BDIK Target Interface

Revised 07/17/2003

metrowerks
Software Starts Here

Metrowerks reserves the right to make changes to any product described or referred to in this document without further notice. Metrowerks makes no warranty, representation or guarantee regarding the merchantability or fitness of its products for any particular purpose, nor does Metrowerks assume any liability arising out of the application or use of any product described herein and specifically disclaims any and all liability. **Metrowerks software is not authorized for and has not been designed, tested, manufactured, or intended for use in developing applications where the failure, malfunction, or any inaccuracy of the application carries a risk of death, serious bodily injury, or damage to tangible property, including, but not limited to, use in factory control systems, medical devices or facilities, nuclear facilities, aircraft navigation or communication, emergency systems, or other applications with a similar degree of potential hazard**.

How to Contact Metrowerks

Corporate Headquarters	Metrowerks Corporation 7700 West Parmer Lane Austin, TX 78729 U.S.A.
World Wide Web	`http://www.metrowerks.com`
Sales	Voice: 800-377-5416 Fax: 512-996-4910 Email: sales@metrowerks.com
Technical Support	Voice: 800-377-5416 Email: support@metrowerks.com

Table of Contents

1

BDIK Target Interface

Overview

This document includes information on the *BDIK* Target Interface and helps you understand how to use this target interface. This document is divided into following sections:

- The BDIK Target Interface Demo section provides answers to common questions and describes how to use the advanced features of the *BDIK* Target Interface.
- The Introduction section introduces the *BDIK* Target Interface.
- The Interfacing Your System and the Target section contains information about the connection between the BDI interface box and the debugger.
- The BDI Interface Software Setup section describes how to setup the BDI interface box using the ABATRON configuration tool. The discussion focuses on the firmware and the initialization list (startup init list).
- The BDIK Target Interface Menu Entries section provides a description of the *BDIK* Target Interface specific menu entries.
- The BDIK Target Interface Dialogs section provides a description of the *BDIK* Target Interface specific dialog boxes.
- The Status Bar Information for the BDIK Target Interface section describes the status bar messages for the *BDIK* Target Interface.
- The Terminal Emulation section describes how to emulate a text terminal between **CPU12**, **CPU16** and **CPU32** derivatives and the debugger.
- TheFlash Programming section describes how to proceed to program on-chip non-volatile memory area.
- TheBDIK Target Interface Environment section lists all the variables used by this target interface to store the configuration.
- The BDIK Target Interface Commands section lists all the debugger commands specific to this target interface.
- The Banked Memory Location Dialog section describes how to use the banked memory model manager with related **CPU12** derivatives.

- The Target Interface Commands Files section describes the *BDIK* Target Interface command files.
- The Index contains all keywords of the *BDIK* Target Interface.

Highlights

- The *BDIK* Target Interface currently supports the BDI - Background Debug Interfaces - designed by **ABATRON AG**, including the **BDI-HS** on CPU12, CPU16 and CPU32, **BDI1000** on the CPU12, CPU16, CPU32, M·CORE and MPC5xx, and the **BDI2000** on the CPU16, CPU32, M·CORE and MPC5xx.
- For CPU12 and HCS12 supports banked memory handling (e.g. *M68HC812A4*, *M68HC912DG128*, *MC9S12DP256* ...). Refer to the Banked Memory Location Dialog section.

Requirements

- Ensure that your hardware target board incorporates a Background Debug Mode - BDM - port for CPU background interfacing with the BDI interface and the debugger. Please check the technical specifications provided by the ABATRON User Manuals and Motorola.
- One free serial communication port of your computer is required to communicate with the BDI interface. You may need to set it up even if you will be using an Ethernet communication instead of an RS-232 serial communication.

2

BDIK Target Interface Demo

Debugging with the BDIK Target Interface

This section provides an overview of debugging with the *BDIK* Target Interface.

With this interface, you can download an executable program from the debugger environment to an external target system based on a Motorola MCU which will execute it. You will also have the feedback of the real target system behaviour to the debugger.

The debugger will fully supervise and monitor the MCU of the target system i.e. control the CPU execution. You can read and write in internal/external memory (even while the CPU is running), single-step/run/stop the CPU, set breakpoints and watchpoints (not all CPUs) in the code.

NOTE Unconcerned Components As the code is executed by an external processor, memory statistics are not available with the BDIK Target Interface. Therefore, Profiling, Coverage analysing and I/O simulation are not available with the BDIK Target Interface.

Starting with the BDIK Target Interface

1. Link the BDI box to the hardware target with the BDM cable provided by ABATRON.

2. Link the BDI box with a serial cable provided by ABATRON to your computer.
3. Make sure to connect the power supply correctly to the BDI interface and the hardware target, as specified by ABATRON.
4. Use the ABATRON configuration tool to setup the BDI box according to your hardware. Later, map this tool in the debugger itself and call the tool from the debugger. It is more secure to have the BDI interface correctly set according to your target system, as communication settings for the ABATRON configuration tool are similar for the debugger. The latest firmware and the initialization list (startup init list) matching your CPU and hardware must be loaded in the BDI. Refer to the ABATRON **User Manual** for your CPU derivative procedure. Refer to the BDI Interface Software Setup section for a quick overview of this procedure.
5. Once you have configured the BDI interface with the tool delivered by ABATRON, start the debugger from a demo project for your CPU derivative.
6. The debugger will get the CPU derivative type from the BDI interface and will sets itself automatically to this CPU family. All debugger windows are updated. *BDI Ready* (then *HALTED,* sometimes) is displayed in the status bar.

Debugging an Application in RAM

1. Select *BDIK* | *Load...* – The *Load Executable File* dialog is opened.
2. Select the file **`BDI_*_FIBORAM.ABS`** and click *Open* – The dialog is closed and the program is loaded.
3. Select *Run* | *Start/Continue* or click . – The application is started.
4. Select *Run* | *Halt* or click .– The execution of the program is stopped.

3

BDIK Target Interface Help

Introduction

Another advanced feature of the debugger for the embedded systems development world is the ability to load different target interfaces, which implements the interface with target systems. The *BDIK* Target Interface is introduced in this document.

This document describes the specific features of the BDIK Target Interface.

With this interface, you can download an executable program from the debugger environment to an external target system based on a Motorola MCU which will execute it. You also have the feedback of the real target system behaviour to the debugger.

The debugger supervises and monitors the MCU of the target system (i.e. control the CPU execution). You can read and write in internal/external memory (even while the CPU is running), single-step/run/stop the CPU, set breakpoints and watchpoints (not all CPUs) in the code.

NOTE **Unconcerned Components** As the code is executed by an external processor, memory statistics are not available with the BDIK Target Interface. Therefore, Profiling, Coverage analysing and I/O simulation will not work with the BDIK Target Interface.

Interfacing Your System and the Target

NOTE BDI Structure, Configuration, Connection to the Host, Connection to the Target, Configuration, Working Modes are described in *ABATRON User Manuals*.

The BDI interface is connected to the host computer either by a serial communication link or by a Ethernet connection. Any available communication port of your host system can be used. The communication protocol between the BDI and your target system is fully handled by the BDI Target driver automatically loaded with the BDIK Target Component. However you can adapt your target system to the BDI interface.

The BDI-to-target system communication uses a single wire serial connection. The target system has to be equipped with a BDM connector/port (see the *BDI User Manual* from ABATRON).

- Make sure that your hardware target board is/has been designed with a Background Debug Mode - BDM - port for CPU background interfacing with the BDI interface and the debugger. Please check the technical specifications provided by ABATRON User Manuals and MOTOROLA.
- One free serial communication port of your computer is required to communicate with the BDI interface. You may need to set it up even if later even if you use Ethernet communication instead of an RS-232 serial communication medium.

BDI Interface Software Setup

The BDIK Target component is delivered during installation and contains all required files, demo projects to use the BDIK debugger, and some BDI setup (init list) files. The drivers delivered with the BDI interface must be installed in order to make sure you use the latest drivers for the BDI interface. These files are delivered on a disk from ABATRON.

You must set up the target MCU through the BDI interface, according to your hardware configuration. Copy all files from the ABATRON disk to a new directory on your computer.

ABATRON provides a .EXE configuration application and a set of configuration files for specific evaluation boards and processors. These files contain microprocessor/microcontroller initialization data, vectors, chip selects for internal/external ROMs/RAMs, running modes, etc. They contain information bound to the MCU and MCU version used, and information bound to the MCU environment on the board (RAM, ROM, PIA, ACIA, etc.). Each of these files is very specific.

Running the ABATRON Configuration Tool

The configuration program (e.g. B10C12.EXE for CPU12 processor with BDI1000, B20MCORE.EXE for the M·CORE with BDI2000, BDIHSHCI.EXE for CPU16/CPU32 with BDI-HS, etc.) can also be run within the debugger on the condition that you browse for it choosing the menu entry **BDIK | Configure BDI Box...** or specify the tool path in the **BDIK | Setup...** dialog (Setup dialog). Otherwise, run the configuration tool directly from the File Manager or the Explorer.

Example with B10C12.EXE Configuration Tool

NOTE Please refer first to ABATRON User Manual for further details about the BDI interface and BDI setup.

Firmware Loading

Select **Setup | Firmware...** to open the firmware dialog.

Set the communication port and the baud rate according to your installation and press the *Connect* button. If the connection is passed, the current BDI firmware/logic is displayed. If **unknown** is displayed for the **Current** firmware/logic, you must load new firmware by pressing the *Update* button.

If you plan to use a Ethernet communication between your computer and the BDI interface, set the IP address reserved for the BDI then press the *Transmit* button. Quit the dialog by pressing the *Ok* button.

Initialization List (Startup Init List) Loading

Select **File | Open...** to load a configuration file (e.g. `HC912DA128.BDI`).

Select **Setup | Init List...** to see and edit (if necessary) the content if this configuration file.

The Startup Init List/configuration file is displayed as shown below. You can edit, add, remove (etc.) “memory write” instructions in this dialog to configure your MCU and MCU environment.

Quit the previous dialog by clicking **OK** and save the settings if necessary.

Communication with the Debugger Setup

Select **Setup | Communication...** to open the Communication Setup dialog.

In this dialog, set the communication for your future use of the BDI with the debugger. Settings made here should be identical to communication settings made in the debugger within the Communication Device Specification Dialog. Press the *Test* button to check the setup then click *OK* to quit this dialog and save the settings if necessary.

BDI Working Mode and Setup/List Transmission

Select **Setup | Mode...** to open the dialog below and download the configuration to the target board by clicking **Transmit**, after setting the required parameters.

Loading the BDIK Target Interface

The target is set in the `[Environment Variables]` section of the project file, through the statement `Target=BDIK`.

The *BDIK* Target Interface automatically detects that the target is connected to your system. If nothing is detected, the Communication Device Specification Dialog pops up: the target is not connected or is connected to a different port.

If no target is set or if a different target is set, load the *BDIK* Target Interface as described below.

In the debugger, select **Component | Set Target...**in the main menu.

The Set Target dialog is displayed. Select *BDIK* in the list of proposed targets and click **OK**.

After a successful target loading, the **Target** menu item is replaced by *BDIK*.

You can change the communication parameter (baud rate and port) by selecting the menu entry *BDIK | Connect...* .

If communication with the BDI Interface could not be established, an error message is displayed followed by the Communication Device Specification Dialog.

In this dialog you can modify the device specification (e.g. Communication Port and baud rate). These settings are saved in the current project and will be used again in future sessions.

BDIK Target Interface Menu Entries

After loading the *BDIK* Target Interface, the **Target** menu item is replaced by *BDIK*.

The **Set Bank...** menu entry is available if the connected target processor is a CPU12/HC12 derivative.

If the connection to the target has failed, the entry **Communication...** of menu *BDIK* is replaced with **Connect...** .

The different entries of the *BDIK* menu are described below:

Load...

Select **BDIK | Load...** to load the application to debug, i.e. a `.ABS` file.

Reset

The menu entry **BDIK | Reset** executes the Reset Command File and resets the hardware target. The BDI interface automatically processes the initialization list (startup init list) stored in the interface.

Communication... or Connect...

Select entry **BDIK | Communication...** or **BDIK | Connect...** to display the Communication Device Specification Dialog. If the connection to the target has failed, the entry **Communication...** of menu *BDIK* is replaced with *Connect...* .

Setup...

Select **BDIK | Setup...** to open the Setup Dialog to set the link to the ABATRON configuration tool, to set the download mode, or to set the *Continue on illegal break (banked hardware breakpoint)* option (only available for HC12/CPU12 derivative).

Configure BDI Box...

Select **BDIK | Configure BDI Box...** to open the configuration tool delivered by ABATRON that you copied on your computer. If no application tool path is currently set in the Setup Dialog, a browser dialog, **"Select BDI Box Configuration Tool"**, is automatically opened to create a link to the configuration tool application. The link is then saved in the Setup Dialog.

Set Bank...

This dialog is only available if the connected processor is a *Motorola CPU12/HC12* derivative. Select the entry **BDIK | Set Bank...** to display the Banked Memory Location Dialog.

Command Files

Select the entry **BDIK | Command Files** to display the Target Interface Commands Files dialog.

Help

Select the entry **BDIK | Help** to open the *BDIK* Target Interface Help File.

BDIK Target Interface Dialogs

This section describes the dialogs which are specific to the *BDIK* Target Interface.

Those dialogs are:

- The Communication Device Specification Dialog.
- The Setup Dialog.
- The Banked Memory Location Dialog (available only if the connected derivative is a Motorola HC12 that supports banking).
- The Target Interface Commands Files dialog.

Communication Device Specification Dialog

The Communication Device Specification dialog pops up automatically if the *BDIK* Target Interface could not establish the communication with the BDI (box) interface. However, this dialog can be opened by selecting the menu entry **BDIK | Communication...** or **BDIK | Connect...** .

If the connection to the target has been successfully achieved (dialog opened using menu entry **BDIK | Communication...**), it is not possible to modify the **Communication Device** edit box. Only the **Show Protocol** check box can be modified.

If the connection to the BDI box has failed (dialog automatically opened or using menu entry **BDIK | Connect...**), it is possible to modify the **Communication Device** edit box.

The Communication Device edit box should contain the communication settings to connect to the BDI box. The syntax of the initialisation string is:

"**COM**_n baudrate_"

where *n* is the COM port number like 1, 2, 3, etc., and where *baudrate* is 9600, 19200, 38400, 57600, 115200, according to the setup done with the ABATRON configuration application, e.g. "**COM1 57600**".

For the communication via an Ethernet and bdiNet, use the following initialisation string:

"**NETWORK** _ip_address port_"

where *ip_address* is the IP address of the BDI box or bdiNet in the form xxx.xxx.xxx.xxx and *port* is the bdiNet port, usually "1" for BDI1000 and BDI2000, e.g. "NETWORK 151.120.25.101 1".

The **Show Protocol** check box allows you to switch on/off the displays of the messages sent between the debugger and the BDI interface. If the **Show Protocol** box is checked, all the commands and responses sent and received are reported in the **Command Line** window.

NOTE	The Show Protocol is a useful debugging feature if there is a communication problem.

NOTE	The settings performed in this dialog are stored for a later debugging session in the **`[BDIK]`** section of the project file.

Setup Dialog

This dialog is opened selecting the menu entry **BDIK | Setup...** .

The BDI Box Configuration Tool Path edit box is set up with the path and application name of the configuration tool from ABATRON. The application tool is automatically browsed when selecting the *BDIK | Configure BDI Box...* menu entry and browsing for the application. Otherwise, press the *Browse...* button to look for the tool. The edit box contains (for example):

```
"C:\tmp\B10c12.exe"
```

In the *Download Mode and Data Transfer Verification*, you can set different options to transfer data from the computer to the BDI box. By default, use the *Verify only first...* option. If necessary, you can set a different option to improve transfer speed or security. By default, data compression is enabled for asynchronous communication channels. With older computers, it is possible that download speed is faster without data compression.

The *Continue on illegal break (banked hardware breakpoint)* option check box is only available for the HC12/CPU12 derivative. You can check this check box to overcome the 2-byte address size on-chip break module, which does not handle the PPAGE (e.g. HC912DG128). Note that internally, the target is halted by the hardware breakpoint (in Flash memory), compared with the breakpoint that you set, then relaunched if not (bank) matching. This feature is available as an optional. Code execution breaks are not handled when this option is set and illegal code execution is not detected. **Please use this option carefully**.

Status Bar Information for the BDIK Target Interface

When the *BDIK* Target Interface has been loaded, specific information is given in the debugger status bar. From left to right, the name of the target CPU and the debugger status (target status) are displayed.

Status Messages

Status messages are described in the following sections.

BDI ready V x.xx

The debugger is ready and waits until a new target or application is loaded. This message is generated once the debugger has been started and the connection to the hardware target has been established by the BDI. "**V x.xx**" is the current BDI firmware version.

No Link To Target

Connection to the target system has failed.

RUNNING

The application is currently executing in the debugger.

HALTED

Execution of the application has been stopped on user request. The menu entry **Run | Halt** or the **Halt** icon in the tool bar has been selected.

RESET

This message is generated when the debugger has been reseted on user request. The menu entry **BDIK | Reset** or the **Reset** icon in the tool bar has been selected, or the command Reset has been used.

Stepping and Breakpoint Messages

Stepping and breakpoint messages are described in the following sections.

STEPPED

Execution of the application has been stopped after a single step on source level. The menu entry **Run | Single Step** or the **Single Step** icon in the tool bar has been selected.

STEPPED OVER

Execution of the application has been stopped after a step over a function call. The menu entry **Run | Step Over** or the **Step Over** icon in the tool bar has been selected.

STOPPED

Execution of the application has been stopped after a step out from function call. The menu entry **Run | Step Out** or the **Step Out** icon in the tool bar has been selected.

TRACED

Execution of the application has been stopped after an single step on assembler level. The menu entry **Run | Assembly Step** or the **Assembly Step** icon in the tool bar has been selected.

BREAKPOINT

Execution of the application has been stopped because a breakpoint has been reached.

WATCHPOINT

Execution of the application has been stopped because a watchpoint has been reached.

Terminal Emulation

The BDIK Target Interface supports the emulation of a terminal. The BDI interface supports this emulation for **CPU12**, **CPU16** and **CPU32**. This allows the target application to write into the debugger ***Terminal*** component. Also, characters typed on the host's keyboard can be directed to the target application. In order to use the terminal emulation, the Terminal component has to be opened in the debugger:

Choose **Component | Open | Terminal** to open the Terminal component.

In order to simulate the terminal I/O, a work space of 4 bytes is needed. The address of this work space has to be configured with the setup program from ABATRON.

For more information, see the section "*Terminal*" in the *User Manual* from ABATRON and check the "`termbgnd.c`" source file for communication primitives on the installation disk for BDI from ABATRON.

Refer to the section ***Terminal*** *Component* in the debugger core manual.

Example for CPU12 Targets:

The following structure is located in unpaged data memory on the target:

```
0x00 RX - Flag (Byte)
0x01 RX - Char (Byte)
0x02-0x03 TX - String Pointer (Word)
```

The address of this structure is defined during BDI box setup. The **TermData** structure address (0x0800) must match with the software setup of the BDI, and exactly match the **Terminal Address** in the BDI Working Mode dialog of the ABATRON tool. Refer to the *BDI Interface Software Setup* section.

While the target is running, the BDI periodically checks if the TX - String Pointer is not zero. Received characters from the host are written to RX - Char, and the RX - Flag is set.

The following is a possible target implementation:

```
typedef struct {
  unsigned char rxFlag;
  unsigned char rxChar;
  char* txBuffer;
} TermDataT;

#define TermData (*((TermDataT*)(0x0800)))
```

```
static char txBuffer[2];

char GetChar(void)
```

```
{
  char rxChar;
  while (TermData.rxFlag == 0); /* wait for input */
  rxChar = TermData.rxChar;
  TermData.rxFlag = 0;
  return rxChar;
}
void PutChar(char ch)
{
  txBuffer[0] = ch;
  txBuffer[1] = 0;
  TermData.txBuffer = txBuffer;
  while (TermData.txBuffer != 0); /*wait for output buffer empty*/
}
void PutString(char *str)
{
  TermData.txBuffer = str;
  while (TermData.txBuffer != 0); /*wait for output buffer empty*/
}
```

Flash Programming

The BDI supports downloading and debugging code that runs in the internal Flash memory of the target CPU. Breakpoints are automatically mapped to the hardware breakpoint registers. To erase the internal flash and to enable writing to flash, Direct Commands to BDI are used. Direct commands to BDI can be executed from a **`.CMD`** command file like in the Preload Command File and the Postload Command File or the *Command Line* component with the BDI command.

This flash programming support is **not available for all CPUs**. Please check for availability in the **User Manual** for your CPU from **ABATRON**.

Use the following sequence to load code into the internal flash:

1. `Flash.Erase`
2. `Flash.Load`

3. Download the code using the normal debugger ***BDIK | Load...*** browser/menu entry or the debugger **`LOAD`** command. Every write to the flash range including **`WB`**, **`WW`**, **`WL`** commands uses the programming algorithm.

```
4. Flash.Idle
```

Examples of HC12/CPU12 Direct Commands

For the Direct Commands, the following default values are used:

HC912B32:

```
FLASH.ERASE [addr=8000] [size=8000] [sram=BDI-Workspace]
FLASH.LOAD [addr=8000] [size=8000] [sram=BDI-Workspace]
```

HC912D60:

```
FLASH.ERASE [addr=8000] [size=8000] [sram=BDI-Workspace]
FLASH.LOAD [addr=1000] [size=F000] [sram=BDI-Workspace]
```

HC912DA/G128:

```
FLASH.ERASE [addr=8000] [size=8000] [sram=BDI-Workspace]
FLASH.LOAD [addr=4000] [size=C000] [sram=BDI-Workspace]
```

Finally, set your Preload Command File and Postload Command File of your project directory with the BDI command as shown:

HC912B32:

Before downloading (in Preload Command File):

```
BDI flash.erase
BDI flash.load
```

After downloading (in Postload Command File):

```
BDI flash.idle
```

HC912D60:

Before downloading (in Preload Command File):

```
BDI flash.erase addr=8000 size=8000
BDI flash.erase addr=1000 size=7000
BDI flash.load
```

After download (in Postload Command File):

```
BDI flash.idle
```

HC912DA128 / HC912DG128:

Before downloading (in Preload Command File):

```
BDI flash.erase addr=08000 size=8000
BDI flash.erase addr=28000 size=8000
BDI flash.erase addr=48000 size=8000
BDI flash.erase addr=68000 size=8000
BDI flash.load
```

After downloading (in Postload Command File):

```
BDI flash.idle
```

BDIK Target Interface Environment

Default Target Setup

As any other target, the *BDIK* Target Interface can be loaded from the **Target** menu or can be set as a default target in the project file.

The target is set in the `[Environment Variables]` section from your project file as shown above. However, if the target is not defined, load the *BDIK* Target Interface interactively. Please refer to the Loading the BDIK Target Interface section.

Example of the `[Environment Variables]` section from your project file:

```
[Environment Variables]
...
Target=BDIK
...
```

NOTE Please see the True-Time Simulator and Real-Time Debugger core manual for further information about the project file.

BDIK Target Interface Environment Variables

This section describes the environment variables which are used by the *BDIK* Target Interface.

The *BDIK* Target Interface specific environment variables are:

- BDICONF
- COMDEV
- COMPRESS
- SHOWPROT
- SKIPILLEGALBREAK
- VERIFY

These variables are stored in the `[BDIK]` section from the project file.

Example of the `[BDIK]` section from the project file:

```
[BDIK]
CMDFILE0=CMDFILE STARTUP ON "startup.cmd"
CMDFILE1=CMDFILE RESET ON "reset.cmd"
CMDFILE2=CMDFILE PRELOAD ON "preload.cmd"
CMDFILE3=CMDFILE POSTLOAD ON "postload.cmd"
COMDEV=COM1 57600
SHOWPROT=0
BDICONF=C:\tmp\B10c12.exe
SKIPILLEGALBREAK=0
VERIFY=1
COMPRESS=1
```

The remainder of this section describes each of the variables available for the *BDIK* Target Interface. The variables are listed in alphabetical order and are divided into several topics.

Topic	Description
Short Description	Provides a short description of the variable.
Syntax	Specifies the syntax of the variable in a EBNF format.
Default	Shows the default setting for the variable.
Description	Provides a detailed description of the variable and how to use it.
Example	Small example of how to use the variable.

BDICONF

Short Description

Defines the ABATRON configuration tool file and path

Syntax

```
BDICONF=ConfigurationToolFileNameandPath
```

where `ConfigurationToolFileNameandPath` is the ABATRON configuration tool file name and path.

Default

The default value does not exist. The string "`Enter here the path to the ABATRON configuration tool.`" is displayed in the edit box.

Description

This variable defines the communication device between the computer and the BDI. It is set according to the **BDI Box Configuration Tool Path** edit box of the Setup Dialog. The **BDI Box Configuration Tool Path** edit box can be set up with the path and application name of the configuration tool from ABATRON. The application tool is automatically browsed when selecting the **BDIK | Configure BDI Box...** menu entry and browsing for the application. Otherwise, press the **Browse...** button to look for the tool.

Example

```
BDICONF=C:\tmp\B10c12.exe
```

COMDEV

Short Description

Defines the communication device between the computer and the BDI.

Syntax

```
COMDEV=COMn baudrate
```

where `n` is the COM port number like 1, 2, 3, etc. and where `baudrate` is 9600, 19200, 38400, 57600, 115200, according to the setup done in the ABATRON configuration application.

For the communication via an Ethernet:

```
COMDEV=NETWORK ip_address port
```

where `ip_address` is the IP address of the BDI box or bdiNet in the form xxx.xxx.xxx.xxx and `port` is the bdiNet port, usually "1" for BDI1000 and BDI2000.

Default

The default value is `COM1 57600`.

Description

This variable defines the communication device between the computer and the BDI. It is set according to the **Communication Device** edit box of the Communication Device Specification Dialog.

Example

```
COMDEV=COM1 57600
```

COMPRESS

Short Description

Sets data transfer compression

Syntax

```
COMPRESS=1|0
```

Default

The default value is **`1`**.

Description

This variable sets the BDI download mode with data compression. By default, data compression is enabled for asynchronous communication channels. With older computers, it is possible that download speed is faster without data compression. It is set according to the **Use Data Compression** check box of the Setup Dialog.

Example

```
COMPRESS=1
```

SHOWPROT

Short Description

Set **Show Protocol** On/Off

Syntax

```
SHOWPROT=1|0
```

Default

The default value is **0**.

Description

If the **Show Protocol** is used, all the commands and responses sent and received are reported in the **Command Line** component of the debugger.

If the variable is set to 1, **Show Protocol** is activated.

This variable is set according to the **Show Protocol** check box of the Communication Device Specification Dialog.

Example

```
SHOWPROT=1
```

NOTE The Show Protocol is a useful debugging feature if there is a communication problem.

SKIPILLEGALBREAK

Short Description

Enables skipping illegal breakpoints

Syntax

```
SKIPPILLEGALBREAK=1|0
```

Default

The default value is `0`.

Description

This variable is set according to the **Continue on illegal break (banked hardware breakpoint)** option check box of the Setup Dialog.

The **Continue on illegal break (banked hardware breakpoint)** option check box is only available for the HC12/CPU12 derivative. You can check this check box to overcome the 2-byte address size on-chip break module which does not handle the PPAGE (e.g. HC912DG128). Note that internally, the target will be halted by the hardware breakpoint (in Flash memory), compared with the breakpoint that you set, then relaunched if not (bank) matching.

Example

```
SKIPILLEGALBREAK=1
```

VERIFY

Short Description

Sets data transfer verification

Syntax

```
VERIFY=0|1|2|3
```

with 0 for no verification at all (fastest mode), 1 for first byte verification only, 2 for all data read back verification, and 3 for only verification (no write).

Default

The default value is **1**.

Description

This variable sets the BDI download mode with data verification. By default, use **Verify only first...** option. If necessary, you can set a different option to improve transfer speed or security. It is set according to the **Data Transfer Verification** radio buttons of the Setup Dialog.

Example

```
VERIFY=1
```

BDIK Target Interface Commands

This section describes the *BDIK* Target Interface specific commands that are used when the *BDIK* Target Interface is set.

The *BDIK* Target Interface specific commands are:

- BANKREG
- BDI
- PROTOCOL
- RESET

Those commands are entered in the Target Interface Commands Files or in the **Command Line** component of the debugger.

This section describes each of the commands available for the *BDIK* Target Interface. The commands are listed in alphabetical order. Each is divided into several topics.

Topic	Description
Short Description	Provides a short description of the command.
Syntax	Specifies the syntax of the command in a EBNF format.
Description	Provides a detailed description of the command and how to use it.
Example	Small example of how to use the command.

BDI

Short Description

Executes any direct BDI command

Syntax

```
BDI <ABATRON_direct_command>
```

where *ABATRON_direct_command* has the following syntax:

```
<Object>.<Action> [<parName>=<parameterValue>]...
```

Description

The BDI command executes any ABATRON direct command. ABATRON direct commands are described in the **User Manual** for your CPU from **ABATRON**. They are commonly used to download to non-volatile memory areas (please see also the Flash Programming section).

Example

```
BDI FLASH.ERASE addr=8000 size=8000 sram=0800
```

BANKREG

Short Description

Sets banked memory handling for HC12/CPU12 derivatives

Syntax

```
BANKREG [PPAGE=<PPAGE_register_adrs>]
    [DPAGE=<DPAGE_register_adrs>]
    [EPAGE=<EPAGE_register_adrs>]
```

Description

CAUTION This command is still available, but for compatibility only. However, it should not be used. The Banked Memory Location Dialog handles the banked memory handling when debugging on HC12/CPU12 derivatives.

The BANREG command lets you define if paging is used, like PPAGE (HC912DG128, HC812A4), DPAGE (HC812A4) or EPAGE (HC812A4). This command must be inserted in the Startup Command File of your project directory. As soon as the command is executed, the specified registers are displayed in the Register component window.

Example

for HC812A4:

```
BANKREG PPAGE=0x35 DPAGE=0x34 EPAGE=0x36
```

for HC912DG128:

```
BANKREG PPAGE=0xFF
```

PROTOCOL

Short Description

Switch on/off the **Show Protocol** functionality

Syntax

```
PROTOCOL ON|OFF
```

Description

If this command is used, all the messages sent to and received from the debugger are reported in the **Command Line** window of the debugger.

The *Show Protocol* facility can also be switched on/off using the corresponding check box in the Communication Device Specification Dialog.

The state of the Show Protocol is stored in the `[BDIK]` section of the project file using variable **`SHOWPROT`**.

Example

```
PROTOCOL ON
```

NOTE The Show Protocol is a useful debugging feature if there is a communication problem.

RESET

Short Description

Reset of the target board

Syntax

```
RESET
```

Description

Use this command to reset the target from the **Command Line** component of the debugger. The Reset Command File is also executed and the BDI interface automatically processes the initialization list (startup init list) stored in the interface.

Example

```
RESET
```

HC12 and HCS12 Banked Memory support

You can define which banked memory format you want to use and its location in the memory of the Motorola HC12 or HCS12 derivative you are using. The PPAGE, DPAGE and the EPAGE formats are supported, if available on the target HC12 or HCS12 derivative.

Banked Memory Location Dialog

This Banked Memory Location dialog box is available only if the connected derivative is a Motorola HC12 (CPU12) or HCS12.

The *Banked Memory Location* dialog box can be opened by selecting the menu entry *"TargetName"* | *Set Ban*k.... (In this section, ***Target Name*** is the name of the target, like ***SDI***, ***Hitex***, ***BDIK***, ***ICD-12***, ***Noral-BDM***, etc.) Using some Target Interfaces, the **Banked Memory Location** dialog box automatically pops up when the Target Interface is used with a Motorola HC12 or HCS12 derivative that supports banking. In this case, it also pops up when the banked memory area locations are not defined in the project file of the current project directory.

In this dialog box you can define which banked memory you want to use and its location. The PPAGE, DPAGE and the EPAGE indexes are supported, if they are available on the currently connected HC12 or HCS12 derivative.

PPAGE Index Tab

The PPAGE index tab of the Banked Memory Location dialog box lets you set up the *PPAGE* banked memory area. Once you have enabled PPAGE memory banking by checking the *Enable Banked Memory* Area check box, you must set the start address and the end address of this memory range.

The *PPAGE* register address must be specified in hexadecimal (e.g. 0x35 for HC812A4, 0xFF for HC912DG128, 0x30 for MC9S12DP256B).

The number of pages must be specified in decimal (e.g. 0 to 256 for HC812A4, 8 for HC912DG128, 64 for the MC9S12DP256B).

NOTE For the Hitex Target Interface, the PPAGE index tab does not appear in this dialog box if the PPAGE register is not available on the currently connected Motorola HC12 derivative. For this Target Interface it is not needed to enter the PPAGE register address.

DPAGE Index Tab

The *DPAGE* index tab of this dialog box lets you set up the *DPAGE* banked memory area. Once you have enabled *DPAGE* memory banking by checking the *Enable Banked Memory Area* check box, you must set the start address and the end address of this memory range.

The number of pages must be specified in decimal (e.g. 0 to 256 for HC812A4).

The *DPAGE* register address must be specified in hexadecimal (e.g. 0x34 for HC812A4).

NOTE For the Hitex Target Interface, the DPAGE index tab does not appear in this dialog box if the DPAGE register is not available on the currently connected Motorola HC12 derivative. For this Target Interface it is not needed to enter the DPAGE register address.

EPAGE Index Tab

The EPAGE index tab of this dialog box lets you set up the EPAGE banked memory area. Once you have enabled EPAGE memory banking by checking the Enable Banked Memory Area check box, you must set the start address and the end address of this memory range.

For some Target Interfaces the number of pages must be specified in decimal (e.g. 0 to 256 for HC812A4).

For some other Target Interfaces, the EPAGE register address must be specified in hexadecimal (e.g. 0x36 for HC812A4).

NOTE For the Hitex Target Interface, the EPAGE index tab does not appear in this dialog box if the EPAGE register is not available on the currently connected Motorola HC12 derivative. For this Target Interface it is not needed to enter the EPAGE register address.

Various Index Tab (not all Target Interfaces)

If you are using an HC12 derivative which supports banking and you don’t want to enable this mechanism, or if you want to use only one bank out of three, you can suppress the automatic display of the *Banked Memory Location* dialog by checking the **Display dialog at connection if banked memory locations not defined** check box.

NOTE The settings entered in this dialog box are stored for a later debugging session in the `["targetName"]` section of the project file.

NOTE When using the HITEX Target Interface and the M68HC12DG128 DProbeHC12-DG, at least one page must be defined from 0x8000 to 0xBFFF. Otherwise, some display problems might be encountered in the Memory component of HI-WAVE.

Associated Commands

The following sections describe the Banked Memory Location Command Line commands which are used by the Target Interface. These variables are:

BANKWINDOW

Those commands can be entered in the Target Interface Commands Files or in the **Command Line** component of *HI-WAVE*.

The Banked Memory Location commands which are used by the Target Interface are described as shown in the following table.

Topic	Description
Short Description	Provides a short description of the command.
Syntax	Specifies the syntax of the command in a EBNF format.
Description	Provides a detailed description of the command and how to use it.
Example	Small example of how to use the command.

The following sections describe each command related to the Banked Memory Location available for the Target Interface. The variables are listed in alphabetical order.

BANKWINDOW

Short Description

Specify a banked memory area and its status (enable/disable).

Syntax

```
BANKWINDOW <bank> [OFF|ON] [<range> <reg> <numofpages>]
```

with

```
bank = (PPAGE | DPAGE | EPAGE)
```

or

```
BANKWINDOW VARIOUS [DLGATCONNECT|NODLGATCONNECT]
```

Description

The command `BANKWINDOW` allows to set up the debugger to work in banked memory model.

Three different Banked Memory Area can be defined: `DPAGE`, `EPAGE` and `PPAGE`. Each banked memory area has an associated bank register, which is displayed in the Register component.

Using `BANWINDOW PPAGE ...` command will have the same effect than using the PPAGE index tab in the Banked Memory Location Dialog.

Using `BANWINDOW DPAGE ...` command will have the same effect than using the DPAGE index tab in the Banked Memory Location Dialog.

Using `BANWINDOW EPAGE ...` command will have the same effect than using the EPAGE index tab in the Banked Memory Location Dialog.

Using `BANWINDOW VARIOUS ...` command will have the same effect than using the Various index tab in the Banked Memory Location Dialog.

A banked memory area is defined by its start address, end address and the address of the Bank register.

The maximum number of pages parameter allows to see in the memory component only the available pages.

The status of the banking mechanism in the debugger is also monitored through this command: a command can be defined, but the debugger banking mechanism can be disabled.

Consider the command:

```
BANKWINDOW PPAGE ON 0x8000..0xBFFF 0x30 64
```

This command allows to use the banked memory model in the debugger using the *MC9S12DP256B*.

This commands means the `PPAGE` register located at address `0x30` must be used to build the PC address when the code is located in banked memory area, from `0x8000` to *0xBFFF*. The `64` first page in the memory map are visible (page 0x3F is the last one).

The `PPAGE` register (located at address `0x30`) will be displayed in the register component.

The bank settings are stored in the `["targetName"]` section of the `PROJECT` file using variable *`BANKWINDOWn`*.

Example

The Banking status can be get typing `BANKWINDOW` without any parameters in the **Command Line** component.

```
in>in>bankwindow
PPAGE Settings:
Status: enabled
Reg. Adr: 0x30
Range: 0x8000 to 0xbfff
Number of Pages: 64

DPAGE Settings:
Status: disabled
Reg. Adr: 0x34
Range: 0x7000 to 0x7fff
Number of Pages: 0

EPAGE Settings:
Status: disabled
Reg. Adr: 0x36
Range: 0x400 to 0x7ff
Number of Pages: 0


in>
```

The status of the **PPAGE Banked Memory area** can be changed:

```
in>BANKWINDOW PPAGE OFF
in>BANKWINDOW
PPAGE Settings:
Status: disabled
Reg. Adr: 0x30
Range: 0x8000 to 0xbfff
Number of Pages: 64

DPAGE Settings:
Status: disabled
Reg. Adr: 0x34
Range: 0x7000 to 0x7fff
Number of Pages: 0

EPAGE Settings:
Status: disabled
Reg. Adr: 0x36
Range: 0x400 to 0x7ff
Number of Pages: 0
```

```
in>
```

Associated Environment Variables

The following sections describe the Banked Memory Location environment variables which are used by the Target Interface. These variables are:

BANKWINDOWn

These variables are stored in the `["targetName"]` section from the project file.

Example of the `[BDIK]` target section from a project file:

```
[BDIK]
BANKWINDOW0=BANKWINDOW PPAGE ON 0x8000..0xBFFF 0x30 64
BANKWINDOW1=BANKWINDOW DPAGE OFF 0x7000..0x7FFF 0x34 256
BANKWINDOW2=BANKWINDOW EPAGE OFF 0x400..0x7FF 0x36 256
```

The Banked Memory Location environment variables which are used by the Target Interface are described as shown in the following table.

Topic	Description
Short Description	Provides a short description of the variable.
Syntax	Specifies the syntax of the variable in a EBNF format.
Default	Shows the default setting for the variable.
Description	Provides a detailed description of the variable and how to use it.
Example	Small example of how to use the variable.

The following sections describe each variable available for the Target Interface. The variables are listed in alphabetical order.

BANKWINDOWn

Short Description

Contains a `BANKWINDOW` *`Command Line`* command to be used to set up the Banked Memory support.

Syntax

```
BANKWINDOWn=<one BANKWINDOW Command Line command>
```

Default

All available banked memory area are disabled by default.

The default PPAGE memory banked area is 0x8000 to 0xBFFF, 8 pages allowed, with PPAGE register at address 0x35.

The default DPAGE memory banked area is 0x7000 to 0x7FFF, 256 pages allowed, with PPAGE register at address 0x34.

The default EPAGE memory banked area is 0x400 to 0x7FF, 256 pages allowed, with PPAGE register at address 0x36.

The default settings for the *VARIOUS* page is that the bank window dialog is displayed automatically when connecting when settings are not done (do only apply to the *Hitex* Target Interface).

Description

The *BANKWINDOWn* variable specifies a command file definition using `BANKWINDOW` *`Command Line`* command. Three or four of those entries should be present in the project file, depending on the Target Interface.

Those variables are used to store the Banked Memory Location definition (range, address, number of pages) and status (enable/disable) specified either with the `BANKWINDOW` *`Command Line`* command the Banked Memory Location Dialog.

Example

```
BANKWINDOW0=BANKWINDOW PPAGE OFF 0x8000..0xBFFF 0x30 64
BANKWINDOW1=BANKWINDOW DPAGE OFF 0x7000..0x7FFF 0x34 256
BANKWINDOW2=BANKWINDOW EPAGE OFF 0x400..0x7FF 0x36 256
BANKWINDOW3=BANKWINDOW VARIOUS DLGATCONNECT
```

Target Interface Commands Files

The Target Interface offers the possibility to play a specific command file on different events:

- at connection: Startup Command File,
- at reset: Reset Command File,

- right before a file is loaded: Preload Command File,
- right after a file has been loaded: Postload Command File.
- right before a "Non Volatile Memory" is erased or right before a file is programmed in "Non Volatile Memory": Vppon Command File. This command file can be used for example to enable a programming voltage by software. This command file is not available for all target interfaces.
- right after a "Non Volatile Memory" has been erased or right after a file has been programmed in "Non Volatile Memory": Vppoff Command File. This command file can be used for example to disable a programming voltage by software. This command file is not available for all target interfaces.

The command files full name and status (enable/disable) can be specified either with the `CMDFILE` Command Line command or using the Target Interface Commands Files dialog.

You can use any *HI-WAVE* command in those files and take advantage of the wide set of commands introduced in the *HI-WAVE manual* to setup the target hardware on one of those events.

Example of a command file content:

```
WB 0x0035 0x00
WB 0x0012 0x11
PROTOCOL OFF
```

- The `WB 0x0035 0x00` command sets memory location `0x35` to `0`.
- The `WB 0x0012 0x11` command sets memory location `0x12` to `0x11`.
- The command `PROTOCOL OFF` switch of the **Show Protocol**.

Startup Command File

The Startup command file is executed by *HI-WAVE* straight after the Target Interface has been loaded.

The Startup command file full name and status (enable/disable) can be specified either with the `CMDFILE STARTUP` *`Command Line`* command or using the **Startup** index of the Target Interface Commands Files dialog.

By default, the `STARTUP.CMD` file located in the current project directory is enabled as the current Startup command file.

Reset Command File

The Reset command file is executed by *HI-WAVE* straight after the reset button, menu entry or **Command Line** command has been selected.

The Reset command file full name and status (enable/disable) can be specified either with the `CMDFILE RESET` *`Command Line`* command or using the **Reset** index of the Target Interface Commands Files dialog.

By default, the `RESET.CMD` file located in the current project directory is enabled as the current Reset command file.

Preload Command File

The Preload command file is executed by *HI-WAVE* right before an application is loaded to the target system through the Target Interface.

The Preload command file full name and status (enable/disable) can be specified either with the `CMDFILE PRELOAD` *`Command Line`* command or using the **Preload** index of the Command Files Dialog.

By default, the `PRELOAD.CMD` file located in the current project directory is enabled as the current Preload command file.

Postload Command File

The Postload command file is executed by *HI-WAVE* right after an application has been loaded to the target system through the Target Interface.

The Postload command file full name and status (enable/disable) can be specified either with the `CMDFILE POSTLOAD` *`Command Line`* command or using the **Postload** index of the Command Files Dialog.

By default, the `POSTLOAD.CMD` file located in the current project directory is enabled as the current Postload command file.

Vppon Command File

The Vppon command file is executed by *HI-WAVE* right before a "Non Volatile Memory" is erased or right before a file is programmed in "Non Volatile Memory" to the target system through the target interface**Non Volatile Memory Control** dialog (***Flash***... menu entry) or FLASH PROGRAM/ERASE commands from *Flash Programming* utilities.

The Vppon command file full name and status (enable/disable) can be specified either with the `CMDFILE VPPON` *`Command Line`* command or using the **Vppon** index of the Command Files Dialog.

By default, the `VPPON.CMD` file located in the current project directory is enabled as the current Vppon command file.

This command file can be used for example to enable a programming voltage by software.

NOTE This command file is not available for all target interfaces.

Vppoff Command File

The Vppoff command file is executed by *HI-WAVE* right after a "Non Volatile Memory" has been erased or right after a file has been programmed in "Non Volatile Memory" to the target system through the target interface **Non Volatile Memory Control** dialog (***Flash***... menu entry) or FLASH PROGRAM/ERASE commands from *Flash Programming* utilities.

The Vppoff command file full name and status (enable/disable) can be specified either with the `CMDFILE VPPOFF` *`Command Line`* command or using the Vppoff index of the Command Files Dialog.

By default, the `VPPOFF.CMD` file located in the current project directory is enabled as the current Vppoff command file.

This command file can be used for example to disable a programming voltage by software.

NOTE This command file is not available for all target interfaces.

Associated Commands

This section describes the Command Files command which can be used when the Target Interface is set.

The Target Interface specific commands is:

`CMDFILE`

Those commands can be entered in the Target Interface Commands Files or in the **Command Line** component of *HI-WAVE*.

This section describes each command available for the Target Interface. The commands are listed in alphabetical order.

Topic	Description
Short Description	Provides a short description of the command.
Syntax	Specifies the syntax of the command in a EBNF format.
Description	Provides a detailed description of the command and how to use it.
Example	Small example of how to use the command.

CMDFILE

Short Description

Defines a command file path, name and status (enable/disable).

Syntax

```
CMDFILE <file kind> ON|OFF ["<file name and path>"]
```

and

```
file kind = STARTUP|RESET|PRELOAD|POSTLOAD|VPPON|VPPOFF
```

Description

The `CMDFILE` command is to be used set up a command file full name and status (disabled/enabled).

This command allows to perform the same settings than using the Command Files dialog through the **Command Line** component.

The settings of a command file are stored in the `["targetName"]` section of the `PROJECT` file using variable *`CMDFILEn`*.

Example

The list of available command files (and their status) can be get typing `CMDFILE` without any parameters in the **Command Line** component.

```
in>CMDFILE
Hitex Target Interface Command Files:
```

```
STARTUP ON startup.cmd
RESET ON reset.cmd
PRELOAD ON preload.cmd
POSTLOAD ON postload.cmd
```

The status of the Startup command file can be changed:

```
in>CMDFILE STARTUP OFF "my own startup.cmd"
in>CMDFILE
Hitex Target Interface Command Files:
STARTUP OFF  my own startup.cmd
RESET ON  reset.cmd
PRELOAD ON preload.cmd
POSTLOAD ON postload.cmd
```

Command Files Dialog

The Target Interface Command Files dialog can be opened selecting menu entry **"TargetName" | Command Files**. (In this section, *TargetName* is the name of the target, like SDI, Hitex, BDIK, ICD-12, Noral-BDM, etc.)

Each index of this dialog corresponds to an event on which a command file can be automatically run from *HI-WAVE*: Startup Command File, Reset Command File, Preload Command File, Postload Command File, Vppon Command File (not available for all targets), or Vppoff Command File (not available for all targets).

The command file in the edit box is executed when the corresponding event occurred.

Using the **Browse** button, you can set up the path and name of the command file.

The **Enable Command File** check box allows to enable/disable a command file on a event. By default, all command files are enabled:

- the default Startup command file is `STARTUP.CMD`,
- the default Reset command file is `RESET.CMD`,
- the default Preload command file is `PRELOAD.CMD`,
- the default Postload command file is `POSTLOAD.CMD`.
- the default Vppon command file is `VPPON.CMD`.
- the default Vppoff command file is `VPPOFF.CMD`.

NOTE The settings performed in this dialog are stored for a later debugging session in the **["targetName"]** section of the **PROJECT** file using variables **CMDFILE0, CMDFILE1, ... CMDFILEn**.

Associated Environment Variables

This section describes the Command Files dialog environment variables which are used by the Target Interface.

CMDFILEn

These variables are stored in the ["targetName"] section from the project file.

Example of the [NORAL FLEX BDM] target section from the project file:

```
[NORAL FLEX BDM]
CMDFILE0=CMDFILE STARTUP ON "startup.cmd"
CMDFILE1=CMDFILE RESET ON "reset.cmd"
CMDFILE2=CMDFILE PRELOAD ON "preload.cmd"
CMDFILE3=CMDFILE POSTLOAD ON "postload.cmd"
CMDFILE4=CMDFILE VPPON ON "vppon.cmd"
CMDFILE5=CMDFILE VPPOFF ON "vppoff.cmd"
```

The following section describes each variable available for the Target Interface. The variables are listed in alphabetical order.

Topic	Description
Short Description	Provides a short description of the variable.
Syntax	Specifies the syntax of the variable in a EBNF format.
Default	Shows the default setting for the variable.
Description	Provides a detailed description of the variable and how to use it.
Example	Small example of how to use the variable.

CMDFILEn

Short Description

Contains a `CMDFILE` *`Command Line`* command to be used to define a command file on a event.

Syntax

```
CMDFILEn=<command file specified using CMDFILE Command Line
    command>
```

Default

All command files are enabled by default.

The default Startup command file is `STARTUP.CMD`,

The default Reset command file is `RESET.CMD`,

The default Preload command file is `PRELOAD.CMD`,

The default Postload command file is `POSTLOAD.CMD`.

The default Vppon command file is `VPPON.CMD`.

The default Vppoff command file is `VPPOFF.CMD`.

Description

The *`CMDFILEn`* variable specifies a command file definition using `CMDFILE` *`Command Line`* command. As there are four *HI-WAVE* command files for the Target Interface, four of those entries should be present.

Those variables are used to store the command files status (enable/disable) and full name specified either with the `CMDFILE` *`Command Line`* command or using the Command Files Dialog.

Example

```
CMDFILE0=CMDFILE STARTUP ON "startup.cmd"
CMDFILE1=CMDFILE RESET ON "reset.cmd"
CMDFILE2=CMDFILE PRELOAD ON "preload.cmd"
CMDFILE3=CMDFILE POSTLOAD ON "postload.cmd"
```

```
CMDFILE4=CMDFILE VPPON OFF "vppon.cmd"
CMDFILE5=CMDFILE VPPOFF OFF "vppoff.cmd"
```

Index