

```

1  #include "stdafx.h"
2  #include "time.h"
3  #include "FormMain.h"
4
5  //#define POLLING
6
7  #ifdef _WIN64
8  #define OS 64
9  #else
10 #define OS 32
11 #endif
12
13 namespace IPCSample {
14
15 unsigned char clip(int val) { return (val <= 255) ? ((val > 0) ? val : 0) : 255; };
16
17 FormMain::FormMain(HWND hostHandle)
18 {
19     InitializeComponent();
20     FrameWidth = 0;
21     FrameHeight = 0;
22     FrameDepth = 0;
23     MetadataSize = 0;
24     bmp = gnew Bitmap( 100 , 100 ,
25         System::Drawing::Imaging::PixelFormat::Format24bppRgb );
26     ipc = nullptr;
27     ipcInitialized = false;
28     frameInitialized = false;
29     Connected = false;
30     Painted = false;
31     radioButtonRange3Sigma->Checked = true;
32     MeasureAreaCount = 0;
33     Margin = groupBoxVideo->Size - pictureBox->Size;
34     FC0 = FC1 = 0;
35     AppendLog = false;
36     ResetPending = false;
37     List = gnew System::Collections::ArrayList(100);
38     char ch[256];
39     ::GetSystemDirectoryA(ch, 256);
40     HotPen = gnew Pen(Color::Red);
41     ColdPen = gnew Pen(Color::Blue);
42     ContrastPen = gnew Pen(Color::Yellow);
43 }
44
45 FormMain::~FormMain()
46 {
47     if (components)
48         delete components;
49 }
50
51 void FormMain::AppInit(void)
52 {
53     Text += String::Format(" (Rel. {0} (x{1}))", Version, OS);
54     Init(160, 120, 2);
55     ipc = gnew IPC(1);
56     //ipc = gnew IPC(1, L"\\IPC Sample.log", 2, false);
57     //ipc = gnew IPC(1, ~(LOGGRP_FRAME | LOGGRP_ALIVE | LOGGRP_STATE));
58
59 #ifdef POLLING
60     Application::Idle += gnew EventHandler(this, &FormMain::Application_Idle);
61 #endif
62 }
63
64 System::Void FormMain::textBoxInstanceName_TextChanged(System::Object^ sender,
65     System::EventArgs^ e)
66 {
67     ReleaseIPC();
68 }
69
70 void FormMain::InitIPC()
71 {
72     HRESULT hr;
73     if(ipc && !ipcInitialized)

```

```

72     {
73         hr = ipc->Init(0, textBoxInstanceName->Text);
74
75         if(FAILED(hr))
76         {
77             ipcInitialized = frameInitialized = false;
78         }
79         else
80         {
81             #ifndef POLLING
82                 ipc->OnServerStopped = gcnew IPC::delOnServerStopped(this,
83                     &FormMain::OnServerStopped );
84                 ipc->SetCallback_OnServerStopped(0, ipc->OnServerStopped);
85
86                 ipc->OnFrameIRInit = gcnew IPC::delOnFrameInit(this,
87                     &FormMain::OnFrameInit );
88                 ipc->SetCallback_OnFrameInit(0, ipc->OnFrameIRInit);
89
90                 ipc->OnNewFrameIREx2 = gcnew IPC::delOnNewFrameEx2(this,
91                     &FormMain::OnNewFrameEx );
92                 ipc->SetCallback_OnNewFrameEx2(0, ipc->OnNewFrameIREx2);
93
94                 ipc->OnConfigChanged = gcnew IPC::delOnConfigChanged(this,
95                     &FormMain::OnConfigChanged );
96                 ipc->SetCallback_OnConfigChanged(0, ipc->OnConfigChanged);
97
98                 ipc->OnInitCompleted = gcnew IPC::delOnInitCompleted(this,
99                     &FormMain::OnInitCompleted );
100                 ipc->SetCallback_OnInitCompleted(0, ipc->OnInitCompleted);
101
102                 ipc->OnFileCommandReady = gcnew IPC::delOnStringSend(this,
103                     &FormMain::OnFileCommandReady);
104                 ipc->SetCallback_OnFileCommandReady(0, ipc->OnFileCommandReady);
105
106                 ipc->OnNewNMEAString = gcnew IPC::delOnStringSend(this,
107                     &FormMain::OnNewNMEAString);
108                 ipc->SetCallback_OnNewNMEAString(0, ipc->OnNewNMEAString);
109
110             #endif
111             hr = ipc->Run(0);
112             ipcInitialized = SUCCEEDED(hr);
113             LastFrameTime = DateTime::Now;
114             Tmax = 0;
115             ResetPending = true;
116             FC0 = FC1 = 0;
117         }
118         label1->Text = hr ? "NOT CONNECTED" : "OK";
119     }
120 }
121
122 void FormMain::ReleaseIPC(void)
123 {
124     Connected = false;
125     if(ipc && ipcInitialized)
126     {
127         if(OldApplRect)
128         {
129             ipc->SetMainWindowEmbedded(0, OldEmbeddedState);
130             ipc->SetMainWindowRect(0, OldApplRect);
131         }
132         ipc->Release(0);
133         ipcInitialized = false;
134     }
135 }
136
137 void FormMain::AppExit(void)
138 {
139     #ifdef POLLING
140         Application::Idle -= gcnew EventHandler(this, &FormMain::Application_Idle);
141     #endif
142     if (startedPix && ipc){
143         ipc->CloseApplication(0);
144     }
145 }

```

```

138     }
139     ReleaseIPC();
140     delete ipc;
141 }
142
143 void FormMain::GetBitmap(Bitmap^ Bmp, array<short>^values)
144 {
145     int stride_diff;
146     // Lock the bitmap's bits.
147     System::Drawing::Rectangle rect = System::Drawing::Rectangle(0, 0, Bmp->Width,
148     Bmp->Height);
149     Imaging::BitmapData^ bmpData = Bmp->LockBits( rect,
150     Imaging::ImageLockMode::ReadWrite, Bmp->PixelFormat );
151     stride_diff = bmpData->Stride - FrameWidth*3;
152
153     // Get the address of the first line.
154     IntPtr ptr = bmpData->Scan0;
155
156     if(checkBoxColors->Checked)
157     {
158         for ( int dst=0, src=0, y=0; y < FrameHeight; y++, dst += stride_diff)
159             for ( int x=0; x < FrameWidth; x++, src++, dst+=3 )
160             {
161                 int C = LOBYTE(values[src] - 16;
162                 int D = HIBYTE(values[src - (src%2)] - 128;
163                 int E = HIBYTE(values[src - (src%2) + 1] - 128;
164                 rgbValues[ dst ] = clip(( 298 * C + 516 * D + 128) >> 8);
165                 rgbValues[ dst+1 ] = clip(( 298 * C - 100 * D - 208 * E + 128) >> 8);
166                 rgbValues[ dst+2 ] = clip(( 298 * C + 409 * E + 128) >> 8);
167             }
168     }
169     else
170     {
171         short mn, mx;
172         GetBitmap_Limits(values, &mn, &mx);
173         double Fact = 255.0 / (mx - mn);
174
175         for ( int dst=0, src=0, y=0; y < FrameHeight; y++, dst += stride_diff)
176             for ( int x=0; x < FrameWidth; x++, src++, dst+=3 )
177                 rgbValues[dst] = rgbValues[dst+1] = rgbValues[dst+2] =
178                 min(max((int)(Fact * (values[src] - mn)), 0), 255);
179     }
180
181     // Copy the RGB values back to the bitmap
182     System::Runtime::InteropServices::Marshal::Copy( rgbValues, 0, ptr,
183     rgbValues->Length );
184
185     // Unlock the bits.
186     Bmp->UnlockBits( bmpData );
187 }
188
189 void FormMain::GetBitmap_Limits(array<short>^Values, short *min, short *max)
190 {
191     int y, Count;
192     double Sum, Mean, Variance;
193     if(!Values) return;
194
195     if(radioButtonRangeMinMax->Checked)
196     {
197         *min = SHRT_MAX;
198         *max = SHRT_MIN;
199         for (y=0; y < FrameSize; y++ )
200         {
201             if(Values[y] != INVALIDPIXEL)
202             {
203                 *min = min(Values[y], *min);
204                 *max = max(Values[y], *max);
205             }
206         }
207     }
208     return;
209 }

```

```

207     Sum = 0;
208     Count = 0;
209     for (y=0; y < FrameSize; y++ )
210         if(Values[y] != INVALIDPIXEL)
211             {
212                 Sum += Values[y];
213                 Count++;
214             }
215     Mean = (double)Sum / Count;
216     Sum = 0;
217     for (y=0; y < FrameSize; y++ )
218         if(Values[y] != INVALIDPIXEL)
219             Sum += (Mean - Values[y]) * (Mean - Values[y]);
220     Variance = Sum / Count;
221     Variance = Math::Sqrt(Variance);
222     Variance *= radioButtonRange1Sigma->Checked ? 1: 3; // 1 or 3 Sigma
223     *min = short(Mean - Variance);
224     *max = short(Mean + Variance);
225 }
226
227
228 void FormMain::Application_Idle(Object ^sender, EventArgs ^e)
229 {
230     #ifdef POLLING
231         if(Connected && frameInitialized)
232             {
233                 int Size = FrameWidth * FrameHeight * FrameDepth;
234                 void *Buffer = new char[Size];
235                 FrameMetadata2 *Metadata = (FrameMetadata2*)(new char[MetadataSize]);
236                 if(ipc->GetFrameQueue(0))
237                     if(SUCCEEDED(ipc->GetFrame2(0, 0, Buffer, Size, Metadata, MetadataSize)))
238                         OnNewFrameEx(Buffer, Metadata);
239                 delete [] Buffer;
240                 delete [] Metadata;
241             }
242     #endif
243 }
244
245 HRESULT FormMain::MainTimer100ms(void)
246 {
247     Painted = false;
248     if(Connected)
249     {
250         if(progressBarInit->Visible)
251         {
252             int i = ipc->GetInitCounter(0);
253             progressBarInit->Value = progressBarInit->Maximum - i;
254             progressBarInit->Visible = (i != 0);
255         }
256     }
257     #ifdef POLLING
258     if(ipcInitialized)
259     {
260         WORD State = ipc->GetIPCState(0, true);
261         if(State & IPC_EVENT_SERVER_STOPPED)
262             OnServerStopped(0);
263         if(State & IPC_EVENT_CONFIG_CHANGED)
264             OnConfigChanged(0);
265         if(!Connected && (State & IPC_EVENT_INIT_COMPLETED))
266             OnInitCompleted();
267         if(State & IPC_EVENT_FRAME_INIT)
268         {
269             int frameWidth, frameHeight, frameDepth, frameMetadataSize;
270             if (SUCCEEDED(ipc->GetFrameConfig(0, &frameWidth, &frameHeight,
271                 &frameDepth)))
272             {
273                 Init(frameWidth, frameHeight, frameDepth);
274                 if (SUCCEEDED(ipc->GetFrameMetadataSize(0, &frameMetadataSize)))
275                 {
276                     MetadataSize = frameMetadataSize;
277                 }
278             }

```

```

279     }
280     if(Connected && (State & IPC_EVENT_FILE_CMD_READY))
281     {
282         wchar_t Filename[1024];
283         ipc->GetPathOfStoredFile(0, Filename, 1024);
284         OnFileCommandReady(Filename);
285     }
286     if (Connected && (State & IPC_EVENT_NEW_NMEA_STRING))
287     {
288         wchar_t NMEAString[1024];
289         ipc->GetNewNMEAString(0, NMEAString, 1024);
290         OnNewNMEAString(NMEAString);
291     }
292 }
293 #endif
294 return S_OK;
295 }
296
297 HRESULT FormMain::MainTimer1000ms(void)
298 {
299     double Mean = 0, Sigma = 0, x, last=0;
300     IEnumrator^ Enum = List->GetEnumerator();
301     while ( Enum->MoveNext() )
302     {
303         x = safe_cast<double>(Enum->Current);
304         Tmax = max(x - last, Tmax);
305         last = x;
306         Mean += x;
307     }
308     Mean /= List->Count;
309     Enum->Reset();
310     while ( Enum->MoveNext() )
311     {
312         x = Mean - safe_cast<double>(Enum->Current),
313         Sigma += x*x;
314     }
315     Sigma = Math::Sqrt(Sigma / List->Count);
316     List->Clear();
317
318     labelFramerate->Text = String::Format(L"{0:0.0}Hz,    \u0394t={1:0.0}ms,
319     \u0394tmax={2:0}ms, \u03C3={3:0.0}ms",
320     1000.0 / Mean, Mean, Tmax, Sigma);
321
322     if(Connected)
323     {
324         labelTempTarget->Text    = String::Format("Target-Temp: {0:##0.0}°C",
325         ipc->GetTempTarget(0));
326         labelTempChip->Text      = String::Format("Chip-Temp: {0:##0.0}°C",
327         ipc->GetTempChip(0));
328         labelTempFlag->Text      = String::Format("Flag-Temp: {0:##0.0}°C",
329         ipc->GetTempFlag(0));
330         labelTempBox->Text       = String::Format("Box-Temp: {0:##0.0}°C",
331         ipc->GetTempBox(0));
332         labelTempOptics->Text    = String::Format("Optics-Temp: {0:##0.0}°C",
333         ipc->GetTempOptics(0));
334
335         for(int i=0; i < groupBoxMeasureAreas->Controls->Count; i++)
336             ((Label^)groupBoxMeasureAreas->Controls->default[i])->Text
337             = String::Format("Temp[{0}]: {1:##0.0}°C", i,
338             ipc->GetTempMeasureArea(0, i));
339     }
340     return S_OK;
341 }
342
343 void FormMain::GetSoftwareInfo(void)
344 {
345     Dev.SoftInfo.Application    = gcnw
346     IPCSample::Version(ipc->GetVersionApplication(0));
347     Dev.SoftInfo.CD_DLL         = gcnw IPCSample::Version(ipc->GetVersionCD_DLL(0));
348     Dev.SoftInfo.HID_DLL        = gcnw
349     IPCSample::Version(ipc->GetVersionHID_DLL(0));
350     Dev.SoftInfo.IPC_DLL        = gcnw IPCSample::Version(ipc->GetVersionIPC_DLL(0));
351 }
352

```

```

343 void FormMain::GetDeviceInfo(void)
344 {
345     GetOptics();
346     GetVideoFormats();
347     Dev.Info.HW_Model = ipc->GetHardware_Model(0);
348     Dev.Info.HW_Spec = ipc->GetHardware_Spec(0);
349     Dev.Info.SerialNumber = ipc->GetSerialNumber(0);
350     Dev.Info.SerialNumberULIS = ipc->GetSerialNumber(0);
351     Dev.Info.FW_MSP = ipc->GetFirmware_MSP(0);
352     Dev.Info.FW_Cypress = ipc->GetFirmware_Cypress(0);
353     Dev.Info.PID = ipc->GetPID(0);
354     Dev.Info.VID = ipc->GetVID(0);
355     Dev.Info.OpticsFOV = ipc->GetOpticsFOV(0, ipc->GetOpticsIndex(0));
356     Dev.Info.OpticsSerialNumber = ipc->GetOpticsSerialNumber(0,
ipc->GetOpticsIndex(0));
357     Dev.Info.TempMinRange = ipc->GetTempMinRange(0, ipc->GetTempRangeIndex(0));
358     Dev.Info.TempMaxRange = ipc->GetTempMaxRange(0, ipc->GetTempRangeIndex(0));
359     Dev.Info.TempRangeDecimal_CaliValue = ipc->GetTempRangeDecimal(0, false);
360     Dev.Info.TempRangeDecimal_EffValue = ipc->GetTempRangeDecimal(0, true);
361 }
362 void FormMain::GetDeviceSetup(void)
363 {
364     int cnt = ipc->GetOpticsCount(0);
365     if(cnt > 0)
366         comboBoxOptics->SelectedIndex = Dev.Setup.OpticsIndex =
ipc->GetOpticsIndex(0);
367     else
368         comboBoxOptics->SelectedIndex = -1;
369     if(ipc->GetTempRangeCount(0) > 0)
370         comboBoxTempRange->SelectedIndex = Dev.Setup.TempRangeIndex =
ipc->GetTempRangeIndex(0);
371     else
372         comboBoxTempRange->SelectedIndex = -1;
373     if(ipc->GetVideoFormatCount(0) > 0)
374         comboBoxVideoFormats->SelectedIndex = Dev.Setup.VideoFormatIndex =
ipc->GetVideoFormatIndex(0);
375     else
376         comboBoxVideoFormats->SelectedIndex = -1;
377
378     checkBoxColors->Checked = TIPCMODE(Dev.Setup.IPCMode = ipc->GetIPCMode(0)) ==
ipcColors;
379     Dev.Setup.FixedEmissivity = ipc->GetFixedEmissivity(0);
380     Dev.Setup.FixedTransmissivity = ipc->GetFixedTransmissivity(0);
381     Dev.Setup.FixedTempAmbient = ipc->GetFixedTempAmbient(0);
382     Dev.Setup.FixedTempReference = ipc->GetFixedTempReference(0);
383     numericUpDownEmissivity->Value = Decimal(Dev.Setup.FixedEmissivity);
384     numericUpDownTempAmbient->Value = Decimal(Dev.Setup.FixedTempAmbient);
385     MeasureAreaCount = ipc->GetMeasureAreaCount(0);
386 }
387
388 void FormMain::GetOptics(void)
389 {
390     int i, Cnt = ipc->GetOpticsCount(0);
391
392     comboBoxOptics->Items->Clear();
393     comboBoxOptics->Text = nullptr;
394     for (i = 0; i < Cnt; i++)
395     {
396         String ^s = String::Format("{0}°", ipc->GetOpticsFOV(0, i));
397         ULONG o_serno = ipc->GetOpticsSerialNumber(0, i);
398         if (o_serno)
399             s += String::Format(" ({0})", o_serno);
400         comboBoxOptics->Items->Add(s);
401     }
402
403     i = ipc->GetOpticsIndex(0);
404     if(i < Cnt) comboBoxOptics->SelectedIndex = i;
405     GetTempRanges();
406 }
407
408 void FormMain::GetTempRanges(void)
409 {
410     int i, Cnt = ipc->GetTempRangeCount(0);

```

```

411     comboBoxTempRange->Items->Clear();
412     comboBoxTempRange->Text = nullptr;
413     for(i=0; i<Cnt; i++)
414         comboBoxTempRange->Items->Add(String::Format("{0:##0.0}°C .. {1:##0.0}°C",
415             ipc->GetTempMinRange(0, i), ipc->GetTempMaxRange(0, i)));
416     i = ipc->GetTempRangeIndex(0);
417     if(i < Cnt) comboBoxTempRange->SelectedIndex = i;
418 }
419
420 void FormMain::GetVideoFormats(void)
421 {
422     int i, Cnt = ipc->GetVideoFormatCount(0);
423
424     comboBoxVideoFormats->Items->Clear();
425     comboBoxVideoFormats->Text = nullptr;
426     String ^sFormat = "{0}x{1}@{2:0}Hz";
427     for(i=0; i<Cnt; i++)
428     {
429         VideoFormat vf;
430         ipc->GetVideoFormat(0, i, &vf);
431         String ^s;
432         if(vf.WidthIR && vf.HeightIR && vf.FramerateIR)
433         {
434             s = String::Format(sFormat, vf.WidthIR, vf.HeightIR, vf.FramerateIR);
435             if(vf.WidthVisible && vf.HeightVisible && vf.FramerateVisible)
436                 s += " / Vis:" + String::Format(sFormat, vf.WidthVisible,
437                     vf.HeightVisible, vf.FramerateVisible);
438             comboBoxVideoFormats->Items->Add(s);
439         }
440         else
441             comboBoxVideoFormats->Items->Add("wrong format");
442     }
443
444     i = ipc->GetVideoFormatIndex(0);
445     if(i < Cnt) comboBoxVideoFormats->SelectedIndex = i;
446 }
447
448 void FormMain::SetOpticsIndex(void)
449 {
450     Dev.Setup.OpticsIndex = ipc->SetOpticsIndex(0,
451         Decimal::ToUInt16(comboBoxOptics->SelectedIndex));
452     GetTempRanges();
453 }
454
455 void FormMain::SetTempRangeIndex(void)
456 {
457     Dev.Setup.TempRangeIndex = ipc->SetTempRangeIndex(0,
458         Decimal::ToUInt16(comboBoxTempRange->SelectedIndex));
459 }
460
461 void FormMain::SetVideoFormatIndex(void)
462 {
463     Dev.Setup.VideoFormatIndex = ipc->SetVideoFormatIndex(0,
464         Decimal::ToUInt16(comboBoxVideoFormats->SelectedIndex));
465 }
466
467 void FormMain::SetFlag(bool flag)
468 {
469     ipc->SetFlag(0, flag);
470 }
471
472 void FormMain::RenewFlag(void)
473 {
474     buttonFlagRenew->Text = String::Format("Renew ({0})", ipc->RenewFlag(0) ?
475         "Success" : "Failed");
476 }
477
478 void FormMain::SetEmissivity(void)
479 {
480     Dev.Setup.FixedEmissivity = ipc->SetFixedEmissivity(0,
481         Decimal::ToSingle(numericUpDownEmissivity->Value));
482 }

```

```

477
478 void FormMain::SetTempAmbient(void)
479 {
480     Dev.Setup.FixedTempAmbient = ipc->SetFixedTempAmbient(0,
481         Decimal::ToSingle(numericUpDownTempAmbient->Value));
482 }
483 void FormMain::Init(int frameWidth, int frameHeight, int frameDepth)
484 {
485     FrameWidth = frameWidth;
486     FrameHeight = frameHeight;
487     FrameSize = FrameWidth * FrameHeight;
488     FrameRatio = (double)FrameWidth / (double)FrameHeight;
489     FrameDepth = frameDepth;
490     FrameCounter1 = LastFrameCounter = FC0 = FC1 = 0;
491     timer1->Enabled = true;
492     bmp = gcnw Bitmap( FrameWidth , FrameHeight ,
493         System::Drawing::Imaging::PixelFormat::Format24bppRgb );
494     System::Drawing::Rectangle rect = System::Drawing::Rectangle(0, 0, bmp->Width,
495         bmp->Height);
496     Imaging::BitmapData^ bmpData = bmp->LockBits( rect,
497         Imaging::ImageLockMode::ReadWrite, bmp->PixelFormat );
498     int stride = bmpData->Stride;
499     bmp->UnlockBits( bmpData );
500     rgbValues = gcnw array<Byte>(stride * FrameHeight);
501     Values = gcnw array<short>(FrameSize);
502
503     groupBoxVideo->Size = Drawing::Size(FrameWidth, FrameHeight) + Margin;
504     groupBoxRangeMode->Top = groupBoxVideo->Bottom + 5;
505     groupBoxFlag->Top = groupBoxRangeMode->Bottom + 5;
506     groupBoxOptics->Top = groupBoxFlag->Bottom + 5;
507     groupBoxFile->Top = groupBoxOptics->Bottom + 5;
508     checkBoxDock->Top = groupBoxFile->Bottom + 5;
509     UpdateSize();
510     frameInitialized = true;
511 }
512 void FormMain::InitMeasureAreas(void)
513 {
514     if(MeasureAreaCount != groupBoxMeasureAreas->Controls->Count)
515     {
516         groupBoxMeasureAreas->SuspendLayout();
517         groupBoxMeasureAreas->Controls->Clear();
518
519         for(int i=0; i < MeasureAreaCount; i++)
520         {
521             Label^ lbl = gcnw Label();
522             lbl->AutoSize = true;
523             lbl->Location = System::Drawing::Point(10, i*15 + 16);
524             lbl->Name = L"labelMeasureArea" + i.ToString();
525             lbl->Text = "";
526             groupBoxMeasureAreas->Controls->Add(lbl);
527         }
528
529         groupBoxMeasureAreas->Size = Drawing::Size(groupBoxMeasureAreas->Size.Width,
530             MeasureAreaCount * 15 + 20);
531         groupBoxMeasureAreas->ResumeLayout();
532         UpdateSize();
533     }
534 }
535 void FormMain::UpdateSize(void)
536 {
537     int right = max(
538         groupBoxVideo->Location.X + groupBoxVideo->Size.Width,
539         groupBoxRangeMode->Location.X + groupBoxRangeMode->Size.Width);
540     int bottom = max(
541         groupBoxMeasureAreas->Location.Y + groupBoxMeasureAreas->Size.Height,
542         checkBoxDock->Location.Y + checkBoxDock->Size.Height);
543
544     groupBoxHostAppl->Visible = checkBoxDock->Checked && Connected;
545     if(groupBoxHostAppl->Visible)
546     {

```

```

545     groupBoxHostAppl->Location = Drawing::Point(right + 10,
groupBoxVideo->Location.Y);
546     groupBoxHostAppl->Size = Drawing::Size(FrameWidth*2 + 20, FrameHeight*2 + 20);
547     right = groupBoxHostAppl->Right;
548     if(!OldApplRect)
549     {
550         Drawing::Rectangle r = ipc->GetMainWindowRect(0);
551         OldApplRect = gcnew Drawing::Rectangle(r.Location, r.Size);
552         OldEmbeddedState = ipc->GetMainWindowEmbedded(0);
553         ipc->SetMainWindowEmbedded(0, true);
554     }
555 }
556 else
557 {
558     if(OldApplRect)
559     {
560         ipc->SetMainWindowEmbedded(0, OldEmbeddedState);
561         ipc->SetMainWindowRect(0, OldApplRect);
562         OldApplRect = nullptr;
563     }
564 }
565
566 this->Size = Drawing::Size(right + 20, bottom + 50);
567 }
568
569 void FormMain::ResizeHostAppl(void)
570 {
571     if(groupBoxHostAppl->Visible && Connected)
572     {
573         double NewWidth = this->ClientSize.Width - groupBoxHostAppl->Left - 20;
574         double NewHeight = (NewWidth + 20) / FrameRatio;
575         if((groupBoxHostAppl->Top + NewHeight + 20) > this->ClientSize.Height)
576         {
577             NewHeight = this->ClientSize.Height - groupBoxHostAppl->Top - 20;
578             NewWidth = (NewHeight - 20) * FrameRatio;
579         }
580         groupBoxHostAppl->Size = Drawing::Size((int)NewWidth, (int)NewHeight);
581         Drawing::Rectangle r =
groupBoxHostAppl->RectangleToScreen(groupBoxHostAppl->ClientRectangle);
582         r.X += 10;
583         r.Y += 20;
584         r.Width -= 20;
585         r.Height -= 30;
586         ipc->SetMainWindowRect(0, r);
587     }
588 }
589
590
591 HRESULT FormMain::OnServerStopped(int reason)
592 {
593     ReleaseIPC();
594     Graphics ^g = Graphics::FromImage bmp;
595     g->FillRectangle(gcnew SolidBrush(Color::Black), 0, 0, bmp->Width, bmp->Height);
596     delete g;
597     pictureBox->Invalidate();
598     return 0;
599 }
600
601 HRESULT FormMain::OnFrameInit(int frameWidth, int frameHeight, int frameDepth)
602 {
603     Init(frameWidth, frameHeight, frameDepth);
604     return 0;
605 }
606
607 // will work with Imager.exe release > 2.0 only:
608 HRESULT FormMain::OnNewFrameEx(void * pBuffer, FrameMetadata2 *pMetadata)
609 {
610     int sz = sizeof(FrameMetadata2);
611     unsigned char bf[88];
612     memcpy(bf, pMetadata, 88);
613     labelFrameCounter->Text = "Frame counter HW/SW: " +
pMetadata->CounterHW.ToString() + "/" + pMetadata->Counter.ToString();
614     String ^s = "PIF ";

```

```

615     for (int i = 0; i < pMetadata->PIFnDI; i++)
616         s += String::Format("  DI{0}:{1}", i + 1, (pMetadata->PIFDI >> i) & 1);
617     for (int i = 0; i < pMetadata->PIFnAI; i++)
618         s += String::Format("  AI{0}:{1}", i + 1, pMetadata->PIFAI[i]);
619     labelPIF->Text = s;
620
621     switch(pMetadata->FlagState)
622     {
623     case fsFlagOpen: labelFlag->Text = "open"; labelFlag->ForeColor = Color::Green;
624     labelFlag->BackColor = labelFlag1->BackColor; break;
625     case fsFlagClose: labelFlag->Text = "closed"; labelFlag->ForeColor =
626     Color::White; labelFlag->BackColor = Color::Red; break;
627     case fsFlagOpening: labelFlag->Text = "opening"; labelFlag->ForeColor =
628     SystemColors::WindowText; labelFlag->BackColor = Color::Yellow; break;
629     case fsFlagClosing: labelFlag->Text = "closing"; labelFlag->ForeColor =
630     SystemColors::WindowText; labelFlag->BackColor = Color::Yellow; break;
631     default: labelFlag->Text = ""; labelFlag->ForeColor = labelFlag1->ForeColor;
632     labelFlag->BackColor = labelFlag1->BackColor;
633     }
634     HotSpot = Drawing::Point((int)pMetadata->HotSpot.x, (int)pMetadata->HotSpot.y);
635     ColdSpot = Drawing::Point((int)pMetadata->ColdSpot.x, (int)pMetadata->ColdSpot.y);
636
637     return NewFrame((short*)pBuffer, pMetadata->Counter);
638 }
639
640 HRESULT FormMain::OnNewFrame(char * pBuffer, int frameCounter)
641 {
642     return NewFrame((short*)pBuffer, frameCounter);
643 }
644
645 HRESULT FormMain::NewFrame(short *ImgBuf, int frameCounter)
646 {
647     DateTime time = DateTime::Now;
648     TimeSpan ts = time - LastFrameTime;
649     List->Add(ts.TotalMilliseconds);
650     LastFrameTime = time;
651
652     FrameCounter0 = frameCounter;
653     FrameCounter1++;
654     if(ResetPending)
655     {
656         Tmax = 0;
657         FC0 = FrameCounter0;
658         FC1 = FrameCounter1;
659         ResetPending = false;
660     }
661     int fc0 = FrameCounter0 - FC0;
662     int fc1 = FrameCounter1 - FC1;
663
664     label2->Text = String::Format("Frames from Imager Application: {0}\nReceived
665     frames: {1}\nNot recveived frames: {2}",
666         fc0, fc1, fc0 - fc1);
667
668     for ( int x = 0; x < FrameSize; x++ ) Values[x] = ImgBuf[x];
669
670     if(!Painted)
671     {
672         GetBitmap bmp, Values);
673         pictureBox->Invalidate();
674         Painted = true;
675     }
676
677     return 0;
678 }
679
680 HRESULT FormMain::OnInitCompleted(void)
681 {
682     progressBarInit->Maximum = ipc->GetInitCounter(0);
683     progressBarInit->Value = 0;
684     progressBarInit->Visible = true;
685     GetSoftwareInfo();
686     GetDeviceInfo();
687     GetDeviceSetup();

```

```

682 InitMeasureAreas();
683 labelVersionAppl->Text = String::Format("Host-Appl.: {0}.{1}.{2}.{3}",
684     Dev.SoftInfo.Application->Major,
685     Dev.SoftInfo.Application->Minor,
686     Dev.SoftInfo.Application->Build,
687     Dev.SoftInfo.Application->Revision);
688 labelVersionIPC->Text = String::Format("IPC-DLL: {0}.{1}.{2}.{3}",
689     Dev.SoftInfo.IPC_DLL->Major,
690     Dev.SoftInfo.IPC_DLL->Minor,
691     Dev.SoftInfo.IPC_DLL->Build,
692     Dev.SoftInfo.IPC_DLL->Revision);
693 labelHW->Text = String::Format("HW: {0}", Dev.Info.HW_Model);
694 labelFW->Text = String::Format("FW: {0}", Dev.Info.FW_Cypress);
695 label1->Text = "Connected with #" + Dev.Info.SerialNumber.ToString();
696 Connected = true;
697 UpdateSize();
698 return S_OK;
699 }
700
701 HRESULT FormMain::OnConfigChanged(long reserved)
702 {
703     GetDeviceInfo();
704     GetDeviceSetup();
705     InitMeasureAreas();
706     return S_OK;
707 }
708
709 HRESULT FormMain::OnFileCommandReady(wchar_t *path)
710 {
711     System::Windows::Forms::MessageBox::Show( gcnew String(path), "File was saved to
712     path:", MessageBoxButtons::OK, MessageBoxIcon::Exclamation );
713     return S_OK;
714 }
715
716 HRESULT FormMain::OnNewNMEAString(wchar_t *path)
717 {
718     Text = gcnew String(path);
719     return S_OK;
720 }
721
722 System::Void FormMain::checkBoxDock_CheckedChanged(System::Object^ sender,
723 System::EventArgs^ e)
724 {
725     UpdateSize();
726 }
727
728 void FormMain::FileOpen(void)
729 {
730     if(ipc)
731     {
732         System::Windows::Forms::OpenFileDialog^ Dlg = gcnew
733         System::Windows::Forms::OpenFileDialog;
734         String^ s = "All files|*.*|Radiom.
735         files|*.ravi;*.jpg;*.tiff|Ravi|*.ravi|Avi|*.avi|Jpeg|*.jpg|Tiff|*.tiff|";
736         Dlg->Filter = s->Remove(s->Length-1);
737         Dlg->FilterIndex = 2;
738         Dlg->CheckFileExists = true;
739         if (Dlg->ShowDialog() == System::Windows::Forms::DialogResult::OK)
740         {
741             wchar_t Filename[1024];
742             pin_ptr<const wchar_t> wch = PtrToStringChars(Dlg->FileName);
743             wcsncpy_s(Filename, 1024, wch);
744
745             USHORT retVal = ipc->FileOpen(0, Filename);
746             if(retVal != 0)
747                 System::Windows::Forms::MessageBox::Show( gcnew String(Filename),
748                 "Could not open file:", MessageBoxButtons::OK, MessageBoxIcon::Error
749                 );
750         }
751     }
752 }
753
754 }
755
756 }
757
758 }

```

```

749 System::Void FormMain::pictureBox_Paint(System::Object^ sender,
System::Windows::Forms::PaintEventArgs^ e)
750 {
751     e->Graphics->DrawImage(bmp, 0, 0);
752     if (checkBoxSpots->Checked)
753     {
754         DrawReticle(e->Graphics, HotSpot, HotPen, 4);
755         DrawReticle(e->Graphics, ColdSpot, ColdPen, 4);
756     }
757 }
758
759 void FormMain::DrawReticle(Drawing::Graphics ^g, Drawing::Point point, Drawing::Pen
^pen, int l)
760 {
761     g->DrawLine(ContrastPen, point.X - l - 1, point.Y - 1, point.X + l - 1, point.Y
- 1);
762     g->DrawLine(ContrastPen, point.X - 1, point.Y - l - 1, point.X - 1, point.Y + l
- 1);
763     g->DrawLine(pen, point.X - l, point.Y, point.X + l, point.Y);
764     g->DrawLine(pen, point.X, point.Y - l, point.X, point.Y + l);
765 }
766
767 void FormMain::StartPix(){
768     String^ command = this->startPixHiddenTextBox->Text;
769     array<String^>^ split = command->Split(' ');
770     String^ path = split[1];
771     String^ flags = "";
772     for (int i = 2; i < split->Length; i++){
773         flags += split[i] + ' ';
774     }
775
776     if (!System::IO::File::Exists(path)){
777         MessageBox::Show("Invalid Path");
778     }
779
780     else{
781         startedPix = System::Diagnostics::Process::Start(path, flags);
782     }
783 }
784
785
786 }
787

```