

USING THE AMULET STARTER KIT WITH A PARALLAX BASIC STAMP AS SLAVE

INTRO

The Amulet Starter Kit not only works well for rapid prototyping, but also fast product implementation and easy, seamless development. In combination with another microprocessor and a peripheral I/O device, the Amulet GUI is a powerful technology. In this experiment, we will be using a Basic Stamp 2 as the externally interfaced microprocessor (because of its popularity and wide spread use in the embedded industry) with a Dallas Semiconductor DS1620 Digital Thermometer. The DS1620 is not particularly relevant to the basic communication between the Amulet GUI and the Stamp, it is used as an example of an application to execute the commands sent by the Amulet. This app note will specifically highlight the hardware setup and software communications protocol, not the user interface created to demo this experiment. It is set up with the Amulet acting as master and the Stamp as the slave.

HARDWARE CONNECTIONS

The easiest way to set up this system is to use a Basic Stamp Activity Board with the Basic Stamp (preferably a Basic Stamp 2 or faster) to talk to the Amulet board via a RS232 (serial) cable. This setup allows for easy connection of the DS1620 and Activity Board to a PC for debugging and programming in addition to connecting to the Amulet unit once the Stamp has been programmed. The schematic for the connection between the Stamp and the board is shown below.

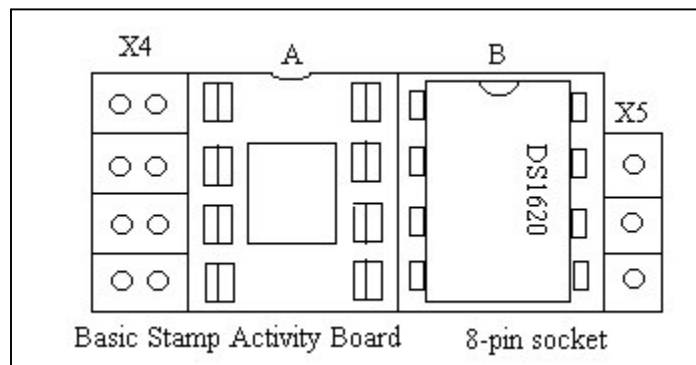


Figure 1. Schematic of Stamp connection to the DS1620 Digital Thermometer

AMULET PROTOCOL (taken from <http://www.amulettechnologies.com/>)

The communications protocol is half-duplex, with the Amulet GUI client acting as master. The Amulet GUI can send ten different types of messages:

- A "Get byte variable" request. (`Amulet:UART.byte(x).value()`)
- A "Get word variable" request. (word = 2 bytes) (`Amulet:UART.word(x).value()`)
- A "Get string variable" request. (`Amulet:UART.string(x).value()`)
- A "Get label variable" request. (`Amulet:UART.label(x).value()`)
- A "Set byte variable" command. (`Amulet:UART.byte(x).setValue(y)`)
- A "Set word variable" command. (`Amulet:UART.word(x).setValue(y)`)
- A "Set string variable" command. (`Amulet:UART.string(x).setValue(y)`)
- An "Invoke Remote Procedure Call (RPC)" command. (`Amulet:UART.invokeRPC()`)
- A "Get byte variable array" request. (`Amulet:UART.bytes(x).value()`)
- A "Get word variable array" request. (`Amulet:UART.words(x).value()`)

If the message is valid, the Stamp should either return the requested data (if a "Get" request) or acknowledge the message (if an "Invoke" or "Set" command). The Amulet recognizes the acknowledge byte 0xF0. If the message is not valid, the server should respond with a negative acknowledge (0xF1).

Table 1 in [Appendix A](#) defines the ten types of messages that can be sent between the GUI and the Stamp. The valid range of values for an RPC request is 0-0xFF. The valid range for byte variable values returned by the Stamp (in response to the "Get Byte variable" request) is also 0-0xFF. The valid range for word variable values returned from the Stamp (in response to the "Get Word variable" request) is 0-0xFFFF. String and label variable values returned from the slave can have a maximum of 252 ASCII (0x20-0x7E) characters plus a null termination character (0x00). The length of the variable array used to store the stream of data (see code example below) should be large enough to hold the largest message the Amulet will be sending for that program.

The Amulet uses an ASCII protocol. This means that the variable 0x1A would be transmitted as 0x31, 0x41, where 0x31 is the ASCII representation of the high nibble "1" and 0x41 is the ASCII representation of the low nibble "A".

NOTE: The Stamp side must respond to every valid Amulet command, even if it is just an acknowledge (0xF0). When commands are not responded to within 200ms, a time-out will occur, and the message will repeat until an answer is received.

Synchronization--As master, the GUI initiates all communications by sending a message to the Stamp. All valid messages from the GUI to the Stamp start with one of ten command bytes: [0xD0], [0xD1], [0xD2], [0xD3], [0xD5], [0xD6], [0xD7], [0xD8], [0xDD] or [0xDE] -- these are considered the Client Start Of Message (CSOM) characters, with the GUI being the client. **NOTE:** These ten CSOM bytes ALWAYS signify the start of a message and they are not allowed in the body of any message. The only valid characters in the body of a message are: ASCII 0-9 (0x31-0x39), and A-F

(0x41-0x46), except in the body of the “Get string,” “Get label,” or “Set string” responses, where all ASCII characters from ‘ ‘ through ‘~’ (0x20-0x7E) are valid. If the Stamp receives any character other than those specified, the message should be considered errant, and the Stamp should start over hunting for a new CSOM character.

All Stamp responses must start with the counterpart of the CSOM character that began the message that is being responded to. The valid Server (Stamp) Start Of Message (SSOM) bytes are: [0xE0], [0xE1], [0xE2], [0xE3], [0xE4], [0xE5], [0xE6], [0xE7], [0xE8], [0xED], or [0xEE]. The body of the response message starts with the counterpart SSOM and is then followed by any optional response data (in ASCII format).

As noted earlier, after receiving the last byte of a valid message from the GUI, the Stamp then has 200ms to respond to the message before the Amulet times out. After 200ms, if there is no response, the GUI will continue transmitting the message until a response is received. After 10 unsuccessful attempts, the Amulet will flush its transmit buffer and reset all UART variables.

When using BASIC Stamp the Amulet should be set to a baud rate of 9600. The Amulet also needs to null terminate each message so that the Stamp can determine when to end each string. The BASICStamp Meta tag should be used to both null terminate each message and set the communication protocol to an appropriate interbyte delay. The Amulet’s BASICStamp Meta tags are as follows:

```
<META NAME="Amulet" Content="BASICStamp.project">  
<META NAME="Amulet" Content="baud.project=9600">
```

SOFTWARE INTERFACE

An abbreviated version of the communication protocol used to make the GUI, Stamp, and thermometer communicate is shown below. The software used to program the Stamp can be found at the following link:

http://www.parallax.com/html_pages/downloads/software/software_basic_stamp.asp

The following code snippets were taken from an actual implementation based on the Basic Stamp 2 micro controller. The code simply polls the serial line – with the SERIN command – until a valid request is received, handles the request – using the IF statements to determine what the Amulet is asking for, and then returns to polling the serial line. This sample code, showing only the service routines, is meant to illustrate the workings of the Amulet protocol, not implement it. The entire code can be found in [Appendix B](#).

```
serial_in:  
  
'*****  
'* This function will poll the serial line looking for values. It will store  
'* the first Byte in AmuletMsg(0), the next in AmuletMsg(1), the third in  
'* AmuletMsg(2), and so on. Each Amulet message will have a minimum of three  
'* bytes. AmuletMsg(0) stores the Amulet's request type, AmuletMsg(1/2) stores
```

```

'* which variable is being used, and any further bytes (if it is a "set" request)
'* contain the value the variable should be set to. The program will hang until
'* either all 10 bytes of the AmuletMsg array are filled, or a Null is sent.
'* Using the BASICStamp Meta in your html page automatically null terminates
'* every command sent. If the stamp will be receiving strings or arrays longer
'* than 7 bytes (not including the 3 bytes for the request and variable being used)
'* than the MsgLength will need to be adjusted accordingly.
*****
SERIN 16, 84, [STR AmuletMsg \6 \0]

Variable.HIGHNIB = AmuletMsg(1)      ' The variable referred to in the Amulet's
Variable.LOWNIB  = AmuletMsg(2)      ' message is put back into a single decimal
                                       ' number ('0' instead of 0x30,0x30)

IF AmuletMsg(0) = $D0 THEN getByte   'asking for a "Get Byte"
IF AmuletMsg(0) = $D1 THEN getWord  'asking for a "Get Word"
IF AmuletMsg(0) = $D2 THEN getString 'asking for a "Get String"
IF AmuletMsg(0) = $D3 THEN getLabel 'asking for a "Get Label"
IF AmuletMsg(0) = $D5 THEN setByte   'asking for a "Set Byte"
IF AmuletMsg(0) = $D6 THEN setWord  'asking for a "Set Word"
IF AmuletMsg(0) = $D7 THEN setString 'asking for a "Set String"
IF AmuletMsg(0) = $D8 THEN invokeRPC 'asking to "Invoke an RPC"
IF AmuletMsg(0) = $DD THEN getByteArr 'asking for a "Get Byte Array"
IF AmuletMSG(0) = $DE THEN getWordArr 'asking for a "Get Word Array"

SEROUT 16, 84, ["F1"]                'if an "unknown" command, send negative acknowledge
GOTO serial_in                       'and then jump to start of loop to wait for next command

*****
'* Handler for a getByte function request
'* Format of request string = 0xD0xx where xx=the variable byte being used
'* Returns 0xE0xxNN, where NN equals the HEX data of the variable xx
*****
getByte:
  B0 = $40
  B1 = $83
  IF Variable = 0 THEN getByteZero
  IF Variable = 1 THEN getByteOne

  ' if Byte 0 has been called to set
  getByteZero:
    SEROUT 16, 84, [224,"00",HEX B0]
    GOTO serial_in

  ' if Byte 1 has been called to set
  getByteOne:
    SEROUT 16, 84, [224,"01",HEX B1]
    GOTO serial_in

*****
'* Handler for a getString function request
'* Format of request string = 0xD2xx, where xx = index of string variable
'* Returns 0xE2xxString+Null to client
'* Returns requested data back to the screen
*****
getString:
  IF Variable = 0 THEN Company
  IF Variable = 1 THEN Thermostat
  IF Variable = 2 THEN Stamp

  Company:
    SEROUT 16, 84, [226,"00", "Amulet Technologies", NULL]
    GOTO serial_in

  Thermostat:
    SEROUT 16, 84, [226,"01", "DS1620 Digital Thermostat", NULL]
    GOTO serial_in

  Stamp:
    SEROUT 16, 84, [226,"02", "Basic Stamp 2 on Activity Board", NULL]
    GOTO serial_in

```

```

'*****
'* Handler for a invoke function request
'* Format of request string = 0xD8xx, where xx = RPC being requested
'* Returns 0xE8xx to client)
'*****
invokeRPC:
'Invoke RPC reads the temp into DSdata in leaves it in Celsius if a "1" and
'reads the temp into DSdata and stores it in Fahrenheit if a "2"

HIGH RST                'Activate the '1620
SHIFTOUT DQ,CLK,LSBFIRST,[Rtemp] 'Request to read temperature
SHIFTIN DQ,CLK,LSBPRE,[DSdata\9] 'Get the temperature reading
LOW RST                 'Deactivate
DSdata = DSdata/2      'Scale reading to whole degrees C

IF Variable = 0 THEN Celsius
IF Variable = 1 THEN Fahrenheit

'RPC 0 is being requested so set the temperature read by the thermostat
'in degrees Celsius
Celsius:
  SEROUT 16, 84, [232, HEX2 Variable]
  GOTO serial_in

'RPC 1 is being requested so set the temperature read by the thermostat
'in degrees Fahrenheit
Fahrenheit:
  DSdata = (DSdata*$/01CC)          'Convert from Celsius to Fahrenheit
  DSdata = DSdata + 32              '(multiply by 1.8, add 32)
  SEROUT 16, 84, [232, HEX2 Variable]
  GOTO serial_in

'*****
'* Handler for a setByte function request
'* Format of request string = 0xD5xxNN, where xx = variable to be set
'* and NN = HEX data
'* Returns 0xE5xxNN to Amulet
'*****
setByte:
IF Variable = 0 THEN setZeroByte
IF Variable = 1 THEN setOneByte

'set Byte to received value

'Byte 0 has been called to set
setZeroByte:
  B0.HIGHNIB = AmuletMsg(3)
  B0.LOWNIB = AmuletMsg(4)
  SEROUT 16, 84, [229,"00",HEX B0]
  GOTO serial_in

'Byte 1 has been called to set
setOneByte:
  B1.HIGHNIB = AmuletMsg(3)
  B1.LOWNIB = AmuletMsg(4)
  SEROUT 16, 84, [229,"01",HEX B1]
  GOTO serial_in

```



Amulet Technologies, LLC

GUI Engines for Embedded Systems

Appendix A

Message	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte N
Amulet Get Byte Variable	0xD0	Variable Hi Nibble	Variable Lo Nibble	None	None	None	None	None
Server Response	0xE0	Variable Hi Nibble	Variable Lo Nibble	Value Hi Nibble	Value Lo Nibble	None	None	None
<hr/>								
Amulet Get Word Variable	0xD1	Variable Hi Nibble	Variable Lo Nibble	None	None	None	None	None
Server Response	0xE1	Variable Hi Nibble	Variable Lo Nibble	----MS Byte---- Value Hi Hi Nibble	----LS Byte---- Value Hi Lo Nibble	----LS Byte---- Value Lo Hi Nibble	----LS Byte---- Value Lo Lo Nibble	None
<hr/>								
Amulet Get String Variable	0xD2	Variable Hi Nibble	Variable Lo Nibble	None	None	None	None	None
Server Response	0xE2	Variable Hi Nibble	Variable Lo Nibble	ASCII char	ASCII char	ASCII char	ASCII	0x00 char
<hr/>								
Amulet Get Label Variable	0xD3	Variable Hi Nibble	Variable Lo Nibble	None	None	None	None	None
Server Response	0xE3	Variable Hi Nibble	Variable Lo Nibble	ASCII char	ASCII char	ASCII char	ASCII	0x00 char
<hr/>								
Amulet Set Byte Variable	0xD5	Variable Hi Nibble	Variable Lo Nibble	Value Hi Nibble	Value Lo Nibble	None	None	None
Server Response	0xE5	Variable Hi Nibble	Variable Lo Nibble	Value Hi Nibble	Value Lo Nibble	None	None	None
<hr/>								
Amulet Set Word Variable	0xD6	Variable Hi Nibble	Variable Lo Nibble	----MS Byte---- Value Hi Hi Nibble	----LS Byte---- Value Hi Lo Nibble	----LS Byte---- Value Lo Hi Nibble	----LS Byte---- Value Lo Lo Nibble	None
Server Response	0xE6	Variable Hi Nibble	Variable Lo Nibble	----MS Byte---- Value Hi Hi Nibble	----LS Byte---- Value Hi Lo Nibble	----LS Byte---- Value Lo Hi Nibble	----LS Byte---- Value Lo Lo Nibble	None
<hr/>								

Amulet Set String Variable	0xD7	Variable Hi Nibble	Variable Lo Nibble	ASCII char	ASCII char	ASCII char	ASCII 0x00 char	
Server Response	0xE7	Variable Hi Nibble	Variable Lo Nibble	ASCII char	ASCII char	ASCII char	ASCII 0x00 char	
<hr/>								
Amulet Invoke Remote Procedure Call (RPC)	0xD8	RPC Hi Nibble	RPC Lo Nibble	None	None	None	None	None
Server Response	0xE8	RPC Hi Nibble	RPC Lo Nibble	None	None	None	None	None
<hr/>								
Amulet Get Byte Variable Array	0xDD	Variable Hi Nibble	Variable Lo Nibble	None	None	None	None	None
Server Response	0xED	Variable Hi Nibble	Variable Lo Nibble	Value Hi Nibble	Value 0x00 Lo Nibble		None	None
<hr/>								
Amulet Get Word Variable Array	0xDE	Variable Hi Nibble	Variable Lo Nibble	None	None	None	None	None
Server Response	0xEE	Variable Hi Nibble	Variable Lo Nibble	-----MS Byte----- Value Hi Hi Nibble	Value Hi Lo Nibble	-----LS Byte----- Value Lo Hi Nibble	Value Lo Lo Nibble 0x00

Table 1. Ten types of messages that can be sent between the client and the server.



Amulet Technologies, LLC

GUI Engines for Embedded Systems

Appendix B

```
*****
*The main loop of this code waits for valid Client Start of Message (CSOM) characters
*then jumps to the appropriate service routine
*****

'{$STAMP BS2}

*****
* Pin Setup - pins to communicate with the DS1620
*****
DQ      CON      15      'Data input/output line
CLK     CON      14      'Synchronizing Clock
RST     CON      13      'Reset/select (high=active)
Wconfig CON      $0C     'Protocol for 'Write Configuration'
CPU     CON      %10     'Config bit: serial thermometer mode
Cont    CON      %00     'Config bit: continuous conversions after start
StartC  CON      $EE     'Protocol for 'Start Conversion'
Rtemp   CON      $AA     'Protocol for 'Read Temperature'
RhiT    CON      $A1     'Protocol for 'Read High-Temperature Setting'
RloT    CON      $A2     'Protocol for 'Read Low-Temperature Setting'

*****
* Variable declarations
*****
' Variables for Stamp-Thermostat communication
Dpdata  VAR      Word    'Word variable holds 9-bit temperature
Thigh   VAR      Word    'Word variable holds 9-bit "max" temp before AC turns on
Tlow    VAR      Word    'Holds 9-bit "min" temp before heater turns on

' Variables for Stamp-Amulet communication
NULL    CON      0       'Null character for end of String
Value   VAR      Word    'The value received for a setByte or setWord command
Variable VAR      Byte   'The Amulet variable (in decimal)
AmuletMsg VAR     Byte(6) 'array that captures message
Str0    VAR      Byte(3) ' two string arrays to hold
Str1    VAR      Byte(3) ' incoming strings
AmuletMsg(6) = NULL      'Null terminate last byte

*****
*Configure the DS1620
*****
LOW RST      'Deactivate '1620 for now
HIGH CLK     'Put clock in starting state
PAUSE 100    'Let things settle down a moment
HIGH RST     'Activate the '1620 and set it for continuous temp conversion
SHIFTOUT DQ,CLK,LSBFIRST,[Wconfig,CPU+Cont]
LOW RST      'Done-deactivate
PAUSE 50     'Wait for EEPROM to self-program
HIGH RST     'Now activate it again and send the start-conversion protocol
SHIFTOUT DQ,CLK,LSBFIRST,[StartC]
LOW RST      'Done-deactivate

serial_in:

*****
* This function will poll the serial line looking for values. It will store
* the first Byte in AmuletMsg(0), the next in AmuletMsg(1), the third in
* AmuletMsg(2), and so on. Each Amulet message will have a minimum of three
* bytes. AmuletMsg(0) stores the Amulet's request type, AmuletMsg(1/2) stores
* which variable is being used, and any further bytes (if it is a "set" request)
* contain the value the variable should be set to. The program will hang until
* either all 10 bytes of the AmuletMsg array are filled, or a Null is sent.
* Using the BASICStamp Meta in your html page automatically null terminates
* every command sent. If the stamp will be receiving strings or arrays longer
* than 7 bytes (not including the 3 bytes for the request and variable being used)
* than the Msg length will need to be adjusted accordingly.
*****
```

```

SERIN 16, 84, [STR AmuletMsg \6 \0]

Variable.HIGHNIB = AmuletMsg(1) ' The variable referred to in the Amulet's message is
Variable.LOWNIB = AmuletMsg(2) ' put back into a single decimal number ('0' instead
                               ' of 0x30,0x30)

IF AmuletMsg(0) = $D0 THEN getByte      'asking for a "Get Byte"
IF AmuletMsg(0) = $D1 THEN getWord     'asking for a "Get Word"
IF AmuletMsg(0) = $D2 THEN getString   'asking for a "Get String"
IF AmuletMsg(0) = $D3 THEN getLabel    'asking for a "Get Label"
IF AmuletMsg(0) = $D5 THEN setByte     'asking for a "Set Byte"
IF AmuletMsg(0) = $D6 THEN setWord     'asking for a "Set Word"
IF AmuletMsg(0) = $D7 THEN setString   'asking for a "Set String"
IF AmuletMsg(0) = $D8 THEN InvokeRPC   'asking to "Invoke an RPC"
IF AmuletMsg(0) = $DD THEN getByteArr  'asking for a "Get Byte Array"
IF AmuletMsg(0) = $DE THEN getWordArr  'asking for a "Get Word Array"

SEROUT 16, 84, ["F1"]                  'if an "unknown" command, send a negative acknowledge
GOTO serial_in                          'and then jump to start of loop to wait for next command

*****
'* Handler for a getByte function request
'* Format of request string = 0xD0xx where xx=the variable byte being used
'* Returns 0xE0xxNN, where NN equals the HEX data of the variable xx
*****

getByte:
  B0 = $40
  B1 = $83
  IF Variable = 0 THEN getByteZero
  IF Variable = 1 THEN getByteOne

  'get Byte 0
  'Byte 0 has been called to set
  getByteZero:
    SEROUT 16, 84, [224,"00",HEX B0]
    GOTO serial_in

  'Byte 1 has been called to set
  getByteOne:
    SEROUT 16, 84, [224,"01",HEX B1]
    GOTO serial_in

*****
'* Handler for a getWord function request
'* Format of request string = 0xD1xx, where xx = variable being requested
'* Returns 0xE1xxPPNN, where PP = high byte of variable xx, and NN = low byte of
'* variable xx
*****

getWord:
  'Read Thigh and Tlow from the Digital Thermometer
  HIGH RST                                'Activate the DS1620
  SHIFTOUT DQ,CLK,LSBFIRST,[RhiT]         'Request to read "max" temp before the AC kicks on
  SHIF TIN DQ,CLK,LSBP RE,[Thigh\9]        'Get Thigh
  LOW RST                                  'deactivate the DS1620
  PAUSE 50
  HIGH RST                                'Activate the DS1620
  SHIFTOUT DQ,CLK,LSBFIRST,[RloT]         'Request to read 'min' temp before heater kicks on
  SHIF TIN DQ,CLK,LSBP RE,[Tlow\9]        'Get Tlow
  LOW RST                                  'deactivate the DS1620
  Tlow = Tlow/2                            'Convert to whole degrees Celsius
  Thigh = Thigh/2                          'Convert to whole degrees Celsius

  IF Variable = 0 THEN TempHLCelsius
  IF Variable = 1 THEN TempHLFahren

```

```
'Word 0 is being requested so return the high/low temp settings of the thermostat
'in degrees Celsius as a word
```

```
TempHLCelsius:
  'Send to Amulet E1,byte "00", data Th,Tl
  SEROUT 16, 84, [225,"00",HEX Thigh,HEX Tlow]
  GOTO serial_in
```

```
'Word 1 is being requested so return the high/low temp settings of the thermostat
'in degrees Fahrenheit as a word
```

```
TempHLFahren:
  Thigh = (Thigh*/$01CC)           'Convert from Celsius to Fahrenheit
  Tlow = (Tlow*/$01CC)
  Thigh = Thigh + 32               '(multiply by 1.8, add 32)
  Tlow = Tlow + 32
  'Send to Amulet E1,byte "01", data Th,Tl
  SEROUT 16, 84, [225,"01",HEX Thigh,HEX Tlow]
  GOTO serial_in
```

```
*****
'* Handler for a getString function request
'* Format of request string = 0xD2xx, where xx = index of string variable
'* Returns 0xE2xxString+Null to client
'* Returns requested data back to the screen
*****
```

```
getString:
```

```
  IF Variable = 0 THEN Company
  IF Variable = 1 THEN Thermostat
  IF Variable = 2 THEN Stamp
```

```
Company:
```

```
  SEROUT 16, 84, [226,"00", "Amulet Technologies", NULL]
  GOTO serial_in
```

```
Thermostat:
```

```
  SEROUT 16, 84, [226,"01", "DS1620 Digital Thermostat", NULL]
  GOTO serial_in
```

```
Stamp:
```

```
  SEROUT 16, 84, [226,"02", "Basic Stamp 2 on Activity Board", NULL]
  GOTO serial_in
```

```
*****
'* Handler for a getLabel function request
'* Format of request string = 0xD3xx, where xx = index of label variable
'* Returns 0xE3xxString+Null to client
'* Returns requested data back to the screen
*****
```

```
getLabel:
```

```
  IF Variable = 0 THEN Cels
  IF Variable = 1 THEN Fahr
```

```
Cels:
```

```
  SEROUT 16, 84, [227,"00", "Temperature in Celsius", NULL]
  GOTO serial_in
```

```
Fahr:
```

```
  SEROUT 16, 84, [227,"01", "Temperature in Fahrenheit", NULL]
  GOTO serial_in
```

```

*****
* Handler for a setByte function request
* Format of request string = 0xD5xxNN, where xx = variable to be set
* and NN = HEX data
* Returns 0xE5xxNN to Amulet
*****

```

setByte:

```

IF Variable = 0 THEN setZeroByte
IF Variable = 1 THEN setOneByte

'set Byte to received value

'Byte 0 has been called to set
setZeroByte:
  B0.HIGHNIB = AmuletMsg(3)
  B0.LOWNIB = AmuletMsg(4)
  SEROUT 16, 84, [229,"00",HEX B0]
  GOTO serial_in

'Byte 1 has been called to set
setOneByte:
  B1.HIGHNIB = AmuletMsg(3)
  B1.LOWNIB = AmuletMsg(4)
  SEROUT 16, 84, [229,"01",HEX B1]
  GOTO serial_in

```

```

*****
* Handler for a setWord function request
* Format of request string = 0xD6xxPPNN, where xx = variable to be set
* PP = high byte of HEX data and NN = low byte of HEX data
* Returns 0xE6xxPPNN to Amulet
*****

```

setWord:

```

IF Variable = 0 THEN ZeroWord
IF Variable = 1 THEN OneWord

'set Word to received value

'Word 0 has been called to set
ZeroWord:
  W0.HIGHBYTE.HIGHNIB = AmuletMsg(3)   'store value received in W0
  W0.HIGHBYTE.LOWNIB = AmuletMsg(4)
  W0.LOWBYTE.HIGHNIB = AmuletMsg(5)
  W0.LOWBYTE.LOWNIB = AmuletMsg(6)
  SEROUT 16, 84, [230,"00", HEX W0]
  GOTO serial_in

'Word 1 has been called to set
OneWord:
  W1.HIGHBYTE.HIGHNIB = AmuletMsg(3)   'store value received in W1
  W1.HIGHBYTE.LOWNIB = AmuletMsg(4)
  W1.LOWBYTE.HIGHNIB = AmuletMsg(5)
  W1.LOWBYTE.LOWNIB = AmuletMsg(6)
  SEROUT 16, 84, [230,"01", HEX W1]
  GOTO serial_in

```

```

*****
* Handler for a setString function request
* Format of request string = 0xD7xxP{PNNQQZ}0, where xx = variable to be set
* followed by the HEX data (up to 7 bytes) and then a Null terminator
* Returns 0xE7xxP{PNNQQZ}0 to Amulet
*****

```

```

setString:
  IF Variable = 0 THEN ZeroString
  IF Variable = 1 THEN OneString

  Counter    VAR    Byte

  'Word 0 has been called to set
ZeroString:
  FOR Counter = 0 TO 2
    Str0(Counter) = AmuletMsg(Counter + 3)
  NEXT
  SEROUT 16, 84, [231,HEX2 Variable, STR Str0, NULL]
  GOTO serial_in

  'Word 1 has been called to set
OneString:
  FOR Counter = 0 TO 2
    Str1(Counter) = AmuletMsg(Counter + 3)
  NEXT
  SEROUT 16, 84, [231,HEX2 Variable, STR Str1, NULL]
  GOTO serial_in

'*****
'* Handler for a invoke function request
'* Format of request invoke RPC = 0xD8xx, where xx = RPC being requested
'* Returns 0xE8xx to client)
'*****

invokeRPC:
  'Invoke RPC reads the temp into DSdata in leaves it in Celsius if a "1" and
  'reads the temp into DSdata and stores it in Fahrenheit if a "2"

  HIGH RST                                'Activate the '1620
  SHIFTOUT DQ,CLK,LSBFIRST,[Rtemp]        'Request to read temperature
  SHIFTTIN DQ,CLK,LSBPRES,[DSdata\9]      'Get the temperature reading
  LOW RST                                  'Deactivate
  DSdata = DSdata/2                        'Scale reading to whole degrees C

  IF Variable = 0 THEN Celsius
  IF Variable = 1 THEN Fahrenheit

  'RPC 0 is being requested so set the temperature read by the thermostat
  'in degrees Celsius
Celsius:
  SEROUT 16, 84, [232, HEX2 Variable]
  GOTO serial_in

  'RPC 1 is being requested so set the temperature read by the thermostat
  'in degrees Fahrenheit
Fahrenheit:
  DSdata = (DSdata*$/01CC)                'Convert from Celsius to Fahrenheit
  DSdata = DSdata + 32                    '(multiply by 1.8, add 32)
  SEROUT 16, 84, [232, HEX2 Variable]
  GOTO serial_in

'*****
'* Handler for a getByteArr function request
'* Format of request byte arr = 0xDDxx where xx=the variable byte being used
'* Returns 0xEDxxNN..., where NN equals the HEX data of the variable xx
'*****

getByteArr:
  IF Variable = 0 THEN ByteArrZero
  IF Variable = 1 THEN ByteArrOne

  'Byte Array 0 has been called to set
ByteArrZero:

```

```

SEROUT 16, 84, [237,"00",HEX 2,HEX 4,HEX 6,HEX 8,NULL]      'sends out bytes 2,4,6,8
GOTO serial_in

'Byte 1 has been called to set
ByteArrOne:
SEROUT 16, 84, [237,"01",HEX 1,HEX 3,HEX 5,HEX 7,NULL]      'sends out bytes 1,3,5,7
GOTO serial_in

'*****
'* Handler for a getWordArr function request
'* Format of request word arr = 0xDExx, where xx = variable being requested
'* Returns 0xEExxPPNNPPNN., where PPNN equals each word of the HEX data of variable xx
'*****

getWordArr:
IF Variable = 0 THEN WordArrZero
IF Variable = 1 THEN WordArrOne

'Word Array 0 has been called to set
WordArrZero:
SEROUT 16, 84, [238,"00","2468","ACE0",NULL]      'sends out words 0x2468, 0xACE0
GOTO serial_in

'Word 1 has been called to set
WordArrOne:
SEROUT 16, 84, [238,"01","1357","9BDF",NULL]      'sends out words 0x1357, 0x9BDF
GOTO serial_in

END

```



Amulet Technologies, LLC

GUI Engines for Embedded Systems