ProfibusDP Driver

by :

TABLE OF CONTENTS

1. Description

The Profibus DP driver allows communication between an ISaGRAF program and the PB3-STT PFB3-SST-PCI board (which has a Profibus DP connection) from Woodhead.

2. Software Material

An industrial PC and a Profibus module.

3. Creation of ".bss" File with Configuration Software

First, insert the ".gsd" file of the device in the module library. This file is usually provided by the manufacturer of the equipment. To do this, copy the file '.gsd' to the directory C: / ProgramFiles / Woodhead / WoodheadProfibus / devicelibrary / Profibus_gsd. Then you just have to create the file " .bss " by following the steps below:

– *Open the Windows I / O configuration window by tapping Start → Programs → Profibus Woodhead → Console. Figure 1 shows the window that opens.*

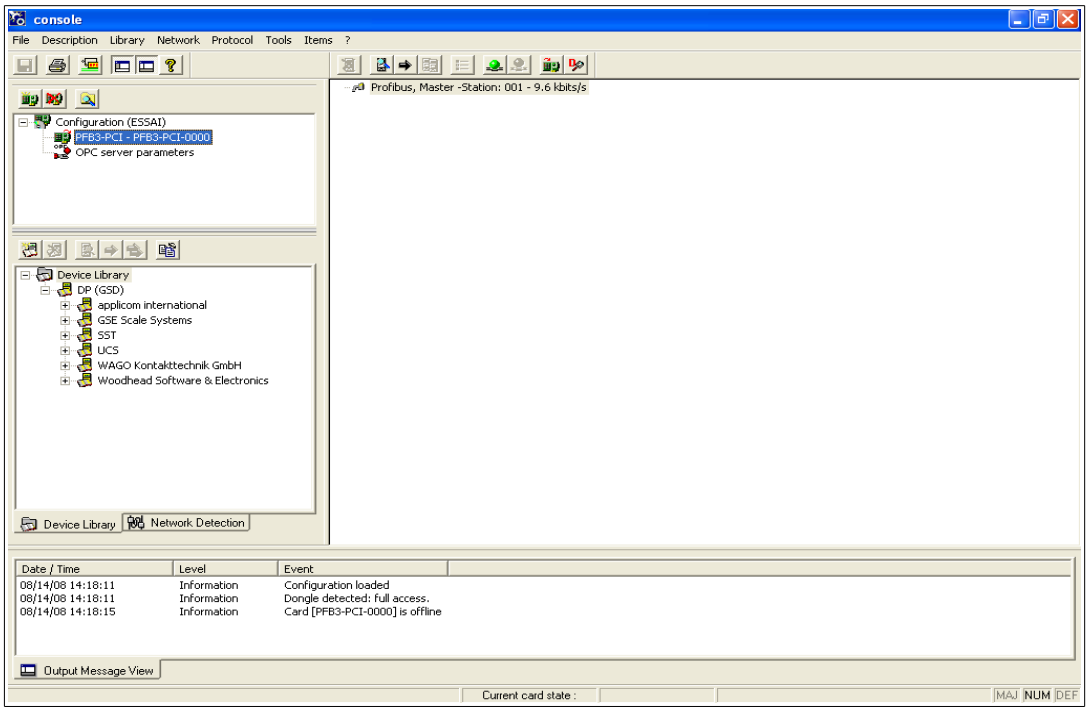

Figure 1

– *Create a new project by selecting the following : File-Configuration Manager. A pop-up window of Figure 2 will appear with the names of the existing projects. Select ‘’New” and fill in the name and the description, where applicable, at the designated areas for that purpose. Validate by clicking OK. The project is created and all that is left to do select the newly-created project and click ‘’Activate”. The Figure 1 page opens with the name of the project in paranthesies after ‘’configuration”.*

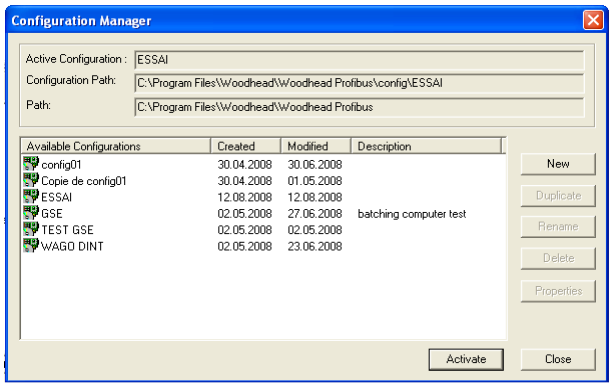

Figure 2

– *The name of the plugged equipment on the QNX target should appear as ‘’device library”.*
– *To select the desired device expand the device name tree by pressing + and then select the model of the device with a single left mouse click.*
– *Insert it into the configuration by pressing the black arrow "Insert in Configuration" (which appears above the device library list when selecting a device model). The window shown in Figure 3 is obtained.*

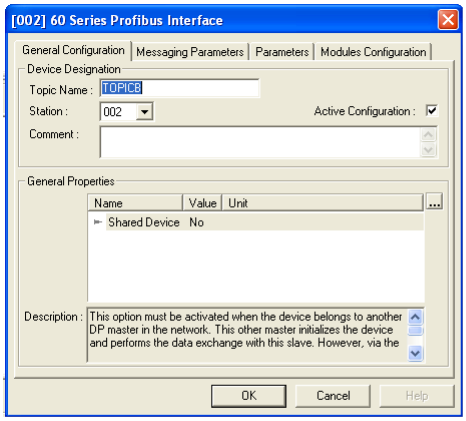

Figure 3

– In the "General Configuration" tab, enter the name (or keep the one assigned by the software) and the station number on which the module is connected. The chosen station must be the same as that physically configured on the module. If you do not know it, you can ask the configuration manager to find it for you. To do this, exit the window of Figure 3 and in the "Device Library" list right click on the "Network Detection" tab and then select the "Read Network Detection" option. When network detection has been completed, the numbers of the connected stations will be displayed. All you have to do is go back to the "General Configuration" tab of the window of Figure 3 and complete the configuration of the stations.

– *In the "Configuration Modules" tab of the same window, insert the input and output modules in the order in which you physically placed them. Click on " add " (see Figure 4) and choose the one you want to add from the list of modules then validate the insertion with the OK button.*

– *When all the modules have been added, press OK to update the configuration and get the page in Figure 5. Note that at any time you can change the order of modules using the "Move Up" " and " Move Down ".*

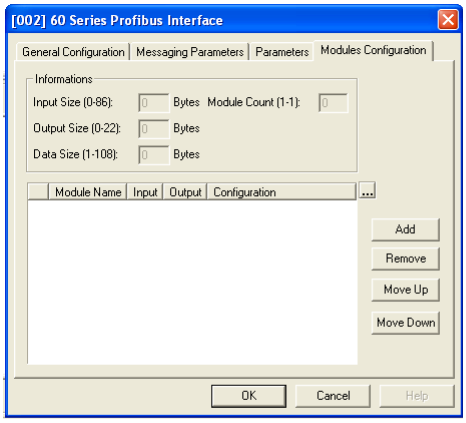

Figure 4

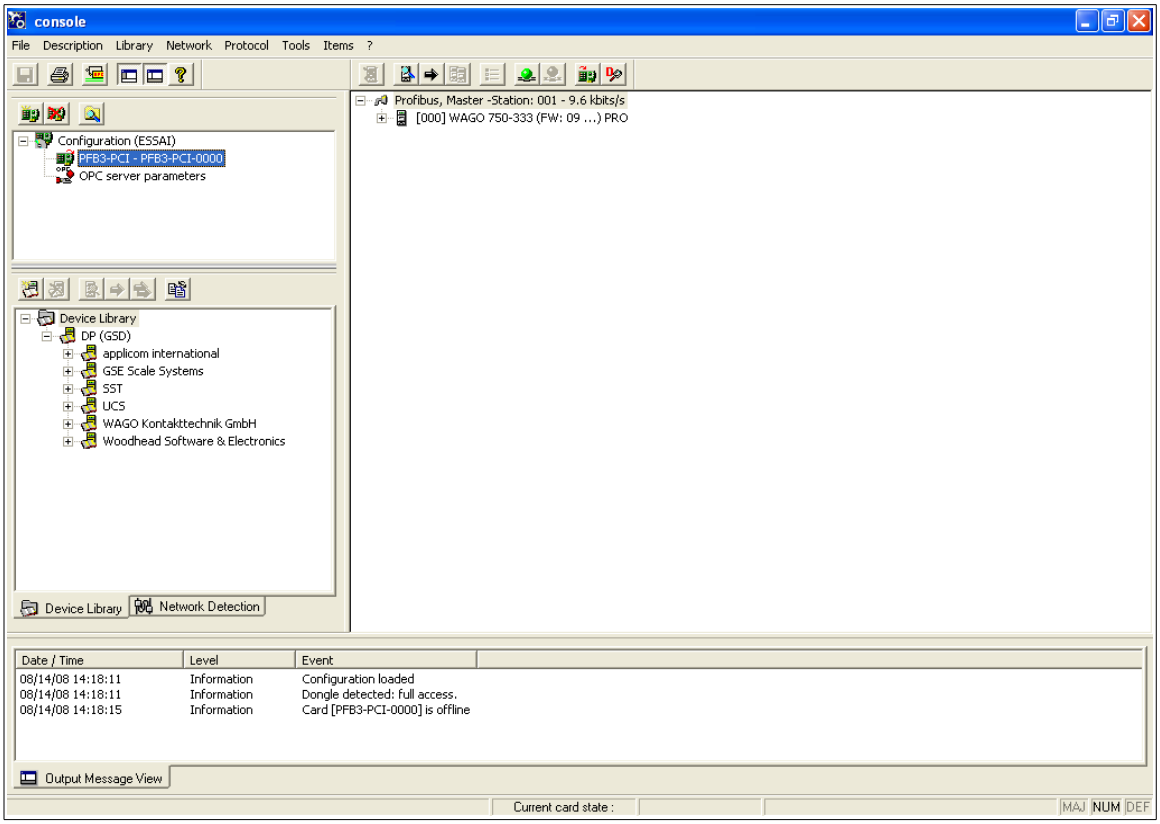

Figure 5

– The configuration is finished, save the whole with the button " Save ". The file " .bss " is thus created and updates each time it is backed up. To retrieve the created file, go to the directory C: / ProgramFiles / Woodhead / WoodheadProfibus / config / configurationname1.

4. Driver Installation and Use

To use the Driver, you first need to transfer the .bss and .ss3 files to the PB3 subdirectory (which the user must create) in the / usr / v2000 directory of the industrial PC, directory in which the real-time programs run. Then put the file " pb3profibus.so " in the BIN subdirectory of the / usr / v2000 directory.

To transfer the files " .bss ", " .ss3 " and " pb3profibus.so " proceed as follows:

If the files are already on target QNX:

- *To position in the v2000 repertory of usr :* ***cd /usr/v2000***
- *Created in the PB3 repertory in v2000 :* ***mkdir PB3***
- *Copy the ".bss" file in the PB3 repertory :* ***mv /path[1]/nameoffile.bss PB3***
- *Copy the ".ss3" file in the PB3 repertory :* ***mv /path/nameoffile.ss3 PB3***
- *Copy the "pb3profibus.so" file in the BIN repertory :* ***mv /path/pb3profibus.so BIN***

If the files are on the Windows development station:

- *Go to the directory where the files to be exported are found :* ***cd /path***
- *Open a ftp session with QNX target :* ***ftp 192.168.1.120***
- *To connect :* ***user : Virgo2000 et password : v2000***
- *To go into binary mode for sending or receiving files :* ***bin***
- *Apply the ".bss" file if on a Windows station otherwise procede to next step :* ***put nameoffile.bss***
- *Apply the".ss3" file if on a Windows station otherwise procede to next step :* ***put nameoffile.ss3***
- *Change directory (de /usr/v2000 à /usr/v2000/bin) :* ***cd bin***
- *Insert driver file (".so") :* ***put pb3profibus.so***
- *End the ftp session :* ***bye***

- *Open telnet session :* ***telnet 192.168.1.120***
- *To connect :* ***user : root***
- *To position in the usr v2000 directory :* ***cd /usr/v2000***
- *Create PB3 directory in v2000 :* ***mkdir PB3***
- *Move ".bss" file in PB3 directory :* ***mv /path/nameoffile.bss PB3***
- *Move ".ss3" file in PB3 directory :* ***mv /path/nameoffile.ss3 PB3***
- *Change the permissions and the driver file modes :*
  cd bin
  chown root : root pb3profibus.so
  chmod a+x pb3profibus.so
  chmod a+s pb3profibus.so
- *End telnet session :* ***exit***

In the Windows development station with ISaGRAF installed, copy the file Profibus_QNX6.txt to the directory C: \ Program Files \ ICS Triplex ISaGRAF \ Run Time 51 \ Qnxnto and import it (File → Import → Definition of PLC) into the project to which we want to integrate the driver. Once the import is completed, go to the I / O Wiring option of the resource you would like to connect to the driver, then add the desired devices and choose the number of inputs and / or outputs you would like. For each of these devices, configure the parameter values as described in the Configuring the Driver Properties section.

1 *Path of the directory where the file is located*

5. Driver Property Configuration

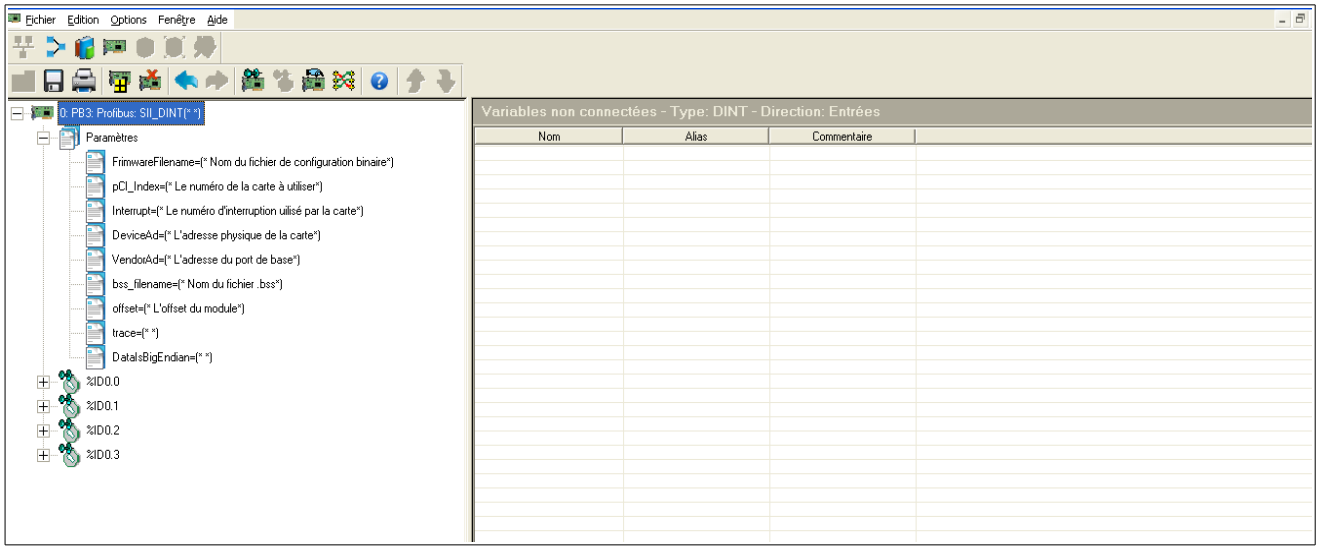

Figure 6

The properties of the Driver that appear as shown in Figure 6 can be changed according to the parameters of the card being used. To configure the driver, you must specify the fixed parameters, giving them the following values:

- PCI_Index: 1. Card number to be used as an integer. For card type PCI SST set the index of the first card at 1 and for card type ISA or PC104 set at 0.
- Interrupt: 3. Integer which indicates the interrupt number used by the card.
- DeviceAd: 33. Hexadecimal represents the card's physical address.
- VendorAd: 133D. Address of the base port in the hexadecimal.

The variable parameters[2]:

- Firmware Filename: Name of binary configuration file: bit pattern / binary code with the ".ss3" extension.
- bss_filename: Name of the file " .bss " used to configure the input and output modules attached to the driver. This is generated using the STT card configuration software under windows as described in chapter creating the file " .bss " using the configuration software.

- Offset: Integer that indicates the offset in byte (8-bit batch) applied to a module. It allows the user to manage the reading and writing of the data.

- Trace: Boolean which allows to activate (TRUE) or to deactivate (FALSE) the displays in the program in order to debug it.

- DataIsBigEndian: Boolean that allows the user to specify whether the device he is using is Big Endian for Motorola (TRUE) or Little Endian processors for Intel (FALSE) processors. Depending on the type specified, the driver will invert the bits of the data so that they are read or written in the correct direction.

2 *See chapters 3 and 7 respectively to create the ".bss" file and to calculate the offsets.*

6. Physical Address Definition

To connect the physical I / O to the Driver, the user must define I / O modules according to the type it wants. This driver has the following types of available:

Type	Channels	Bits
SII_DINT	1-242	32
SIO_DINT	1-242	32
SII_UINT	1-242	16
SIO_UINT	1-242	16
SII_INT	1-242	16
SIO_INT	1-242	16
SDI_BOOL	1-243	
SDO_BOOL	1-243	
SAI_REAL	1-242	32
SAO_REAL	1-242	32

Type :
- SII_DINT : DINT type input module (Double Integer)
- SIO_DINT : DINT type output module
- SII_UINT : UINT type input module (Unsigned Integer)
- SIO_UINT : UINT type output module
- SII_INT : INT type input module (Integer)
- SIO_INT : INT type output module
- SDI_BOOL : BOOL type input module (Boolean)
- SDO_BOOL : BOOL type output module
- SAI_REAL : REAL type input module
- SAO_REAL : REAL type output module

Channels: This is the number of channels that can be used in a module.

Bits : Number of bits that make up each type.

7. Offset Configuration

After adding the desired equipment in IsaGRAF (see chapter Installation and use of the driver), values must be given to the offset parameter. An offset is an 8-bit offset (or 1 byte) that allows the input and output modules to be arranged in order to read and write to specific locations. The channels of the input modules and those of the output modules are each stored in a separate memory space. Thus the offsets of the input modules are independent of those of the output modules and vice versa.

Let us take the case of the input modules to illustrate the calculation of the offsets, the same principle applies for the output modules:

- The first input module is simply positioned at the beginning of the memory space therefore has an offset of 0.

- The position of the second module depends on the number of bits used by the first. Thus if the first is a 4-channel digital input module (SDI_BOOL), 4 bits of memory space must be reserved for it. Since the offset is a multiple of 8, 8 bits of space will be granted for the first module (even if the last 4 are unused). The offset of the second module will ultimately be 1.

- Following this logic, the offset of the third module will depend on the position of the second. Thus if the second is a 4-channel integer double input (SII_DINT) module, since an integer double is represented on 32 bits:

  32bits/8bits=4bytes
  *4bytes*4channels=16bytes*
  Must foresee an offset of 16 +1 (offset of preceding module) =17 for the third module,
  and so on for the other input modules.