Rev. 0.3

Using the PHC Interface

1. Copy the files in folder “Jetmarker print head solution” to a known location on your disk.

2. Add a reference to the “Jetmarker print head controller interface.dll” in your project.

3. The Namespace is “`JMPHS`”.

4. Open the interface with “`JMPHS.PHCInterface phc = new JMPHS.PHCInterface(this);`”

5. Add the event OnConnect if Async connection is wanted by “`phc.OnConnect += new JMPHS.PHCInterface.ConnectDelegate(phc_OnConnect);`”

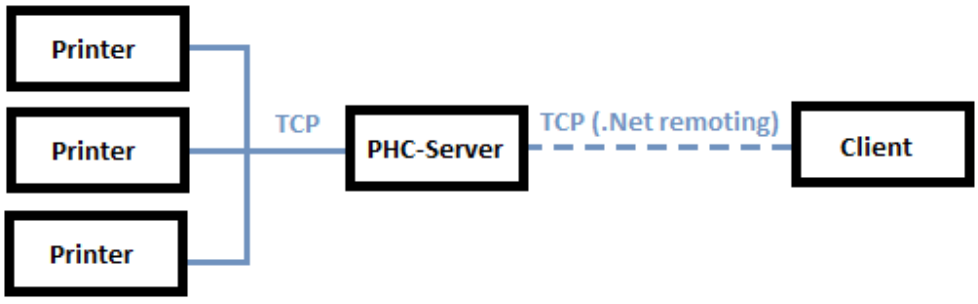
5. Connect to the PHC-Server (Print Head Controller Server) with: “`phc.ConnectAsync(System.Net.IPAddress.Parse("xxx.xxx.xxx.xxx"), 46654);`”. Changing the x’es with the IP on which the PHC-Server is running.

6. You should receive an “OnConnect” event when connected.

NOTE: you might need to open the port used (default: 46654) in your Firewall.

Please take a look at the sample codes supplied in folders:
“.\Test app. (Visual Studio 2010)\Jetmarker print head controller interface tester”
“.\Test app. (Visual Studio 2010)\WindowsFormsApplication2” (this sample is minimal)

System overview



The PHC-Server is communicating with the printer arrays directly on its own network. Any client can connect to the PHC-Server via the “PHCInterface”.

Method description

```
Default constructor.
Use "PHCInterface(ISynchronizeInvoke Synchronizer)" constructor if event callbacks are
required.
PHCInterface()

Use this constructor if event callbacks are required, supply "this" (PHCInterface(this))
as System.Windows.Forms.Form or System.Windows.Forms.Control.
PHCInterface(ISynchronizeInvoke Synchronizer)

Get information about a specific printer
SPrinter GetPrinter(int PrinterId)

Get list of all printers
SPrinter[] GetPrinters()

Get information about a specific array
SArray GetArray(int ArrayId)

Get list of all arrays
SArray[] GetArrays()

Get temperature readings for a printer
STemperature[] GetTemperatures(int PrinterId)

Connect to printer (non blocking). On success or failure the "OnConnect(...)" event is
sent.
bool ConnectAsync(IPAddress Ipaddress, int Port)

Connect tp printer (blocking).
bool Connect(IPAddress Ipaddress, int Port)

Disconnect
void Disconnect()

Add images to image queue - [NOT IMPLEMENTED]
int AddImage(System.Drawing.Image[] image, string ImageId)

Upload images to all arrays - [NOT IMPLEMENTED]
void SetImage(System.Drawing.Image[] image)

Upload an image to the array.
void SetImage(System.Drawing.Image image, int ArrayId)

Download the current image from the array - [NOT IMPLEMENTED]
System.Drawing.Image GetImage(int ArrayId)

Ping the PHC-Server with an integer, replies with the same value as supplied
int Ping(int Input)

Set temperature setpoints, overrides configuration values temporarily. Configuration
values will be re-downloaded to printer upon restart of application.
void SetTemperature(int PrinterId, double ArrayASetpoint, double ArrayBSetpoint, double
FrameASetpoint, double FrameBSetpoint)
       PrinterId = -1 will send setting to all printers
```

```
Set temperature setpoints, overrides configuration values temporarily. Configuration
values will be re-downloaded to printer upon restart of application.
void SetTemperature(int PrinterId, double ArrayASetpoint, double ArrayBSetpoint, double
FrameASetpoint, double FrameBSetpoint, double Tank)
       PrinterId = -1 will send setting to all printers

Set tickle values (nurse pulse), overrides configuration values temporarily.
Configuration values will be re-downloaded to printer upon restart of application.
void SetTickle(int PrinterId, STickleData TickleData)
       PrinterId = -1 will send setting to all printers

Start/Stop a dropwatch
void SetDropwatch(int PrinterId, SDropwatchData DropwatchData)
       PrinterId = -1 will send setting to all printers

Set printer to ignore
void SetSensorIgnore(int PrinterId, bool On)
       PrinterId = -1 will send setting to all printers
```

Events

```
Raised when the connection to PHC-Server succeeds or fails
event ConnectDelegate OnConnect

Raised when an error is detected
event OnError

Raised when a status change is detected
event OnStatus

Raised when a temperature delta of +/- 0.5 degrees is detected
event OnTemperature
```

Types

```
Temperatures sources
enum ETemperatureSource
{
    ThermalArrayA,
    ThermalArrayB,
    FrameArrayA,
    FrameArrayB,
    Tank,
    CpuBoard,
    PowerBoard
}

Temperature information from a specific printer
struct STemperature
{
    public double Temperature;
    public double Setpoint;
    public ETemperatureSource Source;
```

```
}


Printer types
enum EPrinterType
{
    DimatixPolaris512
}

Printer information structure
struct SPrinter
{
    public int Id;
    public string Name;
    public string Location;
    public EDeviceType DeviceType;
    public string PrinterTypeString;
    public IPAddress Ipaddress;

    public override string ToString()
    {
        return string.Format("{0} ({1}, {2}) ", Name, Location, PrinterTypeString);
    }
}

Array information structure
struct SArray
{
    public int Id;
    public string Name;
    public SDevice[] Devices;
    public int EncoderPulses;
    public int EncoderPulsesUnit;
    public int SensorDistance;
    public int FixedPrintSpeed;
    public double FireAmplitudeAdjustment;
    public double FireWidthAdjustement;

    public bool NurseOn;
    public double NurseRiseTime;
    public double NurseFallTime;
    public double NursePulseWidth;
    public double NurseAmplitudeAdjustment;
    public int NurseFrequency;
    public int NurseDelay;

    public double BurstAmplitudeAdjustment;
    public int BurstFrequency;
    public int BurstShots;
    public int BurstDelay;

    public override string ToString()
    {
        return Name;
    }
}
```

```
Device information structure
public struct SDevice
{
    public int Id;
    public string Name;
    public string ArrayName;
    public EDeviceType DeviceType;
    public object DeviceClass;
    public int Position;

    public override string ToString()
    {
        return string.Format("{0} ({1})", Name, ArrayName);
    }
}

Dimatix Polaris 512 information structure
public struct SPolaris512
{
    public string Name;
    public System.Net.IPAddress IpAddress;
    public int Port;
    public SFireCurve FireCurveA;
    public SFireCurve FireCurveB;
    public double TemperatureSetpointArrayA;
    public double TemperatureSetpointArrayB;
    public double TemperatureSetpointFrameA;
    public double TemperatureSetpointFrameB;
    public double TemperatureSetpointTank;
    public SJet[] Jets;
    public int DistanceArrayToA;
    public int DistanceAtoB;
}

Jet information structure
public struct SJet
{
    public int Id;
    public bool Available;
}

Fire curve information structure
public struct SFireCurve
{
    public double RiseTime;
    public double FallTime;
    public double PulseWidth;
    public double Amplitude;
}
```

```
Tickle (nurse pulse) information structure
public struct STickleData
{
    public bool On;
    public int Frequency;
    public int Delay;
    public SFireCurve NurseCurveA;
    public SFireCurve NurseCurveB;
    public double BurstAmplitudeA;
    public double BurstAmplitudeB;
    public int BurstShots;
    public int BurstFrequency;
    public int BurstDelay;
}

Dropwatch information structure
public struct SDropwatchData
{
    public bool On;
    public int Frequency;
    public int Timeout;
    public int NumberOfTimes;
    public SFireCurve FireCurveA;
    public SFireCurve FireCurveB;
}
```