



“Productizing” IPm Applications

Abstract:

This technical note describes suggested techniques to make third party applications programs and I/O drivers function as if they were part of the IPm product platform and interact seamlessly with other IPm applications. Installing your programs as a product package, integrating your configuration utility into the I/O Tool Kit software, and automating configuration loading as part of the complete project are discussed.

Introduction

By following the suggestions in this technical note, you will create applications programs that are “product packaged and completely Linux transparent. (Users will not need any Linux knowledge.) Your applications will be easy to install using simple Windows utilities (supplied by SIXNET), easy to configure, and most importantly will not require repeated applications support from you. Applications created with these techniques will be well integrated into the IPm and the I/O Tool Kit software and appear to be a polished product that has been developed in partnership with SIXNET.

SIXNET will help you market your software products if they follow the guidelines in this file.

Porting Your Applications to Linux

Since so many situations involve porting existing applications to run in Linux (and more specifically an IPm embedded controller), this technical note starts with this subject. Linux is obviously a high level operating system that has equal or better resources than the platform you are porting from. (It is certainly better than Windows!) If you wish, you may start your development by porting your application to any computer running Linux. We suggest using a Pentium running Red Hat Linux. Red Hat can be downloaded for free or purchased for a nominal charge on a convenient CD. Since the IPm runs a very compatible Linux kernel, this is a reasonable way to begin. The most significant difference is that you will not have the IPm I/O database to read and write I/O registers – this is obviously machine specific. You will find the documentation for the IPm I/O calls in the file [ipm_io_calls.pdf](#). With this difference and perhaps a few other application specific exceptions, your ported application will run on any Linux platform.



Seamless Performance with Other IPm Applications

At the heart of the IPm controller is a “shared resource” I/O Database— basically a data structure (complex array) of I/O registers that all applications read and write I/O values. The I/O polling tasks exchange I/O values with physical I/O and registers in other controllers (peer moves). Applications such as ISaGRAF and Sixlog datalogging interface with this database as well. Your applications and I/O drivers simply exchange register values through simple function calls. In this manner, your programs can interact with all existing IPm functionality as well as other applications written by third parties. The I/O database contains discrete (bit), analog (16 integer and unsigned counters), long integers (32 bits) and floating point registers. Access to these registers is made through a library supplied by SIXNET and described in the document: [ipm_io_calls.pdf](#).



Applications Programming Suggestions

You may create your applications programs with a free hand. Just be sure to be conservative with system resources; other applications need to run in addition to yours. Advice on using the IADK (IPm Applications Development Kit) is provided on the SIXNET CD and on the SIXNET web sites. You will need an “IPm Advanced” software license to receive the I/O database library and to access some of the I/O Tool Kit features discussed in this and other associated documents. You will also need a SIXNET controller with an “IPM” in its part number (such as ST-IPM or VT-IPM) as a target platform for development.

User-defined configuration details are best defined in a configuration file that your application finds during run time. The content and format of this file is completely up to you, but we suggest using a simple ASCII format to make the file easy to read and edit when necessary. We suggest using a naming convention for configuration files that allows the user to create file names appropriate for each station or application and that will be easily recognized as one of your files, both upon visual inspection and by your application at run time. Make it easy to use the wild card features in the I/O Tool Kit and other utilities. The use of file prefixes is one good technique. Using a prefix is not required, but may make it easier to identify a set of files for one purpose.

Example of the configuration file for the user’s station #3: station3.xyz_setup
Loading files as station3.* would be an easy way to specify files for station #3.

Create an entry in the “etc/rc3.user” file to have your application start up automatically. Advice on this subject is provided in the file: “iadc_use_guide.pdf” which is supplied as part of the IADK development tools.



Create an Installation Tar File

Create a tar file to install your application in an IPm station. This tar file can copy the required files into the proper directories, set file permissions, edit system files (such as the etc/rc3.user startup file), verify or set required environment variables, verify that other up-to-date firmware files are present, and any other activities. You have a choice of scripting language to use. Be aware that a Perl interpreter is installed in the IPm and is a powerful scripting tool.

The tar file can be installed by the user from the File Operation window in the I/O Tool Kit or the Loader utility as an automated operation. Your application can be loaded into any SIXNET station that contains an IPm engine. This includes the ST-GT-1210 and OEM versions of this industrial controller. Creating a tar file allows the user to install your application cleanly using only convenient Windows software. It will also insure that your applications package is installed properly and avoid unnecessary support calls.

Provide a Configuration Utility

If your application requires user-defined configuration details, we suggest creating a Windows configuration utility for this purpose. If your application is a slave I/O driver it is possible that a configuration utility is not required. The "[Customizing the I/O Tool Kit](#)" file will tell you how to add your driver to the available protocols list for the appropriate communication ports and how to read this configuration from the ports definition files at run-time.

May we suggest that the documentation for using your application be provided as part of the Windows configuration utility? This documentation may be provided in any format (.hlp., .html, .pdf, ...). Typically it is referenced through an action button within the utility entitled "Help".

Integrate Your Configuration Utility into the I/O Tool Kit

Your utility may be integrated into the I/O Tool Kit by following the instructions provided in the technical note, "[Customizing the I/O Tool Kit](#)". Your utility will appear of the I/O Tool Kit "Tools" menu. Through the use of macros that SIXNET has provided, the Tool Kit may pass information about the highlighted (selected) IPm station to your utility. Your utility can automate the loading of the configuration for the target station(s) by adding the resulting configuration file(s) to the "Files to Load" list for the appropriate IPm stations. Refer to the technical note "[Using External Configuration Utilities with the I/O Tool Kit](#)" for more details.

