

```
1: unit GasGunC;
2:
3: interface
4: uses Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
5:     Dialogs ,DaqX;
6:
7: const
8: DIOPort_A      = 0;
9: DIOPort_B      = 1;
10: DIOPort_C      = 2;
11:
12: DIO_Write      = false;
13: DIO_Read       = true;
14:
15: VacuumPumpRelay = 0;
16: SolenoidValveRelay = 1;
17: SensorVoltageSource = 2;
18: SensorChan = 7;
19: Gauge1Chan = 0;
20: Gauge2Chan = 1;
21: Gauge3Chan = 2;
22:
23:
24: var
25:     DaqHandle00      : DaqHandleT;
26:     daqDeviceType00  : DaqDacDeviceType;
27:     daqIODeviceType00 : DaqIODeviceType;
28:     DIOConfig        : byte;
29:     DIODevPort       : DWORD;
30:     DIODevType       : DaqIODeviceType;
31:     DIOExpansionPort : DaqIOExpansionPort;
32:     DIOWhichDevice   : DWord;
33:
34: implementation
35:
36: end.
```

```
1: unit GasGunMain;
2:
3: interface
4:
5: uses
6:   Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
7:   StdCtrls, ExtCtrls, TeeProcs, TeEngine, Chart, GasGunInterface,
8:   Errex, GasGunC, AppEvnts, daqx, Series;
9:
10: type
11:   TGasGunMainForm = class(TForm)
12:     RelatStatuGB: TGroupBox;
13:     vrs: TEdit;
14:     Label1: TLabel;
15:     Label2: TLabel;
16:     srs: TEdit;
17:     SensorStatusGB: TGroupBox;
18:     StrainGaugeStatusGB: TGroupBox;
19:     Label4: TLabel;
20:     Label6: TLabel;
21:     Label7: TLabel;
22:     StrainGauge1VoltageEB: TEdit;
23:     StrainGauge2VoltageEB: TEdit;
24:     StrainGauge3VoltageEB: TEdit;
25:     RoomtempGB: TGroupBox;
26:     RoomAirTempEB: TEdit;
27:     Label8: TLabel;
28:     GroupBox1: TGroupBox;
29:     FireControlBT: TButton;
30:     SensorVoltageSourceGB: TGroupBox;
31:     Label9: TLabel;
32:     SensorVoltageEB: TEdit;
33:     VoltageOnBT: TButton;
34:     VoltageOffBT: TButton;
35:     VacPumpOn: TButton;
36:     VacPumpOff: TButton;
37:     SolValveOff: TButton;
38:     SolValveOn: TButton;
39:     Label10: TLabel;
40:     Label11: TLabel;
41:     RadioButton1: TRadioButton;
42:     Label14: TLabel;
43:     GroupBox2: TGroupBox;
44:     Edit1: TEdit;
45:     Label16: TLabel;
46:     RadioButton3: TRadioButton;
47:     RadioButton4: TRadioButton;
48:     Label15: TLabel;
49:     Timer2: TTimer;
50:     GroupBox3: TGroupBox;
51:     Chart1: TChart;
52:     Series1: TPointSeries;
53:     Series2: TPointSeries;
54:     Series3: TPointSeries;
55:     procedure FormCreate(Sender: TObject);
56:     procedure VacPumpOnClick(Sender: TObject);
57:     procedure VacPumpOffClick(Sender: TObject);
58:     procedure SolValveOnClick(Sender: TObject);
59:     procedure SolValveOffClick(Sender: TObject);
60:     procedure VoltageOnBTClick(Sender: TObject);
61:     procedure VoltageOffBTClick(Sender: TObject);
62:     procedure Timer2Timer(Sender: TObject);
63:     procedure FireControlBTClick(Sender: TObject);
64:   private
65:     { Private declarations }
66:   public
67:     { Public declarations }
68:   end;
69:
70: var
71:   GasGunMainForm: TGasGunMainForm;
```

```
72:   Fired:           Integer;
73:   Counter:         Integer;
74:   SensorVoltage:   Real;
75:   StrainGauge1Voltage: Real;
76:   StrainGauge2Voltage: Real;
77:   StrainGauge3Voltage: Real;
78:
79: implementation
80:
81:   {$R *.DFM}
82:
83: procedure TGasGunMainForm.FormCreate(Sender: TObject);
84: begin
85:   OpenTheDaq(1);
86:
87:   daq9513SetMasterMode(DaqHandle00, DiodtLocal9513, 0, 1, DcsF4, false, false,
88:   DtodDisabled);
89:
90:   daq9513SetCtrMode (DaqHandle00, DiodtLocal9513, 0, 1, DgcNoGating, true,
91:   DcsF4, false, false, false, false, true, DocInactiveLow);
92:
93:   GasGunInterface_Frm.InitRelays;
94:   {Turning all relay status settings to off}
95:   vrs.text := 'OFF';
96:   srs.text := 'OFF';
97:   Counter := 0;
98:
99: end;
100:
101: {Turning on Vacuum Pump Relay via button}
102: procedure TGasGunMainForm.VacPumpOnClick(Sender: TObject);
103: begin
104:   GasGunInterface_Frm.TurnRelayOn(VacuumPumpRelay);
105:   vrs.text := 'ON';
106: end;
107:
108: {Turning off Vacuum Pump Relay via button}
109: procedure TGasGunMainForm.VacPumpOffClick(Sender: TObject);
110: begin
111:   GasGunInterface_Frm.TurnRelayOff(VacuumPumpRelay);
112:   vrs.text := 'OFF';
113: end;
114:
115: {Turning on Solenoid Valve Relay via button}
116: procedure TGasGunMainForm.SolValveOnClick(Sender: TObject);
117: begin
118:   GasGunInterface_Frm.TurnRelayOn(SolenoidValveRelay);
119:   srs.text := 'ON';
120: end;
121:
122: {Turning off Solenoid Valve Relay via button}
123: procedure TGasGunMainForm.SolValveOffClick(Sender: TObject);
124: begin
125:   GasGunInterface_Frm.TurnRelayOff(SolenoidValveRelay);
126:   srs.text := 'OFF';
127: end;
128:
129: procedure TGasGunMainForm.VoltageOnBTClick(Sender: TObject);
130: begin
131:   GasGunInterface_Frm.TurnRelayOn(SensorVoltageSource);
132: end;
133:
134: procedure TGasGunMainForm.VoltageOffBTClick(Sender: TObject);
135: begin
136:   GasGunInterface_Frm.TurnRelayOff(SensorVoltageSource);
137: end;
138: {*****}
139: *      Firing Controls      *
140: {*****}
141:
142: procedure TGasGunMainForm.Timer2Timer(Sender: TObject);
```

```
143: begin
144:   SensorVoltage := Scandata.ReadOneChannel(SensorChan);
145:   SensorVoltageEB.Text := FloatToStr(SensorVoltage);
146:   StrainGauge1Voltage := Scandata.ReadOneChannel(Gauge1Chan);
147:   StrainGauge1VoltageEB.Text := FloatToStr(StrainGauge1Voltage);
148:   StrainGauge2Voltage := Scandata.ReadOneChannel(Gauge2Chan);
149:   StrainGauge2VoltageEB.Text := FloatToStr(StrainGauge2Voltage);
150:   StrainGauge3Voltage := Scandata.ReadOneChannel(Gauge3Chan);
151:   StrainGauge3VoltageEB.Text := FloatToStr(StrainGauge3Voltage);
152: end;
153:
154:
155: procedure TGasGunMainForm.FireControlBTClick(Sender: TObject);
156: var
157:
158:   loop : boolean;
159:   SensorV : real;
160:   str1 : string;
161:   str2 : string;
162:   str3 : string;
163:   DateTime : TDateTime;
164:   Hour, Min1, Min2, Sec1, MSec1, Sec2, MSec2: Word;
165:   eTime : real;
166:   counts : WORD;
167:   count1 : WORD;
168:   count2 : WORD;
169:   count3 : WORD;
170:   something : WORD;
171:   velocity: real;
172:   inc : word;
173:
174: begin
175: If Radiobutton1.Checked Then
176: Begin
177:   loop := true;
178:   count1 := 0;
179:   count2 := 0;
180:   count3 := 0;
181:
182:   series1.Clear;
183:   series2.Clear;
184:   series3.Clear;
185:
186:   //DecodeTime(Time,Hour,Min1,Sec1,MSec1);           //System Based Time Measurement
187:
188:   GasGunInterface_Frm.TurnRelayOn(SolenoidValveRelay);
189:
190:   daq9513SetHold(DaqHandle00, DiodtLocal9513,0, 1, 0);
191:
192:   daq9513MultCtrl(DaqHandle00, DiodtLocal9513, 0, DmccArm, true, false, false,
193: false, false);
194:
195:   daq9513MultCtrl(DaqHandle00, DiodtLocal9513, 0, DmccSave, true, false, false,
196: false, false);
197:
198:   daq9513GetHold(DaqHandle00, DiodtLocal9513, 0, 1, count1);
199:
200: while loop
201: do
202: begin
203:   daq9513MultCtrl(DaqHandle00, DiodtLocal9513, 0, DmccSave, true, false, false,
204: false, false);
205:   daq9513GetHold(DaqHandle00, DiodtLocal9513, 0, 1, count2);
206:   SensorV := Scandata.ReadOneChannel(SensorChan);
207:   if SensorV > 3.5 then Loop := false;
208: end;
209:
210:   daq9513MultCtrl(DaqHandle00, DiodtLocal9513, 0, DmccDisarm, true, false, false,
211: false, false);
212:
213:   GasGunInterface_Frm.TurnRelayOff(SolenoidValveRelay);
```

```
214:
215: {DecodeTime(Time,Hour,Min2,Sec2,MSec2); //System Time Based measurement
216:
217: eTime := (Sec2-Sec1) + (MSec2-Msec1)/1000;} //System Time Based measurement
218:
219: If (count2-count1) > 0 Then
220: begin
221: str1 := 'Projectile Velocity ' + FloatToStr(2/((count2 - count1)/1000)) + ' ft/s';
222: str2 := 'Projectile Travel Time ' + FloatToStr((count2-count1)/1000) + 's';
223: showmessage(str1);
224: showmessage(str2);
225: end
226: else
227: If (count2-count1)=0 Then
228: Begin
229: str1 := 'Projectile Velocity - ' + 'infinity' + ' ft/s';
230: str2 := 'Projectile Travel Time ' + '0' + 's';
231: showmessage(str1);
232: showmessage(str2);
233:
234: radiobutton1.Checked := False;
235:
236: End;
237: {*****
238: *                               Strain Gauge Measurements                               *
239: *****}
240:
241: loop := true;
242: something :=1;
243:
244: daq9513SetHold(DaqHandle00, DioldtLocal9513,0, 1, 0);
245:
246: daq9513MultCtrl(DaqHandle00, DioldtLocal9513, 0, DmccArm, true, false, false,
247: false, false);
248:
249: daq9513MultCtrl(DaqHandle00, DioldtLocal9513, 0, DmccSave, true, false, false,
250: false, false);
251:
252: daq9513GetHold(DaqHandle00, DioldtLocal9513, 0, 1, count1);
253:
254: while loop
255: do
256: begin
257: daq9513MultCtrl(DaqHandle00, DioldtLocal9513, 0, DmccSave, true, false, false,
258: false, false);
259: daq9513GetHold(DaqHandle00, DioldtLocal9513, 0, 1, count2);
260: StrainGauge1Voltage := Scandata.ReadOneChannel(Gauge1Chan);
261: Series1.AddXY( (count2-count1), StrainGauge1Voltage);
262: Series2.AddXY( (count2-count1), StrainGauge2Voltage);
263: Series3.AddXY( (count2-count1), StrainGauge3Voltage);
264: if count2 - count1>100 then Loop := false;
265: end;
266:
267: end
268:
269: end;
270:
271: end.
```

```
1: unit GasGunInterface;
2:
3: interface
4:
5: uses
6:   Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
7:   Dialogs, Daqx, GasGunC;
8:
9: Const
10:   MaxChannels    = 16;
11:   ADVOLTS        = 5;
12:
13: var
14:   DaqPort        : word;
15:   err            : DaqError;
16:
17: procedure CreateScanDataObject(StartChan,EndChan:DWORD; pts:longint;
18:                               Freq:Single; Ga : byte);
19: type
20:   PScanData = ^TScanData;
21:   TScanData = array [0..1] of Word;
22:
23:   PScanTags = ^TScanTags;
24:   TScanTags = array [0..1] of Byte;
25:
26:   PRealData = ^TRealData;
27:   TRealData = array [1..MaxChannels,0..1] of Single;
28:
29: type
30:   PReadScanObject = ^TReadScanObject;
31:   TReadScanObject = object
32:     NoOfChannels      : Integer;
33:     StartChannel,
34:     EndChannel        : DWORD;
35:     PointsToAcquire   : longint;
36:     Frequency         : single;
37:     Gain              : byte;
38:     OneShot           : byte;
39:     Trigger           : word;
40:     Level             : byte;
41:     CalFactor         : array [1..MaxChannels] of Single;
42:     constructor Init(StartChan,EndChan:DWORD; pts:longint;
43:                     Freq:Single; Ga : byte);
44:     destructor Done;
45:     function   ReadOneChannel(ChNo: integer): real;
46:
47:   protected
48:     TotalData        : longint;
49: end;
50:
51: var
52:   ScanData        : PReadScanObject;
53:   RawData         : PScanData;
54:   Buffer           : PScanData;
55:   Tags            : PScanTags;
56:   RealData        : PRealData;
57:
58: type
59:   TGasGunInterface_Frm = class(TForm)
60:   private
61:     { Private declarations }
62:   public
63:     { Public declarations }
64:     procedure InitRelays;
65:     procedure TurnRelayOn(ChNo : integer);
66:     procedure TurnRelayOff(ChNo : integer);
67:   end;
68:
69: var
70:   GasGunInterface_Frm: TGasGunInterface_Frm;
71:
```

```
72: implementation
73:
74: {$R *.DFM}
75: {*****}
76: procedure TGasGunInterface_Frm.InitRelays;
77: var
78:   cfg : DWord;
79:   err : DaqError;
80: begin
81:   if daqIOGet8255Conf(DaqHandle00,DIO_Write,DIO_Write,DIO_Read,DIO_Write,cfg) <>
82:     DerrNoError then showmessage('DIO Port ERROR');
83:   DIOConfig      := cfg;
84:   DIODevType     := DioldtLocal8255;
85:   DIODevPort     := Diodp8255A;
86:   DIOExpansionPort := Dioep1;
87:   DIOWhichDevice := 0;
88:
89: end;
90: {*****Turning Relays On*****}
91: procedure TGasGunInterface_Frm.TurnRelayOn(ChNo : integer);
92: var
93:   error      : DaqError;
94:   bitValue   : longbool;
95:   cfg : Dword;
96: begin
97:   bitValue :=true; {Ord(true)=1}
98:   error := daqIOWriteBit(DaqHandle00,DIODevType,
99:     DIODevPort,DIOWhichDevice,DIOExpansionPort,
100:   ChNo,bitValue);
101: end;
102: {*****Turning Relays Off*****}
103: procedure TGasGunInterface_Frm.TurnRelayOff(ChNo : integer);
104: var
105:   error      : DaqError;
106:   bitValue   : longbool;
107:   cfg : Dword;
108: begin
109:   bitValue := false; {Ord(false)=0}
110:   error := daqIOWriteBit(DaqHandle00,DIODevType,
111:     DIODevPort,DIOWhichDevice,DIOExpansionPort,
112:   ChNo,bitValue);
113: end;
114:
115: procedure CreateScanDataObject(StartChan,EndChan:DWORD; pts:longint;
116:   Freq:Single; Ga : byte);
117:
118: begin
119:   New(ScanData, Init(StartChan,EndChan,pts,Freq,Ga));
120: end;
121:
122: constructor TReadScanObject.Init(StartChan,EndChan:DWORD; pts:longint;
123:   Freq:Single; Ga : byte);
124:
125: begin
126:
127:   StartChannel      := StartChan;
128:   EndChannel        := EndChan;
129:   NoOfChannels      := EndChannel-StartChannel+1;
130:   PointsToAcquire   := pts;
131:   Frequency         := Freq;
132:   Gain              := Ga;
133:   OneShot           := 0;
134:   Level             := 0;
135:   Trigger           := 0;
136:   TotalData         := NoOfChannels*PointsToAcquire;
137:   GetMem(RawData,   TotalData*SizeOf(Word));
138:   GetMem(Buffer,    TotalData*SizeOf(Word));
139:   GetMem(Tags,      TotalData*SizeOf(Byte));
140:   GetMem(RealData,  TotalData*SizeOf(Single)*MaxChannels);
141: end;
142:
```

```
143: destructor TReadScanObject.Done;  
144: begin  
145: end;  
146:  
147: function TReadScanObject.ReadOneChannel(ChNo: integer): real;  
148: var  
149:   Flags   : integer;  
150:   Gains    : integer;  
151:   sample   : word;  
152:   Temp1    : real;  
153: begin  
154:   Gains := DgainX1;  
155:   Flags := DafAnalog + DafSingleEnded + DafBipolar;  
156:   daqAdcRd(DaqHandle00,ChNo,sample,Gains,flags);  
157:   { sample := sample shr 0; }  
158:   Temp1 := (sample-2048*16)*(ADVOLTS)/(2048*16);  
159:   Result := Temp1;  
160: end;  
161:  
162: end.
```

```
1: unit Errex;
2:
3: interface
4:
5: uses DaqX, GasGunInterface, GasGunC;
6:
7:
8:
9: const
10: DV_DaqBook00 = 0;
11:
12:
13: procedure TestError(Device: integer);
14: procedure SetTheErrorHandler;
15: procedure OpenTheDaq(Device: integer);
16:
17: // Daqx error handler prototypes
18: procedure ErrorHandler( errCode: DaqError ); stdcall;
19:
20: implementation
21:
22: uses SysUtils, Dialogs;
23:
24:
25: *****
26: // This unit demonstrates how to initialize a daqx device and
27: // how to select and error handler
28: //
29: // Functions used:
30: // daqOpen( daqName )
31: // daqOnline( handle, online )
32: // daqSetDefaultErrorHandler( errHandler )
33: // daqSetErrorHandler( handle, errHandler )
34: // daqAdcSetTrig( handle, triggerSource, rising, level, hysteresis, channel );
35: // daqClose( handle )
36: //
37: procedure TestError(Device: integer);
38: var
39:   online: longbool;
40:   { drVersion: DWORD; // Use to receive driver version}
41:   { hwVersion: DWORD; // Use to receive hardware version}
42: begin
43:
44: end;
45:
46: *****
47: // Error handler for daq errors.
48: //
49: procedure ErrorHandler( errCode: DaqError ); stdcall;
50: var
51:   msg: string;
52: begin
53:   if ord(errCode)=39 then
54:     begin
55:       ShowMessage('ERROR - Thermocouple Malfunction');
56:
57:     end
58:   else
59:     begin
60:       msg:= 'Message from procedure ErrorHandler' + Chr(13) + Chr(10);
61:       msg:= msg + 'DaqComp error number ' + IntToStr(ord(errCode)) + ' occurred.';
62:       MessageDlg(msg , mtError, [mbOk], 0);
63:     end;
64:
65: end;
66:
67:
68: *****
69: // Procedure to set the error handler. Complains and stops if unable.
70: //
71: procedure SetTheErrorHandler;
```

```
72: begin
73: (* if (daqSetErrorHandler(addr(ErrorHandler)) <> DerrNoError ) then
74:   begin
75:     MessageDlg('Unable to set default error handler. Exiting program', mtError, [mbOk], 0);
76:     Halt;
77:   end;
78:   HeatChamberMainFrm.Memo1.Lines.Add( 'Error handler set to procedure "ErrorHandler"');
79:   // Disable the choice for the driver built-in error handler since it isn't available
80:   // after setting another handler until the DAQX.DLL is reloaded.
81:   HeatChamberMainFrm.DefaultErrBtn.Enabled:= False;
82:
83: *)
84: end;
85:
86: //*****
87: // Procedure to open a DaqBook or DaqBoard depending on which is selected on Form1.
88: //
89: procedure OpenTheDaq(Device: integer);
90: var
91:   err:      DaqError;
92: begin
93:   DaqHandle00 := daqOpen('DaqBook0');
94: end;
95:
96:
97: end.
```