AX-BDM12A

Background Debug Module for the Motorola 68HC12 Microcontrollers

USERS MANUAL

2813 Industrial Lane.
Garland, TX 75041
972-926-9303 FAX: 972-926-6063
email: sales@axman.com
web: http://www.axman.com

TABLE OF CONTENTS

1 AX-BDM12A Features

The AX-BDM12A is a Background Development Mode (BDM) cable and source level debug IDE software package for the **Motorola 68HC12** family microcontrollers. The cable with supporting host software provides access to all 68HC12x on-chip resources through the BDM access port on the device. The AX-BDM12 module connects directly to a standard Motorola 6-pin BDM connector on your target development board. No expensive emulator or PC add-on card is required – your PC communicates with the AX-BDM12 module via a standard host PC Parallel port (all cables included).

Powerful, easy to use Windows Debug software gives you advanced real-time debugging features including:

- Single step, Step Over, Go and Trace execution
- Up to 100 software Breakpoints
- Non-intrusive memory and register view and modify in real time
- Copy, Paste, Move, Fill target memory thru windows clipboard
- Built-in on the fly assembler/disassembler
- Automatic Source level symbol definitions for several compilers
- Variable Watch window – use source symbols or make up your own variables on the fly
- Open unlimited number of unique Program and Data windows
- Powerful Macro file operation support automates initialization and programming operations
- Automatic file I/O, register modification and logging during program execution
- EEPROM and FLASH Programming function
- Target Frequency selection of 16Mhz, 12Mhz, 8Mhz, 4Mhz or 2Mhz oscillator frequency

1.1 DEVICE SUPPORT and LIMITATIONS

AX-BDM12A provides full support for the following devices:

MC68HC812A4 MC68HC912B32 MC68HC912BC32 MC68HC912D60A

MC68HC912DG128A

Support is available for the M9S12 series devices (9S12DP256) with the following limitations:

1) **Maximum oscillator frequency of 16Mhz or 8Mhz E clock.**
2) **Must operate at the default clock frequency, no clock change from configured setting.**
3) **Flash may need UNSECURED for operation, will not sign-on to a secured device.**

2 WHAT YOU WILL NEED

2.1 Hardware Checklist

Host	An IBM compatible computer with SPP, ECP, or EPP parallel port
Memory	Minimum of 4 megabytes
Display	Monochrome or color (color recommended)
Target	HC12 or Target board with power supply and 6-pin standard BDM connector (Target Development Board)
Debugger	AX-BDM12 background debugger (provided)
Cables	Parallel port and target cables (provided)

2.2 Software Checklist

Operating System	Windows 95 or higher
Assembler	Motorola 68HC12 compatible assembler
Debugger software	Windows-based background debugger software (**provided**)

3 INSTALLATION AND STARTUP

3.1 Software Installation

Use the following set-up procedure to enable the AX-BDM12 debug software on your PC:

1. Insert the "**68HC12**" Support Disk into your CD drive.
2. If you have AUTO-RUN enabled for the CD, wait for the menu and select AX-BDM12.

   Otherwise, browse the CD with Windows Explorer and select the D:\Utilities\BDM12\Setup.exe file (where D: =

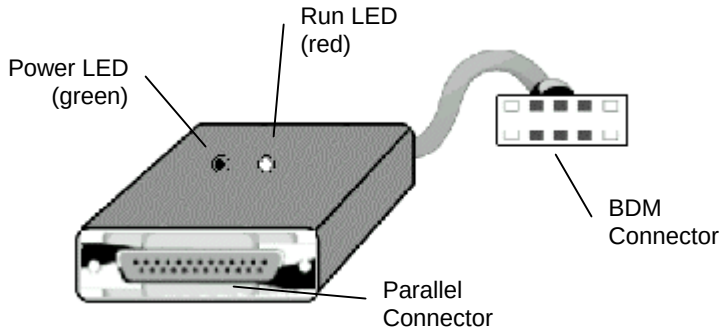


Figure1 Debugger Hardware

   CD drive). Right click the file and select Open. The BDM12 software will be installed.
3. For NT, Win2000, and XP systems the PORT95 drivers will need to be installed also.

   Browse the CD with Windows Explorer and select the D:\Utilities\BDM12\PORT95.exe file. Right click the file and select Open. The Port drivers will be installed.

3.2 Hardware Installation

There are two connectors on the debugger board. The DB-25 pin connector is a standard parallel interface to the PC. The other connector is a 10-pin connector with the 4 end pins filled so only the inner 6-pins can be connected. This connector should fit any standard Motorola BDM port connector.

Follow these steps **IN THIS ORDER** to install the AX-BDM12 hardware:

1. Connect the BDM Connector to the BDM Debug Port connector on your Target Development Board. Be sure to line up the side of the connector closest to the **RED STRIPE** to Pin 1 of the BDM socket.
2. Apply power to your Target Development Board.
3. Attach the supplied DB-25 Parallel Cable to the Parallel Connector on the AX-BDM12.
4. Attach the other end of the Parallel Cable (or connector) to a free Parallel Port on your PC.

If everything is connected and power is on, the Green LED on the BDM should be on.

NOTE: If you do not apply power to your target board before it is connected to the PC, abnormal operation may result. To assure proper operation, follow the above steps in the order given.

Figure 2 shows a typical setup using the AX-BDM12 background debugger. The serial port connector is optional since it may not be available on your target board.

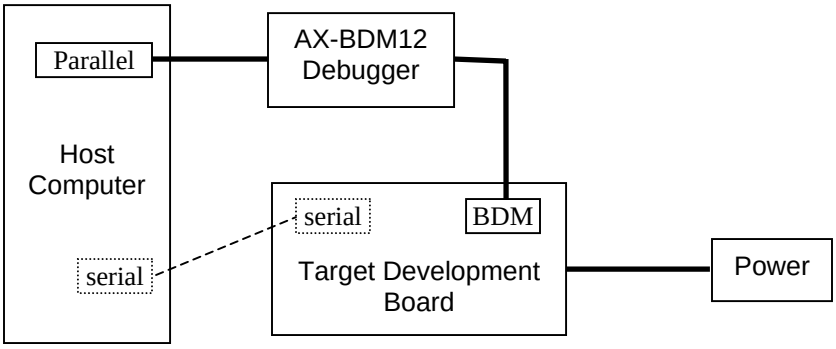


Figure 2. Typical Setup

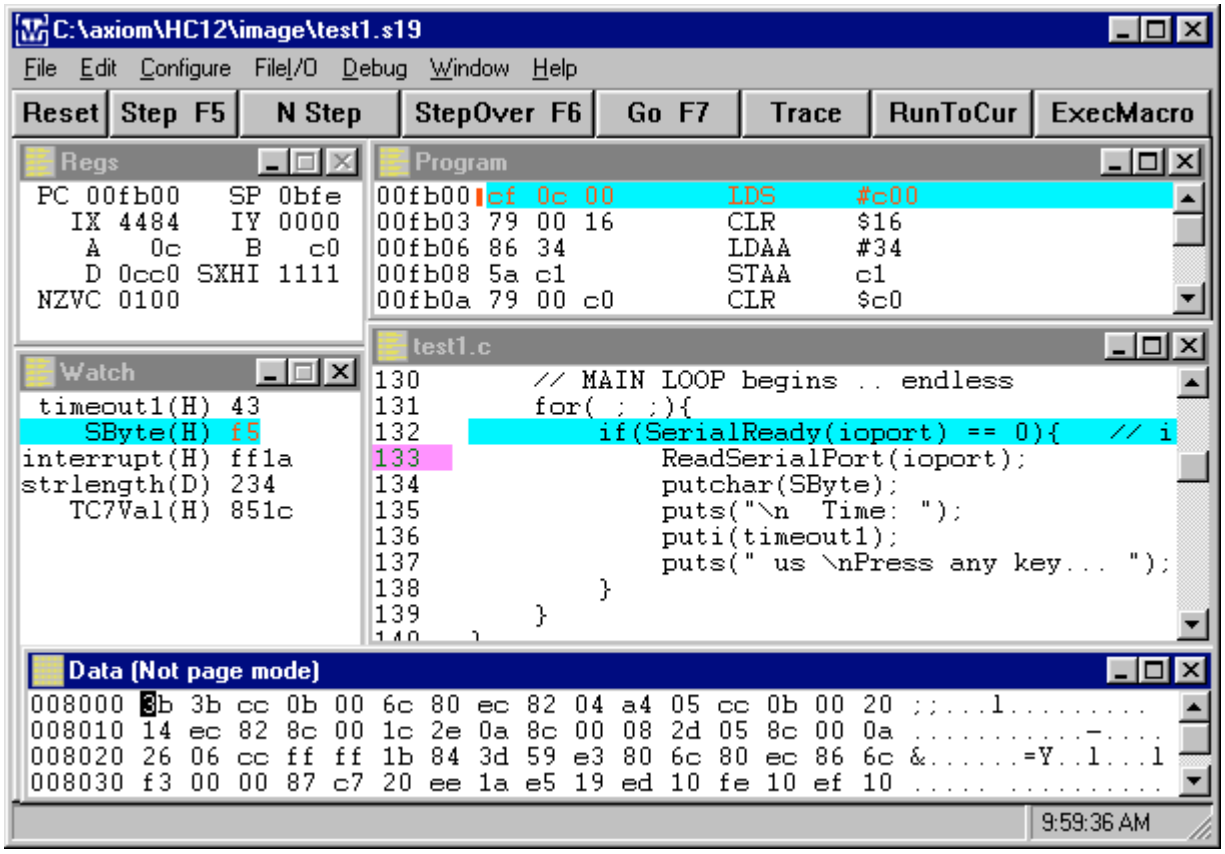
3.3 Target Hardware and BDM Software Configuration Procedure

The AX-BDM12 may not communicate with the Target correctly the first time. If the communication error message occurs, select ok and perform the configuration check list below. The Target Board must be physically configured to Single Chip Mode in order to work with the AX-BDM12. The AX-BDM12 software will then, on reset, configure the target HC12 to the memory mode specified in the configuration menus or initialization macro file (see Menu / Configure and target HC12 device specifics for more information). The end of this manual has a sample configuration for several Axiom target boards with different supporting HC12 devices. Please refer to the target board or HC12 device type for specific configuration examples. Following is a quick check list for configuration:

1) Target board or device set to single-chip mode with the MODA and MODB signals.
2) HCS12 target boards or devices with a MODC option have the option Enabled for Special Modes.
3) Target board has power applied and BDM cable is correctly installed.
4) The BDM12 software configuration (Configure – configure menus) has been set for the following:
   a) Device type selected. (Note: set breakpoints to software, see target details)
   b) Reset Mode (Note: HCS12 devices = single-chip mode and macro file should enable bus)
   c) Reset Macro file enabled if needed (see target notes).
   d) Set Target Clock (oscillator) frequency. For older non “A” type BDM pods, use the “NOT SPECIFIED” setting.
   e) Select “Save Settings” in the main Configure menu and close the BDM12 software.
   f) Launch the BDM12 software again, the target should sign-on without error message and the registers should be modifiable.

4 THE DEBUGGER SOFTWARE

When you start the AX-BDM12 background debugger, you should see a screen display similar to shown below with a valid sign-on to the target device:



The BLUE HILIGHTED LINE is the cursor position. The ■ symbol after an address or line number denotes the current Program Counter position. The text on the current instruction line will be highlighted in RED.

The debugger window has three major parts: Menus, Tool Buttons, and Windows. A detailed description of each follows.

4.1 Menus and Software Operation

4.1.1 File Menu

Load S19 or HEX	Load a file in Motorola S19 format into RAM on the target board. If symbols or source listing is available in the same directory they will be loaded also.
Load and Execute Macro	Load and execute a MACRO script file. Use this menu to select a utility macro file (erase or programming for example) to be loaded and executed. See the Macro Operations chapter for more information.
Load symbol	Load a symbol file for use by the debugger. The symbol file should be created by the compiler or assembler and will facilitate symbolic debugging. If a compatible LISTING file exists, the symbols table will be loaded automatically.
Exit	Exit the background debugger program.
<files>	A list of the five most recent file names used by the debugger are automatically appended at the end of this menu.

4.1.2 Edit Menu

Copy to Clipboard Copy HC12 memory data to the Windows clipboard.

Paste from Clipboard Paste data from the Windows clipboard to HC12 memory.

The data copied to or from the clipboard should be ASCII test characters organized in hexadecimal format (for example 3f23). Each data word is separated by a new line break.

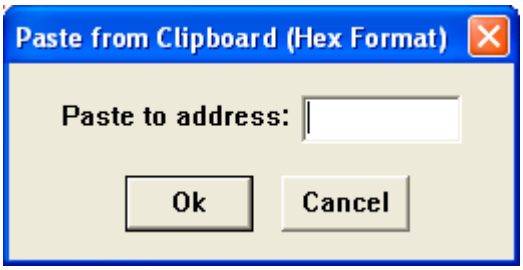
For example:

```
ffff
bfdb
fffd
ff7b
ef5e
…
```

The program will verify this format before data is pasted into the HC12 memory space.

Copy to Clipboard: Copies a range of data from the HC12 memory to the Windows clipboard. You are prompted to fill in the starting and ending memory addresses to be copied to the clipboard. After you click on Ok button, the contents of the memory are copied to clipboard in hexadecimal format.

Paste from Clipboard: Copies the contents in the clipboard to HC12 memory. You are prompted for the starting address where the date will be copied to. The contents in the clipboard should contain one 4 digit hexadecimal number per line. As soon as an error is encountered (for example, a non valid hexadecimal value), the pasting process will terminate.



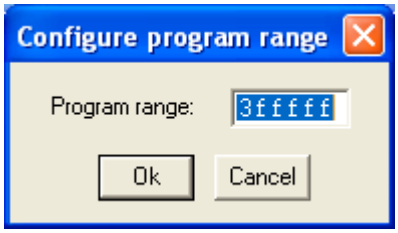
4.1.3 Configure Menu

The Configure menu provides software option settings and the Configure target menus. A check mark will appear in the menu next to the enabled settings. The software settings should be enabled with valid target communication or they may not update or may be lost. The target configuration will be saved as long as the **Save settings** or S**ave on exit** option is selected.

Enabled Character	Configure Menu	Operation Description
	Configure	Target configuration menus – see chapter below
√	**Run to Main**	Option enables a program to execute until the "Main" label or other label defined in the **Configure – Reset Mode** dialog menu is encountered in the source code after a Reset is performed. The operation will occur on new file loading or activating RESET tool button.
	Program window scroll range	Set the maximum address of the Program window.
	Data window scroll range	Set the maximum address of the Data windows
	Save setting	Saves the current desktop configuration. The information that is saved includes the windows opened, window size and position (including register, program, data, source, and watch windows), and the starting address of each data window. This information is stored in the programs INI file so the next time you start the program, these settings will automatically load.
√	**Save on Exit**	If this menu item is checked, the desktop configuration will automatically be saved when you exit the program. Whether or not this menu is checked, the starting address of each data window will always be saved in the INI file when the program exits.

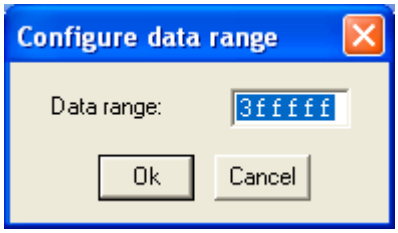
Example Program window scroll range:

Set the maximum address of the program window. This range should be set to the minimum of current device physical address range. For example a MC68HC912B32 maximum address is 0xFFFF and a M9S12DP256 maximum address is 0xFFFFF.



Example Data window scroll range:

Set the maximum address of the Data window.



4.1.4 Configure Target Sub-Menus

The configure target sub-menus provide the primary device and initial operation settings. Incorrect options will prevent the BDM12 from operating the target device correctly. Special attention should be made to the target clock setting which is critical for the BDM to sign-on to the target device.

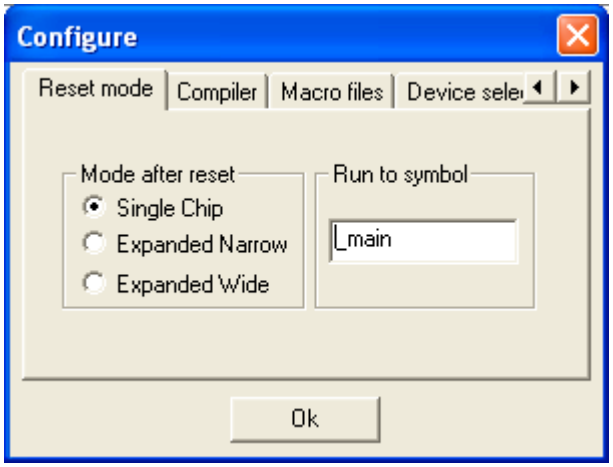
Configure Target Sub-menus	Operation Description
Reset Mode	HC12 Mode after Reset and "Run to Symbol"
Compiler	Select Compiler type for symbol loading
Macro files	Select and enable automatic execution macros for File loading or Reset initialization.
Device select	Select Target HC12 device, breakpoint type, and addressing mode.
Font	Select default window character fonts
Target clock speed	Select the target oscillator frequency

RESET Mode: Sets the target HC12 mode at Reset and whenever the Reset tool button is pressed.

If expanded memory is to be applied (i.e. external to the HC12 device) with the Expanded modes, a Reset macro is normally required to enable the bus controls also.

NOTE: HCS12 or 9S12 type devices always apply Single-Chip Mode. The Reset macro file should change the device mode for these devices.

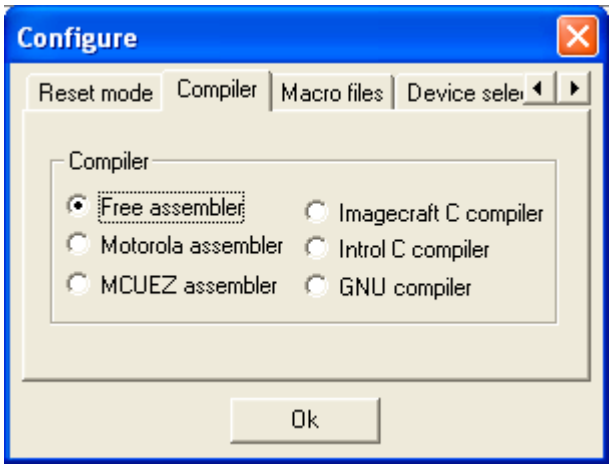
Run to symbol: This is the symbol defined for the BDM12 software "**Run to Main"** option when enabled.



Configure Compiler: Selects which Compiler/Assembler software is applied for symbolic source code loading. If the Compiler/Assembler you're using is not listed, try picking one that is close or for C code the GNU. The default "Free Assembler" (AS12) setting expects a generic Motorola listing file to be generated, which is used to pick up debugging information such as symbols, addresses, etc.

You can debug without picking the correct tool here, but you may not have access to symbols and the Source window will be blank, since source level debugging is not possible without the tool specific output (i.e. listing, map, elf/coff files).

NOTE: All compiler output files need to be in the same directory.

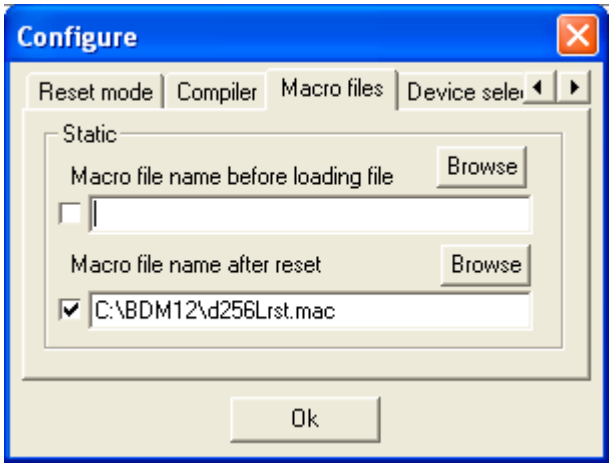


Macro files: Sets automatic execution macro files for the File loading or Reset operations. A check mark next to the file name will cause the macro in the box to be executed.

Use the Browse button to select different macro files.

See the MACRO Files chapter for more information on the Macros.

There are fully commented example macro files (**.mac**) included in the BDM12 installation directory that can be applied or modified



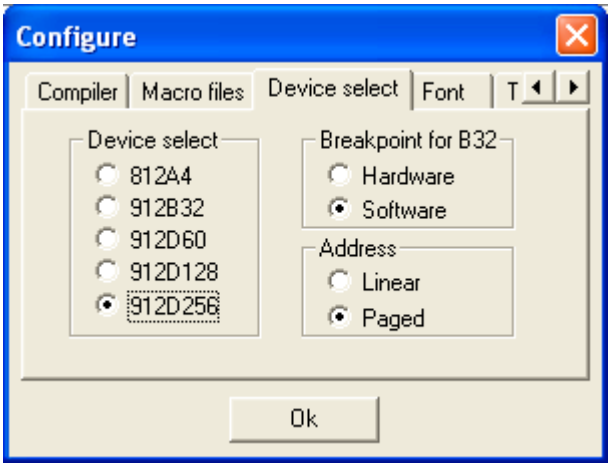
Macro file name before file loading: The selected and enabled macro file here will execute prior to any file being loaded from the File – Load S19 or Hex menu. This option allows chip select or write enable options to be configured before target memory is written.

Macro file name after Reset: The selected and enabled macro file here will execute after a target Reset operation. This option allows for subsequent target HC12 initialization of registers and memory space after a Reset occurs. The Reset macro will enable bus controls and device operating mode as needed to configure the target development memory map.

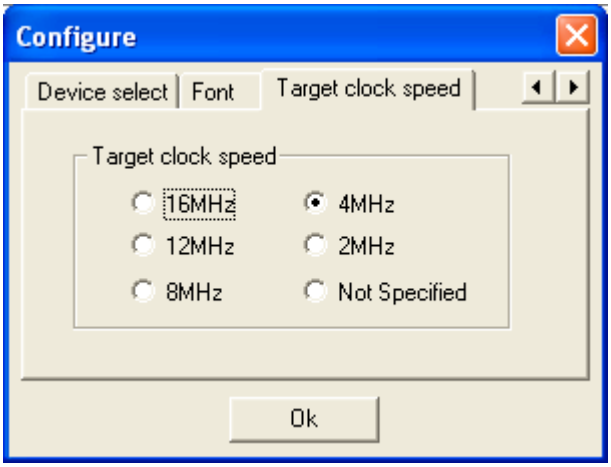
Device Select: Selection of supported HC12 target device. Note the 912D256 target is the HCS12 or 9S12 target selection.

Breakpoint for B32: Breakpoint type for the B32 or BC32 target device. All other target devices use the software breakpoint at this time.

Address: This setting determines the S record format for loading of object code. The linear selection is for reading standard S1, S2, and S3 type record files. The Paged setting is for reading the special BANKED S2 type record defined by Motorola to support HC12 paged memory.



Target clock speed: Select the default oscillator frequency (input clock to the target). The **Not Specified** setting is for the older non "A" version of the BDM12 pod.



4.1.5 File I/O menu

The File I/O menu contains powerful tools used to facilitate your program debugging. The debugger can read data from a file and place it in the 68HC12 D register during run time. The contents of the D register can also be sent to an output file. To use the this function you must set up the file name **first**, then use the "Start" and "Stop" functions from the File I/O menu.

Enabled Character	File I/O menu	Operation Description
	Start	Start the external file input or output (In or Out file must be defined first)
	Stop	Stop the external file input or output
√	**In file**	Select Input file
√	**Out file**	Select Output file

To use an input file, move the cursor in program window to a LDD EXT (load D register by extended addressing) instruction. Select “In file” from the File I/O menu. A dialog window appears as shown below.

Select the file name that contains your input data and click on the check box. Finally click on the Ok button. Next time when you click on the File I/O menu, you can see the Infile is checked. This means that your input file is correctly set up.

To use an output file, move the cursor in program window to a STD EXT (store D register by extended addressing) instruction. Select "Out file" from the File I/O menu. A dialog window appears similar to the one above. Select a file name that you want data to be written to and click on the check box. Finally click on the Ok button. Next time when you click on the File I/O menu, you can see the Out file is checked. This means that your output file is correctly set up.

Finally, select “Start” from the File I/O menu to start the simulation. Every time your program encounters the LDD instruction where you have set up your input file name, a single data word will be read from the In file and stored in the D register.

Similarly, if a STD instruction where you’ve set up your output file name is encountered, the contents of the D register will be stored in the Out file.

The simulation can be stopped by selecting “Stop” from the File I/O menu. The simulation will also stop automatically whenever the end of the input file is reached. When the simulation is stopped the input and output files are automatically closed.

To disable either the input file or the output file, click on the check box next to the file name to uncheck it.

NOTE:

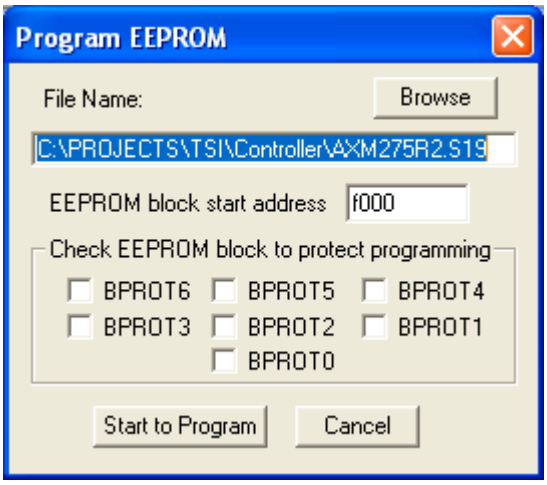
1. The File I/O feature drastically slows down program execution so you cannot use it to test real-time operations. Also, it may take longer than expected to finish the simulation.
2. The file I/O functions only store the contents of the D register in a file or read data from a file and store it in the D register. The memory in the instruction is not changed. In order to change memory using this feature you must use another instruction to load or save the contents of the D register.

4.1.6 Debug menu

Debug menu	Operation Description
Step	Executes one instruction at the current Program Counter position in the Program and/or Source Windows. Same operation as the STEP button or F5 function key.
Step Over	Same as the "Step" function except it will not go through a subroutine. Instead, program execution continues until the instruction following the call to the subroutine is reached. Same operation as the Step Over button or F6 function key.
N Step	Configures the N Step button operation. N Step provides multiple Step operations.
GO	Initiates real time program execution from the current program counter location. Same operation as the GO button or F7 function key.
Run to cursor	Performs a GO command and stops execution at the highlighted instruction location in the active Program or source windows. Same operation as the run to cursor button or F8 function key.
Program internal EEPROM	Program the internal EEPROM memory of an HC812A4, HC912B32, or HC912BC32. All other devices will apply the Macro style programming.
Program external EEPROM	Program the external EEprom memories on a supported Axiom board.
Program B32 flash EEPROM	Program the internal flash memory of a B32 or BC32 device.
Erase B32 flash EEPROM	Erase the internal flash memory of a B32 or BC32 device.
Program D60 flash EEPROM	Program the internal flash memory of a D60 device. (not D60A)
Erase D60 flash EEPROM	Erase the internal flash memory of a D60 device. (not D60A)

N Step sub-menu: N Step sub menu allows some configuration of the N step function.

Program internal EEPROM sub-menu: Sub-menu allows selection of file and setting of supported device EEPROM base address and block protection.



4.1.7 Window menu

The Window Menu functions control how the windows are displayed in the program.

Window menu	Operation Description
Program	Opens a program window. Multiple Program windows can be open at the same time.
Data	Opens a data window. Multiple Data windows can be open at the same time.
Source	Opens the source window associated with the current program loaded in memory. Only one source window can be opened.
Watch	Opens the variable watch window. Only one watch window maybe opened. Several watched variables can be place in the window.
Cascade	Cascades all the open windows on the screen.
Tile	Tiles all the open windows on the screen.
Arrange icon	Arrange all the minimized windows in neat order
<windows>	All the windows currently open will be appended to the end of this menu.

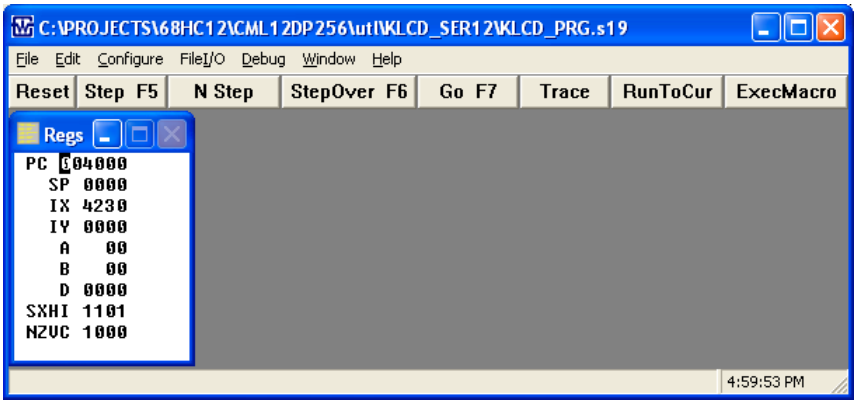
4.1.8 Help menu

Help menu	Operation Description
Contents	Opens the Help index and associated help files
Search for help	Provides searching for subject of the help files
About Hc12bgnd	Provides software revision and information

5 Windows

There are 5 types of windows provided on the BDM desktop screen: Register, Program, Data, Source, and Watch. The Register window is dedicated and always available. All other windows can be opened or closed with the Window menu. Multiple Program and Data windows are available to view different sections of memory. The Trace Push Button will open a pop-up type window to display trace results.

5.1 Register window



The register window provides target device CPU internal core register view and access. Only one register window is possible. A register value change during code execution will be indicated in RED text.

PC = Program Counter or Instruction pointer

SP = Stack Pointer register

IX = Index 'X' register

IY = Index 'Y' register

A = Accumulator 'A' or MSB of the 'D' register

B = Accumulator 'B' or LSB of the 'D' register

D = Accumulator 'D' or double accumulator A | B

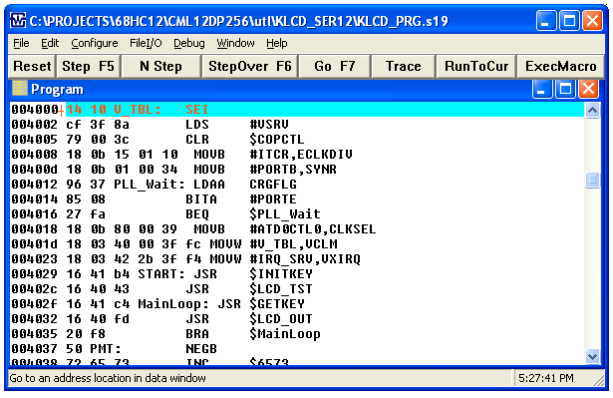
SXHI = CCR (condition code register) high nibble bits. S = Stop bit, X = X bit, H = Half Carry flag, I = Interrupt bit

NZVC = CCR (condition code register) low nibble bits. N = Sign flag, Z = Zero flag, V= Overflow flag, C = Carry flag

To change the contents of a register, right click the register value to be modified, and type in the new value in hex format. The register will be updated after the least significant digit (right most hex number) is entered. If the register window is not visible it will be available in the lower left corner of the BDM screen to reopen. Note: another window may obstruct the view to reopen the register window, move the obstructing windows to find the register window.

See the target device CPU reference manual for more details on the core registers and their operation.

5.2 Program window



Displays the program memory in three parts left to right on the display: Address, Opcode, and assembly mnemonics. The assembly mnemonics are interpreted (disassembled) according to the data in the target memory. Symbols from the listing or symbol table may also be shown if they are loaded.

The highlighted line is the current cursor position. A '+' symbol next to the address and Red text notes the current program counter or instruction pointer position.

Breakpoints can be set or cleared by a left click on the address field. A set breakpoint is indicated by a pink highlight on the address field.

More than one Program Window may be opened.

A right click in the Program window will present the Program Pop-up menu.

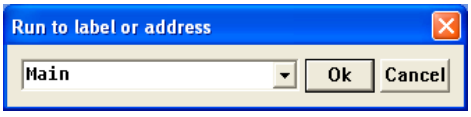
5.2.1 Program Window Pop-up menu

The Program Pop-up menu provides utility operations and Program window settings. A right click in the Program window will present the Program Pop-up menu.

Run to cursor	Run application code to the indicated cursor position in the Program window and stop.
Run to Label (or address)	Run application code to the identified label or address location and stop. See Run to Label below.
Edit	Edit the instruction on the cursor line, in-line assembler operation. See Edit below for more detail.
Go to Break Point	Change Program window memory view to a current breakpoint address. See Go to Break below.
Go to Address	Change Program window memory view to a particular starting address, See Go to address below.
Go to Symbol	Change Program window memory view to a symbol address. See Go to Symbol below.
Go to PC	Change Program window memory view to current PC register address.
PC = Address at Cursor	Change PC register value to the current address selected by the cursor.
Clear all BKPT	Clear all set breakpoints.
√ **Program Page Enable**	If checked, Paged Program operation of Program window is enabled. Program window view for addresses above 64K bytes (0xFFFF) will be viewed via the Program Page. Do not use this setting if loaded file is not Paged.

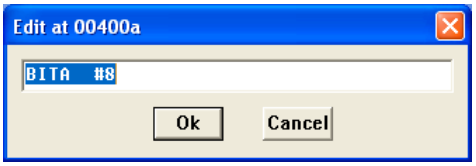
Run to Label or Address:

Pop-up window will query for the Label name, symbol name or address value to run to then stop execution.



Edit:

Allows Editing of the program in memory at the current Cursor Position. This is an in-line assembly operation and must be performed in Ram type memory space or the new instruction will not be installed.

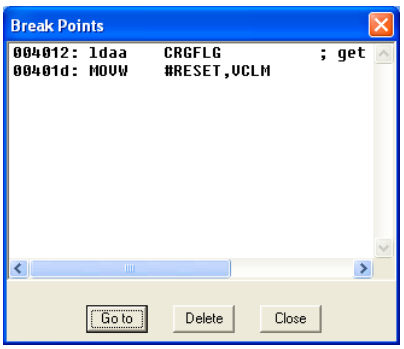


Standard assembly mnemonic instructions can be typed in this box. The operands in the instruction (such as an immediate number, an extended address) must be either hexadecimal numbers or symbols used in the program (if symbols are loaded).

For example type in instructions like LDD #1234, LDD $4321, or BRA MY_SYMBOL. If an error is found in the instruction typed in, the program will issue a warning beep and the error must be corrected or select the "Cancel" button.

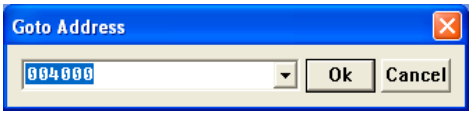
Go to Break Point:

If any break points are set, this menu item will quickly move the Program window cursor to one of the breakpoint locations. The Break Points pop-up will open showing the currently set breakpoints. Select one of the breakpoints to go to or delete the breakpoint.



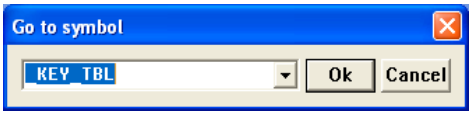
Go to Address:

Move Program window cursor to an address location or select an address from a list previously entered by clicking on the down arrow. The cursor will change to the address selected.

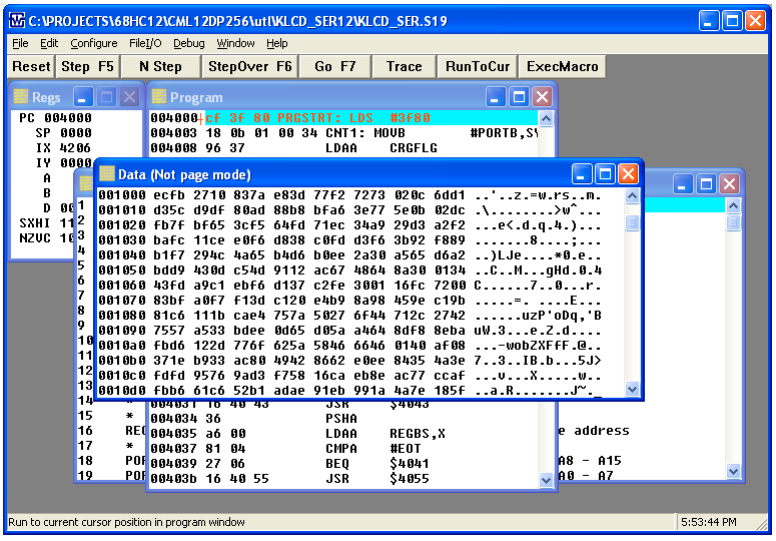


Go to Symbol:

If a symbol table is loaded, use this menu to jump quickly to one of the symbols. This is very helpful if setting a breakpoint at a subroutine during debugging is wanted. Click on the down arrow in the dialog box, all the symbols loaded will show up for selection.



5.3 Data Window



Displays target memory content by starting address in left column followed by memory data. More than one data window can be opened.

The window title bar indicates the view mode and if page views are enabled. See the Pop-up window options for more details.

Corresponding ASCII text values in the data can be displayed on the right of the hex data when enabled. Also, data width and byte or word view is selectable in the Pop-up window options.

To edit values in the data window place the cursor or left click on the data value to be changed and type in the new value. Note that the last digit (right digit) of the word or byte must be modified before the value will update. Only RAM type memory or registers can be modified in the data window.

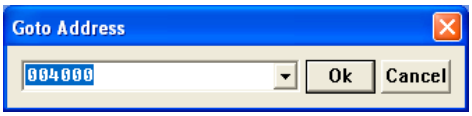
A right click in the Data window will open the Data Pop-up menu.

5.3.1 Data Window Pop-up menu

The Data Pop-up menu provides utility operations and Data window Page View settings. A right click in the Data window will present the Data Pop-up menu. Note that only one Page view setting may be selected per data window.

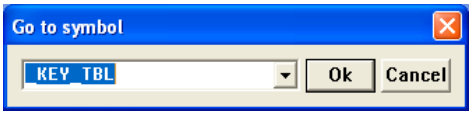
Go to address	Change Data window memory view to a particular starting address, See Go to address below.
Go to symbol	Change Data window memory view to a symbol address. See Go to Symbol below.
Go to PC	Change Data window memory view to the current PC register address.
Fill Block	Fill a range or block of addresses with given data. See Fill Block details below.
Move Block	Move a block of data from one starting address location to another. See Move Block details below.
Toggle data per Line (16 or 8)	Change the data view width between 16 bytes / 8 words or 8 bytes / 4 words.
Display as Word (Byte)	Change the data view between 8 bit Byte and 16 bit Word view.
(√) Display ASCII data	If checked, will enable the corresponding ASCII text represented by the data will be viewed on the left side of the data window.
(√) Program Page view	If checked, will enable Program Page view in the data window. Memory above 64K byte (0xFFFF) addresses will be viewed by Program Pages. Do not use this setting if Program Paging is not supported by the target device.
(√) Data Page view	If checked, will enable Data Page view with access to the Data Page at the 0x7000 address. The target device must support Data Paging to use this view.
(√) Extra page at 0x0000 view	If checked, will enable Extra Page view with access to the Extra Page at the 0x0000 address. The target device must support the Extra Page to use this view.
(√) Extra page at 0x0400 view	If checked, will enable Extra Page view with access to the Extra Page at the 0x0400 address. The target device must support the Extra Page to use this view.
(√) Not Page Mode	If checked, will disable Page views and provide a standard linear memory view. This view is compatible with all targets.

Go to Address:



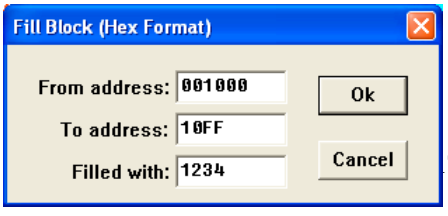
Move Data window base to an address location or select an address from a list previously entered by clicking on the down arrow. The cursor will change to the address you select.

Go to Symbol:



If a symbol table is loaded, use this menu to move the data window to one of the defined symbol locations. This is very helpful in finding and viewing variable data. Click on the down arrow in the dialog box, all the symbols loaded will show up for selection.

Fill Block:

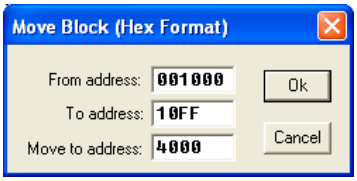


Fill a block of memory with provided data. The from address and to address represent the block range of memory that will be filled with the data. The

addresses should be an even number boundary due to the block operation operates in word or 16 bit data size. Data entered should be 16 bit or word size data.

Move Block:

Move or copy a block of data from one memory location to another memory location. The from address and to address represent the block range of memory that will be moved or copied. The move to address is the base or starting address of the memory to be updated. The addresses should be an even number boundary due to the block operation operates in word or 16 bit data size.



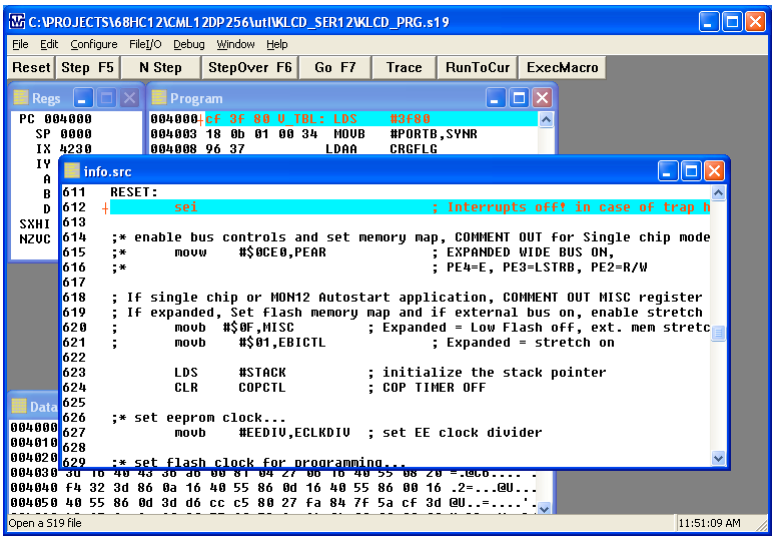
5.3.2 Source Window

Displays symbolic source code of current file loaded by the BDM if the source is compatible and can be found on the host.

The highlighted line is the current cursor position. A '+' symbol next to the line number and red text denotes the current program counter position.

Breakpoints can be set or cleared by left clicking on the line number. The line number will be highlighted in pink if a breakpoint is set on that line. Note that Breakpoints will only set on instruction source lines. Source window breakpoints will not set on comments, text, or Label only source lines.

Right click in the source window will open the Pop-up menu for the source window.



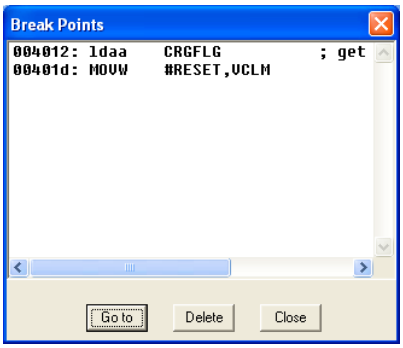
5.3.3 Source Window Pop-up menu

The Source Pop-up menu provides utility operations and window settings. A right click in the Source window will present the Source Pop-up menu.

Run to cursor	Run application code to the indicated cursor position in the Source window and stop. If the cursor is not on a valid instruction line, the program will not execute.
Go to PC	Change Source window view to current PC register address.
Go to Break Point	Change Source window view to a current breakpoint address. See Go to Break below.
Open Source file	Open the source file indicated. See Open source File below.
Find	Find a text entry in the source file. See Find below.
Find next (F3)	Find the next text entry identified by the Find operation.
TAB size	Set the source window view Tab character size. See TAB size below.

Go to Break Point:

If any break points are set, this menu item will quickly move the Source window cursor to one of the breakpoint locations. The Break Points pop-up will open showing the currently set breakpoints. You can then select one of the breakpoints to go to or delete the breakpoint.

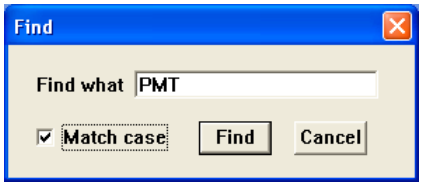


Open Source file:

Prompt window opens to Select source file to open in source window. The down arrow will provide a list of available source files. The BDM software will generate a source file named “info.src” to display source information.

Find:

Prompt window opens to enter text string to be found in the source file. Match case option allows exact text match to be found otherwise upper or lower case is not checked. The cursor will move to the line that contains the matched text when the operation is performed. If the text cannot be found, an error prompt will be displayed.



Tab size:

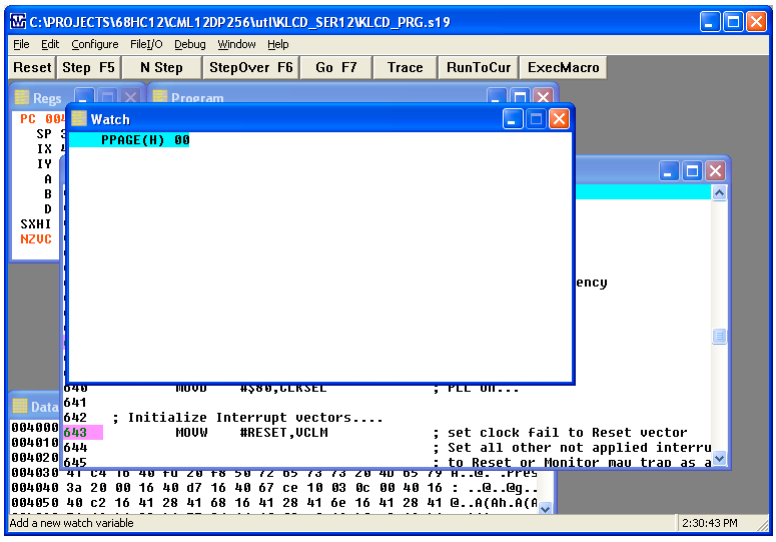
Tab size effects the source window view. A prompt window opens to set the Tab size space count. Tab size space count can be set from 1 to 8 spaces.

5.4 Watch Window

The Watch window provides variable or register view with the associated label name for easy viewing. The data displayed will be updated on every break of execution or with the non-intrusive update operation (see the non-intrusive update section in this manual).

Only one Watch window can be open.

The Watch window provides a pop-up menu for adding or removing watched items. Right click in the window to open the pop-up menu.



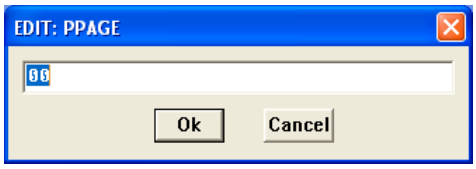
5.4.1 Watch Window Pop-up menu

The Watch Pop-up menu provides utility operations for the Watch window. A right click in the Watch window will present the Watch Pop-up menu.

Edit	Edit the value of the contents of the watched variable highlighted by the cursor. See Edit below.
Add variable	Add a new variable to the watch window. See Add and Modify below.
Modify setting	Modify the settings of the variable highlighted by the cursor. See Add and Modify below.
Move up	Move position of watched variable highlighted by the cursor up the list in the watch window.
Move down	Move position of watched variable highlighted by the cursor down the list in the watch window.
Delete	Delete the watched variable highlighted by the cursor from the watch window.
Open config	Open a Watch window configuration file. Watch window set-up can be saved in configuration files. See Watch configuration save and restore below.
Save config as	Save the current Watch window configuration for future use. See Watch configuration save and restore below.

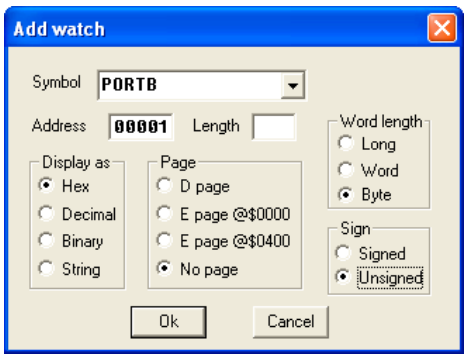
Edit:

Allows Editing the watched variable contents. The variable at the cursor position will be prompted in a edit window. A new value can be typed in to the variable.



Add or Modify setting:

Allows adding or changing the settings of a watched variable. The Modify setting will open the prompt window for the variable highlighted by the cursor.



Symbol: The name it may be picked from the loaded symbol list with the down arrow or a reference name typed in.

Address: The memory location of the variable can be typed in. If it is defined in the symbol list the address will be inserted automatically.

Display as: Select the number base or text string setting for the variable view wanted. The view type will be displayed in the watch window next to the variable name in parentheses. Example: PPAGE = Hex value, watch display = PPAGE (H) xx

Length: If the variable is a string, the length should be set to the character count to be displayed. Example: Value_out = a string variable with 8 character ASCII digits or text. Length = 8, Address = first character or lowest address, word length = do not care.

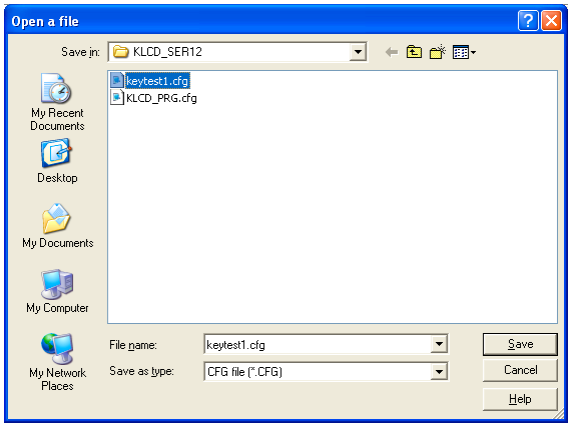
Page: The location of the variable if it is on a Paged access or in the linear memory space (No Page).

Word Length: If the variable is NOT a string and is a number, the Word length selection will set the size of the number to be displayed. 8 bits = byte, 16 bits = word or 2 bytes, 32 bits = long word or 4 bytes.

Sign: All values but Decimal should be unsigned. If the variable has the Decimal view selected then the sign value can be enabled. The watch display will indicate if the number is negative sign by adding the negative sign character.

Open configuration or save configuration as:

Allows saving a watch window configuration or loading a previously saved configuration. Both operations open the pop-up window to query for a file to open or file name to save. The .cfg extension is applied by default.



5.5 Window Notes:

1. All windows can be minimized and maximized. The windows maybe arranged in any order and stacked for preference.
2. Opening many program and data windows will slow down debugging since it takes time to update the contents of these windows.

6 Tool Bar Buttons:

There are seven push-buttons below the menu. You can click on these push-button with the left mouse button to quickly perform certain functions. The push-buttons that are used most often have hot keys associated with them. You can alternately press the buttons hot keys to perform the same functions even more quickly.

Following are descriptions of each of the buttons:

6.1 Reset Push Button

Reset Hardware Resets the Target and software. The program counter is returned to the program start position. The Reset macro is executed if enabled.

6.2 Step F5 Push Button

Step F5 Execute a Single Step function. Performs same operation as the Debug menu Step item. Hot Key F5 performs the same operation.

6.3 Nstep Push Button

N Step Execute a multiple Step operation until the user stops the stepping by clicking the Button again. Button name will change to “Stepping” during the operation. Stepping rate can be adjusted in the Debug menu NStep dialog window.

6.4 StepOver F6 Push Button

StepOver F6 Execute a single Step Over operation. Performs the same operation as the Debug menu Step Over item. Hot key F6 performs same operation.

6.5 GO (BREAK) F7 Push Button

Go F7

Begins program execution in real time. Performs same operation as the Debug menu Go item. Button name and operation will change to **BREAK** during execution. The **Break** button will stop execution when it is pressed. Hot key F7 performs same operation.

6.6 Trace Push Button

Trace

Begins the Trace operation. Button has two states: **Trace** and **Tracing**. Clicking on the button will toggle between the two states. When the **Trace** state is shown, clicking on the button will start executing the program in Trace mode and the button label will change to "**Tracing**". Clicking on this button again will stop tracing execution and the Trace buffer pop-up window will open.

The Trace buffer will display the last 64 instructions executed with the register status following each instruction.

Tracing is dedicated BDM operation and is much slower than operation in real time.

6.7 RuntoCursor F8 Push Button

RuntoCur

Run to Cursor button begins program execution from the current program counter (PC register) location and continues until the instruction under the highlighted cursor position in the Program or Source window is reached. This is a fast way of implementing a temporary breakpoint. Same operation as Debug menu Run to Cursor item. Hot key is F8.

7 Macro Files

Macro files are simple text BDM command files that instruct the BDM to perform certain operations automatically. Operation of macros are divided into to two basic types: Reset or initialization type and flash programming or erasing type. Macro operations write to target memory and registers, no read operation is provided. Several example macros are provided for Axiom Mfg. development boards.

Macro files are executed from three different locations in the BDM software: Files – Load and execute Macro or the ExecMacro Push Button (same file will be launched); Configure – Compiler, Macros Tab menu for File or Reset macro types, and Debug - Program or Erase Flash EEPROM macro. Refer to the section needed for details on each macro launching or type.

7.1 Macro Commands

The macro command set is small and very easy to use. Syntax is similar to common assemblers where a space or tab delimits fields and a semicolon indicates a comment line or text. All numbers must be HEX format. Upper or lower case characters do not matter.

ip <data>	Update the IP (Instruction Pointer or PC - Program Counter) with the data. Data size should be word or 16 bits.
reset	Reset the Target. Same operation as the Reset push button.
dmm.b <addr> <data>	Write a Byte (8 bits) of data to the address specified.
dmm.w <addr> <data>	Write a Word (16 bits) of data to the address specified. Address can be aligned or unaligned
load <filename>	Load S record file into memory. The file name should include the full path name or the file should be placed in the BDM software

	installation directory / folder. The file extension must be .S19, .SX or .HEX. Otherwise the file will not be loaded. File type cannot be banked or paged, and must be linear. The target memory space for the file must be RAM type.
erase <exe> <buffer> <mask>	Erase target Flash or EEPROM memory with algorithm. Exe is the starting address of the loaded algorithm executable. Buffer is data memory buffer address on the target. Mask is the error condition mask word. See Erase macro below for more details.
prog <file><exe><buffer><mask>	Program or load Flash or EEPROM memory with algorithm. File is S record to be loaded. Exe is the starting address of the loaded algorithm executable. Buffer is the data memory buffer address on the target. Mask is the error condition mask word. See Prog macro below for more details.

7.2 Reset or Initialization Macro

The reset or initialization macro is enabled in the Configure – Configure Compiler, Macros window. Two types of macro can be enabled here: Before File Load and After Reset. The Before File Load macro can unlock memory space or perform some other function to prepare for a memory update on the target. A Reset macro will execute after the Reset Push Button is clicked. This type of macro can configure target device registers and pre-load data into target memory as required. Typical application of a Reset macro would be to enable the Expanded Bus for memory access on the target so that a code file can be loaded. Example Reset macro listing that enables the external bus :

```
dmm.b 0B E2      ; Write MODE register (0x0B) with byte 0xE2 data.
; Mode = expanded wide with EMK bit set for external program paging.
dmm.b 0A 0C      ;  Write PEAR register (0x0A) with byte 0x0C data.
; PEAR  = R/W and LSTB* signal enabled for bus control.
;dmm.b 13 0c      ; Write MISC register (0x13) with byte 0x0C data.
;  Set maximum bus stretch for external memory, internal Flash off for external ram access by BDM.
```

7.3 Program or Erase Macro

The Program or Erase macros provide for access to a loaded algorithm executable required for Flash type memory modifications. These macros are performed with the File – Load and Execute macro menu or the Debug – Program or Erase Flash EEPROM menu. The two methods of launching this type of macro require different macro content, see the section below for the method to be applied. External files referred to in the macro must have a complete and correct path name to the file. The referred to files in the macro may also be placed in the BDM software installation directory without directory path information needed. The erase and program operations can be performed in the same macro but the provided examples do not apply this method.

Both of these macros typically perform three operations: Configure, Load, and Execute. The Configure step is optional and would be needed to setup the target registers, for example the flash clock setting, for the program or erase operation. Load step is required to load the flash algorithm executable in target memory. The executable is a small application program written for the target device to operate from internal ram memory typically. Then the Erase or Prog command will execute the flash update operation by sending the referenced file to the target executable application and provide error reporting determined by any cleared bits in the Mask word. The Eraee or Prog command should only be applied in macros that are launched by the Files – Load and execute macro method and NOT in the Debug – Erase or Program flash EEPROM method. The Debug menu allows for the file to be selected for this method of operation.

Following is an example of each macro applying the Files – Load and Execute macro launching method.

7.3.1 Erase Macro Example

The Erase macro typically performs a flash memory erase operation. Following is an example flash erase macro for a 9S12DP256 target device operating with a 4Mhz oscillator.

```
dmm.b 100 15     ;  Set flash clock register (FCLKDIV/0x100), to 4Mhz time base setting.
```

dmm.b 104 FF ; Turn flash protection off

LOAD DP256FE.S19 ; load Flash erase algorithm file

ERASE 2000 1000 FFFF ; perform erase operation

Notes:
1) The dmm.b commands configure the target's flash control registers for correct clock and disable protection.
2) The Load command places the flash erase algorithm program into the target device memory space. The S record file will be loaded at the address locations contained in the file.
3) The ERASE command executes the application algorithm by setting the PC to the 2000 hex location and performing a GO command. The erase algorithm program will erase the flash if possible and ends execution with a BGND instruction to return control to the BDM. The BDM will inspect the error mask for any change and report the error code to the user.

7.3.2 PROG Macro Example

The Prog macro typically performs a flash memory program or load operation. The S record type must be a Paged or Banked type S2 record for correct operation. Following is an example flash programming macro for a 9S12DP256 target device operating with a 16Mhz oscillator.

dmm.b 100 4B ; Set flash clock register (FCLKDIV / 0x100), to 16Mhz time base setting.
dmm.b 104 FF ; Turn flash protection off

LOAD DP256Fpg.S19 ; load Flash programming algorithm file

PROG Myfile.s19 2000 1000 FFFF; perform erase operation
Notes:
1) The dmm.b commands configure the target's flash control registers for correct clock and disable protection.
2) The Load command places the flash programming algorithm program into the target device memory space. The S record file will be loaded at the address locations contained in the file.
3) The PROG command executes the application algorithm by:
   a) Loading up to 64 bytes of the Myfile.s19 into the target memory buffer at the 0x1000 address.
   b) Set the S record data start address and number of bytes provided for the algorithm program. See Algorithm programs for more detail.
   c) Set the PC to the algorithm program start address of 0x2000 and perform a GO command.
   d) The Programming algorithm will load the data provided in the memory buffer area into the flash starting at the provided start address. Execution ends with a BGND instruction to return control to the BDM. The BDM will inspect the error mask for any change and report the error code to the user.

A sample macro script is included in the program director called: **`HC12BGND.MAC`**. The following MACRO commands are currently available:

`ip` `<data>`	write the data to IP (Program Counter)
`reset`	reset the Microcontroller
`dmm.b` `<addr> <data>`	write data (byte) to address
`dmm.w` `<addr> <data>`	write data (word) to address. Address can be aligned or unaligned
`load` `<filename>`	load a file into memory. The file name should include the full path name. The file extension must be S19 or HEX. Otherwise the file would not be loaded.

NOTE:

1. The number of spaces between each field is not important.
2. The case is NOT sensitive.
3. Both the <address> and <data> must be in HEX format, for example, 400 will be treated as 400 HEX instead of 400 DECIMAL. The program will check the format. If it is invalid, memory will not be loaded.

4. Anything following a semicolon ";" is considered a comment and will be ignored.

8 INI File

The **HC12BGND.INI** file contains the custom settings used by the debugger software. Manually editing the INI file will probably never be needed. Information will be stored in this file automatically when the BDM software is executed and settings are saved.

The INI file has the following sections:

Window section Contains the number and sizes of the windows opened. You can edit some of the lines to change this setting. For example, you can change the directory where your code will be loaded.

Path section Stores the S19 or HEX file names loaded, the MACRO file name, and the file names for file I/O. The S19 or HEX files will be appended to the FILE menu. The maximum number of these files is five.

Config section Contains miscellaneous configuration settings used by the program.

Watch section Stores the variables in the watch windows.

9 Troubleshooting Tips

- If your target board uses EEPROMs for code storage, you will need to install RAM in those sockets while debugging. After debugging is finished you can re-install the EEPROMs for programming.
- After you add or change memory on the board, make sure you set the correct Expanded mode under the Config / Mode after reset menu, then press the Reset button.

Insert target device or board particular set up tables here

The following chart shows the switch settings you must use to set the Axiom CMD12a4 development board to single chip mode depending on the memory map you are using. Wide mode gives you the fastest speed while Narrow mode is slower but frees up an I/O port.

Memory Mode	CMD12 Board - Mode Switch Setting
Single Chip, Internal Memory Map only	1-2 ON, 3-5 OFF
Single Chip for 8-bit Narrow Expanded Programming Mode	1-4 ON, 5 OFF
Single Chip for 16-bit Wide Expanded Programming Mode	1,2,4 ON, 3,5 OFF
	PB68HC12 Board
Single Chip Mode	Jumper A & B instaled

If you're using a development board other than the CMD12A4, consult your board documentation for the proper procedures for Single Chip Mode configuration.

After your Target board is powered ON and the debugger is connected to the PC, you can start the Background Debugger Software by double clicking on the AX-BDM12 Icon that was installed under Windows.

If you receive an error message, check all the cable connections and be sure power is supplied to the board. Verify the CONFIGURE settings for device type, clock rate, and mode. See the Hardware Installation section for more information.

Make sure there's no conflicting device using the same Parallel port on your PC. Also make sure no other software is running in the background that could interfere with the debugger software. Exit all other windows applications and restart windows just to make sure.