

```
1 Imports System
2 Imports System.Collections.Generic
3 ''' <summary>
4 ''' TOS6200シリーズ 機種依存クラスモジュール
5 ''' 2022.02.01 Ver1.00 SE 矢島
6 ''' Copyright by Kikusui Electronics Corp. 2022
7 ''' </summary>
8 Public Class ClsTos6200
9     '-----
10    '内部変数
11    '-----
12    Private DSR_Mnemonic As String 'DSR?の戻り値をビット演算した結果の文字列
13    Private DSR_Value As Integer   'DSR?の戻り値(数値)
14
15    '-----
16    'プロパティ
17    '-----
18    Public ReadOnly Property TestStatus() As String
19        Get
20            Return DSR_Mnemonic
21        End Get
22    End Property
23    Public ReadOnly Property DsrValue() As Integer
24        Get
25            Return DSR_Value
26        End Get
27    End Property
28
29    '-----
30    '関数
31    '-----
32    ''' <summary>
33    ''' DSR?の戻り値を文字列に変換してプロパティで読めるようにする
34    ''' </summary>
35    ''' <param name="iRet">DSR?のリードバック値</param>
```

```
36 Private Sub SetProp_DSR(iRet As Integer)
37
38     'DSR?の値をビット演算してモニタで返す
39     '一番上位に立っているビットが最終的に有効になる。
40     '例：ReadyでProtectionだと返されるのはProtectionになる。
41     '複数の値が立っていることを確認したい場合はDsrValueプロパティの戻り値を
42     If iRet And 1 Then DSR_Mnemonic = "Ready"
43     If iRet And 2 Then DSR_Mnemonic = "Inv Set"
44     If iRet And 4 Then DSR_Mnemonic = "Test"
45     If iRet And 8 Then DSR_Mnemonic = "Test On"
46     If iRet And 16 Then DSR_Mnemonic = "Pass"
47     If iRet And 32 Then DSR_Mnemonic = "Fail"
48     If iRet And 64 Then DSR_Mnemonic = "STOP"
49     If iRet And 128 Then DSR_Mnemonic = "Protection"
50
51     'DsrValueプロパティに値をセットする
52     DSR_Value = iRet
53
54 End Sub
55
56 ''' <summary>
57 ''' 試験終了を待つMON?で試験結果を取得する(非同期動作)
58 ''' </summary>
59 ''' <returns></returns>
60 Public Async Function GetResult_ASync(instr As ClsVisaCommunication, token As System.Threading.CancellationToken) As Task >
    (Of String)
61
62     Dim iDSR As Integer           'TESTステータスレジスタの値(Integer変換後)
63     Dim sMON() As String         'MON?応答の配列(文字列)
64     Dim sResult As String        '関数戻り値
65
66     Try
67
68         'ループ1 試験が開始されたことを確認するループ
69         Do
```

```
70      Await Task.Delay(100).ConfigureAwait(False)      '100ms待つ
71
72      iDSR = instr.VisaGetQuery_int("DSR?")      'DSR?でデータステータスレジスタの状態を取得する
73      SetProp_DSR(iDSR)      'DSR?の戻り値をビット演算してプロパティで読めるようにする
74
75      'DSR?の戻り値をビット演算する。
76      '試験が開始されていたらループを抜けて次のループに移る
77      If iDSR And 4 Then
78          Exit Do
79
80      End If
81
82      'タスクキャンセル要求が入っていたら測定停止後Cancelを返して返して関数を抜ける
83      If (token.IsCancellationRequested) Then
84          instr.VisaSendCommand("STOP")      'ストップ
85          instr.VisaSendCommand("LOC")      'ローカル
86          sResult = "Cancel"      '戻り値
87          DSR_Mnemonic = sResult      'プロパティ設定
88          Return sResult      '関数を終了する
89          Exit Function
90
91      End If
92
93      Loop
94
95      'ループ2 試験終了をモニターするループ
96      Do
97          Await Task.Delay(50).ConfigureAwait(False)      '50ms待つ
98
99          sMON = instr.VisaGetQuery("MON?").Split(",")      'MON?でデータステータスレジスタの状態を取得し、カンマで分割して
100             sMON配列に格納する
101          iDSR = Integer.Parse(sMON(0))      'MON?応答の中のDSR?の値を取得
102          SetProp_DSR(iDSR)      'MON?のDSR?相当戻り値をプロパティで読めるようにセット
103
104          'DSR?の戻り値をビット演算する。
```

```
104 '試験が終了して(=ビット2が落ちて)いたら試験結果を処理する
105 If Not iDSR And 4 Then
106     sMON(0) = Me.DSR_Mnemonic 'MON?の判定の値はDSR?の値と同じもので帰ってきているのでニモニク ㇿ
107     (文字)に置き換える
108     sResult = String.Join(",", sMON) 'MON?を分解した配列をもう一度CSVに組み立てなおして返す
109     Exit Do 'ループを抜けて次に進む
110 End If
111
112 'タスクキャンセル要求が入っていたら測定停止後Cancelを返して関数を終了する
113 If (token.IsCancellationRequested) Then
114     instr.VisaSendCommand("STOP") 'ストップ
115     instr.VisaSendCommand("LOC") 'ローカル
116     sResult = "Cancel" '戻り値
117     DSR_Mnemonic = sResult 'プロパティ設定
118     Return sResult
119     Exit Function
120
121 End If
122 Loop
123
124 Catch ex As Exception '例外処理
125     MessageBox.Show("通信エラーが発生しました。VISAアドレスや接続、電源を確認してください",
126         "GetResult_ASyncモジュール",
127         MessageBoxButtons.OK,
128         MessageBoxIcon.Error)
129     instr.VisaSendCommand("STOP") 'ストップ
130     instr.VisaSendCommand("LOC") 'ローカル
131     sResult = "Error" '戻り値
132     DSR_Mnemonic = sResult 'プロパティ設定
133
134 End Try
135
136 Return sResult '戻り値
137
```

```
138 End Function
139
140 ''' <summary>
141 ''' 取得した試験結果(MON?の戻り値)を指定したフォーマットに編集する
142 ''' </summary>
143 ''' <param name="instr">計測器のインスタンス</param>
144 ''' <param name="sResult">試験結果(MON?の戻り値)</param>
145 ''' <param name="sDatetime">測定開始日時</param>
146 ''' <param name="memoryNumber">メモリ番号</param>
147 ''' <returns></returns>
148 Public Function EditResults(instr As ClsVisaCommunication, sResult As String, sDatetime As String, memoryNumber As Integer) As String
149
150     '結果フォーマット
151     '試験番号, -, -, 試験開始日(年), 試験開始日(月), 試験開始日(日), 試験開始時刻(時), 試験開始時刻(分), 試験開始時刻(秒), 判定時の電圧値 (出力電圧), 判定時の電流値 (出力電流), 判定時の抵抗値 (導通抵抗の測定値), 判定時の試験時間, 判定結果, 限度値の取込 (抵抗値), 判定時の周波数(出力), メモリ番号 (試験条件), メモリ名 (試験条件)
152
153     Dim sResultList As New List(Of String) '試験結果のList型(入れ替えをしやすいするため)
154     Dim sResultArray() As String          '試験結果の配列
155     Dim sResultTmp() As String             'sResult編集用一時配列
156     Dim sMem() As String                  'メモリ内容 配列
157
158     '受け取ったカンマ区切りデータを配列にする
159     sResultTmp = sResult.Split(",") 'MON?の応答データを配列に分割 判定, 測定電圧値(V), 測定電流値(A), 最大抵抗値(または電圧値)(Ω またはV), 通常抵抗値(または電圧値)(Ω またはV), 測定時間(秒)
160     sMem = instr.VisaGetQuery("MEM? " + memoryNumber.ToString).Split(",") 'メモリ情報(カンマ区切り)を配列に分割
161
162     '試験結果をリストで作成する
163     sResultList.Add(My.Settings.TestNumber.ToString) '試験番号
164     sResultList.Add("") 'ブランク
165     sResultList.Add("") 'ブランク
166     sResultList.Add(sDatetime) '試験開始日時 ※カンマ区切りなので分解せず追加する
167     sResultList.Add(sResultTmp(1)) '測定電圧値(V)
168     sResultList.Add(sResultTmp(2)) '測定電流値(A)
```

```
169         sResultList.Add(sResultTmp(3))           '最大抵抗値(または電圧値)(ΩまたはV)
170         sResultList.Add(sResultTmp(5))           '測定時間(秒)
171         sResultList.Add(sResultTmp(0))           '判定結果
172         sResultList.Add(instr.VisaGetQuery("UPP?")) '限度値(上限抵抗値)
173         sResultList.Add(instr.VisaGetQuery("FREQ?")) '周波数(Hz)
174         sResultList.Add(memoryNumber.ToString)    'メモリ番号
175         sResultList.Add(sMem(0))                  'メモリ名称
176
177         '作成したリストを配列に変換する
178         sResultArray = sResultList.ToArray()
179
180         '配列に変換したリストをカンマ区切りデータとして返す
181         Return String.Join(",", sResultArray)
182
183     End Function
184
185     ''' <summary>
186     ''' 指定したメモリの呼び出しとメモリ名の取得
187     ''' </summary>
188     ''' <param name="instr">計測器のインスタンス</param>
189     ''' <param name="iMemoryNo">読み出したいメモリ番号(0~99)</param>
190     ''' <returns>
191     '''     正常終了:メモリ名称
192     '''     異常終了:"Error"
193     ''' </returns>
194     Public Function GetMemName(instr As ClsVisaCommunication, iMemoryNo As Integer) As String
195
196         Dim sResult() As String
197
198         '-----
199         ' メモリ番号が範囲外の際の処理
200         '-----
201         If iMemoryNo > 99 Or iMemoryNo < 0 Then
202             MessageBox.Show("指定可能な範囲は0~99までです",
203                 "エラー",
```

```
204         MessageBoxButtons.OK,
205         MessageBoxIcon.Error)
206     Return "Error"
207     Exit Function '関数を終了する
208 End If
209
210 '-----
211 ' メモリリコールと内容の読み出し
212 '-----
213 Try
214     instr.VisaSendCommand("REC " + Str(iMemoryNo)) '指定したメモリ番号をリコールする
215     sResult = instr.VisaGetQuery("MEM? " + Str(iMemoryNo)).Split(",") 'メモリ内容を読み出し、カンマ区切りの処理をし
        'で配列に渡す
216 Catch ex As Exception
217     Return "Error"
218     Exit Function
219
220 End Try
221
222 '-----
223 '取得したメモリ内容をメッセージボックスで簡易表示する
224 '-----
225 MessageBox.Show("メモリ名 " + sResult(0) + vbCrLf +
226     "試験電流(A) " + sResult(1) + vbCrLf +
227     "上限基準値(Ω) " + sResult(2) + vbCrLf +
228     "下限基準値(Ω) " + sResult(3) + vbCrLf +
229     "タイマー(秒) " + sResult(4) + vbCrLf +
230     "周波数(Hz) " + sResult(5) + vbCrLf +
231     "下限判定 (ON:1 OFF:0) " + sResult(6) + vbCrLf +
232     "オフセット (ON:1 OFF:0) " + sResult(7) + vbCrLf +
233     "タイマー (ON:1 OFF:0) " + sResult(8) + vbCrLf,
234     "メモリ内容")
235
236 '-----
237 '取得したメモリ名を返す
```

```
238         '-----
239         Return sResult(0)
240
241     End Function
242 End Class
243
```