UNIVERSITY OF MASSACHUSETTS LOWELL
Department of Electrical and Computer Engineering

Program 1	**16.472 / 16.572** **Embedded Real Time Systems**	**Spring 2014**

Wireless Sensor Network

The objective of this program is to implement the base station receiver for a wireless sensor network. In addition to the base station, there are multiple sensor nodes, each of which relays packets containing climactic data to the base station via the RF network. You are to write a C program to run on the STM32F107 micro-controller that accepts messages from the sensor nodes and display them on the screen. Your program must decode the incoming packets to interpret the messages.

PACKET STRUCTURE

Communication from the sensor nodes consists of message packets, which are transferred via the wireless network. The general packet format is shown below. The first 3 bytes of the packet constitute the packet preamble consisting of 3 "sync bytes." The preamble enables synchronization with the start of a new packet. In case of a bad packet, you find the beginning of the next packet by searching forward in the input stream until encountering the correct preamble.

Byte 3 (payload data byte 0) gives the packet length including the preamble, and the checksum. Next is a sequence of data bytes. Data byte 0 (payload data byte 1) gives the address of the destination node – your destination address is 1. Data byte 1 (payload data byte 2) gives the source node address. Data byte 2 (payload data byte 3) gives the message type. The last byte in the packet is a checksum byte, which is the XOR of all other bytes in the packet. Thus, the XOR of the entire packet, including the checksum, is zero.

You are to provide a packet parser whose job is to extract the payload from the packet as shown below. The payload is just the contents of the packet, excluding the preamble bytes and the checksum byte. Also, the length byte in the payload gives the number of bytes in the complete payload, including the header bytes and the data bytes.

Byte Location in Packet	Packet	Payload	Byte in Payload
0	0x03		
1	0xEF		
2	0xAF	Payload	Byte in Payload
3	Packet Length in Bytes	Payload Length in Bytes	0
4	packet data byte [0]	Destination Address	1
5	packet data byte [1]	Source Address	2
6	packet data byte [2]	Message Type	3
7	packet data byte [3]	payload data byte [0]	4
8	packet data byte [2]	payload data byte [1]	5
9	packet data byte [3]	payload data byte [2]	6
…	…	…	
Packet Length – 2	packet data byte [Packet Length – 5]	payload data byte [Payload Length – 4]	Payload Length - 1
Packet Length - 1	Checksum Byte (XOR of all bytes in the packet except this byte)		

SPECIFIC MESSAGE PACKETS

The 8 message packets that your program must process are described below:

Message 0x01 - "Temperature Message"

The temperature message reports temperature -100 ˚F ≤ T ≤ +120˚ F.

Byte Location in Payload	Value
3	0x01
4	8-bit signed temperature

Message 0x02 - "Barometric Pressure Message"

Report the barometric pressure 800 ≤ P ≤ 1100 millibars.

Byte Location in Payload	Value
3	0x02
4	Most significant byte of 16-bit unsigned pressure
5	Least significant byte of 16-bit unsigned pressure

Message 0x03 - "Humidity Message"

The humidity message reports the humidity 0 ≤ H ≤ 100%, and dew point -100 ˚F ≤ D ≤ +120 ˚F.

Byte Location in Payload	Value
3	0x03
4	8-bit signed dew point
5	8-bit unsigned humidity

Message 0x04 - "Wind Message"

Report the wind direction 0 ≤ W ≤ 359 degrees, and the wind speed 0 ≤ S ≤ 300.0 mph *in tenths of a mph*.

Byte Location in Payload	Value	
3	0x04	
4	7 4	3 0
	BCD 10^2 digit of wind speed	BCD 10^1 digit of wind speed
5	7 4	3 0
	BCD 10^0 digit of wind speed	BCD 10^{-1}digit of wind speed
6	Most significant byte of 16-bit unsigned wind direction	
7	Least significant byte of 16-bit unsigned wind direction	

Message 0x05 - "Solar Radiation Intensity Message"

Report the solar radiation intensity 0 ≤ R ≤ 1300 watts per square meter.

Byte Location in Payload	Value
3	0x05
4	Most significant byte of 16-bit unsigned radiation intensity
5	Least significant byte of 16-bit unsigned radiation intensity

Message 0x06 - "Date/Time Message"

Report the date and time.

Byte Location in Payload	Value
3	0x06
4	Most Significant Byte of 32-bit Packed Date/Time
5	Second Most Significant Byte of 32-bit Packed Date/Time
6	Third Most Significant Byte of 32-bit Packed Date/Time
7	Least Significant Byte of 32-bit Packed Date/Time

32-Bit Packed Date/Time Format

Bytes 1 - 4 of a Date/Time Message contain a calendar date and a time of day packed into 32-bits as follows:

31 27	26 21	20 9	8 5	4 0
Hour (0-23)	Minute (0-59)	Year (0-4095)	Month 1-12	Day (1-31)

Message 0x07 – "Precipitation Message"

Report today's precipitation depth $0.00 \leq P \leq 20.00$ inches *in hundredths of an inch*.

Byte Location in Payload	Value	
3	0x07	
4	7 4	3 0
	BCD 10^1 digit of depth	BCD 10^0 digit of depth
5	7 4	3 0
	BCD 10^{-1} digit of depth	BCD 10^{-2} digit of depth

Message 0x08 - "ID Message"

Report the ID of the node. The ID is a sequence of up to 10 ASCII characters.

Byte Location in Payload	Value
3	0x08
4	ID Character 0
5	ID Character 1
6	ID Character 2
	…
N+3	ID Character N-1

Exception Handling

The table below lists some possible communications errors and exceptions that may occur. Your program is to detect and report such exceptions, and then continue processing at the next packet.

DESCRIPTION / MESSAGE
Preamble Byte 1 Error
Preamble Byte 2 Error
Preamble Byte 3 Error
Checksum Error
Packet Length < 8
Destination Address ≠ 1
Unknown Message Type

Packet Source

Since we actually don't have an RF link for this program, the bytes of the message packets will be read in one at a time from the STM32F107's RS232 receiver (Rx). The packets will be sent by a PC application called COMAPP.EXE.

SAMPLE RUN 1

```
What COM Port (1<=port<=16, or ENTER to abort)? 1

Packet File Name [pkts.dat]  or term or '.' to quit:


Delay Factor [DF=100] 0 = Manual Mode or '.' to quit:

SOURCE NODE 2: TEMPERATURE MESSAGE
  Temperature = -16

SOURCE NODE 3: BAROMETRIC PRESSURE MESSAGE
  Pressure = 1025

SOURCE NODE 4: HUMIDITY MESSAGE
  Dew Point = -65 Humidity = 16

SOURCE NODE 5: WIND MESSAGE
  Speed = 214.3 Wind Direction = 257

SOURCE NODE 6: SOLAR RADIATION MESSAGE
  Solar Radiation Intensity = 513

SOURCE NODE 7: DATE/TIME STAMP MESSAGE
  Time Stamp = 1/24/2013 6:30

SOURCE NODE 8: PRECIPITATION MESSAGE
  Precipitation Depth = 12.34


Packet File Name [pkts.dat]  or '.' to quit:


Delay Factor [DF=100] 0 = Manual Mode or '.' to quit:

SOURCE NODE 9: SENSOR ID MESSAGE
  Node ID = Node-9

SOURCE NODE 2: TEMPERATURE MESSAGE
  Temperature = -16

SOURCE NODE 3: BAROMETRIC PRESSURE MESSAGE
  Pressure = 1025

SOURCE NODE 4: HUMIDITY MESSAGE
  Dew Point = -65 Humidity = 16

SOURCE NODE 5: WIND MESSAGE
  Speed = 214.3 Wind Direction = 257

SOURCE NODE 6: SOLAR RADIATION MESSAGE
  Solar Radiation Intensity = 513

SOURCE NODE 7: DATE/TIME STAMP MESSAGE
  Time Stamp = 1/24/2013 6:30

SOURCE NODE 8: PRECIPITATION MESSAGE
  Precipitation Depth = 12.34


Packet File Name [pkts.dat]  or '.' to quit:
```

SAMPLE RUN 2 (WITH PACKET ERRORS)

```
Packet File Name [pkts.dat]  or term or '.' to quit: errs.dat


Delay Factor [DF=50] 0 = Manual Mode or '.' to quit: 100

SOURCE NODE 2: TEMPERATURE MESSAGE
  Temperature = -16
*** ERROR: Bad Preamble Byte 1

SOURCE NODE 4: HUMIDITY MESSAGE
  Dew Point = -65 Humidity = 16
*** ERROR: Bad Preamble Byte 2

SOURCE NODE 6: SOLAR RADIATION MESSAGE
  Solar Radiation Intensity = 513

SOURCE NODE 7: DATE/TIME STAMP MESSAGE
  Time Stamp = 1/24/2013 6:30
*** ERROR: Bad Preamble Byte 3

SOURCE NODE 9: SENSOR ID MESSAGE
  Node ID = Node-9
*** ERROR: Checksum error

SOURCE NODE 3: BAROMETRIC PRESSURE MESSAGE
  Pressure = 1025
*** ERROR: Bad Packet Size

SOURCE NODE 5: WIND MESSAGE
  Speed = 214.3 Wind Direction = 257
*** ERROR: Unknown Message Type

SOURCE NODE 7: DATE/TIME STAMP MESSAGE
  Time Stamp = 1/24/2013 6:30
*** INFO: Not My Address

SOURCE NODE 9: SENSOR ID MESSAGE
  Node ID = Node-9


Packet File Name [errs.dat]  or '.' to quit:
```

REQUIRED FILES

Your program must be divided up into separate modules as described below:

FILE	CONTENTS
Prog1.c	This is the module contains the main program and any functions not defined in the modules described below.
PktParser.c PktParser.h	This module contains the function ParsePkt() and any needed auxiliary functions
Error.c Error.h	This module provides a function that other modules may call to display error messages in a consistent format.
CPU.h	Defines size specific types CPU_INT08U, CPU_INT08S, etc.
BSP.h BSP.c	This module, which is supplied with the DateDemo download, provides board support to initialize the Micrium Eval. Board.
BSP_Ser.h BSP_Ser.c	This is the RS232 driver module.
BSP_Periph.h BSP_Periph.c	This module provides STM32F107 peripheral device initialization functions.

REQUIRED TYPE DEFINITIONS

Below are two required type definitions that must be used in the indicated files:

PktParser.c

```
typedef struct
{
   CPU_INT08U  payloadLen;
   CPU_INT08U  data[1];
} PktBfr;
```

Prog1.c

```
#pragma pack(1) // Don't align on word boundaries
typedef struct
{
   CPU_INT08U              payloadLen;
   CPU_INT08U              dstAddr;
   CPU_INT08U              srcAddr;
   CPU_INT08U              msgType;
   union
      {
         CPU_INT08S        temp;
         CPU_INT16U        pres;
         struct
            {
            CPU_INT08S     dewPt;
            CPU_INT08U     hum;
            } hum;
         struct
            {
            CPU_INT08U     speed[2]; //See Note Below
            CPU_INT16U     dir;
            } wind;
         CPU_INT16U        rad;
         CPU_INT32U        dateTime;
         CPU_INT08U        depth[2];      //See Note Below
         CPU_INT08U        id[10];        //See Note Below
      } dataPart;
} Payload;
NOTE: Do not use hard-coded constants.
```

REQUIRED FUNCTION

Your module PktParser.c must define the following function:

Function Prototype	
`void ParsePkt(void *pktBfr);`	
Purpose	
Read one packet from the Rx, extracting and returning the packet payload.	
Output Parameters	
pktBfr	A pointer to the returned payload

PROGRAM SUBMISSION REQUIREMENTS

Hand in the items 1 and 2 at the beginning of class on the due date:

1. A computer printout of the source files in the following order: "Prog1.c," "PktParser.h," "PktParser.c," "Error.h," and "Error.c."
2. A state diagram showing the algorithm used by your packet parser.
3. Computer printouts of two sample runs using the packet files "pkts.dat" and "errs.dat".

Zip up the three source files from item 1. Name the zip file LASTNAME.zip, and submit it via blackboard. These source files must be identical to the printouts from item 1. If you make changes, then take back your submission and submit the changed version. Make sure that your zip file includes only the requested files and no folders.

NOTE: Programs received after 6:30 PM on the due date will be considered late.

OTHER REQUIREMENTS

1. You must use the required types exactly as defined above.
2. Do not use real data types (float, double, or long double).
3. Use only the CPU-specific data types defined in the header file "CPU.h." Do not use C's built-in types like int, char, unsigned, etc.
4. Your Packet Parser module must be completely generic, having no knowledge of the contents of a packet. It sees the packet contents simply as a sequence of bytes.
5. Prefix error messages with the ASCII Alarm character '\a' e.g. BSP_Ser_Printf ("\a*** ERROR: Checksum Error")
6. You are required to use bit-wise and shift operators for bit manipulation, e.g., to extract the fields from packed date/time value. *Bit fields are not allowed. You are also not allowed to use arithmetic operators (use "|," not "+;" Use "<<" abd ">>," not "*," "/," or "%."*
7. Make the filename input robust. If a bad file name is entered, display an error message and ask the user to reenter the name. Allow the user to keep trying until he succeeds, or cancels (see #8 below).
8. Make file name input friendly. Allow the user to cancel and exit the program by supplying an empty line in response to the file name prompt (an empty line means that the user presses <ENTER> only).
9. Your program's output must appear exactly as shown in the sample run, producing exactly the same output format.
10. Use `#define` directives to define symbolic constants rather than hard-coding constants into your program. For example specify array dimensions (size) with a symbolic constant. Then, it is easy to change the sizes of all of the arrays.
11. Use descriptive names throughout your program. Strive to make your code so readable that is self-explanatory.
12. Use a consistent indentation scheme to help show the nested structure of your program.
13. Use a consistent capitalization convention to distinguish variables from constants. The preferred convention is that constant names, type names, and function names have the first letter of each word capitalized (e.g., HeaderLength, Payload, PktParser()), and variable names, member names, and parameter names, have the first letter of every word except the first capitalized (e.g., bfr, dataLen, payloadBfr). Thus, constants, types, and functions always start with a capital letter, and variables, members, and parameters start with lower case. The point of this convention is to be able to distinguish "l-values" (can appear on the left side of assignment) from "r-values."
14. Include a sufficient number of comments to explain your program. See your instructor's sample code for an example of acceptable commenting.
15. Do not use goto statements.
16. Do not use global variables.
17. No function may be longer than one page (60 lines).
18. Do not use redundant code. If the same sequence of statements is needed in more than one place, make it a function.
19. Keep your functions simple. Overly complex solutions will be penalized. If your program seems overly complex, it probably is. This means that you did not spend enough time designing, before you started typing code. KEEP IT SIMPLE!
20. Do not use malloc(), free(), or any similar dynamic memory allocation.
21. Eliminate all compiler-warning messages.