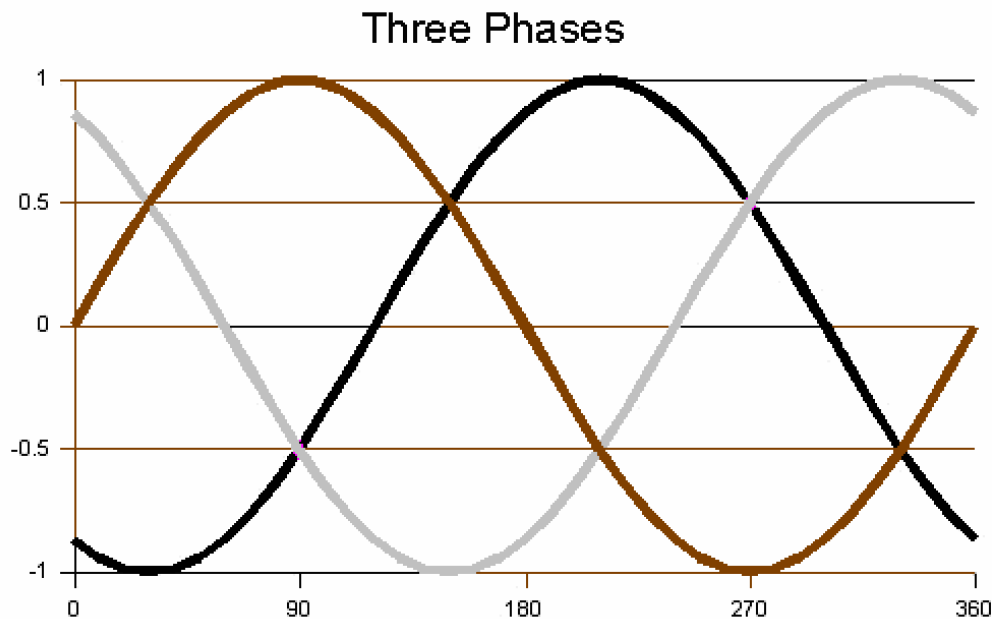


Neutrino Board as a 3-Channel Oscilloscope

Three-phase electric power in Europe for houses consists of 5 wires:

L1=Brown, L2=Black, L3=Grey, N=Blue, G=Green/yellow striped. N and G are connected.

We want to show the phase shift of the voltage (L1,N), (L2,N) and (L3,N), ...

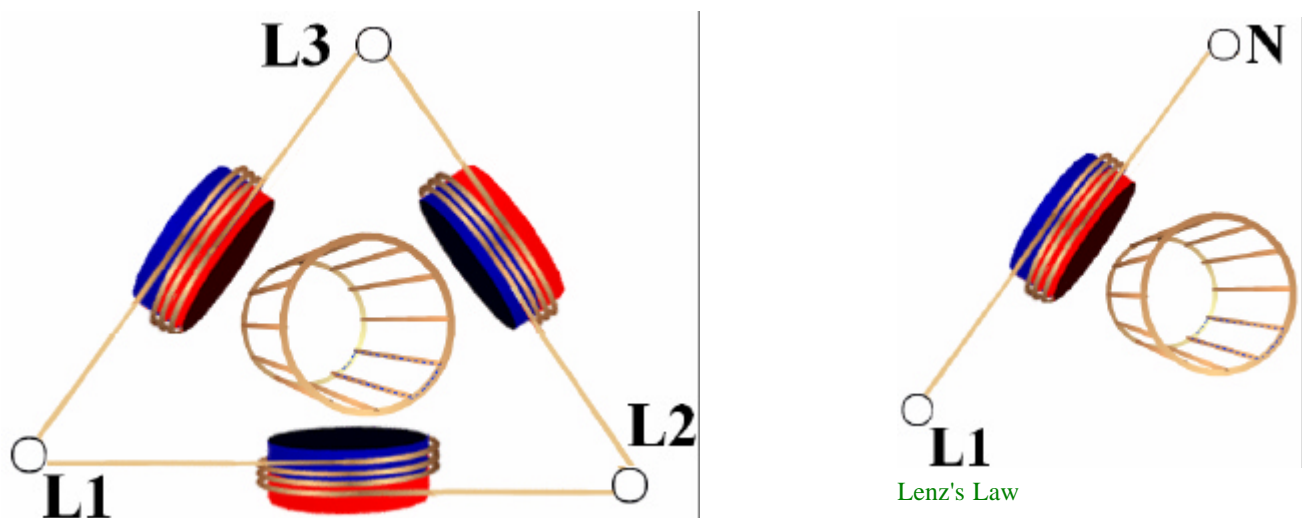


A voltage cycle of a three-phase system, from 0° to ($360^\circ = 2\pi$). The plotted lines represent the variation of instantaneous voltage (or current) with respect to time.

... to demonstrate, that this is the reason for a three-phase a.c. motor to rotate so well.

The elementary three-phase a.c. motor has a 3-coil-stator displaced 120° from each other.

When the rotor is excited with sinusoidal changing magnetic fields, eddy currents appear and the cage starts to rotate.

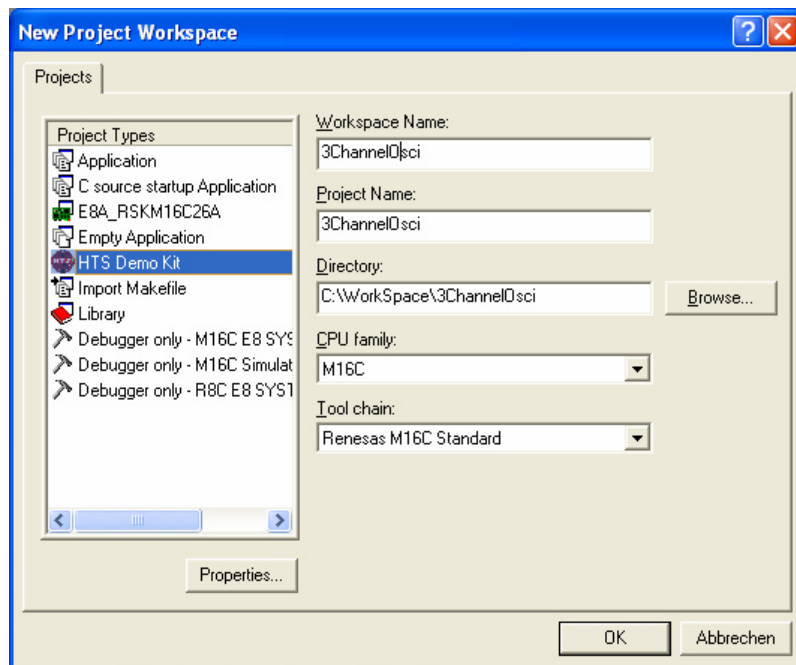


3-Channel Oscilloscope

We have three parts in our project:

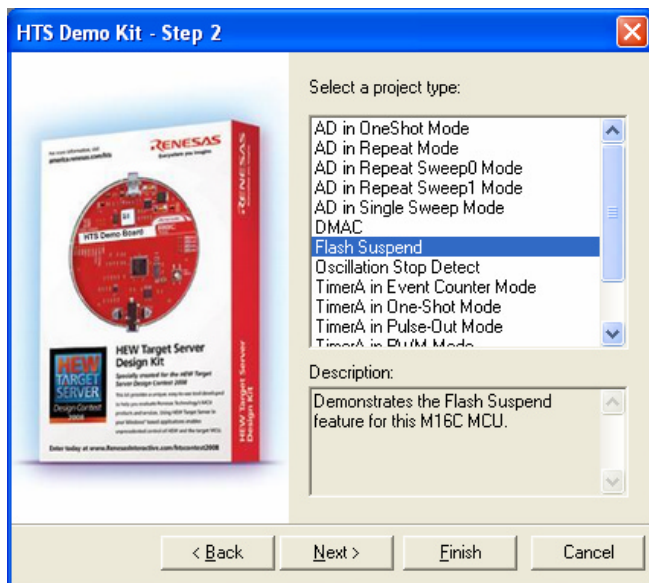
- 1:** The HEW-project: 3ChannelOsci
- 2:** The hardware to connect with the Neutrino board
- 3:** The C++ Express 2008 application 3ChannelOsci_C

To 1: File/New Workspace

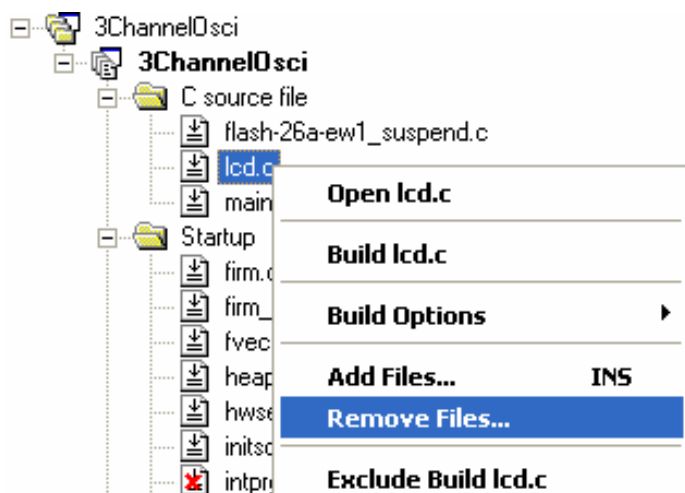


3ChannelOsci

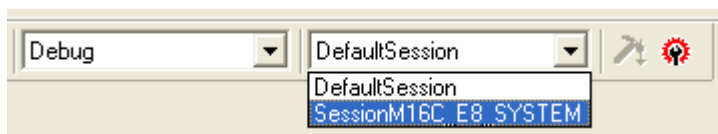
Use the demo project: Flash Suspend, in that we will delete lcd.c and manipulate main_flsh_suspend.c



Flash Suspend



Remove: lcd.c



Choose the SessionM16C_E8_SYSTEM for Debug

Change main_flash_suspend.c into:

```
#include "sfr26a.h"
#include "HTSdemoDef.h"

// EW1 CPU Rewrite function definitions with suspend
#include "flash-26A-ew1_suspend.h"
#include "main.h"

char ad_buffer[4];
```

```

void main(void)
{
    unsigned int i;
    unsigned long n;
    unsigned long flash_adr;

    p7_0 = 1; //RED_LED = LED_OFF;
    p7_2 = 1; //AMB_LED = LED_OFF;
    p7_4 = 1; //GRN_LED = LED_OFF;
    p7_6 = 1; //ORG_LED = LED_OFF;
    pd7_0 = 1; //out
    pd7_2 = 1; //out
    pd7_4 = 1; //out
    pd7_6 = 1; //out

    //ADC
    //adcon1 = 0b00101000; //10 bit mode
    adcon1 = 0b00110000; //8 bit mode
    adcon2 = 0b00000000;

    // turn on red LED to signal the measurement
    p7_0 = 0; //RED_LED = LED_ON;

    // Erase Block 2 (0xF8000 - 0xFBFFF)
    //FlashErase( 2 );

    ad_buffer[3] = 0;

    flash_adr = 0x0F8000; //0xF8000 + 0x800 = 0xF8800
    //flash_adr = 0x0F8800; //0xF8800 + 0x800 = 0xF9000
    for(i=0; i<512; i++)
    {
        adcon0 = 0b11000100; //set input to AD2 of Neutrino board = P104 = AN4
        adst=1; //start ADC
        while(adst == 1){}; //Wait for A/D conversion;
        adcon0 = 0b11000101; //set input to AD3 of Neutrino board = P105 = AN5
        adst=1; //start ADC
        while(adst == 1){}; //Wait
        adcon0 = 0b11000111; //set input to AD4 of Neutrino board = P107 = AN7
        adst=1; //start ADC
        while(adst == 1){}; //Wait
        ad_buffer[0] = ad4l; // ad4l=AD4L, the value of the lower byte of ad4
        ad_buffer[1] = ad5l;
        ad_buffer[2] = ad7l;
        FlashWrite( flash_adr, ad_buffer, 4); //always plus 4
        flash_adr = flash_adr + 4;
    }

    // Turn on the green LED to signal that the measurement is done
    p7_0 = 1; //RED_LED = LED_OFF;
    p7_4 = 0; //GRN_LED = LED_OFF;
    for(;;);
}

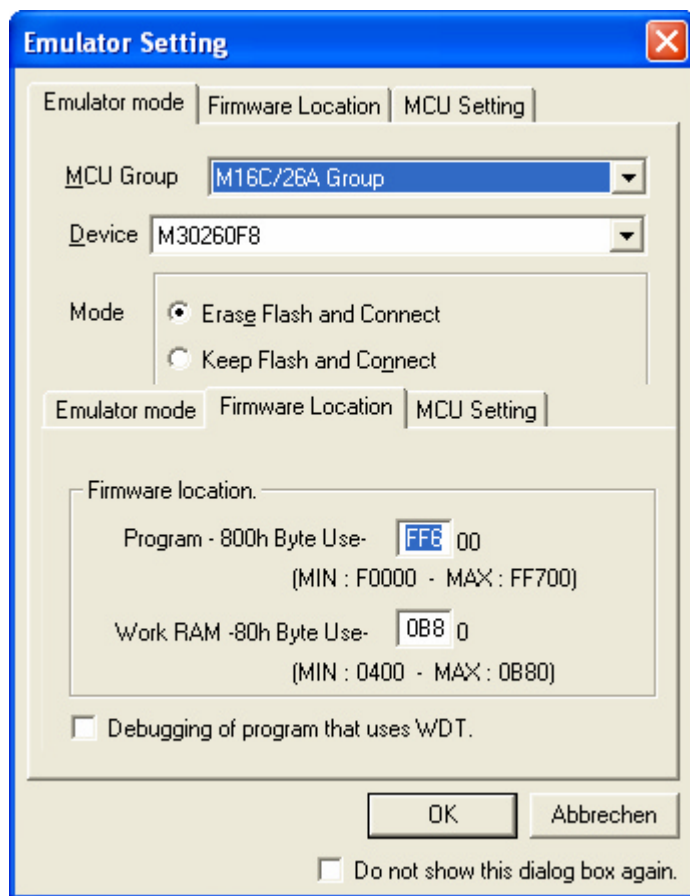
```

In HEW we choose: Setup/Emulator/System... and synchronize the Flash memory: Flash memory to PC.

A Build/Build All ends up with 0 Errors and 0 Warnings.


Now we connect the hardware for our measurement with the Neutrino board.

With Debug/Connect we download the MCU program into the Neutrino RAM/ROM.

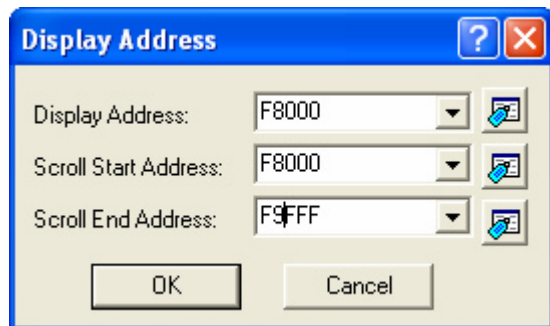


... we need no power supply, since we use USB

Choose in HEW: Debug/Reset_Go. We see for a short time the red LED, but a part of a second later the green LED signalled the end of the measurement.

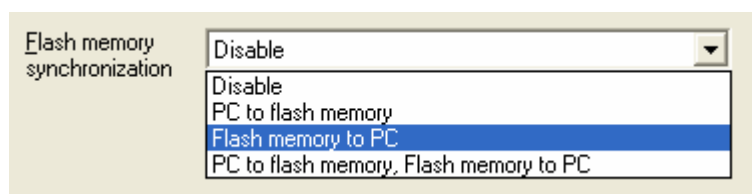
Press the red STOP  button but do not disconnect HEW with the Neutrino board.

Where are the data of our measurement? Therefore we choose in HEW: View/CPU/Memory...



```
flash_adr = 0x0F8000; //0xF8000 + 0x800
           = 0xF8800
```

You will see nothing, if you forgot “Setup/Emulator/System... and synchronize the Flash memory: Flash memory to PC”.



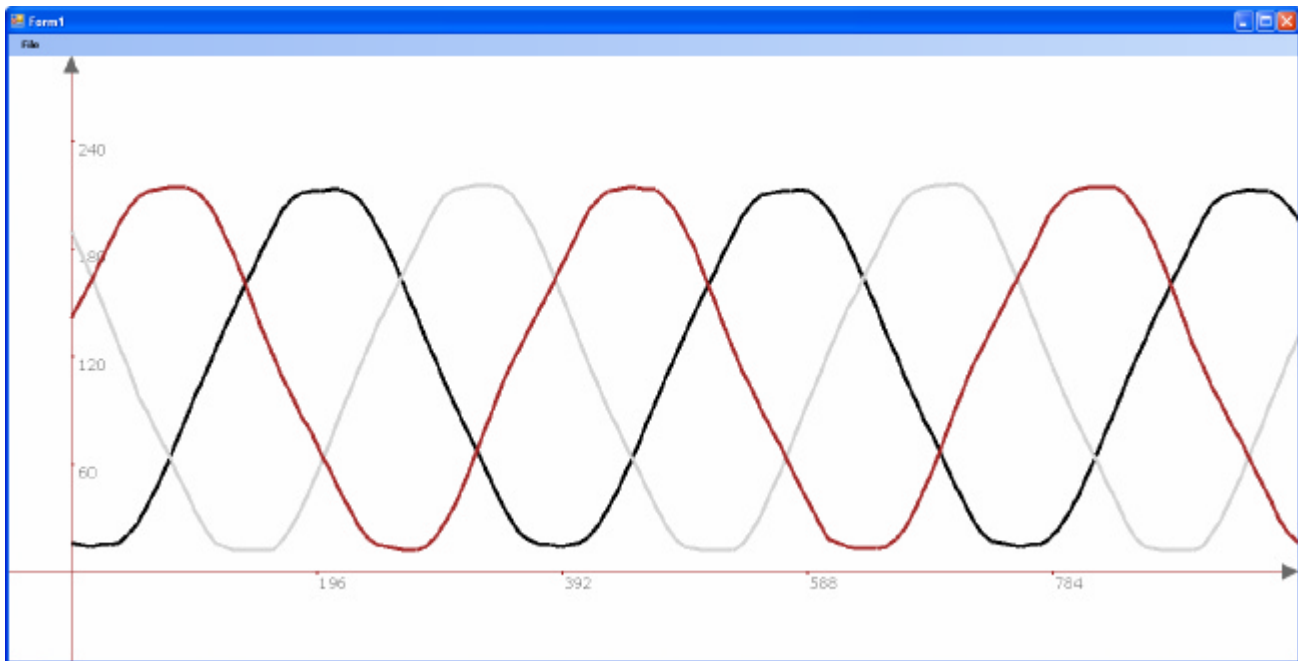
| Address | +0 | +1 | +2 | +3 | +4 | +5 | +6 | +7 | +8 | +9 | +A | +B | Refresh | |
|---------|----|----|----|----|----|----|----|----|----|----|----|----|---------|---|
| F8000 | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | F |
| F8010 | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | F |

Setup/Emulator/System... and synchronize the Flash memory: Flash memory to PC. Otherwise you will see this FF-field instead of ...

| Address | +0 | +1 | +2 | +3 | +4 | +5 | +6 | +7 | +8 | +9 | +A | +B | +C | +D | +E | +F |
|---------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| F8000 | 10 | BE | 8D | 00 | 0F | B2 | 97 | 00 | 0E | A6 | A1 | 00 | 0F | 99 | AC | 00 |
| F8010 | 0F | 8B | B7 | 00 | 10 | 7E | C2 | 00 | 15 | 71 | CB | 00 | 1B | 62 | D1 | 00 |
| F8020 | 26 | 58 | D4 | 00 | 30 | 4D | D5 | 00 | 3C | 43 | D6 | 00 | 49 | 37 | D6 | 00 |
| F8030 | 56 | 2C | D6 | 00 | 63 | 21 | D4 | 00 | 6F | 17 | CF | 00 | 7B | 10 | C7 | 00 |
| F8040 | 87 | 0E | BC | 00 | 93 | 0C | B0 | 00 | 9E | 0C | A3 | 00 | A9 | 0C | 94 | 00 |
| F8050 | B4 | 0C | 86 | 00 | BF | 0C | 79 | 00 | C9 | 11 | 6D | 00 | D0 | 19 | 62 | 00 |
| F8060 | D3 | 23 | 57 | 00 | D4 | 2F | 4E | 00 | D4 | 3B | 43 | 00 | D5 | 49 | 38 | 00 |

... our measurement data.

The C++ program of part 3 will draw these data, stored in the Flash memory at address 0x0F8000, and that looks like this:



L1=Brown, L2=Black, L3=Grey

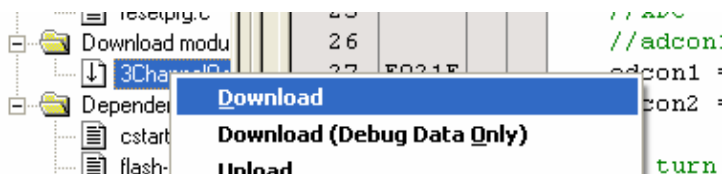
If we put the three transformers in one 3Out Power Strip, then we get one graph, since the phase shift is zero.

Therefore we change main_flsh_suspend.c:

```
//flash_adr = 0x0F8000;//0xF8000 + 0x800 = 0xF8800
```


```
flash_adr = 0x0F8800;//0xF8800 + 0x800 = 0xF9000
```

HEW: Build/Build All

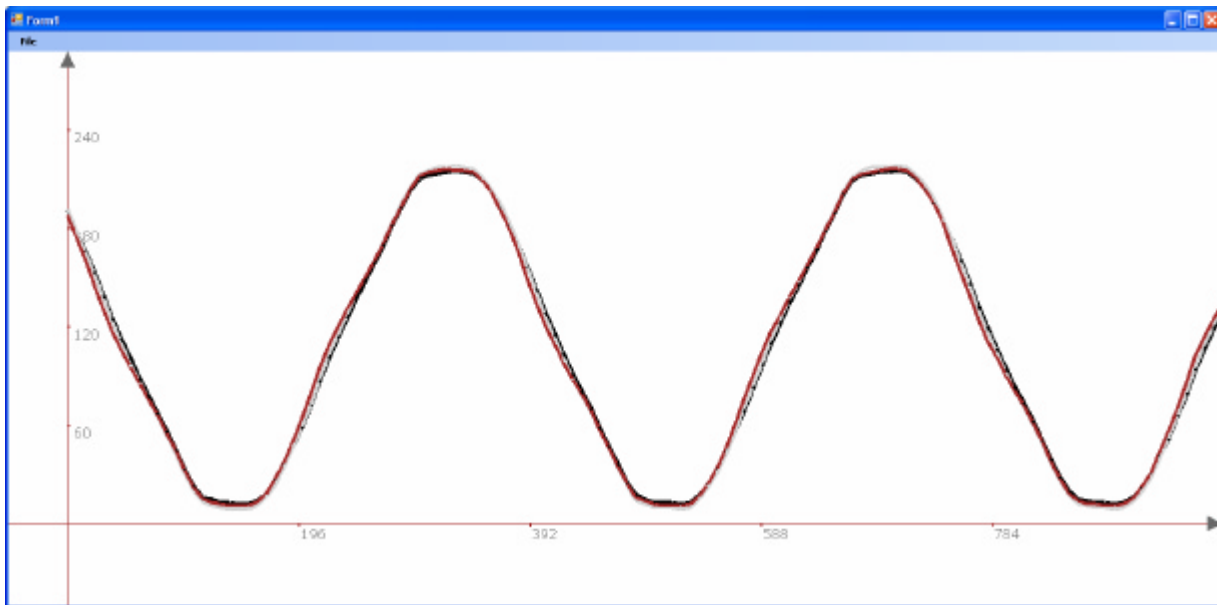


Send the changes to the Neutrino board

HEW: Debug/Reset_Go

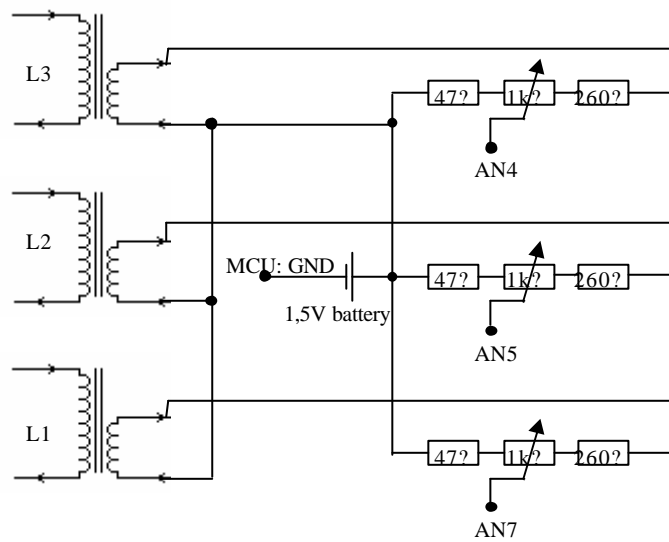
Press the red STOP  button but do not disconnect HEW with the Target board.

Start your WinXP C++ application to draw the data, stored in the Flash memory at address 0x0F8800.

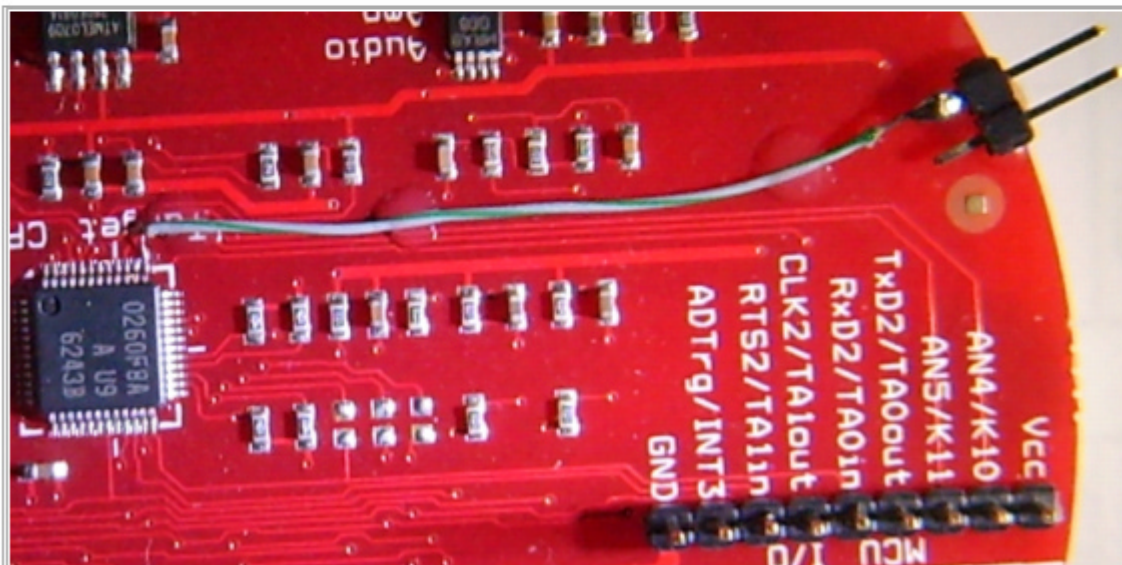


Three transformers in a 3Out Power Strip at L1

To 2: To connect L1, L2 and L3 with the Neutrino board, we use three old AC Adapters just as AC transformers; the DC functionality of the AC Adapters is not further necessary.



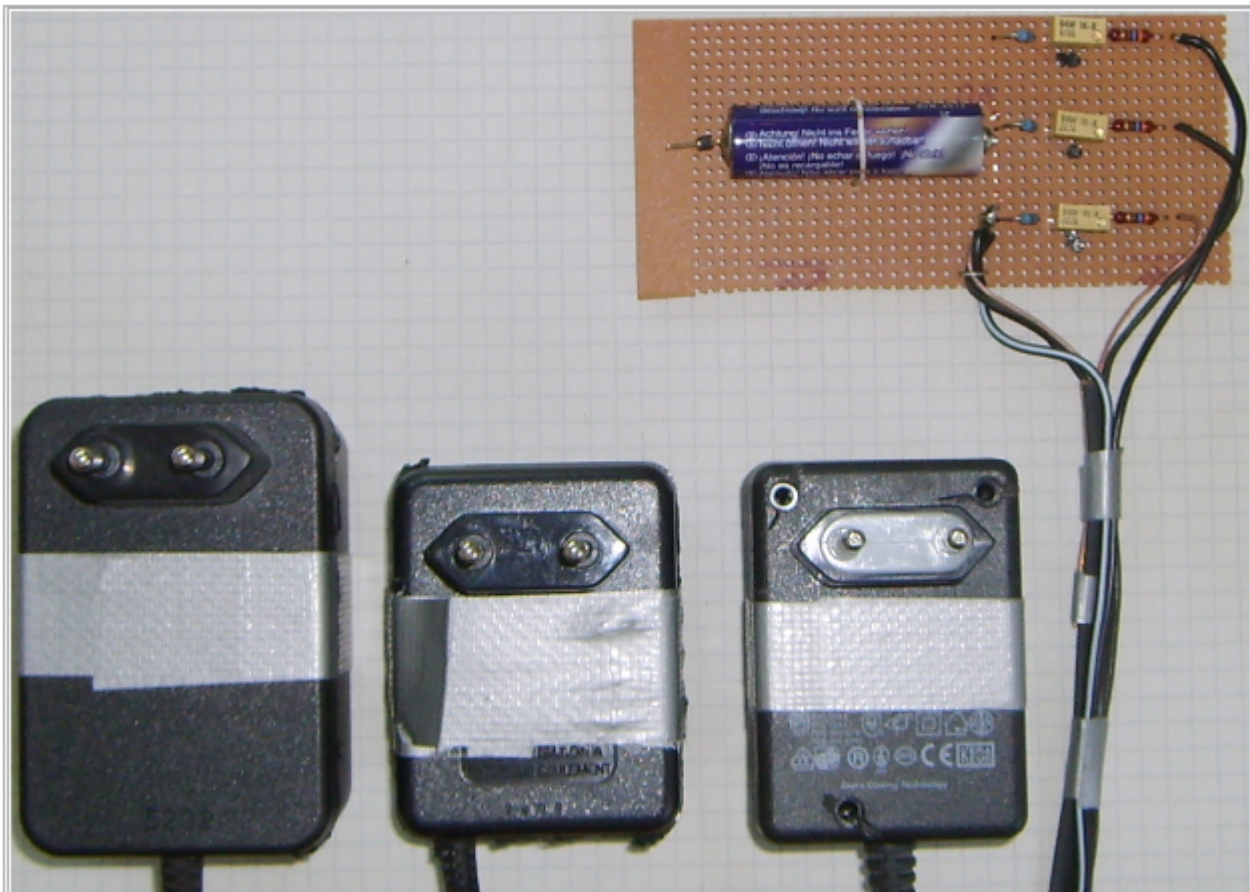
The battery lifts the voltage for the controller into a positive domain.



We use Target-CPU-Pin 37 as AN7

GND

AN5,AN4



Three old AC Adapters



L1, L2 and L3 with phase shift



Only L1 without phase shift

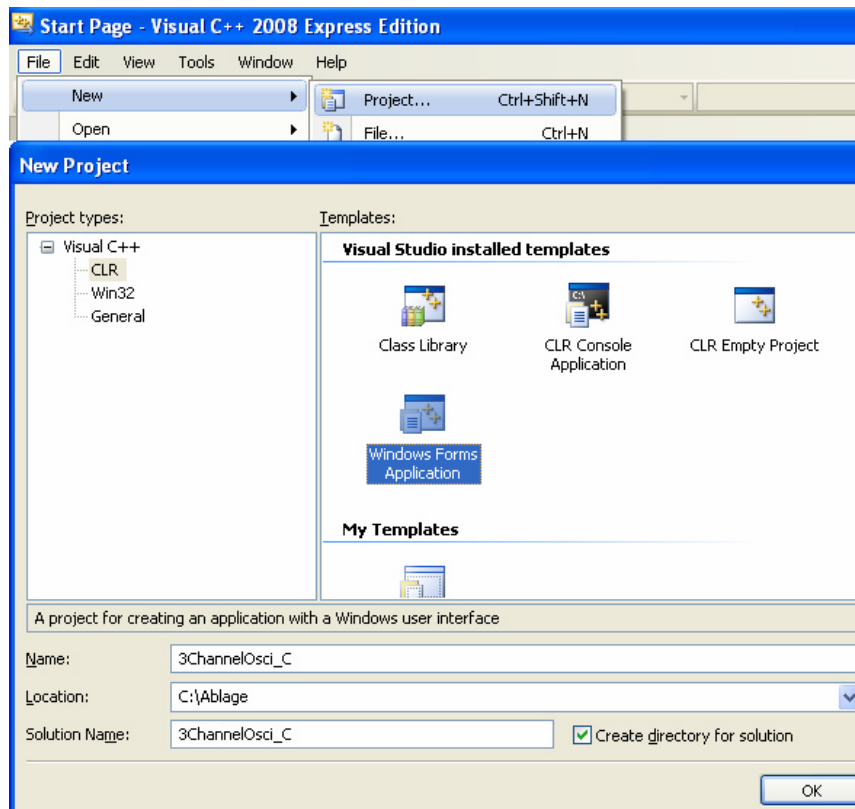
To 3:

A C++ Application to Draw the Neutrino Flash Memory Data

The next lines describe the steps in detail for you to create such an application.

You need from <http://www.microsoft.com/express/download/> the free software *Visual C++ 2008 Express Edition*.

Next start the C++ IDE and create a new project.



Create a C++/CLR/Windows Forms Application 3ChannelOsci_C in a location. My location is C:\Ablage

Create the directory C:\Ablage\3ChannelOsci_C\Interop and copy the to HTS-files C:\Programme\Renesas\HTS Demo Kit\Simple Example\interop.hewtargetserverlib.1.5.dll C:\Programme\Renesas\HTS Demo Kit\Simple Example\interop.serverlib.1.0.dll into it.

A RMB-click in Visual C++ 2008 into Form1 at the Form1.h[Design] page enables us to “View Code”.

We see something like this:

```
#pragma once
namespace My3ChannelOsci_C {
    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;
    /// <summary>
    /// Summary for Form1
```

...

To that code we add the following line: `using namespace HEWTARGETSERVERLib;`

Press the F7-key (Build/Build Solution). We get the error C2871: 'HEWTARGETSERVERLib' : a namespace with this name does not exist.

We need new project/properties... by pressing Alt + F7. Choose Add New Reference... and the page **Browse** and add the reference: C:\Ablage\3ChannelOsci_C\Interop\interop.hewtargetserverlib.1.5.dll

Press F7(Build/Build Solution) again and you will see, that the error is away.

Now we add the paint-method and a menu to our application.

Press Alt + Enter and the Properties-window opens or activates. Press the flash or bolt icon and look for the paint method and double-click the name **paint**. Visual C++ 2008 writes the code for us.

To test the method, expand the code for example to:

```
#pragma endregion
private: System::Void Form1_Paint(System::Object^ sender, System::Windows::Forms::PaintEventArgs^ e) {
    Graphics ^ g = e->Graphics;
    System::Drawing::Font^ fn = gcnew System::Drawing::Font("Verdana", 12);
    System::Drawing::SolidBrush^ br = gcnew System::Drawing::SolidBrush(Color::Blue);
    g->DrawString("Hallo", fn, br, 10, 30);
}
```

Press F5 and the YES button to compile and run the program.

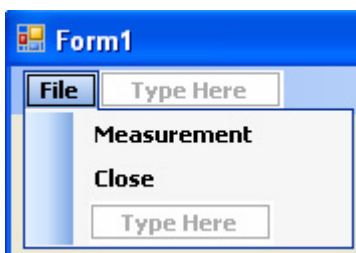
For the menu choose View/ToolBox or press Ctrl+Alt+X and click on MenuStrip among Menus & Toolbars.

Click into Form1 at the Form1.h[Design] page to add a menu.

Double-click on **Close** and add the line: `Close();`

```
private: System::Void closeToolStripMenuItem_Click(System::Object^ sender, System::EventArgs^ e) {
    Close();
}
```

Press F5 and the YES button to test the program.



To get C++ to write the code double-click on a menu entry

Now we will draw a coordinate system. Therefore we introduce some variables:


```

/// </summary>
public ref class Form1 : public System::Windows::Forms::Form
{
    public:
    Form1(void)
    {
        InitializeComponent();
    }
...

```

becomes

```

/// </summary>
public ref class Form1 : public System::Windows::Forms::Form
{
    double xmin, xmax, ymin, ymax, xstretchfactor, ystretchfactor;
    double yA;
    public:
    Form1(void)
    {
        InitializeComponent();
        yA = 0;
        xmax = 980;
        xmin = -50;
        ymax = 300;
        ymin = -50;
        xstretchfactor = this->ClientRectangle.Width / (xmax - xmin);
        ystretchfactor = this->ClientRectangle.Height / (ymax - ymin);
    }
}

```

and behind `#pragma endregion` we introduce two transformations and the coordinate-procedure.

```

int Form1::xX(double x)
{
    return (int)(xstretchfactor * (x - xmin));
}

```

```

int Form1::yY(double y)
{
    return (int)(ystretchfactor * (ymax - y));
}

```

```

void Form1::coordinatesystem(Graphics^ g, double dx, double dy, double here_y_axis)
{
    System::Drawing::Font^ fn = gcnew System::Drawing::Font("Verdana", 12);
    System::Drawing::SolidBrush^ br = gcnew System::Drawing::SolidBrush(Color::DimGray);
    Pen^ pen = gcnew Pen( Color::Brown,1.0f);
    array<Point>^ pxA =
    {
        Point(xX(xmax),yY(0)),
        Point(xX(xmax) - 18,yY(0) - 9),
        Point(xX(xmax) - 18,yY(0) + 9),
        Point(xX(xmax),yY(0))
    };
    int h = 25; //menuStrip1.Height;
    array<Point>^ pyA =

```

```

{
    Point(xX(here_y_axis) + 9, yY(ymax) + 18 + h),
    Point(xX(here_y_axis), yY(ymax)-2 + h),
    Point(xX(here_y_axis) - 9, yY(ymax) + 18 + h),
    Point(xX(here_y_axis) + 9, yY(ymax) + 18 + h)
};

g->DrawLine(pen, xX(xmin), yY(0), xX(xmax), yY(0));//x-axis
g->FillPolygon(br, pxA);//arrow x-axis

g->DrawLine(pen, xX(here_y_axis), yY(ymin), xX(here_y_axis), yY(ymax));//y-axis
g->FillPolygon(br, pyA);//arrow y-axis

double x = dx;
while (x < xmax)
{
    g->DrawString(x.ToString("#.#"), fn, br, (float)xX(x), (float)(yY(0) + 3));
    g->DrawRectangle(pen, xX(x), yY(0), 1, 3);
    x += dx;
}
x = -dx;
while (x > xmin)
{
    g->DrawString(x.ToString(), fn, br, (float)xX(x), (float)(yY(0) + 3));
    g->DrawRectangle(pen, xX(x), yY(0), 1, 3);
    x -= dx;
}
x = dy;
while (x < ymax)
{
    g->DrawString(x.ToString("0.#####"), fn, br, (float)(xX(here_y_axis) + 5), (float)yY(x));
    g->DrawRectangle(pen, xX(here_y_axis), yY(x), 3, 1);
    x += dy;
}
x = -dy;
while (x > ymin)
{
    g->DrawString(x.ToString("0.#####"), fn, br, (float)(xX(here_y_axis) + 5), (float)yY(x));
    g->DrawRectangle(pen, xX(here_y_axis), yY(x), 3, 1);
    x -= dy;
}
}

```

and we change the paint-method to:

```

System::Void Form1_Paint(System::Object^ sender, System::Windows::Forms::PaintEventArgs^ e) {
    Graphics ^ g = e->Graphics;
    g->Clear(Color::White);
    coordinatesystem(g, xmax / 5, ymax / 5, yA);
}

```

Press F5 and the YES button to test the program.

As you know how to introduce and initialize for example the variable xmax, we introduce and initialize now variables to use HTS – the Target Server.

```
HEWTARGETSERVERLib::HewServer1Class^ hts;
double dx, x_start;
unsigned int adr, gDMA_DataBuff_Address, n_gDMA_DataBuff, ARRAY_SIZE;

n_gDMA_DataBuff = 2048 / 4;
ARRAY_SIZE = 0x800;
gDMA_DataBuff_Address = 0xF8000;
dx = xmax/n_gDMA_DataBuff;
x_start = 0;
adr = gDMA_DataBuff_Address;
try //connect to HEW server
{
    hts = gcnew HEWTARGETSERVERLib::HewServer1Class;
}
catch (Exception ^ e)
{
    String ^ msg = e->Message; // avoid warning
    hts = nullptr;
}
if (hts == nullptr)//3
{
    String ^ msg = "Missing Target Server Library"+ Environment::NewLine;
    MessageBox::Show(msg, "Error", MessageBoxButtons::OK, MessageBoxIcon::Error);
    return;
}

cli::array<unsigned char>^ GetABC_data(unsigned int address)
{
    cli::array<unsigned char> ^ data = gcnew array<unsigned char>(ARRAY_SIZE);
    cli::array<unsigned char> ^% tdata = data;
    try
    {
        if (nullptr != hts)
            hts->GetDirectMemory(address, address + (ARRAY_SIZE - 1), 1, tdata);
        return data;
    }
    catch (Exception ^ exception)
    {
        Console::WriteLine(exception->Message);
    }
    return data;
}

void Form1::draw_the_3_channels(Graphics^ g)//5
{
    System::Drawing::Pen^ myPen = gcnew System::Drawing::Pen(System::Drawing::Color::Green);
    myPen->Width = 4.0f;
    int xAvon, yAvon, xAbis, yAbis;
    int xBvon, yBvon, xBbis, yBbis;
    int xCvon, yCvon, xCbis, yCbis;
    unsigned int n;
```

```

//read the Neutrino board AD_values(AN4, AN5, AN7)
cli::array<unsigned char>^ y = GetABC_data(adr);
if (x_start <= xmax)
{
    xAvon = xX(x_start); yAvon = yY(y[0+0]);
    xBvon = xX(x_start); yBvon = yY(y[0+1]);
    xCvon = xX(x_start); yCvon = yY(y[0+2]);
    for (n = 0; n < ARRAY_SIZE; n+=4)
    {
        myPen->Color = System::Drawing::Color::Black;
        xAbis = xX(x_start); yAbis = yY(y[n+0]);
        g->DrawLine(myPen, xAvon, yAvon, xAbis, yAbis);
        xAvon = xAbis; yAvon = yAbis;
        myPen->Color = System::Drawing::Color::LightGray;
        xBbis = xX(x_start); yBbis = yY(y[n+1]);
        g->DrawLine(myPen, xBvon, yBvon, xBbis, yBbis);
        xBvon = xBbis; yBvon = yBbis;
        myPen->Color = System::Drawing::Color::Brown;
        xCbis = xX(x_start); yCbis = yY(y[n+2]);
        g->DrawLine(myPen, xCvon, yCvon, xCbis, yCbis);
        xCvon = xCbis; yCvon = yCbis;
        x_start = x_start + 4*dx;
    }
}
}

```

and again we change the paint-method to:

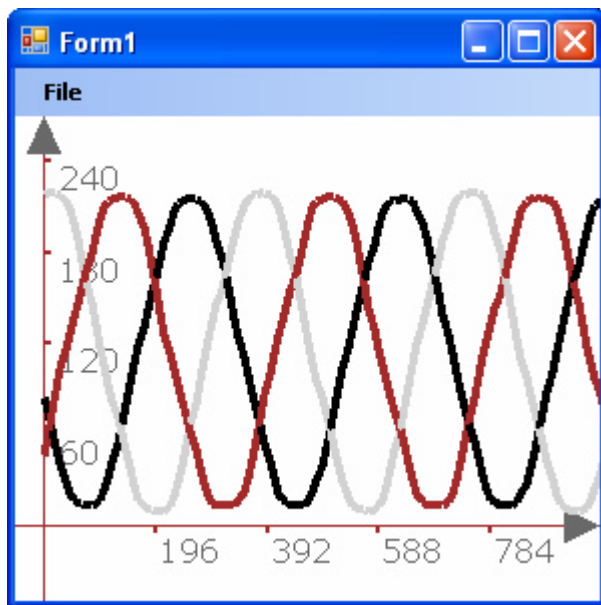
```

System::Void Form1_Paint(System::Object^ sender, System::Windows::Forms::PaintEventArgs^ e) {
    Graphics ^ g = e->Graphics;
    g->Clear(Color::White);
    coordinatesystem(g, xmax / 5, ymax / 5, yA);
    draw_the_3_channels(g);
}

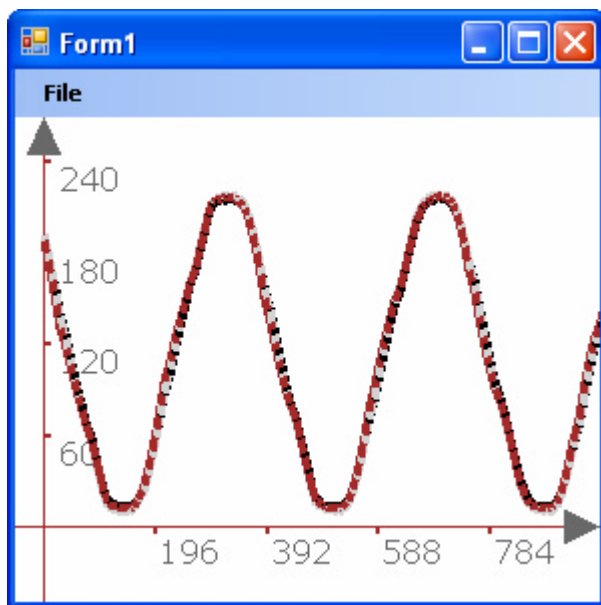
```

Now we start HEW, take a measurement of L1, L2 and L3 with the Neutrino board at Flash-address 0x0F8000 and by keeping the Flash memory a second measurement of L1, L2=L1 and L3=L1 at Flash-address 0x0F8800.

If we compile the C++ program with address: gDMA_DataBuff_Address = 0xF8000; we get:



And if we compile the C++ program with address: `gDMA_DataBuff_Address = 0xF8800;` we get:



With the menu entry **Measurement** we can switch between 0xF8000 and 0xF8800.

Click into Form1 at the Form1.h[Design] page and then double-click on **Measurement** and add the lines:

```
System::Void measurementToolStripMenuItem_Click(System::Object^ sender, System::EventArgs^ e) {
    x_start = 0;
    if(gDMA_DataBuff_Address == 0xF8000)
        gDMA_DataBuff_Address = 0xF8800;
    else
        gDMA_DataBuff_Address = 0xF8000;
    adr = gDMA_DataBuff_Address;
    this->Refresh();
}
```

At the end we spend our C++ application the ability to follow window-size-changes by users. According to the introduction of the paint-method, we double-click in the Properties-window of Form1 on **SizeChanged** to create a corresponding method and add the lines:

```
System::Void Form1_SizeChanged(System::Object^ sender, System::EventArgs^ e) {
    xstretchfactor = this->ClientRectangle.Width / (xmax - xmin);
    ystretchfactor = this->ClientRectangle.Height / (ymax - ymin);
    x_start = 0;
    adr = gDMA_DataBuff_Address;
    this->Refresh();
}
```

If you change `x_start = x_start + 4*dx;` into `x_start = x_start + dx;` the program will draw more data – try it!

The whole C++ file is this:

```
#pragma once
```

```
namespace My3ChannelOsci_C {

    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;
    using namespace HEWTARGETSERVERLib;

    /// <summary>
    /// Summary for Form1
    ///
    /// WARNING: If you change the name of this class, you will need to change the
    /// 'Resource File Name' property for the managed resource compiler tool
    /// associated with all .resx files this class depends on. Otherwise,
    /// the designers will not be able to interact properly with localized
    /// resources associated with this form.
    /// </summary>
    public ref class Form1 : public System::Windows::Forms::Form
    {
        double xmin, xmax, ymin, ymax, xstretchfactor, ystretchfactor;
        double yA;
        HEWTARGETSERVERLib::HewServer1Class^ hts;
        double dx, x_start;
        unsigned int adr, gDMA_DataBuff_Address, n_gDMA_DataBuff, ARRAY_SIZE;

    public:
        Form1(void)
        {
            InitializeComponent();
            yA = 0;

            xmax = 980;
            xmin = -50;
            ymax = 300;
            ymin = -50;
            xstretchfactor = this->ClientRectangle.Width / (xmax - xmin);
            ystretchfactor = this->ClientRectangle.Height / (ymax - ymin);
            n_gDMA_DataBuff = 2048 / 4;
            ARRAY_SIZE = 0x800;
            gDMA_DataBuff_Address = 0xF8000;
            dx = xmax/n_gDMA_DataBuff;
            x_start = 0;
            adr = gDMA_DataBuff_Address;
            try //connect to HEW server
            {
                hts = gcnew HEWTARGETSERVERLib::HewServer1Class;
            }
            catch (Exception ^ e)
            {
                String ^ msg = e->Message; // avoid warning
                hts = nullptr;
            }
            if (hts == nullptr)//3
            {
                String ^ msg = "Missing Target Server Library" + Environment::NewLine;
            }
        }
    };
}
```

```

        MessageBox::Show(msg, "Error", MessageBoxButtons::OK, MessageBoxIcon::Error);
        return;
    }
}

protected:
    /// <summary>
    /// Clean up any resources being used.
    /// </summary>
    ~Form1()
    {
        if (components)
        {
            delete components;
        }
    }

private: System::Windows::Forms::MenuStrip^ menuStrip1;
private: System::Windows::Forms::ToolStripMenuItem^ fileToolStripMenuItem;
private: System::Windows::Forms::ToolStripMenuItem^ measurementToolStripMenuItem;
private: System::Windows::Forms::ToolStripMenuItem^ closeToolStripMenuItem;

protected:

private:
    /// <summary>
    /// Required designer variable.
    /// </summary>
    System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
    /// <summary>
    /// Required method for Designer support - do not modify
    /// the contents of this method with the code editor.
    /// </summary>
    void InitializeComponent(void)
    {
        this->menuStrip1 = (gcnew System::Windows::Forms::MenuStrip());
        this->fileToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
        this->measurementToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
        this->closeToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
        this->menuStrip1->SuspendLayout();
        //
        // menuStrip1
        //
        this->menuStrip1->Items->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(1) { this-
>fileToolStripMenuItem});
        this->menuStrip1->Location = System::Drawing::Point(0, 0);
        this->menuStrip1->Name = L"menuStrip1";
        this->menuStrip1->Size = System::Drawing::Size(292, 24);
        this->menuStrip1->TabIndex = 0;
        this->menuStrip1->Text = L"menuStrip1";
        //
        // fileToolStripMenuItem
        //
        this->fileToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(2)
{ this->measurementToolStripMenuItem,
    this->closeToolStripMenuItem});
        this->fileToolStripMenuItem->Name = L"fileToolStripMenuItem";
        this->fileToolStripMenuItem->Size = System::Drawing::Size(38, 20);
        this->fileToolStripMenuItem->Text = L"File";
        //
        // measurementToolStripMenuItem
        //
        this->measurementToolStripMenuItem->Name = L"measurementToolStripMenuItem";
        this->measurementToolStripMenuItem->Size = System::Drawing::Size(164, 22);
        this->measurementToolStripMenuItem->Text = L"Measurement";
        this->measurementToolStripMenuItem->Click += gcnew System::EventHandler(this,
&Form1::measurementToolStripMenuItem_Click);
        //
        // closeToolStripMenuItem
        //
        this->closeToolStripMenuItem->Name = L"closeToolStripMenuItem";
        this->closeToolStripMenuItem->Size = System::Drawing::Size(164, 22);
        this->closeToolStripMenuItem->Text = L"Close";
        this->closeToolStripMenuItem->Click += gcnew System::EventHandler(this, &Form1::closeToolStripMenuItem_Click);
        //
        // Form1
        //
        this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
        this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
        this->ClientSize = System::Drawing::Size(292, 266);
        this->Controls->Add(this->menuStrip1);
        this->MainMenuStrip = this->menuStrip1;
        this->Name = L"Form1";
    }

```

```

        this->Text = L"Form1";
        this->Paint += gcnew System::Windows::Forms::PaintEventHandler(this, &Form1::Form1_Paint);
        this->SizeChanged += gcnew System::EventHandler(this, &Form1::Form1_SizeChanged);
        this->menuStrip1->ResumeLayout(false);
        this->menuStrip1->PerformLayout();
        this->ResumeLayout(false);
        this->PerformLayout();
    }
#pragma endregion

    int Form1::xX(double x)
    {
        return (int)(xstretchfactor * (x - xmin));
    }

    int Form1::yY(double y)
    {
        return (int)(ystretchfactor * (ymax - y));
    }

    void Form1::coordinatesystem(Graphics^ g, double dx, double dy, double here_y_axis)
    {
        System::Drawing::Font^ fn = gcnew System::Drawing::Font("Verdana", 12);
        System::Drawing::SolidBrush^ br = gcnew System::Drawing::SolidBrush(Color::DimGray);
        Pen^ pen = gcnew Pen( Color::Brown, 1.0f);

        array<Point>^ pxA =
        {
            Point(xX(ymax), yY(0)),
            Point(xX(ymax) - 18, yY(0) - 9),
            Point(xX(ymax) - 18, yY(0) + 9),
            Point(xX(ymax), yY(0))
        };
        int h = 25; //menuStrip1.Height;
        array<Point>^ pyA =
        {
            Point(xX(here_y_axis) + 9, yY(ymax) + 18 + h),
            Point(xX(here_y_axis), yY(ymax)-2 + h),
            Point(xX(here_y_axis) - 9, yY(ymax) + 18 + h),
            Point(xX(here_y_axis) + 9, yY(ymax) + 18 + h)
        };

        g->DrawLine(pen, xX(xmin), yY(0), xX(ymax), yY(0)); //x -axis
        g->FillPolygon(br, pxA); //arrow x-axis

        g->DrawLine(pen, xX(here_y_axis), yY(ymin), xX(here_y_axis), yY(ymax)); //y-axis
        g->FillPolygon(br, pyA); //arrow y-axis

        double x = dx;
        while (x < ymax)
        {
            g->DrawString(x.ToString("#.#"), fn, br, (float)xX(x), (float)(yY(0) + 3));
            g->DrawRectangle(pen, xX(x), yY(0), 1, 3);
            x += dx;
        }
        x = -dx;
        while (x > xmin)
        {
            g->DrawString(x.ToString(), fn, br, (float)xX(x), (float)(yY(0) + 3));
            g->DrawRectangle(pen, xX(x), yY(0), 1, 3);
            x -= dx;
        }
        x = dy;
        while (x < ymax)
        {
            g->DrawString(x.ToString("0.####"), fn, br, (float)(xX(here_y_axis) + 5), (float)yY(x));
            g->DrawRectangle(pen, xX(here_y_axis), yY(x), 3, 1);
            x += dy;
        }
        x = -dy;
        while (x > ymin)
        {
            g->DrawString(x.ToString("0.####"), fn, br, (float)(xX(here_y_axis) + 5), (float)yY(x));
            g->DrawRectangle(pen, xX(here_y_axis), yY(x), 3, 1);
            x -= dy;
        }
    }

    cli::array<unsigned char>^ GetABC_data(unsigned int address)
    {
        cli::array<unsigned char> ^ data = gcnew array<unsigned char>(ARRAY_SIZE);
        cli::array<unsigned char> ^% tdata = data;
        try
        {

```

```

        if (nullptr != hts)
            hts->GetDirectMemory(address, address + (ARRAY_SIZE - 1), 1, tdata);
        return data;
    }
    catch (Exception ^ exception)
    {
        Console::WriteLine(exception->Message);
    }
    return data;
}

void Form1::draw_the_3_channels(Graphics^ g)//5
{
    System::Drawing::Pen^ myPen = gcnew System::Drawing::Pen(System::Drawing::Color::Green);
    myPen->Width = 4.0f;
    int xAvon, yAvon, xAbis, yAbis;
    int xBvon, yBvon, xBbis, yBbis;
    int xCvon, yCvon, xCbis, yCbis;
    unsigned int n;
    //read the Neutrino board AD_values(AN4, AN5, AN7)
    cli::array<unsigned char>^ y = GetABC_data(adr);
    if (x_start <= xmax)
    {
        xAvon = xX(x_start); yAvon = yY(y[0+0]);
        xBvon = xX(x_start); yBvon = yY(y[0+1]);
        xCvon = xX(x_start); yCvon = yY(y[0+2]);
        for (n = 0; n < ARRAY_SIZE; n+=4)
        {
            myPen->Color = System::Drawing::Color::Black;
            xAbis = xX(x_start); yAbis = yY(y[n+0]);
            g->DrawLine(myPen, xAvon, yAvon, xAbis, yAbis);
            xAvon = xAbis; yAvon = yAbis;
            myPen->Color = System::Drawing::Color::LightGray;
            xBbis = xX(x_start); yBbis = yY(y[n+1]);
            g->DrawLine(myPen, xBvon, yBvon, xBbis, yBbis);
            xBvon = xBbis; yBvon = yBbis;
            myPen->Color = System::Drawing::Color::Brown;
            xCbis = xX(x_start); yCbis = yY(y[n+2]);
            g->DrawLine(myPen, xCvon, yCvon, xCbis, yCbis);
            xCvon = xCbis; yCvon = yCbis;
            x_start = x_start + 4*dx;
        }
    }
}

private: System::Void Form1_Paint(System::Object^ sender, System::Windows::Forms::PaintEventArgs^ e) {
    Graphics ^ g = e->Graphics;
    //System::Drawing::Font^ fn =gcnew System::Drawing::Font("Verdana", 12);
    //System::Drawing::SolidBrush^ br = gcnew System::Drawing::SolidBrush(Color::Blue);
    //g->DrawString("Hallo", fn, br, 10, 30);
    g->Clear(Color::White);
    coordinatesystem(g, xmax / 5, ymax / 5, yA);
    draw_the_3_channels(g);
}

private: System::Void closeToolStripMenuItem_Click(System::Object^ sender, System::EventArgs^ e) {
    Close();
}

private: System::Void measurementToolStripMenuItem_Click(System::Object^ sender, System::EventArgs^ e) {
    x_start = 0;
    if(gDMA_DataBuff_Address == 0xF8000)
        gDMA_DataBuff_Address = 0xF8800;
    else
        gDMA_DataBuff_Address = 0xF8000;
    adr = gDMA_DataBuff_Address;
    this->Refresh();
}

private: System::Void Form1_SizeChanged(System::Object^ sender, System::EventArgs^ e) {
    xstretchfactor = this->ClientRectangle.Width / (xmax - xmin);
    ystretchfactor = this->ClientRectangle.Height / (ymax - ymin);
    x_start = 0;
    adr = gDMA_DataBuff_Address;
    this->Refresh();
}

};
}

```

... have fun,

Edgar Marx
edgarmarx@t-online.de

