```
/*
        C program to simulate receiver side logic to decipher:
        1. Frame is valid or not
        2. Type of frame (I-frame, S-frame, U-frame)
        3. Data
        4. Checksum bits  ( Assuming only 2-byte)
        5. code bits
        6. Sequence numbers
        7. P/F bit values
        8. Addresses, single byte or multi-byte
*/

-----------------------------------------------------------------------------------------------------

#include<stdio.h>

#include<malloc.h>

// If any of the values of valid array becomes 0, the data is valid

static int valid[3]={1,1,1};          //Flag1, Address, Flag2


static int vData = 1;          //Data will be present for I-frame and U-frame


/* IDENTIFYING THE ADDRESS TYPE */
int Address(int bit[],int len)

{
        /*  As the address ends with bit 1 at the last position in case of both single-byte and
multi-byte addresses, we check for the last bit position excluding the flag bits (8)+first 7 bits
of the address field(7)=14 i.e;(0-14)bit positions. */

        int i=15,c=1;

        while(bit[i] == 0 && i<len)

        {

                i+=8;

                c++;

        }

        if(bit[i] == 0 && !(i<len))          // Invalid Address means it shouldn't end with 1
```

```
            valid[1]=0;
        return c;
}
```

/* IDENTIFYING THE TYPE OF FRAME */

```
char Frame(int bit[],int bytes)
{
```

/* The structures of S,I and U-frames vary based on the bit positions in the control field as : Frame Control Field

(0 1 2 3 4 5 6 7)

I-frame : 0 .....

S-frame : 1 0 .....

U-frame : 1 1 ..... */

```
        int p = 8*(bytes+1);        /* Since the control field begins at the end of the
                                       Address field.*/
        if(bit[p] == 0)
            return 'I';
        else if(bit[p] == 1 && bit[p+1] == 0)
            return 'S';
        else
            return 'U';
}
```

/* IDENTIFYING DATA BITS IN THE BIT STREAM */

```
int Data(int bit[],char fr,int bytes,int len)
{
```

/* If the frame is S-frame, then there is no data.
If the frame is I-frame, User information exists.
If the frame is U-frame, System information exists.
If any kind of information exists, it begins at the end of control field. */

```
        int i=0,p = bytes+2,q = 8*p;
```

```
if(fr == 'S')
{
        if (len > (8*p)+24 )        /*  8 * (start_flag + control + address bytes) +
                                        cksum(16) + end_flag   */
        {
          vData = 0;
          return i;
        }
        else
                printf("\nNo Data Bits");
}
else if(fr == 'I')
{
        if (len <= (8*p)+24 )       /*  8 * (start_flag + control + address bytes) +
                                        cksum(16) + end_flag   */
        {
          vData = 0;
          return i;
        }
        printf("\nUser Information is:");
        for(i=q;i<len-24;i++)
                printf("%d",bit[i]);
}
else
{
        if (len <= (8*p)+24 )       /*  8 * (start_flag + control + address bytes) +
                                        cksum(16) + end_flag   */
```

```
                {
                  vData = 0;
                  return i;
                }
                printf("\nSystem Information is:");
                for(i=q;i<len-24;i++)
                        printf("%d",bit[i]);
        }
   return i;
}
```

/* IDENTIFYING CHECKSUM */

```
void CkSum(int bit[],char fr,int bytes,int len,int d)
{
```

/* Assuming that the Checksum is only 2-byte long, we make put the condition that the Checksum ends at totalLength-8.
We check the same for S,I,U-frames. */

```
        int i,p = bytes+2;
        printf("\nChecksum Bits: ");
        if(fr=='S')
        {
                for(i=8*p;i<len-8;i++)
                        printf("%d",bit[i]);
        }
        else
                for(i=d;i<len-8;i++)
                        printf("%d",bit[i]);
```

```
}

/* IDENTIFYING CODE BITS */
void Code(int bit[],char fr,int bytes)

{
        /* If the frame is I-frame, no code bits will be present.
          Otherwise, the code bits are:

          Frame                    CODE BITS in Control Field
                                        ( 0 1 2 3 4 5 6 7 )
          S-frame:                        - -
          U-frame:                        - -  - - -
         */

        int p = 8*(bytes+1);

        if(fr == 'I')

                printf("\nNo Code Bits");

        else if(fr == 'S')

                printf("\nCode Bits are: %d %d",bit[p+2],bit[p+3]);

        else

                printf("\nCode Bits are: %d %d %d %d
%d",bit[p+2],bit[p+3],bit[p+5],bit[p+6],bit[p+7]);

}

/* IDENTIFYING P/F BIT */

void PFval(int bit[],int bytes,char fr)

{
        /*The P/F bit in all the three frames is:
          Frame          P/F in Control Field..
                          ( 0 1 2 3 4 5 6 7 )
          S (or) I (or) U:              -
        */


        int p = 8*(bytes+1);
```

```
        printf("\nP/F Value = %d",bit[p+4]);
}

/* IDENTIFYING SEQUENCE NUMBER */
void Seq(int bit[],char fr,int bytes)
{
        /* If the frame is S-frame or U-frame, Sequence bits are absent.
           If the frame id I-frame, the position of Sequence bits are given by:
           Bit Positions in Control Field:  0 1 2 3 4 5 6 7
                                                - - -
        */
        int p=8*(bytes+1);
        if(fr == 'S' || fr == 'U')
                printf("\nNo Sequence Number");
        else
                printf("\nSequence Number (Binary format) is:
%d%d%d",bit[p+1],bit[p+2],bit[p+3]);
}

/* CHECKING THE VALIDITY OF THE FRAME */
 /* A frame is invalid when:
   1. S-Frame contains data
   2. I or U-Frame doesn't contain data
   3. Either a single byte or a multi bytes address doesn't terminate with a 1
   4. When the flag bits are not properly placed
   5. When Checksum at sender and receiver doesn't match
   (However, we doesn't check the checksums here because we don't know the generator as
well as checksum at sender side)
 */
 int Valid(int *bit)
 {
        if(valid[0] == 0 || valid[1] == 0 || valid[2] == 0 || vData == 0)
        {
```

```
                printf("\nGIVEN FRAME IS INVALID");
                return 0;
        }
        else
        {
                printf("\nGIVEN FRAME IS VALID");
                return 1;
        }
}

/* CHECKING START_FLAG */
 void Flag1(int *bit)
{
        int i=0;
        if(bit[0] != 0 || bit[7] != 0)
                valid[0]=0;
        for(i=1;i<7;i++)
                if(bit[i] != 1)
                {
                        valid[0]=0;
                        break;
                }
}

/* CHECKING END_FLAG */
 void Flag2(int *bit,int len)
```

```
{
        int i=len;
        if(bit[len-1] != 0 || bit[len-8] != 0)
                valid[2]=0;
        for(i=len-7;i<len-1;i++)
                if(bit[i] != 1)
                {
                        valid[2]=0;
                        break;
                }
}

/* MAIN BEGINS */

int main()
{
        int len,bytes,i,d,*bit,validity=0;
        char fr;
        //Assuming that the receiver received the following bit stream
        printf("\nEnter the length of the Bit Stream received: ");
        scanf("%d",&len);
        bit=(int *)malloc(len * sizeof(int));
        printf("\nEnter the Bit Stream:\n");
        for(i=0;i<len;i++)
                scanf("%d",&bit[i]);

        Flag1(bit);
```

```
        Flag2(bit,len);
        bytes = Address(bit,len);          //returns the type of address to 'bytes'
        fr = Frame(bit,bytes);             //returns a character indicating frame type
        d = Data(bit,fr,bytes,len);
        validity = Valid(bit);
        if(validity != 0)
        {
                printf("\nThe frame is: %c",fr);          //displays data if any
                CkSum(bit,fr,bytes,len,d);        //prints checksum in binary format
                Code(bit,fr,bytes);          //prints code bits if any
                Seq(bit,fr,bytes);           //prints sequence number if any
                PFval(bit,bytes,fr);         //prints P/F bit
                printf("\n%d byte Address",bytes);
        }
        return 0;
}
```

OUTPUT:

Success #stdin #stdout 0s 10320KB comments (0)

stdin copy

```
56
0 1 1 1 1 1 1 0  0 0 0 0 0 1 1 1  0 1 1 1 0 1 1 1  1 0 1 0 1 0 1 1  0 0 0 0 0 1 0 1  0 1 1 0 1 0 1 0
0 1 1 1 1 1 1 0
```

stdout copy

```
Enter the length of the Bit Stream received:
Enter the Bit Stream:

User Information is:10101011
GIVEN FRAME IS VALID
The frame is: I
Checksum Bits: 0000010101101010
No Code Bits
Sequence Number (Binary format) is: 111
P/F Value = 0
1 byte Address
```

Success #stdin #stdout 0s 4360KB comments (0)

stdin copy

```
64
0 1 1 1 1 1 1 0  0 0 0 1 1 1 1 0  0 0 0 0 0 1 1 1  0 1 1 1 0 1 1 1  1 0 1 0 1 0 1 1  0 0 0 0 0 1 0 1
0 1 1 0 1 0 1 0  0 1 1 1 1 1 1 0
```

stdout copy

```
Enter the length of the Bit Stream received:
Enter the Bit Stream:

User Information is:10101011
GIVEN FRAME IS VALID
The frame is: I
Checksum Bits: 0000010101101010
No Code Bits
Sequence Number (Binary format) is: 111
P/F Value = 0
2 byte Address
```

Success #stdin #stdout 0s 4508KB comments (0)

stdin copy

```
72
0 1 1 1 1 1 1 0  0 0 0 1 1 1 1 0  0 0 0 0 1 0 1 0  0 0 0 0 0 1 1 1  0 1 1 1 0 1 1 1  1 0 1 0 1 0 1 1
0 0 0 0 0 1 0 1  0 1 1 0 1 0 1 0  0 1 1 1 1 1 1 0
```

stdout copy

```
Enter the length of the Bit Stream received:
Enter the Bit Stream:

User Information is:10101011
GIVEN FRAME IS VALID
The frame is: I
Checksum Bits: 0000010101101010
No Code Bits
Sequence Number (Binary format) is: 111
P/F Value = 0
3 byte Address
```

Success #stdin #stdout 0s 9432KB comments (0)

stdin copy

```
72
0 1 1 1 1 1 1 0  0 0 0 1 1 1 1 0  0 0 0 0 1 0 1 0  0 0 0 0 0 1 1 1  1 1 1 1 0 1 1 1  1 0 1 0 1 0 1 1
0 0 0 0 0 1 0 1  0 1 1 0 1 0 1 0  0 1 1 1 1 1 1 0
```

stdout copy

```
Enter the length of the Bit Stream received:
Enter the Bit Stream:

System Information is:10101011
GIVEN FRAME IS VALID
The frame is: U
Checksum Bits: 0000010101101010
Code Bits are: 1 1 1 1 1
No Sequence Number
P/F Value = 0
3 byte Address
```

Success #stdin #stdout 0s 9432KB comments (0)

stdin copy

```
64
0 1 1 1 1 1 1 0  0 0 0 1 1 1 1 0  0 0 0 0 0 1 1 1  1 1 1 1 0 1 1 1  1 0 1 0 1 0 1 1  0 0 0 0 0 1 0 1
0 1 1 0 1 0 1 0  0 1 1 1 1 1 1 0
```

stdout copy

```
Enter the length of the Bit Stream received:
Enter the Bit Stream:

System Information is:10101011
GIVEN FRAME IS VALID
The frame is: U
Checksum Bits: 0000010101101010
Code Bits are: 1 1 1 1 1
No Sequence Number
P/F Value = 0
2 byte Address
```

Success #stdin #stdout 0s 4312KB comments (0)

stdin copy

```
56
0 1 1 1 1 1 1 0  0 0 0 0 0 1 1 1  1 1 1 1 0 1 1 1  1 0 1 0 1 0 1 1  0 0 0 0 0 1 0 1  0 1 1 0 1 0 1 0
0 1 1 1 1 1 1 0
```

stdout copy

```
Enter the length of the Bit Stream received:
Enter the Bit Stream:

System Information is:10101011
GIVEN FRAME IS VALID
The frame is: U
Checksum Bits: 0000010101101010
Code Bits are: 1 1 1 1 1
No Sequence Number
P/F Value = 0
1 byte Address
```

Success #stdin #stdout 0s 9432KB comments (0)

stdin copy

```
48
0 1 1 1 1 1 1 0  0 0 0 0 0 1 1 1  1 0 1 1 0 1 1 1  0 0 0 0 0 1 0 1  0 1 1 0 1 0 1 0  0 1 1 1 1 1 1 0
```

stdout copy

```
Enter the length of the Bit Stream received:
Enter the Bit Stream:

No Data Bits
GIVEN FRAME IS VALID
The frame is: S
Checksum Bits: 0000010101101010
Code Bits are: 1 1
No Sequence Number
P/F Value = 0
1 byte Address
```

Success #stdin #stdout 0s 4532KB comments (0)

stdin copy

```
56
0 1 1 1 1 1 1 0  0 0 0 0 0 1 1 0  0 0 0 0 0 1 1 1  1 0 1 1 0 1 1 1  0 0 0 0 0 1 0 1  0 1 1 0 1 0 1 0
0 1 1 1 1 1 1 0
```

stdout copy

```
Enter the length of the Bit Stream received:
Enter the Bit Stream:

No Data Bits
GIVEN FRAME IS VALID
The frame is: S
Checksum Bits: 0000010101101010
Code Bits are: 1 1
No Sequence Number
P/F Value = 0
2 byte Address
```

Success #stdin #stdout 0s 9432KB comments (0)

stdin copy

```
64
0 1 1 1 1 1 1 0  0 0 0 0 0 1 1 0  0 0 1 1 0 0 1 0  0 0 0 0 0 1 1 1  1 0 1 1 0 1 1 1  0 0 0 0 0 1 0 1
0 1 1 0 1 0 1 0  0 1 1 1 1 1 1 0
```

stdout copy

```
Enter the length of the Bit Stream received:
Enter the Bit Stream:

No Data Bits
GIVEN FRAME IS VALID
The frame is: S
Checksum Bits: 0000010101101010
Code Bits are: 1 1
No Sequence Number
P/F Value = 0
3 byte Address
```

Success #stdin #stdout 0s 4508KB comments (0)

stdin copy

```
64
0 1 1 1 1 1 1 0  0 0 0 0 0 1 1 0  0 0 1 1 0 0 1 0  0 0 0 0 0 1 1 0  1 0 1 1 0 1 1 0  0 0 0 0 1 0 0 0
0 1 1 0 1 0 1 0  0 1 1 1 1 1 1 0
```

stdout copy

```
Enter the length of the Bit Stream received:
Enter the Bit Stream:

GIVEN FRAME IS INVALID
```

Success #stdin #stdout 0s 9432KB comments (?)

stdin copy

```
56
0 1 1 1 1 1 1 0  0 0 0 0 0 1 1 1  1 0 1 1 0 0 1 0  1 0 1 1 0 1 1 0  0 0 0 0 0 1 0 0  0 1 1 0 1 0 1 0
  0 1 1 1 1 1 1 0
```

stdout copy

```
Enter the length of the Bit Stream received:
Enter the Bit Stream:

GIVEN FRAME IS INVALID
```

Success #stdin #stdout 0s 4524KB comments (0)

stdin copy

```
48
0 1 1 1 1 1 1 0  0 0 0 0 0 1 1 1  0 0 1 1 0 0 1 0  0 0 0 0 0 1 0 0  0 1 1 0 1 0 1 0  0 1 1 1 1 1 1 0
```

stdout copy

```
Enter the length of the Bit Stream received:
Enter the Bit Stream:

GIVEN FRAME IS INVALID
```

Success #stdin #stdout 0s 4172KB comments (0)

stdin copy

```
48
0 1 1 1 1 1 1 0  0 0 0 0 0 1 1 0  0 0 1 1 0 0 1 0  0 0 0 0 0 1 0 0  0 1 1 0 1 0 1 0  0 1 1 1 1 1 1 0
```

stdout copy

```
Enter the length of the Bit Stream received:
Enter the Bit Stream:

GIVEN FRAME IS INVALID
```

Success #stdin #stdout 0s 4252KB comments (0)

stdin copy

```
48
0 1 1 1 1 1 1 0  0 0 0 0 0 1 1 1  1 1 0 1 0 0 1 0  0 0 0 0 0 1 0 0  0 1 1 0 1 0 1 0  0 1 1 1 1 1 1 0
```

stdout copy

```
Enter the length of the Bit Stream received:
Enter the Bit Stream:

GIVEN FRAME IS INVALID
```

Success #stdin #stdout 0s 9432KB comments (0)

stdin copy

```
48
0 1 1 1 1 1 1 0  0 0 0 0 0 1 1 0  1 1 0 1 0 0 1 0  0 0 0 0 0 1 0 0  0 1 1 0 1 0 1 0  0 1 1 1 1 1 1 0
```

stdout copy

```
Enter the length of the Bit Stream received:
Enter the Bit Stream:

GIVEN FRAME IS INVALID
```