

MultiPro programmer technical notes

This document describes how to switch the mode of the MultiPro by sending instructions to the programmer.

Instructions are transmitted with this protocol:

Port initialisation: RTS is high and DTR is high

Full instruction:<10*Chr(21)><instruction><Qs instruction>

Answer from MultiPro:<answer><Qs Instruction+answer>

(Qs is checksum, the programmer starts calculating the answer checksum by taking the Qs of the instruction as begin value)

This is the Delphi source for calculating Checksum Qs and transmitting the instruction:

```
Function Checksum(OldValue:Byte;NewValue:Byte):Byte;
```

```
Begin
```

```
    Result:=OldValue XOR NewValue;
```

```
    If (Result AND 128)=128 then Result:=Result SHL 1
```

```
        else Result:=(Result SHL 1) + 1;
```

```
End;
```

```
procedure SendInstruction(Command,SubCommand:Byte);
```

```
var Str:string;
```

```
    I:integer;
```

```
    Ch,Chksm:Char;
```

```
begin
```

```
    Str:="";
```

```
//Start with 10 times Chr(21)
```

```
    for I:=1 to 10 do Str:=Str+Chr(21);
```

```
//Select the Command
```

```
    Str:=Str+Chr(Command);
```

```
//Select the Sub-Command
```

```
    Str:=Str + Chr(SubCommand);
```

```
//Begin checksum is 0
```

```
    Chksm:=#$00;
```

```
//Calculate the checksum of the instruction string
```

```
    For I:=1 to Length(Str) do
```

```
        Chksm:=Chr(CheckSum(Ord(Chksm),Ord(Str[I])));
```

```
    Str:=Str+Chksm;
```

```
//Open the comport and set the RTS and DTR line in high state
```

```
    Form1.Comport1.Open;
```

```
    Form1.Comport1.SetRTS(True);
```

```
    Form1.ComPort1.SetDTR(True);
```

```
//Send the instruction string
```

```
    For I:=1 to Length(Str) do
```

```
        begin
```

```
            Ch:=Str[I];
```

```
            delay(5);
```

```
            Form1.Comport1.Write(Ch,1,True);
```

```
        end;
```

```
//close the com-port  
Form1.Comport1.Close;  
end;
```

An instruction is formed by <command><subcommand>

Command 00 High Level instructions

Subcommand:

- 00 Idle mode, Programmer in idle state --> No power on smartcard
- 01 Phoenix 3.57MHz
- 02 Phoenix 6.00MHz
- 03 Smartmouse 3.57MHz
- 04 Smartmouse 6.00MHz
- 05 PonyProg 3.57MHz (Default)
- 06 PonyProg 6.00MHz
- 07 JDM emulation

Example.

If you want to set the programmer in Smartmouse 6.00Mhz, you can use this program-line:

‘SendInstruction(00,04);’

The program will force RTS and DTR in high state and then send these characters: ‘15 15 15 15 15 15 15 15 00 04 FE’

The programmer answers only with Qs.

Note: Please use only PonyProg 3.57MHz to program Atmel microcontrollers. Some Atmel controllers can only handle frequencies up to 4MHz.