Report No:

AN109

Title:

Remote Application Control of Equinox ISP Programmers using the ISP-PRO application

Author:	Date:	Version Number:
John Marriott	19th June 2012	1.04

Contents

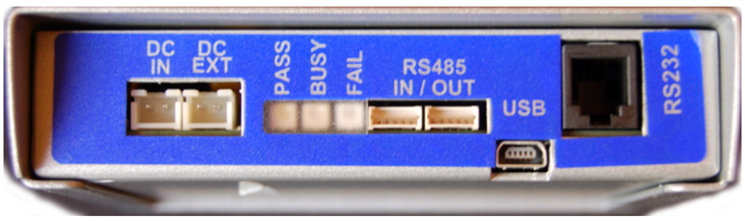
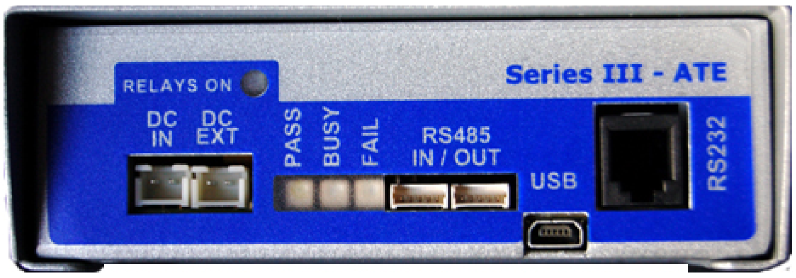
1.0 Overview

This application note describes how to control any Equinox ISP Programmer from a custom ***'Remote Application'*** written in any control language e.g. Visual Basic, C++, Labview, Teststand etc.

The ISP-PRO software interacts with a so-called ***'Interface Database'*** to provide feedback and control for each programmer connected to the programming network. This architecture allows target system related data such as programming statistics, unique data etc. to be logged for every programming operation. It also allows the Equinox programmers to be controlled via a ***'Remote Application'*** such as Labview or Visual Basic (requires chargeable license upgrades).

1.1 Supported Programmers

The Equinox ISP Programmers detailed in the table below are compatible with ***'Remote Application Control'*** via ISP-PRO. Each programmer requires a 'License Upgrade' to enable compilation / execution of Programming Scripts and also to be controlled from a Remote Application.

Programmer picture	**Programmer name**
	ISP nano Series III Single channel ISP programmer
	ISP nano Series III - ATE Single channel ISP programmer with relay isolation
	ISP nano mux8 8 channel Multiplexed programming system

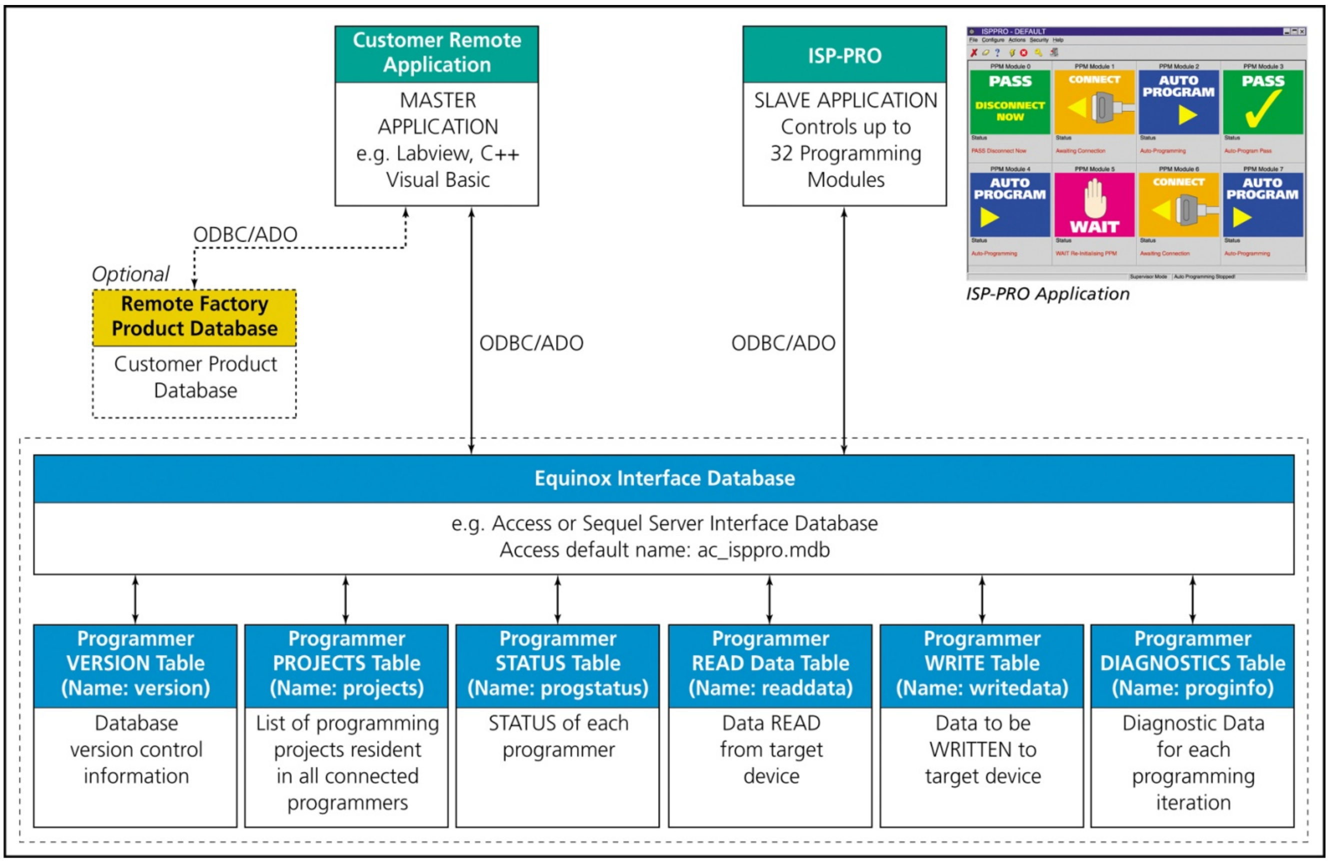
	ISP nano mux4 4 channel Multiplexed programming system
	ISP nano mux2 2 channel Multiplexed programming system
	ISP nano Series IV-ATE Single channel ISP programmer With plug-in connector module, relay signal isolation and opto-isolated TTL control port
	ISP nano gang Multi-Channel Gang Production ISP Programming Systems **ISPnano** **Series IV** **Gang Systems** **4 / 6 / 8 channel systems** Multi-channel parallel programming systems for programming multiple DUTs concurrently on a **‘PCB Panel’**

	PPM4 MK1 High Speed Production ISP Programming Module

1.2 Programmer Control Methodology

It is possible to control from 1 up to 32 programmers from a single PC using the Equinox ISP-PRO software application. Equinox have chosen an ***'Interface Database'*** as the method to control the programmers. This database is simply a collection of tables in a database which are shared between the Equinox ISP-PRO application and the Remote Application – see fig. 1.3. The ***'Remote Application'*** is the so-called ***'Master Application'*** and this application controls the sequencing of a ***'Programming Script'*** running within the ISP-PRO application (Slave Application).

Fig. 1.3 – Interface Database Architecture


Customer Remote Application
MASTER APPLICATION e.g. Labview, C++ Visual Basic
ISP-PRO
SLAVE APPLICATION Controls up to 32 Programming Modules
PASS
DISCONNECT NOW
CONNECT
AUTO PROGRAM
WAIT
ISP-PRO Application
ODBC/ADO
Optional
Remote Factory Product Database
Customer Product Database
ODBC/ADO
ODBC/ADO
Equinox Interface Database
e.g. Access or Sequel Server Interface Database
Access default name: ac_isppro.mdb
Programmer VERSION Table (Name: version)
Database version control information
Programmer PROJECTS Table (Name: projects)
List of programming projects resident in all connected programmers
Programmer STATUS Table (Name: progstatus)
STATUS of each programmer
Programmer READ Data Table (Name: readdata)
Data READ from target device
Programmer WRITE Table (Name: writedata)
Data to be WRITTEN to target device
Programmer DIAGNOSTICS Table (Name: proginfo)
Diagnostic Data for each programming iteration

Both the ***'Remote Application'*** and ISP-PRO can read / write to the fields in the tables in the database.

1.3 Why use an Interface Database ?

The Interface Database supports the following functionality:

- Keeps track of up to 32 programmers at the same time by creating a new ***'Database Record'*** for each attached programmer.
- Allows any ***'Remote Application'*** which supports reading / writing to a database to control the programmers.
- Supports logging of full diagnostics for every programming iteration to a table in the database.
- The diagnostics are time / date / ID stamped and so can be traced back to individual Target Systems if required.
- Supports writing / reading of unique data such as calibration data, serial numbers etc to / from the Target Device. This unique data can be passed via fields in the database and is therefore recorded (logged) for every programming iteration. This allows a full log to be created of every Target Board programmed listing the actual data which was read / written to this Target System.
- The ***'Interface Database'*** is independent of Windows version. As long as either an ADO or ODBC connection can be established to the database, then this solution will work on any PC.

1.4 What is a Programming Script ?

A ***'Programming Script'*** is a list of low-level commands which ISP-PRO executes in sequence in order to instruct the programmer to program a Target Device. These scripts are created using a special utility within EQTools called ***'Script Builder'***. The script runs from beginning to end and then either terminates or loops back to the beginning again.

A simple example of a ***'Programming Script'*** would be the case where there are a number of ***'Standalone Programming Projects'*** have already been uploaded into a programmer and a ***'Remote Application (RA)'*** is required to execute one of these projects. The RA therefore passes the name of the ***'Standalone Programming Project'*** to be executed via a field in the database and the script running within ISP-PRO then reads the project name and instructs the programmer to execute it.

The Programming Script actions would be as follows:

- Wait for Remote Application (RA) to tell the script to execute
- Acknowledge that the script has started by changing the state of the ***progstatus.Status*** field in the database.
- Read the name of the ***'Standalone Programming Project'*** to be executed from the ***progstatus.AutoProgram1*** field in the database
- Execute the ***'Standalone Programming Project'***
- Let the RA know that the script has finished by writing '***PASS – DISCONNECT'*** or ***'FAIL – DISCONNECT'*** into the ***progstatus.Status*** field.
- End

The overall sequence of events would be as follows:

- Remote Application (RA) passes the name of the ***'Standalone Programming Project'*** to be executed by writing the name into the ***progstatus.AutoProgram1*** field in the database.
- RA instructs the script to start by writing eg. ***'RA_START'*** into the ***progstatus.Status*** field.
- Script starts execution within ISP-PRO
- The script executes the ***'AutoProgram1'*** project specified in the ***progstatus.AutoProgram1*** field in the database.
- The RA now polls the ***progstatus.Status*** field waiting for the ***AutoProgram1*** operation to finish.
- On completion, ISP-PRO will set the ***progstatus.Status*** field to either ***'PASS – DISCONNECT'*** or ***'FAIL – DISCONNECT'***.
- The RA must then process the PASS / FAIL result.
- ISP-PRO then waits for the next new Database Record to be inserted and the process is then repeated.

2.0 Interfacing with ISP-PRO

All communication with ISP-PRO is managed using the "ISP-PRO Interface Database". This method has been chosen to be totally independent from network issues or Windows API functions. You can initiate a program process from Access directly or take any programming language with ability to read / write to a database via eg. ADO, SQL or ODBC.

2.1 Database Connection (ADO)

During the installation of both EQTools and ISP-PRO, an empty Interface Database will be copied to your system into the ***\program files\equinox\db*** directory. This is simply a collection of data tables in an Access 2000 database. An ***'ADO database link'*** is automatically created and maintained by both applications allowing them to read / write to this database via this ADO connection.

You can now access this database using the ADO link ***"ac_isppro".***

ISP-PRO is by default using this ADO link to the database.

You can change this behaviour so that ISP-PRO uses your own custom Access or Sequel Server database if required. This can be achieved by following the instructions below.

- start ISP-PRO
- login
- Click ***<Setup>***
- Select ***<Database Options>*** tab
- select the required database connection method
- set username and password if necessary

Please also refer to the ISP-PRO Manual. If you select a different database make sure the structure is compatible to the ISP-PRO Interface Database.

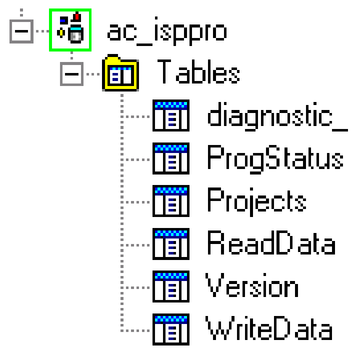
2.2 Database Connection (ODBC)

If your Remote Application does not support ***'ADO'*** then it is necessary to set up and use an ODBC database connection instead. This is typical of National Instruments – Labview which requires special ***'SQL Drivers'*** in order to communicate with a database.

Please see the ***'ISP-PRO – Creating an ODBC Connection'*** instructions for further details.

2.3 Database Table Structure

The ISP-PRO Interface Database consists of at least the following six tables.

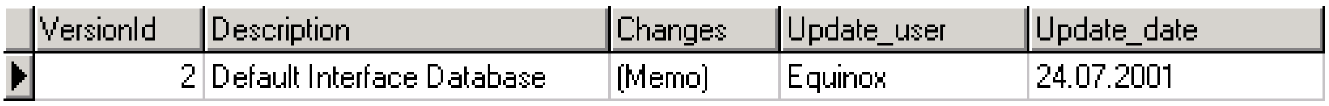


- ac_isppro
  - Tables
    - diagnostic_
    - ProgStatus
    - Projects
    - ReadData
    - Version
    - WriteData

Please note:
The **‘Diagnostic_’** table has been renamed in EQTools Version 2 to **‘ProgInfo’**.

2.3.1 Version Table - Database version tracking

This table stores historical information about the version of this database. Every time the database is updated for any reason, a new database ‘VersionID’ is created and a list of all the changes is recorded against this ID.



VersionId	Description	Changes	Update_user	Update_date
2	Default Interface Database	(Memo)	Equinox	24.07.2001

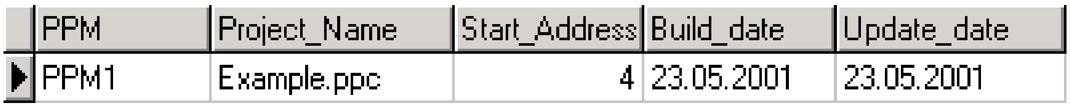
Any ‘Programming Script’ generated using EQTools – Script Builder can check for a specific database version. This is not activated per default. To activate it follow these steps:

- start EQTools
- open a script in the Script Wizard
- select Databases
- activate Validate Database Version Details
- pressing <Update> stores the current database version into the script

The script will now check for the specified database version and will refuse execution if the database version has changed.

2.3.2 Projects Table

This table contains information about any programmers which are currently connected to the PC and details the ‘Programmer Name’ and then lists every ‘Programming Project‘ which is currently resident in each programmer. This information is permanently updated by ISP-PRO.



PPM	Project_Name	Start_Address	Build_date	Update_date
PPM1	Example.ppc	4	23.05.2001	23.05.2001

Every ‘Programming Project’ found in an individual programmer is listed as a separate entry in the table.

The field names are as follows:

- **PPM**
This field contains the name of the programmer (Production Programming Module – PPM) where the ‘Programming Project’ is stored.

- **Project_Name**
This field contains the name of a ‘Programming Project’. If version control is enabled, it will also contain a version number and a date.

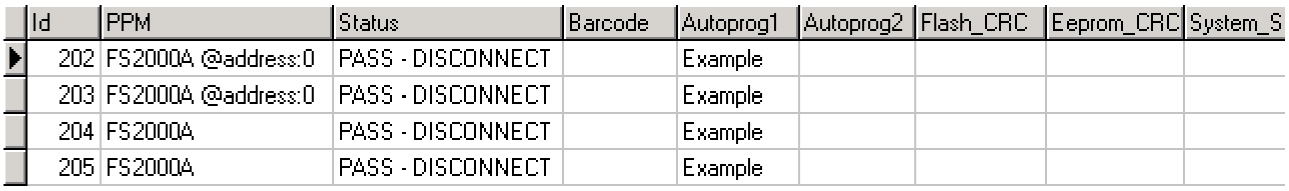
- **Start_Address**
This field gives information about the internal address of this project in the programmer. Please refer to the ISP-PRO User Guide for detailed information.

- **Build_date**
This field contains information about the compilation date of this project. Please refer to the ISP-PRO User Guide for detailed information.

- **Update_date**
This field contains information about the update date of this project. Please refer to the ISP-PRO User Guide for detailed information.

2.3.3 Programmer Status Table – progstatus

This table is used to display the current 'Status' of each programmer on the network. It can also be used by the Remote Application to initiate the exectution of a 'Programming Script' or to advance a 'Programming Script' to the next stage in the script.



Id	PPM	Status	Barcode	Autoprog1	Autoprog2	Flash_CRC	Eeprom_CRC	System_S
202	FS2000A @address:0	PASS - DISCONNECT		Example				
203	FS2000A @address:0	PASS - DISCONNECT		Example				
204	FS2000A	PASS - DISCONNECT		Example				
205	FS2000A	PASS - DISCONNECT		Example				

This table gives information about the current state of each program process - whether it has failed with a certain error code or passed. The ISP-PRO application constantly polls this table and can initiate programming processes as soon as a new entry appears. It will then use this entry in the database to update status information to the Remote Application

The fields in the ***ProgStatus*** table are as follows:

- **PPM**
  This field defines which programmer is being controlled. ISP-PRO maintains a list of all attached programmers.

- **Status**
  This field contains the current status of the stated programming channel. At the beginning of a script, ISP-PRO will poll this field waiting for the Remote Application (RA) to change the status from eg. 'WAITING' to eg. 'RA_START'. During the actual execution of the script, ISP-PRO will constantly update the Status field with the current programming

  Writing ***'ABORT'*** into this field causes ISP-PRO to halt the current script. ISP-PRO will write ***'FAIL or PASS – DISCONNECT'*** into this field to indicate a completed program process.

- **Script Name**
  This field allows the name the Programming Script to be executed to be passed from the RA to ISP-PRO. To use this functionality, it is necessary to load the 'WaitForScript.esf' script in ISP-PRO.

- **AutoProgram1**
  This field is used to pass the project name of the AutoProgram1 project to ISP-PRO. A list of all available projects is stored in the Projects table.

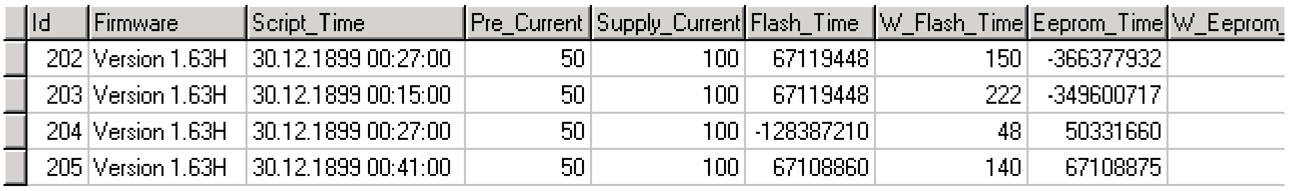
- **AutoProgram2**
  Use this field to pass the project name of AutoProgram2 to ISP-PRO.

- **System Message**
  ISP-PRO will save information about the current state of the programmer into this field. You should display the contents of this field to the end user.

- **Error Message**
  If an error occurred ISP-PRO will store a short error message in this field. More detailed information about the error will be stored in the table diagnostics.

2.3.4 Diagnostics Table – ProgInfo

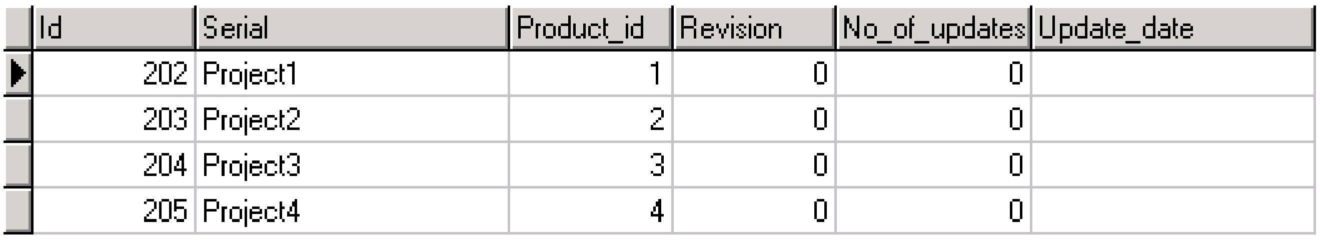
This table contains more detailed information about exactly what happened during every programming process. You will find the programmer's firmware, the serial number, target voltages, programming time and the calibration byte of the target device here, as long as the programmer was able to provide this data. The diagnostics table will allow you to locate programming problems.



Id	Firmware	Script_Time	Pre_Current	Supply_Current	Flash_Time	W_Flash_Time	Eeprom_Time	W_Eeprom_
202	Version 1.63H	30.12.1899 00:27:00	50	100	67119448	150	-366377932	
203	Version 1.63H	30.12.1899 00:15:00	50	100	67119448	222	-349600717	
204	Version 1.63H	30.12.1899 00:27:00	50	100	-128387210	48	50331660	
205	Version 1.63H	30.12.1899 00:41:00	50	100	67108860	140	67108875	

2.3.5 Write and Read Data Tables - EQWriteData and EQReadData

These tables are used when custom data is to be written to or read from the target device. The Equinox 'Visual Basic' and 'Labview' demo programs both use these tables to transfer data.



Id	Serial	Product_id	Revision	No_of_updates	Update_date
202	Project1	1	0	0	
203	Project2	2	0	0	
204	Project3	3	0	0	
205	Project4	4	0	0	

If you intend to create your own ***'Remote Application'*** or just improve our demo application, please consider creating other tables like ***WriteData*** and ***ReadData*** instead. You will have to provide as much columns as there is data to be written to the target device. In this small example we intend to write a serial number, a product ID and information about the revision and updates into the target device.

3.0 Remote Application

This section details how to set up a Remote Application to control Equinox programmers via the ISP-PRO application. You will not find programming language specific terms except for some simple SQL commands. You can start from this introduction and use any programming language you like as long as it supports SQL and ODBC or another way to create and edit entries in the ISP-PRO database. You can even control the programmer just with Access or the Database Explorer 'ADOExplorer' which is installed by EQTools and ISP-PRO.

3.1 Connecting to the database

Since the database is the only way to communicate with ISP-PRO, you first have to connect to this database. EQTools Version4 and ISP-PRO Version 4 now install and use an ***'ADO Database Connection'*** by default. If your Remote Application does not support 'ADO', then an 'ODBC Connection' should be established instead. Please see the 'ISP-PRO – Getting Started Guide' for instructions on setting up an ODBC Database Connection.

3.2 Obtaining a list of programmers and projects

Before starting a programming process you have to know which programmers are connected to ISP-PRO. You have already learned that ISP-PRO maintains a list of the programmers and their projects in the database. All information you need is stored there. You will need a SQL command like this to retrieve the data:

```
select PPM, Project_Name from Projects
```

You should give the user a possibility to select the programmer and project or you will have to hardcode it in your Remote Application. Please be aware that the projects expect certain target devices. The script can be set to check for the target signature to make sure the correct device is attached to the programmer.

- start EQTools
- open the script in Script Wizard
- in Script Tasks activate Check Signature
- in Check Signature make sure that Check Signature is activated

3.3 Starting a programming process

In order to start a programming process, all you have to do is to create a New Entry in the table progstatus and set the Status to be e.g. 'WAITING' This signals to ISP-PRO that the script is to start execution. Unfortunately it is not that easy. You still have to define on which programmer this programming process should be executed, and which project and script you wish to use. The script name must be stored in Script Name, the projects in AutoProgram1 or AutoProgram2. In SQL this can be achieved as follows:

```
insert into progstatus (Script_Name, AutoProgram1, AutoProgram2, Status)
values (’ascript’, ’project1’, ’project2’, ’WAITING’);
```

This entry will get an ID using an autonumber. Using this ID you can find this entry again and check for the status or abort the programming process. Since the autonumber is simply counting up, the easiest way to get the entry you just created is to find the highest ID in the table.

```
select max(id) from progstatus;
```

3.3.1 Selecting the script to execute

If the Remote Application is to pass the name of the Programming Script to be executed to ISP-PRO, then ISP-PRO must be running the 'WaitForScript'esf' script.

- Start ISP-PRO
- Login
- Click ***<Setup>***
- In ***<PPM Settings>***, select the required ***'Programminng Channel'*** and then set this programming channel to use the script file called ***'WaitForScript.esf'***.

A copy of the ***'WaitForScript.esf'*** script file is located in the ***\Scripts*** directory of ISP-PRO. If you know that you need only one script you can just select this. Make sure the script you chose is in the same directory as specified in ***'Script Settings'***. When writing the script name into the entry you don't need the file name extension ***'.esf'***.

3.3.2 Selecting the Programming Project(s) to execute

The ***'WaitForScript.esf'*** expects values for required Programming Projects in the fields ***AutoProgram1 and AutoProgram2***. The available projects for each programmer are listed in the ***'Projects'*** table. Writing a project name that is not in the list will result in an error. The script makes sure that the project is in the programmer before execution. If version control is activated the script will also refuse to run when the version string is not completely equal.

3.4 Polling the Programmer Status

ISP-PRO will provide status information of the programming process using the very same entry you created when starting the process. The Remote Application must poll the ***Progstatus.status*** field at regular intervals in order to find out where the programmer is up to in the script. It is also a good idea to display the status to the user.

```
select status from progstatus where id=myid;
```

3.5 Resuming a two step programming process

The programming process can be divided into two parts. After finishing the first part ISP-PRO will wait for the remote application to give a signal to continue. It will then set the status to ***'AUTOPROG1 FINISHED'***.
The remote application can now work with previously read data and prepare data to be written to the target device. Use SQL commands like this:

```
insert into WriteData (ID, datatowrite) values (...)
```

The ID here must be the same as the programming process ID.
ISP-PRO will wait until the remote application sets the status to CONTINUE before starting the next project.

```
update progstatus set status='CONTINUE' where ID=myid;
```

3.6 Aborting a programming process

The ***'Remote Application'*** is also allowed to abort a running programming process. This may be required if the RA and ISP-Pro have got out of synchronisation.

To abort a programming process part way through, simply write ***'ABORT'*** into the ***progstatus.status*** field. ISP-PRO will then stop the execution of the script on the specified programming channel.

For example:

```
update progstatus set status='ABORT' where ID=myid;
```

3.7 flushing the database

When programming and testing your remote application you will probably not want to start programming processes all the time. If you just close ISP-PRO during these tests and start it again when everything is finished you will find that it starts to process all these test entries.

It is possible to instruct ISP-PRO to delete all old database entries on start up to avoid this problem. When activating this feature please be aware that every restart of the script will clean the database and set all not yet started processes to ***'autoaborted'***.

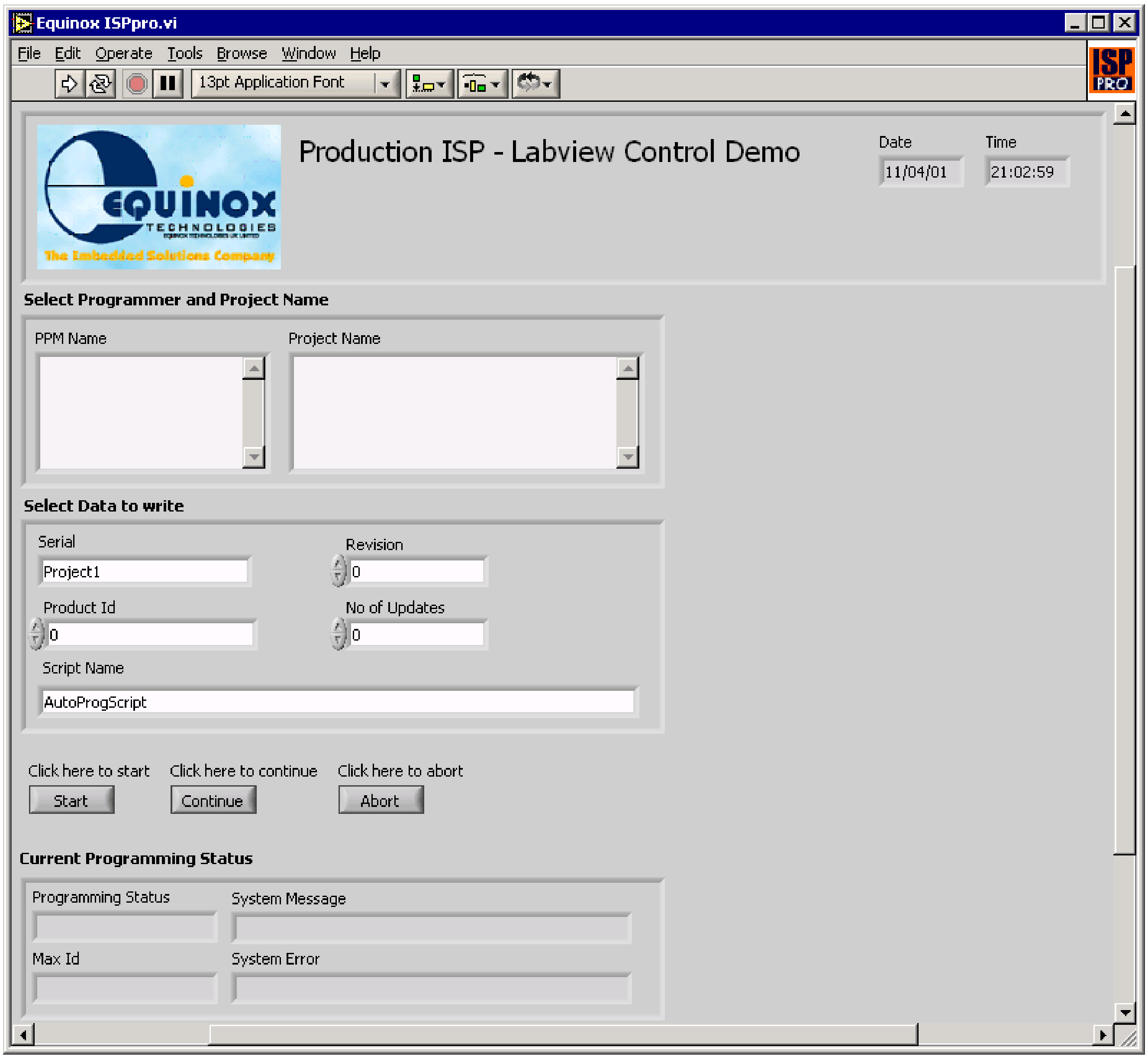
- Start EQTools
- Open a script in the Script Wizard
- In Target Connect/Disconnect select Wait for Remote Application (RA) to create a 'New Entry' -> No disconnection
- Select or deselect Flush database first of any existing records with this value

4.0 LabView 6.0 – ISP-PRO Demo Application

In this chapter you will learn how to use LabView 6.0 with Database Connectivity Toolkit 1.0 to initiate and abort program processes, how to select a project to program and how to transfer data to be written into the Flash or EEPROM.

You should have received a copy of the ***'LabView demo'*** application. You can use it as a starting point to build your own application or as a reference. It is fully functional, meaning that you can start, continue and abort programming processes with this demo. Nevertheless you will not learn how to program LabView. Please refer to your LabView programming guide for more detailed answers.

4.1 Graphical User Interface



Production ISP - Labview Control Demo

Date 11/04/01
Time 21:02:59

Select Programmer and Project Name

PPM Name
Project Name

Select Data to write

Serial: Project1
Revision: 0
Product Id: 0
No of Updates: 0
Script Name: AutoProgScript

Click here to start — Start
Click here to continue — Continue
Click here to abort — Abort

Current Programming Status

Programming Status
System Message
Max Id
System Error

4.1.1 Select Programmer and Project Name

The GUI should contain listboxes to allow the user to select a programmer and a project from this programmer.
How to get a list of programmers and projects is described in section 1.2.2.

4.1.2 Select Data to write

The values in these textfields will be written into the table EQWriteData. The script name however will be taken to select the script for this program process.

4.1.3 Current Programming Status

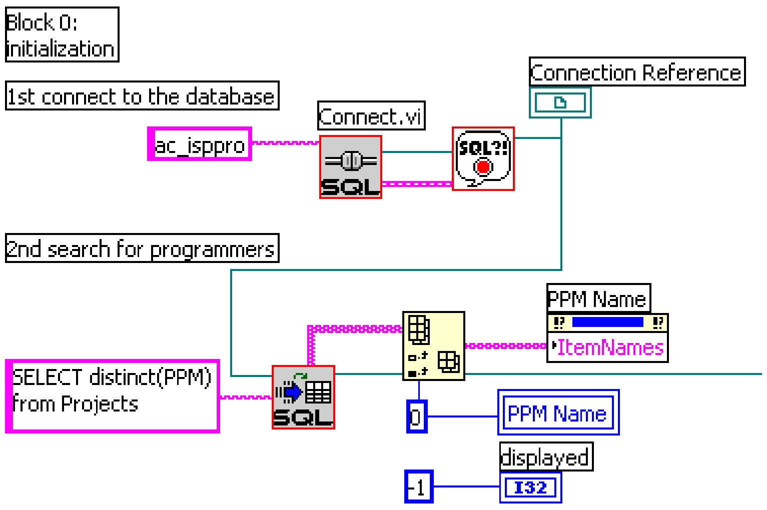
These four textfields are permanently updated and give you the current status of the program process.

4.1.4 Starting, Continuing and Aborting a program process

Three buttons to Start, Continue and Abort a program process are placed on the GUI.
The actual intelligence behind this is stored in the Block Diagram.

4.2 Block Diagram

4.2.1 Block 0: Initialization



Before starting any program processes you have to make sure that you can speak to ISP-PRO.

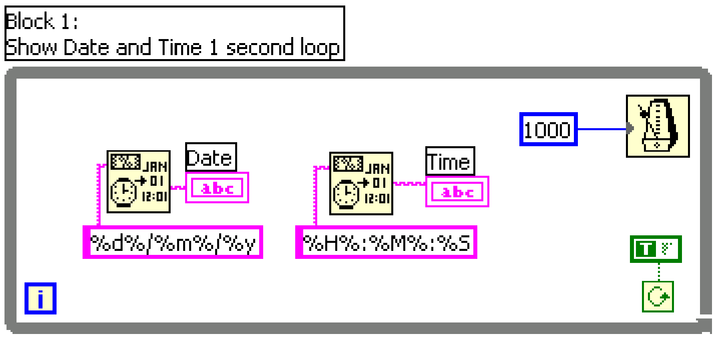
4.2.1.1 connecting to the database

First of all you have to connect to the database. Using the connect.vi you can define which ODBC link you want to open (default: “ac ISP-PRO”). It returns a Connection Reference which will be used throughout the program.

4.2.1.2 search for programmers

The second step is to read out a list of all programmers from the database (see also section 1.2.2). The SQL command SELECT distinct(PPM) from Projects will get a list of all attached programmers. Although the resultset has only one column, you have to extract this column to get an array of strings. This can than be stored in PPMName.ItemNames.

4.2.2 Block 1: Show Date and Time loop

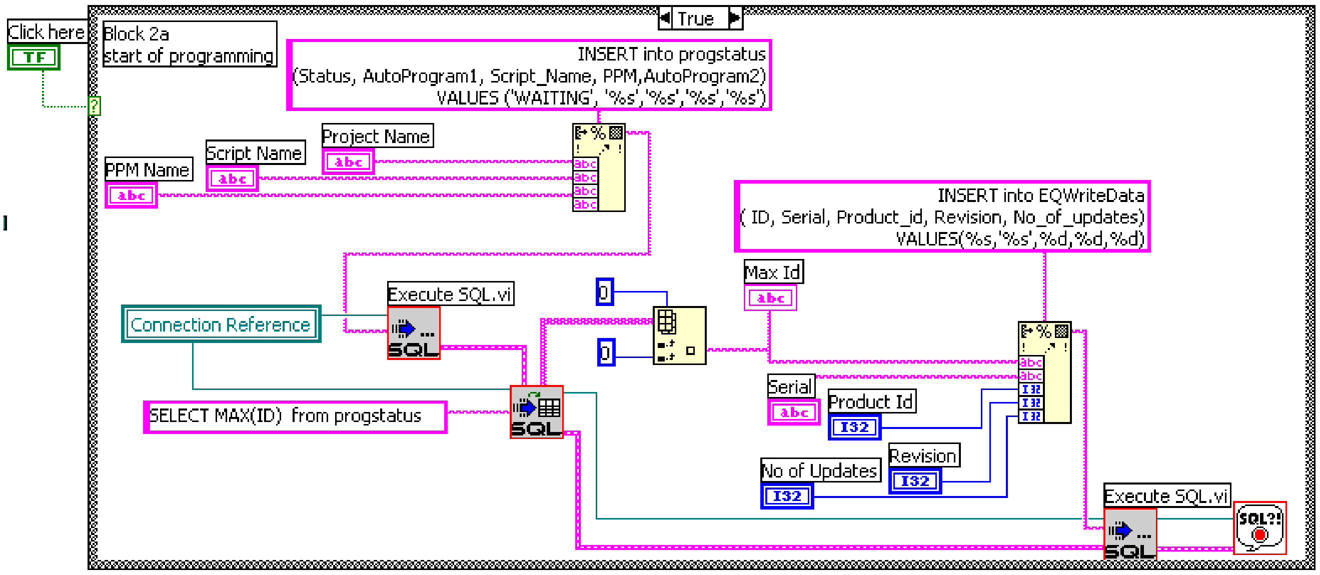


This loop displays the current date and time in two text fields on the GUI.

4.2.3 Block 2: Operation loop

This loop is the actual programming loop. It will run indefinitely unless you stop the program. If a button is pressed or if status information is to be loaded from the database or if the user selects another PPM the corresponding block will be executed.

4.2.3.1 Block 2a: start of programming



This block is executed when the button Start is pressed. It will insert a record into the progstatus table. All the required information is taken from the text fields and list boxes of the GUI.
After inserting the new entry into the database, you'll need its ID to reference it for all later reads and writes. This can be done by asking for the latest ID in the table progstatus. The SQL command

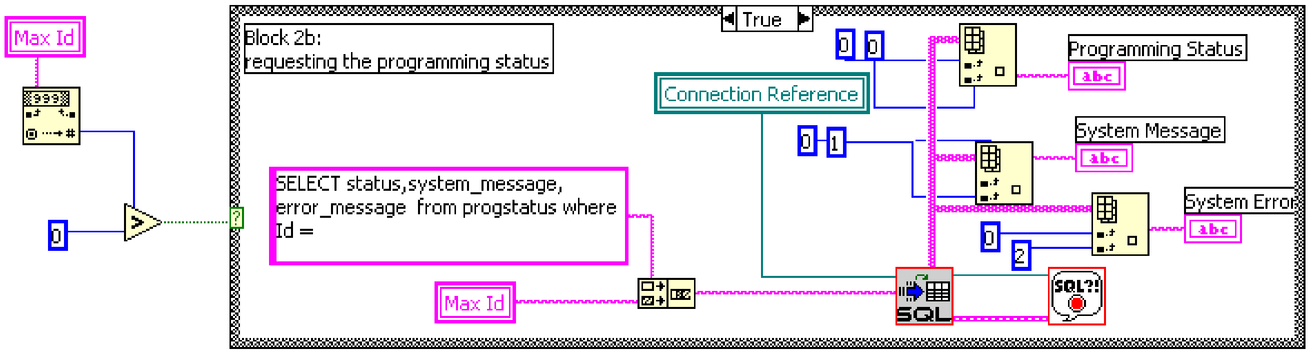
```
SELECT MAX(ID) from progstatus
```

gives us a two dimensional array containing only the maxid.
Straight after starting the program process a new entry in the EQWriteData table will be created to take all the additional data which has to be written to the target. In this example the SQL command

```
INSERT  into  EQWriteData  (  ID,  Serial,  Product_id,  Revision,  No_of_updates)
VALUES(...)
```

with appropriate values creates this entry.

4.2.3.2 Block 2b: requesting the programming status

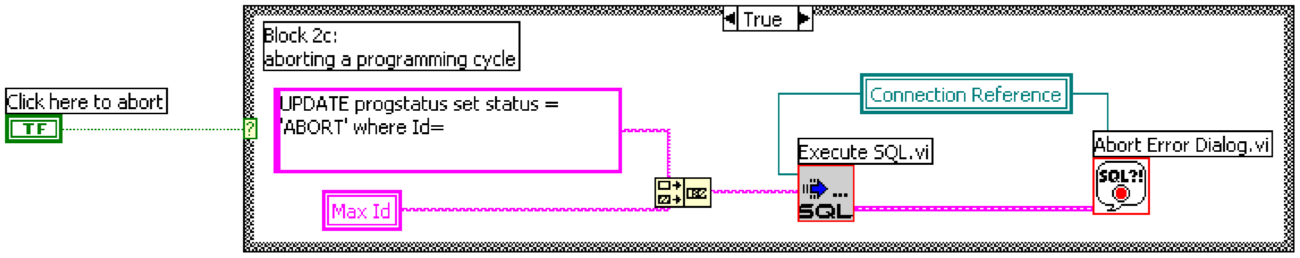


As soon as a program process has started this block gets executed and refreshes the status information. The following SQL command is used:

```
SELECT status, system_message, error_message from progstatus where Id = maxid
```

The resultset is then split into system message, error message and status and will be displayed in the corresponding text fields.

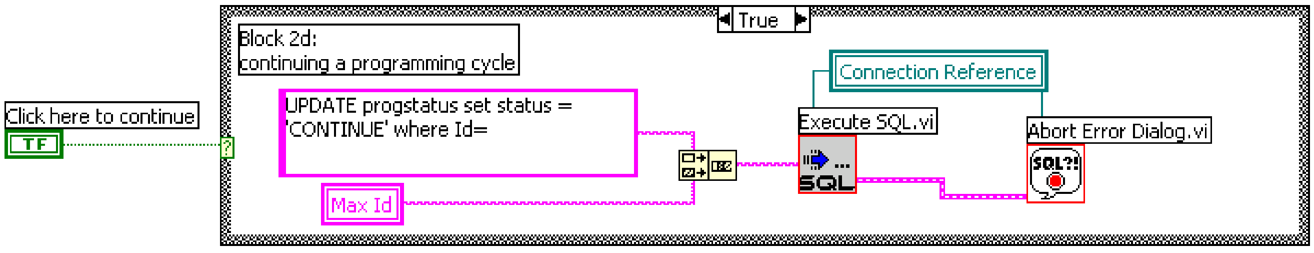
4.2.3.3 Block 2c: aborting a programming cycle



To abort a programming cycle you only have to set the status of its progstatus entry to ABORT. The SQL command to execute is:

```
UPDATE progstatus set status = ’ABORT’ where Id= maxid
```

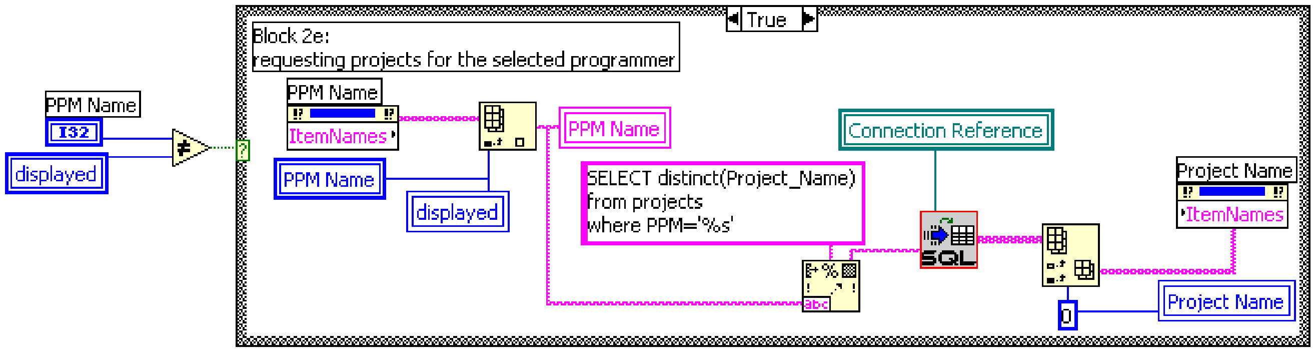
4.2.3.4 Block 2d: continuing a programming cycle



After finishing AutoProgram1 ISP-PRO will wait for the remote application to send a continue signal. This can be done by setting the status of its progstatus entry to CONTINUE. The SQL command to execute is:

```
UPDATE progstatus set status = ’CONTINUE’ where Id= maxid
```

4.2.3.5 Block 2e: requesting projects for the selected programmer

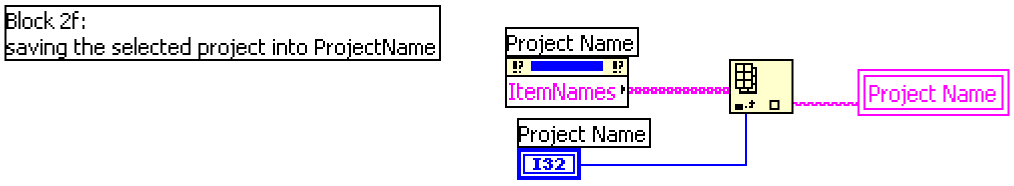


When selecting another PPM or when the program has just been started the projects list has to be refreshed. You can use the following SQL command to get a list of projects for the selected programmer:

```
SELECT distinct(Project_Name) from projects where PPM='programmer'
```

The resultset is a two dimensional array from which only the first column is needed. This column will be extracted and stored into the project name listbox.

4.2.3.6 Block 2f: saving the selected project into ProjectName

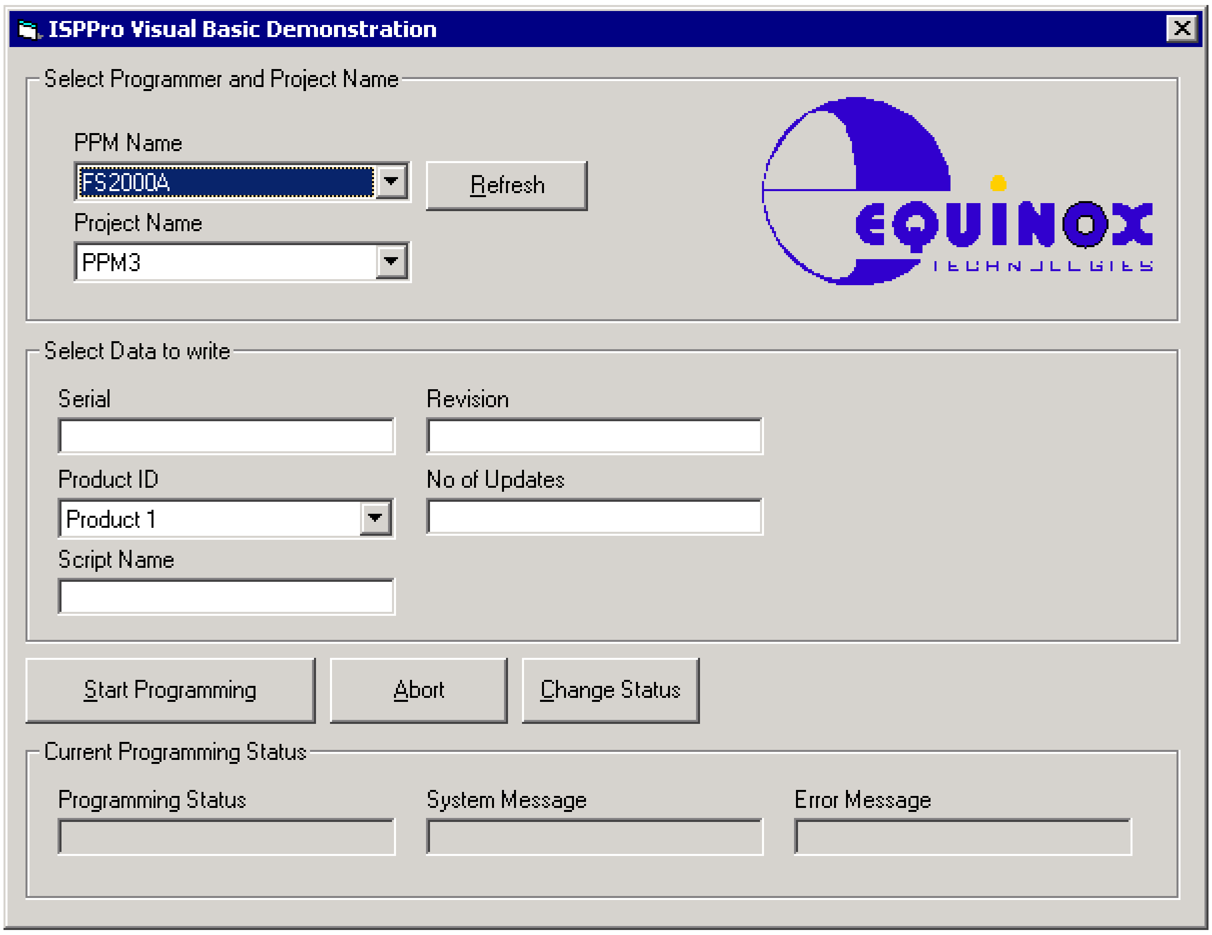


The currently selected project will be stored into a variable for easier usage.

5.0 Visual Basic – ISP-PRO Demo Application

In this chapter you will learn how to use Microsoft Visual Basic for Applications 6.0 to initiate and abort program processes, how to select a project to program and how to transfer data to be written into the Flash or EEPROM.
You should have received a copy of the Visual Basic demo application. You can use it as a starting point to build your own application or as a reference. It is fully functional, meaning that you can start, continue and abort programming processes with this demo.
This example will not teach you how to program in Visual Basic. Please refer to your Visual Basic programming guide for more detailed help.

5.1 Graphical User Interface


ISPPro Visual Basic Demonstration

Select Programmer and Project Name

PPM Name: FS2000A

Project Name: PPM3

Select Data to write

Serial:

Revision:

Product ID: Product 1

No of Updates:

Script Name:

Current Programming Status

Programming Status:

System Message:

Error Message:

5.1.1 Select Programmer and Project Name

The GUI should contain listboxes to allow the user to select a programmer and a project from this programmer.

5.1.2 Select Data to write

The values in these textfields will be written into the table EQWriteData. The script name however will be taken to select the script for this program process.

5.1.3 Current Programming Status

These three textfields are permanently updated and give you the current status of the program process.

5.1.4 Start Programming, Abort and Change Status

Three buttons to Start, Continue and Abort a program process are placed on the GUI.

5.2 Source code

This demo is meant to control only one programming process using only one database connection. This is the reason why the database connection (conn), the recordset (rsProg) and the current ID (MaxID) are global variables.
All the relevant code is stored in the subs.

5.2.1 connect to the database

Because the database connection has to be set up as the very first step, it is already executed when loading the form. The Visual Basic code for this is:

```
Set conn = New ADODB.Connection
conn.ConnectionString = "Data Source=ac_isppro"
conn.Open
```

5.2.2 getting a list of programmers

To give the user the ability to select a PPM you have to read the list of connected PPMs from the table and store it in the combobox.

```
Dim rsPPMS
Set rsPPMS = New ADODB.Recordset
rsPPMS.ActiveConnection = conn
’Select unique Programmer names
rsPPMS.Open ("SELECT DISTINCT PPM from projects")
’Add to Projects combo-box
cbPPM.Clear
While Not rsPPMS.EOF
  cbPPM.AddItem (rsPPMS("PPM"))
  rsPPMS.MoveNext
Wend
```

5.2.3 getting list of projects

As soon as the user selects a programmer the list of projects has to be updated. You will read the projects for this programmer from the same table and store it into another combobox.

```
Dim rsProjects
Set rsProjects = New ADODB.Recordset
rsProjects.ActiveConnection = conn
’Select unique Programmer names
rsProjects.Open ("SELECT Project_Name from projects where PPM = ’" & _
  cbPPM.Text & "’")
’Add to Projects combo-box
cbProjectName.Clear
While Not rsProjects.EOF
cbProjectName.AddItem (rsProjects("Project_Name"))
rsProjects.MoveNext
Wend
```

5.2.4 starting a programming process

When all the relevant information has been entered and the user hits the <Start> button, the Visual Basic application has to create an entry in the progstatus table to tell ISP-PRO that another programming process has to be started.

```
conn.Execute ("INSERT into progstatus " & _
  " (Status, AutoProgram1, Script_Name) VALUES " & _
  " (’WAITING’, ’" & cbProjectName.Text & "’,’" & _
    txtScriptName.Text & "’)")
```

5.2.5 watching the status

ISP-PRO will keep track of the programming process and collect status information. The remote application has to check and display the status.

```
rsProg.Open ("SELECT status, system_message, error_message " & _
  " from progstatus where Id=" & MaxId)
txtStatus.Text = rsProg("status")
```

5.2.6 changing the status

If you have selected a two-step script, ISP-PRO will wait until after the first Auto-Program operation has finished. The remote application may now prepare data to be written for the second project and resume the programming process by writing CONTINUE into the status.

```
conn.Execute ("UPDATE progstatus set status = ’CONTINUE’ " & _
" where Id=" & MaxId)
```

5.2.7 aborting a programming process

In order to abort a programming process the remote application has to change the status to ABORT. ISP-PRO will then stop the programming process.

```
conn.Execute ("UPDATE progstatus set status = ’ABORT ’ " & _
"where Id=" & MaxId)
```