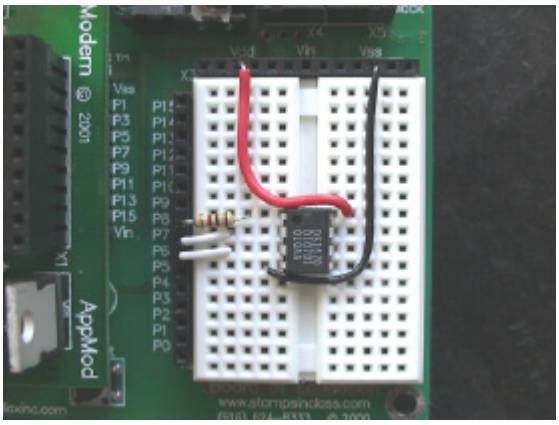


Figure 8: Example #2 Schematic

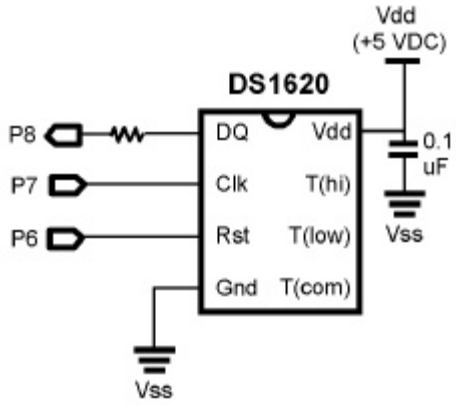
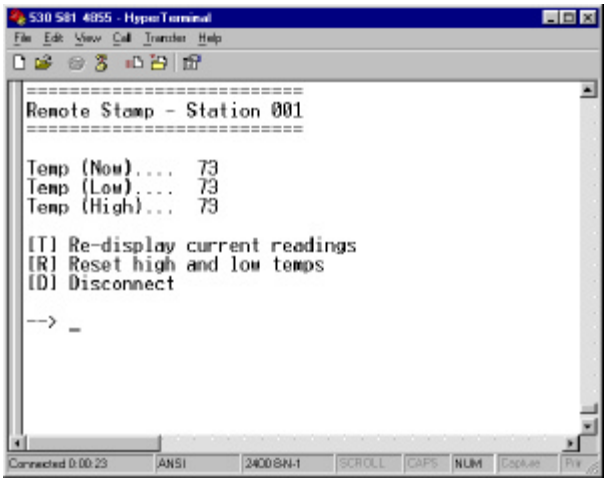


Figure 9: HyperTerminal Connected to Stamp Modem



```
'{$STAMP BS2}

' -----[ Title ]-----------------------------------------------------------
'
' File..... Temperature (CH1786Temperature.BS2)
' Purpose... Stamp Modem Demo Program with DS1620
' Author.... Parallax, Inc.


' -----[ Program Description ]---------------------------------------------
'
' This program monitors a Dallas Seminconductor DS1620 digital thermometer
' while waiting for an incoming call.  When a call is received, the Stamp
' causes the modem to answer the call then displays temperature data on
' the remote terminal.


' -----[ I/O Definitions ]--------------------------------------------------
'
' modem pins
'
TX1          CON    15                      ' CH1786 "Tx" pin
RX1          CON    14                      ' CH1786 "Rx" pin
RI_          VAR    In12                    ' ring indicator
DCD_         VAR    In13                    ' carrier detect

' DS1620 pins
'
Rst          CON    6                       ' DS1620.3
Clk          CON    7                       ' DS1620.2
DQ           CON    8                       ' DS1620.1


' -----[ Constants ]-------------------------------------------------------
'
True         CON    1
False        CON    0

No           CON    1
Yes          CON    0

T2400        CON    396                     ' 2400 baud for modem

LF           CON    10                      ' line feed character
FF           CON    12                      ' form feed (clear remote screen)


' DS1620 commands
'
RTmp         CON    $AA                     ' read temperature
WTHi         CON    $01                     ' write TH (high temp register)
WTLo         CON    $02                     ' write TL (low temp register)
RTHi         CON    $A1                     ' read TH
RTLo         CON    $A2                     ' read TL
Strt         CON    $EE                     ' start conversion
StpC         CON    $22                     ' stop conversion
WCfg         CON    $0C                     ' write configuration register
RCfg         CON    $AC                     ' read configuration register


NTasks       CON    3                       ' total number of tasks
```

```
' -----[ Variables ]------------------------------------------------------
'
tmpIn        VAR   Word                    ' 9-bit temp input from DS1620
nFlag        VAR   tmpIn.Bit8              ' negative flag
hlfBit       VAR   tmpIn.Bit0              ' half degree C bit
tempF        VAR   Word                    ' converted fahrenheit value
tempC        VAR   Byte                    ' converted celcius value
tmpNow       VAR   Word                    ' current temperature
tmpLo        VAR   Word                    ' low temp
tmpHi        VAR   Word                    ' high temp

sign         VAR   Byte                    ' - for negative temps
sLo          VAR   Byte
sHi          VAR   Byte

inByte       VAR   Byte                    ' input from user terminal
cmd          VAR   Byte                    ' command pointer
answer       VAR   Byte                    ' user response to prompt

task         VAR   Byte                    ' task control variable

riFltr       VAR   Byte                    ' for ring indicator filter


' -----[ EEPROM Data ]-----------------------------------------------------
'


' -----[ Initialization ]---------------------------------------------------
'
Init:
  tmpLo = $FFFF                            ' start with opposite extremes
  tmpHi = 0

I_Modm:
  PAUSE 250                                ' allow modem to power up
  ' train modem for speed
  '
  SEROUT TX1, T2400, 10, ["AT", CR]
  SERIN  RX1, T2400, 2500, Error, [WAIT ("OK")]
  PAUSE 250
  ' auto answer on second ring (S0=2)
  ' set max time for carrier detect to 30 secs (S7=30)
  '
  SEROUT TX1, T2400, 10, ["ATS0=2 S7=30", CR]
  SERIN  RX1, T2400, 2500, Error, [WAIT ("OK")]

I_1620:
  HIGH Rst                                 ' alert the DS1620
  SHIFTOUT DQ,Clk,LSBFIRST,[WCfg]          ' write configuration
                                           ' use with CPU; free run mode
  SHIFTOUT DQ,Clk,LSBFIRST,[%00000010]
  LOW Rst
  PAUSE 10                                 ' pause for DS1620 EE write cycle
  HIGH Rst
  SHIFTOUT DQ,Clk,LSBFIRST,[Strt]          ' start temp conversions
  LOW Rst
  debug "Initializing DS1620",cr
NoDCD: IF DCD_ = Yes THEN NoDCD            ' make sure DCD is clear


' -----[ Main Code ]--------------------------------------------------------
```

```
'

Main:  GOSUB ScanT                              ' get current temperature
  IF DCD_ = Yes THEN GetMdm                     ' call received

  BRANCH task, [Task0, Task1, Task2]
  GOTO Main

Task0:                                          ' task code here
  '
  task = 1                                      ' select a specific task
  GOTO NextT                                    ' go do it

Task1:                                          ' task code here
  '
  task = 2
  GOTO NextT

Task2:                                          ' task code here
  '
  task = 0
  GOTO NextT

NextT:                                          ' task = task + 1 // NTasks
                                                ' round-robin to next task
  GOTO Main
  END


' -----[ Subroutines ]------------------------------------------------------
'

' ==============
' Modem Routines
' ==============

' error with modem
' - structured as seperate routine to allow user indications/enhancements
'
Error:                                          ' additional code here
  PAUSE 1000
  GOTO I_Modm                                   ' try to initialize again


GetMdm:  PAUSE 5000                             ' let other end get ready
Modm1: GOSUB DoMenu                             ' show readings and menu
Get1:  SERIN RX1, T2400, [inByte]               ' wait for input

                                                ' process user input
  cmd = 99
                                                ' convert letter to digit (0..5)
  LOOKDOWN inByte, ["tTrRdD"], cmd
  cmd = cmd / 2                                 ' fix for BRANCH
                                                ' branch to handler
  BRANCH cmd, [Cmd0, Cmd1, Cmd2]
  GOTO Modm1

Cmd0:  GOSUB ScanT                              ' get current temp
  GOTO Modm1
Cmd1:  GOSUB RstT                               ' reset high and low
  GOTO Modm1
Cmd2:  GOSUB Discon                             ' disconnect from user
  IF answer = No THEN Modm1                     ' stay with user
```

```
  GOTO NoDCD                          ' back to the beginning


' clear remote terminal and display menu
'
DoMenu:
  debug "DoMenu",cr
SEROUT TX1, T2400, [FF]
  SEROUT TX1, T2400, ["=========================", CR, LF]
  SEROUT TX1, T2400, ["Remote Stamp - Station 001", CR, LF]
  SEROUT TX1, T2400, ["=========================", CR, LF]
  SEROUT TX1, T2400, [LF]
  SEROUT TX1, T2400, ["Temp (Now).... ", sign, DEC tmpNow, CR, LF]
  SEROUT TX1, T2400, ["Temp (Low).... ", sLo,  DEC tmpLo, CR, LF]
  SEROUT TX1, T2400, ["Temp (High)... ", sHi,  DEC tmpHi, CR, LF]
  SEROUT TX1, T2400, [LF]
  SEROUT TX1, T2400, ["[T] Re-display current readings", CR, LF]
  SEROUT TX1, T2400, ["[R] Reset high and low temps", CR, LF]
  SEROUT TX1, T2400, ["[D] Disconnect", CR, LF]
  SEROUT TX1, T2400, [LF, "--> "]
  RETURN


' reset high and low temperatures
'
RstT:  SEROUT TX1, T2400, [CR, LF, LF, "Reset? "]
  GOSUB YesNo
  IF answer = No THEN RstX
  GOSUB ScanT
  tmpLo = tmpNow
  sLo = sign
  tmpHi = tmpNow
  sHi = sign
RstX:  RETURN


' disconnect
'
Discon:  SEROUT TX1, T2400, [CR, LF, LF, "Disconnect? "]
  GOSUB YesNo
  IF answer = No THEN DiscX
  SEROUT TX1, T2400, [CR, LF, LF, "Disconnecting.", CR, LF]

' return modem to command state
' and hang up
'
  PAUSE 2000
  SEROUT TX1, T2400, ["+++"]
  PAUSE 2000
  SEROUT TX1, T2400, 10, ["ATH0", CR]
DiscX: RETURN


' confirm for [Y]es or [N]o
' and get user input (default = No)
'
YesNo: SEROUT TX1, T2400, ["Are you sure? (Y/N) : "]
  answer = No

' get answer
' - but only wait for 5 seconds
'
  SERIN RX1, T2400, 5000, YesNoX, [inByte]
```

```
  IF inByte = "y" THEN IsYes
  IF inByte = "Y" THEN IsYes
  GOTO YesNoX
IsYes: answer = Yes
YesNoX:  RETURN

' process ring indicator
' - filters pulsing ring indicator
' - waits for about 0.25 second of no RI pulsing before returning
'
DoRing:
' your code here
'  (i.e., count number of rings)
'
RIWait:  riFltr = 0                             ' clear the "no pulses" counter
RIchk: IF RI_ = Yes THEN RIWait                 ' still pulsing
  riFltr = riFltr + 1                           ' not pulsing, increment count
  IF riFltr > 50 THEN RIx                       ' RI clear now
  PAUSE 5                                       ' 5 ms between RI scans
  GOTO RIchk                                    ' check again
RIx:  RETURN                                    ' done - outta here


' ===============
' DS1620 Routines
' ===============

' get current temperature
' -- update high and low readings
'
ScanT: HIGH Rst                                 ' alert the DS1620
  SHIFTOUT DQ,Clk,LSBFIRST,[RTmp]               ' read temperature
  SHIFTIN DQ,Clk,LSBPRE,[tmpIn\9]               ' get the temperature
  LOW Rst
  GOSUB GetF                                    ' convert to Farhenheit
  tmpNow = tempF
  IF (tmpLo < tmpNow) THEN THigh
  tmpLo = tmpNow                                ' set new low
  sLo = sign
  debug "Scan T",cr
THigh: IF (tmpHi > tmpNow) THEN TDone
  tmpHi = tmpNow                                ' set new high
  sHi = sign
TDone: RETURN


' convert reading from 1/2 degrees input (rounds up)
'
GetC:  IF nFlag = 0 THEN CPos                   ' check negative bit (8)
  sign = "-"                                    ' set sign
  tempC = -tmpIn / 2 + hlfBit                   ' if neg, take 2's compliment
  GOTO CDone
CPos:  sign = " "
  tempC = tmpIn / 2 + hlfBit
CDone: RETURN


' convert (1/2 degrees C) to Fahenheit with rounding
' -- general equation (for whole degrees): F = C * 9 / 5 + 32
'
GetF:  sign = " "
  IF nFlag = 0 THEN FPos1
  tmpIn = -tmpIn & $FF                          ' convert from negative
```

```
  IF tmpIn < 36 THEN FPos0
FNeg:  sign = "-"
  tempF = tmpIn * 9 / 10 + hlfBit - 32
  GOTO FDone
FPos0: tempF = 32 - (tmpIn * 9 / 10 + hlfBit)
  GOTO FDone
FPos1: tempF = tmpIn * 9 / 10 + 32 + hlfBit
FDone: RETURN
```