MOTOROLA

Embedded SDK (Software Development Kit)

PC Master Software User Manual

SDK111/D
Rev. 1.5, 02/20/2002

MOTOROLA intelligence everywhere™ | digital dna™

How to reach us:
USA/EUROPE/Locations Not Listed: Motorola Literature Distribution; P.O. Box 5405, Denver, Colorado 80217. 1–303–675–2140 or 1–800–441–2447

JAPAN: Motorola Japan Ltd.; SPS, Technical Information Center, 3–20–1, Minami–Azabu. Minato–ku, Tokyo 106–8573 Japan. 81–3–3440–3569

ASIA/PACIFIC: Motorola Semiconductors H.K. Ltd.; Silicon Harbour Centre, 2 Dai King Street, Tai Po Industrial Estate, Tai Po, N.T., Hong Kong. 852–26668334

Technical Information Center: 1–800–521–6274

HOME PAGE: http://www.motorola.com/semiconductors/

Contents

Chapter 5
Project Options

Chapter 6
Scripting From the HTML Framework

Chapter 7
License

List of Figures

About This Document

This manual describes the PC master software user application for use with Motorola's Embedded Software Development Kit, (SDK).

Audience

This document targets software developers implementing the PC master software user application within software applications.

Organization

This manual is arranged in the following sections:

- **Chapter 1, Introduction** — provides a brief overview of this document
- **Chapter 2, Questions and Answers** — provides answers to frequently-asked questions about the PC master software user application
- **Chapter 3, Installation** — describes requirements and directions for enabling the PC master software
- **Chapter 4, Program Usage** — contains a description of user interface for the PC master software
- **Chapter 5, Project Options** — explains available application and project options
- **Chapter 6, Scripting From the HTML Framework** — explains the use of HTML pages within the PC master software
- **Chapter 7, License** —provides the license required to use this product

Suggested Reading

We recommend that you have a copy of the following references:

- *DSP56800 Family Manual*, DSP56800FM/AD
- *DSP56824 User's Manual*, DSP56824UM/AD
- *Inside CodeWarrior: Core Tools*, Metrowerks Corp.

Conventions

This document uses the following notational conventions:

Typeface, Symbol or Term	Meaning	Examples
`Courier Monospaced Type`	Code examples	`//Process command for line flash`
Italic	Directory names, project names, calls, functions, statements, procedures, routines, arguments, file names, applications, variables, directives, code snippets in text	...and contains these core directories: *applications* contains applications software... ...CodeWarrior project, *3des.mcp* is... ...the *pConfig* argument.... ...defined in the C header file, *aec.h*....
Bold	Reference sources, paths, emphasis	...refer to the **Targeting DSP56F80x Platform** manual.... ...see: **C:\Program Files\Motorola\Embedded SDK\help\tutorials**
Blue Text	Linkable on-line	...refer to Chapter 7, License....
Number	Any number is considered a positive value, unless preceded by a minus symbol to signify a negative value	3V -10 DES^{-1}
ALL CAPITAL LETTERS	# defines/ defined constants	# define INCLUDE_STACK_CHECK
Brackets [...]	Function keys	...by pressing function key [F7]
Quotation marks, "..."	Returned messages	...the message, "Test Passed" is displayed.... ...if unsuccessful for any reason, it will return "NULL"...

Definitions, Acronyms, and Abbreviations

The following list defines the acronyms and abbreviations used in this document. As this template develops, this list will be generated from the document. As we develop more group resources, these acronyms will be easily defined from a common acronym dictionary. Please note that while the acronyms are in solid caps, terms in the definition should be initial capped ONLY IF they are trademarked names or proper nouns.

DSP	Digital Signal Processor or Digital Signal Processing
FFT	Fast Fourier Transforms
FIR	Finite Impulse Response
I/O	Input/Output
IDE	Integrated Development Environment
IIR	Infinite Impulse Response
LSB	Least Significant Bit
MIPS	Million Instructions Per Second
MSB	Most Significant Bit
OnCE™	On-Chip Emulation
SDK	Software Development Kit
SPI	Serial Peripheral Interface
SRC	Source

References

The following sources were used to produce this book:

1. *DSP56800 Family Manual*, DSP56800FM/AD
2. *DSP56824 User's Manual*, DSP56824UM/AD
3. *Embedded SDK Programmer's Guide*, SDK101/D

Chapter 1
Introduction

The PC master software application is a software tool initially created for developers of Motor Control applications but it may be extended to any application development. This tool allows control of an application remotely from a user-friendly graphical environment running on a PC. It also provides the ability to view real-time application variables in both textual and graphical form.

1.1 PC Master Software Features

Main features:

- Graphical environment
- Visual Basic Script or Java Script can be used for control of target board
- Easy to understand navigation
- Connection to target board is possible over a network, including the Internet
- Demo mode with password protection support
- Visualization of real-time data in Scope window
- Acquisition of fast data changes using integrated Recorder
- Value interpretation using custom defined text messages
- Built-in support for standard variable types (integer, floating point, bit fields)
- Several built-in transformations for real type variables
- Automatic variable extraction from Metrowerks' CodeWarrior linker output files (MAP, ELF)
- Remote control of application execution

Chapter 2
Questions and Answers

First question: why place this topic immediately after the introduction? The reason is really quite practical. While writing this User Manual, the following questions were raised. Since the answers to these questions clarified terms and topics described further in the manual, it was decided to put this topic before those that are more detailed and perhaps less easily understood.

2.1 Why do I need it?

The primary goal of developing PC master software was to deliver a tool for debugging and demonstrating Motor Control algorithms and applications. The result was a tool with the versatility to be used for multipurpose algorithms and applications. Some real-world uses include:

- Real-Time debugging - PC master software allows users to debug applications in true REALTIME through its ability to watch variables. Moreover, it allows debugging at the algorithm level, which helps to shorten the development phase.
- Diagnostic tool - PC master software's remote control capability allows it to be used as a diagnostic tool for debugging customer applications remotely across a network.
- Demonstrations - PC master software is an outstanding tool for demonstrating algorithm or application execution and variable outputs.
- Education - PC master software may be used for educational purposes. It' application control features allow students to play with the application in demonstration mode to learn to control program execution and to study the effects on algorithms.

2.2 What does it do?

PC master software communicates with the target system application via serial communication to read and write application internal variables. PC master software provides the following visualization features for displaying thin variable information in a friendly format:

- Oscilloscope - provides monitoring/visualization of application variables in the same manner as a classical oscilloscope with CRT.

Note: Monitoring rates are limited by the serial communication rate.

- Recorder - provides monitoring/visualization of application variables that are changing at a rate faster than the sampling rate of the oscilloscope. While the Scope periodically reads PC master software variable values and plots them in real-time, the Recorder is running on the target board. Variable values are sampled into a buffer on the board, then the sampled data is downloaded from board to PC master software. This mechanism allows a much shorter sampling period and enables sampling and plotting of very quick actions.

2.3 Why is it so great?

The algorithm can be demonstrated in one block, or divided into several blocks, depending on which better reflects the algorithm structure. The blocks may then be demonstrated, one block at a time. Each block's input parameters may be explored to observe how they affect output parameters. Each block has a description tab for explaining algorithm details.

2.4 What could I do with it if I follow the instructions?

Using the accompanying demo project, it is easy to learn how to use PC master software by playing with the project's defined blocks and parameters. The demo project allows you to understand how to control the application as well. You can go into details of each item, check its properties, change parameters and determine how each can be used in your application. For a detailed explanation of the parameters, see **Chapter 4**.

2.5 How is it connected to a target development board?

PC master software requires a serial communication port on the target development hardware. Connection is made using a standard RS-232 serial cable. On one side, the cable is plugged into the PC serial port (COM1, COM2 or other) and, on the opposite side, into the target development board's serial connector.

2.6 What are all of these dialog boxes for?

In **Chapter 4**, you will see pictures with dialog boxes. These dialog boxes are used as a questionnaire, where you will enter parameters describing, for example, one algorithm block or application variable and its visualization.

2.7 How does a project relate to my application?

One target board application may have multiple PC master software projects, each performing different purposes. For example, three specific PC master software projects can work with the same board application to provide three different purposes:

- to provide information used during debug process
- to provide service maintenance capabilities
- may be used for learning about your application during operator training phase

2.8 How do I set up remote control? Why would I want to?

For remote control, you need at least two computers connected via a network, one running the stand-alone program PC master software Server and the second running the standard PC master software program. The target development board is then connected to the computer running PC master software Server.

Remote control operation is valuable for performing remote diagnostics. A customer's applications may be diagnosed remotely by connecting the target development board to the remote PC and then running the PC master software locally with a service project for this customer's application. Remote diagnostics can now be performed without travelling to the customer site.

2.9 What is the Watch-grid?

Watch-grid is one of the panes in PC master software. It shows selected application variables and their content in readable format. The application variables displayed are selected separately in the block property settings of each project block.

2.10 What is the Recorder?

Recorder is created in software on the target development board and stores changes of variables in real-time. You can define the time interval of storing and which variables will be stored. After the requested number of variable samples are stored within the Recorder buffer on the target board, they are downloaded from the board to PC master software, where they are displayed in the Recorder pane. This Recorder buffer is located in memory on the target development board and is limited in size. The main advantage of the Recorder is the ability to sample very fast actions.

2.11 What is the Oscilloscope?

PC master software Oscilloscope is similar to the classical hardware oscilloscope. It shows graphically selected variables in real-time. The variable values are read from the board application in real-time through the serial communication line. Oscilloscope GUI looks similar to the Recorder, but in contrast to it, the sampling speed of variables is limited by the communication data rate between the PC master software and the target board application. The default communication rate is 9600bps.

Chapter 3
Installation

3.1 System Requirements

The PC master software application can run on any computer with Microsoft Windows NT, 98 or 95 (+DCOM) operating system with Internet Explorer 4.0 or higher installed.

The following requirements result from those for the Internet Explorer 4.0 application:

Computer: 486DX/66 MHz or higher processor (Intel Pentium recommended)

Operating system: Microsoft Windows NT, Windows 98 or Windows 95 + DCOM pack

Memory: Windows 95 or 98: 16 Mb RAM minimum (32 Mb recommended); for Windows NT: 32 Mb RAM minimum (64 Mb recommended)

Required software: Internet Explorer 4.0 or higher installed. For selected features (e.g. regular expression-based parsing), Internet Explorer 5.5 or higher is required.

Hard drive space: 8 MB

Other hardware requirements: Mouse, serial RS-232 port for local control, network access for remote control

3.2 Target Development Board Requirements

PC master software relies on the following items to be provided by the target development board:

Interface: Serial communication port

Data Memory: Recorder buffer - buffer size can be defined in the target board application; default size is 500 words

Program Memory: Required size is approximately two kwords in the default full configuration; optionally, some features can be removed to reduce required program memory size

3.3 Enabling PC Master Software on Target Application

To enable the PC master software operation on the DSP target board application, add the following line to the *appconfig.h* file: *#define INCLUDE_PCMASTER.*

See the **SDK Programmer's Guide** for further configuration information.

3.4 How to Install

The PC master software application is part of the Motorola Embedded SDK and may be selectively installed during SDK installation.

Chapter 4
Program Usage

This chapter contains a detailed description of the user interface for the PC master software program.

4.1 Application Window Description

When the application is started, the main window is displayed on the screen. When there is no project loaded, the welcome page is displayed in one part of the window. The initial look of the main window is shown in **Figure 4-1**.

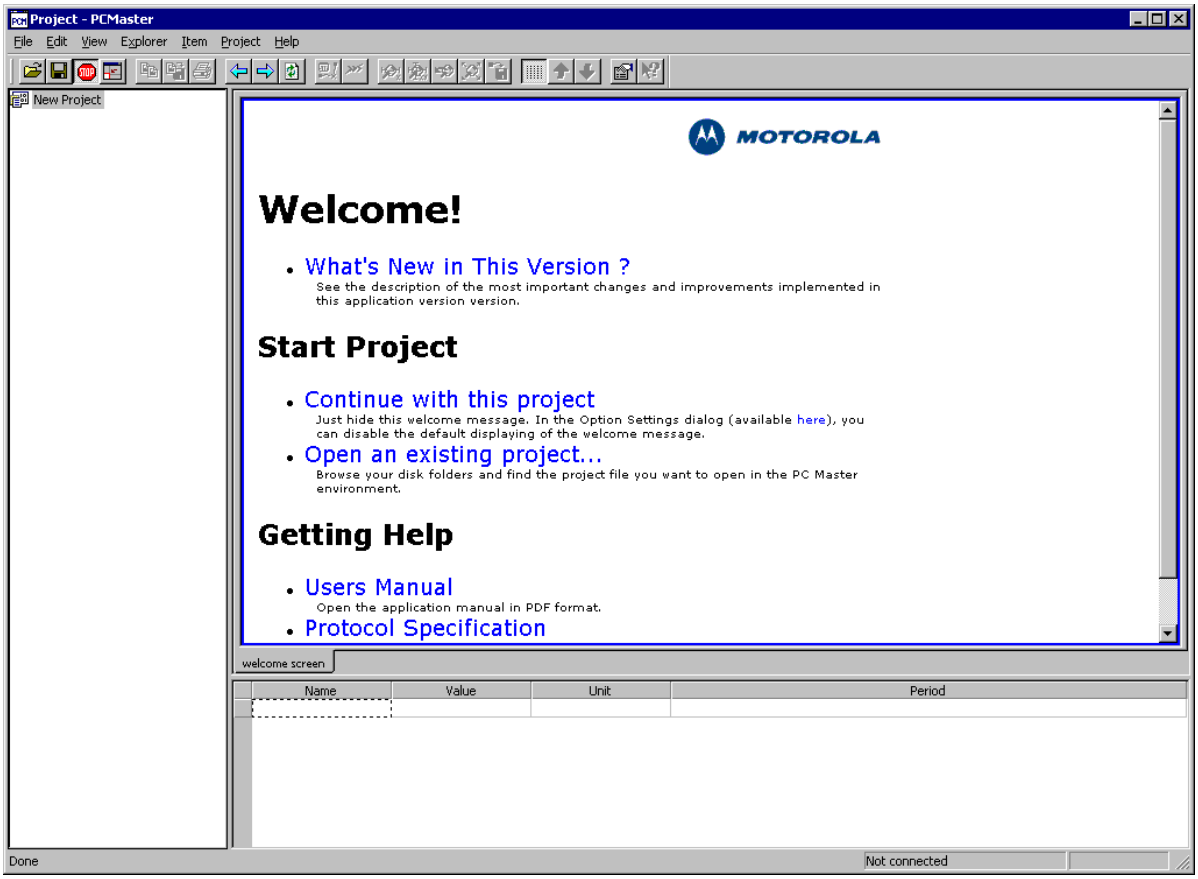

Figure 4-1. Initial Application Window

The welcome page contains links to the documentation and to the application help. There are also several other links corresponding to the standard menu commands (for example, the *Open project* command).

In the rest of this chapter, the application usage is shown on the example motor control project. The project, together with the source code of the related embedded application, can be found in Motorola's Embedded Software Development Kit.

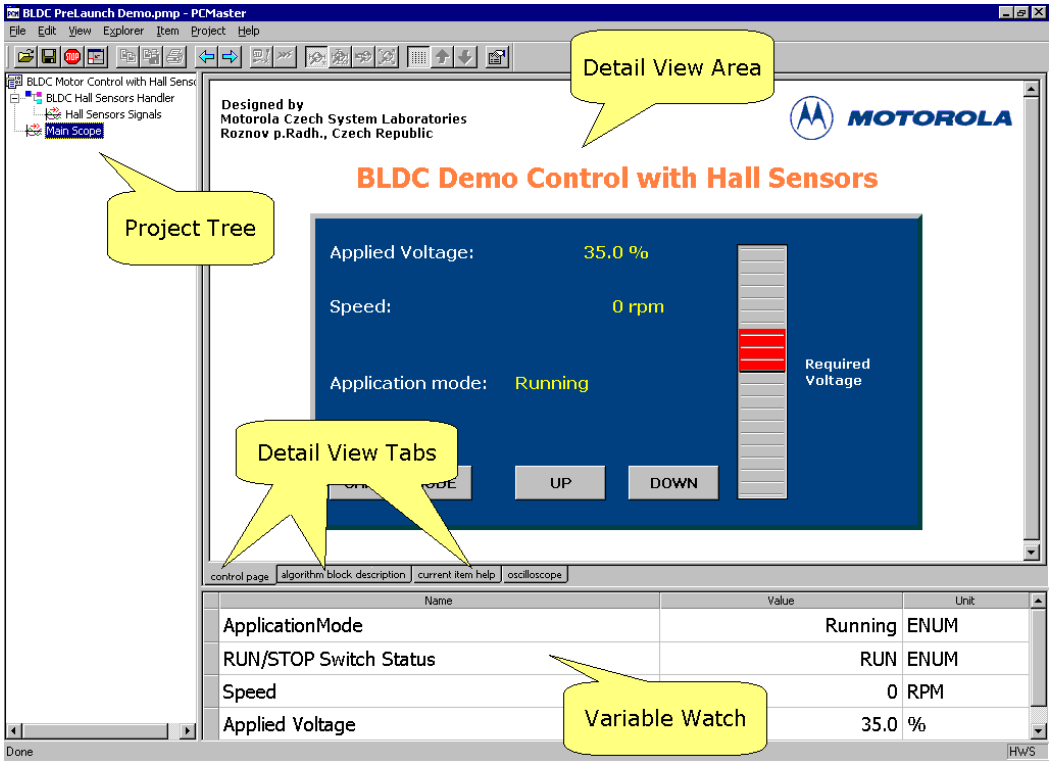

Figure 4-2. Application Window

As shown in **Figure 4-2**, the application window consists of three panes: the *Project Tree* pane, the *Detail View* pane and the *Variable Watch* pane. There is also the optional *Fast Access* pane, which can display in the lower half of the *Project Tree* pane and is further described in **Section 4.5.3**.

The *Project Tree* pane contains a logical tree structure of the application being monitored/controlled. Users can add project sub-blocks, Scope, and Recorder definitions to the project block in a logical structure to form a *Project Tree*. This pane provides point and click selection of defined *Project Tree* elements.

The *Detail View* pane dynamically changes its content depending on the item selected in the *Project Tree*. Depending on the type of the item selected in the tree, this pane also provides several tabs to sub-pages of additional information associated with the item.

- *control page* = An HTML page created for controlling the target system
- *algorithm block description* = An HTML page or another document whose URL is defined in the selected *Project Tree* item's properties
- *current item help* = Another HTML document whose URL is defined in the Scope or Recorder properties.
- *oscilloscope* = A real-time graph displaying application variables as defined in the Scope properties
- *recorder* = A graph displaying recorded application variables as defined in the Recorder properties

The *control page*, when defined, is available for all *Project Tree* items to allow the user to control the board at any time. The content of the *algorithm block description* page changes with the *Project Tree* item selected. When a Scope or Recorder is selected from the *Project Tree*, the *current item help* and *oscilloscope/recorder* chart pages are also available.

The *Variable Watch* pane contains the list of variables assigned to the watch. The pane displays the immediate variable values and also enables you to change them (if this is enabled in the variable definition).

All the information related to one application is stored in a single project file with the extension "*.pmp*" **.** This information includes the project settings and options, the *Project Tree*, *Detail View* HTML pages, real-time chart definitions, watch interface settings, variables and commands, stimulators and more.

4.1.1 Project Tree

When a new project is created, the *Project Tree* window contains an empty structure with just one root project block called *"New Project"*. You can then change properties of this block or add sub-blocks, Scopes or Recorders to the structure.

Property changes and *Project Tree* additions can be done in two ways:

- Select an item in the *Project Tree* and right mouse click to use the local menu
- Select an item in the *Project Tree* and select main menu *"Item"* pull-down

4.1.1.1 Project Block and Sub-block

The *Project block* should cover an integral component of the selected algorithm. *Sub-blocks* may be added should the user care to break the algorithm into multiple blocks. Each block has its own *algorithm block description* page, (shown in **Figure 4-19**), watch variables and commands. All of these can be defined in the *Project block properties* dialog, as shown in **Figure 4-3**.

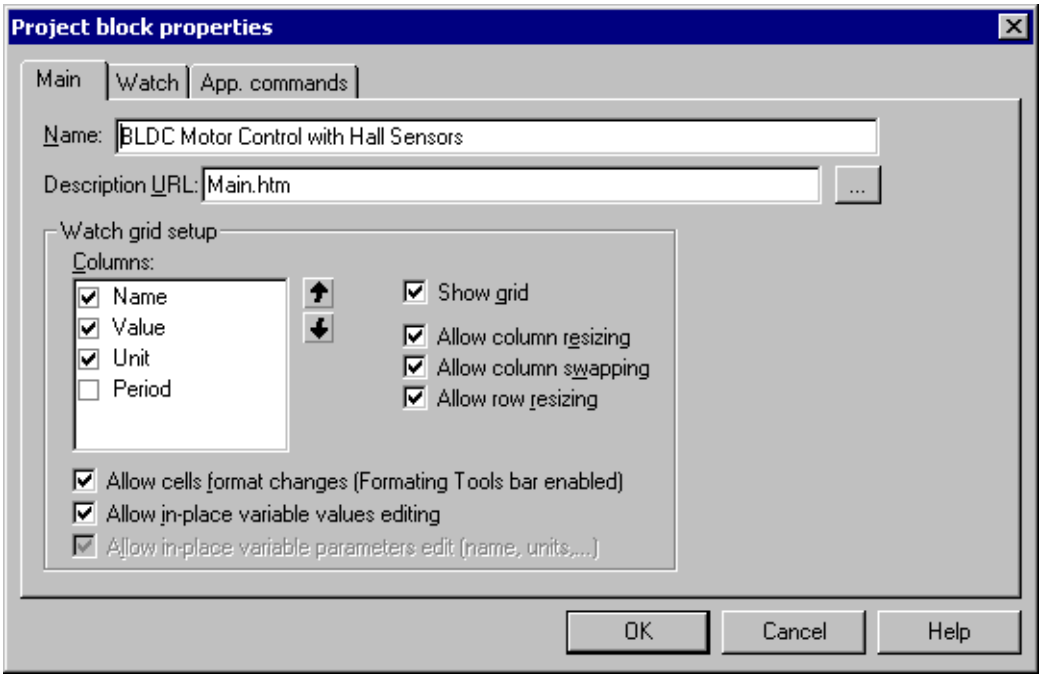

Figure 4-3. Project Block Properties - Main Page

The *Main* page contains the following user configuration items.

- *Name* = Name of Project block that will be displayed in the *Project Tree*
- *Description URL* = Select a description URL or .htm or .html file to be shown in the *Detail View* pane under the *algorithm block description* tab. This file may be created with any HTML editor such as MS Front Page Express or Netscape Composer.
- *Watch-Grid setup* = You can select columns to display in the Watch-Grid (Name, Value, Unit, Period); specify column order using the Up and Down arrow buttons; check the grid behavior options (column resizing/swapping, row resizing); allow format changes to grid cells with Watchbar (see **Section 4.6.2**) and edit in-place variable values by checking the next option boxes.

The *Watch* page shown in **Figure 4-4** selects which PC master software project variables are to be watched in the context of this project block.

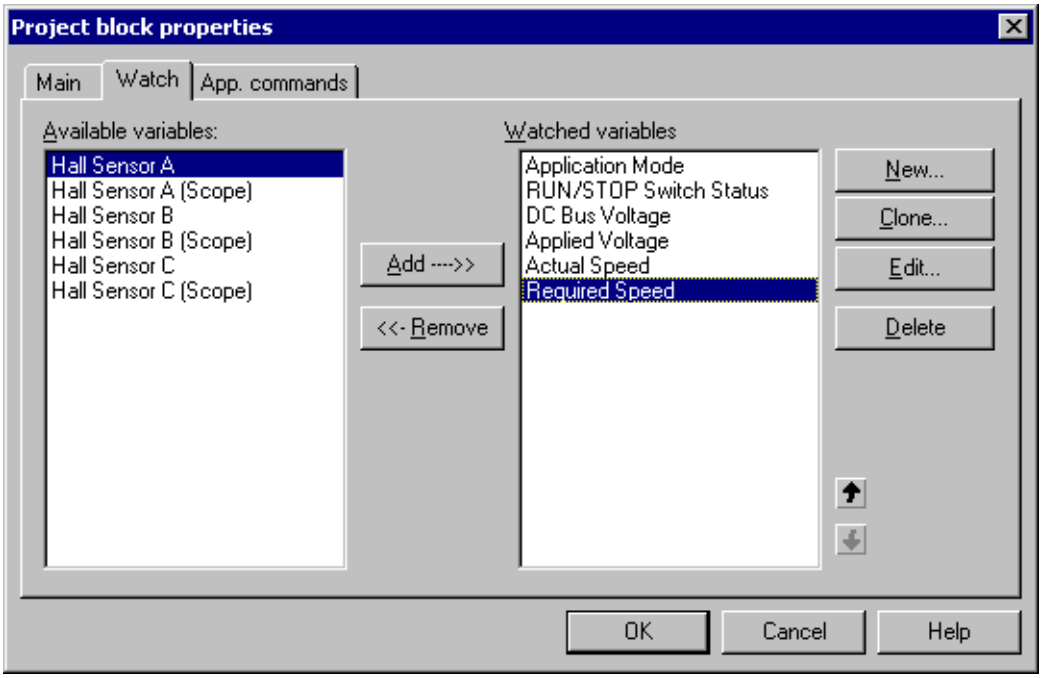

Figure 4-4. Project Block Properties - Watch Page

The variables in the *Watched variables* list are the project variables which are currently selected for watching in the Watch-Grid. The *Available variables* list contains the remaining available project variables not selected for watching with the current block item selected in the tree. You can use the following buttons:

- *Add/Remove* = Moves variables into and out of the *Watched variables* window
- *New* = Creates a new variable (see **Section 4.2**)
- *Clone* = Creates a new variable based on a copy of the selected variable
- *Edit* = Changes the selected variable properties
- *Delete* = Deletes the selected variable from the project
- *Up/Down arrows* = Sets the display order of the watched variables in the Watch-Grid

PC master software communicates with the board application by reading/writing variables and/or sending commands. See **Section 4.3** to learn how to define a PC master software project command. As the variable appearance in the Watch-Grid can be dependent on the block selected in the *Project Tree* pane, the availability of application commands can also be dependent on the selected block. The *App. commands* page, shown in **Figure 4-5**, sets which commands will be available in the *Fast Access* pane in the context of this project block and also enables the management of application commands.

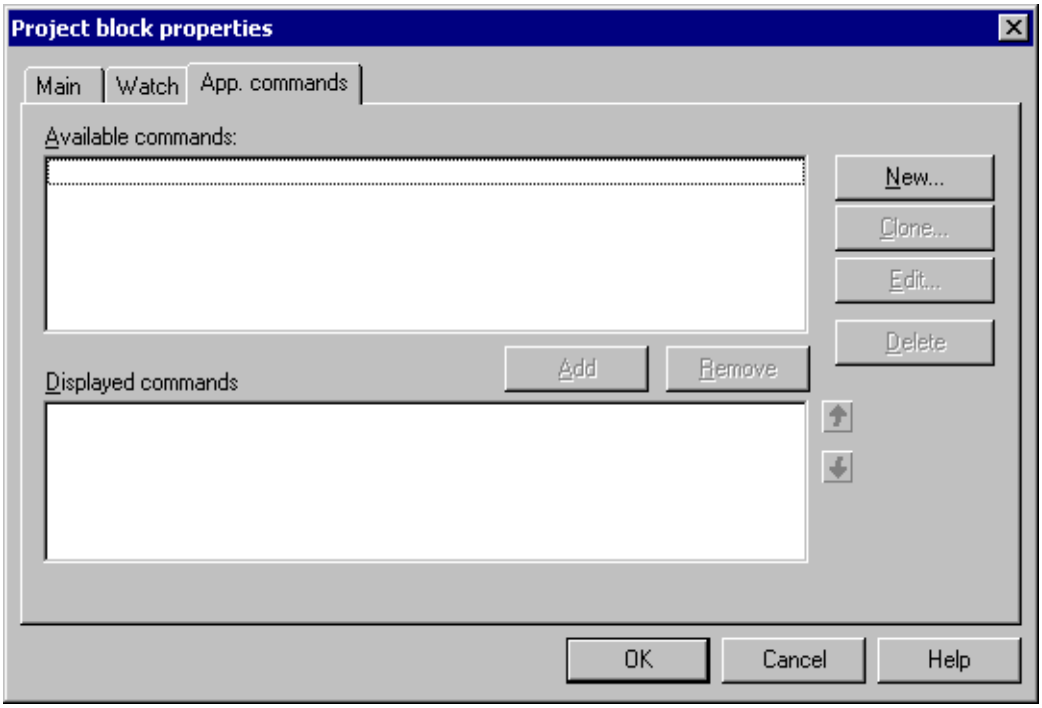

Figure 4-5. Project Block Properties - App. commands Page

The commands are listed in the *Available commands* window. Use the *Add* button to move a command into the *Displayed commands* window and to make it available in the *Fast Access* pane. Use the *Remove* button for reverse operation. You can use the following buttons:

- *New* = Creates a new command (see **Section 4.3**)
- *Clone* = Creates a new command based on a copy of the selected command
- *Edit* = Changes the selected command properties
- *Delete* = Deletes the selected command
- *Up/Down arrows* = Sets the display order of commands in the *Fast Access* pane

4.1.1.2 Scope

The *Scope* item in the *Project Tree* structure defines a new real-time oscilloscope chart to be shown in the *Detail View* pane. The Scope properties window, shown in **Figure 4-6**, allows you to configure the appearance and characteristics of the scope chart.

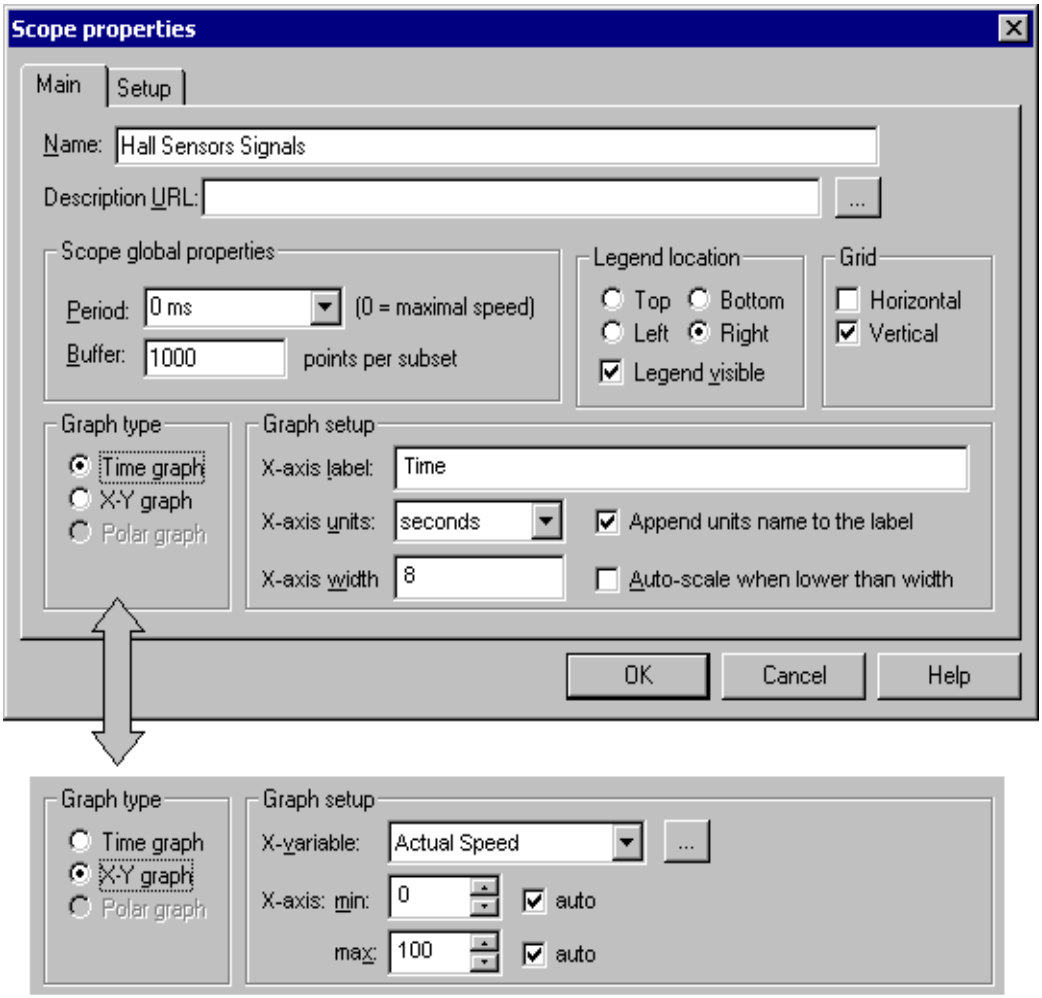

Figure 4-6. Scope properties - Main Page

The *Main* page contains these user configuration items:

- *Name* = The name of the Scope item that will be displayed in the *Project Tree*
- *Description URL* = Specify the URL of the document or local path to a file to be shown in the *Detail View* pane under the *current item help* tab. This file may be created with any HTML editor, such as MS Front Page Express or Netscape Composer, and should explain the chart variables and settings to the user.
- *Scope global properties* = Common properties for all scope variables
  - *Period* = Oscilloscope sampling period
  - *Buffer* = The number of samples in one data subset in the chart
- *Legend location* = Set the visibility and location of the chart legend
- *Grid* = Choose the horizontal and/or vertical grid lines to be displayed in the chart
- *Graph type* = Select the mode of oscilloscope operation
  - *Time graph* = A variable (*values* versus *time)* will be displayed in the chart
  - *X-Y graph* = Inter-variable dependencies (*value* versus *value*) will be displayed in the chart
- *Graph setup* (for *Time graph*):
  - *X-axis label* = Specify the name displayed for the X axis
  - *X-axis units* = Select the axis units

  - — *X-axis width* = Specify the range of the X axis
  - — *Auto-scale X axis until width is reached* = Scales the axis width after Scope start when the length of subset is shorter than the *X axis width*
- *Graph setup* (for *X-Y graph*):
  - — *X-variable* = Selects the variable whose values will be used for X axis values
  - — *X-axis min* = Sets the X axis lower limit value (checks the *auto* box to enable X axis auto-scaling)
  - — *X-axis max* = Sets the X axis upper limit value (checks the *auto* box to enable X axis auto-scaling)

The *Setup* page is used to assign variables to be displayed on the oscilloscope chart. Up to eight *Chart vars* (seven in *X-Y* mode) may be selected for display in the chart and assigned to a maximum of five Y-blocks. Select one of eight positions and browse for a variable in the drop-down list below the list to set or change a variable at this position. Select the first (empty) item in the drop-down list to clear the selected position in the list. Each chart variable is assigned to a Y block.

The Y block is a graph element represented by one left Y axis and, optionally, one right Y axis also. The Y blocks can be drawn separately, or overlapped in the graph.

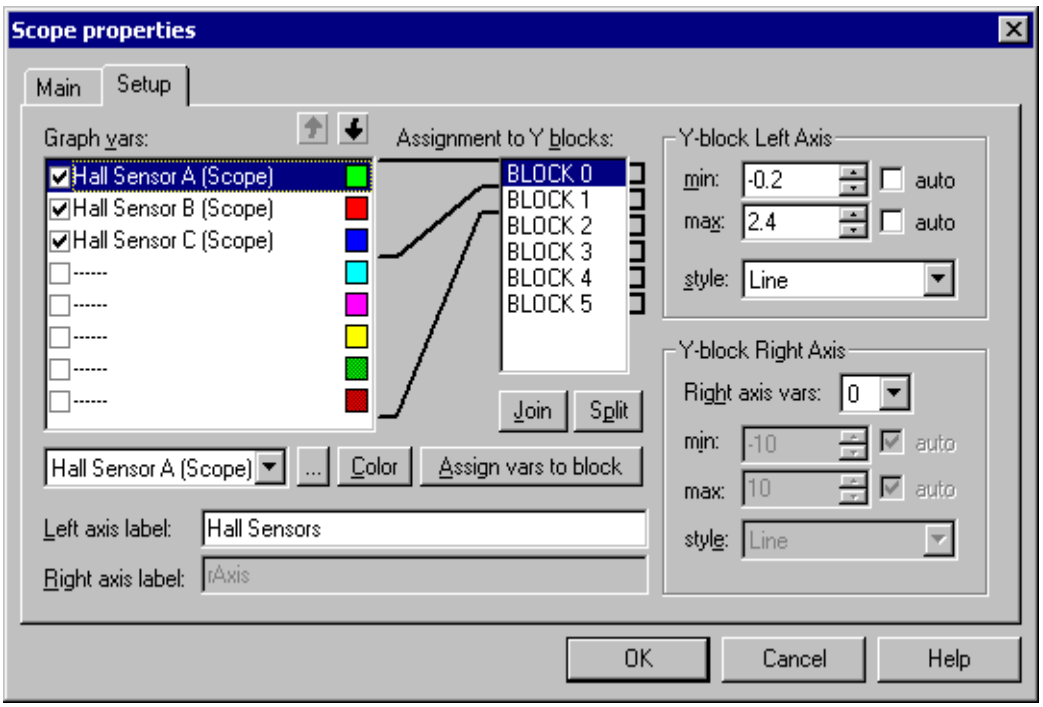

Figure 4-7. Scope Properties - Set-up Page

- *Assign vars to block* button = Assigns successive selected *Chart vars* to a Y block. Select the chart variables you want to group into one Y block and press this button. The simple assignment of three variables into single Y block is shown in **Figure 4-7**.
- In the *Y-block Left axis* frame, set the axis range by specifying the *min* and *max* axis value, or check *auto* box to enable automatic minimum and/or maximum tracking. Select a *Style* of drawing the data subsets from the drop-down list box.
- Type the *Left axis label*, which will be assigned to a selected Y-block.

The resulting oscilloscope chart is displayed in **Figure 4-8**.

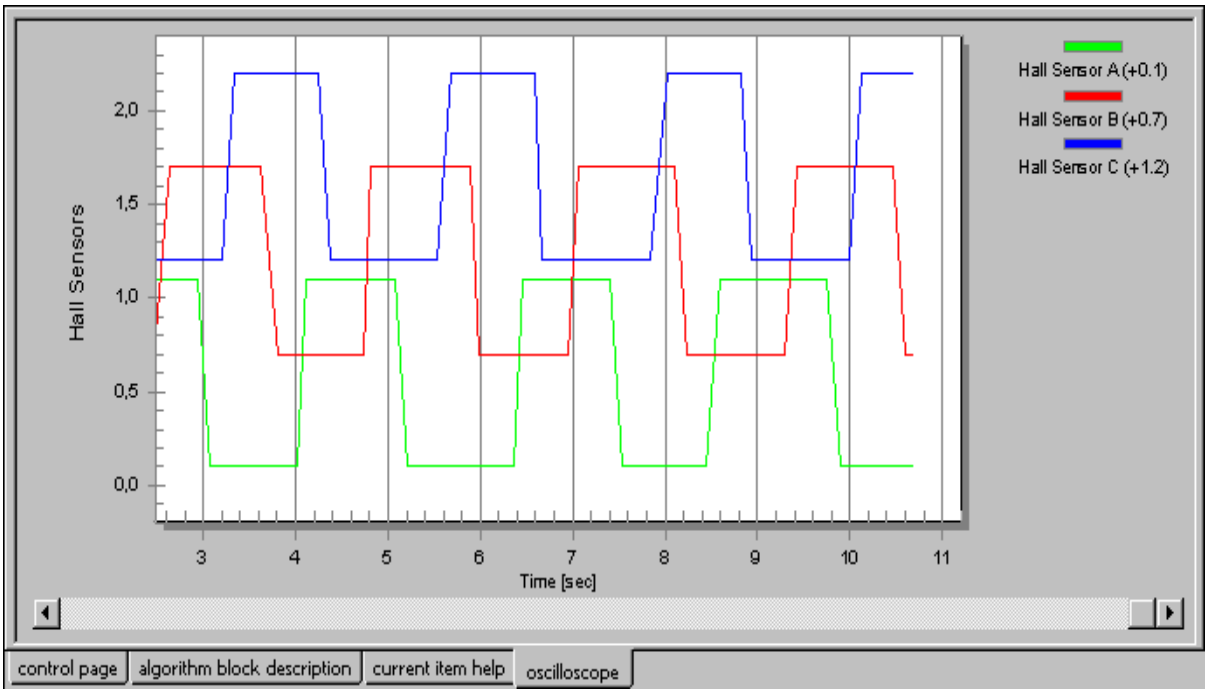

Figure 4-8. Basic Oscilloscope Chart

- To define a separate *Right axis* for the selected Y-block, select the number of *Right axis vars* that will use the right axis within the Y-block. The value of *Right axis vars* specifies the number of variables (counting from the bottom of the *Chart vars* list) that are assigned to the right axis within the Y-block.
- As with *Left axis*, specify *min*, *max*, *style* and *axis label* for the right axis in the selected Y block.

In **Figure 4-9**, we added the fourth variable to the Y-block and set *Right axis vars* to one. This means that the added variable (*Speed*) will be assigned to the Right axis labeled *RPM*.

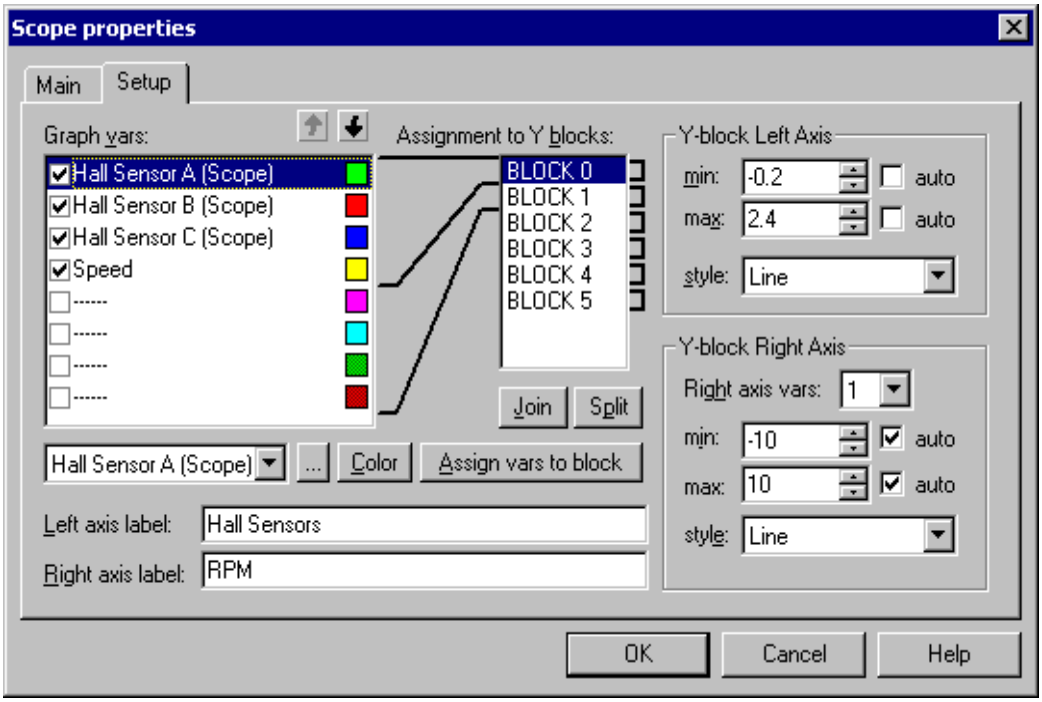

Figure 4-9. Fourth Variable Added

The resulting chart is displayed in **Figure 4-10**

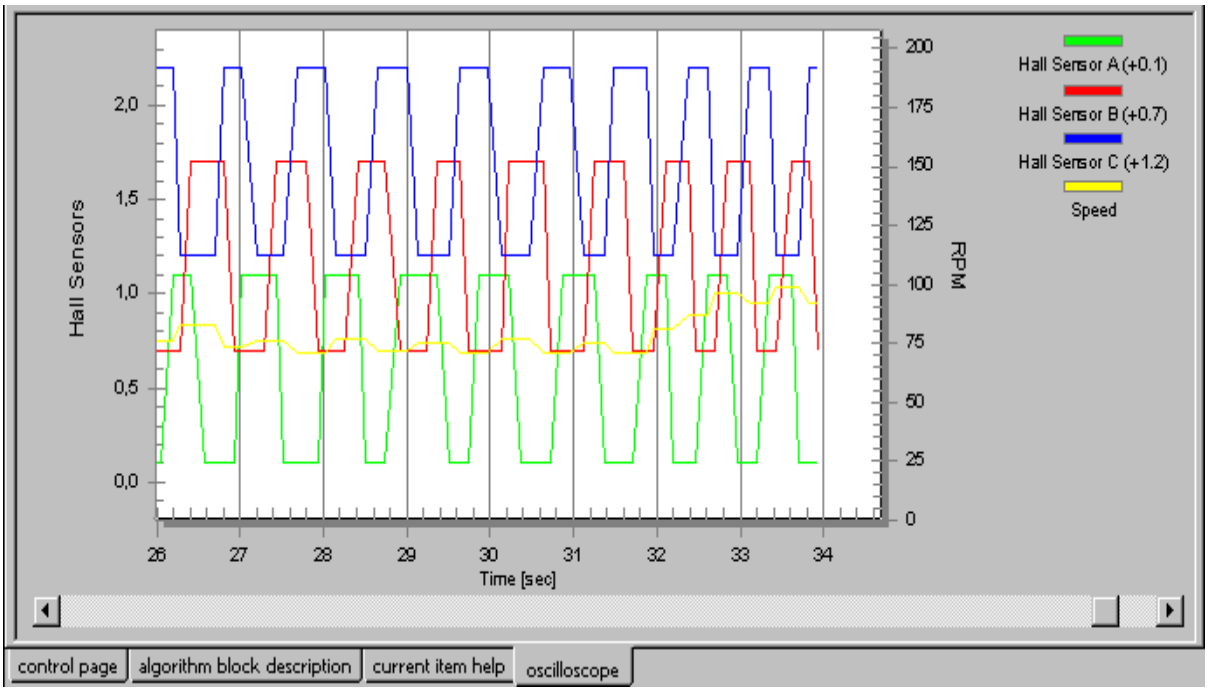

Figure 4-10. Resulting Oscilloscope Chart

Figure 4-11 shows the addition of a fifth variable, *Period Time*, put into a separate Y-block.

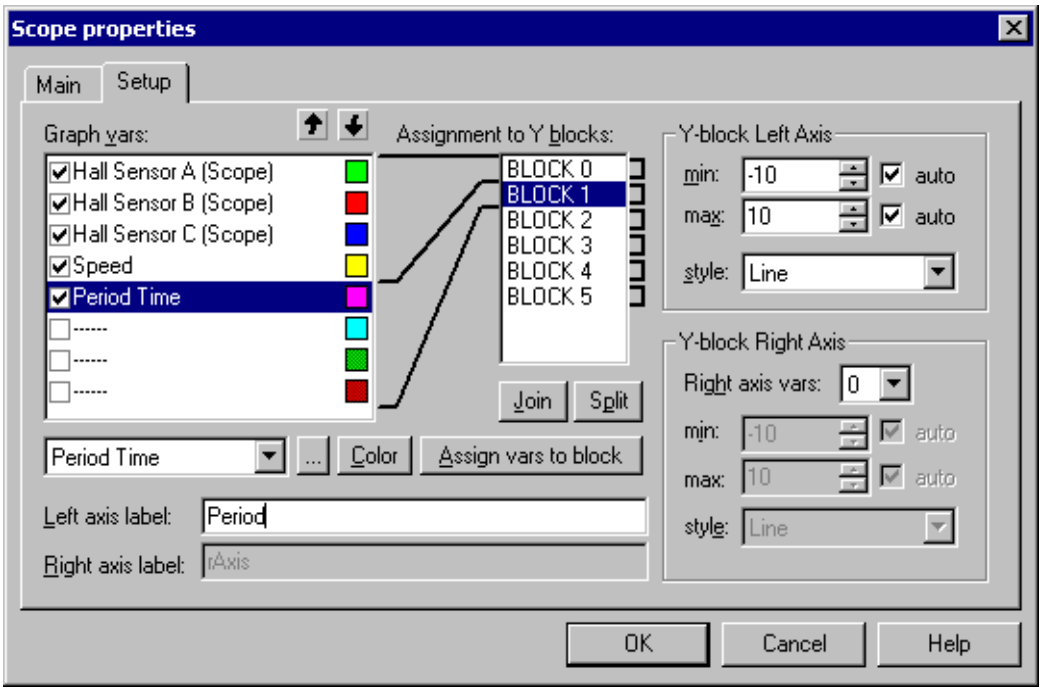

Figure 4-11. Fifth Variable Added in Separate Y-block

The resulting chart is displayed in **Figure 4-12**

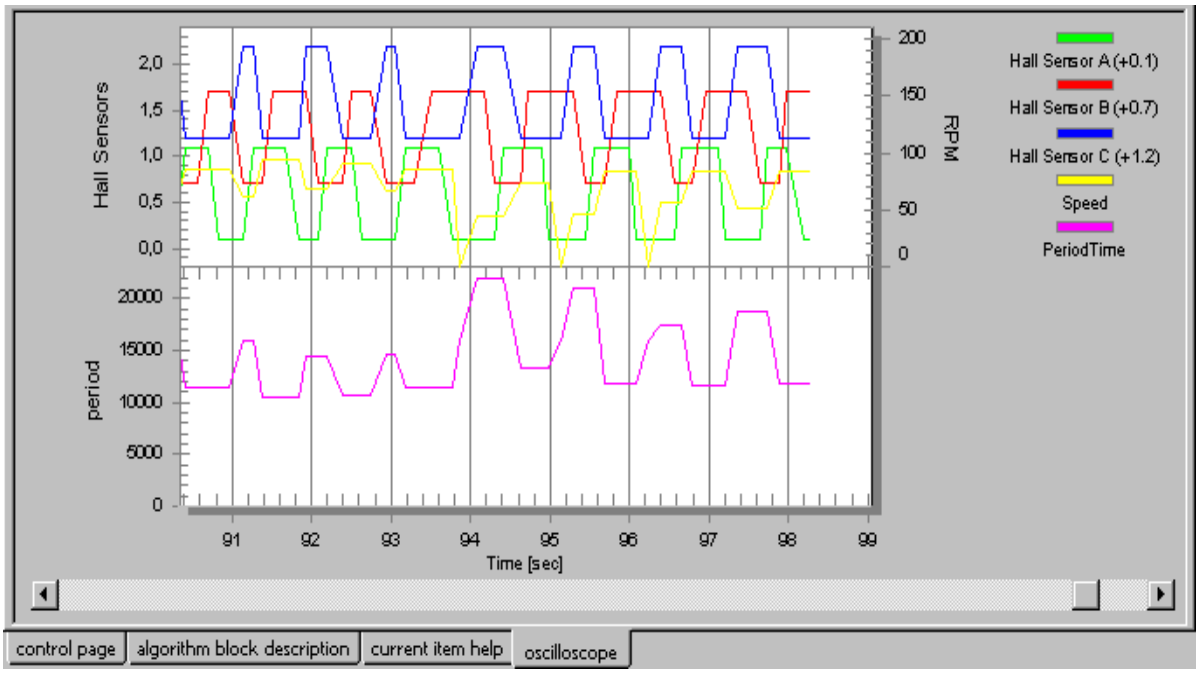

Figure 4-12. Resulting Oscilloscope Chart

- *Join* button = Several successive Y-blocks can be joined and set to be overlapped. The overlapped blocks then behave as a block with multiple left axes and, optionally, multiple right axes as well.

Figure 4-13 shows the joining of two Y-blocks defined in previous examples.

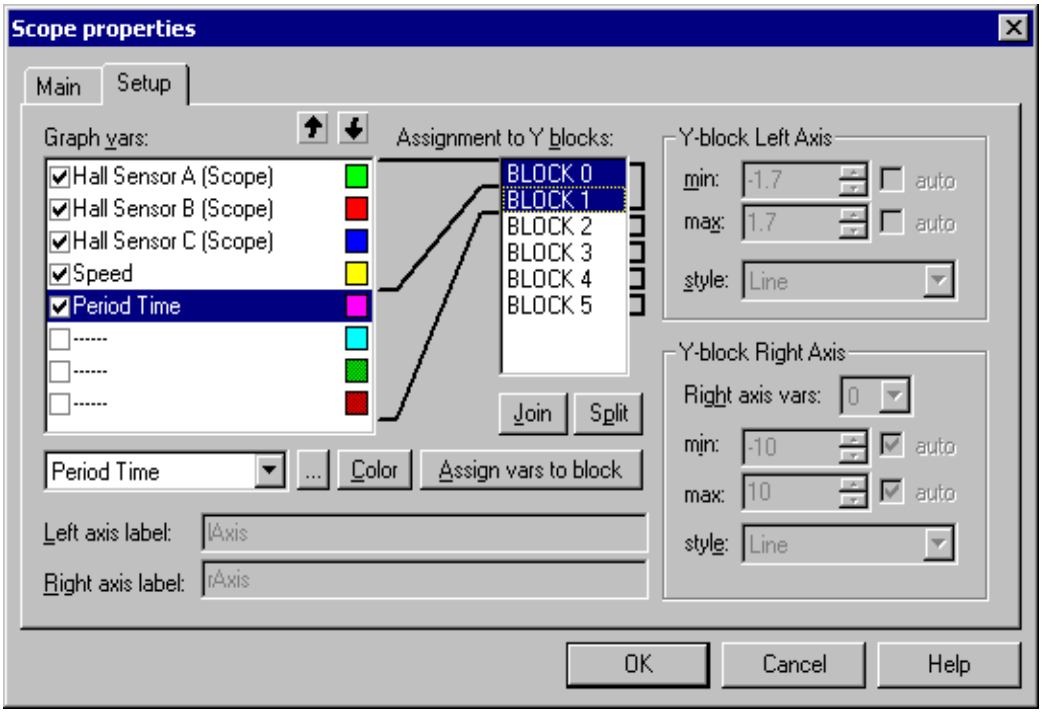

Figure 4-13. Two Y-blocks Joined

Figure 4-14 displays the Oscilloscope Chart that resulted when Y-blocks were set to overlap.

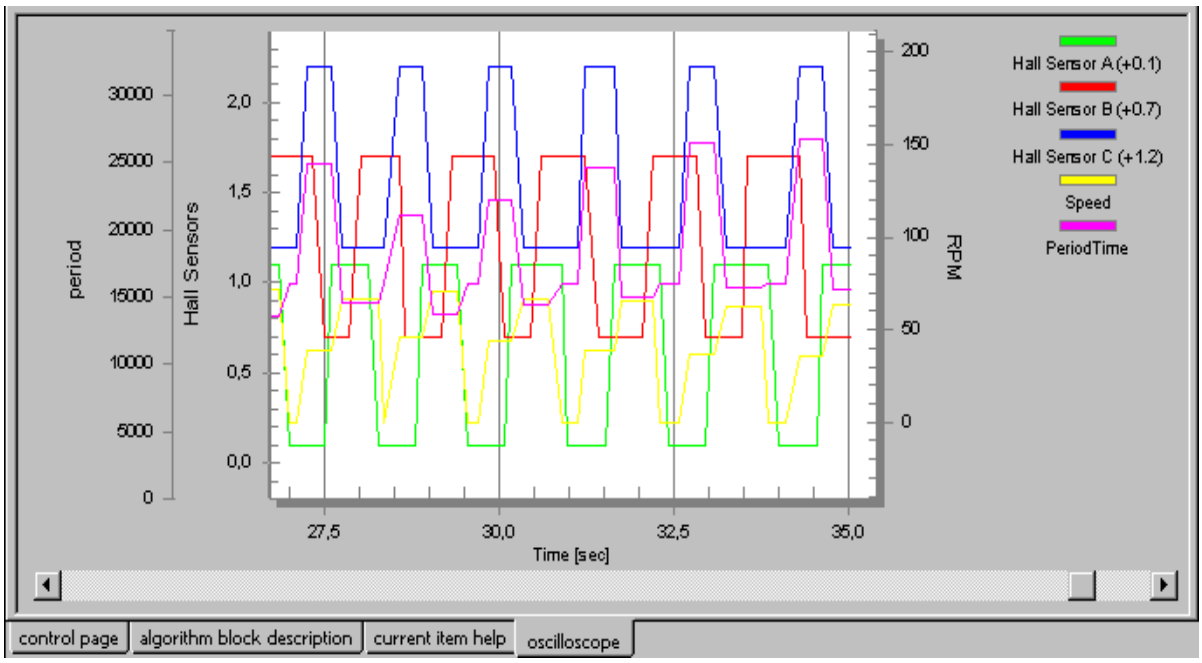

Figure 4-14. Resulting Oscilloscope Chart (Y-blocks Overlap)

4.1.1.3 Recorder

The *Recorder* item in the *Project Tree* structure defines a real-time recorder chart to be shown in the *Detail View* pane. While the Scope periodically reads variable values and plots them in real time, the Recorder is running on the target board, reads application variables, and sends them to the PC master software tool in a burst mode fashion. The recorder variables are continually sampled and stored into a circular buffer in the target board application. When the trigger event is detected by the PC master software in the board application, data samples are counted until the number of *Recorder samples* is reached. At this point, data is sent to the PC master software application. This mechanism enables use of much shorter sampling period and enables sampling and plotting very quick actions.

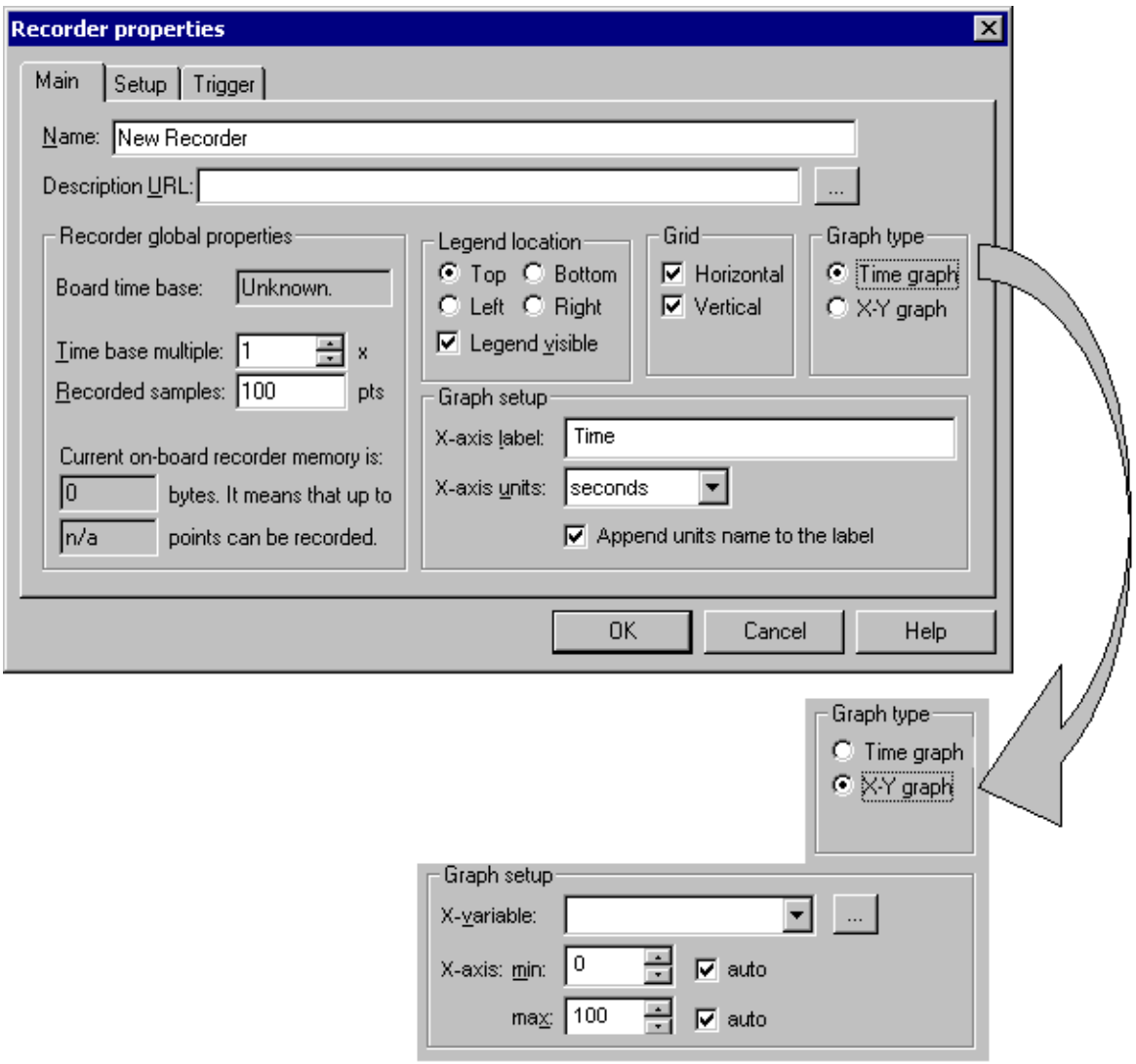

Figure 4-15. Recorder Properties - Main Page

The *Main* page contains the following user configuration items.

- *Name* = The name of the Recorder item that will be displayed in the *Project Tree*

- *Description URL* = Specify the document's URL or local path to a file to be shown in the *Detail View* pane under the *current item help* tab. This file may be created with any HTML editor, such as MS Front Page Express or Netscape Composer, and should explain the chart variables and settings to the user.
- *Recorder global properties* = Common properties for all Recorder variables
  - *Board time base* = A sampling period preset by the board application. In the SDK, the time base value can be adjusted by setting PC_MASTER_RECORDER_TIME_BASE in the *appconfig.h* file during the board application development.
  - *Time base multiple* = Sets an integer multiple of the time base to extend the sampling period used for recorder operation
  - *Recorded samples* = The number of samples buffered for one recorded subset
  - *On board recorder memory* = Displays the amount of on-board application memory allocated for recorder operation. Based on the memory size, recorded variables format, and the number of recorded variables, the maximum number of points which fit in the recorder's memory is calculated and displayed. The *Recorded samples* value set should be lower than this result.
- *Legend location* = Sets the visibility and location of the chart legend
- *Grid* = Chooses the horizontal and/or vertical grid lines to be displayed in the chart
- *Graph type* = Selects the mode of recorder operation
  - *Time graph* = A variable (*values* versus *time*) will be displayed in the chart
  - *X-Y graph* = Inter-variable dependencies (*value* versus *value*) will be displayed in the chart
- *Graph setup* (for *Time graph*):
  - *X-axis label* = Specify the name displayed for X axis
  - *X-axis units* = Selects the axis units
- *Graph setup* (for *X-Y graph*):
  - *X-variable* = Selects the variable whose values will be used for X axis values
  - *X-axis min* = Sets the lower limit value of the X axis (check the *auto* box to enable auto-scaling of the X axis)
  - *X-axis max* = Sets the upper limit value of the X axis (check the *auto* box to enable auto-scaling of the X axis)

The *Setup* page of the *Recorder properties* dialog, shown in **Figure 4-16**, looks exactly the same as the appropriate page of the *Scope properties* dialog. For more information about how to add variables to the recorder chart and how to set up the chart itself, see **Section 4.1.1.2**.

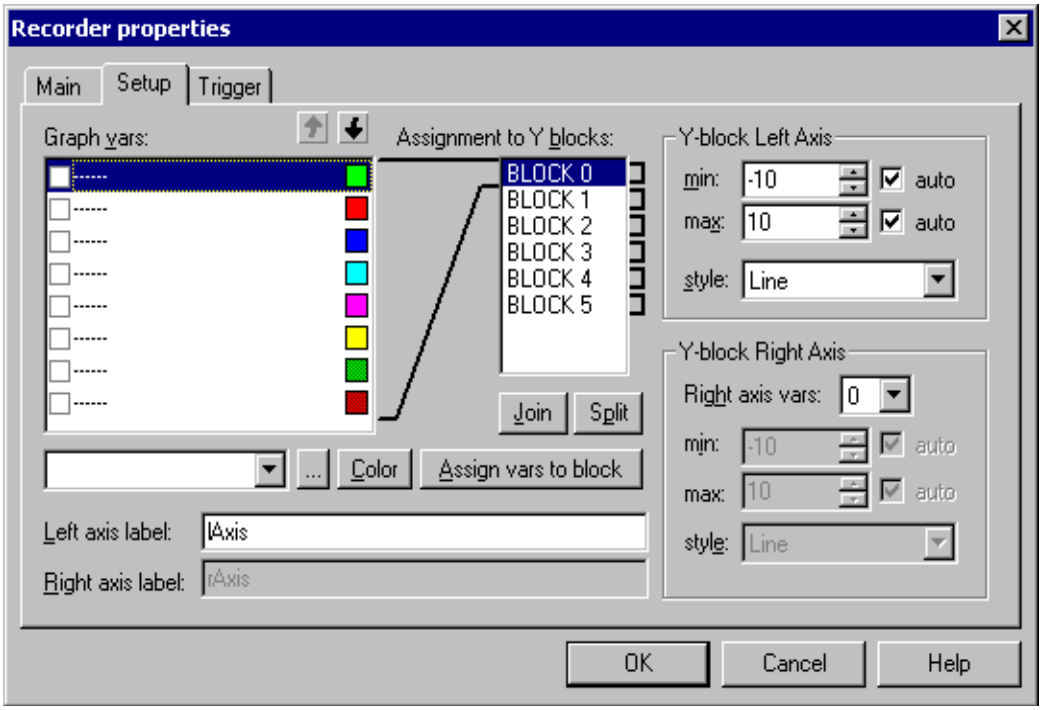

Figure 4-16. Recorder Properties - Setup Page

The *Trigger* page, shown in **Figure 4-17**, defines the Recorder start conditions. The trigger starts the Recorder when the *Trigger variable* exceeds the specified *Threshold value* with the selected type of slope, (positive = rising edge or negative = falling edge).

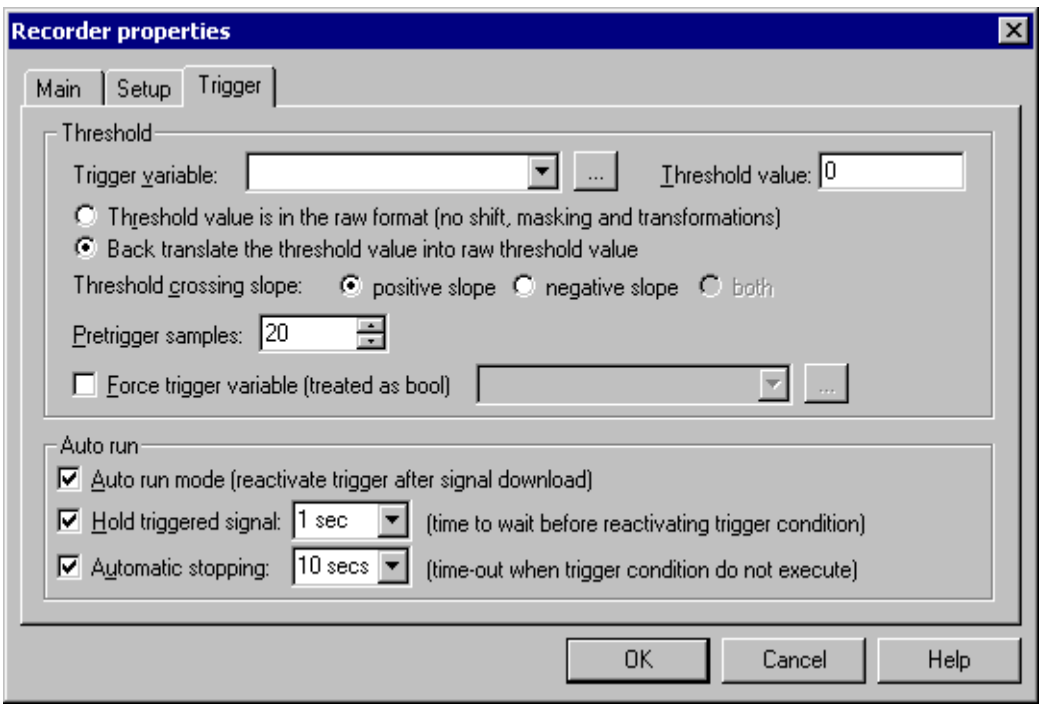

Figure 4-17. Recorder Properties - Trigger Page

- *Threshold* = Specify conditions for the trigger event
  - *Trigger variable* = Selects a variable to be tracked for the trigger event
  - *Threshold value* = The value of the variable being recorded that, when crossed, causes the trigger event. Specify a threshold value and select whether the value is in raw format (as in the board application) or whether it must be translated back into raw format (e.g., when a real-type transformation is defined for the variable and you want to apply an inverse transformation on the trigger value before it is set as threshold in the embedded application).
  - *Threshold crossing slope* = Selects the slope on which the threshold crossing is monitored
  - *Pretrigger samples* = Specify the number of samples to save and display prior to the trigger event
- *Auto run* = Specifies conditions for reactivating the trigger for repeated recording
  - *Auto run mode* = Select to enable repeated recording. After detecting the trigger event, filling the buffer and downloading the buffer data to PC master software, the trigger is automatically reactivated and new data is downloaded immediately after the next trigger event occurs.
  - *Hold triggered signal* = Check this box and specify how long to wait after one signal is displayed in the chart and before reactivating the trigger.
  - *Automatic stopping* = Check this box and specify the maximum time period for detecting the trigger event. If the event is not detected within the specified time, the sampling is unconditionally stopped, and actual buffer data is downloaded.

4.1.2 Detail View

The *Detail View* is a multipage pane. Availability of various pages in the *Detail View* depends on the type of item selected in the *Project Tree*.

The *control page*, when defined in project *Options* (**Section 5.4**), is available for all *Project Tree* items to allow the user to control the board at any time. The content of the *algorithm block description* page changes with the *Project Tree* item selected. When a Scope or Recorder item is selected in the *Project Tree*, the *current item help* and *oscilloscope/recorder* chart pages are also available.

4.1.2.1 Control Page

The *Control Page* is an HTML page created for board application control. It typically contains the scripts-enhanced form or forms which enable user-friendly control of the embedded application. The URL of the page or the path to the HTML file with a page source code can be specified in project *Options* dialog described in **Section 5.4**.

The control page for a simple motor control application is shown in **Figure 4-18**.

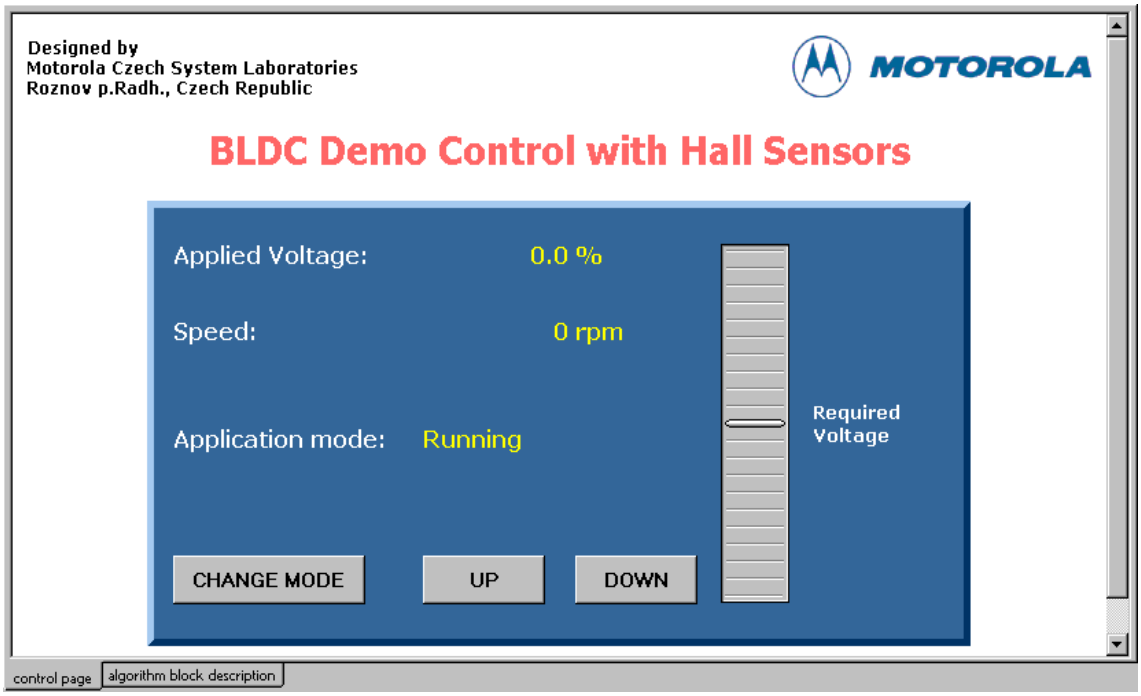

Figure 4-18. Example of Control Page

HTML scripting and other techniques to create active control pages are described in **Section 6.2**.

4.1.2.2 Algorithm Block Description

The *Algorithm Block Description* page in the *Detail View* pane is designated for placing an HTML page describing the selected block functionality. The URL or local path to the source file is specified in block item properties dialog, described in **Section 4.1.1.1**. This page is displayed when it is defined and when the user selects the appropriate block item or any of its child scope or recorder items.

As the standard HTML page, this page can also contain the scripts or other controls, but it is not a common practice.

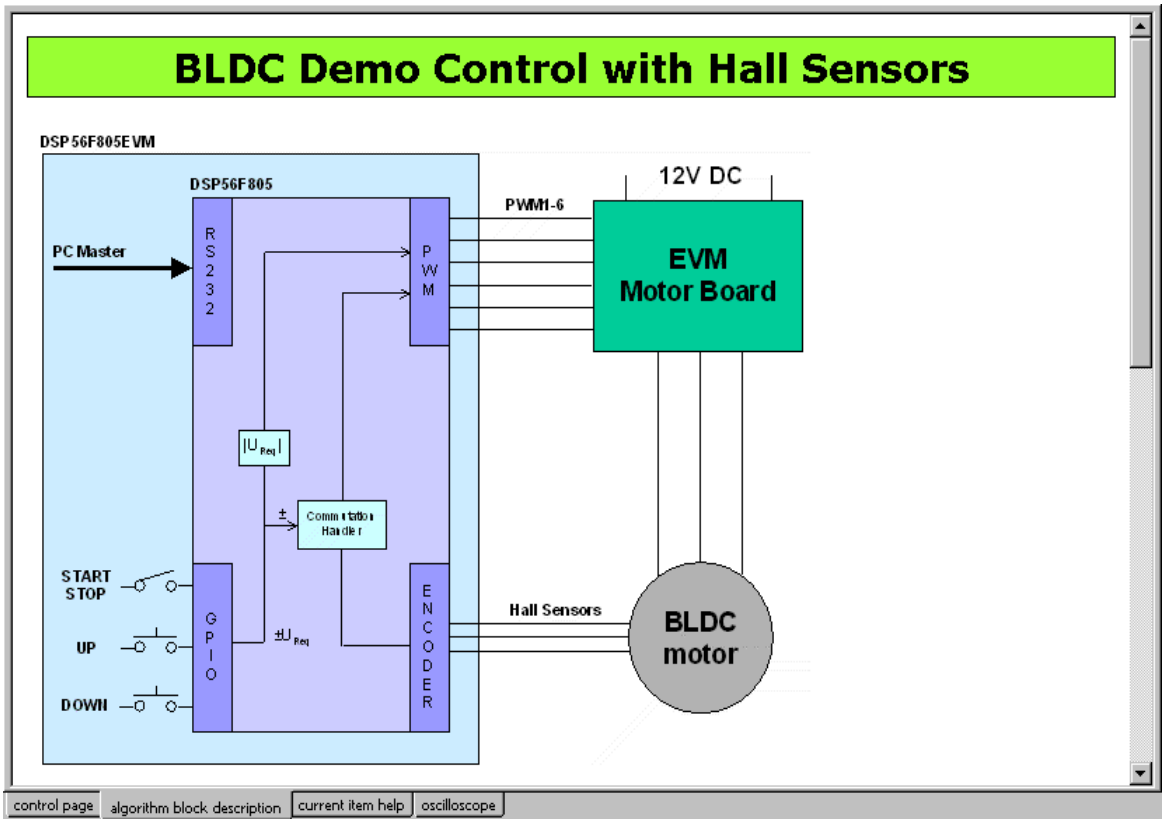

Figure 4-19. Example of an Algorithm Block Description Page

4.1.2.3 Current Item Help

The *Current item help* tab is designated for placing an HTML page describing the selected *Scope* or *Recorder*. This page should contain such information as definitions or use instructions and is specified as the Description URL (see **Figure 4-3**) for scope and recorder items. A simple example of such a page is shown in **Figure 4-20**.

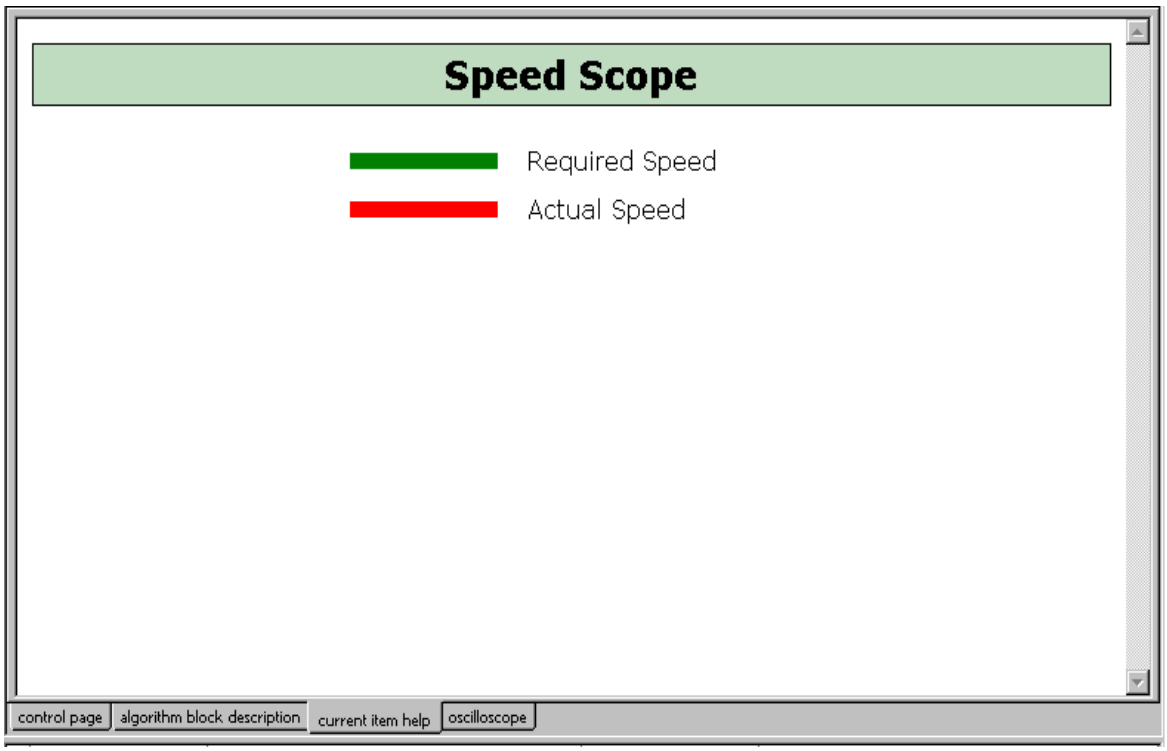

Figure 4-20. Example of Current Item Help Tab

4.1.2.4 Oscilloscope / Recorder

The *Oscilloscope page* in the *Detail View* pane contains the real-time chart representing tracked variables, as shown in **Figure 4-21**. Similarly, the *Recorder page* contains the chart created from the recorded data. Select *Scope* in the project tree to view the oscilloscope page or select *recorder* to view the recorder page.

See **Section 4.1.1.2** and **Section 4.1.1.3** for more information about setting up the oscilloscope or recorder views.

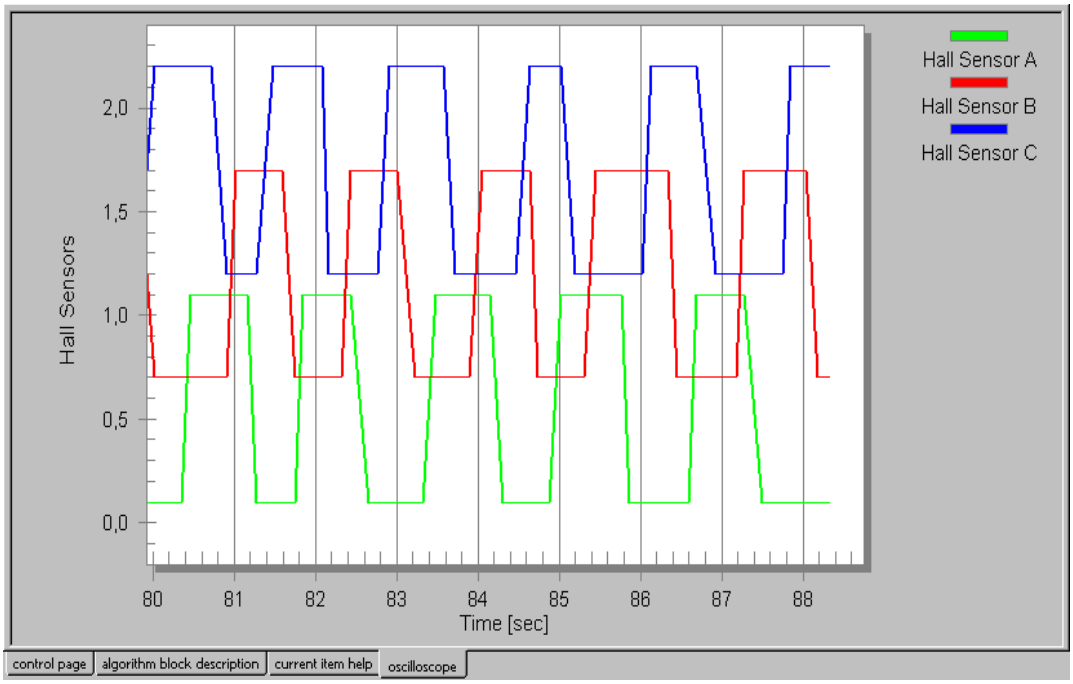

Figure 4-21. Example of Oscilloscope View

4.1.3 Watch-Grid

The *Watch-Grid* pane in the bottom of the application window contains the list of watch variables. The selection of watch variables and their graphical properties are defined separately for each project block. As a result, the *Watch-Grid* pane changes its contents each time a different project block is selected.

During the definition of the variable, the variable name, units, and number format are specified. Moreover, the *Watch bar* can be used to change the graphical look of the variable, including font type and size, foreground and background color, and alignment. Refer to **Section 4.6.2** for details.

Read-only variables can only be monitored. Variables with changes allowed (modifying enabled in the variable definition) can be altered from the *Watch-Grid* pane. Details about variables are found in **Section 4.2**.

4.2 Variables

The PC master software communicates with the board application via a well-defined communication protocol. This protocol supports sending commands from the PC application to the target board application and reading or writing its variables. All commands and variables used in the PC master software project must be specified within the project.

The *Variables list* dialog box, shown in **Figure 4-22**, can be opened by selecting the *Variables* item from the *Project* menu. It can also be opened from other project-development points, where it could be used to manage variables (e.g. *Scope Setup*).

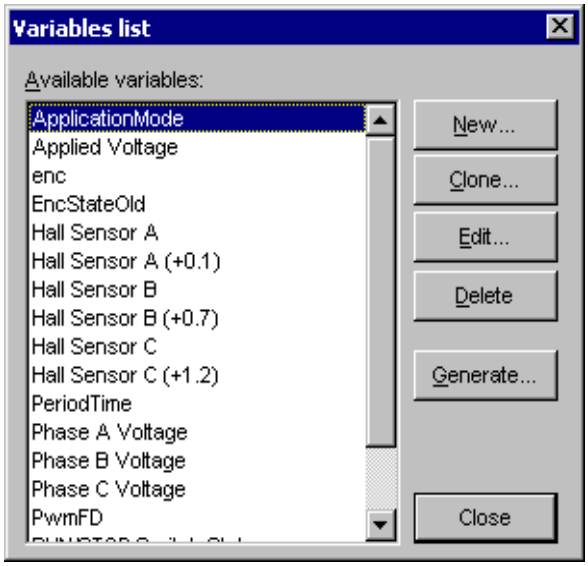

Figure 4-22. Variables list Dialog Box

To define a new variable, use the button *New*. This will open the *Variable* dialog box, where you can set the variable properties described in **Section 4.2.1**. When you need to create a copy of an existing variable, select the original variable and press the *Clone* button.

The *Edit* button opens the same *Variable* dialog box for changing the selected variable properties. After pressing the *Delete* button, the user is asked for confirmation of deletion and the selected variable will be deleted.

Generate button opens the interface for mass creation of variable objects based on the symbols loaded from an embedded application executable file. It is described in **Section 4.2.2**; loading of symbol files is described in **Section 5.2**.

4.2.1 Variable Settings

The *Variable* definition dialog has two tabs: *Definition*, shown in **Figure 4-23**, and *Modifying*, shown in **Figure 4-24**.

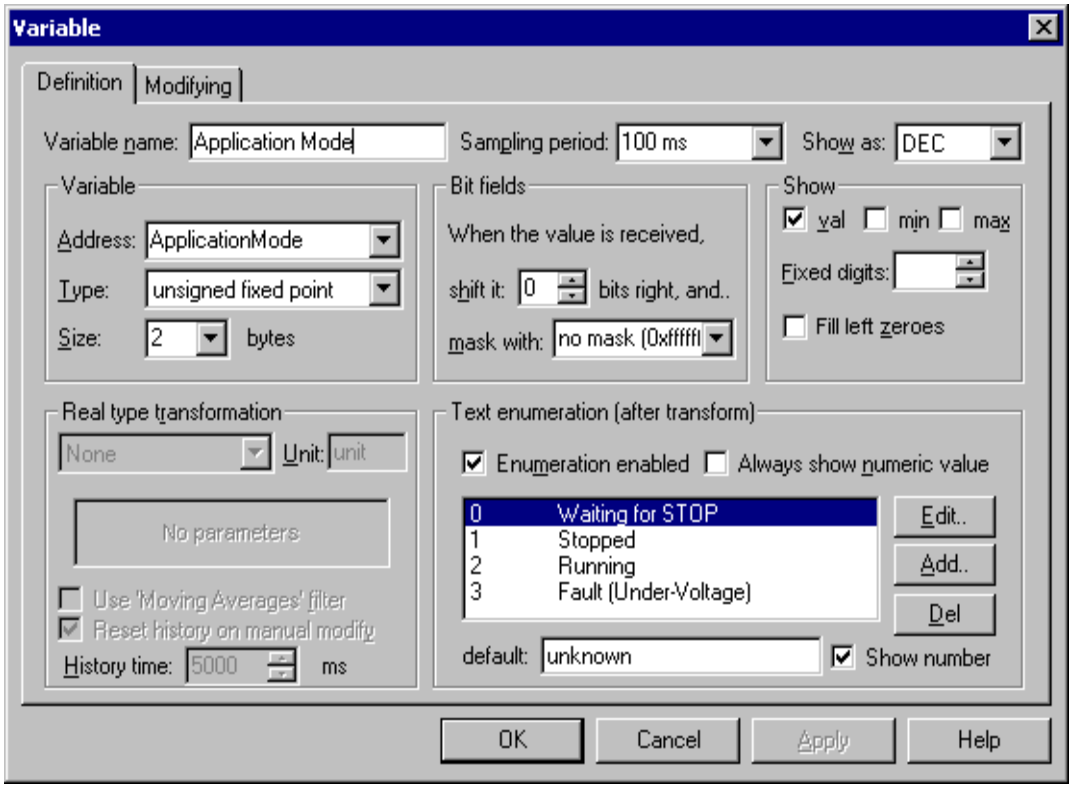

Figure 4-23. Variable Dialog Box - Definition Tab

On the *Definition* tab, you specify a general variable properties.

- *Variable name* = Specify the variable name as the variable identifier in the project
- *Sampling period* = The time period of reading the variable value from the board when the variable is displayed in the variable watch
- *Shows as* = A format in which the variable value is printed in the watch window. Select the proper format from the drop-down list (`DEC`, `HEX`, `BIN`, `ASCII` or `REAL`).
- *Variable* panel = Information about the variable as it is defined in the embedded application
  - *Address* = The physical address of the variable in the target application memory. Although you can type the direct hexadecimal value, it is recommended that you select a symbol name of the application variable from the drop-down list. A symbol table can be loaded directly from the embedded application executable (if it is in standard ELF/Dwarf1 format) or from the text-based MAP file generated by the linker. See **Section 5.2** for more information about loading the symbol tables from selected files.
  - *Type* = Select the variable type as it is defined in the target application (*unsigned fixed point, signed fixed point, floating point IEEE, fractional <-1,1), unsigned fractional <0,2)* or *string*)
  - *Size* = Specify the size of the variable as it is defined in target application
- *Bit fields* = The parameters for extracting the single bit or bit groups from a given variable
  - *Shift* = Specify the number of bits the received value is right-shifted before it is masked
  - *Mask* = Select or specify the mask value which is *AND*ed with the shifted value

    Using the *Shift* and *Mask* fields, you can extract any bit field from the received variable. For example: to extract the most significant bit from the 16-bit integer value, you would specify 15-bit shifting and one-bit mask (0x1).

- *Show* = According to the value display format selected in the *Show as* field, this set of parameters controls how the variable value is actually printed
  - *val, min, max* = Check the boxes if you want the variable watch to display the immediate variable value and/or the detected peak values. The peak values can be reset by right-clicking the variable entry in the watch window and selecting the menu command *Reset MIN/MAX*.
  - *Fixed digits* (for *Show as* set to `DEC`, `HEX` or `BIN`) = Prints numeric values left-padded by zeroes or spaces to a given number of digits
  - *Fixed digits* (for *Show as* set to `REAL`) = Prints floating-point numeric values with a constant number of digits after the decimal point
  - *Zero terminated* (for *Show as* set to string) = The string values are printed only to the first occurrence of zero character. For string values, you can also select whether to display unprintable characters as HEX numbers (or question marks) and a few other string-specific settings.
- *Real type transformation* = When the *Show as* format is set to `REAL`, you can define further post-processing numeric transformation, which is applied to the variable value.
  - Transformation type
    - *linear: ax + b* : Specify the *a* and *b* constants of the linear transformation *y = ax + b*. The '*a*' and '*b*' parameters can be specified as a numeric values or by the name of the project variables whose immediate value (last valid value) is then used as the parameter.
    - *linear two points* : If it is more convenient for you to specify the linear transformation by two points, rather than by the parameters *a* and *b*, fill in the two coordinate points *(x1, y1)* and *(x2, y2)*. As the parameter values, you can again specify the numeric values or variable names.
    - *hyp: d/(ax+b) + c* : Specify the parameters *a*, *b*, *c* and *d* of a hyperbolic transformation function.
  - *Unit* = The name of unit displayed in the variable watch
  - *Use "Moving Averages" filter* = When monitoring a noisy action, you might want to display the average value instead of the immediate one
  - *History time* = The time interval from which the average value is computed
- *Text enumeration* = This allows you to describe the meaning of certain variable values and assign the text label to each of them, which is then displayed in the variable watch together with or instead of the numeric value. Use *Edit*, *Add* and *Del* buttons to manage the look-up table with value to text label assignment.
  - *Default* = Specify the default text label, which is displayed when no matching text is found in the look-up table.

The *Modifying page*, shown in **Figure 4-24**, contains settings and restrictions for variable value modifications.

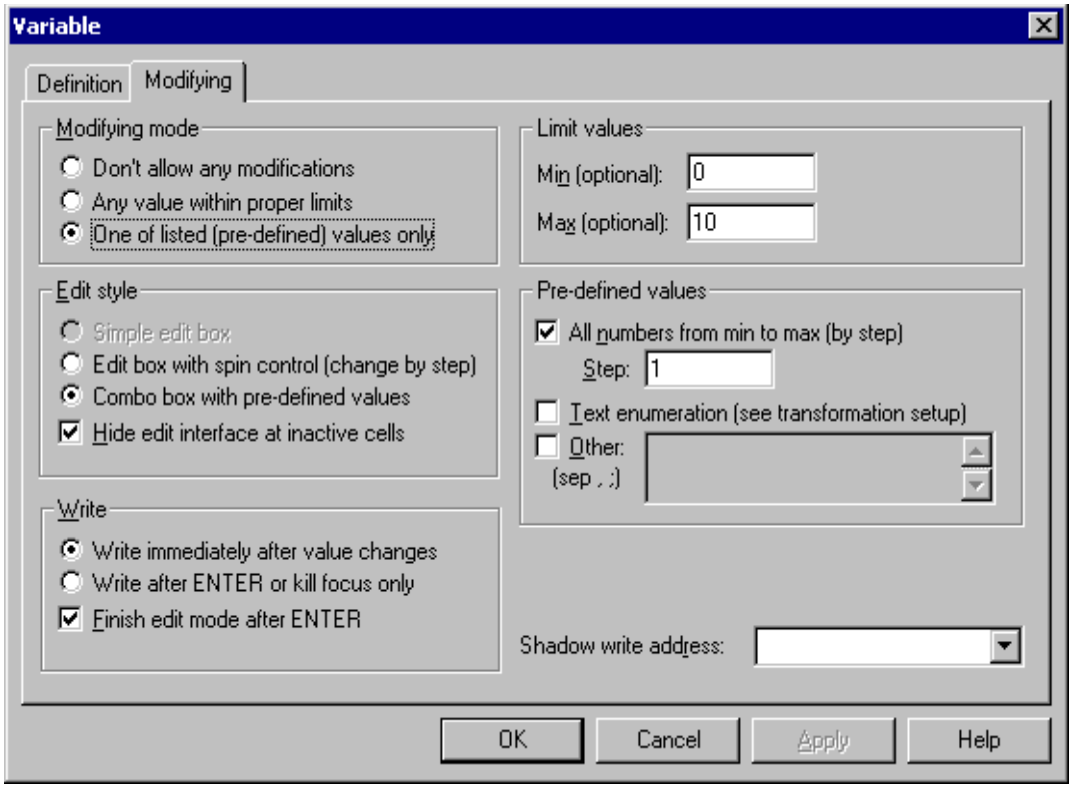

Figure 4-24. Variable Dialog Box - Modifying Tab

- *Modifying mode*
  - — *Don't allow any modifications* = The variable is read-only; all other settings on this page are disabled
  - — *Any value within proper limits* = You can specify the *Min* and/or *Max* value. The value the user enters in the watch window is then validated with the specified limits.
  - — *One of listed values only* = Once you have specified a list of values, only those values will be accepted in the watch window to be written. The acceptable values can be specified in the *Pre-defined values* group.
    - – *All numbers from min to max* = All the numbers from "*min*" to "*max*" by "*step*" will be treated as predefined values
    - – *Text enumeration* = Treat all values from the text enumeration look-up table as predefined
    - – *Other* = Specify any other predefined values (comma or semicolon separated)
- *Edit style* = Select the look of the edit interface for a given variable, which is displayed in the appropriate cell in the watch window grid
  - — *Edit box with spin control* = The variable value edit interface will be displayed in the form of an edit box with two spin arrows for incrementing and decrementing the value
  - — *Combo box with pre-defined values* = The variable value edit interface will be displayed in the form of a drop-down list box. The predefined values will be available in the list.
  - — *Hide edit interface at inactive cells* = The variable edit interface will be hidden when the appropriate cell in the watch grid looses a keyboard focus
- *Write* style = Specify exactly when the new variable value is actually sent to the board application.
  - — *Write immediately after each value changes* = The modified variable value will be sent to the

embedded application each time the user presses the spin arrow button or selects a new value in the drop-down list box.

— *Write after ENTER or kill focus only* = The modified variable value will not be sent to the embedded application until the user does not press the Enter key

4.2.2 Generating Variables

The *Generate* button in the variable list dialog (**Figure 4-25**) opens the *Generate variable* dialog box shown in **Figure 4-26**.

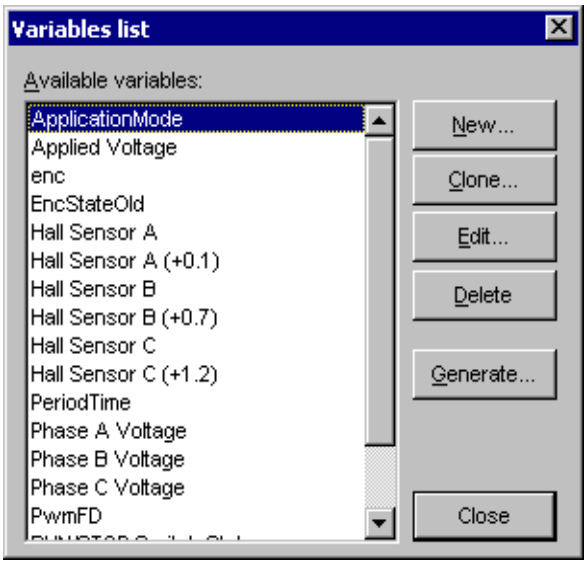

Figure 4-25. Project Variable List

In this dialog, the user can automatically generate the variable objects for the symbols loaded from an embedded application executable (ELF) or a linker MAP file (see **Section 5.2** for more information about symbol tables).

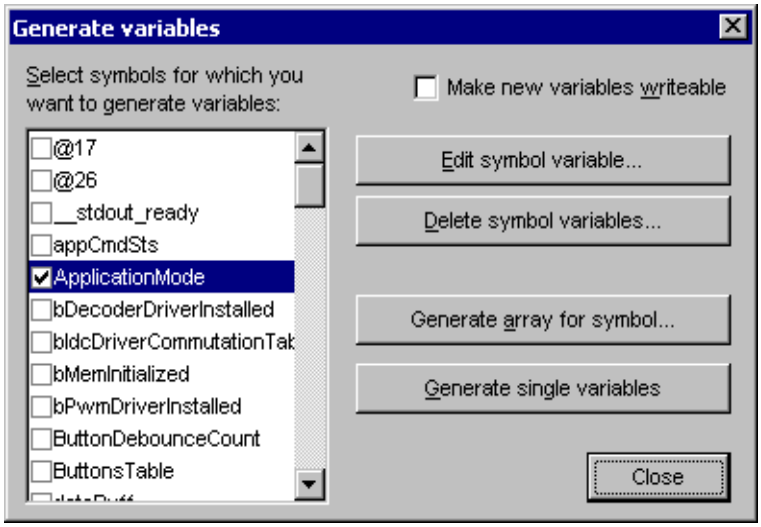

Figure 4-26. Generating Variables

The list in the dialog shows all the symbols available in the project as they were read from the current symbol file. The symbols for which the variable are already defined are marked with the check mark.

- *Edit symbol variable* = Press this button to edit a variable bound to the selected symbol, if any
- *Delete symbol variable* = Press to delete a variable bound to the selected symbol(s)
- *Generate single variables* = Generates a new variable with the same name as the symbol and with proper address, type and size settings. After creation, you can push the *Edit symbol variable* button to see and change other settings in the *Variable* dialog box
- *Generate array for symbol* = Enables you to generate a set of variables encapsulating the selected symbol and its successive locations (offsets)

4.3 Commands

The list of *Application commands* defined in the project can be opened by selecting the *Project* / *Commands* menu and is shown in **Figure 4-27**. You can use the buttons *New*, *Clone*, *Edit* and *Delete* to manage the list. It is very similar to variables management:

- *New* button = Creates a new application command
- *Edit* button = Edits properties of the selected application command
- *Clone* button = Creates a new command as a copy of the selected command
- *Delete* button = Deletes the selected command
- *Send* button = Opens the interface which enables the command to be sent to the embedded application

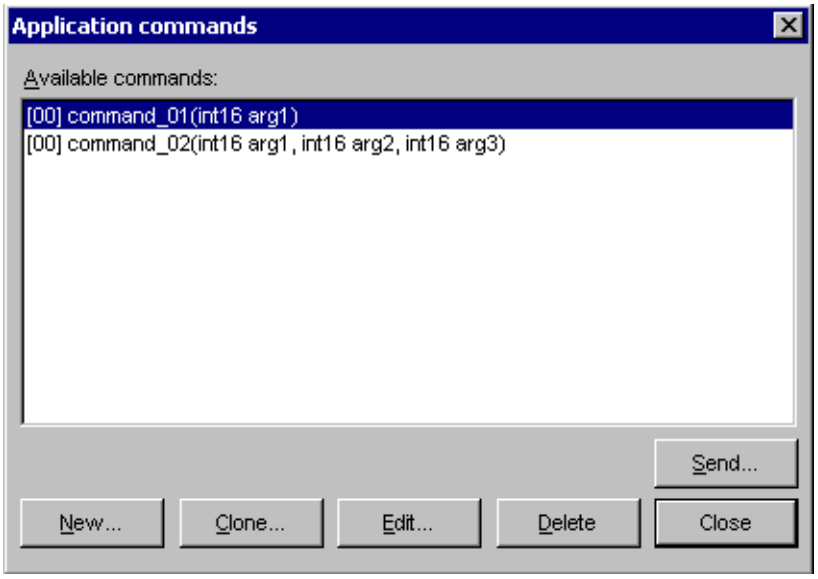

Figure 4-27. Project Application Commands

In the *Send application command* dialog box which follows after pressing the *Send* button, specify the command parameters (if any) and you can send the command to the embedded application. The dialog is shown in **Figure 4-28**. For each argument, you can define the help message, which is displayed in this dialog when typing the argument value as shown in **Figure 4-29**.

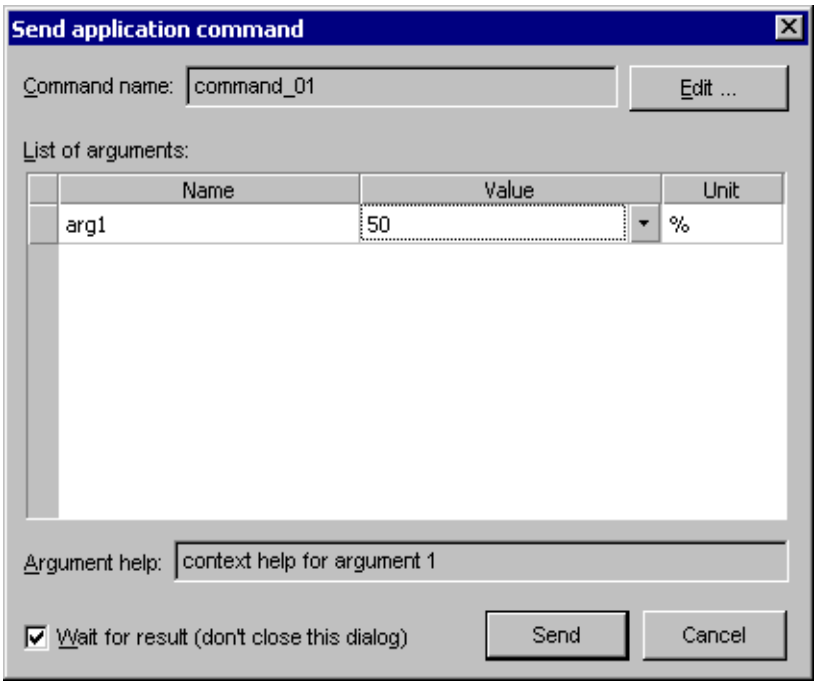

Figure 4-28. Sending Application Command

If you want to wait for the data to be returned from the board (a command result) without closing the dialog, check the *Wait for result* box. Before sending the command, you can review or edit the command definition.

When defining or editing the command, the *Application command* dialog box is opened. The first of three pages of that dialog is shown in **Figure 4-29**.

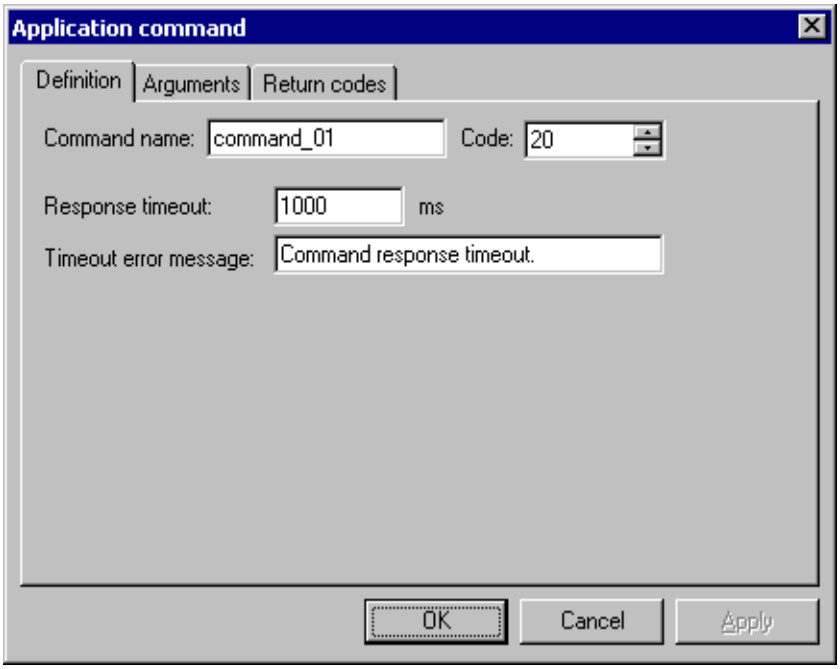

Figure 4-29. Application Command - Definition Tab

On the *Definition* tab, enter the *Command name* used in the project and specify the one-byte command *Code* which identifies the command in the target board application. The command codes and their purposes, as well as the command return codes and their purposes, come from the board application developer.

The *Response timeout* is the maximum time interval in milliseconds that the PC master software waits for response from the board application. If the the embedded application does not acknowledge the command and respond to it before this timeout occurs, the text entered into the *Timeout error message* field appears in alert window.

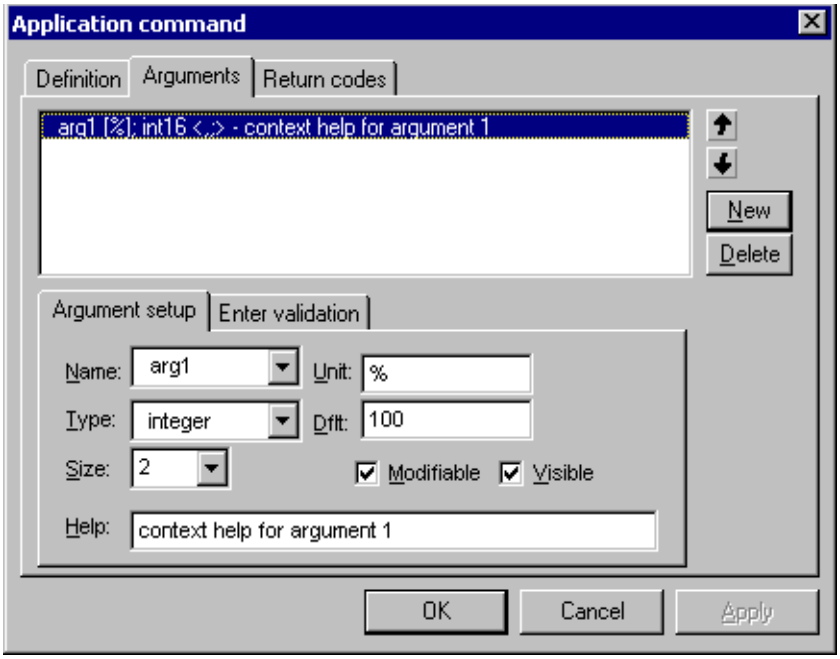

Figure 4-30. Application Command - Arguments Tab, page 1

The *Arguments* tab, shown in **Figure 4-30**, is used for definition of command arguments. Commands which do not have arguments will have an empty argument list. Commands can have arguments to pass a value to the target board application together with the command code.

Use the *New* button to create a new argument, the *Delete* button to delete an argument selected in the list, and the up and down *arrows* to change the arguments' order.

On the *Argument setup* sub-page, you define the selected argument parameters:

- *Name* = Specify the argument name as it should appear in the list and in the *Send application command*dialog when the user is prompted for argument values. You can also select the existing argument name from the drop-down list box.
- *Type* = Specify the argument's numeric value (*integer* or *floating point*)
- *Size* = Specify the argument's value size in bytes
- *Unit* = Specify any text which will be displayed as argument units. This text is not sent to the target application.
- *Dflt* = Enter the default value of the argument. This value will be set in the argument list of the *Send application command* dialog. If empty, the user must type the value any time he sends the command.
- *Modifiable* = Unless this box is checked, the user is not allowed to change the default argument value in the argument list of *Send application command* dialog.
- *Visible* = If this box is not checked, the argument will not be displayed in the argument list and its default value will always be sent to the target application.
- *Help* = Write any text information which will be shown in the *Send application command* dialog when the user will be prompted for an argument value.

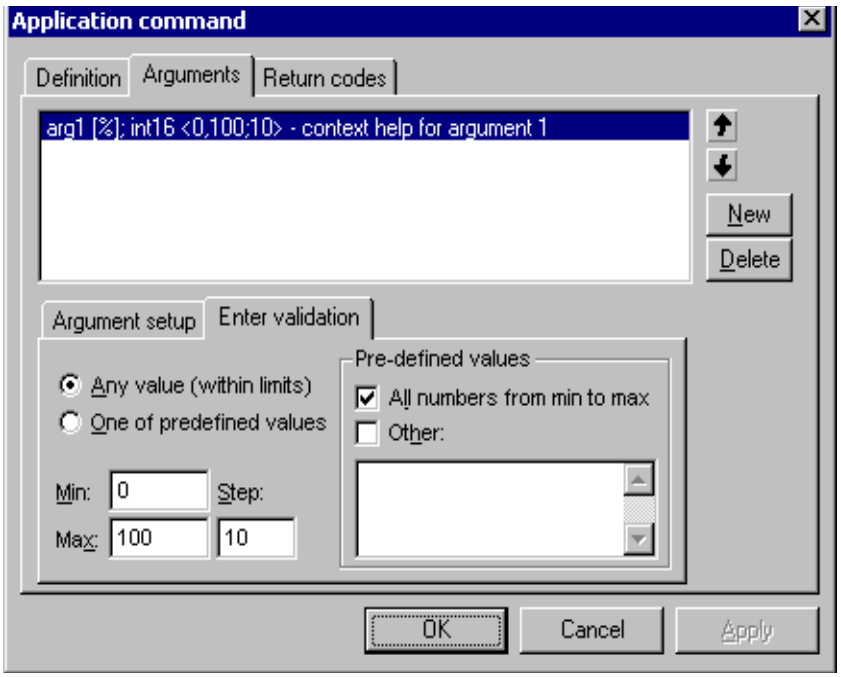

Figure 4-31. Application Command - Arguments Tab, page 2

On the *Enter validation* sub-page, shown in **Figure 4-31**, you define the validation criteria for the argument value:

- Specify which values are allowed for the argument
  - *Any value* = Any numeric value is allowed as the argument value. The value must be between *Min* and *Max* limits, if these are set.
  - *One of predefined values* = Only one of the values defined in the *Pre-defined values* fields can be supplied as an argument value
- Pre-defined values
  - All numbers from *min* to *max* = When this box is checked, all numbers between *Min* and *Max* limits, incremented by *Step*, are considered to be valid for the argument value
  - *Other* = Check this box and specify the list of other values (comma separated) which are valid as argument value

The *Return codes* page, shown in **Figure 4-32**, is used for specifying the command return code messages. To create a return code, enter the return code value in hexadecimal (0x00) or decimal form in the *code* field at the lower left of the page; enter the return code message in the next field; and click the *New* button. The return code item will appear in the list. Repeat to create all desired return codes. A *Message icon* may be assigned to each return code message from the panel at the lower right of the page. It will then appear in the message dialog, together with the text of message.

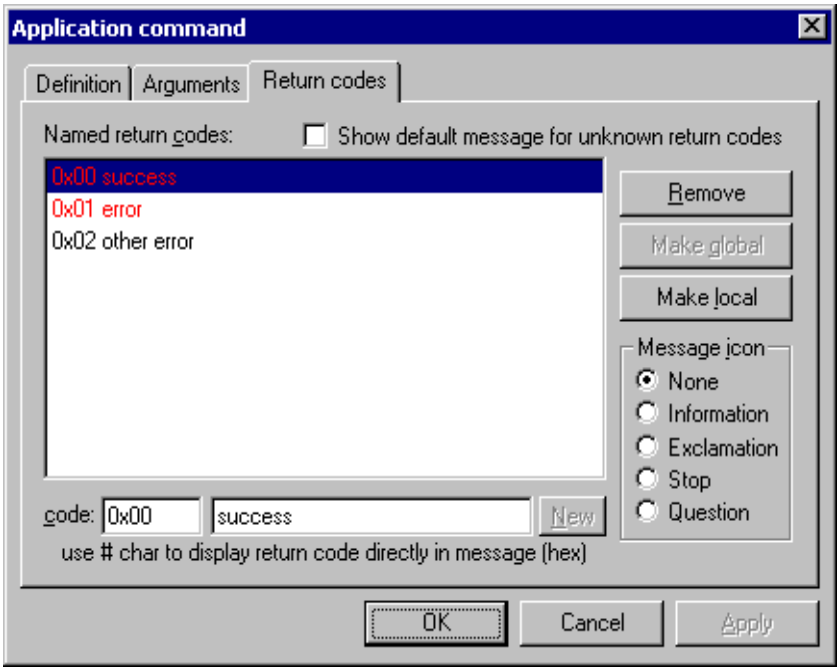

Figure 4-32. Application Command - Return Codes Tab

Return codes can be local or global. Local return codes apply to a single command, while global return codes are valid for all commands of the project. To switch between local and global validity, use the *Make local* and *Make global* buttons.

Check the *Show default messages for unknown return codes* box at the top of the page to pop up a standard message box with return code when an unlisted code returns from the board application.

4.4 Importing Project Files

When preparing your project, you may want to reuse the variables, commands, scope and recorder definitions or watch definitions you created in previous projects. Selecting the *File / Import* menu command opens a dialog in which you can select objects defined in different projects and import them to the current project.

The first dialog of the *Import* procedure is shown in **Figure 4-33**. After specifying the name of the original project file, you select and check the project tree items you wish to have in your current project. You can also select the target block item under which you want the imported items to be created.

Together with imported tree items, all referenced objects, such as variables or application commands, are also automatically imported. Using the switch radio-buttons below the lists, you can specify how the referenced objects are created:

- *Overwrite existing* = When there is an object, such as a variable, imported with a tree item, for example, an oscilloscope, the current project is searched for anobject of the same type (variable) and with the same name. If found, it is overwritten with the one imported.

- *Bind to existing* = If an object with the same name is found, it is not overwritten, but the imported tree item binds to it
- *Always create new* = All referenced objects are created, even if they already exist in the current document; in such a case, the name will be duplicated
- *Merge imported root item* = When importing the root item, it is possible to merge its variable watch definition with the watch of the root item in the current project. When this option is not checked, the root item will be imported and inserted as a standard block item.

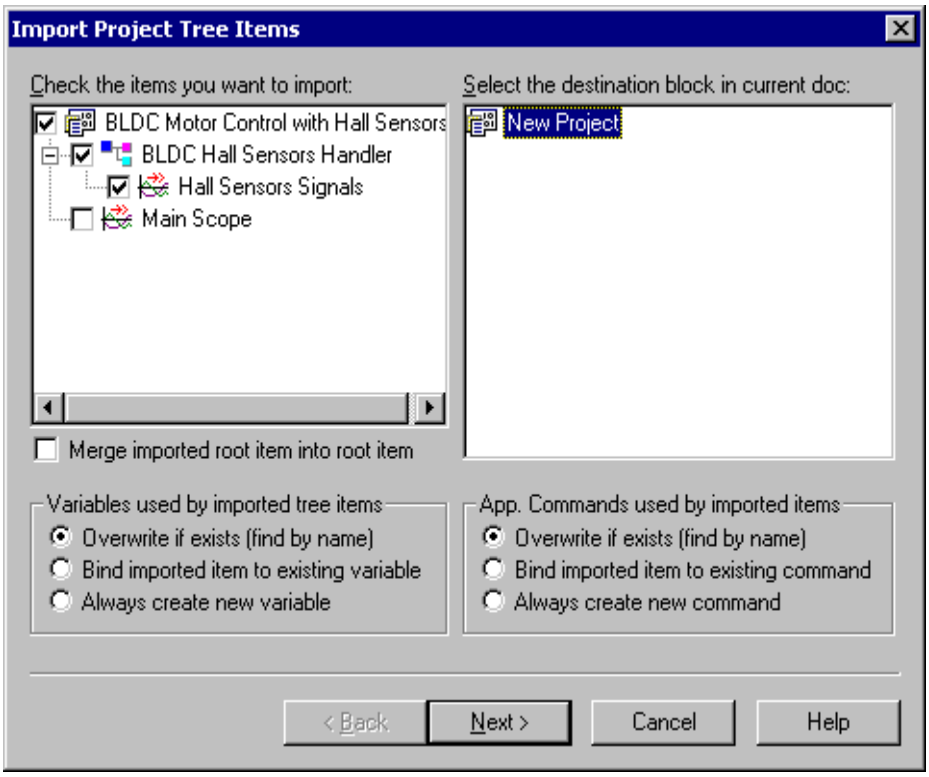

Figure 4-33. Import Project Tree Items

Pressing the *Next* button opens the second part of the *Import* procedure, where you can select additional objects to be imported. The second dialog is shown in **Figure 4-34**. The three check-box lists contain the objects found in the source project file:

- *Variables* = Put a check mark by each variable you want to import. You don't need to import variables which are referenced from the tree items selected in previous dialog from **Figure 4-33**; such variables are always unconditionally imported.
- *App. commands* = Put a check mark by each application command definitions you want to import. As with variable objects, you don't need to check commands which are referenced from the block tree items selected in previous dialog.
  - *Global return messages* = If this box is checked, the application commands return codes and messages are imported from the source document.
  - *Overwrite existing* = The existing return codes are overwritten with those being imported.
- *Stimulators* = Put a check mark by each variable stimulator you want to import.

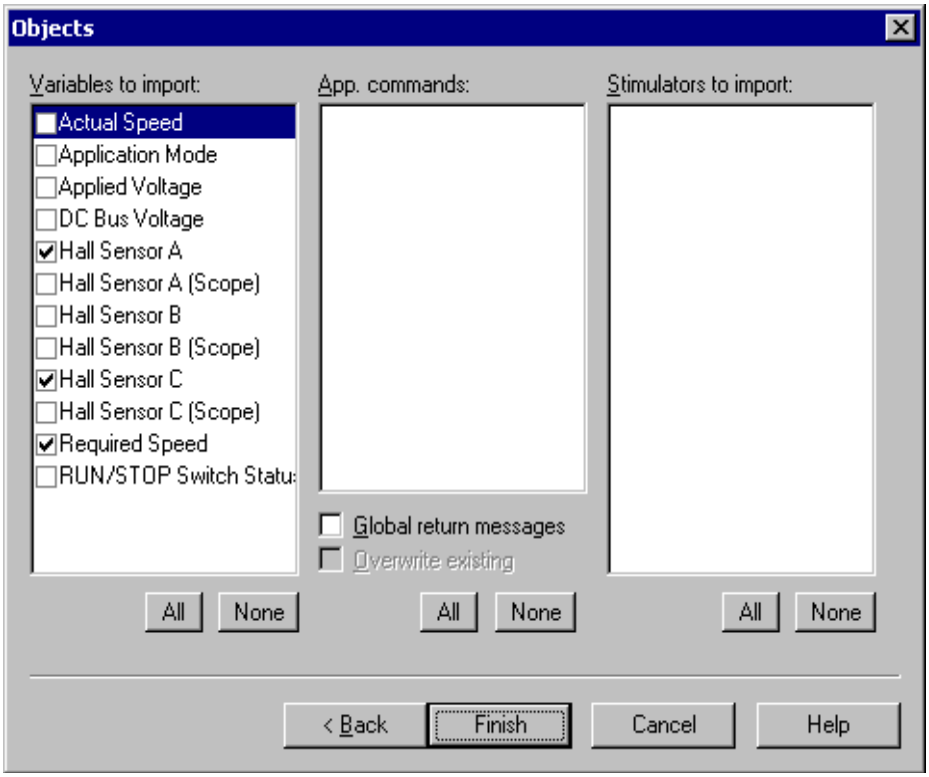

Figure 4-34. Import Project Objects

4.5 Menu description

4.5.1 File Menu

Selecting *New Project* creates a new empty project, while *Open Project* opens an existing project file, which has the extension *.pmp*.

Import Wizard enables you to import selected objects (*Project Tree* items, variables, commands, stimulators) from an existing project (*.pmp* file) into the current project.

Save Project saves the open project into the current file, while *Save As* allows users to specify a new filename for the current project.

Stop Communication pauses the communication with the board and unlocks the communication port. When the communication is released from the paused state, the symbol file is automatically checked for changes. If a difference is found, the user can choose to reload the new version of the file.

Print prints the content of current window, when possible. Currently, printing is supported only for HTML description and control pages. *Print Setup* opens the standard *Print Setup* dialog.

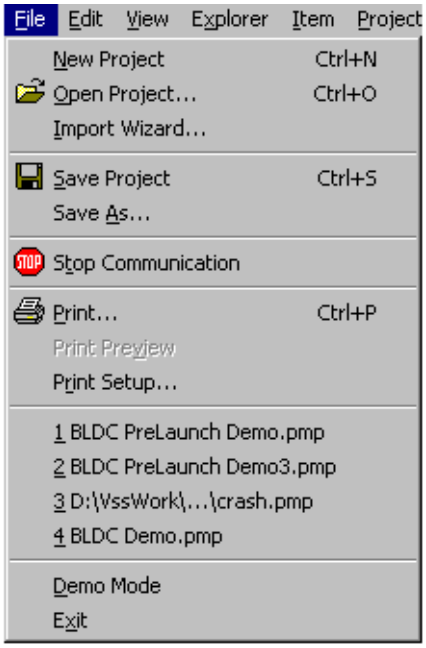

Figure 4-35. File Menu

The list of the most recently loaded projects: Selecting one of these items loads the specified project, just like using the *Open Project* command.

Demo Mode is a switch to enter or leave the PC master software application demo mode. While in *Demo Mode*, you cannot modify any important project and PC master software settings.

Exit exits the application.

4.5.2 Edit Menu

Edit menu contains standard clipboard manipulation commands (*Cut*, *Copy* and *Paste*).

Copy Special is enabled when the *Oscilloscope* or *Recorder* graph is active. The command enables saving the graph image to the clipboard or to a file in a different format or size. The set-up dialog is shown in **Figure 4-36**.

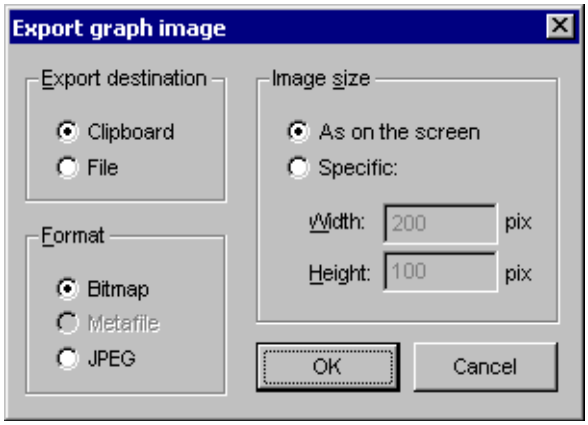

Figure 4-36. Export graph image Dialog

4.5.3 View Menu

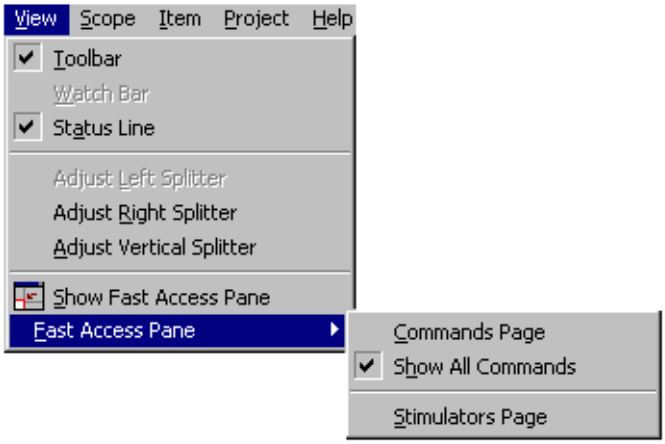

Figure 4-37. View Menu

In the *View* menu, you can select whether to show or hide the *Toolbar*, the *Watch Bar* and the *Status Line*.

Another feature of the *View* menu is the ability to adjust the size of individual panes without using a mouse. Use *Adjust Left Splitter* for changing the height of the *Fast Access* pane in the *Tree* pane. *Adjust Right Splitter* can be used for changing the height of the *Detail View* pane (HTML pages and charts). Finally, *Adjust Vertical Splitter* changes the width of the *Project Tree* pane.

Show Fast Access Pane switches the display of the *Fast Access* pane, a small helper window located at the bottom-left part of the application window. It contains two pages: the *Commands Page*, which lists currently-defined application commands and the *Stimulators Page*, which lists all stimulator objects. When the menu item *Show all project commands* is checked, the PC master software is forced to show all application commands defined in the project. The content of the *commands* page is then independent of which block is currently selected from the *Project Tree*. Otherwise, only commands selected for displaying in the current tree context are displayed; see **Figure 4-5** and **Section 4.1.1.1** for more information.

4.5.4 Explorer Menu

The *Explorer* sub-menu is available when an HTML page (*Control* page, *Block Description* page or *Chart Variables Info*) is displayed in the *Detail View*. When a *Chart View* page is displayed in the *Detail View*, then the *Explorer* menu item is replaced by the *Scope* or *Recorder* sub-menu.

Items *Back*, *Forward* and *Refresh* represent commands for the Internet Explorer window embedded in the PC master software. They are used to move through previously-visited pages and to refresh the page contents.

Fonts sets the font size for the current Internet Explorer window.

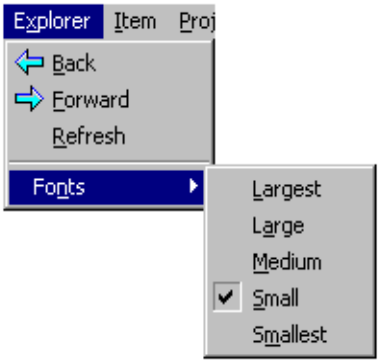

Figure 4-38. Explorer Menu

4.5.5 Scope Menu

The *Stop Scrolling* and *Stop Data* commands cause the PC master software to stop moving (rolling) the oscilloscope chart and to enter a mode in which the chart can be zoomed and the data series can be examined with the data cursor.

The difference between these two commands is that *Stop Scrolling* stops the picture, but allows incoming data to be appended to the end of the chart, while *Stop Data* also stops the incoming data.

Export Picture opens the *Export graph* image dialog, where you can specify the picture format and export the picture to the clipboard or to a file.

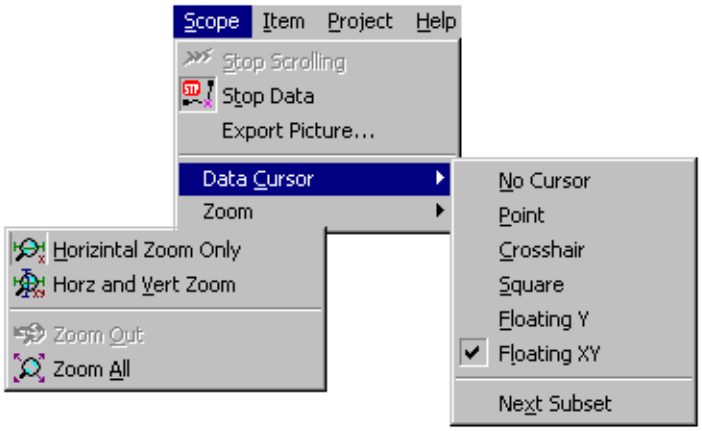

Figure 4-39. Scope Menu

Data cursor enables you to select the style of data cursor for examining the chart values. The *Next Subset* sub-item selects the next chart line to be examined.

The *Zoom* submenu consists of zooming modes and commands. The *Horizontal Zoom Only* mode allows “x-axis only” zooming; *Horz and Vert Zoom* allows free rectangle zooming.

4.5.6 Item Menu

The content of this sub-menu depends on the selected object within the PC master software application window. The same menu is displayed when you click the right mouse button on this object. The menu usually consists of the *Delete* item for deleting the selected object and the *Properties* item for opening an object properties dialog.

4.5.7 Project Menu

Variables opens the Variables list dialog box, where you can manage all project variables at once.

Commands opens the Application commands dialog box, where you can manage all project commands.

The *Reload Map File* command updates the physical addresses of board application variables from the *.map* file.

Options opens the multipage dialog box for all project option settings.

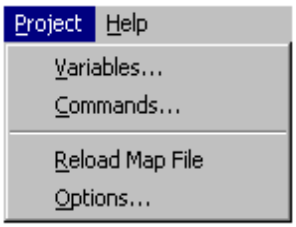

Figure 4-40. Project Menu

4.5.8 Help Menu

About PC Master opens a dialog box with a variety of information, including the application version, build, and copyright.

4.6 Toolbars description

4.6.1 Toolbar

The *toolbar* allows quick access to some of the menu commands.

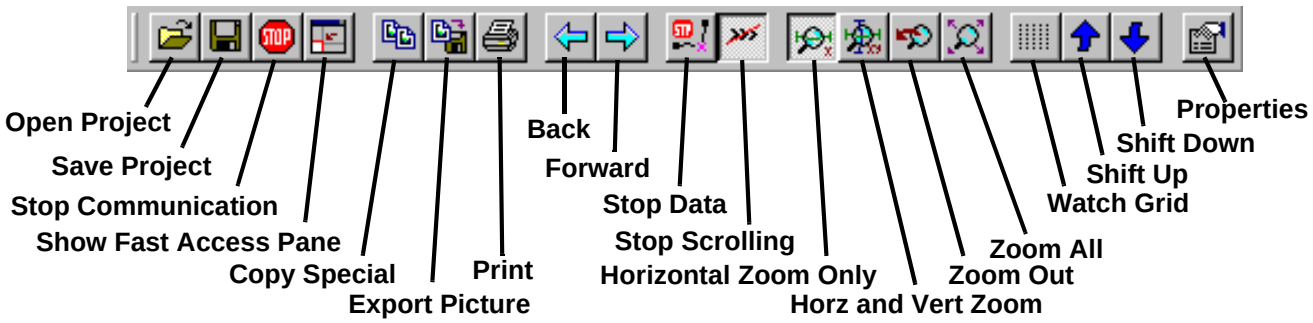

Figure 4-41. Toolbar

4.6.2 Watch Bar

The *Watch Bar* is available when the *Watch-Grid* pane has the focus and *Watch Pane* is selected from the *View* menu. It contains buttons with which you can change various graphical attributes of variables.

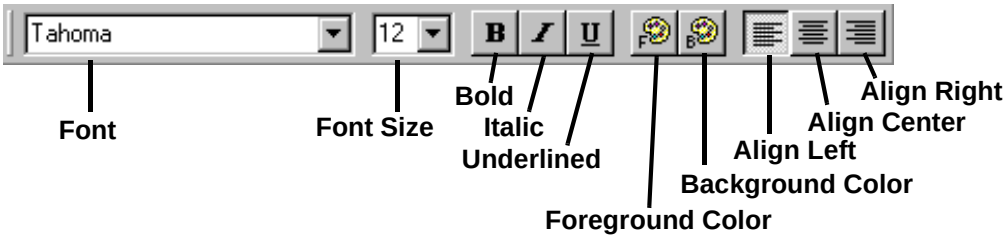

Figure 4-42. Watch Bar

Chapter 5
Project Options

To set application and project options, use the *Options* dialog, which can be activated by selecting *Options* from the *Project* menu. The dialog consists of several pages, each dedicated to a separate control or interface topic.

5.1 Communication

To set up the parameters related to communication between PC master software application and target board, select the first tab, as shown in **Figure 5-1**.

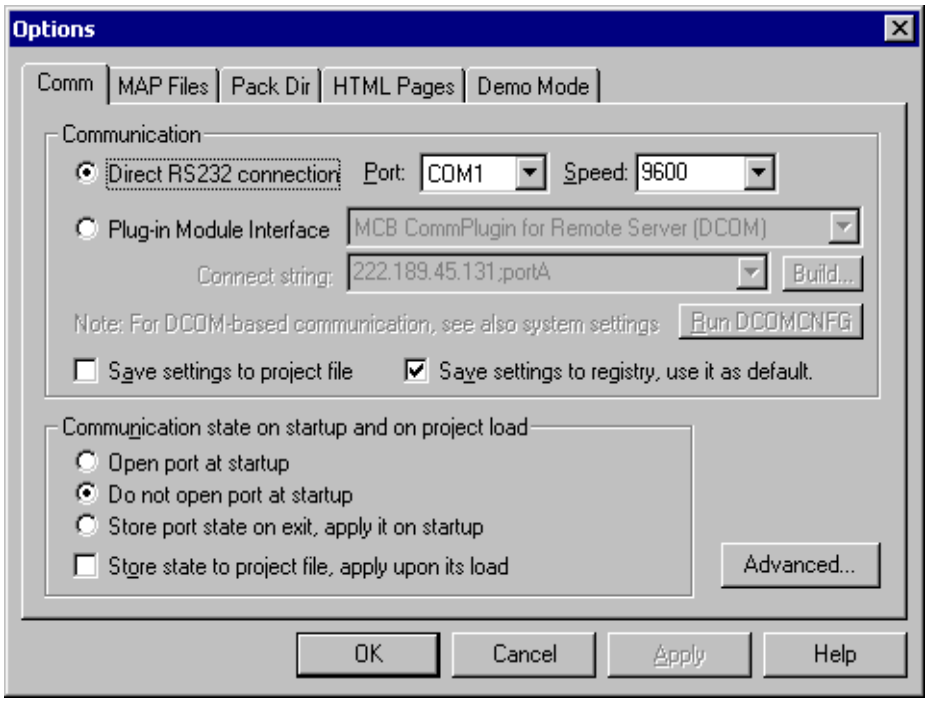

Figure 5-1. Communication Options

- *Communication*
  - — *Direct RS232 connection* = The standard serial interface (3-wire RS232 cable) will be used to connect to the target application
    - – *Port* = Select the serial port on which the board is accessed

  - *Speed* = Select or specify the communication baud rate. This speed must correspond with the embedded application settings.

— *Plug-in Module Interface* = The communication with an embedded device will be handled by a separate (custom) plug-in module. The module must conform to the Microsoft COM specification and must be registered in the system registry database. See protocol and communication library documentation for more information.
  - The drop-down list contains all plug-in modules known (registered) in the local system.
  - *Connect string* = The plug-in module configuration is saved in string form, internally called a “connect string”. You can directly specify this string, or you can press the *Build* button to open the module-specific configuration dialog.
  - *Run DCOMCNFG* = If the custom communication module internally uses Microsoft DCOM technology, the DCOM should be properly set up at the system level. This button launches the DCOM system configuration utility.

— *Save settings to project file* = If checked, the communication parameters will be stored within the project file during the next *Save Project* operation. When the project is next loaded, the communication settings will be restored and used instead of the defaults.

— *Save settings to registry* = If checked, the settings are stored in the local computer registry database. These settings will be used as default when the application is started next time.

Note: There are two communication plug-in modules distributed with the PC master software and installed together with it. Both modules handle communication with the remote server--the first uses DCOM to connect to the remote server and the second uses the standard HTTP text-based protocol.

When using the DCOM-based communication to remote server, the security issues must be considered when connecting over the network. Both sides of the communication must have the DCOM properly set up on the system level by running the DCOMCNFG utility. Also, the remote server must be properly installed on the remote side. If there are problems connecting to the remote computer, contact your local network administrator.

- *Communication state* = By selecting one of three options, you can set whether the communication port will be opened or not when the application is started.
  - *Open port at start-up*
  - *Do not open port at start-up*
  - *Store port status on exit, apply it on start-up*

— *Store status to project file, apply it on its load* = If checked, the state of the communication port will be saved to a project file during the next *Save Project* operation. The state will be then restored the next time the project is loaded.

5.2 Symbol Files

When defining project variables, it is often useful to specify the physical address of the target memory location as the symbol name instead of direct hexadecimal value. The symbol table can be loaded directly from the embedded application executable if it is the standard ELF or Dwarf1debugging format. For other cases, the text MAP file generated by the linker may be loaded and parsed for symbol information.

In the project, you can specify multiple files which contain the symbol table. Later, you can switch between different symbol tables from different files by selecting the menu command *Project / Select Symbol File*. For example, with the DSP embedded application tested on an EVM board, you can have two symbol files specified in the project--one for code running from RAM memory and the second for code running from Flash.

Figure 5-2 shows the *MAP File* tab in the project *Options* dialog.

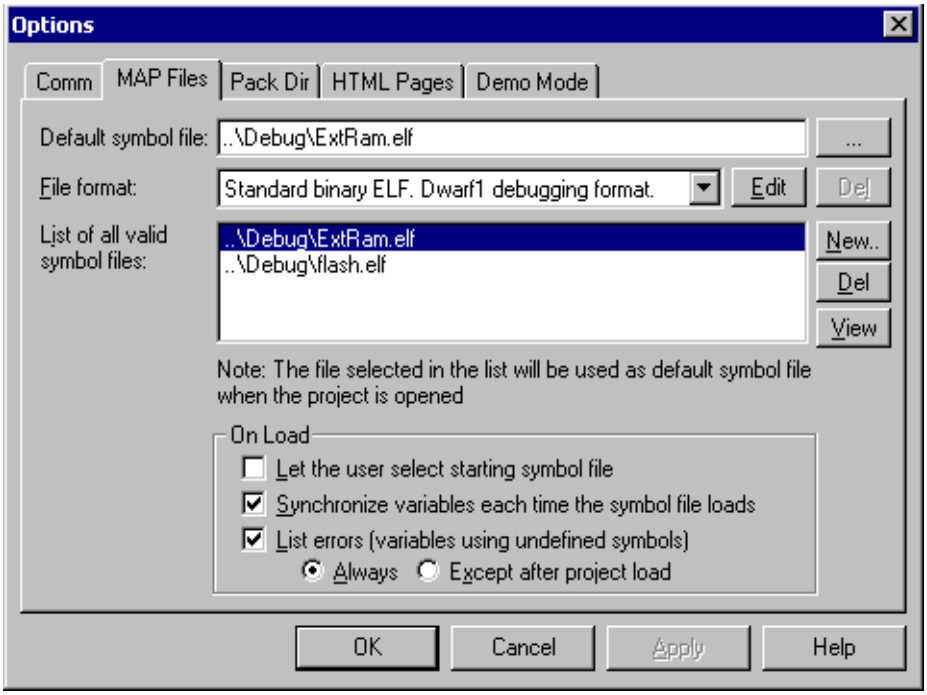

Figure 5-2. Symbol Files Options

- *Default symbol file* = Specify the name of the symbol file, which will be loaded by default. You can press the browse button (...) to locate the file in the standard open file dialog.
- *File format* = Select the format of the file
  - — *Standard binary ELF* = Choose this option for an ELF file
  - — *Hiware MAP File 509* = Choose this option for HiWare Smart Linker v5.0.9
  - — *Define new Regular Expression-based parser* = Choose this to define a new text file parser based on regular expression pattern matching; see **Section 5.2.1**
- *List of valid symbol files* = The list of symbol files defined in the project. Use *New* and *Del* buttons to manage the list.
- *View* = View the symbol table parsed from the selected file

Note: The paths to symbol files may be specified in several forms. It can be the absolute path on the local disk, the path relative to the directory where the project file is stored or the path relative to the current “pack” directory. The latter is the most suitable for project deployment to other computers; see **Section 5.3** for more information.

- *On Load* panel options control the actions performed with symbol files after the project is loaded:
  - — *Let the user select the symbol file* = When checked, the user is prompted to select the initial symbol file when the project is loaded.
  - — *Synchronize variables each time the map file loads* = When checked, the variables are

automatically updated by new symbol addresses after the project is loaded

— *List errors* = When this box is checked, all the variables using the symbols missing in the loaded symbol table are listed in a special dialog box. The user is able to edit the corrupted variables or select another symbol file.

5.2.1 Regular Expression-based MAP File Parser

When your compiler or linker does not support ELF output format and the MAP file cannot be parsed by the built-in HiWare 5.0.9 parser, you must describe the internal MAP file structure by using the regular expression pattern.

It would be out of the scope of this document to describe the theory of regular expression pattern matching, but a simple example might help to understand the strength of this technology. In the example, we will define the parser of *xMap* files generated by Metrowerks Compiler and Linker for the Motorola DSP56800 processor family.

The example of the *xMap* line, which describes the global symbol `length` might look like the following:

```
00002320 00000001 .bss    Flength(bsp.lib pcmasterdrv.o  )
00002321 00000001 .bss    Fpos(bsp.lib pcmasterdrv.o  )
```

First, we must describe the line format by the regular expression pattern. The following pattern

```
[0-9a-fA-F]+\s+[0-9a-fA-F]+\s+\S+\s+F\w+
```

could be read as follows:

- first, there are one or more hexadecimal digits: `[0-9a-fA-F]+`
- followed by one or more spaces (or another white characters): `\s+`
- followed again by one or more hexadecimal digits: `[0-9a-fA-F]+`
- followed again by one or more white characters: `\s+`
- followed by a set of any non-white characters: `\S+`
- followed by one or more white characters again: `\s+`
- followed by the character F
- followed by one or more alphanumeric (word) characters: `\w+`

To identify the sub-patterns which describe the symbol parameters, we put them in the round parentheses:

```
([0-9a-fA-F]+)\s+([0-9a-fA-F]+)\s+\S+\s+F(\w+)
```

Now we must identify the sub-pattern indexes for individual symbol values:

- The first sub-pattern corresponds with the symbol address (index 1)
- The next sub-pattern corresponds with the symbol size (index 2)
- The next sub-pattern corresponds with the symbol name (index 3)

Supply all the parameters prepared above in the regular expression dialog as shown in **Figure 5-3**.

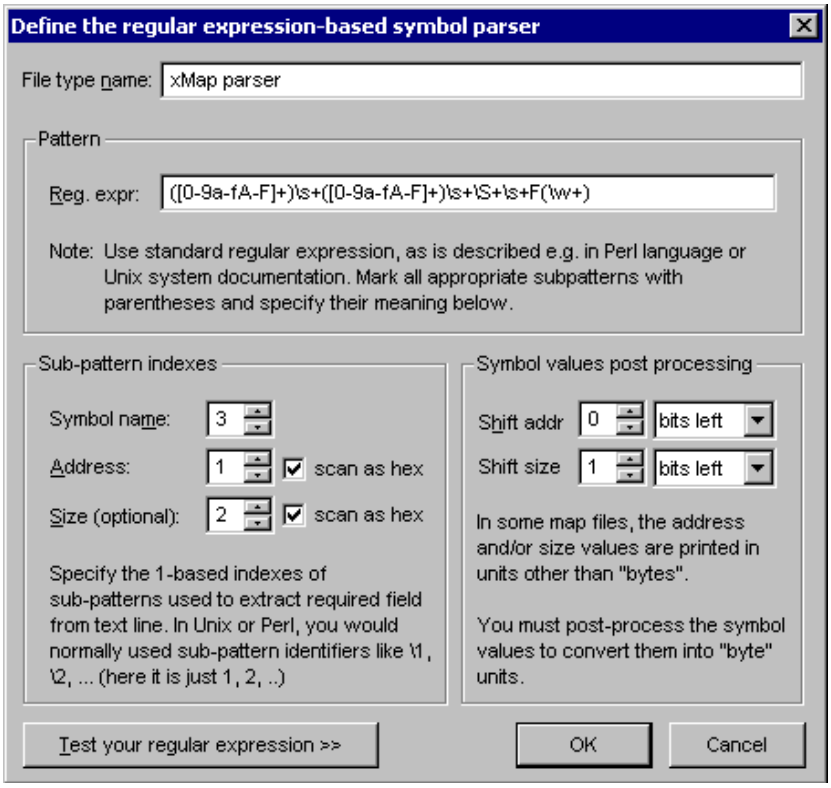

Figure 5-3. Regular Expression-based xMap File Parser

By pressing *Test your regular* expression button, the dialog vertically expands and the test panel is displayed as shown in **Figure 5-4**. Cut and paste a single line from the *xMap* file to *Input line* field and press the button *Execute reg. expression*:

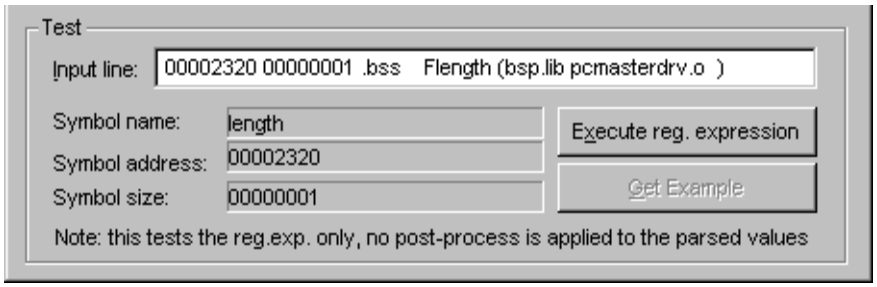

Figure 5-4. Testing Your Regular Expression

The Symbol name, Symbol address and Symbol size output fields should be displayed as shown in figure **Figure 5-4**. If you have errors, please be sure that you have Internet Explorer 5.5 or higher installed on your machine for the regular expression functionality.

To finish our example, you must set a one bit *shifting* of the "*size*" value in the *Symbol post-processing* parameters. This is because the symbol size is printed in word units (16-bit) in the *xMap* file, but we need this value in bytes.

Note: All regular expression parser definitions created will automatically be saved to the project file and to the local computer registry database for future use.

5.3 Packing Resource Files into Project File

The project often uses data from many different files. For example, each HTML page displayed for a project tree item or each image used on such a page is stored in a separate file. This might cause problems when you want to deploy or distribute a project to different computers. Using the *Pack* options shown in **Figure 5-3**, you can choose to pack the resource files into the project file when it is saved.

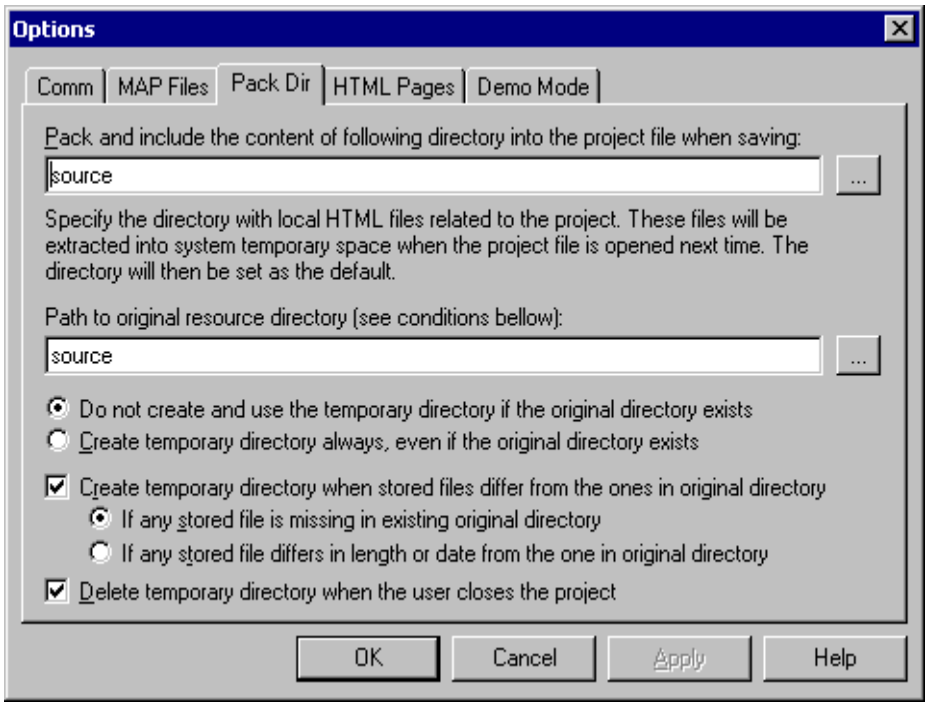

Figure 5-5. Pack Directory Options

To pack resource files into the project file, put all these files into one directory or a directory structure and specify the path to that directory in both fields of the *Pack Dir* page. By pressing the browse button (...), you can find and select the directory in the standard *open directory* dialog. The path to the directory can be specified by a path relative to the directory where the project file is saved.

When the project with the packed files is loaded next time by the application, the *original* path, entered in the second entry of the *Pack Dir* page, is checked. If it is missing, or if one of the files in it differs from the files originally packed in the project, a temporary directory is created in the system temporary space and the packed files are extracted into that directory. The temporary directory is then set as a default for the rest of resource files. It is thus very important to specify the PC master software paths to HTML files or to the symbol file relative to the directory specified in *Pack Dir* dialog.

The *Pack Dir* page displays the path to the temporary directory if it is currently in use, but still remembers the path to the original resource directory.

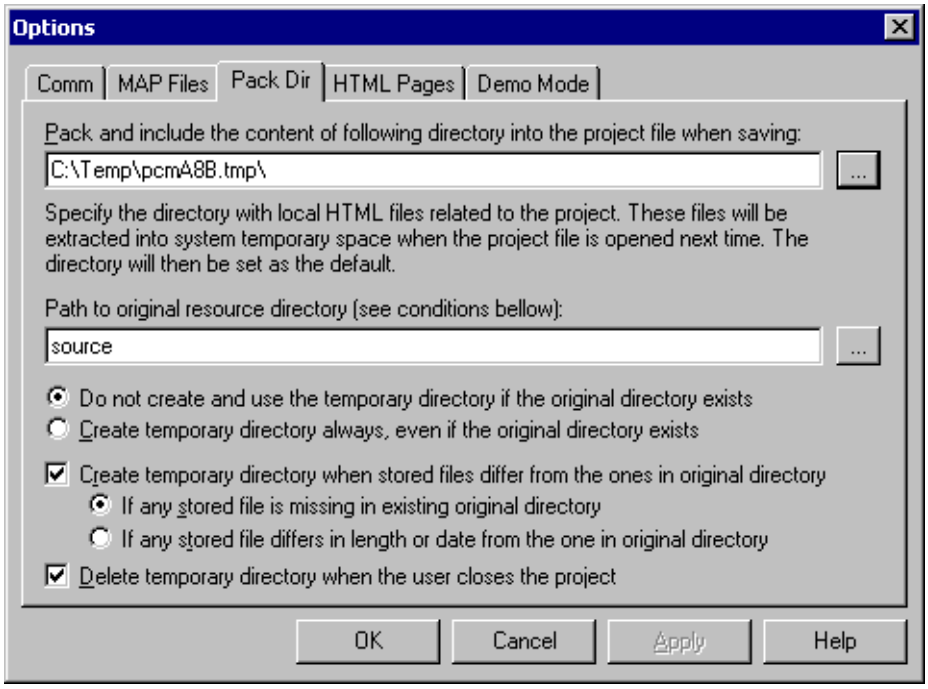

Figure 5-6. Pack Directory Options, Example 2

Using the other Pack Directory options shown in **Figure 5-6**, you can set the conditions under which the temporary directory is created.

5.3.1 Resource Files Manager

Before deploying a complex project which uses many external resources or symbol files, it is worth verifying that all file paths are correct and in the proper relative format. The path relative format must in turn be properly evaluated when the files are extracted to a temporary directory on different hosts.

The Resource Files Manager can be activated in the menu *Project / Resource Files*. The typical look of the window is shown in **Figure 5-7**

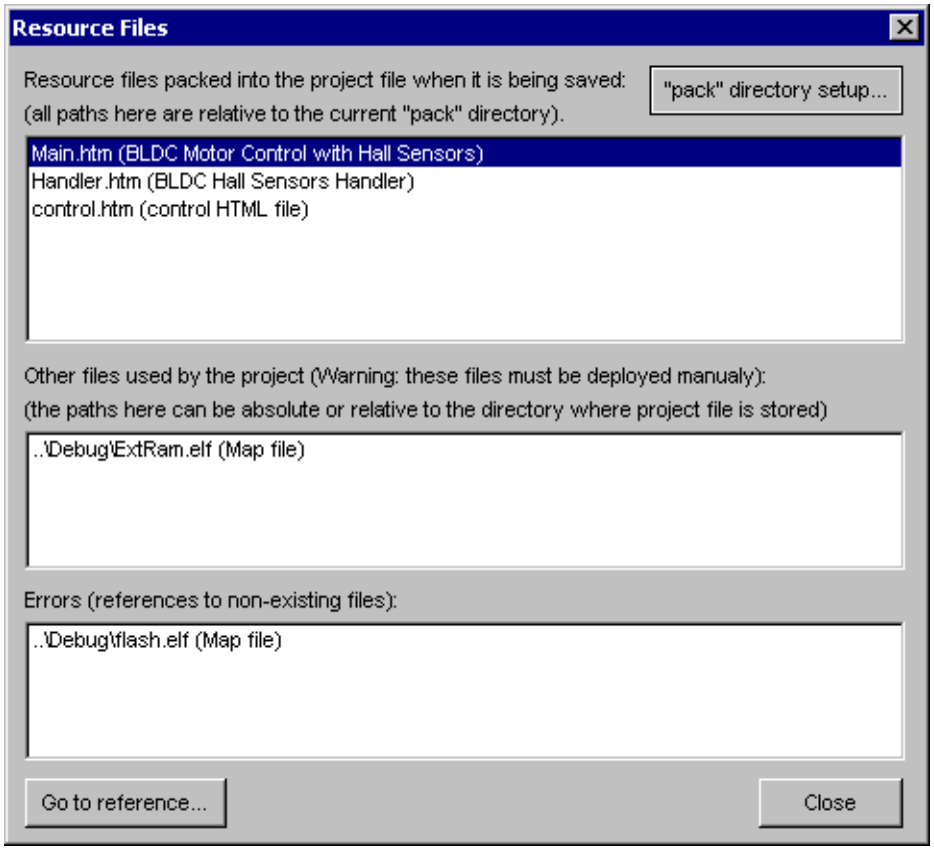

Figure 5-7. Resource Files Manager

The dialog displays all external files directly referenced by the project. Note that there can also be other files (e.g., images) referenced indirectly from the HTML pages. You should verify whether the files lie in the pack directory and whether the HTML pages point to them using a relative path.

- The first list in the dialog contains the files located in the current pack directory. These files will be properly stored in the project file when it is saved. On other hosts, the files will be extracted in a temporary space if needed.
- The second list in dialog contains the files which are successfully used by the project, but are located outside the pack directory. Their file names are specified either by absolute path or by a path relative to the directory where project is stored. However, the files will not automatically be stored in the project file and you will have to deploy them manually.
- The third list contains references to nonexisting files.

- *"Pack" directory setup* = Pressing this button opens the project *Options* dialog, where you can redefine the pack directory location and other settingsa
- *Go to reference* = Opens the dialog in which the selected file is referenced. This can be either a *Properties* dialog for a project tree item or a project *Options* dialog for shared HTML or symbol files.

5.4 HTML Pages

The project uses HTML pages to display the description and controls related to the selected item in the project tree. The path or URL of the pages are specified in the property dialog for each tree item. On the *HTML Pages* options tab, shown in **Figure 5-8**, you can specify the paths to common HTML files, displayed in the application main window.

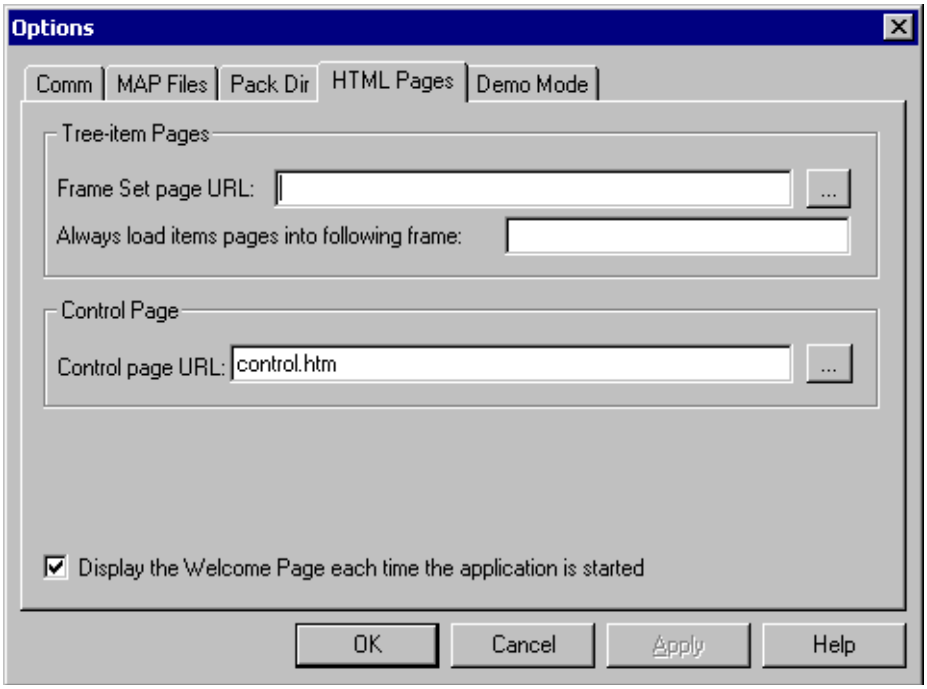

Figure 5-8. HTML Pages Options

Specify the *Frame Set page URL* if you want to divide the pages displayed for project tree items into the frames. The page specified in this field must contain the `<FRAMSET>` declaration, with one frame dedicated as a target for the description pages. The name of the target frame must be specified exactly.

The HTML code can contain frames, controls and scripts (i.e., Visual Basic Script) which can be used to access the attached target board through defined variables and commands. A special HTML page dedicated to the control task can be defined and can be displayed in the separated tab in the detail view of the application main window.

5.5 Demo Mode

An important part of the PC master software's capabilities is the demonstration and description of the target board application. It is essential that the demonstration project, once prepared, is not accidentally modified. To prevent modification, the project's author can lock the project against changes by switching it into the *Demo Mode*. See **Figure 5-9** for details of the *Demo Mode* tab.

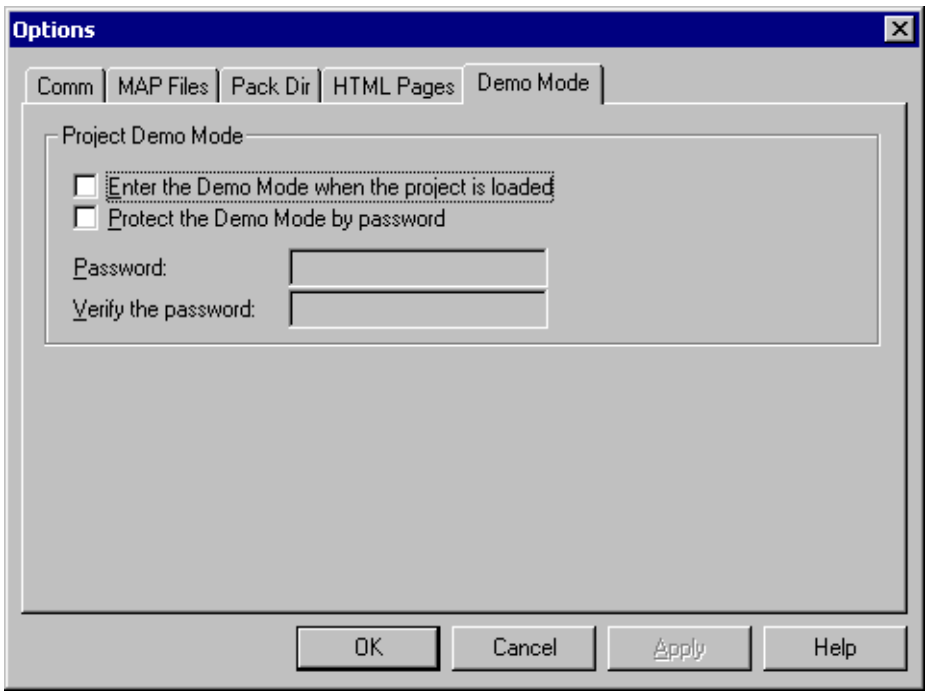

Figure 5-9. Demo Mode Options

When the *Enter the Demo Mode...* box is checked, the demo mode is activated automatically after the project is loaded. The Demo Mode can be started manually by selecting *Demo Mode* from the *File* menu. Exit from Demo Mode can be protected by a password.

In the Demo Mode, the user cannot change the *Project Tree* item properties, cannot add or remove the tree items, and cannot change any project options, except those on the communication page.

When the user asks to leave the Demo Mode, the warning message shown in **Figure 5-10** appears, and the user is prompted for the password if the Demo Mode is password-protected.

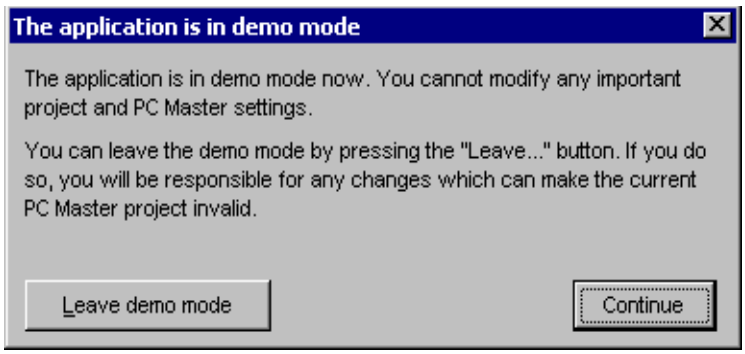

Figure 5-10. Exit Demo Mode Confirmation Dialog

Chapter 6
Scripting From the HTML Framework

The HTML pages used in the PC master software are rendered using the standard Microsoft Internet Explorer component, which first appeared in its version 4.0 and is (and, hopefully, will be) supported through its future versions.

The advantage of using the Internet Explorer component is that it supports scripting languages embedded in the HTML code and the use of the scripts to embed and access third-party ActiveX controls. The PC master software application implements an ActiveX component, exposing the basic functions required to access and control the target board application.

The following text is recommended only for HTML page creators experienced with scripting and ActiveX controls.

6.1 Special HTML Hyperlinks

When creating HTML pages which are to be displayed in the PC master software environment, the author can use special hyperlinks to navigate in the PC master software project tree or to invoke defined application commands; see **Section 4.3**.

Application command invocation hyperlinks include:

- *HREF=pcmaster:cmd:cmdname(arguments)* sends an application command *cmdname*. Command argument values can be specified in round brackets. An argument with a defined default value may be omitted.
- *HREF=pcmaster:cmdw:cmdname(arguments)* sends an application command *cmdname* and waits until the target application processes it
- *HREF=pcmaster:cmddlg:cmdname(arguments)* displays “Send Application Command” dialog for the application command *cmdname*. Any specified argument values are filled into appropriate fields in the dialog.

The following PC master software navigational hyperlinks were introduced in version 1.1:

- *HREF=pcmaster:selitem:itemname:tabname* selects a project tree item named *itemname* and displays detail view and watch view contents exactly as if the user had selected the item manually. It then selects the specified tab in detail view.

Valid *tabname* identifiers are:

– *ctl,ctltab,control,controltab* = Control Page tab
– *blk,blktab,blkinfo,blkinfotab* = Algorithm Block Description tab
– *info,infotab,iteminfo,iteminfotab* = Current Item Help tab
– *scope,osc,oscilloscope,osctab,scopetab* = Oscilloscope tab
– *recorder,rec,rectab,recordertab* = Recorder tab

6.2 PC Master Software ActiveX Object

The PC master software object is registered in the system registry during each start of the PC master software application. Its class ID (CLSID) is `{48A185F1-FFDB-11D3-80E3-00C04F176153}`; its registered name is "`MCB.PCM.1`"; its version independent name is "`MCB.PCM`".

PC master software functions can be called from any HTML code via the PC master software ActiveX control.

Insert the PC master software ActiveX control into your HTML code by the Class ID number (see the example below) and set the dimensions (height and width) to zero to make the object invisible.

```
<object name="PCMaster" width="0" height="0"
classid="clsid:48A185F1-FFDB-11D3-80E3-00C04F176153">
```

The PC master software ActiveX control provides the following functions:

- *SendCommand* sends PC master software-defined command
- *SendCommandDlg* opens command dialog and send a PC master software-defined command
- *ReadVariable* reads value from a PC master software-defined variable
- *WriteVariable* writes value to a PC master software-defined variable

PC master software version 1.1 adds following functions:

- *ReadMemory, ReadMemoryHex* reads a block of memory from a specified location
- *GetCurrentRecorderData* retrieves data currently displayed in the recorder chart
- *GetCurrentRecorderSerie* retrieves one data series from the currently-displayed recorder chart
- *StartCurrentRecorder* starts the currently-displayed recorder
- *StopCurrentRecorder* stops the currently-displayed recorder
- *GetCurrentRecorderState* retrieves the current recorder status code and text

PC master software version 1.1 also implements one callback event:

- *OnRecorderDone*, called when the currently-selected recorder finishes downloading new recorder data

6.2.1 Description of ActiveX Functions

6.2.1.1 *SendCommand*

SendCommand (*bsCmd*, *bWait* [, *bsRetMsg*])

This function sends a PC master software-defined application command to the target application.

Arguments:

bsCmd [in]	String value with the command name and arguments to be sent. The command must be defined in the currently-open PC master software project
bWait [in]	Boolean value which selects whether to wait for the result of the command; select *True* (non-zero) to wait, *False* (zero) not to wait
bsRetMsg [out]	String value returned after the command invocation. When an error occurs, this value contains an error message; otherwise, it contains a message defined in the PC master software project as command return value

Return Value:

Boolean Value	*"True*" is returned if the command has been sent without errors. "*False"* is returned if the command could not be found in PC master software or if there was a communication error. See *bsRetMsg,* **Section 6.2.1.2**, for error messages.

6.2.1.2 *SendCommandDlg*

SendCommandDlg (*bsCmd* [, *bsRetMsg*])

This function invokes the PC master software *Send Application Command* dialog.

Arguments:

bsCmd [in]	String value with the command name and arguments which should be displayed in the dialog. The command must be defined in the currently-open PC master software project
bsRetMsg [out]	Text returned after the command invocation. When an error occurs, this value contains an error message

Return Value:

Boolean value	*"True"* is returned if the command has been sent without errors. "*False"* is returned if the command could not be found in PC master software. See *bsRetMsg*, **Section 6.2.1.2**, for error messages.

6.2.1.3 *ReadVariable*

ReadVariable (*bsVar*, *vValue* [, tValue, *bsRetMsg*])

This function reads value from a PC master software-defined variable.

Arguments:

bsVar [in]	String value with the name of the variable to be read. The variable must be defined in the currently-open PC master software project
vValue [out]	Returns numeric representation of the variable *bsVar*
tValue [out]	Returns string value which represents the variable format and units necessary for displaying the variable value
bsRetMsg [out]	When an error occurs, this value contains an error message; otherwise, it is empty

Return Value:

Boolean value	*"True"* is returned if the variable has been read without errors. "*False"* is returned if the specified variable was not found in PC master software or if a communication error occured. See *bsRetMsg,* **Section 6.2.1.2**, for error messages.

6.2.1.4 *WriteVariable*

WriteVariable(*bsVar*, *vValue* [, *bsRetMsg*])

This function writes a value to a PC master software-defined variable.

Arguments:

bsVar [in]	A string value with the name of the variable to be written. The variable must be defined in the currently-open PC master software project
vValue [in]	Value to write to the variable
bsRetMsg [out]	When an error occurs, this value contains an error message; otherwise, it is empty

Return Value:

Boolean value	*"True"* is returned if the variable has been written without errors. "*False"* is returned if the specified variable was not found in PC master software, if the value passed was invalid, or if a communication error occured. See *bsRetMsg,* **Section 6.2.1.2**, for error messages.

6.2.1.5 *ReadMemory*

ReadMemory (*vAddr, vSize, arrData [, bsRetMsg]*)

ReadMemory_t (*vAddr, vSize, arrData [, bsRetMsg]*)

ReadMemoryHex (*vAddr, vSize, bsRet [, bsRetMsg]*)

These functions read a block of memory from the target application. They differ in how the data is returned to the caller.

- *ReadMemory* returns data in the safearray of variants, which can be used in both scripting languages like VBScript and in compiled languages like Visual Basic.
- *ReadMemory_t* returns data in the safearray of type “unsigned 1 byte integer”, which can be used in compiled languages like Visual Basic. Using typed arrays is faster than using arrays of variants.
- *ReadMemoryHex* returns data in the string value. Each byte is represented in hexadecimal form by two characters.

Arguments:

vAddr [in]	The address of the memory block to be read. This can be either an absolute numeric address, a symbol name valid in the current PC master software project, or a symbol name plus any numeric offset
vSize [in]	The size of memory block to be read
arrData [out]	The return array of the values
bsRetMsg [out]	When an error occurs, this value contains an error message; otherwise, it is empty

Return Value:

Boolean value	*“True”* is returned if the memory block has been read without errors. “*False”* is returned if the specified address was invalid or if a communication error occurred. See *bsRetMsg,* **Section 6.2.1.2**, for error messages.

6.2.1.6 *GetCurrentRecorderData*

GetCurrentRecorderData (*arrData [,arrSerieNames,timeBaseSec,bsRetMsg]*)

GetCurrentRecorderData_t (*arrData [,arrSerieNames,timeBaseSec,bsRetMsg]*)

These functions retrieve data currently displayed in the recorder chart. They differ in how the data is returned to the caller.

- *GetCurrentRecorderData* returns data in the safearray of variants, which can be used in both scripting languages like VBScript and in compiled languages like Visual Basic
- *GetCurrentRecorderData_t* returns data in the safearray of type “double floating point”, which can be used in compiled languages like Visual Basic. Using typed arrays is faster than using arrays of variants.

Arguments:

arrData [out]	Return, two-dimensional array of values; the first dimension is series index, the second dimension is value index
arrSerieNames [out]	Return array with names of series on appropriate indexes
timeBaseSec [out]	Returned time between individual recorded samples in second; this value can be 0 if no time base is set in PC master software recorder
bsRetMsg [out]	When an error occurs, this value contains an error message; otherwise, it is empty

Return Value:

Boolean value	*“True”* is returned if the data has been retrieved successfully from the recorder item. “*False”* is returned if there is no valid data in the recorder or there is no currently-active recorder. See *bsRetMsg,* **Section 6.2.1.2**, for error messages.

6.2.1.7 *GetCurrentRecorderSerie*

GetCurrentRecorderSerie (*bsSerieName, arrData [,timeBaseSec,bsRetMsg]*)

GetCurrentRecorderSerie_t (*bsSerieName, arrData [,timeBaseSec,bsRetMsg]*)

These functions retrieve one data series from the currently-displayed recorder chart. They differ in how the data is returned to the caller.

- *GetCurrentRecorderSerie* returns data in the safearray of variants, which can be used in both scripting languages like VBScript and in compiled languages like Visual Basic.
- *GetCurrentRecorderSerie_t* - returns data in the safearray of type “double floating point”, which can be used in compiled languages like Visual Basic. Using typed arrays is faster than using arrays of variants.

Arguments:

bsSerieName [in]	String with the name of the series whose data is to be retrieved
arrData [out]	Return array of the values
timeBaseSec [out]	Returned time between individual recorded samples in seconds. If no time base is set in the PC master software recorder, this value can be 0.
bsRetMsg [out]	When an error occurs, this value contains an error message; otherwise, it is empty

Return Value:

Boolean value	*“True”* is returned if the data has been retrieved successfuly from the recorder item. “*False”* is returned if there is no valid data in the recorder or if there is no recorder currently active. See *bsRetMsg,* **Section 6.2.1.2**, for error messages.

6.2.1.8 *StartCurrentRecorder*

StartCurrentRecorder (*[bsRetMsg]*)

This function starts the currently-displayed recorder.

Argument:

bsRetMsg [out]	When an error occurs, this value contains an error message; otherwise, it is empty

Return value:

Boolean value	"*True*" is returned if the recorder has been started successflly. "*False*" is returned if there is no recorder currently active. See *bsRetMsg,* **Section 6.2.1.2**, for error messages.

6.2.1.9 *StopCurrentRecorder*

StopCurrentRecorder (*[bsRetMsg]*)

This function manually stops the currently-displayed recorder.

Argument:

bsRetMsg [out]	When an error occurs, this value contains an error message; otherwise, it is empty

Return Value:

Boolean value	*"True"* is returned if the recorder has been successfully stopped. "*False"* is returned if there is no recorder currently active. See *bsRetMsg,* **Section 6.2.1.2** for error messages.

6.2.1.10 *GetCurrentRecorderState*

GetCurrentRecorderState (*nState [, bsRetMsg]*)

This function retrieves current recorder status code and assigned status text.

Arguments:

nState [out]	Return numeric value identifying current recorder state. Valid states codes are: 0=idle; 1=starting; 2=running; 3=downloading results; 4=holding received signal; 5=error; 6=manually stopping; 7=not initialized
bsRetMsg [out]	When an error occurs, this value contains an error message; otherwise, it contains text description of current recorder state

Return Value:

Boolean value	*"True"* is returned if the recorder state has been read successfully . "*False"* is returned if there is no recorder currently active. See *bsRetMsg,* **Section 6.2.1.2**, for error messages.

6.2.2 Description of ActiveX Events

Version 1.1 of PC master software implements an ActiveX event model and defines callback events which can be handled by the container application (Internet Explorer). Currently, only one event is defined.

`OnRecorderDone()`

This event is called when the currently-selected recorder finishes downloading new data.

6.2.3 Examples

PC master software ActiveX control functions can be used within HTML script code, JavaScript, or VisualBasic Script. The following example demonstrates the use in VisualBasic Script:

```
<script language="VBScript">

Function GetSpeed()

   succ = pcm.ReadVariable("Speed", vValue, tValue, bsRetMsg)

   If succ then
      idSpeed.InnerText = tValue

      If vValue > 100 then
         pcm.SendCommand("SlowDown()", 1)
      End If
   Else
      idError.InnerText = bsRetMsg
   End If

End Function

</script>

<object classid="clsid:48A185F1-FFDB-11D3-80E3-00C04F176153"
height="0" width="0" id="pcm" name="pcm">

<input TYPE=button VALUE="Get Speed" onClick="GetSpeed()">
```

6.2.4 Notes to JavaScript Users

JavaScript scripting language does not support [out] arguments when calling ActiveX object methods. Since the PC master software ActiveX object uses this feature, JavaScript users must define helper VBScript functions when calling out to the PC master software ActiveX. An example of a small “wrapper” JavaScript object, which handles this task for the `ReadVariable` method, follows. It can easily be extended for other PC master software ActiveX methods.

```
<script language="VBScript">

' VBScript methods first call applicative ActiveX method and then
' store all returned 'out' values into JScript object properties
```

```
Function ReadVariableVB(varName, obj)
   succ = pcm.ReadVariable(varName, vValue, tValue, retMsg)
   obj.retMsg = retMsg
   obj.vValue = vValue
   obj.tValue = tValue
   ReadVariableVB = succ
End Function

</script>

<script language="JScript">

// JavaScript proxy object constructor
function PCMASTER()
{
   // PCMASTER object member functions
   this.ReadVariable = new Function("varName",
      "return ReadVariableVB(varName, this);" );

   // here you can declare other method using VB
   // calls (WriteVariable, ReadMemory, ...)

   // define member variables which will receive all
   // possible 'out' values from VB call
   this.retMsg = "";
   this.vValue = 0;
   this.tValue = "";
}

// one instance of the proxy object is enough for all JScript code
var jspcm = new PCMASTER();

// example of accessing PC Master ActiveX from JScript
function OnTestRead()
{
   succ = jspcm.ReadVariable("variable1");
   alert(succ);
   alert(jspcm.retMsg);
   alert(jspcm.vValue);
   alert(jspcm.tValue);
}

</script>

<input TYPE=button VALUE="Test Read" onClick="OnTestRead()">
<object classid="clsid:48A185F1-FFDB-11D3-80E3-00C04F176153"
height="1" width="1" id="pcm" name="pcm" >
```

Chapter 7
License

7.1 Limited Use License Agreement

LIMITED USE LICENSE AGREEMENT

PLEASE READ THIS AGREEMENT CAREFULLY BEFORE USING THIS SOFTWARE. BY USING OR COPYING THE SOFTWARE, YOU AGREE TO THE TERMS OF THIS AGREEMENT.

The software in either source code form ("Source") or object code form ("Object") (cumulatively hereinafter "Software") is provided under a license agreement ("Agreement") as described herein. Any use of the Software, including copying, modifying, or installing the Software so that it is usable by or accessible by a central processing unit, constitutes acceptance of the terms of the Agreement by the person or persons making such use or, if employed, the employer thereof ("Licensee") and if employed, the person(s) making such use hereby warrants that he has the authority of their employer to enter this license agreement. If Licensee does not agree with and accept the terms of this Agreement, Licensee must return or destroy any media containing the Software or materials related thereto, and destroy all copies of the Software.

The Software is licensed to Licensee by Motorola Incorporated ("Motorola") for use under the terms of this Agreement. Motorola retains ownership of the Software. Motorola grants only the rights specifically granted in this Agreement and grants no other rights. Title to the Software, all copies thereof and all rights therein, including all rights in any intellectual property including patents, copyrights, and trade secrets applicable thereto, shall remain vested in Motorola.

For the Source, Motorola grants Licensee a personal, non-exclusive, non-assignable, revocable, royalty-free right to use, copy, and make derivatives of the Source solely in a development system environment in order to produce object code solely for operating on a Motorola semiconductor device having a central processing unit ("Derivative Object").

For the Object and Derivative Object, Motorola grants Licensee a personal, non-exclusive, non-assignable, revocable, royalty-free right to copy, use, and distribute the Object and the Derivative Object solely for operating on a Motorola semiconductor device having a central processing unit.

Licensee agrees to: (a) not use, modify, or copy the Software except as expressly provided herein, (b) not distribute, disclose, transfer, sell, assign, rent, lease, or otherwise make available the Software, any derivatives thereof, or this license to a third party except as expressly provided herein, (c) not remove, obliterate, or otherwise defeat any copyright, trademark, patent or proprietary notices related to the Software (d) not in any form export, re-export, resell, ship or divert or cause to be exported, re-exported, resold, shipped, or diverted, directly or indirectly, the Software or a direct product thereof to any country which the United States government or any agency thereof at the time of export or re-export requires an export license or other government approval without first obtaining such license or approval.

THE SOFTWARE IS PROVIDED ON AN "AS IS" BASIS AND WITHOUT WARRANTY OF ANY KIND INCLUDING (WITHOUT LIMITATION) ANY WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL MOTOROLA BE LIABLE FOR ANY LIABILITY OR DAMAGES OF ANY KIND INCLUDING, WITHOUT LIMITATION, DIRECT OR INDIRECT OR INCIDENTAL OR CONSEQUENTIAL OR PUNITIVE DAMAGES OR LOST PROFITS OR LOSS OF USE ARISING FROM USE OF THE SOFTWARE OR THE PRODUCT REGARDLESS OF THE FORM OF ACTION OR THEORY OF LIABILITY (INCLUDING WITHOUT LIMITATION, ACTION IN CONTRACT, NEGLIGENCE, OR PRODUCT LIABILITY) EVEN IF MOTOROLA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. THIS DISCLAIMER OF WARRANTY EXTENDS TO LICENSEE OR USERS OF PRODUCTS AND IS IN LIEU OF ALL WARRANTIES WHETHER EXPRESS, IMPLIED, OR STATUTORY, INCLUDING IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR PARTICULAR PURPOSE.

Motorola does not represent or warrant that the Software is free of infringement of any third party patents, copyrights, trade secrets, or other intellectual property rights or that Motorola has the right to grant the licenses contained herein. Motorola does not represent or warrant that the Software is free of defect, or that it meets any particular requirements or need of the Licensee, or that it conforms to any documentation, or that it meets any standards.

Motorola shall not be responsible to maintain the Software, provide upgrades to the Software, or provide any field service of the Software. Motorola reserves the right to make changes to the Software without further notice to Licensee.

The Software is not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Software could create a situation where personal injury or death may occur. Should Licensee purchase or use the Software for any such unintended or unauthorized application, Licensee shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the Software.

The term of this Agreement is for as long as Licensee uses the Software for its intended purpose and is not in default of any provisions of this Agreement. Motorola may terminate this Agreement if Licensee is in default of any of the terms and conditions of this Agreement.

This Agreement shall be governed by and construed in accordance with the laws of the State of Arizona and can only be modified in a writing signed by both parties. Licensee agrees to jurisdiction and venue in the State of Arizona.

By using, modifying, installing, compiling, or copying the Software, Licensee acknowledges that this Agreement has been read and understood and agrees to be bound by its terms and conditions. Licensee agrees that this Agreement is the complete and exclusive statement of the agreement between Licensee and Motorola and supersedes any earlier proposal or prior arrangement, whether oral or written, and any other communications relative to the subject matter of this Agreement.

Index