Calculating the checksum of an ASTM document

Environment: C#, VB, C++ .NET

Using the ASTM protocol to communicate with a laboratory instrument can be an interesting project, and one of the finer points of ASTM is calculating the checksum on transmitted and received documents. Each segment (row of message) of data includes a checksum when either the laboratory instrument or the LIS (Laboratory Information System) transmits the document. The benefits are obvious in a mission (life) critical application to ensure what was sent is exactly what is received.

Following is an example of a frame of ASTM data transmitted from a laboratory instrument:

```
□5R|2|^^^1.0000+950+1.0|15|||^5^||V||34001637|20080516153540|20080516153602|34001637\r□3D\r\n
```

The format of an ASTM frame: **<STX><Frame Data><CR><ETX><CHECKSUM 1><CHECKSUM 2><CR><LF>**

The frame starts with a <STX> character (0x02) and ends with a <ETX> character (0x03), or a <ETB> if a partial frame. Just prior to the <ETX> is a carriage return \r (no newline character), **The C# escape sequences \r and \n are used purely to be able to display the string in a web page, but you will have to replace the \r with an ASCII 13 or HEX 0D. In addition the trailing \r\n characters must be replaced with ASCII 13 and ASCII 10 or HEX 0D 0A**. That being said you can run the string above through the C# method shown below and the escape sequences will be translated correctly.

All of the characters AFTER the <STX> and up to and including the <ETX> (or <ETB>) character is the data used to calculate the checksum for this message. The following C# method calculates the checksum by adding the ASCII values of each character in the string and then performing a mod 256 calculation on the total value for all characters in the ASTM frame. The checksum is then appended to the end of the message, and in this case the value is 3D. Remember the checksum is displayed as a 2 byte HEX string of characters.

Following is the HEX representation of the same ASTM frame with the control and checksum characters noted:

Address	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	Dump
	STX																
00000000	02	35	52	7c	32	7c	5e	5e	5e	31	2e	30	30	30	30	2b	.5R|2|^^^1.0000+
00000010	39	35	30	2b	31	2e	30	7c	31	35	7c	7c	7c	5e	35	5e	950+1.0|15|||^5^
00000020	7c	7c	56	7c	7c	33	34	30	30	31	36	33	37	7c	32	30	||V||34001637|20
00000030	30	38	30	35	31	36	31	35	33	35	34	30	7c	32	30	30	080516153540|200
00000040	38	30	35	31	36	31	35	33	36	30	32	7c	33	34	30	30	80516153602|3400
00000050	31	36	33	37	0d	03	33	44	0d	0a							1637..3D..
					CR	ETX	CHK	CHK	CR	LF							

Important: The maximum character limit for a frame in ASTM is 247 characters (including overhead). If a given message is less than 240 characters, it is sent in an end frame with a <ETX>, checksum, and a <CR><LF>. If a message is greater than 240 characters, the message is sent in intermediate frames containing this sequence <ETB>, checksum, and a <CR><LF>.

To use this method you must include the following classes in the project

```
public class T
{
    public const byte ENQ = 5;
    public const byte ACK = 6;
    public const byte NAK = 21;
    public const byte EOT = 4;
    public const byte ETX = 3;
    public const byte ETB = 23;
    public const byte STX = 2;
    public const byte NEWLINE = 10;
    public static byte[] ACK_BUFF = { ACK };
    public static byte[] ENQ_BUFF = { ENQ };
    public static byte[] NAK_BUFF = { NAK };
    public static byte[] EOT_BUFF = { EOT };
}
```

The class T above defines most of the control characters required to communicate in ASTM protocol. Following is a simple C# method to calculate the checksum

of an ASTM frame of data.

```
/// <summary>
/// Reads checksum of an ASTM frame, calculates characters after STX,
/// up to and including the ETX or ETB. Method assumes the frame contains an ETX or ETB.
/// </summary>
/// <param name="frame">frame of ASTM data to evaluate</param>
/// <returns>string containing checksum</returns>
public string GetCheckSumValue(string frame)
{
    string checksum = "00";

    int byteVal = 0;
    int sumOfChars = 0;
    bool complete = false;

    //take each byte in the string and add the values
    for (int idx = 0; idx < frame.Length; idx++)
    {
        byteVal = Convert.ToInt32(frame[idx]);

        switch (byteVal)
        {
            case T.STX:
                sumOfChars = 0;
                break;
            case T.ETX:
            case T.ETB:
                sumOfChars += byteVal;
                complete = true;
                break;
            default:
                sumOfChars += byteVal;
                break;
        }
```

```
        if (complete)
            break;
    }

    if (sumOfChars > 0)
    {
        //hex value mod 256 is checksum,
        //return as hex value in upper case
        checksum = Convert.ToString(sumOfChars % 256, 16).ToUpper();
    }

    //if checksum is only 1 char then prepend a 0
    return (checksum.Length == 1 ? checksum.PadLeft("0", 2) : checksum);
}
```