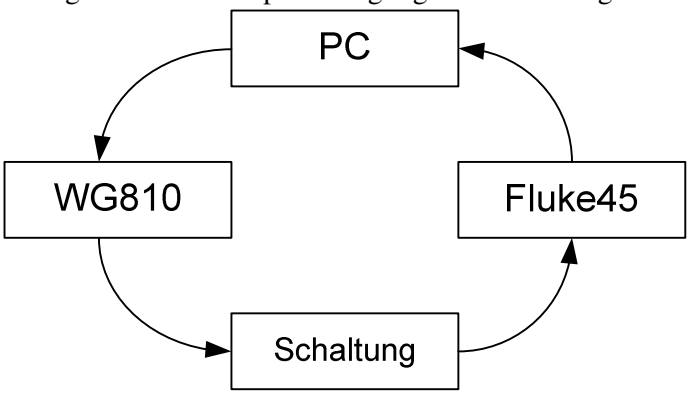
Gegenstand	Klasse	Datum		Name	Seite
LAB5	5xHMIA			Frequenzgang	1 von 24
Übung			Gruppe		Punkte
01					

AUFGABENSTELLUNG

Schreiben Sie ein Programm am PC unter JAVA, das mit Hilfe eines Frequenzgenerators und eines Messgerätes den Amplitudengang eines beliebigen Vierpols aufnehmen kann.



gegeben:

- Klasse SerialInterface
- Klasse WG810
- Klasse Fluke45 (rudimentär)

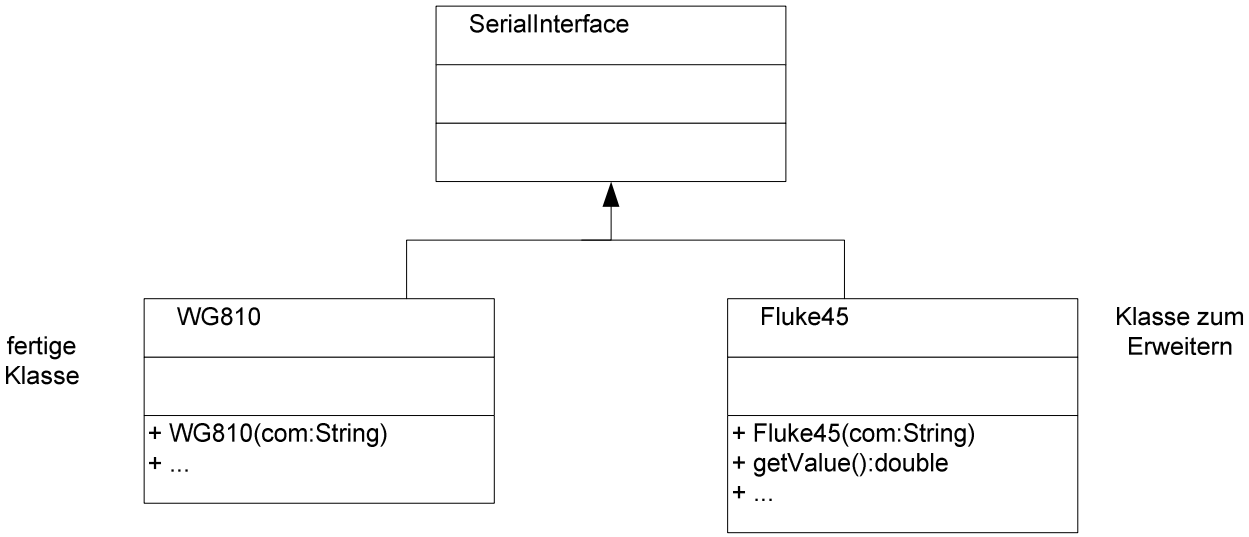
a) Lesen Sie über eine einfache Methode Daten vom Fluke45 Multimeter über die serielle Schnittstelle.

b) Schreiben Sie ein vollständiges Programm, das den Amplitudengang eines Vierpols aufnehmen kann. (WG810 einstellen, Daten in Liste speichern, Liste in Datei speichern)

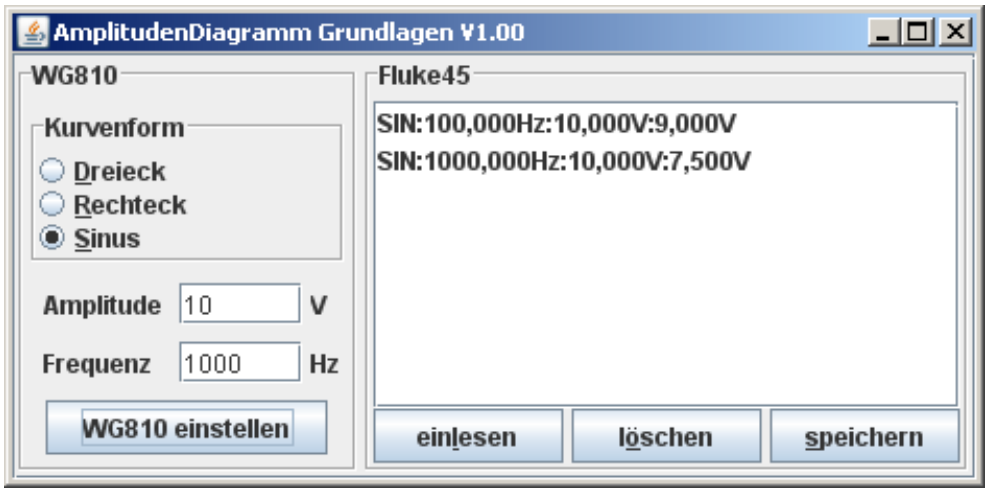
c) Erweitern Sie das Programm für das automatische Aufnehmen des Amplitudengangs eines Vierpols. (Anfangsfrequenz, Endfrequenz, Anzahl der Punkte, Amplitude, Schrittzeit, …)

d) Erweitern Sie die Klasse Fluke45 um ‚sinnvolle' Funktionen.

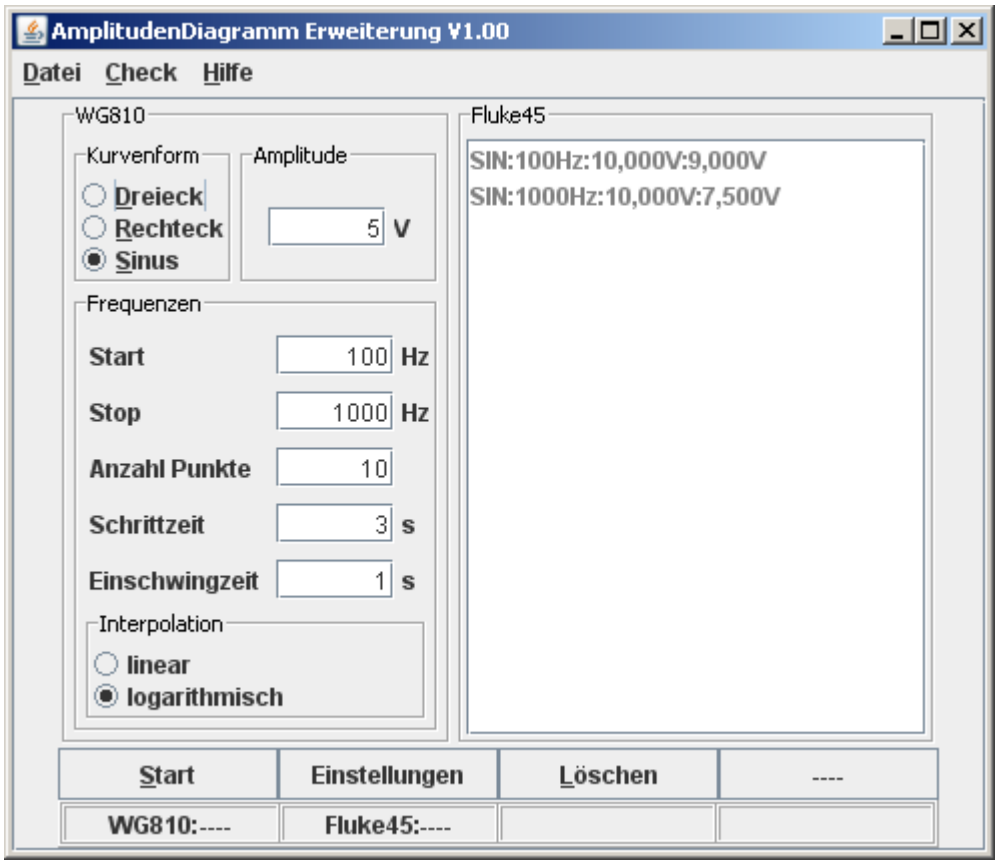
Klassendiagramme:



ad b) Grundlagen --- Vorschlag für GUI:



ad c) Erweiterung --- Vorschlag für GUI



Weitere Klassen, die zur Anwendung kommen können:

- Messwert
- Messwerte
- EinstellungenDlg

Klasse **SerialInterface**:

```
package la1ue03;

import java.io.*;
import java.util.*;
import javax.comm.*;

public class SerialInterface
{
    private CommPortIdentifier portIdentifier = null;
    private SerialPort         port = null;
    private OutputStream       os = null;
    private OutputStreamWriter osw = null;
    private BufferedWriter     bw = null;
    private InputStream        is = null;
    private InputStreamReader  isr = null;
    private BufferedReader     br = null;

    public SerialInterface(String name) throws Exception
    {
        Enumeration e =  CommPortIdentifier.getPortIdentifiers();
        while (e.hasMoreElements())
        {
            CommPortIdentifier cpi = (CommPortIdentifier)e.nextElement();
            System.out.println(cpi.getName());
            if (cpi.getPortType()==CommPortIdentifier.PORT_SERIAL)
                if (cpi.getName().equalsIgnoreCase(name))
                {
                    portIdentifier = cpi;
                    break;
                }
        }
        if (portIdentifier==null)
            throw new Exception("port not found!");
    }

    static Vector<String> getAllInterfaces()
    {
        Vector<String> v = new Vector<String>();
        Enumeration e =  CommPortIdentifier.getPortIdentifiers();
        while (e.hasMoreElements())
        {
            CommPortIdentifier cpi = (CommPortIdentifier)e.nextElement();
            System.out.println(cpi.getName());
            if (cpi.getPortType()==CommPortIdentifier.PORT_SERIAL)
            {
                v.add(cpi.getName());
            }
        }

        return v;
    }

    public void open() throws Exception
    {
        if (port!=null)
```

```
            close();
        port = (SerialPort)portIdentifier.open("SerialInterface",3000);
        port.enableReceiveTimeout(3000);
        os = port.getOutputStream();
        is = port.getInputStream();

        osw = new OutputStreamWriter(os);
        bw = new BufferedWriter(osw);

        isr = new InputStreamReader(is);
        br = new BufferedReader(isr);
    }

    public void close()
    {
        if (port!=null)
        {
            port.close();
            port=null;
        }
    }

    public void write(byte[] buffer) throws Exception
    {
        os.write(buffer);
        os.flush();
    }

    public void write(String s) throws Exception
    {
        bw.write(s);
        bw.flush();
    }

    public String readLine() throws Exception
    {
        return(br.readLine());
    }

    public byte[] read(int len) throws Exception
    {
        byte[] buffer = new byte[len];
        if (is.read(buffer,0,len)<len)
            throw new Exception("timeout occured!");
        return buffer;
    }

    public void setSerialPortParams(int baudrate,int dataBits,
            int stopBits,int parity)
            throws UnsupportedCommOperationException
    {
        if (port!=null)
            port.setSerialPortParams(baudrate,dataBits,stopBits,parity);
    }

    public void setDTR(boolean b)
    {
```

```
        if(port!=null)
            port.setDTR(b);
    }

    public void setRTS(boolean b)
    {
        if(port!=null)
            port.setRTS(b);
    }

    public void clear() throws Exception
    {
        while (is.available()!=0)
            is.read();
    }
}
```

Klasse **Fluke45** (rudimentär):

```
package la1ue03;

import javax.comm.*;

public class Fluke45 extends SerialInterface
{
    String data;

    /**
     * Creates a new instance of Fluke45
     */
    public Fluke45(String name) throws Exception
    {
        super(name);
        open();
        setSerialPortParams(9600,SerialPort.DATABITS_8,
                SerialPort.STOPBITS_1,SerialPort.PARITY_NONE);
     }


    public double getValue() throws Exception
    {
        write("VAL?\n");
        String s = readLine();
        String s2 = readLine();
        String s3 = readLine();
/*
        System.out.println(s);
        System.out.println(s2);
        System.out.println(s3);
*/
        double value = Double.parseDouble(s2);
        return value;
    }

    public static void main(String[] args)
    {
        Fluke45 mg1=null;
        try
        {
            mg1 = new Fluke45("COM7");
            mg1.clear();
            for(int i=0;i<10;i++)
            {
                double value = mg1.getValue();
                System.out.format("%d: %.03f%n",i+1,value);
                Thread.sleep(1000);
            }
        }
        catch (Exception e)
        {
            System.out.println(e.getMessage());
        }
        finally
        {
```

```
                if(mg1!=null)
                {
                    mg1.close();
                }
            }
        }
    }
```

Klasse **WG810**:

```
package la1ue03;

import javax.comm.*;
/** Fehlerliste<BR>
 *  10.11.2005: getOffset(): negative Offsets korrigiert
 *  26.10.2007: setKurvenform(char form, boolen visible) ergänzt
 *
 */
public class WG810 extends SerialInterface
{
    private byte paramSend[] ;
    private byte paramRead[];

    public static final char OFFSETSIGPOS = 'p';
    public static final char OFFSETSIGNEG = 'n';
    public static final char OFFSETSIGPOSNEG = 'x';

    public static final int MODNONE = 0;
    public static final int MODFM = 1;
    public static final int MODAM = 2;

    public static final char TRIGCONT = 'c';
    public static final char TRIGONESHOT = 'n';
    public static final char TRIGONESHOTANDTRIG = 'N';
    public static final char TRIGEXT = 'e';

    public static final char FORMSIN = 'S';
    public static final char FORMRECT = 'R';
    public static final char FORMTRI = 'T';
    public static final char FORMARBMORPH = 'D';
    public static final char FORMDC = 'G';
    public static final char FORMNOISE = 'N';

    public static final int OUTPUTOFF = 0;
    public static final int OUTPUTM20DB = 1;
    public static final int OUTPUT0DB = 2;

    public static final char CLKINTERN = 'i';
    public static final char CLKSINGLE = 's';
    public static final char CLKSINGLEANDCLOCK = 'S';
    public static final char CLKEXTERN = 'e';
    public static final char CLKSINGLEANDRESETANDCLOCK = 'z';

    public static final char MORPHMANUAL = 'm';
    public static final char MORPHEXTERN = 'e';
    public static final char MORPHTIME = 't';

    public static final int BITS8 = 8;
    public static final int BITS12 = 12;
    public static final int BITS16 = 16;

    /** Creates a new instance of WG810 */
    public WG810(String name) throws Exception
    {
        super(name);
```

```
        open();

setSerialPortParams(57600,SerialPort.DATABITS_8,SerialPort.STOPBITS_1,Seria
lPort.PARITY_NONE);
        paramSend = new byte[7];
        paramRead = new byte[7];
    }

    private byte getCRC(byte[] par)
    {
        return (byte)(par[1]^par[2]^par[3]^par[4]^par[5]);
    }

    private void printDebug(byte par[], boolean character)
    {
        for( int i=0; i<par.length; i++)
            if(character)
                System.out.print(((char)par[i])+" ");
            else
                System.out.print(((int)par[i])+" ");
        System.out.println();
    }

    public void setRemote(boolean remote) throws Exception
    {
        paramSend[0] = (byte)'X';
        paramSend[1] = (byte)'R';
        paramSend[2] = remote ? (byte)0 : (byte)1;
        paramSend[3] = (byte)0;
        paramSend[4] = (byte)0;
        paramSend[5] = (byte)0;
        paramSend[6] = getCRC(paramSend);

        write(paramSend);
        paramRead = read(7);
//        printDebug(paramSend,true);
//        printDebug(paramRead,true);
    }

    public boolean getRemote() throws Exception
    {
        paramSend[0] = (byte)'X';
        paramSend[1] = (byte)'r';
        paramSend[2] = (byte)0;
        paramSend[3] = (byte)0;
        paramSend[4] = (byte)0;
        paramSend[5] = (byte)0;
        paramSend[6] = getCRC(paramSend);

        write(paramSend);
        paramRead = read(7);

//        printDebug(paramSend,true);
//        printDebug(paramRead,true);
        return (paramRead[2]=='0') ? true : false;
    }
```

```
    public void setDefault() throws Exception
    {
        paramSend[0] = (byte)'X';
        paramSend[1] = (byte)'S';
        paramSend[2] = (byte)'D';
        paramSend[3] = (byte)'e';
        paramSend[4] = (byte)'f';
        paramSend[5] = (byte)0;
        paramSend[6] = getCRC(paramSend);

        write(paramSend);
        try
        {
            paramRead = read(7);
        }
        catch (Exception e)
        {
            ;
        }
    }

    public String getTyp() throws Exception
    {
        String str;
        paramSend[0] = (byte)'X';
        paramSend[1] = (byte)'&';
        paramSend[2] = (byte)0;
        paramSend[3] = (byte)0;
        paramSend[4] = (byte)0;
        paramSend[5] = (byte)0;
        paramSend[6] = getCRC(paramSend);

        write(paramSend);
        paramRead = read(7);
        int wert = paramRead[2]*256 + paramRead[3];
        str = "WG";
        str += wert;
        str += ", " + paramRead[4] + " Kanäle, ";
        str += paramRead[5] == 'm' ? "Morphing möglich" : "kein Morphing
möglich";
        return str;
    }

    public int getKanalAnzahl() throws Exception
    {
        String str;
        paramSend[0] = (byte)'X';
        paramSend[1] = (byte)'!';
        paramSend[2] = (byte)0;
        paramSend[3] = (byte)0;
        paramSend[4] = (byte)0;
        paramSend[5] = (byte)0;
        paramSend[6] = getCRC(paramSend);

        write(paramSend);
        paramRead = read(7);
```

```
        return paramRead[5];
    }

    public String getFirmware() throws Exception
    {
        String str="";
        paramSend[0] = (byte)'X';
        paramSend[1] = (byte)'e';
        paramSend[2] = (byte)0;
        paramSend[3] = (byte)0;
        paramSend[4] = (byte)0;
        paramSend[5] = (byte)0;
        paramSend[6] = getCRC(paramSend);

        write(paramSend);
        paramRead = read(7);

        str += (char)paramRead[2];
        str += (char)paramRead[3];
        str += (char)paramRead[4];
        str += (char)paramRead[5];

        return str;
    }

    public void setFrequenz(double freq, boolean visible) throws Exception
    {
        int exp = 3;
        int ifreq=0;
        while(freq >= 10000)
        {
            freq/=10;
            exp++;
        }

        while(freq<1000)
        {
            freq*=10;
            exp--;
        }
        ifreq = (int)freq;

        paramSend[0] = (byte)'X';
        paramSend[1] = (byte)'F';
        paramSend[2] = (byte)(ifreq/256);
        paramSend[3] = (byte)(ifreq%256);
        paramSend[4] = (byte)exp;
        paramSend[5] = (byte)(visible?-1:0);
        paramSend[6] = getCRC(paramSend);

        write(paramSend);
        paramRead = read(7);
    }

    public double getFrequenz() throws Exception
    {
        paramSend[0] = (byte)'X';
```

```
        paramSend[1] = (byte)'f';
        paramSend[2] = (byte)0;
        paramSend[3] = (byte)0;
        paramSend[4] = (byte)0;
        paramSend[5] = (byte)0;
        paramSend[6] = getCRC(paramSend);

        write(paramSend);
        paramRead = read(7);

        int hi = (int)paramRead[2];if(hi<0) hi+=256;
        int lo = (int)paramRead[3]; if(lo<0) lo+=256;
        int pow = (int)paramRead[4]; if(pow<0) pow +=256;

        double wert = hi*256 + lo;
        wert *= Math.pow(10, pow-3);
        return wert;
    }

    public void setAmplitude(double amplitude, boolean visible) throws
Exception
    {
        int iAmplitude = (int)(amplitude*1000);

        paramSend[0] = (byte)'X';
        paramSend[1] = (byte)'A';
        paramSend[2] = (byte)(iAmplitude/256);
        paramSend[3] = (byte)(iAmplitude%256);
        paramSend[4] = (byte)0;
        paramSend[5] = (byte)(visible?-1:0);
        paramSend[6] = getCRC(paramSend);

        write(paramSend);
        paramRead = read(7);
    }

    public double getAmplitude() throws Exception
    {
        paramSend[0] = (byte)'X';
        paramSend[1] = (byte)'a';
        paramSend[2] = (byte)0;
        paramSend[3] = (byte)0;
        paramSend[4] = (byte)0;
        paramSend[5] = (byte)0;
        paramSend[6] = getCRC(paramSend);

        write(paramSend);
        paramRead = read(7);

        int hi = (int)paramRead[2];if(hi<0) hi+=256;
        int lo = (int)paramRead[3]; if(lo<0) lo+=256;

        return ((double)hi*256 + lo)/1000;
    }

    public void setOffset(double offset, boolean visible) throws Exception
    {
```

```
        int iOffset = (int)(offset*1000);

        paramSend[0] = (byte)'X';
        paramSend[1] = (byte)'O';
        paramSend[2] = (byte)(iOffset/256);
        paramSend[3] = (byte)(iOffset%256);
        paramSend[4] = (byte)0;
        paramSend[5] = (byte)(visible?-1:0);
        paramSend[6] = getCRC(paramSend);

        write(paramSend);
        paramRead = read(7);
    }

    public double getOffset() throws Exception
    {
        paramSend[0] = (byte)'X';
        paramSend[1] = (byte)'o';
        paramSend[2] = (byte)0;
        paramSend[3] = (byte)0;
        paramSend[4] = (byte)0;
        paramSend[5] = (byte)0;
        paramSend[6] = getCRC(paramSend);

        write(paramSend);
        paramRead = read(7);

        int hi = (int)paramRead[2];
        int lo = (int)paramRead[3]; if(lo<0) lo+=256;

//        this.printDebug(paramRead, false);

        return ((double)hi*256 + lo)/1000;
    }

    public void setFilter(boolean filter, boolean visible) throws Exception
    {
        paramSend[0] = (byte)'X';
        paramSend[1] = (byte)'G';
        paramSend[2] = (byte)(filter?1:0);
        paramSend[3] = (byte)0;
        paramSend[4] = (byte)0;
        paramSend[5] = (byte)(visible?-1:0);
        paramSend[6] = getCRC(paramSend);

        write(paramSend);
        paramRead = read(7);
    }

    public boolean getFilter() throws Exception
    {
        paramSend[0] = (byte)'X';
        paramSend[1] = (byte)'g';
        paramSend[2] = (byte)0;
        paramSend[3] = (byte)0;
        paramSend[4] = (byte)0;
        paramSend[5] = (byte)0;
```

```
        paramSend[6] = getCRC(paramSend);

        write(paramSend);
        paramRead = read(7);

        return paramRead[2] == 1;
    }

    public void setOffsetMode(char mode, boolean visible) throws Exception
    {
        if(mode != this.OFFSETSIGPOS && mode != this.OFFSETSIGNEG && mode
!=this.OFFSETSIGPOSNEG)
            throw new Exception("falscher Offsetmode!!");
        paramSend[0] = (byte)'X';
        paramSend[1] = (byte)'W';
        paramSend[2] = (byte)0;
        paramSend[3] = (byte)0;
        paramSend[4] = (byte)0;
        byte bmode=(byte)mode;
        if(visible)
            bmode-=128;
        paramSend[5] = bmode;
        paramSend[6] = getCRC(paramSend);

        write(paramSend);
        paramRead = read(7);
    }

    public char getOffsetMode() throws Exception
    {
        paramSend[0] = (byte)'X';
        paramSend[1] = (byte)'w';
        paramSend[2] = (byte)0;
        paramSend[3] = (byte)0;
        paramSend[4] = (byte)0;
        paramSend[5] = (byte)0;
        paramSend[6] = getCRC(paramSend);

        write(paramSend);
        paramRead = read(7);

        return (char)paramRead[5];
    }

    public void setDutyCycle(int prozent, boolean visible) throws Exception
    {
        paramSend[0] = (byte)'X';
        paramSend[1] = (byte)'D';
        paramSend[2] = (byte)prozent;
        paramSend[3] = (byte)0;
        paramSend[4] = (byte)0;
        paramSend[5] = (byte)(visible?-1:0);
        paramSend[6] = getCRC(paramSend);

        write(paramSend);
        paramRead = read(7);
    }
```

```
    public int getDutyCycle() throws Exception
    {
        paramSend[0] = (byte)'X';
        paramSend[1] = (byte)'d';
        paramSend[2] = (byte)0;
        paramSend[3] = (byte)0;
        paramSend[4] = (byte)0;
        paramSend[5] = (byte)0;
        paramSend[6] = getCRC(paramSend);

        write(paramSend);
        paramRead = read(7);
        return (int)paramRead[2];
    }

    public void setSyncPosition(int position, boolean visible) throws
Exception
    {
        byte lo = (byte)position;
        position /= 256;
        byte mi = (byte)position;
        position /= 256;
        byte hi = (byte)position;
        paramSend[0] = (byte)'X';
        paramSend[1] = (byte)'Z';
        paramSend[2] = (byte)hi;
        paramSend[3] = (byte)mi;
        paramSend[4] = (byte)lo;
        paramSend[5] = (byte)(visible?-1:0);
        paramSend[6] = getCRC(paramSend);

        write(paramSend);
        paramRead = read(7);
        this.printDebug(paramSend,true);
    }

    public int getSyncPosition() throws Exception
    {
        paramSend[0] = (byte)'X';
        paramSend[1] = (byte)'z';
        paramSend[2] = (byte)0;
        paramSend[3] = (byte)0;
        paramSend[4] = (byte)0;
        paramSend[5] = (byte)0;
        paramSend[6] = getCRC(paramSend);

        write(paramSend);
        paramRead = read(7);

        int hi = (int)paramRead[2]; if(hi<0) hi+=256;
        int mi = (int)paramRead[3]; if(mi<0) mi+=256;
        int lo = (int)paramRead[4]; if(lo<0) lo+=256;

        return (hi*256+mi)*256 + lo;
    }
```

```
    public void setModulation(int modulation, boolean visible) throws
Exception
    {
        if(modulation != this.MODNONE && modulation != this.MODAM &&
modulation !=this.MODFM)
            throw new Exception("falsche Modulation!!");
        paramSend[0] = (byte)'X';
        paramSend[1] = (byte)'M';
        paramSend[2] = (byte)modulation;
        paramSend[3] = (byte)0;
        paramSend[4] = (byte)0;
        paramSend[5] = (byte)(visible?-1:0);
        paramSend[6] = getCRC(paramSend);

        write(paramSend);
        paramRead = read(7);
    }

    public int getModulation() throws Exception
    {
        paramSend[0] = (byte)'X';
        paramSend[1] = (byte)'m';
        paramSend[2] = (byte)0;
        paramSend[3] = (byte)0;
        paramSend[4] = (byte)0;
        paramSend[5] = (byte)0;
        paramSend[6] = getCRC(paramSend);

        write(paramSend);
        paramRead = read(7);

        return (int)paramRead[2];
    }

    public void setTriggerModus(char trigger, boolean visible) throws
Exception
    {
        if(trigger != this.TRIGCONT && trigger != this.TRIGONESHOT &&
trigger != this.TRIGONESHOTANDTRIG && trigger != this.TRIGEXT)
            throw new Exception("falscher Triggermode!!");
        paramSend[0] = (byte)'X';
        paramSend[1] = (byte)'T';
        paramSend[2] = (byte)trigger;
        paramSend[3] = (byte)0;
        paramSend[4] = (byte)0;
        paramSend[5] = (byte)(visible?-1:0);
        paramSend[6] = getCRC(paramSend);

        write(paramSend);
        paramRead = read(7);
    }

    public char getTriggerModus() throws Exception
    {
        paramSend[0] = (byte)'X';
        paramSend[1] = (byte)'t';
        paramSend[2] = (byte)0;
```

```
        paramSend[3] = (byte)0;
        paramSend[4] = (byte)0;
        paramSend[5] = (byte)0;
        paramSend[6] = getCRC(paramSend);

        write(paramSend);
        paramRead = read(7);

        return (char)paramRead[2];
    }

    public void setKurvenform(char form, boolean visible) throws Exception
    {
        setKurvenform(form,0,visible);
    }

    public void setKurvenform(char form, int nr, boolean visible) throws
Exception
    {
        if(form != this.FORMSIN && form != this.FORMRECT && form !=
this.FORMTRI &&
           form != this.FORMARBMORPH && form != this.FORMDC && form !=
this.FORMNOISE)
            throw new Exception("falsche Kurvenform");

        if(form == this.FORMARBMORPH)
            if(nr<1 || nr>7)
                throw new Exception("falsche Kurvennummer");

        paramSend[0] = (byte)'X';
        paramSend[1] = (byte)'K';
        paramSend[2] = (byte)form;
        paramSend[3] = (byte)(form==this.FORMARBMORPH ? nr : 0);
        paramSend[4] = (byte)0;
        paramSend[5] = (byte)(visible?-1:0);
        paramSend[6] = getCRC(paramSend);

        write(paramSend);
        try
        {
            paramRead = read(7);
        }
        catch(Exception e)
        {
            ;
        }
    }

    public char getKurvenform() throws Exception
    {
        paramSend[0] = (byte)'X';
        paramSend[1] = (byte)'k';
        paramSend[2] = (byte)0;
        paramSend[3] = (byte)0;
        paramSend[4] = (byte)0;
        paramSend[5] = (byte)0;
        paramSend[6] = getCRC(paramSend);
```

```
        write(paramSend);
        paramRead = read(7);

        char ret = (char)paramRead[2];

        if(ret==this.FORMARBMORPH)
            ret=(char)(paramRead[3]+48);

        return ret;
    }

    public void setAusgangsModus(int modus, boolean visible) throws
Exception
    {
        if(modus != this.OUTPUTOFF && modus != this.OUTPUTM20DB && modus
!=this.OUTPUT0DB)
            throw new Exception("falscher Ausgangsmodus!!");
        paramSend[0] = (byte)'X';
        paramSend[1] = (byte)'Q';
        paramSend[2] = (byte)modus;
        paramSend[3] = (byte)0;
        paramSend[4] = (byte)0;
        paramSend[5] = (byte)(visible?-1:0);
        paramSend[6] = getCRC(paramSend);

        write(paramSend);
        paramRead = read(7);
    }

    public int getAusgangsModus() throws Exception
    {
        paramSend[0] = (byte)'X';
        paramSend[1] = (byte)'q';
        paramSend[2] = (byte)0;
        paramSend[3] = (byte)0;
        paramSend[4] = (byte)0;
        paramSend[5] = (byte)0;
        paramSend[6] = getCRC(paramSend);

        write(paramSend);
        paramRead = read(7);

        return (int)(paramRead[2]-48);
    }

    public void setClockSource(char source, boolean visible) throws
Exception
    {
        if(source !=  this.CLKINTERN && source != this.CLKSINGLE && source
!=this.CLKSINGLEANDCLOCK &&
           source      !=      this.CLKEXTERN      &&      source      !=
this.CLKSINGLEANDRESETANDCLOCK)
            throw new Exception("falsche Taktquelle!!");
        paramSend[0] = (byte)'X';
        paramSend[1] = (byte)'P';
        paramSend[2] = (byte)0;
```

```
        paramSend[3] = (byte)0;
        paramSend[4] = (byte)0;
        byte bsource = (byte)source;
        if(visible)
            bsource -= 128;
        paramSend[5] = (byte)bsource;
        paramSend[6] = getCRC(paramSend);

        write(paramSend);
        paramRead = read(7);
    }

    public char getClockSource() throws Exception
    {
        paramSend[0] = (byte)'X';
        paramSend[1] = (byte)'p';
        paramSend[2] = (byte)0;
        paramSend[3] = (byte)0;
        paramSend[4] = (byte)0;
        paramSend[5] = (byte)0;
        paramSend[6] = getCRC(paramSend);

        write(paramSend);
        paramRead = read(7);

        return (char)(paramRead[5]);
    }

    public void setMorphingParameter(char par, int nr, boolean visible)
throws Exception
    {
        if(par != this.MORPHMANUAL && par != this.MORPHEXTERN && par
!=this.MORPHTIME)
            throw new Exception("falscher Morphing Parameter!!");
        if(par == this.MORPHEXTERN)
            nr=0;
        paramSend[0] = (byte)'X';
        paramSend[1] = (byte)'[';
        paramSend[2] = (byte)0;
        paramSend[3] = (byte)0;
        paramSend[4] = (byte)nr;
        byte bpar = (byte)par;
        if(visible)
            bpar -= 128;
        paramSend[5] = (byte)bpar;
        paramSend[6] = getCRC(paramSend);

        write(paramSend);
        paramRead = read(7);

        this.printDebug(paramSend,true);
        this.printDebug(paramRead,true);
    }

    public String getMorphingParameter() throws Exception
    {
        String str = "";
```

```
        paramSend[0] = (byte)'X';
        paramSend[1] = (byte)']';
        paramSend[2] = (byte)0;
        paramSend[3] = (byte)0;
        paramSend[4] = (byte)0;
        paramSend[5] = (byte)0;
        paramSend[6] = getCRC(paramSend);

        write(paramSend);
        paramRead = read(7);

        this.printDebug(paramSend,true);
        this.printDebug(paramRead,true);

        int par3 = (int)paramRead[4];
        if(par3 < 0)
            par3+=256;

        if(paramRead[5] == this.MORPHEXTERN)
            str = "e";

        if(paramRead[5] == this.MORPHMANUAL)
            str = "m" + par3;

        if(paramRead[5] == this.MORPHTIME)
            str = "t" + par3;

        return str;
    }

    public void startProgrammierung(int position, int nr) throws Exception
    {
        byte lo = (byte)position;
        position /= 256;
        byte mi = (byte)position;
        position /= 256;
        byte hi = (byte)position;
        paramSend[0] = (byte)'X';
        paramSend[1] = (byte)'B';
        paramSend[2] = (byte)hi;
        paramSend[3] = (byte)mi;
        paramSend[4] = (byte)lo;
        paramSend[5] = (byte)nr;
        paramSend[6] = getCRC(paramSend);

        write(paramSend);
        paramRead = read(7);
//        this.printDebug(paramSend,true);
    }

    public boolean getStartFertig(int position, int nr) throws Exception
    {
        byte lo = (byte)position;
        position /= 256;
        byte mi = (byte)position;
        position /= 256;
        byte hi = (byte)position;
```

```
        paramSend[0] = (byte)'X';
        paramSend[1] = (byte)'b';
        paramSend[2] = (byte)hi;
        paramSend[3] = (byte)mi;
        paramSend[4] = (byte)lo;
        paramSend[5] = (byte)nr;
        paramSend[6] = getCRC(paramSend);

        write(paramSend);
        paramRead = read(7);
//        this.printDebug(paramSend,true);

        return paramRead[5] == 1;
    }

    public void stopProgrammierung(int position, int nr) throws Exception
    {
        byte lo = (byte)position;
        position /= 256;
        byte mi = (byte)position;
        position /= 256;
        byte hi = (byte)position;
        paramSend[0] = (byte)'X';
        paramSend[1] = (byte)'U';
        paramSend[2] = (byte)hi;
        paramSend[3] = (byte)mi;
        paramSend[4] = (byte)lo;
        paramSend[5] = (byte)nr;
        paramSend[6] = getCRC(paramSend);

        write(paramSend);
        paramRead = read(7);
//        this.printDebug(paramSend,true);
    }

    public boolean getStopFertig(int position, int nr) throws Exception
    {
        byte lo = (byte)position;
        position /= 256;
        byte mi = (byte)position;
        position /= 256;
        byte hi = (byte)position;
        paramSend[0] = (byte)'X';
        paramSend[1] = (byte)'u';
        paramSend[2] = (byte)hi;
        paramSend[3] = (byte)mi;
        paramSend[4] = (byte)lo;
        paramSend[5] = (byte)nr;
        paramSend[6] = getCRC(paramSend);

        write(paramSend);
        paramRead = read(7);
//        this.printDebug(paramSend,true);

        return paramRead[5] == 1;
    }
```

```
    public void setMorphModus(int nr) throws Exception
    {
        paramSend[0] = (byte)'X';
        paramSend[1] = (byte)'*';
        paramSend[2] = (byte)0;
        paramSend[3] = (byte)0;
        paramSend[4] = (byte)0;
        paramSend[5] = (byte)nr;
        paramSend[6] = getCRC(paramSend);

        write(paramSend);
        paramRead = read(7);
//        this.printDebug(paramSend,true);
    }

/** Diese Funktion wurde aus zusätzlicher Dokumentation ennommen
 */
    public void setProgrammierModus() throws Exception
    {
        paramSend[0] = (byte)'X';
        paramSend[1] = (byte)'C';
        paramSend[2] = (byte)0;
        paramSend[3] = (byte)0;
        paramSend[4] = (byte)0;
        paramSend[5] = (byte)0;
        paramSend[6] = getCRC(paramSend);

        write(paramSend);
        paramRead = read(7);
//        this.printDebug(paramSend,true);
//        this.printDebug(paramRead,true);
    }

    /** Achtung: Diese Methode funtioniert noch nicht!!! <BR>
     *           Bei Aufruf wird die Sinus Kurvenform zerstört bis zum Aus-
und Einschalten!!!
     */
    public void progKurve(int []data, int nr, int bits) throws Exception
    {
            boolean ok=false;
            setProgrammierModus();
            try
            {
                startProgrammierung(3,1);
            }
            catch (Exception e)
            {
                ;
            }
            do
            {
                ok=getStartFertig(3, 2*data.length);
            }
            while(!ok);

            sendKurve(data, bits);
            Thread.sleep(500);
```

```
        try
        {
            stopProgrammierung(3, 1);
        }
        catch(Exception e)
        {
            ;
        }

        do
        {
            getStopFertig(3, 2*data.length);
        }
        while(!ok);
    }

    public void sendKurve(int f[], int bit) throws Exception
    {
        for(int i=0; i<f.length; i++)
        {
            if(bit==8)
                f[i] <<= 8;
            if(bit==12)
                f[i] <<= 4;
        }

        byte bf[] = new byte[2*f.length];

        for(int i=0; i<f.length; i++)
        {
            bf[2*i] = (byte)f[i];
            bf[2*i+1] = (byte)(f[i]>>=8);
        }
        write(bf);
    }

    /**
     * @paramSend args the command line arguments
     */
    public static void main(String[] args)
    {
        try
        {
            WG810 wg810 = new WG810("COM6");
            wg810.setRemote(true);
            System.out.println("Remote: "+wg810.getRemote());
            wg810.setDefault();
            System.out.println("Typ: "+wg810.getTyp());
            System.out.println("Kanäle: "+wg810.getKanalAnzahl());
            System.out.println("Firmware: "+wg810.getFirmware());
            System.out.println("Remote: "+wg810.getRemote());
            Thread.sleep(2000);
            wg810.setKurvenform(wg810.FORMTRI, 0, true);
            System.out.println("Kurvenform: " + wg810.getKurvenform());
            wg810.setFrequenz(20000,true);
            System.out.println("Frequenz: " + wg810.getFrequenz());
            wg810.setAmplitude(2.0,true);
```

```
            System.out.println("Amplitude: " + wg810.getAmplitude());
            wg810.setOffsetMode(wg810.OFFSETSIGPOSNEG, true);
            System.out.println("Offset: " + wg810.getOffsetMode());

            wg810.setOffset(0.0,true);
            System.out.println("Offset: " + wg810.getOffset());

//            wg810.setDutyCycle(60,true);
            System.out.println("DutyCycle: " + wg810.getDutyCycle());

//            wg810.setAusgangsModus(wg810.OUTPUT0DB, true);
            System.out.println("Ausgangsmodus:              "              +
wg810.getAusgangsModus());

//            wg810.setFilter(true,true);
            System.out.println("Filter: " + wg810.getFilter());

//            wg810.setSyncPosition(56,true);
            System.out.println("SyncPosition: " + wg810.getSyncPosition());

//            wg810.setModulation(wg810.MODNONE, false);
            System.out.println("Modulation: " + wg810.getModulation());

//            wg810.setTriggerModus(wg810.TRIGCONT, false);
            System.out.println("Triggermode: " + wg810.getTriggerModus());

//            wg810.setClockSource(wg810.CLKINTERN,true);
            System.out.println("Taktquelle: " + wg810.getClockSource());

//            wg810.setMorphingParameter(wg810.MORPHMANUAL, 3, true);
//                          System.out.println("Morphing  Parameter:  "  +
wg810.getMorphingParameter());


            // Ein erster Versuch Kurvendaten in den Funktionsgenerator zu
schreiben
//            int daten[] = {0,0,255,255,255,255,128,128};
//            wg810.progKurve(daten, 1, wg810.BITS8);

            wg810.setRemote(false);
            System.out.println("Remote: "+wg810.getRemote());
            wg810.close();
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }
}
```