Borland Pascal Development Environment/ RETOS o.s.

Introduction

Use the Borland Pascal Development Environment/ Retos o.s. to create and debug complex or simple applications to be used in the KITV40, KIT386EX control units with the TERM01, TERM05, TERM06 terminals, in the TERM10 industrial terminal or in the KOMPAKT control system. Use this environment when your task requires a relatively fast time response.

Borland Pascal

The user can use a rich variety of procedures and features included in the SofCon libraries to write his/her own programs in Borland Pascal, V7.0 (necessary knowledge of dynamic variables and at least basic knowledge of object programming).

SofCon Libraries

SofCon® Ltd. offers a number of library modules to create applications and debug programs. The library units are designed to be used with the Borland Pascal compiler. Generally, the units are supplied in .TPU format. The user can use files containing "interface" sections, manuals and examples of how to use each library. A few selected libraries are supplied in the source format and, on negotiation, SofCon® Ltd. can also provide the user with some more libraries in the source format.

Depending on the platform on which the programs are to be running, the libraries fall into two categories:

- **DOS platform** - When you compose your application using these libraries, you must run it on a MS DOS-based personal computer. When running on a PC processor, the application will run faster than on the MCP platform discussed later. You must simulate inputs and outputs for the controlled process using simulator modules that you can run while using RETOS o.s., parallel to the application you are developing (for example, the TERM10B simulator on a PC screen). To monitor a real-time process, extend your PC with the PC-KITV40 board including IO-bus and P-bus, connected to additional boards of the SofCon configuration set.
- **MCP platform** - If you use these libraries to compose your application, you must run the application on a real HW SofCon configuration set including the KITV40 or the KIT386EX processor board. To make the debugging process easier, connect a VGA card to the screen and a PC keyboard to the KITV40 set using the EXPP01 board. For the KIT386EX configuration set, you need to connect the VGA card to the PC104 bus. You can use these additional peripherals to add debugging outputs and inputs to your application

In addition, you can group the libraries into several categories depending on the type of activity they are supposed to perform:

- **Communication libraries** - They contain a hierarchical model of operating any communications. The bottom layer provides communication using a particular HW channel, the top layer provides a secured protocol for messaging. For communication, use the SofCon's proprietary PRN protocol or specified third-party protocols, such as TECOMAT(TECO), Sbus (SAIA), DF1(Allan Bradley), E-Bisinc(Eurotherm), LECOM(Lenze), Festo, SMS (GSM modules) and more.
- **Libraries to create terminals and generate menus** - Contain a hierarchical object model of operating the TERM10, TERM03, TERM01, TERM05 a TERM07 terminals. They provide both terminal screen listings (3 layers - underlying bitmap, graphical objects, and text) and the menu operation for each screen.

- **SofCon card drivers** - Contain basic drivers for more sophisticated HW boards (A/D, D/A converters, etc., LED displays, etc.)
- **Control libraries** - Provide one-dimension control algorithm objects. Included are PID algorithms, adaptive settings PID, internal model PID, etc. All algorithms provide hitless switching between the automatic and the manual mode. Control algorithm constants also allow hitless modification during the regulation process. The parent object of the controller is also available. The programmer only creates the inheritor, for which he defines new virtual methods to acquire input values, and a method to perform an action. The required control algorithm is specified while the control object instant is being created. Even without perfect knowledge of the currently used control algorithms, this simple and fast approach allows to create controllers of temperature, pressure, revolution speed. The number of simultaneously running controllers is limited only by the processor's performance.
- **Other general libraries** - the most important general libraries include libraries to operate FLASH memory, libraries implementing RAM and ROM disks in the memory space over 1MB for the KIT386EX processor board, further libraries for printer operation and many other mathematical support libraries

ReTOS Real Time Operating System

The supplied ReTOS o.s. enhances Borland Pascal to provide features characteristic for specialized parallel and real-time languages. You can divide a single complex control task into multiple parallel tasks called processes. ReTOS o.s. provides process switching, allows specifying and changing the priority between processes, provides resources for time management and real-time process control. You can forward messages between processes using a number of various message boxes. The system provides resources to operate interruptions.

ReTOS-DEBUGGER Booting and Debugging Program

You use ReTOS-DEBUGGER to deploy, boot and debug applications. This PC-based program uses a serial line to communicate with a BIOS monitor or a Kernel monitor of the given application in KITV40, KIT386EXR. The ReTOS debugger allows to place your application to separate memory segments, and record the application to the Flash or RAM memory or to the connected EPROM simulator using a serial communication line.

ReTOS DEBUGGER can further display and modify memory contents, global variable values, or values of inputs and outputs. It enables to view the state of the ReTOS o.s., that is, the state of processes, the Ready and Delay queues, and the state of clipboards. The ReTOS debugger can also trace a program and insert break instructions into the program that is being debugged, provided it is placed in the RAM memory.

Methods to Debug the Currently Developed Applications

Choose one of the following methods to debug your application:

- First, write your application to run directly on a DOS-based PC. To do this, compose the application including DOS libraries, add the TERM10 simulator and examine the behaviour of the program directly on your PC. At this stage, you can use any debugging features provided by the Borland Pascal environment.
- At the second stage, extend your PC with the PC-KITV40 board. To this board (it includes IO-Bus and P-Bus), you may attach the KIT HW set without a processor board. This stage allows to connect real inputs and outputs. The actual control algorithm, however, is still performed by the PC processor. Note that the dynamic behaviour of your application is different (faster) at this stage from the way it will behave after it is transferred to the KITxxx processor board.
- The next stage is to run the application using the original processor board. Use ReTOS DEBUGGER and the system serial channel placed on the processor board to record the application to the processor board's RAM or FLASH. If you prefer to record the application code to the EPROM memory, use ReTOS DEBUGGER to create a .BIN file for the EPROM programmer, or record this file to the EPROM simulator

that you connect to the processor board instead of the real EPROM memory.To a certain extent, this option allows to send the debugging listings directly to the TERM10 screen or any other connected terminal. Another option is to temporarily extend the KITV40 configuration with EXPP01 and ISO-VGA (8-bit) boards, or the KIT386EX configuration with PC104-VGA, and to send the debugging listings to the temporarily connected classical VGA monitor.

- The last option is the full utilization of ReTOS DEBUGGER. If you choose this option, you must record the application to the RAM memory and temporarily attach to it the SW monitoring module that is likely to slow down the application's performance. The application must not use the system serial channel for its own use. This done, you can do your debugging directly using ReTOS DEBUGGER from the connected PC, linked to the application using the system serial channel.

Conclusion

Borland Pascal Development Environment / ReTOS o.s. is a comprehensive development tool, allowing to effectively develop most complex applications to be used in drive control, production line regulation, etc.

This tool is meant to be used by application developers who are, at the same time, professional programmers. Compared to KIT-Builder, the other optional development environment, Borland Pascal Development Environment / ReTOS requires more in-depth knowledge of programming.