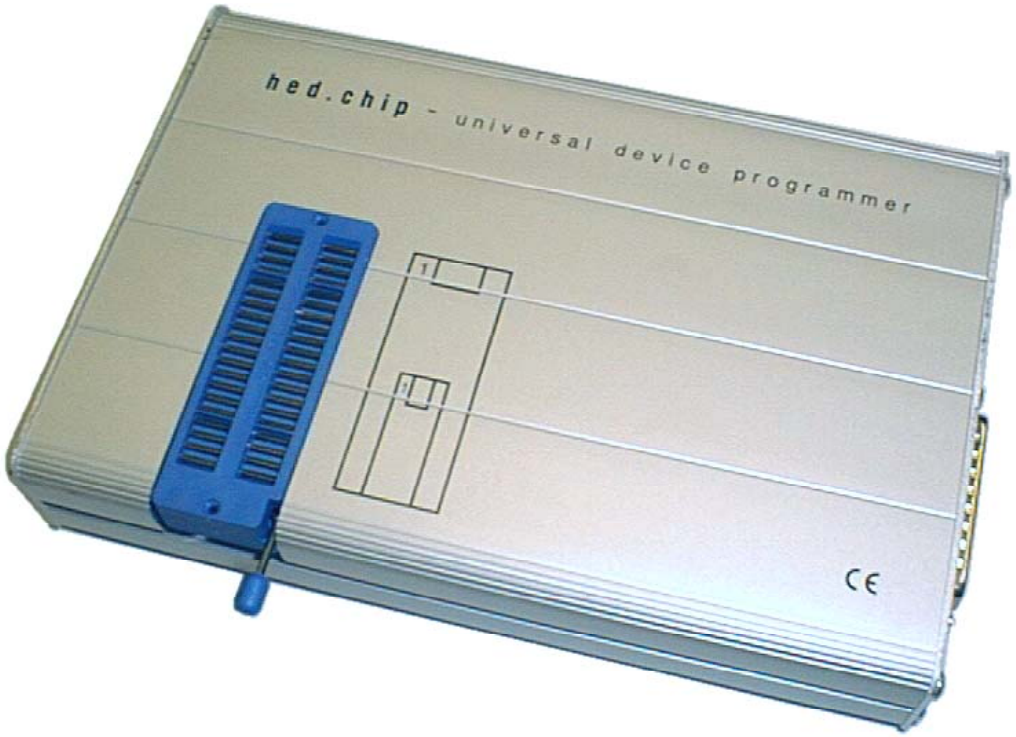

hed.chip - universal device programmer
CE

1 General instructions for usage

1.1 Important information

This programmer is only suitable for the devices mentioned in the device list. Other types of devices will be rejected by the software, although unsuitable devices may already be damaged by the initialization.

Devices must be correctly inserted according to the markings next to the test socket. Small devices must be placed as near as possible to the locking lever on the ZIF DIP40 test socket.

The programmer can only be used with machines that are 100% IBM PC compatible.

hed.chip and the manual are designed for users with a basic knowledge of electronics, and it is assumed that the user has experience in handling electrical equipment and electronic devices. All electronics safety precautions must be followed.

The user's hardware and software is not known and no responsibility can be taken for damage to customers' equipment and materials.

1.2 Introduction

hed.chip is a universal device programmer. The selection of programmable devices is oriented towards the needs of the developer and is continuously being updated. The programming algorithms are contained in the PC software and can be updated for specification changes and new devices. **hed.chip** can be operated from the DOS command line, or by using a graphical windows user interface with a powerful device database.

By using simple, cheap adapters SMD devices in PLCC and SOIC packages can be used and special adapters enable the programming of more devices.

This users guide consists of several parts. The programmer and its basic operation are described in this chapter. Chapter 2 describes the special properties of the various devices. A readme file gives information on the current version of the software. Since the graphical interface is self-explanatory, this guide concentrates on the usage of the DOS command line program HEDCHIP.EXE. This guide is also available as a Windows help file. Any required information can be viewed via a contents directory, an index or by searching for key words in the text.

1.3 hed.chip hardware versions

hed.chip exists in two slightly different hardware versions. All programmers sold after 01.11.99 are of version 2. Version 1 has been sold to Germany and Austria only. If you need English language support for version 1, please contact Hoepping Elektronik Design.

1.4 Hardware and software requirements

Command line operation

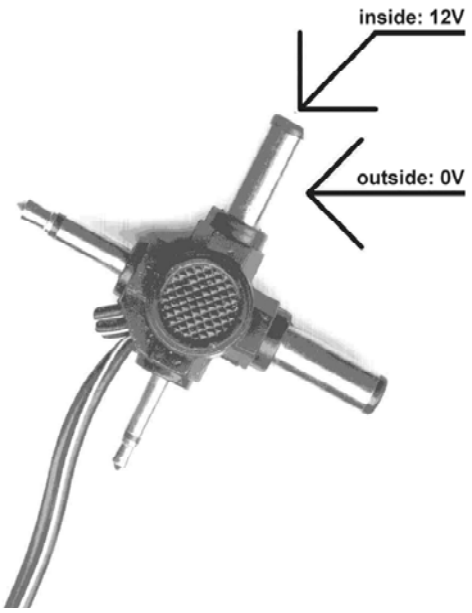
IBM compatible PC, 80386 or higher recommended, DOS 5.0 or DOS-task in Windows 95/98 or Windows NT 4.0 (P166MMX required) , fully IBM compatible printer port.

Graphical user interface HC95

IBM compatible PC, Pentium P100 with 32 MB RAM, Windows 95/98 or Windows NT 4.0 (P166MMX required), fully IBM compatible printer port.

1.5 Connection and software installation

hed.chip is connected to a parallel (printer) port. The port must be recognized by the machine's BIOS at boot-up stage. Ports LPT1 to 4 are supported. Modern, bi-directional ports need to be set to standard mode by jumper or BIOS setup. A 25 pin 1:1 data cable with male/female plugs is required.



A 12V / 800mA unregulated AC adapter that gives 12 to 15V output voltage at currents between 50mA and 500mA is required as the power supply.

hed.chip has a low-voltage coaxial connector with an internal diameter of 1.95 and 2.1 mm for connecting the AC adapter. Most AC adapters have this kind of connector in the form of a cross-shaped unit. See picture on the left.

The polarity of the power supply must be set correctly. **hed.chip** can not be damaged by using an incorrect setting, however it will only work if the correct polarity has been set. See picture on the left for the correct setting.

The voltage setting must be 12V.

A suitable AC adapter and data cable can be obtained from us.

The DOS software can simply be copied to a suitable directory on the hard disk.

The Windows software is installed by starting the setup program. For Windows NT, administrator's rights are necessary.

Internation Power Supply

Usually, we deliver hed.chip with a power supply suitable for Germany, Switzerland, and Austria. This power supply has a wall socket connector compatible with these countries and it requires 220 to 240VAC, 50Hz.

For countries that have different requirements, we supply optionally an International Power Supply. This power supply has wall socket connectors for the US, UK and Germany. It requires 100 to 240VAC, 50 or 60Hz

Plug and Prog

hed.chip supports the user when setting up the power supply connection. Simply start the HEDCHIP.EXE program without any parameters. HEDCHIP.EXE can detect which port the programmer is connected to, and can also detect whether the power supply is correctly connected.

```
hedchip<CR>                                   ; ← user input
hed.chip - universal device programmer, Version 2.62

Test LPT1 - hed.chip found – not ready
```

Connect the power supply now.
Set the polarity switch of the power supply.
Set the voltage selector to 12V.

The program will continue automatically if the power supply is connected.
Cancel: press Escape

The power supply can now be connected. The Software keeps testing whether the power supply is connected. The following message will appear once all the connections and adjustments are correct:

Test LPT1 - hed.chip found - ready
Repeat test (Y/N)
Hardware-Version: 0002

The programmer hardware version (in this case: 2) is displayed.

When using Windows NT, **hed.chip** is mostly found on LPT2, even if the machine only has one LPT port. This is normal and there is no reason for concern.

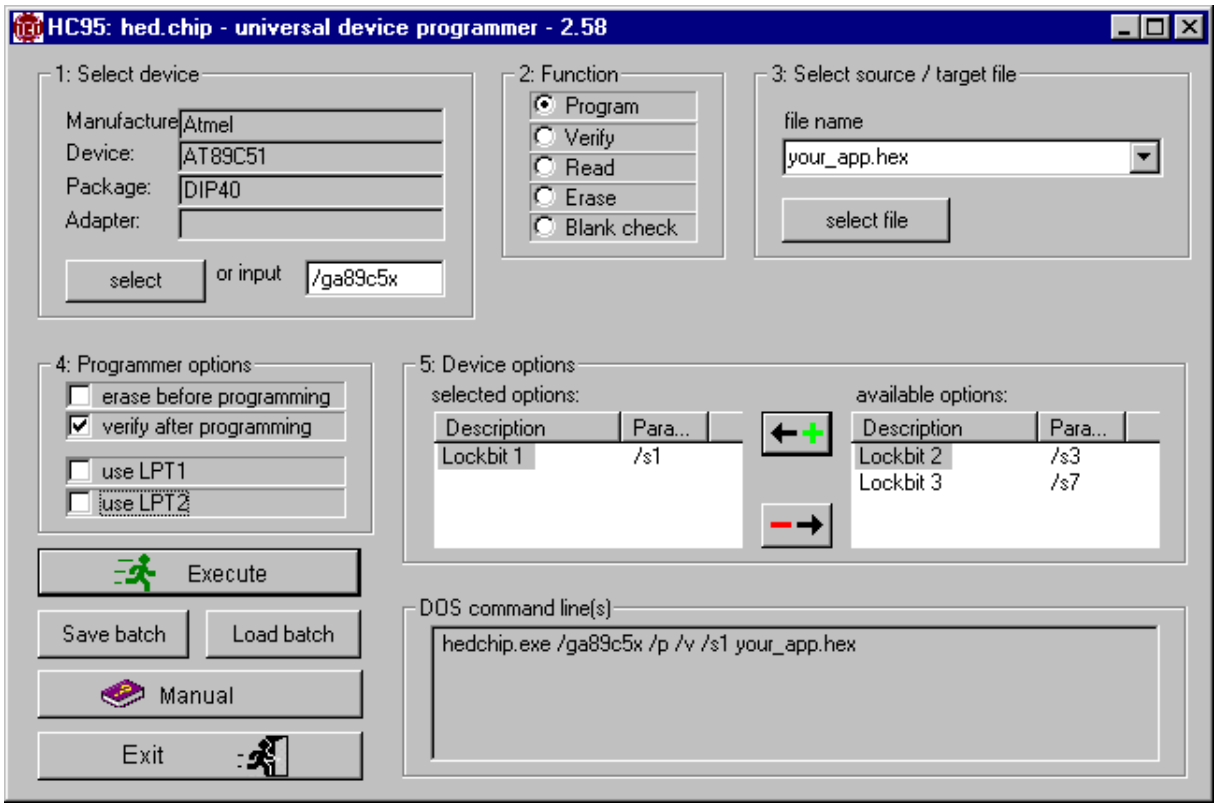
1.6 Operation

The DOS program HEDCHIP.EXE does the actual programming. It converts the input from the user and the source data file into commands for the programmer. Nevertheless, nobody is forced to concern themselves with the fine details of the DOS command line.

hed.chip can be operated in three different ways:

1. Using the Windows program HC95. A graphical user interface with a database allows you to select devices, files, and parameters. HC95 builds up a DOS command line from the user's input. This command line can then be executed automatically. Messages and help files are available both in German and English and the appropriate language is displayed according to the computer setting.
2. The DOS command line created by HC95 can also be saved in a batch file. Batch files can be executed on the same machine or on another machine. The HC95 program settings can be read back by loading a batch file produced by HC95. This is used to repeat device programming precisely.
3. Using HEDCHIP.EXE at the DOS or Windows 95/98/NT command prompt.

Windows: HC95

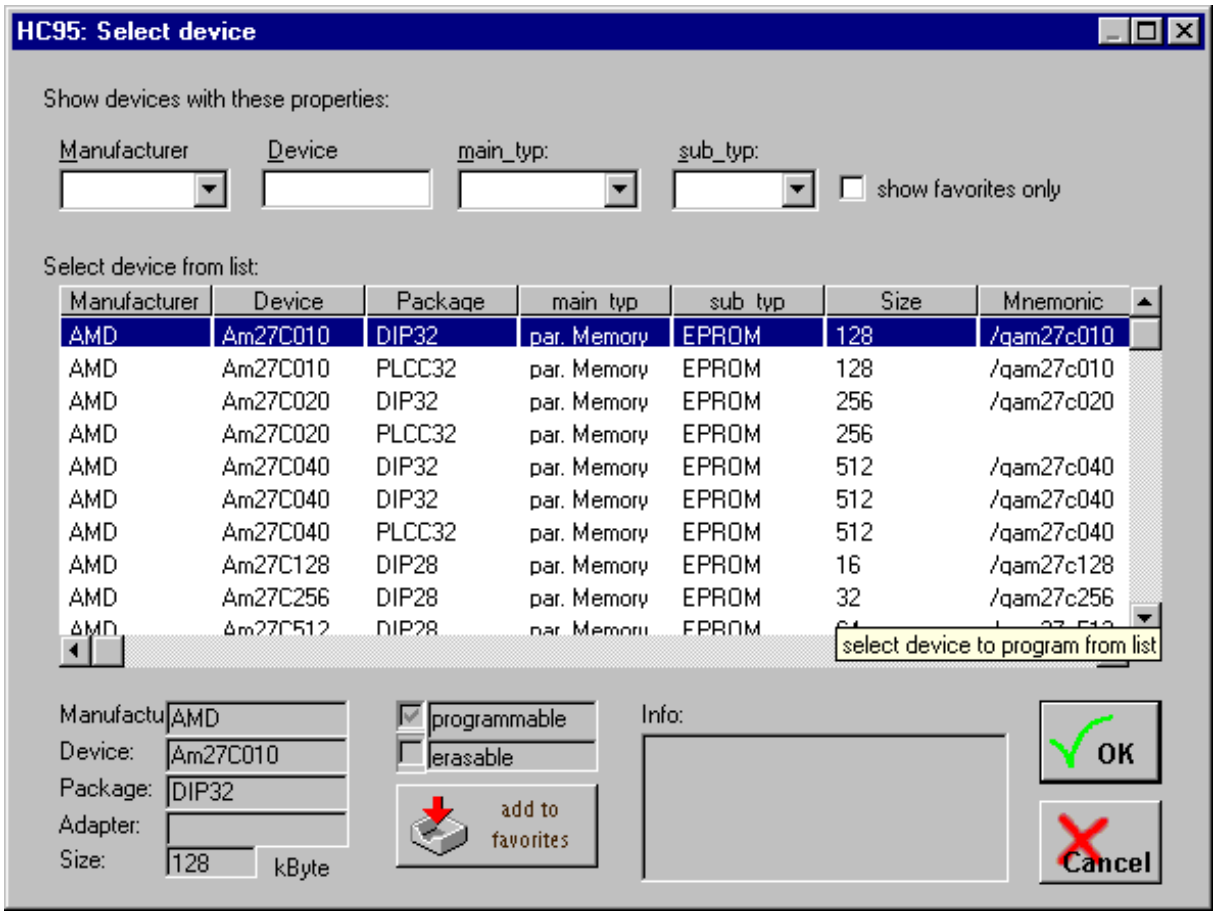


The HC95 main window

The picture above shows the HC95 graphical user interface. The controls are grouped together in a logical sequence.

1. Device selection: a click on the 'select'-button will open the device selection window. More information below. Advanced users who are familiar with the command line can input the device mnemonic into the edit field, although this disables any input validation checks by HC95.
2. Function selection: here you can select whether the device is to be programmed, verified, read, erased, or blank checked.
3. Source or target file selection: here you can select a source file for programming or verification, as well as a target file for saving the contents of a device.
4. Programmer options:
   - 4.1. Erase before programming: HEDCHIP.EXE always carries out a blank check before programming. If this option is activated, a non-blank device is automatically erased.
   - 4.2. Verify after programming: this option makes the programmer carry out verification automatically after programming (comparison with the source file).
   - 4.3. Use LPT1, use LPT2: checking one of these options disables the automatic detection of the programmer on one of the printer ports (Plug and Prog). This can be useful to avoid conflicts with other hardware and software.
   - 4.4. Additional Options can be set using the menu "Extras\Additional Options. These options set when blank checks are executed-
5. Device options: some devices have additional features such as write-protection or read-protection. The available options can be selected from the listbox.

Device selection:



HC95 has a database with all programmable devices and their properties. The device to be programmed must be selected from the list in the middle of the window. The properties of the selected device are displayed in the boxes under the list. The list is very long and it would be rather tedious to scroll from A (as in AMD) to X (as in Xicor). Therefore, the list can be reduced by entering selection criteria in one or more of the four fields above the list. Manufacturer, device name and type can be used as a criteria. Any combination, including the unrestricted use of wild cards '?' and '*', is possible. For example: "*28F*" in the device field leads to the display of all FLASH devices whose name includes the letters "28F". The '?' replaces exactly one character and the '*' replaces any number of characters in the string.

Favorites: frequently used devices

As an alternative to searching through the database, frequently used devices can be marked as favorites.

The button can be used to add a device to the list of favorites or remove it. The button offers the appropriate choice depending on whether the device is already a favorite.

Checking the option 'favorites only' will display a personal selection of devices.

Programming with HC95

After making all the necessary selections, HC95 displays the created command line in the box 'DOS command lines(s)'. For devices that require special handling there may be more than one line.

The command line can be executed by pressing the 'Execute'-button. This opens a DOS window in which the command is executed. The window closes automatically and a message regarding the success or failure of the operation is displayed.

The command line can be saved as a batch file by pressing 'save batch'.

Windows: working with HC95 batch files

The created batch file can, for example, be used on another machine. This makes sense if the other machine does not have Windows. The HC95 program settings can be read back by loading a batch file produced by HC95. This is used to repeat device programming precisely.

Windows: HC95 default settings

Default settings help to make the work easier. If you have hed.chip connected to LPT2, you might want to automatically load this parallel port setting as a default setting when you start HC95.

When starting HC95, the name of a batch file produced by HC95 can be supplied. If no file name is given, the settings stored in DEFAULT.BAT are loaded.

Save default settings

If you want the current settings to be restored when next starting HC95, use the 'save batch' function to save the settings under the name DEFAULT.BAT in the HC95 directory.

Removing unsuitable or incorrect default settings

Delete the DEFAULT.BAT file in the HC95 directory. The next time HC95 is started a new DEFAULT.BAT file will be automatically created.

Using different settings

If you have to carry out certain programming tasks repeatedly, you can store the necessary settings in several files. You can create links to HC95 and supply the name of such a batch file on the desktop or in the start menu.

You should: set up a link to HC95.EXE on the desktop. Then edit the link properties. Insert a space behind "<path>\HC95.EXE" and then the name of your batch file.

Windows: Check sum calculation

When programming or reading a device, the software calculates a check sum. This check sum can be used to quickly verify a software version contained in a programmed device. For that to work you need to note the check sum displayed after programming in your project documentation. The following algorithm is used to calculate the check sum:

```
for (int i = 0; i < len; i++)
    sum += (unsigned char)(*(pData +i) + 1);
```

The individual bytes are first incremented and the added up to a 32 bit check sum. While incrementing, 255 overflows to 0. This makes 255 neutral to the check sum. An empty device has the check sum 0. The unprogrammed parts of a partially programmed device have no influence on the check sum.

This method to calculate the check sum fails when programming devices that have additional configuration information. Microchip PIC microcontroller and Lattice GAL fall into this category of devices.

DOS: HEDCHIP.EXE

If you are working with DOS, or want to integrate the programmer into your development system, you can work directly with HEDCHIP.EXE. The DOS return codes needed for including HEDCHIP.EXE in your own development system are listed in Chapter 3.

The operation of HEDCHIP.EXE is carried out by using parameters in the command line. No further user interaction is necessary. All parameters start with a forward slash '/' (=Shift-7). Only the file name of the source file is entered without the forward slash as the final parameter in the command line. All other parameters can be entered in whatever order you prefer. HEDCHIP.EXE interprets the first parameter without a forward slash '/' as a file name, and stops processing of the command line.

The command line must always contain the following elements:

Device mnemonic	/gMNEMONIC; eg. /ga16v8, /gi87c5x, etc.
Command parameter	/? Display command overview
	/b Display device list
	/d Direct mode (suppresses keyboard queries)
	/e Erase device
	/l Blank check
	/p Program device
	/r Save device content in file
	/v Compare device with file

For the command parameters /p (program), /r (read), and /v (verify), a file name of the source or target file is required.

HEDCHIP.EXE processes the following file formats: JEDEC (*.JED), Intel HEX (*.HEX), Motorola S-Record (*:MOT) and binary data (all other file names).

Suitable JEDEC files can be created using CUPL, GAL Development System GDS3.5 or easyABEL. Check sums contained in these files are not evaluated. The complete file name, including the extension eg .JED, .HEX, or .BIN, must supplied. Files with the extension .JED, .HEX or .MOT are automatically converted to binary data format. All the other extensions are interpreted as binary data and directly programmed into the device without being converted.

In addition to the command parameter /p (program), the following optional parameters can be supplied:

/e	Erase device if not blank
/sn	Program security bits. For 'n', the number of the bit to be programmed must be inserted, eg: /s1, /s2, /s3
/v	Verify programming or erasure
/n	no blank check before programming
/m	When used with /p/v/e: no blank check after erasing a device.

HEDCHIP.EXE automatically finds the port that the programmer is connected to. Automatic detection can be disabled by using one of the following parameters:

/lpt1	hed.chip on LPT1
/lpt2	hed.chip on LPT2

With the command parameter /r the device content is saved in a target file. Simple PLDs, 16V8, 20V8, 18V10, 22V10 and 20RA10 are saved in JEDEC files. A file name with the extension .JED

must be supplied so that this file can be programmed into another device. Data from all other devices, complex PLD, microcontrollers, and memory devices is saved in binary files. In this case, a file name with the extensions .JED, .HEX or .MOT must not be used.

Existing files are overwritten without warning.

If the source file is too large for the device used, there is no error message. **hed.chip** always uses the minimum of device memory and file size for the program and verify operations.

The direct mode is activated by using the /d parameter. This parameter is intended for use in batch files. It suppresses any user interaction. The software bypasses any 'press any key' situation and any 'yes/no/cancel' queries are automatically answered with 'no'. When setting up batch files, you should first test the batch without the /d parameter. If all possibilities have been tested (with and without inserted device, blank and programmed device), you can eliminate annoying keyboard queries by using /d parameter.

1.7 Information regarding Windows NT4.0

As with Windows 95, programming can be done using the Windows graphical user interface or the command line. HEDCHIP.EXE automatically recognizes the operating system. For functions which require direct hardware access, drivers are automatically loaded and unloaded.

These drivers have been specially designed for Windows NT. **hed.chip** is completely compatible with the Windows NT operating system. Details about where to find files and what registry keys are written during installation can be found in the Windows help for HC95.

Selecting the parallel port

When using Windows NT, you should not supply the port number for the programmer. You will find that on most machines the software locates the programmer on LPT2, even if your machine has only one LPT port. The Windows NT virtual DOS environment always has support for three or four LPT ports.

1.8 Programming of read-protection and/or write-protection

Read-protection provides protection against non-authorized copies of the software for microcontrollers and PLD devices. With MCS51 microcontrollers, this multi-level protection is called 'lock bits'; PLD devices have a 'security fuse' for this purpose.

Many electrically erasable memory devices (FLASH, EEPROM) have a write-protection, thus preventing a crashed processor accidentally altering the content of the memory device. Depending on the type, parts of the device or the complete device can be protected. Some forms of the write-protection are irreversible, whereas others can be deactivated.

The protection features of the various devices differ widely. Explanations are available with descriptions of the respective devices.

The /s parameter is used for programming write-protection and read-protection. The parameter is given optionally for programming and, in doing so, the desired protection level is given as a number. Some examples:

```
hedchip /gl22v10 /p /v /e /s1 myapp.jed      ; Lattice GAL22V10 erase, program,
                                             ; verify and read protect.
hedchip /ga89c5x /p /v /e /s7 myapp.hex      ; Atmel controller erase, program, verify, and program
                                               all 3 lock bits.
```

Of course, read-protected devices can neither be read nor copied by **hed.chip**. Depending on the device, the following occurs:

- the device is recognized and also the activated protection is detected. **hed.chip** outputs the corresponding messages.
- the device is recognized and appears to be blank. Before programming, such devices must be erased by using HEDCHIP.EXE only with the /e command parameter.
- the device cannot be identified. With such devices, HEDCHIP.EXE asks the user if he wants to continue anyway. Until now, this has only applied to Atmel AT89C5x series controllers.

Some memory devices have several write-protection features. Some protection features can only be activated in sequence, whereas others are available independently. You can find out what protection features are available and how they are activated by reading the description of the respective device.

Write-protected memory devices can be read and copied by **hed.chip**. To reprogram a write-protected device, it must be erased beforehand. This applies even if only a part of the device is protected.

The /s parameter is also used to program other special features of certain devices. It is used to set the 'polarity option' with Atmel series AT17C FPGA configuration memories.

1.9 Insertion of devices

The device to be programmed can be inserted at any time. Please refer to the picture on the right for orientation and positioning of the device.

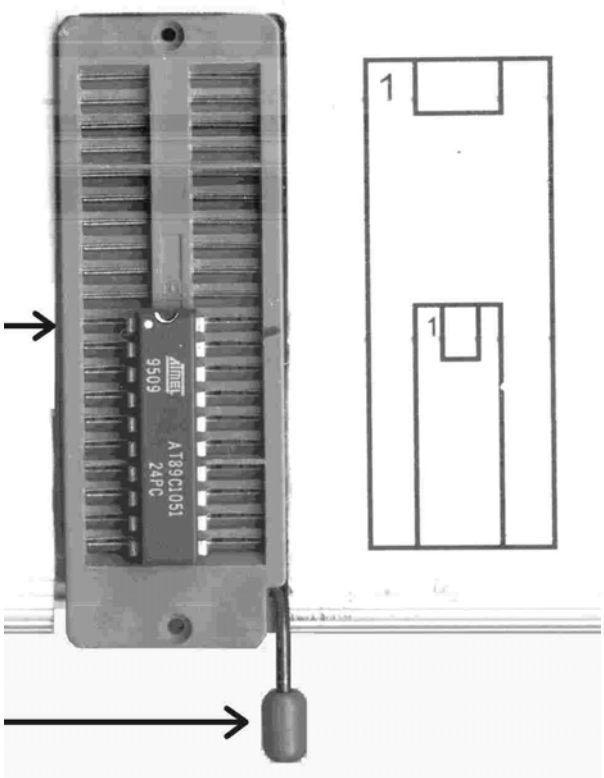


Photo:
Test socket of programmer with Atmel AT89C1051-24PC inserted.

1.10 Adapter

Adapters are necessary for devices in SMD packages, eg. PLCC or SOIC. Adapters for microcontrollers in PLCC44 packages or memory devices in PLCC32 packages are available on the market. Adapters that connect the signals from the DIP40 test socket to the corresponding pins in the PLCC package should be selected. In the case of parallel memory devices (EPROM, EEPROM, FLASH), the software was designed so that all these devices can be programmed using a DIP32 to PLCC32 adapter. The device mnemonic appropriate for both the device and its package must be used. If desired, you can also build such general adapters yourself. These adapters are named according to the

device package. A combination with two test sockets is available for the two most widely used PLCC types: memory devices in PLCC32 and MCS51 microcontrollers in PLCC44. As the same circuit board is used in each case, you can also add the second test socket yourself.

You cannot build adapters which adapt the programmer to special requirements of certain devices yourself. The names of these adapters are taken from the package and the devices they are used for. An Atmel ATV750 in the DIP24 package is programmed using the DIP750 adapter, and the PLCC750 adapter is used for the corresponding PLCC package.

Insertion of the adapter into the programmer

Most adapters have markings which show how the adapter should be inserted into the test socket of the programmer. The following rule applies for all adapters with DIP test sockets. The locking levers of the adapter test socket and the programmer test socket must point in the same direction.

1.11 Homemade adapters

Adapters for the PLCC packages of devices, for which there are also corresponding DIP packages, can be homemade. A function guarantee for these adapters, or those purchased from other sources, is not available.

1.12 Device identification and selection of the correct device mnemonic

Definition of mnemonic: System designed to aid memory. Within the hed.chip software, devices are identified by numbers. The device mnemonics are identifiers which stand for specific devices.

hed.chip attempts to identify the device in the test socket of the programmer. It will reject any device that does not correspond to the device mnemonic given in the command line. This serves to protect the valuable devices. Protection is not absolute, as the supply voltage and, in some cases, also the programming voltage must be applied for the identification check. It is therefore very important to use the correct mnemonic. If a device cannot be identified or programmed, this could be due to an activated protection feature of the device, despite use of the correct mnemonic. Most protected devices are either no longer identifiable or appear to be blank. Details concerning specific devices can be found in Chapter 2.

hed.chip displays a list of available mnemonics if the program is just used with the /b parameter. eg:

```
hedchip /b                                  ; displays list of mnemonics
```

2 Device details

2.1 Programmable logic - about PLDs and GALs

The term PLD (= Programmable Logic Device) can be applied to a wide range of devices. It applies to devices ranging from simple TTL-PROMs up to gate arrays, into which complete processors can be programmed.

The term GAL is a protected copyright of the Lattice company. These are simple PLDs, sometimes also called SPLD. Devices with more functionality are called complex PLDs, or CPLD.

hed.chip programs a selection of popular and versatile devices. Here is a list of programmable devices that were available when this manual was printed:

ATF16V8	ATF20V8	ATF22V10	
GAL16V8	GAL20V8	GAL18V10	GAL22V10
GAL6001	GAL6001B	GAL6002B	GAL20RA10
PALCE16V8	PALCE20V8	PALCE22V10	

The complex PLDs from Atmel (ATV750, ATV2500) make particular demands on the programmer hardware. Adapters DIP750, PLCC750, DIP2500, PLCC2500, PLCC1500 respectively are used for these devices.

To develop an application for a PLD, a JEDEC file must be created. The PLD development system, eg CUPL, Gal Development System GDS 3.5 or easyABLE Version 4.3, converts the logic equations into such a JEDEC file. It can also simulate the expected behaviour of the PLD.

Check sums and test vectors in the JEDEC file are ignored. You can edit the JEDEC file with a normal text editor, if desired.

If the logic equations are contained in a file named MYAPP.PLD, CUPL creates file MYAPP.JED out of this. This can then be programmed into the device using **hed.chip**. In the case of an application for ATF22V10, the following command line must be used.

```
hedchip /ga22v10 /p myapp.jed
```

By using the additional /v parameter, the programmer will verify the programming operation. By using /e, the device is erased if the blank check fails.

```
hedchip /ga22v10 /p /v /e myapp.jed
```

Security fuse

If the device is to be protected against reading and copying, the security fuse can be programmed. With the CUPL development system, the instruction to do this can be given when the JEDEC file is created.

CUPL then inserts an instruction *G1 into the JEDEC file, causing **hed.chip** to program the security fuse of the device. If this is not desired, you can either remove this instruction from the JEDEC file, or use the additional /s0 in the command line. The /s1 parameter in the command line programs the security fuse, even if the JEDEC file contains the instruction *G0.

Command line parameters take precedence over instructions in the JEDEC file, whether or not the security fuse is to be programmed.

With some PLDs, **hed.chip** can test the security fuse, and will produce an error message if an attempt is made to read a protected device.

Other protected PLDs appear to be blank. Such devices can also not be erased by using /e additionally when programming. In this case, the device must be erased in a separate operation:

```
hedchip /gl22v10 /e                    ; example for GAL22V10
```

Copy devices

With **hed.chip**, PLDs can also be read and copied. For simple PLDs (16V8, 20V8, 18V10, 22V10, and 20RA10), **hed.chip** creates a JEDEC file similar to the one created by CUPL. When reading these devices, a file name must be used with the .JED extension. Complex PLDs (GAL6001/2, Atmel ATV-Serie) are read using a binary data format. For these devices, a .JED extension must not be used in the file name. **hed.chip** can program these binary files in other devices of the same type. eg:

```
hedchip /gatv750 /r myapp.bin          ;Atmel ATV750 into file MAYAPP.BIN
hedchip /gatv750 /p/v myapp.bin        ; program other device of the same type
```

ATF16V8, ATF20V8, ATF22V10

Atmel specifies that these PLD devices with FLASH memory technology have to be conditioned prior to initial programming. This means that the whole device is completely programmed twice with 0, and erased again afterwards. Verifying errors can be ignored during conditioning. A JEDEC file suitable for conditioning is part of the software supplied with **hed.chip**. Example for ATF20V8:

```
hedchip /ga20v8 /p/e conditio.jed      ; program once
hedchip /ga20v8 /p/e conditio.jed      ; program twice
hedchip /ga20v8 /e/v                   ; erase, blank check
```

The command for conditioning can be included in a batch file, which is recommended anyway.

ATV750(B) and ATV2500(B):

Special adapters, DIP750, DIP2500, PLCC750, PLCC2500 respectively, are required for the devices ATV750(B) and ATV2500(B). There are two jumpers on these adapters. For programming ATV750 and ATV2500, both jumpers must be set and removed for programming ATV750B and ATV2500B.

The DIP750 adapter is also used for AT22V10/L and AT22V10B/L devices. This is an older version of the type 22V10 based on EPROM memory technology.

AMD PALCE-series

hed.chip supports PALCE16V8H/Q and PALCE20V8H/Q. PALCE22V10H/Q in revisions 4 and 5 is supported. These appear in catalogues as PALCE22V10H-25PC4. You have to erase all PALCE devices before programming, even if they are new or blank.

AMD PALCE16V8 and PALCE20V8 are mostly compatible with the corresponding devices from Atmel and Lattice. There is a small difference in the way the register outputs are fed back into the AND-matrix. In most cases, source files created for GAL16V8 and GAL20V8 can be programmed into a PALCE device without making any changes.

To avoid any incompatibilities, the correct target device (GAL16V8, GAL20V8, or PALCE16V8, PALCE20V8) should be used when creating the JEDEC source file. The differences between these devices were described in the 1/94 German issue of Elektor magazine on page 52.

PALCE22V10 can be substituted for GAL22V10 without restriction, and the same JEDEC files can be used in development and programming.

2.2 MCS51 microcontroller

hed.chip can program almost all CMOS MCS51 versions from AMD, Atmel, Dallas, Intel, Philips, Siemens, SST, Temic and Winbond. Adapters are available for controllers in the PLCC44 and SOIC20 packages.

The correct device mnemonic must be used in the command line. hed.chip validates the manufacturer and device ID of the controller, and then automatically selects the correct programming algorithm.

If an application, eg for a Philips 87C52, is to be developed, an Intel Hex file or a binary file must be created using a cross assembler or cross compiler respectively. hed.chip can then program this into the controller.

```
hedchip /gp87c5x /p/v myapp.hex
```

In the above example, it is necessary to use the /gp87c5x mnemonic, and not /gi87c5x for Intel controllers. If the manufacturer ID does not match the mnemonic, hed.chip will reject the device. The same applies if the device ID is not known to hed.chip. We will create software updates for new devices as quickly as possible.

Please note: there is no error message or warning if the source file is too large for the controller's memory.

Lock bits

MCS51 controllers have two or more so-called lock bits for protection:

Parameter	Lock bits	Function
S1	1	Protects against further programming
S3	1 + 2	Protects against reading of the program memory. As it is still possible to run programs in external memory, the protection is not 100% secure. S1 is contained in S3.
S7	1 + 2 + 3	Prevents programs being run from external memory. With lock bit 3 set, the state of the EA# pin is without significance. S1 and S3 are contained in S7. Not all devices have this lock bit.

When S2 or higher protection is used, the programmer will no longer recognize the device because the manufacturer ID can no longer be read, or the device will appear blank. An attempt to program such a device which seems to be blank leads to an error message 'device not programmable'.

If a device is not accepted by hed.chip, or cannot be programmed, it should be erased. For the lock bits, the /s1, /s3, or /s7 parameters must be used in addition to /p in the command line. The higher lock bits automatically include the lesser lock bits; /s7 therefore programs all lock bits for all types of MCS51 controllers.

```
hedchip /gp87c5x /p /v /s7 myapp.hex
```

Some MCS51 devices have an encryption array. Programming of this protection measure is not supported by hed.chip, since, as far as we know, there is no meaningful application for this.

Atmel AT89C** series controllers

can be erased electrically using **hed.chip**. Also the smaller versions in the DIP20 package can be directly inserted into the test socket of **hed.chip**. These devices can be erased and reprogrammed in one operation:

```
hedchip /ga89c5x /p /v /e myapp.hex          ; for AT89C51/2, LV51/2
hedchip /ga89cx051 /p /v /e myapp.hex        ; for AT89C1051/2051
```

If any lock bits are set, the erase operation must be done separately.

The /ga89c5x-5 mnemonic is used for AT89C5x version with 5V programming voltage. AT89LV** may be programmed using the same settings.

AT89C51RC und AT89C55WD

These devices are programmed using the device mnemonic /ga89c5x2. Read-protected devices must be erased in a separate call of the programming software.

```
hedchip /ga89c5x2 /e                         ; Erasure (required for read-protected devices)
hedchip /ga89c5x2 /p /v /s7 myapp.hex        ; Program, Verify, Read-Protect device
```

Atmel AT89S** series controllers

can be erased and programmed using **hed.chip**. **hed.chip** can also activate the SPI security fuse and program the EEPROM data memory of the AT89S8252. Two separate mnemonics are used to program the FLASH program memory and the EEPROM data memory. The protection can only be used in connection with the /gs89sxxxx mnemonic. The erase operation always affects both memories(FLASH program memory, EEPROM data memory), and deactivates the protection features.

```
hedchip /ga89sxxxx /p/v/e/s7 myapp.hex       ; erase, program, verify, write-/read-protect
                                               AT89S8252 or AT89S53 FLASH
hedchip /ga89sxxxx /p/v/e/s15 myapp.hex      ; ditto, program SPI security fuse
hedchip /ga89sxxxx /p/v/e/s7/s8 myapp.hex    ; ditto, /s7/s8 corresponds to /s15
hedchip /ga89seeprom /p/v myapp.hex          ; program, verify AT89S8252 EEPROM
```

If program memory and data memory are to be programmed and protected, the following sequence has to be used: erase device, program EEPROM data memory, program and protect FALSH program memory. This also protects the data memory.

AT89LS** can be programmed using the same settings.

Atmel/Temic (A)T89C51R*2 series controllers

These controllers have special options in the HSB register. After erasure these option have been reset to the default state. Temic has been purchased by Atmel and Temic products are being integrated in the Atmel product line. Some devices are fully identical while other devices differ (eg.: Atmel AT89C51RD2 and Temic T89C51RD2).

Option	Unprogrammed	Programmed
Lockbit 1	MOVC is enabled in external code	MOVC is disabled from external code
Lockbit 2	Program memory can be read using a programmer.	Reading the program memory is inhibited
Lockbit 3	Code execution in external memory is allowed.	Code execution in external memory is inhibited
XRAM	XRAM is activated	XRAM is deactivated

Option	Unprogrammed	Programmed
OSC	OSCA is activated (AT89C51IC2)	OSCB is activated (AT89C51IC2)
BLJB	Program execution starts at 0x0000	Program execution starts at 0xFC00 (boot loader)
BLLB	Programming of boot loader segment allowed (T89C51RD2)	Programming of boot loader segment inhibited (T89C51RD2)
X2	Standard mode (12clk)	X2 mode (6clk)

The user interface offers programming of the options available in the selected controller. The contents of the HSB is copied into the XAF at address 0x0004 where it can be accessed programmatically.

Philips 87C7** series controllers

The DIP752 adapter is required for the 87C749 and 87C752 Philips controller. Please note that the 87C748, ‘749, ‘751, and ‘752 controllers are programmed using /gp87c7xx mnemonic, however the 87C750 is programmed using the /gp87c750 mnemonic. Controllers made by Signetics are handled like Philips devices.

Philips P89C5** series controllers

Of this series, hed.chip supports the types P89C51Uxxx, P89C52Uxxx, P89C54Uxxx, P89C58Uxxx, P89C51RC+, -RD+, P89C51RB2, -RC2, and RD2. All of these microcontrollers have the usual 3 lock bits.

In addition, the P89C51RC+, -RD+, -RB2, -RC2, and -RD2 have a programmable status byte. The status byte is programmed by using the command line parameter /s8 additionally when programming this device.

```
hedchip /gp89c5x /p/v/e/s15          ; erase, program, verfy, program all 3 lock bits and
                                       program status byte of P89C51RC+
```

When these devices are erased, the status byte is also erased and the boot vector is set to the factory default value.

In addition, P89C51RB2, -RC2, and RD2 have a 6x clock mode. In this mode, the process takes 6 clock cycles per machine cycle. In other words, it is running twice as fast at the same clock frequency. By default, this controller is in 6x clock mode. It can be set to 12x clock mode by using the command line parameter /s16 additionally when programming this device. Setting the device to 12x clock mode is a one-time operation. Once programmed, the device cannot be changed back to 6x clock mode.

Dallas High Speed Controller DS87C520/530

The DS87C520 and ‘530 devices have a watchdog timer. This watchdog timer can generate a RESET. The watchdog timer runs continuously, but a RESET is generated only if the corresponding function is enabled. The device must be programmed using the /gd87c5x0-w mnemonic if the RESET by watchdog timer function is to be enabled automatically after a RESET.

Siemens SAB-C513A

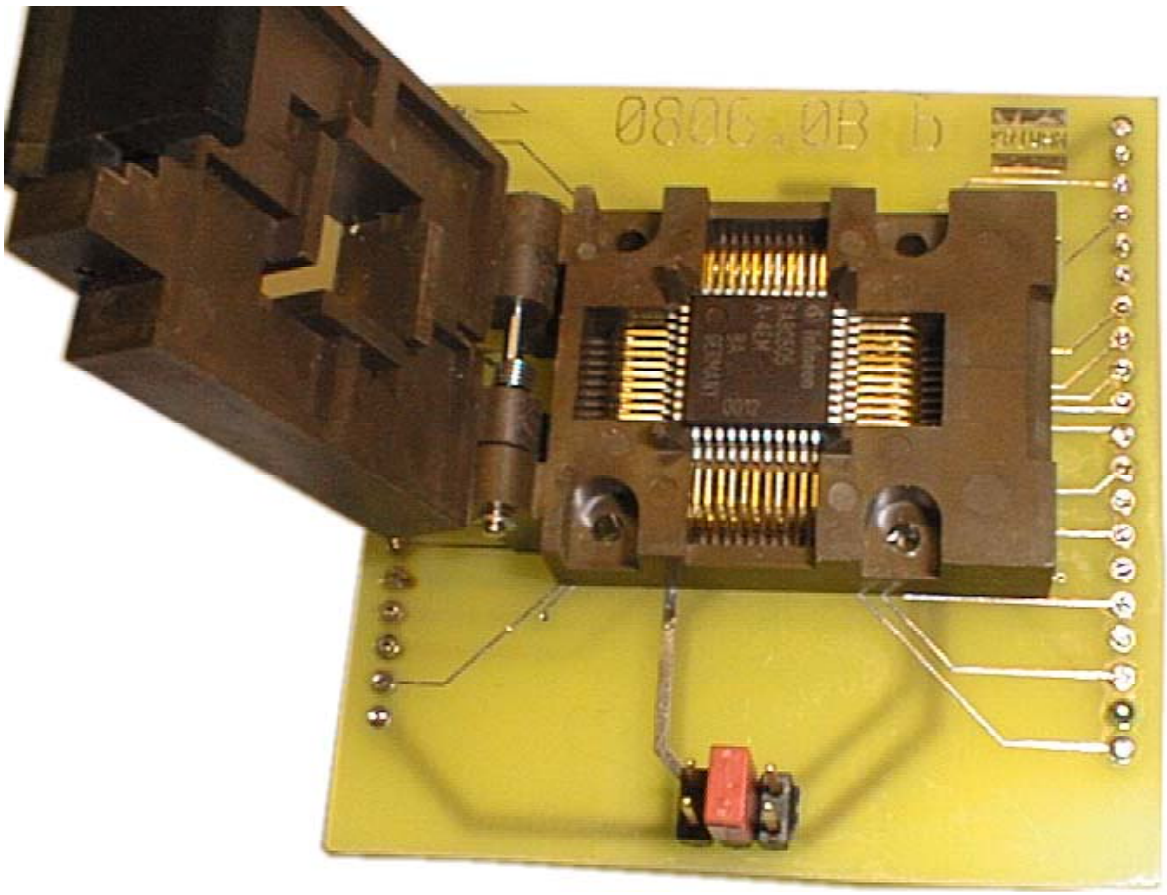
hed.chip also supports the Siemens SAB-C513A-H device. This Siemens device is only intended for development. It has no lock bits. When using devices with the "ES-BA" marking, the Siemens Errata Sheet, Release 1.2 from 20th Sept. 1995 has to observed.

Siemens C505A-4E, C505CA-4E

These Siemens microcontrollers have 32 kByte of OTP EPROM memory (OTP = One Time Programmable). They are available in the PQFP44 package only.

The pin functions differ from the usual assignment. Pins 38/39 are used for analog power of the AD-converter. Pin 17 is used for the digital power supply.

Because of this a special PQFP44_C505 adapter is required.



Picture: Note the orientation of the C505A device and the setting of the jumper.

SST89F5*

The MCS51 microcontrollers made by SST have a unique feature. They have two separate blocks of FLASH program memory. Block 0 is the primary memory, and has 16 kbytes (SST89F54) or 32 kbytes (SST89F58). Block 1 has 4 kbytes and is located at address 0xF000. The unique feature is that the microcontroller can write to its own FLASH program memory. This is referred to as 'In-Application Programming' in the data sheet.

The /gsst89f5x_0 device mnemonic is used for programming block 0.

The /gsst89f5x_1 device mnemonic is used for programming block 1. When programming source files in Intel Hex format, care must be taken that the source file does not contain an offset. To do this, use the assembler instruction '.phase 0xf000' instead of '.org 0xf000'.

The erase operation always erases both memory blocks.

These microcontrollers have no lock bits. Instead, the read-protection and write-protection is determined by the content of the byte at address 0xFFF of memory block 1. This byte is called 'Security Byte'. Write-protection is advisable to protect against unintentional memory changes.

Parameter	Sec. Byte	Function
/S0	0xFF	No protection
/S85	0x55	Both FLASH memory blocks are protected (hard lock)
/S245	0xF5	Only block 1 is protected (hard lock)
/S5	0x05	Both memory blocks are protected, but can be programmed using In-Application Programming (soft lock).

Protection can either be activated through the content of the source file for block 1, or through parameters in the command line. Parameters in the command line take precedence over values for the security byte in the source file. If both memory blocks are to be programmed, protection can only be activated when programming the second memory block.

Temic TSC87C51

This microcontroller has no lock bits. As hed.chip cannot program the encryption array, there is no protection against reading the controller memory. The encryption array does not offer effective protection anyway.

Temic TS87C52X2

At the same clock frequency, this microcontroller is twice as fast as ordinary MCS51 microcontrollers. It has the usual 3 lock bits to protect the software against reading and copying.

2.3 Atmel AVR-RISC microcontroller

This new microcontroller family is based on an upgraded version of the MCS51 family periphery, and a newly developed processor core. The processor was optimized to support the programming language 'C', but also offers a user-friendly assembly language.

The AVR-RISC controllers have FLASH program memory and EEPROM data memory. They can be programmed using parallel access with a programmer or using serial access IN-CIRCUIT. Both memory types and the device options can be programmed using **hed.chip**. The erase operation is common to both memory types. This means that erasing the FLASH program memory also automatically erases the EEPROM data memory, and vice versa.

The FLASH program memory must be blank prior to programming. The EEPROM data memory can be reprogrammed without previous erasure.

When programming the FLASH program memory, /gavr20 and /gavr40 mnemonics are used for the devices in DIP20 and DIP40 packages respectively. When programming the EEPROM data memory, /gavr20e and /gavr40e mnemonics are used for devices in the DIP20 and DIP40 packages respectively.

Lock bits and fuses

Using lock bits, the FLASH program memory can be protected against alteration and reading. Further options can be set using two fuses. These can only be programmed, using a device programmer and not in circuit using serial access via SPI. The /s parameter can be used to activate these options following programming of the FLASH program memory.

Parameter	Lock bits	Function
S1	1	Protects FLASH program memory against further programming
S3	1 + 2	Protects FLASH program memory against reading. S3 includes S1.
S4	RCEN	AT90S1200/2313: activates using the internal oscillator of the watchdog timer as clock source for the processor. Without an external crystal, the controller works at a clock frequency of approximately 1 MHz.
S4	SPI disable	AT90S4414/8515: disables serial in-circuit programming.
S8	SPI disable	AT90S1200/2313: disables serial in-circuit programming
S8	FSTRT	AT90S4414/8515: selects the short RESET-delay time after a power-on. This is intended for fast-starting clock sources.

These options can be used in any desired combination. You can either add up the values yourselves or use several /s parameters in the command line.

```
hedchip /gavr20 /p/v/e /s15 yourapp.bin          ; program lock bit, SPI disable and RCEN
hedchip /gavr20 /p/v/e /s3/s4/s8 yourapp.bin     ; does the same as using  /s15
```

If the FLASH program memory is read, **hed.chip** saves the state of the fuses (SPI disable, RCEN, ...) as the last byte in the target file. When programming a device with such a file, the fuses are also programmed accordingly.

ATtiny2313

The ATtiny2 is a successor to the AT90S2313 microcontroller. The ATtiny2313 has more and different fuses than the AT90S2313. These fuses can only be programmed from the source file. To do that the source file must contain a data word to be programmed into the fuses at address 2048 (= 0x800). When reading from a device the contents of the fuse word is stored at location 2048 in the target file.

Errata, 31.10.2006: The high byte of the fuse word cannot be read. The high byte is always read to be 0xFF. The attempt to program a value other than 0xFF into this location fails. It is unknown if the programming fails or if the immediate verification of that programming fails. Consequently, programming a value other than 0xFF will always result in a programming error but the programmed value may or may not be effective. If you read a new device into a file the fuse word will be read to be 0xFF64.

ATmega

While the devices of the ATmega are similar to the AT90S-devices, they have a lot more fuses and security options. The fuses can only be programmed with data in the source file. In the source file immediately following the code for the FLASH program memory there must be a byte (eg.: ATMEGA161) or a word (eg.: ATMEGA163) containing the value for the fuse register. When erasing a device, hed.chip restores the default factory settings for the fuses.

Example: ATMEGA161
Address: 0x4000
Fuse-Byte (8 Bit):

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	X	BOOTRST	SPIEN	BODLEVEL	BODEN	CKSEL(2)	CKSEL(1)	CKSEL(0)
Default:	0	1	0	1	1	0	1	0

The source file for the device ATMEGA161 including programming of the fuses has the size of 16385 bytes.

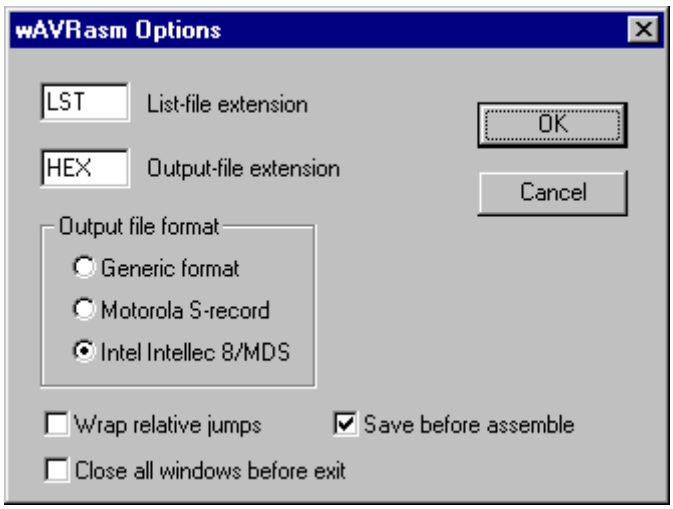
ATmega Lockbits

ATmega devices can alter their own program memory. The program memory is divided into a block called application block and boot loader block. Writing and reading these blocks can be disabled separately. Unnecessary write operations should be inhibited to avoid corruption of the program memory in the event of a software crash.

Parameter	Lockbits	Funktion
S1	LB1	Protects against further writing of the FLASH program memory by a device programmer.
S2	LB2	Protects against reading of the FLASH program memory by a device programmer.
S4	BLB01	Protects against programming the application block.
S8	BLB02	Protects against reading the application block.
S16	BLB11	Protects against programming the boot loader block.
S32	BLB12	Protects against reading the boot loader block.

Atmel AVR Assembler 1.30

This assembler for the AT90S series can be obtained free of charge from Atmel's web site. Source files intended for programming should be generated using the following settings:



The "Intel Intellec 8/MDS" format corresponds to the standard Intel Hex format. If an ".eseg" sector is created, the assembler outputs data intended for the EEPROM data memory of the controller. This file is automatically assigned a name with the file extension ".EEP", with no possibility of assigning a different file name.

The data format corresponds to the format for the program. Using the settings on the left, the assembler produces a source file for the EEPROM that is in Intel Hex format, but does not have the file extension ".HEX". This file has to be renamed, so that it can be correctly converted during programming. eg:

You assemble the file yourapp.asm. The following are created from this:

```
yourapp.hex                ; source file for FLASH program memory
yourapp.eep                ; source file for EEPROM data memory needs to be
                           ; renamed for programming.
```

The FLASH program memory is programmed using YOURAPP.HEX. The YOURAPP.EPP file is renamed as EEPROM.HEX. It is necessary to change the name so that the file is recognized as an Intel Hex file. Then the file is programmed into the EEPROM data memory of the controller. The option 'erase before programming' should not be activated. Note: the erase operation always erases both memories (FLASH and EEPROM) in the controller.

2.4 Microchip PIC microcontrollers

With the help of the UNIPIC adapter, **hed.chip** also supports a large number of PIC microcontrollers. **hed.chip** meets all requirements specified by Microchip for a 'production quality programmer'. The adapter has suitable test sockets for devices in the DIP18, DIP28, and DIP40 packages. A precision socket is used for devices in the DIP8 package. The adapter is available in two versions:

1. UNIPIC18: equipped with a DIP18 test socket and a DIP8 precision socket. You can add the test DIP28 and DIP40 sockets yourselves if necessary.
2. UNIPIC: fully equipped with the DIP18, DIP28, and DIP40 test sockets and a DIP8 precision socket.

PIC microcontrollers have either EPROM or FLASH program memory. Depending on the device type, a word consists of 12, 14, or 16 bits. In addition to the standard program memory, these devices have a configuration word and 16 bits memory for a user ID. The 16 bits of the user ID are contained in 4 memory words. When reading, the user ID and configuration word are also read. The target file is therefore always 10 bytes larger than the program memory of the device.

User ID

PIC microcontrollers have 16 bit user ID (Customer ID Code). This user ID is stored in a special address space in 4 locations. Each of them can be programmed with 4 bits of the user ID. **hed.chip** can take a user ID from the source file and program it into the intended location. In the source file, the user ID must be located after the program memory data. If the source file is at least 10 bytes larger than the program memory of the device, the last 10 bytes are programmed into the user ID and configuration word.

Example for PIC16C84: the device has 1024 words of program memory. This corresponds to a 2048 bytes source file. The following 4 words (= 8 bytes) are interpreted as user ID. The lower 4 bits of every word are programmed as user ID into the device. To avoid verification error messages, the upper 12 bits must be 0. A source file for PIC16C84 with user ID and configuration word has 2058 bytes.

Configuration Word

The configuration word is a memory location in a special address space used to configure the microcontroller. The bits of the configuration word set the clock generator to certain clock sources and influence the operation modes of the timers and the watchdog. Further bits are used to prevent the program memory being read. After erasing the device, all bits of the configuration word are set to 1. During programming device, options can be used to program one or more bits of the configuration word to 0. The following table gives an overview of the options and corresponding command line parameters that can be used for programming. When programming, the graphical user interface HC95 offers the available device options for the chosen device. You can select any combination of options.

Parameter	Option	Function
/S1 /S2	FOSC0 FOSC1	Oscillator Selection Bit. Options FOSC0 and FOSC1 can be used to select different modes for the clock generator of the microcontroller. RC Oscillator: default, neither Option FOSC0 nor FOSC1 is programmed. Controller is used with combination of resistor/capacitor. HS Oscillator: program FOSC0. Controller is used with high frequency crystal. XT Oscillator: program FOSC1. Controller is used with medium frequency crystal. LP Oscillator: program FOSC0 and FOSC1. Controller is used with low frequency crystal.
/S4	WDTE WDTEN	Watchdog Timer Enable. The watchdog timer is activated by default after a reset. Use option WDTE when programming to deactivate the watchdog timer.
/S8	PWRTE PWRTE# PWRTEN# PWRTEN#	PWRTE: Power Up Timer Disable Bit. With some devices, the timer is activated by default after a reset and can be deactivated using this option. PWRTE#: Power Up Timer Enable Bit. With some devices, the timer is deactivated by default after a reset, and can be activated using this option. HC95 offers the appropriate option for programming the respective device.
/S16	BODEN BOREN	Brown Out Enable Bit. Devices with this option can detect slow decreases in the operating voltage.
/S16	FOSC2	PIC12C67x only: Oscillator Selection Bit. This option may only be used in combination with FOSC0 and/or FOSC1.
/S32	CP CP0	Read-protection. Some devices have one option (CP) to read-protect the device. Other devices have two options for this; each of them protects half of the memory. CP0 protects the upper half.
/S32	LVP	Low Voltage Programming enable. When set, pin RB3 has PGM function. When reset, pin RB3 has IO function, for programming Vpp must be applied to MCLR.
/S64	CP1	Read-protection. Protects the lower half of the memory.
/S64	DP	PIC16CR83/84, PIC16F83/84: Data EEPROM read protection
/S64	BORV0	PIC16C773/4: Brown Out Voltage. Sets the voltage level for Brown Out Detection.
/S128	MCLRE	PIC12C508/9 only: Master Clear pin Enable Bit. Programming this option disables the MCLR pin. It is internally connected to Vdd.

Parameter	Option	Function
/S128	WRT	FLASH Program Memory Write Enable. Programming this option disables the controller's ability to reprogram its own FLASH memory.
/S128	BORV1	PIC16C773/4: Brown Out Voltage. Sets the voltage level for Brown Out Detection.

The configuration word can also be programmed from the source file. For this, the configuration word must be located directly behind the user ID in the source file. When reading these devices, the user ID and the configuration word are stored in the source file following the normal content of the device memory. Devices can be copied including the user ID and the configuration word. The exact layout for the bits of the configuration word depends on the specific device. There are differences, even if the devices have the same options.

Read-protection in UV-erasable PIC microcontrollers

Microchip recommends that microcontrollers in a windowed ceramic package should not be read-protected. This means that the /s32 and /s64 device options should not be used when programming these devices. In our experience, even high intensity UV light will not completely erase these devices when the read-protection has been enabled.

The read-protection can be activated by programming with a corresponding value for the configuration word in the source file. When doing this, an error message is generated stating that the device is not programmable. The read-protection can also be activated unintentionally when the device is programmed with a file not suitable for this type of device.

Data EEPROM

Some devices, eg.: PIC16F84, have a data EEPROM. This can be programmed by selecting the devices PICDATA64, PICDATA128 or PICDATA256 from the device list.

To program the 128 bytes of data EEPROM of a PIC16F870 select PICDATA128 from the list. On the command line, use the device mnemonic /gpicdata128.

The data EEPROM has a word size of 8 bits. But when writing the memory, words of 16 bits size must be loaded into the device. Of these 16 bits the lower 8 bits are programmed into the EEPROM.

The data EEPROM must be programmed before programming the FLASH or EPROM program memory. Follow this sequence:

1. Erase the respective device, eg PIC16F870. This also erases the data EEPROM.
2. Program the data EEPROM, for a PIC16F870 select PICDATA128 from the device list.
3. Program the FLASH or EPROM program memory. If required, read protect the device.

MPLAB development system

hed.chip supports this development system. Program data, User ID and Configuration Word from source files generated by MPLAB are programmed into the appropriate locations of the PIC microcontroller.

For this, the file format generated by MPLAB is converted to the HEDCHIP format. This conversions is done automatically by the HEXBIN.EXE program. MPLAB places the User-ID at different locations depending on the word size of the controller. There is a 12-bit format and a 14-bit format.

File Formats:

1. HEDCHIP format:
   HEDCHIP expects to find the User ID and the Configuration Word at the following locations.
   User ID at address following immediately the program memory. The exact location depends on the size of the program memory.
   Configuration Word at address following immediately the User ID.
   Example for a PIC16C84:
   Program memory: 0400h words = 0800h bytes = 2 kByte
   User-Id at address 0800h (byte-address)
   Configuration Word at address 0808h (byte address)
2. MPLAB 12Bit format:
   User ID at address immediately following the program memory. For a PIC12C508 this is the address 0400h (byte-address).
   Configuration Word at address 1ffeh (byte address).
   HEXBIN.EXE moves the Configuration Word from address 1ffeh to address 408h.
3. MPLAB 14Bit format:
   User ID at address 4000h (byte-address)
   Configuration Word at address 400eh (byte address)
   HEXBIN.EXE moves the User ID from address 4000 to the address immediately following the programm memory (eg.: 0800h for a PIC16C84). It moves the Configuration Word from address 400eh to address <program memory size + 8> (eg.: 0808h for a PIC16C84).

Recommended operating procedure:

Place all options except read protection in the source file using MPLAB. If you are using UV-erasable microcontrollers that require an oscillator calibration value, place the appropriate value into the source file. Do not program the read protection during the developement stage or if you are using UV-erasable microcontrollers. Do not use the source file to program the read protection.

For production use the HC95 device option list and select CP or CP0+CP1 to read protect the device. Options selected in the HC95 device option list and options configured using MPLAB are cummultative. Selecting CP in the HC95 programs "adds" the read protection to the options that are programmed from data in the source file.

Overview of the supported PIC microcontrollers

The following table lists the supported microcontrollers and states the device mnemonics to be used for programming. Some device mnemonics are used for several devices that do not differ with regard to their physical programming, eg: PIC16C61 and PIC16C71 are both programmed using the /gpic16c61 mnemonic. All devices are contained in the database of HC95 and the correct mnemonic will be automatically used for programming. The UNIPIC adapter has the appropriate socket for every microcontroller. Devices must be inserted into the socket that fits them exactly, eg: a PIC16C84 in DIP18 may only be inserted into the DIP18 test socket. PIC microcontrollers can not be identified by the programmer. It is very important to select the correct device. This applies especially to devices with or without ‘A’ as the last letter in the device name, eg: PIC16C62 and PIC16C62A are not identical.

Device	Package	Mnemonic	Programmable Options
PIC12C508/A	DIP8	/gpic12c508	FOSC0, FOSC1, WDTE, CP, MCLRE
PIC12C509/A	DIP8	/gpic12c509	FOSC0, FOSC1, WDTE, CP, MCLRE
PIC12C671	DIP8	/gpic12c671	FOSC0, FOSC1, FOSC2, WDTE, PWRTE, CP0, CP1, MCLRE

Device	Package	Mnemonic	Programmable Options
PIC12C672	DIP8	/gpic12c672	FOSC0, FOSC1, FOSC2, WDTE, PWRTE, CP0, CP1, MCLRE
PIC12CE518	DIP8	/gpic12c508	FOSC0, FOSC1, WDTE, CP, MCLRE
PIC12CE519	DIP8	/gpic12c509	FOSC0, FOSC1, WDTE, CP, MCLRE
PIC12CE673	DIP8	/gpic12c671	FOSC0, FOSC1, FOSC2, WDTE, PWRTE, CP0, CP1, MCLRE
PIC12CE674	DIP8	/gpic12c672	FOSC0, FOSC1, FOSC2, WDTE, PWRTE, CP0, CP1, MCLRE
PIC16C61	DIP18	/gpic16c61	FOSC0, FOSC1, WDTE, PWRTE, CP
PIC16C62	DIP28	/gpic16c62	FOSC0, FOSC1, WDTE, PWRTE, CP0, CP1
PIC16C620	DIP18	/gpic16c620	FOSC0, FOSC1, WDTE, PWRTE#, BODEN, CP0, CP1
PIC16C620A	DIP18	/gpic16c620	FOSC0, FOSC1, WDTE, PWRTE#, BODEN, CP0, CP1
PIC16C621	DIP18	/gpic16c621	FOSC0, FOSC1, WDTE, PWRTE#, BODEN, CP0, CP1
PIC16C621A	DIP18	/gpic16c621	FOSC0, FOSC1, WDTE, PWRTE#, BODEN, CP0, CP1
PIC16C622	DIP18	/gpic16c62a	FOSC0, FOSC1, WDTE, PWRTE#, BODEN, CP0, CP1
PIC16C622A	DIP18	/gpic16c62a	FOSC0, FOSC1, WDTE, PWRTE#, BODEN, CP0, CP1
PIC16C62A	DIP28	/gpic16c62a	FOSC0, FOSC1, WDTE, PWRTE#, BODEN, CP0, CP1
PIC16C62B	DIP28	/gpic16c62a	FOSC0, FOSC1, WDTE, PWRTE#, BODEN, CP0, CP1
PIC16C62C	DIP28	/gpic16c62a	FOSC0, FOSC1, WDTE, PWRTE#, BODEN, CP0, CP1
PIC16C63	DIP28	/gpic16c63	FOSC0, FOSC1, WDTE, PWRTE#, BODEN, CP0, CP1
PIC16C63A	DIP28	/gpic16c63	FOSC0, FOSC1, WDTE, PWRTE#, BODEN, CP0, CP1
PIC16C64	DIP40	/gpic16c62	FOSC0, FOSC1, WDTE, PWRTE, CP0, CP1
PIC16C64A	DIP40	/gpic16c62a	FOSC0, FOSC1, WDTE, PWRTE#, BODEN, CP0, CP1
PIC16C65	DIP40	/gpic16c65	FOSC0, FOSC1, WDTE, PWRTE, CP0, CP1
PIC16C65A	DIP40	/gpic16c63	FOSC0, FOSC1, WDTE, PWRTE#, BODEN, CP0, CP1
PIC16C65B	DIP40	/gpic16c63	FOSC0, FOSC1, WDTE, PWRTE#, BODEN, CP0, CP1
PIC16C66	DIP28	/gpic16c66	FOSC0, FOSC1, WDTE, PWRTE#, BODEN, CP0, CP1
PIC16C67	DIP40	/gpic16c66	FOSC0, FOSC1, WDTE, PWRTE#, BODEN, CP0, CP1
PIC16C71	DIP18	/gpic16c61	FOSC0, FOSC1, WDTE, PWRTE, CP
PIC16C710	DIP18	/gpic16c710	FOSC0, FOSC1, WDTE, PWRTE#, BODEN, CP
PIC16C711	DIP18	/gpic16c711	FOSC0, FOSC1, WDTE, PWRTE#, BODEN, CP
PIC16C712	DIP18	/gpic16c621	FOSC0, FOSC1, WDTE, PWRTE#, BODEN, CP0, CP1
PIC16C716	DIP28	/gpic16c62a	FOSC0, FOSC1, WDTE, PWRTE#, BODEN, CP0, CP1
PIC16C717	DIP18	/gpic16c717	FOSC0, FOSC1, FOSC2, WDTE, PWRTE, MCLRE, BORV0, BORV1, CP
PIC16C72	DIP28	/gpic16c62a	FOSC0, FOSC1, WDTE, PWRTE#, BODEN, CP0, CP1
PIC16C72A	DIP28	/gpic16c62a	FOSC0, FOSC1, WDTE, PWRTE#, BODEN, CP0, CP1
PIC16C73	DIP28	/gpic16c65	FOSC0, FOSC1, WDTE, PWRTE, CP0, CP1
PIC16C73A	DIP28	/gpic16c63	FOSC0, FOSC1, WDTE, PWRTE#, BODEN, CP0, CP1
PIC16C73B	DIP28	/gpic16c63	FOSC0, FOSC1, WDTE, PWRTE#, BODEN, CP0, CP1
PIC16C74	DIP40	/gpic16c65	FOSC0, FOSC1, WDTE, PWRTE, CP0, CP1

Device	Package	Mnemonic	Programmable Options
PIC16C745	DIP28	/gpic16c745	FOSC0, FOSC1, WDTEN, PWRTEN#, CP0, CP1
PIC16C74A	DIP40	/gpic16c63	FOSC0, FOSC1, WDTE, PWRTE#, BODEN, CP0, CP1
PIC16C74B	DIP40	/gpic16c63	FOSC0, FOSC1, WDTE, PWRTE#, BODEN, CP0, CP1
PIC16C76	DIP28	/gpic16c66	FOSC0, FOSC1, WDTE, PWRTE#, BODEN, CP0, CP1
PIC16C765	DIP40	/gpic16c745	FOSC0, FOSC1, WDTEN, PWRTEN#, CP0, CP1
PIC16C77	DIP40	/gpic16c66	FOSC0, FOSC1, WDTE, PWRTE#, BODEN, CP0, CP1
PIC16C770	DIP20	/gpic16c770	FOSC0, FOSC1, FOSC2, WDTE, PWRTE, MCLRE, BORV0, BORV1, CP
PIC16C771	DIP20	/gpic16c771	FOSC0, FOSC1, FOSC2, WDTE, PWRTE, MCLRE, BORV0, BORV1, CP
PIC16C773	DIP28	/gpic16c773	FOSC0, FOSC1, WDTEN, PWRTEN#, BOREN, BORV0, BORV1, CP0, CP1
PIC16C774	DIP28	/gpic16c773	FOSC0, FOSC1, WDTEN, PWRTEN#, BOREN, BORV0, BORV1, CP0, CP1
PIC16C781	DIP20	/gpic16c781	FOSC0, FOSC1, FOSC2, WDTE, PWRTE, MCLRE, BORV0, BORV1, CP
PIC16C782	DIP20	/gpic16c770	FOSC0, FOSC1, FOSC2, WDTE, PWRTE, MCLRE, BORV0, BORV1, CP
PIC16C84	DIP18	/gpic16c84	FOSC0, FOSC1, WDTE, PWRTE, CP
PIC16C923	PLCC68	/gpic16c923	FOSC0, FOSC1, WDTE, PWRTE#, CP0, CP1
PIC16C924	PLCC68	/gpic16c924	FOSC0, FOSC1, WDTE, PWRTE#, CP0, CP1
PIC16CE623	DIP18	/gpic16c620	FOSC0, FOSC1, WDTE, PWRTE#, BODEN, CP0, CP1
PIC16CE624	DIP18	/gpic16c621	FOSC0, FOSC1, WDTE, PWRTE#, BODEN, CP0, CP1
PIC16CE625	DIP28	/gpic16c62a	FOSC0, FOSC1, WDTE, PWRTE#, BODEN, CP0, CP1
PIC16CR62	DIP28	/gpic16c62a	FOSC0, FOSC1, WDTE, PWRTE#, BODEN, CP0, CP1
PIC16CR64	DIP40	/gpic16c62a	FOSC0, FOSC1, WDTE, PWRTE#, BODEN, CP0, CP1
PIC16CR83	DIP18	/gpic16cr83	FOSC0, FOSC1, WDTE, PWRTE#, CP, DP
PIC16CR84	DIP18	/gpic16cr84	FOSC0, FOSC1, WDTE, PWRTE#, CP, DP
PIC16F627	DIP18	/gpic16f627	FOSC0, FOSC1, FOSC2, WDTE, PWRTE#+BODEN, CP0+CP1+CPD, LVP, MCLRE
PIC16F628	DIP18	/gpic16f628	FOSC0, FOSC1, FOSC2, WDTE, PWRTE#+BODEN, CP0+CP1+CPD, LVP, MCLRE
PIC16F83	DIP18	/gpic16f83	FOSC0, FOSC1, WDTE, PWRTE#, CP
PIC16F84	DIP18	/gpic16f84	FOSC0, FOSC1, WDTE, PWRTE#, CP
PIC16F84A	DIP18	/gpic16f84	FOSC0, FOSC1, WDTE, PWRTE#, CP
PIC16F870	DIP28	/gpic16f870	FOSC0, FOSC1, WDTE, PWRTE#, BODEN, CP, DP, WRT
PIC16F871	DIP40	/gpic16f870	FOSC0, FOSC1, WDTE, PWRTE#, BODEN, CP, DP, WRT
PIC16F872	DIP28	/gpic16f870	FOSC0, FOSC1, WDTE, PWRTE#, BODEN, CP, DP, WRT
PIC16F873	DIP28	/gpic16f873	FOSC0, FOSC1, WDTE, PWRTE#, BODEN, CP0, CP1, WRT
PIC16F873A	DIP28	/gpic16f873a	FOSC0, FOSC1, WDTEN, PWRTEN#, BOREN, LVP, CP, WRT
PIC16F874	DIP40	/gpic16f873	FOSC0, FOSC1, WDTE, PWRTE#, BODEN, CP0, CP1, WRT

Device	Package	Mnemonic	Programmable Options
PIC16F874A	DIP40	/gpic16f873a	FOSC0, FOSC1, WDTEN, PWRTEN#, BOREN, LVP, CP, WRT
PIC16F876	DIP28	/gpic16f876	FOSC0, FOSC1, WDTE, PWRTE#, BODEN, CP0, CP1, WRT
PIC16F876A	DIP28	/gpic16f876a	FOSC0, FOSC1, WDTEN, PWRTEN#, BOREN, LVP, CP, WRT
PIC16F877	DIP40	/gpic16f876	FOSC0, FOSC1, WDTE, PWRTE#, BODEN, CP0, CP1, WRT
PIC16F877A	DIP40	/gpic16f876a	FOSC0, FOSC1, WDTEN, PWRTEN#, BOREN, LVP, CP, WRT

PIC16CR83/84, PIC16F83/84

The PIC16CR83 and the PIC16CR84 devices have EPROM program memory. The PIC16F83 and the PIC16F84(A) devices have FLASH program memory.

In addition to the program memory, these device have 64 bytes of EEPROM data memory. For programming the EEPROM data memory, the device mnemonic /gpicdata64 is used. The data to be programmed must reside in a separate file beginning at address 0. The data in the file must be word-oriented. Of every word, the low-byte is programmed into the device.

Eg.: data and program memory of a PIC16F84 is to be programmed. The program memory is to be read-protected:

```
hedchip /gpicdata64 /p /e /v datafile.hex        ; erases, programs and verifies EEPROM data
                                                   memory.
hedchip /gpic16f84 /p /e /v /s32 codefile.hex    ; erases, programs, verifies and read-protects FLASH
                                                   program memory.
```

The PIC16CR83 and the PIC16CR84 devices have a device option that read-protects the EEPROM data memory. The graphical user interface HC95 offers the device option DP for that purpose. In the command line the parameter /s64 is used. The device option CP (corresponds to the command line parameter /s32) activates the program memory read-protection.

PIC12C5XX, RC-Oscillator calibration

With these devices, the last memory location of the EPROM program memory serves as a calibrating value for the RC oscillator. Microchip specifies that this memory location is programmed with an MOVLW command for loading the calibration value. With new devices, this memory location is already programmed accordingly. The value 0c80h corresponds to the assembler instruction MOVLW 080h. This command is executed as the first command after a reset, and the program counter moves onto 0000h. This feature is taken into account in the hed.chip software. Nevertheless, some details must be observed.

1. In UV-erasable devices, this memory location is also erased. It must be reprogrammed with a suitable value: eg: 0c80h = MOVLW 080h. If desired, the oscillator can also be calibrated with another value.
2. When programming, the source file must be smaller than the memory size or it must contain 0c80h for this address. Otherwise, there will be messages that the device cannot be programmed or that it failed verification.
3. The MOVLW XX instruction loads a value (eg: 080h) into the W register. The user-written program of the controller has to store this value in the OSCCAL register at address 05ch.

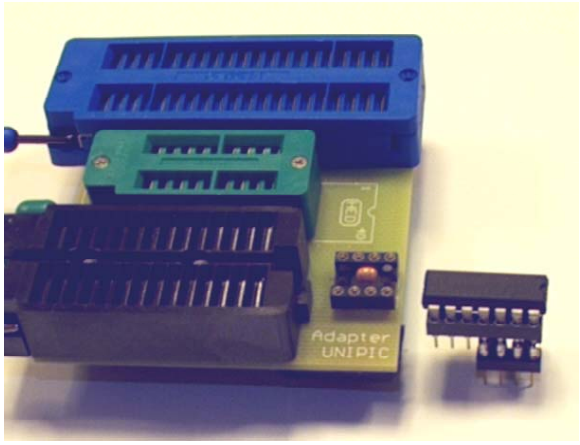
PIC12C67X, RC-Oscillator calibration

With these devices, the last memory location of the EPROM program memory serves as a calibrating value for the INTRC oscillator mode (device options FOSC1 or FOSC0+FOSC1). In principle, this is the same as for PIC12CXX. Instead of using the instruction MOVLW, the instruction RETLW is programmed into the last memory location of the EPROM program memory.

PIC16F87xA

These devices have 128 or 256 bytes of EEPROM data memory. This data memory can be programmed using the mnemonics /gpicdata128 or /gpicdata256. The EEPROM data memory cannot be erased using these mnemonics. To erase the data memory you have to erase the FLASH program memory using the mnemonics /gpic16f873a or /pic16f876a.

PIC16F630, Insertion into adapter UNIPIC



These devices in the DIP14 package must be inserted into the DIP8 socket of the adapter UNIPIC or UNIPIC18. Pin 1 of the device must be inserted into pin 1 of the DIP8 socket.

In the case of the adapter UNIPIC an intermediate adapter must be made of 2 sockets DIP8 and DIP14. This will avoid a collision with the DIP28 test socket.

See picture.

PIC12F629, PIC16F630 config word

The config word of these devices contains a “Band Gap Referenz” preprogrammed by Microchip. The devices cannot be programmed with a value for the config word contained in the source file. To program the various device options, the settings in for “device options” HC95 must be used.

The attempt to program the device with a config word in the source file, will either fail or it will overwrite the preprogrammed settings.

2.5 Toshiba microcontroller

Using special adapters made by Toshiba, these microcontroller can be programmed with EPROM programming algorithms. The following controllers are currently supported by hed.chip:

Device	Package	Mnemonic	Adapter, Memory, Comments
TC571000AD	DIP32	/gam27c010	No adapter required, 128kByte
TC571001AD	DIP32	/gam27c010	No adapter required, 128kByte
TMP88PH40N	DIP28	/gtmp88ph40	Toshiba BM11196, 16kByte
TMP88PH40M	SOIC288	/gtmp88ph40	Toshiba BM11195, 16kByte
TMP88PS43F	P-LQFP80	/gtmp88ps43	Toshiba BM11180A, 64kByte

2.6 Serial memory devices

EEPROMs. serial 2-wire interface, I²C

hed.chip programs I²C-EEPROMs, ranging from 128 Bytes 24C01A up to 32 kbyte 24C256. The programming algorithm is suited to devices made by Atmel, series AT24C**. No special features of these devices are used, so that devices from other manufacturers are also programmable. Devices from many manufacturers have already been tested and included in the device list.

Some I2C-EEPROM have a programmable write-protection. In the case of the SGS Thomson devices from the ST24C** and ST25C** series, this protection is determined by the content of the last two memory locations in the device and the level of PRE# input. To use this protection mechanism, the source file must contain suitable values for these memory locations.

There are also low-voltage versions of some modern devices. All the devices known to me can however also be programmed at 5V. Some customers use **hed.chip** to program I2C-EEPROMs IN-CIRCUIT. For these customers some low-voltage programming algorithms have been implemented into the **hed.chip** software. In the device list, these are shown as devices of types 'AT24LV***'.

Philips PCF85C-2**

hed.chip supports the PCF8582, PCF8594 and PCF8598 devices. These devices are very similar to the 24C02, 24C04 and 24C08 devices. The only difference is, that when reading the internal address pointer does not increment beyond the 256-bytes page border.

EEPROMs, serial 3-wire interface, SPI

hed.chip programs the Atmel AT25*** series and the Xicor X25*** and X25F0** series. These devices have a write-protection that covers a quarter, half or all of the device. For programming the write-protection, the /s parameter must be used.

/s0	device unprotected
/s1	first quarter protected
/s2	first half protected
/s3	complete device write-protected

To program protected devices using **hed.chip**, the device must first be erased. To do this, the additional use of the /e parameter in the command line suffices. SPI-EEPROMs from other manufacturers will be implemented in the future.

EEPROMs, Microwire-Interface

hed.chip supports the EEPROM series with Microwire interface. The SERMEM adapter is required. The types 93C06, 93C46, 93C56, and 93C66 have been implemented. More types will follow.

FPGA-Configuration Memories series AT17C***

are programmed in the SERMEM adapter. In this adapter, devices in the DIP8 package can be inserted. For other packages, only the appropriate connections between the device in PLCC20 or SOIC20 to the DIP8 socket must be made. Atmel's application note 'FPGA Configuration EEPROM Program Specification' contains all the necessary information. hed.chip automatically identifies the inserted device. The polarity option is determined by 4 bytes in the source file. A polarity set to active low means that the signal OE is active when 0V is applied to the device.

Device	Address	Content	Polarity	Option
AT17C65	02000h	FF FF FF FF	active LOW	(RESET/oe#)
AT17C65	02000h	00 00 00 00	active HIGH	(reset#/OE)
AT17C128	04000h	FF FF FF FF	active LOW	(RESET/oe#)
AT17C128	04000h	00 00 00 00	active HIGH	(reset#/OE)
AT17C256	08000h	FF FF FF FF	active LOW	(RESET/oe#)
AT17C256	08000h	00 00 00 00	active HIGH	(reset#/OE)

Any other value in the source file leaves the polarity option unchanged. In this case, hed.chip will report a verification error at address 2000h, 4000h, or 8000h respectively.

The polarity option can also be programmed using the following additional /s parameters when programming. Command line parameters take precedence over data in the source file.

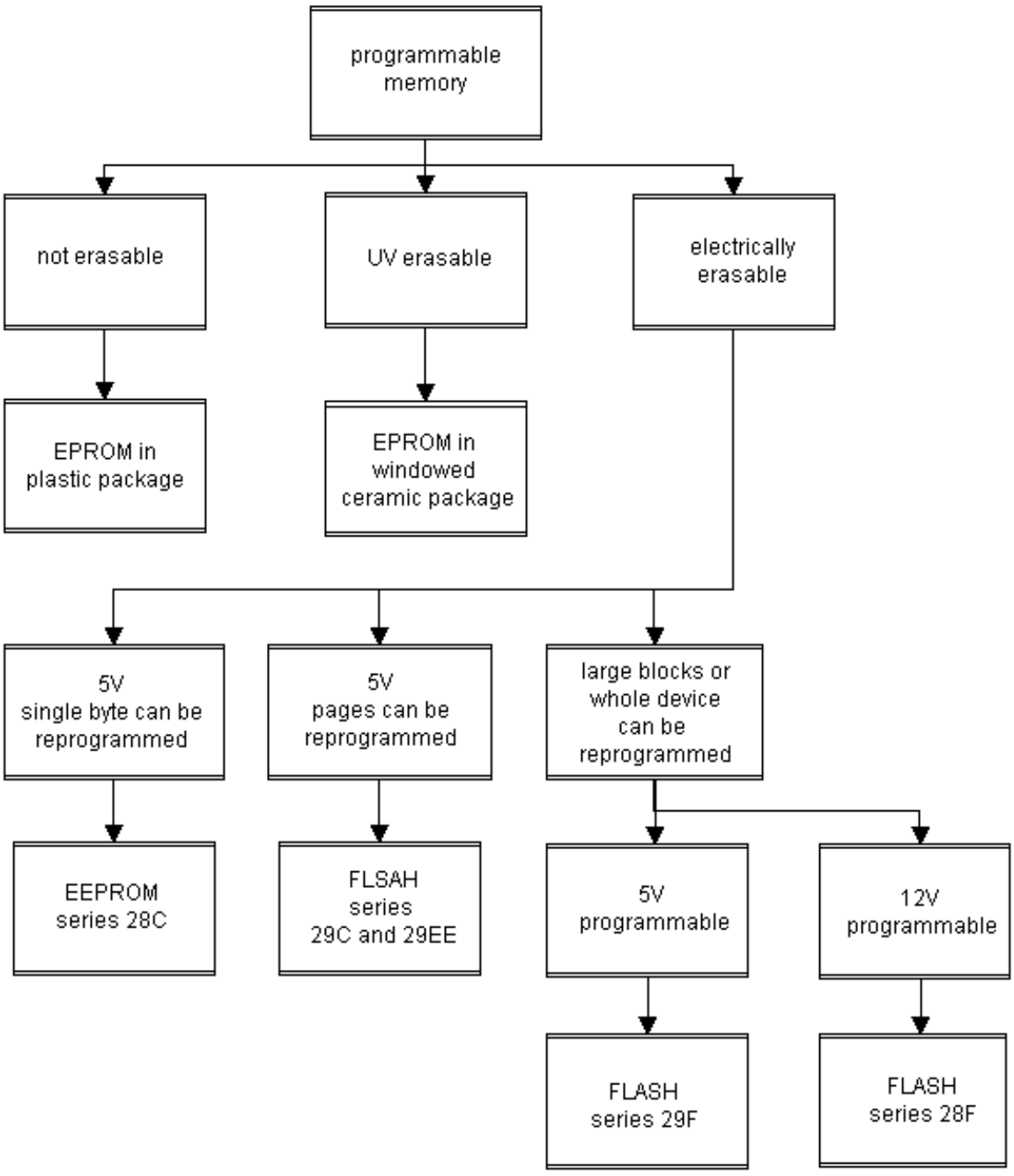
S0:	polarity is determined by source file (default)
S1:	polarity active LOW (RESET/oe#)
S2:	polarity active HIGH (reset#/OE)

If there is data in the source file for programming the polarity option, but a different setting is forced by a command line parameter, the device is programmed according to the command line parameters. If the device programmed in this way is compared to the original file, a verification error is displayed.

When erasing serial memory devices, the complete device is programmed with 0FFh and, in the case of the AT17C series devices, the polarity option is set to active low. Erasure is not required before programming.

2.7 Parallel memory devices

hed.chip programs EPROMs ranging from 8 kbyte to 512 kbyte, as well as EEPROMs and FLASH-PEROM ranging from 0.5 kbyte to 512 kbyte.



hed.chip uses the HEXBIN.EXE program to convert HEX files into binary files. The use of an offset is not supported.

eg: an EPROM is to be programmed that will later be seen by a processor at address 08000h. You can either use appropriate software to convert the Intel Hex file into binary format, or you can use assembler directives to prevent an offset in the HEX source file. Use 'phase' instead of '.org' in the assembler source. If this is not observed, a binary file is created which contains 0FFh up to address 08000h and, following that, the actual data to be programmed.

The current version of the HEXBIN.EXE program does not support segmented addressing. In this case, the HEX to binary conversion must be carried out by the user with a suitable tool (eg: hed.HexEd – a modern HexEditor).

EPROMs

The manufacturers of EPROMs point out that exact observance of the specifications is absolutely necessary for optimum programmability and long term data retention. Although most programming algorithms are very similar, every manufacturer has his own ideas about how these devices are to be programmed. You should first check the device list for the correct device mnemonic before programming. If the correct device mnemonic is used, hed.chip will carry out the programming precisely according to the manufacturer's specifications.

The latest algorithms specified by all manufacturers in the device list have been used. In doing so, it became apparent that in some cases the device name has not changed in the last 10 years, although old data books use other algorithms with mostly longer pulse times. The algorithms used at that time correspond most closely to the algorithms for the M2764A, M27128A, and M2756 made by SGS Thomson. The mnemonics for these devices are: s2764, s27128, and s27256. Use these mnemonics to program very old devices.

With the PLCC32 adapter, some, but not all, EPROMs in PLCC32 package can be programmed. Programmable devices in the PLCC32 package can be found in the device list.

EPROMs 2708, 2716, and2732

These devices are not supported by the programmer. They require voltages in excess of 20V and, in some cases, several different supply voltages for programming. In most applications, they can be replaced using CMOS versions (27C16 and 27C32). The manufacturer's data sheets must be used for checking the device's pin layout for the individual application. TMS2716 made by Texas Instruments uses a pin layout that differs from the standard used for these devices.

Erasing EPROMs with UV light

EPROMs can be erased simply by using sun light. It takes about 2 weeks and the result does not meet the manufacturer's specifications. For correct erasure a special UV eraser machine is used. With this it takes about 15 minutes to erase an EPROM. It is important not to shorten the erasure time. Insufficiently erased EPROMs typically display the following symptoms: The blank check fails occasionally and when repeatedly verifying a programmed device, differences are reported at varying locations.

EEPROM, series 28C

With EEPROMs, single bytes can be programmed individually. The larger devices also allow the programming of several bytes within a page in one write operation. Where available, this is used to speed up the programming. Also, where available, **hed.chip** programs the write-protection of these devices if the /s1 parameter is given when programming.

For programming memory pages and to activate the write-protection, bytes must be written successively with a maximum delay in between. Normally, this can be done in a DOS task of multitasking systems, such as Windows 95 or Windows NT 4.0, but it cannot be guaranteed for all circumstances. The DOS task should be running in full screen mode, and all settings should be optimized for maximum performance. It is advisable to verify the programming using the /v parameter. If verification errors occur repeatedly, the machine must be booted using DOS for programming.

eg: AT28C256 is to be programmed, verified and write protected:

```
hedchip lpt2 /g28c256 /p /v /s1 myapp.hex          ; 28C256 in DIP28
```

The write-protection of these devices is a useful feature. In the application it is a reliable protection against unintentional write operations. These can be triggered by turning on the supply voltage.

It is not necessary to erase EEPROMs before programming. However, the write-protection can only be disabled by erasing the device:

```
hedchip /g28c256 /e
```

Non-volatile SRAM

These devices combine ordinary RAM with a battery in a module. The advantage is that these devices can be reprogrammed without restriction of the number of programming cycles. They are available with and without integrated real time clock.

When programming devices with real time clock:

- Do not erase the device, even if it is not blank. These devices can be reprogrammed without being erased first.
- Do not program the memory used for the clock circuitry. For a SGS Thomson M48T18 the source file should not be greater than 01FF8h bytes.

hed.chip can write to the registers used for the clock circuitry. No consideration is taken what effect this will have on the real time clock.

FLASH, series 29C and 29EE

These 5V-only programmable FLASH devices are similar to EEPROMs. Unlike EEPROMs, all bytes within a memory page must be written in one write operation. Bytes not written in a page are erased by the internal write operation. Typically, a memory page consists of 128 bytes.

hed.chip programs devices made by most manufacturers. More devices are being tested and implemented in the programming software. **hed.chip** evaluates the device ID and sets memory size and programming parameters accordingly. Unknown devices or devices with a manufacturer's ID that does not match the device mnemonic are rejected. New devices are being added to the device list continuously.

The general write-protection of these devices can be activated using the /s1 parameter when programming. Advice given for programming memory pages in EEPROMs also applies to these devices.

eg: program Atmel AT29C010 and activate write-protection:

```
hedchip /ga29cxxx /p /v /e /s1 myapp.bin
```

The additional parameters have the following effects:

/v	Programming and erasure are verified.
/e	If not blank, the device is erased before programming. If the write-protection of the device is enabled, this is required for programming.
/s1	Write-protection is activated after programming.

FLASH with boot block write-protection

Currently applies to: Atmel AT29C020, AT29C040, Winbond W29C020, W29C040

By using the /sn parameter when programming, a memory block at the beginning and/or at the end of the memory may be protected against further programming. In systems where the memory may be updated, this guards important core routines against unintentional changes.

By using the /s parameter, any combination of general write protection (SDP) and boot block locks can be activated:

/s1	SDP (general, reversible write protection)
/s2	Lower Address Boot Block Lock (LABBL)
/s3	SDP + LABBL
/s4	Higher Address Boot Block Lock (HABBL)
/s5	SDP + HABBL
/s6	LABBL + HABBL
/s7	SDP + LABBL + HABBL

You can either use several /s parameters in one command line, or add up the numbers. The two following examples are identical. Both command lines activate the two boot block locks and the general write-protection (SDP):

```
hedchip /ga29cxxx /p /v /e /s7 myapp.bin
hedchip /ga29cxxx /p /v /e /s1 /s2 /s4 myapp.bin
```

The boot block write-protection (LABBL and HABBL) is irreversible. Further programming of a device protected in this way is possible, but requires the following considerations:

1. If the general write-protection (SDP) is enabled, the device must be erased. This will not actually erase the device, but it will deactivate the general write-protection.
2. If the Lower Address Boot Block Lock (LABBL) is activated, the device must be read first. When programming, the protected memory area must be programmed with the data previously read.

Explanation: **hed.chip** can only program devices continually beginning at address 00000h. Since the locked memory cannot be reprogrammed, the detection of programming errors must be avoided. Memory cells that cannot be changed must be written with the exact same data that is already there.

Winbond, series 29EE and 29C

Some devices are delivered with the SDP write protection enabled. These devices cannot be programmed in this state. To disable the SDP write protection they must be erased.

```
hedchip /gw29eexxx /e                    ; erases series 29EE and 29C device
```

FLASH, series 29F

Like series 29C and 29EE, these devices may be programmed without an increased programming voltage. The distinction is that they must be erased before programming. **hed.chip** can neither activate nor deactivate the sector protection feature of these devices.

hed.chip evaluates the device ID and sets memory size and programming parameters accordingly.

The /g29fxxx device mnemonic is used for programming these devices:

```
hedchip /g29fxxx /p/v/e your_app.bin     ; erases, programs, and verifies 29F-FLASH
```

FLASH, series 49F

These devices are very similar to the series 29F devices. They have a boot block protection that can be activated by using the /s2 parameter when programming. Once activated, this protection cannot be deactivated.

Some – but not all – devices have a RESET-Pin. This signal has the following functions:

- RESET = low: All outputs of the device go to high-impedance state.
- RESET = high: Normal read/write operation
- RESET = 12V: The boot block can be erased and programmed even if the boot block protection has been activated. This does not deactivate the protection. When 12V is removed, the boot block is again read-only.

Use device mnemonics 29fxxx and 29lvxxx for devices without RESET pin.

Use device mnemonics 49f00x and 49lv00x for devices with RESET pin.

FLASH, series 28F

Series 28F FLASH devices require a programming voltage of 12V. They must be erased before programming. The /g28fxxx device mnemonic is used for all devices in this series. **hed.chip** evaluates the device ID and sets memory size and programming algorithm accordingly, eg:

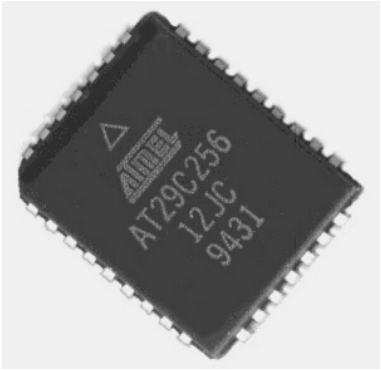
```
hedchip /g28fxxx /p/v/e your_app.bin          ; erases, programs, and verifies 28F-FLASH
```

FLASH Intel 28F001B

These devices have a boot block that is always protected by hardware. In the 28F001BX-T, this block is located at address 01E000h, and in the 28F001BX-B, it is located at address 0. The boot block can only be written or erased when 12V is applied to the RP# pin (pin30). To do this using **hed.chip**, the following is necessary: a wire connection must be set connecting pins 1 and 30. To do this, a piece of wire can be inserted into the test socket together with the device. Blank check and verification should be executed without this wire connection.

Memory devices in the PLCC32 package

A PLCC32 to DIP32 adapter can be used to program devices in this package. The **hed.chip** software has been designed so that this one adapter can be used for devices that are also available in the DIP24, DIP28 and DIP32 packages. However, there is one exception: For EPROM 27C512 in the PLCC32 package the PLCC32_28 adapter is required.



The device list states which devices in PLCC32 package are supported. Depending on the device, the device mnemonic required for a device in PLCC32 package may or may not differ from the device mnemonic used for the same device in the DIP package.

Example for 28C256

A 28C256 in the PLCC32 package is to be erased, programmed, verified, and finally write-protected. Since the pin layout of the PLCC32 package differs from the layout of the DIP28 package, the /g28c256plcc device mnemonic must be used.

```
hedchip /g28c256plcc /p/v/e/s1 myapp.hex                    ; not: /g28c256
```

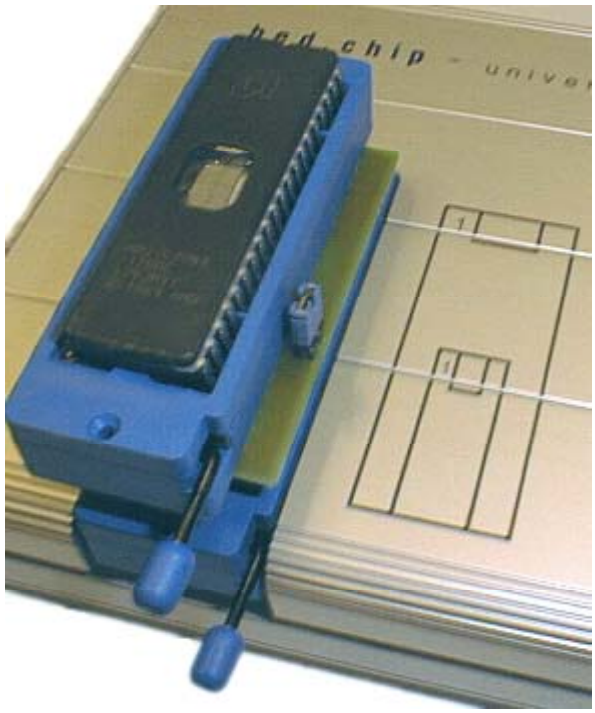
16-Bit memory devices in the DIP40 package

These devices can be programmed using the MEM16_DIP40 adapter.

When programming memory devices, the jumper on the adapter must be set. (Remove the jumper for some special Atmel AT90S-series microcontrollers).

Please note:

- Orientation of the device in the adapter.
- Orientation of the adapter on the programmer.
- Jumper is set.



Low-voltage

hed.chip also supports low-voltage devices. Many devices that can be programmed using 5V and some 3.3V-only devices are already in the device list. More 3.3V-only devices are being implemented and added to the device list. Customers requests for specific devices are given a higher priority.

3 DOS return codes

When HEDCHIP.EXE terminates, a code is returned to the operating system. The list enumerates all possible codes and explains the conditions under which a specific code will be returned. The return code can be used for conditional branches in batch programs. See Chapter 1.11 for details.

Code	Description
000	Operation successful. In the case of a blank check or verification, the result is blank or equal respectively.
001	Unexpected end of command line. The help text has been displayed.
002	The device mnemonic is missing. A list of available mnemonics has been displayed.
003	Plug and Prog did not detect the programmer. The Plug and Prog function has tested all LPT ports of the system. hed.chip was not found.
004	The command parameter (eg: /p for programming) is missing.
005	The file name is missing. For /p and /v command parameters, the name of a source file is required. For /r, the name of a target file is required.
006	Illegal parameter in command line. Most likely cause: you supplied a device mnemonic unknown to HEDCHIP.EXE.
007	Unknown printer port. Most likely cause: you tried to access LPT2, but your system does not have this port.
008	HEDCHIP.EXE could not start one of its sub-programs: HEXBIN.EXE and JEDECASM.COM. These are used to convert Intel-HEX and JEDEC files to binary format. The program executables must be in the same directory as HEDCHIP.EXE
009	HEXBIN.EXE or JEDECASM.COM has generated an error. These programs are automatically used to convert Intel-HEX or JEDEC files to binary format. This error is generated if the format of the source file does not comply with the standard.
010	HEDCHIP.EXE terminated by the user. HEDCHIP.EXE gave the choice to continue or to abort a function. The user chose 'abort' or 'don't continue'
011	File IO error. A file was not found, could not be opened, read, or written to.
012	Bad operating system. HEDCHIP.EXE does not support this operating system.
013	Initialisation of operating systems extensions failed. Possible causes: Under Windows NT, HEDCHIP.EXE loads a driver that handles hardware access. This driver could not be loaded or failed to operate. HED_SUPP.DLL must be in the same directory as HEDCHIP.EXE. In the Windows drivers directory (‘<WINNT>\system32\drivers’) there must be HED_DRV.SYS.

Code	Description
129	NOT EQUAL or NOT BLANK. HEDCHIP.EXE performed a verification or a blank check. The result is 'not equal' or 'not blank' respectively. This result is also generated if the device is read-protected.
130	Device could not be erased. Possible causes: 1. An error was detected during erasure. 2. A blank check after erasure resulted in 'NOT BLANK'. A blank check is performed automatically if the 'verify after programming' programmer option is checked.
131	Device could not be programmed. Possible causes: 1. An error was detected during programming 2. Verification performed after programming resulted in 'NOT EQUAL'. This verification is performed automatically if the 'verify after programming' programmer option is checked.
132	Security fuse, lock bits, or write-protection could not be programmed. For some devices, this error is also generated if the device does not have such a protection or that protection level.
133	Device could not be read. The device was to be read into a file. During this operation an error occurred.
134	Device could not be identified. HEDCHIP.EXE tries to identify the device before performing any action on it. The identification is attempted twice, giving the user a chance to insert the correct device. Possible causes: device non-functional, device read-protected, no device, wrong device.
135	Device could not be read. Reason: the device is read protected. A read protected device can neither be read nor copied.
255	Programmer does not respond. Possible causes: the programmer is not connected to the computer or the power supply is off. This error is only generated if one of the programmer options 'use LPT1' or 'use LPT2' is checked. Deactivate these options and let HEDCHIP.EXE assist you in setting up the programmer.

4 Adapters

Adapters are required for SMD devices or devices requiring special handling by the programmer. This list contains short descriptions of the currently available adapters. Other models can be designed at short notice.

Adapter type	Description
Adapter MEM16_DIP40	Special adapter for 16-Bit memory devices (eg.: SGS Thomson M27C4002) and Atmel Microcontrollers, series AT90S, with AD converter, eg. : AT90S8535-10PC
Adapter DIPMEM	Special adapter for EPROM 27C16. Has DIP40 test socket. hed.chip hardware version 1 requires this adapter also for various other devices.
Adapter PLCC2500	Special adapter for Atmel ATV2500H and ATV2500B devices in the PLCC44 package. Has PLCC44 test socket.
Adapter PLCC20	For PLD devices in the PLCC20 package. Has PLCC20 test socket.
Adapter PLCC28	For PLD devices in the PLCC28 package. Has PLCC28 test socket.
Adapter PLCC32	For parallel memory devices (EPROMs, FLASH, etc.). This adapter translates the pins for DIP32 devices 1:1 into the PLCC package. The software has been designed so that devices which are also available in the DIP28 package can be programmed using this adapter. Has PLCC32 test socket.
Adapter PLCC32_28	For parallel memory devices. Translates PLCC32 to DIP28. This Adapter is required for EPROM 27C512 in the PLCC32 package. The 27C512 is the exception to the rule that all devices in the PLCC32 package can be programmed using the PLCC32 adapter. The pins 1, 12, 17, and 26 of the PLCC32 socket not connected. Has PLCC32 test socket
Adapter PLCC32/44	For parallel memory devices (EPROMs, FLASH, etc.) and MCS51 microcontroller. Has PLCC32 and PLCC44 test socket.
Adapter PLCC44	For MCS51 microcontroller in the PLCC44 package. Has PLCC44 test socket.
Adapter PLCC52	For Dallas DS87C530. Has low cost PLCC52 socket.
Adapter PLCC68_40	For MCS51 microcontroller in the PLCC68 package. Has PLCC68 test socket.
Adapter PLCC750	Special adapter for Atmel AT22V10, ATV750/L and ATV750B devices in the PLCC28 package. Has PLCC28 test socket.
Adapter PQFP44_C505	Special adapter for Siemens/Infineon C505A and C505CA
Adapter SERMEM	Adapter with DIP8 socket for certain types of serial EEPROM. Uses the same printed circuit board as the SOIC20 adapter. A SOIC20 test socket may be installed when required. Has DIP8 precision socket.

Adapter type	Description
Adapter SOIC20	For PLD devices and MCS51 microcontroller. This translates the pins of the DIP20 package 1:1 into the SOIC20 package. Uses the same printed circuit board as the SERMEM adapter. A DIP8 socket may be installed when required. Has SOIC20 test socket.
Adapter TQFP44	For MCS51 microcontroller in the TQFP44 package. Has TQFP44 test socket.
Adapter UNIPIC	Special adapter for PIC microcontroller. Has DIP18, DIP28, DIP40 test socket and DIP8 precision socket.
Adapter UNIPIC18	Special adapter for PIC microcontroller. Has DIP18 test socket and DIP8 precision socket. DIP28 and DIP40 test sockets are not included but can be installed when required.
International Power Supply	Power supply with wall socket connectors for the US, UK and Germany. Wide input voltage range: 100 to 240VAC, 50 or 60Hz

5 Device list

Version 3.26 from 06.03.2008

This list states:

- which devices can be programmed with **hed.chip**.
- (if required) which adapters must be used. Note: The comment "on request" only refers to the exact line it appears in. eg: AMD Am27c020 in the DIP32 package can be programmed using **hed.chip**; a suitable adapter for this device in the PLCC32 package can be produced on request.
- which device mnemonic must be used for HEDCHIP.EXE in the command line.

Manufacturer	AMD			
Device	**Package**	**Mnemonic**	**Adapter**	**Comment**
Am27C010	DIP32	/gam27c010		
	PLCC32	/gam27c010	PLCC32	
Am27C020	DIP32	/gam27c020		
	PLCC32			on request
Am27C040	DIP32	/gam27c040		
	PLCC32	/gam27c040	PLCC32	
Am27C1024	DIP40	/geprom16_t1	MEM16_DIP40	Set jumper
Am27C128	DIP28	/gam27c128		
Am27C2048	DIP40	/geprom16_t1	MEM16_DIP40	Set jumper
Am27C256	DIP28	/gam27c256		
	PLCC32	/gam27c256	PLCC32_28	
Am27C4096	DIP40	/geprom16_t1	MEM16_DIP40	Set jumper
Am27C512	DIP28	/gam27c512_2		
	PLCC32	/gam27c512_2	PLCC32_28	
Am27C64	DIP28	/gam27c64		
Am27LV010/B	DIP32	/gam27c010		
	PLCC32	/gam27c010	PLCC32	
Am27LV020/B	DIP32	/gam27c020		
	PLCC32			on request
Am28F010	DIP32	/g28fxxx		
	PLCC32	/g28fxxx	PLCC32	
Am28F010A	DIP32	/g28fxxx		
	PLCC32	/g28fxxx	PLCC32	
Am28F020	DIP32	/g28fxxx		
	PLCC32	/g28fxxx	PLCC32	
Am28F020A	DIP32	/g28fxxx		
	PLCC32	/g28fxxx	PLCC32	
Am28F256	DIP32	/g28fxxx		
	PLCC32	/g28fxxx	PLCC32	
Am28F256A	DIP32	/g28fxxx		
	PLCC32	/g28fxxx	PLCC32	
Am28F512	DIP32	/g28fxxx		
	PLCC32	/g28fxxx	PLCC32	
Am28F512A	DIP32	/g28fxxx		
	PLCC32	/g28fxxx	PLCC32	
Am29F002B(B/T)	DIP32	/g29f00x		Sector Protection
	PLCC32	/g29f00x	PLCC32	not programmable
	TSOP32	/g29f00x	TSOP32	
Am29F002N(B/T)	DIP32	/g29fxxx		Sector Protection not programmable

Manufacturer	AMD			
Device	**Package**	**Mnemonic**	**Adapter**	**Comment**
	PLCC32	/g29fxxx	PLCC32	
	TSOP32	/g29fxxx	TSOP32	
Am29F010	DIP32	/g29fxxx		Sector Protection
	PLCC32	/g29fxxx	PLCC32	not programmable
Am29F040	DIP32	/g29fxxx		Sector Protection
	PLCC32	/g29fxxx	PLCC32	not programmable
Am29LV010B	PLCC32	/g29lvxxx	PLCC32	Sector Protection not programmable
Am29LV040B	PLCC32	/g29lvxxx	PLCC32	Sector Protection not programmable
N87C52T2	DIP40	/gam87c5x		
	PLCC44	/gam87c5x	PLCC44	
PALCE16V8H/Q	DIP20	/gam16v8		
PALCE20V8H/Q	DIP24	/gam20v8		
	PLCC28	/gam20v8	PLCC28	
PALCE22V10H/Q	DIP24	/gam22v10		-PC4 and -PC5 only

Manufacturer	Amic			
Device	**Package**	**Mnemonic**	**Adapter**	**Comment**
A29001	DIP32	/g29fxxx		
	PLCC32	/g29fxxx	PLCC32	
	TSOP32	/g29fxxx	TSOP32	
A290011	DIP32	/g29fxxx		
	PLCC32	/g29fxxx	PLCC32	
	TSOP32	/g29fxxx	TSOP32	
A29002	DIP32	/g29fxxx		
	PLCC32	/g29fxxx	PLCC32	
	TSOP32	/g29fxxx	TSOP32	
A290021	DIP32	/g29fxxx		
	PLCC32	/g29fxxx	PLCC32	
	TSOP32	/g29fxxx	TSOP32	
A29010	DIP32	/g29fxxx		
	PLCC32	/g29fxxx	PLCC32	
	TSOP32	/g29fxxx	TSOP32	
A29040A	DIP32	/g29fxxx		Sector Protection
	PLCC32	/g29fxxx	PLCC32	not programmable
	TSOP32	/g29fxxx	TSOP32	

Manufacturer	ASD			
Device	**Package**	**Mnemonic**	**Adapter**	**Comment**
AE29F1008	DIP32	/ga29cxxx		
	PLCC32	/ga29cxxx	PLCC32	
AE29F2008	DIP32	/ga29cxxx		
	PLCC32	/ga29cxxx	PLCC32	

Manufacturer	Atmel			
Device	**Package**	**Mnemonic**	**Adapter**	**Comment**
AT17C128	DIP8	/ga17cxxx	SERMEM	

Manufacturer	Atmel			
Device	**Package**	**Mnemonic**	**Adapter**	**Comment**
	PLCC20	/ga17cxxx	on request	
	SOIC20	/ga17cxxx	on request	
AT17C256	DIP8	/ga17cxxx	SERMEM	
	PLCC20	/ga117cxx	on request	
	SOIC20	/ga17cxxx	on request	
AT17C65	DIP8	/ga17cxxx	SERMEM	
	PLCC20	/ga17cxxx	on request	
	SOIC20	/ga17cxxx	on request	
AT17C65B	DIP8	/ga17c65b	SERMEM	B-Version has B in
	PLCC20		on request	date label
	SOIC20		on request	
AT22LV10/L	DIP24	/gat22v10	DIP750	
	PLCC28	/gat22v10	PLCC750	
AT22V10/L	DIP24	/gat22v10	DIP750	
	PLCC28	/gat22v10	PLCC750	
AT22V10B	DIP24	/gat22v10	DIP750	
	PLCC28	/gat22v10	PLCC750	
AT24C01	DIP8	/g24c01		
AT24C01A	DIP8	/g24c01a		
AT24C02	DIP8	/g24c02		
AT24C04	DIP8	/g24c04		
AT24C08	DIP8	/g24c08		
AT24C128	DIP8	/g24xc128		
AT24C16	DIP8	/g24c16		
AT24C164	DIP8	/g24c16		
AT24C256	DIP8	/g24xc256		
AT24C32	DIP8	/g24xc32		
AT24C64	DIP8	/g24xc64		
AT24LV02	DIP8	/g24lv02		low-voltage, 3.3V
AT24LV128	DIP8	/g24lv128		low-voltage, 3.3V
AT24LV256	DIP8	/g24lv256		low-voltage, 3.3V
AT25010	DIP8	/ga25010		
AT25020	DIP8	/ga25020		
AT25040	DIP8	/ga25040		
AT25080	DIP8	/ga25080		
AT25128	DIP8	/g25128		
AT25160	DIP8	/ga25160		
AT25320	DIP8	/ga25320		
AT25640	DIP8	/ga25640		
AT27BV010	PLCC32	/ga27c010	PLCC32	
AT27BV020	PLCC32	/ga27c020	PLCC32	
AT27BV040	PLCC32	/ga27c040	PLCC32	
AT27BV256	DIP28	/ga27c256		
	PLCC32	/ga27c256	PLCC32_28	
AT27BV512	DIP28	/ga27c512_2		
	PLCC32	/ga27c512_2	PLCC32	
AT27C010/L	DIP32	/ga27c010		
	PLCC32	/ga27c010	PLCC32	
AT27C020	DIP32	/ga27c020		
	PLCC32	/ga27c020	PLCC32	
AT27C040	DIP32	/ga27c040		
	PLCC32	/ga27c040	PLCC32	
AT27C080	DIP32	/ga27c080_2		

Manufacturer	Atmel			
Device	**Package**	**Mnemonic**	**Adapter**	**Comment**
AT27C1024	DIP40	/geprom16_t1	MEM16_DIP40	Set jumper
AT27C2048	DIP40	/geprom16_t1	MEM16_DIP40	Set jumper
AT27C256R	DIP28	/ga27c256		
	PLCC32	/ga27c256	PLCC32_28	
AT27C4096	DIP40	/geprom16_t1	MEM16_DIP40	Set jumper
AT27C512R	DIP28	/ga27c512_2		
	PLCC32	/ga27c512_2	PLCC32	
AT27LV010A	PLCC32	/ga27c010	PLCC32	
AT27LV020A	PLCC32	/ga27c020	PLCC32	
AT27LV040A	PLCC32	/ga27c040	PLCC32	
AT27LV256A	DIP28	/ga27c256		
	PLCC32	/ga27c256	PLCC32_28	
AT27LV512A	DIP28	/ga27c512_2		
	PLCC32	/ga27c512_2	PLCC32	
AT28C04	DIP24	/g28c04		
	PLCC32	/g28c04plcc	PLCC32	
AT28C16/E	DIP24	/g28c16		
	PLCC32	/g28c16plcc	PLCC32	
	SOIC24		on request	
AT28C17	DIP28	/g28c17		
	PLCC32	/g28c16plcc	PLCC32	
	SOIC28		on request	
AT28C256	DIP28	/g28c256		
	PLCC32	/g28c256plcc	PLCC32	
	PGA28		on request	
	SOIC28		on request	
AT28C64/X	DIP28	/g28c64		
	PLCC32	/g28c64plcc	PLCC32	
	SOIC28		on request	
AT28C64B	DIP28	/g28c64b		
	PLCC32	/g28c64bplcc	PLCC32	
	SOIC28			
AT28HC256	DIP28	/g28c256		
	PLCC32	/g28c256plcc	PLCC32	
	PGA28		on request	
	SOIC28		on request	
AT28HC64B	DIP28	/g28c64b		
	PLCC32	/g28c64bplcc	PLCC32	
	SOIC28		on request	
AT29BV010A	DIP32	/ga29lvxxx		
	PLCC32	/ga29lvxxx	PLCC32	
AT29BV020	PLCC32	/ga29lvxxx	PLCC32	
AT29BV040A	TSOP32	/ga29lvxxx	TSOP32	
AT29C010A	DIP32	/ga29cxxx		
	PLCC32	/ga29cxxx	PLCC32	
AT29C020	DIP32	/ga29cxxx		
	PLCC32	/ga29cxxx	PLCC32	
AT29C040	DIP32	/ga29cxxx		
AT29C040A	DIP32	/ga29cxxx		
AT29C256	DIP28	/ga29c256		
	PLCC32	/ga29c256plcc	PLCC32	
AT29C257	PLCC32	/ga29cxxx	PLCC32	

Manufacturer	Atmel			
Device	**Package**	**Mnemonic**	**Adapter**	**Comment**
AT29C512	DIP32	/ga29cxxx		
	PLCC32	/ga29cxxx	PLCC32	
AT29LV010A	DIP32	/ga29lvxxx		
	PLCC32	/ga29lvxxx	PLCC32	
AT29LV020	PLCC32	/ga29lvxxx	PLCC32	
AT29LV040A	PLCC32	/ga29lvxxx	PLCC32	
	TSOP32	/ga29lvxxx	TSOP32	
AT29LV256	DIP28	/ga29lv256		
	PLCC32	/ga29lv256plcc	PLCC32	
AT29LV512	DIP32	/ga29lvxxx		
	PLCC32	/ga29lvxxx	PLCC32	
AT34Cxxx				to be implemented
AT49BV001/T	DIP32	/g49lv00x		with RESET pin,
	PLCC32	/g49lv00x	PLCC32	Low Voltage
AT49BV001N/T	DIP32	/g29lvxxx		without RESET pin,
	PLCC32	/g29lvxxx	PLCC32	Low Voltage
AT49BV002/T	DIP32	/g49lv00x		with RESET pin,
	PLCC32	/g49lv00x	PLCC32	Low Voltage
AT49BV002N/T	DIP32	/g29lvxxx		without RESET pin,
	PLCC32	/g29lvxxx	PLCC32	Low Voltage
AT49BV010	DIP32	/g29lvxxx		Low Voltage
	PLCC32	/g29lvxxx	PLCC32	
AT49BV020	DIP32	/g29lvxxx		Low Voltage
	PLCC32	/g29lvxxx	PLCC32	
AT49BV040/T	DIP32	/g29lvxxx		Low Voltage
	PLCC32	/g29lvxxx	PLCC32	
AT49BV512	DIP32	/g29lvxxx		Low Voltage
	PLCC32	/g29lvxxx	PLCC32	
AT49F001/T	DIP32	/g49f00x		with RESET pin
	PLCC32	/g49f00x	PLCC32	
AT49F001N/T	DIP32	/g29fxxx		without RESET pin
	PLCC32	/g29fxxx	PLCC32	
AT49F002/T	DIP32	/g49f00x		with RESET pin
	PLCC32	/g49f00x	PLCC32	
AT49F002N/T	DIP32	/g29fxxx		without RESET pin
	PLCC32	/g29fxxx	PLCC32	
AT49F010	DIP32	/g29fxxx		
	PLCC32	/g29fxxx	PLCC32	
AT49F020	DIP32	/g29fxxx		
	PLCC32	/g29fxxx	PLCC32	
AT49F040/T	DIP32	/g29fxxx		
	PLCC32	/g29fxxx	PLCC32	
AT49F512	DIP32	/g29fxxx		
	PLCC32	/g29fxxx	PLCC32	
AT49LV001/T	DIP32	/g49lv00x		with RESET pin,
	PLCC32	/g49lv00x	PLCC32	Low Voltage
AT49LV001N/T	DIP32	/g29lvxxx		without RESET pin,
	PLCC32	/g29lvxxx	PLCC32	Low Voltage
AT49LV002/T	DIP32	/g49lv00x		with RESET pin
	PLCC32	/g49lv00x	PLCC32	
AT49LV002N/T	DIP32	/g29lvxxx		without RESET pin,
	PLCC32	/g29lvxxx	PLCC32	Low Voltage
AT49LV010	DIP32	/g29lvxxx		Low Voltage

Manufacturer	Atmel			
Device	**Package**	**Mnemonic**	**Adapter**	**Comment**
	PLCC32	/g29lvxxx	PLCC32	
AT49LV020	DIP32	/g29lvxxx		Low Voltage
	PLCC32	/g29lvxxx	PLCC32	
AT49LV040/T	DIP32	/g29lvxxx		Low Voltage
	PLCC32	/g29lvxxx	PLCC32	
AT89C1051	DIP20	/ga89cx051		
	SOIC20	/ga89cx051	SOIC20	
AT89C1051U	DIP20	/ga89cx051		‘1051 with UART
	SOIC20	/ga89cx051	SOIC20	
AT89C2051	DIP20	/ga89cx051		
	SOIC20	/ga89cx051	SOIC20	
AT89C4051	DIP20	/ga89cx051		
	SOIC20	/ga89cx051	SOIC20	
AT89C51	DIP40	/ga89c5x		
	PLCC44	/ga89c5x	PLCC44	
	TQFP44	/ga89c5x	TQFP44	
AT89C51-5	DIP40	/ga89c5x-5		Vpp = 5V
	PLCC44	/ga89c5x-5	PLCC44	Vpp = 5V
	TQFP44	/ga89c5x-5	TQFP44	
AT89C51IC2	DIP40	/gt89c51rc2-5		
	PLCC44	/gt89c51rc2-5	PLCC44	
	TQFP44	/gt89c51rc2-5	TQFP44	
AT89C51RB2	DIP40	/gt89c51rc2-5		
	PLCC44	/gt89c51rc2-5	PLCC44	
	TQFP44	/gt89c51rc2-5	TQFP44	
AT89C51RC	DIP40	/ga89c5x2		
	PLCC44	/ga89c5x2	PLCC44	
	TQFP44	/ga89c5x2	TQFP44	
AT89C51RC2	DIP40	/gt89c51rc2-5		
	PLCC44	/gt89c51rc2-5	PLCC44	
	TQFP44	/gt89c51rc2-5	TQFP44	
AT89C52	DIP40	/ga89c5x		
	PLCC44	/ga89c5x	PLCC44	
	TQFP44	/ga89c5x	TQFP44	
AT89C52-5	DIP40	/ga89c5x-5		Vpp = 5V
	PLCC44	/ga89c5x-5	PLCC44	Vpp = 5V
	TQFP44	/ga89c5x-5	TQFP44	
AT89C55	DIP40	/ga89c5x		
	PLCC44	/ga89c5x	PLCC44	
	TQFP44	/ga89c5x	TQFP44	
AT89C55WD	DIP40	/ga89c5x2		
	PLCC44	/ga89c5x2	PLCC44	
	TQFP44	/ga89c5x2	TQFP44	
AT89LS51	DIP40	/ga89sxx2		
	PLCC44	/ga89sxx2	PLCC44	
	TQFP44	/ga89sxx2	TQFP44	
AT89LS52	DIP40	/ga89sxx2		
	PLCC44	/ga89sxx2	PLCC44	
	TQFP44	/ga89sxx2	TQFP44	
AT89LS53	DIP40	/ga89sxxxx		
	PLCC44	/ga89sxxxx	PLCC44	
	TQFP44	/ga89sxxxx	TQFP44	

Manufacturer	Atmel			
Device	**Package**	**Mnemonic**	**Adapter**	**Comment**
AT89LS8252	DIP40	/ga89sxxxx		FLASH program
	PLCC44	/ga89sxxxx	PLCC44	memory
	TQFP44	/ga89sxxxx	TQFP44	
AT89LS8252	DIP40	/ga89seeprom		EEPROM data
	PLCC44	/ga89seeprom	PLCC44	memory
	TQFP44	/ga89seeprom	TQFP44	
AT89LV51	DIP40	/ga89c5x		
	PLCC44	/ga89c5x	PLCC44	
	TQFP44	/ga89c5x	TQFP44	
AT89LV51-5	DIP40	/ga89c5x-5		Vpp = 5V
	PLCC44	/ga89c5x-5	PLCC44	Vpp = 5V
	TQFP44	/ga89c5x-5	TQFP44	
AT89LV52	DIP40	/ga89c5x		
	PLCC44	/ga89c5x	PLCC44	
	TQFP44	/ga89c5x	TQFP44	
AT89LV52-5	DIP40	/ga89c5x-5		Vpp = 5V
	PLCC44	/ga89c5x-5	PLCC44	Vpp = 5V
	TQFP44	/ga89c5x-5	TQFP44	
AT89LV55	DIP40	/ga89c5x		
	PLCC44	/ga89c5x	PLCC44	
	TQFP44	/ga89c5x	TQFP44	
AT89S51	DIP40	/ga89sxx2		
	PLCC44	/ga89sxx2	PLCC44	
	TQFP44	/ga89sxx2	TQFP44	
AT89S52	DIP40	/ga89sxx2		
	PLCC44	/ga89sxx2	PLCC44	
	TQFP44	/ga89sxx2	TQFP44	
AT89S53	DIP40	/ga89sxxxx		
	PLCC44	/ga89sxxxx	PLCC44	
	TQFP44	/ga89sxxxx	TQFP44	
AT89S8252	DIP40	/ga89sxxxx		FLASH program
	PLCC44	/ga89sxxxx	PLCC44	memory
	TQFP44	/ga89sxxxx	TQFP44	
AT89S8252E	DIP40	/ga89seeprom		EEPROM data
	PLCC44	/ga89seeprom	PLCC44	memory
	TQFP44	/ga89seeprom	TQFP44	
AT89S8253	DIP40	/ga89sxx3		FLASH program
	PLCC44	/ga89sxx3	PLCC44	memory
	TQFP44	/ga89sxx3	TQFP44	
AT89S8253E	DIP40	/ga89seeprom3		EEPROM data
	PLCC44	/ga89seeprom3	PLCC44	memory
	TQFP44	/ga89seeprom3	TQFP44	
AT90S1200	DIP20	/gavr20		FLASH program
	SOIC20	/gavr20	SOIC20	memory
AT90S1200E	DIP20	/gavr20e		EEPROM data
	SOIC20	/gavr20e	SOIC20	memory
AT90S2313	DIP20	/gavr20		FLASH program
	SOIC20	/gavr20	SOIC20	memory
AT90S2313E	DIP20	/gavr20e		EEPROM data
	SOIC20	/gavr20e	SOIC20	memory
Attiny2313	DIP20	/gavrtiny20		See remarks !!!
	SOIC20	/gavrtiny20	SOIC20	
AT90S4414	DIP40	/gavr40_2		FLASH program memory

Manufacturer	Atmel			
Device	**Package**	**Mnemonic**	**Adapter**	**Comment**
AT90S4414E	DIP40	/gavr40e_2		EEPROM data memory
AT90S8515	DIP40	/gavr40_2		FLASH program memory
AT90S8515E	DIP40	/gavr40e_2		EEPROM data memory
ATF16V8B/BQ/BQL	DIP20	/ga16v8		
	PLCC20	/ga16v8	PLCC20	
	SOIC20	/ga16v8	SOIC20	
ATF20V8B/BQ/BQL	DIP24	/ga20v8		
	PLCC28	/ga20v8	PLCC28	
	SOIC24		on request	
ATF22V10B/BQ/BQL	DIP24	/ga22v10		
	PLCC28	/ga22v10	PLCC28	
	SOIC28		on request	
ATMEGA161	DIP40	/gatmega_2		FLASH Program
	TQFP44	/gatmega_2	TQFP44	Memory
ATMEGA161	DIP40	/gatmega_e_2		EEPROM Data
EEPROM	TQFP44	/gatmega_e_2	TQFP44	Memory
ATV2500B	DIP40	/gatv2500b	DIP2500	remove 2 jumpers
	PLCC44	/gatv2500b	PLCC2500	remove 2 jumpers
ATV2500H/L	DIP40	/gatv2500	DIP2500	set 2 jumpers
	PLCC44	/gatv2500	PLCC2500	set 2 jumpers
ATV750/L	DIP28	/gatv750	DIP750	set 2 jumpers
	PLCC28	/gatv750	PLCC750	set 2 jumpers
	SOIC24		on request	
ATV750B/BQ/BL/BQL	DIP28	/gatv750b	DIP750	remove 2 jumpers
	PLCC28	/gatv750b	PLCC750	remove 2 jumpers
	SOIC24		on request	
T89C51RB2	DIP40	/gt89c51rc2-5		
	PLCC44	/gt89c51rc2-5	PLCC44	
	TQFP44	/gt89c51rc2-5	TQFP44	
T89C51RC2	DIP40	/gt89c51rc2-5		
	PLCC44	/gt89c51rc2-5	PLCC44	
	TQFP44	/gt89c51rc2-5	TQFP44	
T89C51RD2	DIP40	/gt89c51rx2-5		
	PLCC44	/gt89c51rx2-5	PLCC44	
TS87C52X2	DIP40	/gt87c5x		identical to Temic
	PLCC44	/gt87c5x	PLCC44	TS87C52X2

Manufacturer	Bright			
Device	**Package**	**Mnemonic**	**Adapter**	**Comment**
Bm29F040	DIP32	/g29fxxx		Sector Protection
	PLCC32	/g29fxxx	PLCC32	not programmable

Manufacturer	Catalyst			
Device	**Package**	**Mnemonic**	**Adapter**	**Comment**
25C128	DIP8	/g25128		
28F001	DIP32	/gi28f00x		
	PLCC32	/gi28f00x	PLCC32	
28F010	DIP32	/g28fxxx		
	PLCC32	/g28fxxx	PLCC32	
28F020	DIP32	/g28fxxx		

Manufacturer	Catalyst			
Device	**Package**	**Mnemonic**	**Adapter**	**Comment**
	PLCC32	/g28fxxx	PLCC32	
28F512	DIP32	/g28fxxx		
	PLCC32	/g28fxxx	PLCC32	
93C66	DIP8	/g93c66	SERMEM	
CAT27C210/I	DIP40	/geprom16_t1	MEM16_DIP40	Set jumper
CAT28C16A	DIP24	/g28c16		
	PLCC32	/g28c16plcc	PLCC32	
CAT28C17A	DIP28	/g28c17		
	PLCC32	/g28c16plcc	PLCC32	
CAT28C256	DIP28	/gcat28c256		
	PLCC32	/gcat28c256plcc	PLCC32	
CAT28C64B	DIP28	/gcat28c64b		
	PLCC32	/gcat28c64bplcc	PLCC32	
CAT28C65B	DIP28	/gcat28c64b		
	PLCC32	/gcat28c64bplcc	PLCC32	

Manufacturer	Dallas			
Device	**Package**	**Mnemonic**	**Adapter**	**Comment**
DS1220AB/AD	DIP24	/g28c16		
DS1225AB/AD	DIP28	/g28c64		
DS1230AB/AD	DIP28	/gsram256		
DS1245AB/AD	DIP32	/gsram1024		
DS87C520	DIP40	/gd87c5x0		Watchdog disabled
	DIP40	/gd87c5x0-w		Watchdog enabled
	PLCC44	/gd87c5x0	PLCC44	Watchdog disabled
	PLCC44	/gd87c5x0-w	PLCC44	Watchdog enabled
DS87C530	PLCC52	/gd87c5x0	PLCC52	Watchdog disabled
	PLCC52	/gd87c5x0-w	PLCC52	Watchdog enabled
DS87C550	PLCC68	/gd87c5x0	PLCC68_40	Watchdog disabled
	PLCC68	/gd87c5x0-w	PLCC68_40	Watchdog enabled
DS89C420	DIP40	/gd89cxxx		
	PLCC44	/gd89cxxx	PLCC44	

Manufacturer	Eon Silicon Devices			
Device	**Package**	**Mnemonic**	**Adapter**	**Comment**
EN29F002B/T	DIP32	/g29f00x		
	PLCC32	/g29f00x	PLCC32	
	TSOP32	/g29f00x	TSOP32	
EN29F002NB/T	DIP32	/g29fxxx		
	PLCC32	/g29fxxx	PLCC32	
	TSOP32	/g29fxxx	TSOP32	

Manufacturer	Fairchild			
Device	**Package**	**Mnemonic**	**Adapter**	**Comment**
FM27C256	DIP28	/gn27c256		
	PLCC32	/gn27c256	PLCC32_28	
FM27C512	DIP28	/gf27c512_2		
	PLCC32	/gf27c512_2	PLCC32_28	

Manufacturer	Fujitsu			
Device	**Package**	**Mnemonic**	**Adapter**	**Comment**
MBM27C1024	DIP40	/geprom16_t1	MEM16_DIP40	Set jumper
MBM27C4096	DIP40	/geprom16_t1	MEM16_DIP40	Set jumper

Manufacturer	Hitachi			
Device	**Package**	**Mnemonic**	**Adapter**	**Comment**
HN27C101AG/AP	DIP32	/gh27c101		
HN27C1024H	DIP40	/geprom16_t1	MEM16_DIP40	Set jumper
HN27C256AG/AP	DIP28	/gh27c256		
	PLCC32	/gh27c256	PLCC32_28	
HN27C301AG/AP	DIP32	/gh27c301		
HN27C4001G	DIP32	/gh27c4001		
HN27C4096	DIP40	/geprom16_t1	MEM16_DIP40	Set jumper
HN28F101	DIP32	/g28fxxx		
	PLCC32	/g28fxxx	PLCC32	

Manufacturer	Holtek			
Device	**Package**	**Mnemonic**	**Adapter**	**Comment**
93LC46	DIP8	/g93c46	SERMEM	
93LC56	DIP8	/g93c56	SERMEM	
93LC66	DIP8	/g93c66	SERMEM	
HT24LC02	DIP8	/g24c02		
HT24LC04	DIP8	/g24c04		
HT24LC08	DIP8	/g24c08		
HT24LC16	DIP8	/g24c16		
HT27C010	DIP32	/ght27c010		Also: HT27LC010
	PLCC32	/ght27c010	PLCC32	
HT27C020	DIP32	/ght27c020		Also: HT27LC020
	PLCC32	/ght27c020	PLCC32	
HT27C040	DIP32	/ght27c040		Also: HT27LC040
	PLCC32	/ght27c040	PLCC32	
HT27C4096	DIP40	/geprom16_t2	MEM16_DIP	Also: HT27LC4096
HT27C512	DIP32	/ght27c512_2		Also: HT27LC512
	PLCC32	/ght27c512_2	PLCC32_28	

Manufacturer	Hynix			
Device	**Package**	**Mnemonic**	**Adapter**	**Comment**
Hy29F002	PLCC32	/g29f00x	PLCC32	
	TSOP32	/g29f00x	TSOP32	
Hy29F040A	PLCC32	/g29fxxx	PLCC32	
	TSOP32	/g29fxxx	TSOP32	

Manufacturer	Integrated Silicon Solution Inc. (ISSI)			
Device	**Package**	**Mnemonic**	**Adapter**	**Comment**
IS24C02	dip8	/g24c02		
IS24C04	dip8	/g24c04		
IS27C010	dip32	/gis27c010		
	plcc32	/gis27c010	PLCC32	
IS27C020	dip32	/gis27c020		

Manufacturer	Integrated Silicon Solution Inc. (ISSI)			
Device	**Package**	**Mnemonic**	**Adapter**	**Comment**
	plcc32	/gis27c020	PLCC32	
IS27C2048	DIP40	/geprom16_t1	MEM16_DIP40	Set jumper
IS27C256	dip28	/gis27c256		
	PLCC32	/gis27c256	PLCC32_28	
IS27HC010	dip32	/gis27c010		
	plcc32	/gis27c010	PLCC32	
IS27HC256	dip28	/gis27c256		
	PLCC32	/gis27c256	PLCC32_28	
IS27LV010	dip32	/gis27c010		
	plcc32	/gis27c010	PLCC32	
IS27LV020	plcc32	/gis27c020	PLCC32	
IS28F010	dip32	/g28fxxx		
	plcc32	/g28fxxx	PLCC32	
IS28F020	dip32	/g28fxxx		
	plcc32	/g28fxxx	PLCC32	
IS93C46-3	DIP8	/g93c46	SERMEM	
IS93C56-3	DIP8	/g93c56	SERMEM	
IS93C66-3	DIP8	/g93c66	SERMEM	

Manufacturer	Intel			
Device	**Package**	**Mnemonic**	**Adapter**	**Comment**
27C210	DIP40	/geprom16_t1	MEM16_DIP40	Set jumper
27C220	DIP40	/geprom16_t1	MEM16_DIP40	Set jumper
27C240	DIP40	/geprom16_t1	MEM16_DIP40	Set jumper
28F001	DIP32	/g28fxxx		See relevant section
	PLCC32	/g28fxxx	PLCC32	in manual
28F010	DIP32	/g28fxxx		
	PLCC32	/g28fxxx	PLCC32	
28F020	DIP32	/g28fxxx		
	PLCC32	/g28fxxx	PLCC32	
28F256A	DIP32	/g28fxxx		
	PLCC32	/g28fxxx	PLCC32	
28F512	DIP32	/g28fxxx		
	PLCC32	/g28fxxx	PLCC32	
82802AB	PLCC32	/gi82802	PLCC32	
82802AC	PLCC32	/gi82802	PLCC32	
87C51/FA/FB/FC	DIP40	/gi87c5x		old and new version
	PLCC44	/gi87c5x	PLCC44	
87C52	DIP40	/gi87c5x		
	PLCC44	/gi87c5x	PLCC44	
87C54	DIP40	/gi87c5x		
	PLCC44	/gi87c5x	PLCC44	
87C58	DIP40	/gi87c5x		
	PLCC44	/gi87c5x	PLCC44	

Manufacturer	Lattice, SGS Thomson			
Device	**Package**	**Mnemonic**	**Adapter**	**Comment**
GAL16LV8C/D	DIP20	/gL16Lv8		Low Voltage
	PLCC20	/gL16Lv8	PLCC20	
GAL16LV8Z/ZD	DIP20	/gL16Lv8		Low Voltage
	PLCC20	/gL16Lv8	PLCC20	

Manufacturer	Lattice, SGS Thomson			
Device	**Package**	**Mnemonic**	**Adapter**	**Comment**
GAL16V8A/B/C/D	DIP20	/gL16v8		
	PLCC20	/gL16v8	PLCC20	
GAL16V8Z/ZD	DIP20	/gL16v8		
	PLCC20	/gL16v8	PLCC20	
GAL18V10/B	DIP20	/gL18v10		
	PLCC20	/gL18v10	PLCC20	
GAL20LV8C/D	DIP24	/gL20Lv8		Low Voltage
	PLCC28	/gL20Lv8	PLCC28_24	
GAL20LV8Z/ZD	DIP24	/gL20Lv8		Low Voltage
	PLCC28	/gL20Lv8	PLCC28_24	
GAL20RA10	DIP24	/gL20ra10		
	PLCC28	/gL20ra10	PLCC28_24	
GAL20V8A/B/C/D	DIP24	/gL20v8		
	PLCC28	/gL20v8	PLCC28_24	
GAL20V8Z/ZD	DIP24	/gL20v8		
	PLCC28	/gL20v8	PLCC28_24	
GAL22LV10C/D	DIP24	/gL22Lv10		Low Voltage
	PLCC28	/gL22Lv10	PLCC28_24	
GAL22LV10Z/ZD	DIP24	/gL22Lv10		Low Voltage
	PLCC28	/gL22Lv10	PLCC28_24	
GAL22V10/B/C/D	DIP24	/gL22v10		
	PLCC28	/gL22v10	PLCC28_24	
GAL22V10Z/ZD	DIP24	/gL22v10		
	PLCC28	/gL22v10	PLCC28_24	
GAL6001/B	DIP24	/gL6001		
	PLCC28	/gL6001	PLCC28_24	
GAL6002B	DIP24	/gL6002		
	PLCC28	/gL6002	PLCC28_24	

Manufacturer	Macronix			
Device	**Package**	**Mnemonic**	**Adapter**	**Comment**
MX26C512A	DIP28	/gam27c512_2		Erase with hed.eraser
	PLCC32	/gam27c512_2	PLCC32_28	
MX27C1000	DIP32	/gmx27c1000		also: MX27L1000
	PLCC32	/gmx27c1000	PLCC32	
MX27C1000A	DIP32	/gmx27c1000a		
	PLCC32	/gmx27c1000a	PLCC32	
MX27C2000	DIP32	/gmx27c2000		also: MX27L2000
	PLCC32	/gmx27c2000	PLCC32	
MX27C2000A	DIP32	/gmx27c2000a		
	PLCC32	/gmx27c2000a	PLCC32	
MX27C256	DIP28	/gis27c256		
	PLCC32	/gis27c256	PLCC32_28	
MX27C4000	DIP32	/gmx27c4000		also: MX27L4000
	PLCC32	/gmx27c4000	PLCC32	
MX27C4000A	DIP32	/gmx27c4000a		
	PLCC32	/gmx27c4000a	PLCC32	
MX27C512	DIP32	/gmx27c1000		also: MX27L1000
	PLCC32	/gmx27c1000	PLCC32	
MX27C512	DIP28	/gam27c512_2		
	PLCC32	/gam27c512_2	PLCC32_28	

Manufacturer	Macronix			
Device	**Package**	**Mnemonic**	**Adapter**	**Comment**
MX27C8000	DIP32	/gmx27c8000		
	PLCC32	/gmx27c8000	PLCC32	
MX27C8000A	DIP32	/gmx27c8000a		
	PLCC32	/gmx27c8000a	PLCC32	
MX28F1000P	DIP32	/gmx28fxxxx		
	PLCC32	/gmx28fxxxx	PLCC32	
MX29F001T/B	DIP32	/g29fxxx		Sector protection not
	PLCC32	/g29fxxx	PLCC32	programmable
MX29F002T/B	DIP32	/g29fxxx		Sector protection not
	PLCC32	/g29fxxx	PLCC32	programmable
MX29F022	DIP32	/g29fxxx		Sector protection not
	PLCC32	/g29fxxx	PLCC32	programmable
MX29F040	DIP32	/g29fxxx		Sector protection not
	PLCC32	/g29fxxx	PLCC32	programmable
MX29LV040	PLCC32	/g29lvxxx	PLCC32	Sector protection not programmable

Manufacturer	Microchip			
Device	**Package**	**Mnemonic**	**Adapter**	**Comment**
24AA16	DIP8	/g24c16		
24AA32A	DIP8	/g24xc32		
24AA64	DIP8	/g24xc64		
24C00	DIP8	/g24c00		
24C01A	DIP8	/g24c01a		
24C02A	DIP8	/g24c02		
24C04A	DIP8	/g24c04		
24C08	DIP8	/g24c08		
24C164	DIP8	/g24c16		
24C16B	DIP8	/g24c16		
24C32A	DIP8	/g24xc32		
24C65	DIP8	/g24xc64		Security nicht pro-grammierbar
24LC128	DIP8	/g24xc128		auch 24AA128, 24FC128
24LC16B	DIP8	/g24c16		
24LC256	DIP8	/g24xc256		auch 24FC256
24LC32A	DIP8	/g24xc32		
24LC64	DIP8	/g24xc64		
24LCS52	DIP8	/g24lcs52		Write Protection is irreversible
27C128	DIP28	/gm27c128		
27C256	DIP28	/gm27c256		
	PLCC32	/gm27c256	PLCC32_28	
27C512A	DIP28	/gm27c512a_2		
	PLCC32	/gm27c512a_2	PLCC32_28	
27C64	DIP28	/gm27c64		
28C04A	DIP24	/g28c04		
	PLCC32	/g28c04plcc	PLCC32	
28C16A	DIP24	/g28c16b		
	PLCC32	/g28c16bplcc	PLCC32	
28C17A	DIP28	/g28c17		
	PLCC32	/g28c16plcc	PLCC32	
28C64A	DIP28	/g28c64		

Manufacturer	Microchip			
Device	**Package**	**Mnemonic**	**Adapter**	**Comment**
	PLCC32	/g28c64plcc	PLCC32	
93C56A/B	DIP8	/g93c56	SERMEM	
93C66A/B	DIP8	/g93c66	SERMEM	
93C76	DIP8	/g93c76	SERMEM	
93C86	DIP8	/g93c86	SERMEM	
93LC46	DIP8	/g93c46	SERMEM	
93LC46A/B	DIP8	/g93c46	SERMEM	
93LC56	DIP8	/g93c56	SERMEM	
93LC66A/B	DIP8	/g93c66	SERMEM	
93LC76B	DIP8	/g93c76	SERMEM	
93LC86B	DIP8	/g93c86	SERMEM	
PIC12C508/A	DIP8	/gpic12c508	UNIPIC18	
PIC12C509/A	DIP8	/gpic12c509	UNIPIC18	
PIC12C671	DIP8	/gpic12C671	UNIPIC18	
PIC12C672	DIP8	/gpic12C672	UNIPIC18	
PIC12CE518	DIP8	/gpic12c508	UNIPIC18	
PIC12CE519	DIP8	/gpic12c509	UNIPIC18	
PIC12CE673	DIP8	/gpic12C671	UNIPIC18	
PIC12CE674	DIP8	/gpic12C672	UNIPIC18	
PIC12F629	DIP8	/gpic12f629	UNIPIC18	See manual !
PIC12F675	DIP8	/gpic12f629	UNIPIC18	See manual !
PIC16C61	DIP18	/gpic16c61	UNIPIC18	
PIC16C62	DIP28	/gpic16c62	UNIPIC	
PIC16C620	DIP18	/gpic16c620	UNIPIC18	
PIC16C620A	DIP18	/gpic16c620	UNIPIC18	
PIC16C621	DIP18	/gpic16c621	UNIPIC18	
PIC16C621A	DIP18	/gpic16c621	UNIPIC18	
PIC16C622	DIP18	/gpic16c62a	UNIPIC18	
PIC16C622A	DIP18	/gpic16c62a	UNIPIC18	
PIC16C62A	DIP28	/gpic16c62a	UNIPIC	
PIC16C62B	DIP28	/gpic16c62a	UNIPIC	
PIC16C62C	DIP28	/gpic16c62a	UNIPIC	
PIC16C63	DIP28	/gpic16c63	UNIPIC	
PIC16C64	DIP40	/gpic16c62	UNIPIC	
PIC16C64A	DIP40	/gpic16c62a	UNIPIC	
PIC16C65	DIP40	/gpic16c65	UNIPIC	
PIC16C65A	DIP40	/gpic16c63	UNIPIC	
PIC16C65B	DIP40	/gpic16c63	UNIPIC	
PIC16C66	DIP28	/gpic16c66	UNIPIC	
PIC16C67	DIP40	/gpic16c66	UNIPIC	
PIC16C71	DIP18	/gpic16c61	UNIPIC18	
PIC16C710	DIP18	/gpic16c710	UNIPIC18	
PIC16C711	DIP18	/gpic16c711	UNIPIC18	
PIC16C712	DIP18	/gpic16c621	UNIPIC18	
PIC16C716	DIP28	/gpic16c62a	UNIPIC	
PIC16C717	DIP18	/gpic16c717	UNIPIC18	
PIC16C72	DIP28	/gpic16c62a	UNIPIC	
PIC16C72A	DIP28	/gpic16c62a	UNIPIC	
PIC16C73	DIP28	/gpic16c65	UNIPIC	
PIC16C73A	DIP28	/gpic16c63	UNIPIC	
PIC16C73B	DIP28	/gpic16c63	UNIPIC	

Manufacturer	Microchip			
Device	**Package**	**Mnemonic**	**Adapter**	**Comment**
PIC16C74	DIP40	/gpic16c65	UNIPIC	
PIC16C745	DIP28	/gpic16c745	UNIPIC	
PIC16C74A	DIP40	/gpic16c63	UNIPIC	
PIC16C74B	DIP40	/gpic16c63	UNIPIC	
PIC16C76	DIP28	/gpic16c66	UNIPIC	
PIC16C765	DIP40	/gpic16c745	UNIPIC	
PIC16C77	DIP40	/gpic16c66	UNIPIC	
PIC16C770	DIP20	/gpic16c770	UNIPIC18	see Windows help for details about DIP20
PIC16C771	DIP20	/gpic16c771	UNIPIC18	see Windows help for details about DIP20
PIC16C773	DIP28	/gpic16c773	UNIPIC	
PIC16C774	DIP40	/gpic16c773	UNIPIC	
PIC16C781	DIP20	/gpic16c781	UNIPIC18	see Windows help for details about DIP20
PIC16C782	DIP20	/gpic16c770	UNIPIC18	see Windows help for details about DIP20
PIC16C84	DIP18	/gpic16c84	UNIPIC18	
PIC16C923	PLCC68	/gpic16c923		Adapter on request
PIC16C924	PLCC68	/gpic16c924		
PIC16CE623	DIP18	/gpic16c620	UNIPIC18	
PIC16CE624	DIP18	/gpic16c621	UNIPIC18	
PIC16CE625	DIP28	/gpic16c62a	UNIPIC	
PIC16CR62	DIP28	/gpic16c62a	UNIPIC	
PIC16CR64	DIP40	/gpic16c62a	UNIPIC	
PIC16CR83	DIP18	/gpic16cr83	UNIPIC18	also PIC16LCR83
PIC16CR84	DIP18	/gpic16cr84	UNIPIC18	also PIC16LCR84
PIC16F627	DIP18	/gpic16f627	UNIPIC18	also PIC16LF627
PIC16F628	DIP18	/gpic16f628	UNIPIC18	also PIC16LF628
PIC16F630	DIP14	/gpic12f629	UNIPIC18	See manual !
PIC16F676	DIP14	/gpic12f629	UNIPIC18	See manual !
PIC16F83	DIP18	/gpic16f83	UNIPIC18	also PIC16LF83
PIC16F84	DIP18	/gpic16f84	UNIPIC18	also PIC16LF84
PIC16F84A	DIP18	/gpic16f84	UNIPIC18	also PIC16LF84A
PIC16F870	DIP28	/gpic16f870	UNIPIC	
PIC16F871	DIP40	/gpic16f870	UNIPIC	
PIC16F872	DIP28	/gpic16f870	UNIPIC	
PIC16F873	DIP28	/gpic16f873	UNIPIC	
PIC16F873A	DIP28	/gpic16f873a	UNIPIC	
PIC16F874	DIP40	/gpic16f873	UNIPIC	
PIC16F874A	DIP40	/gpic16f873a	UNIPIC	
PIC16F876	DIP28	/gpic16f876	UNIPIC	
PIC16F876A	DIP28	/gpic16f876a	UNIPIC	
PIC16F877	DIP40	/gpic16f876	UNIPIC	
PIC16F877A	DIP40	/gpic16f876a	UNIPIC	
PICDATA128		/gpicdata128	UNIPIC	
PICDATA256		/gpicdata256	UNIPIC	
PICDATA64		/gpicdata64	UNIPIC	

Manufacturer	Mitsubishi			
Device	**Package**	**Mnemonic**	**Adapter**	**Comment**
M5M28F101A	DIP32	/g28fxxx		
	PLCC32	/g28fxxx		

Manufacturer	National Semiconductor (NSC)			
Device	**Package**	**Mnemonic**	**Adapter**	**Comment**
GAL16V8	DIP20	/gL16v8		identical to Lattice
	PLCC20	/gL16v8	PLCC20	GAL16V8
GAL20V8	DIP24	/gL20v8		identical to Lattice
	PLCC28	/gL20v8	PLCC28	GAL20V8
NM24C02	DIP8	/g24c02		
NM24C03	DIP8	/g24c02		
NM24C04	DIP8	/g24c04		
NM24C05	DIP8	/g24c04		
NM24C08	DIP8	/g24c08		
NM24C09	DIP8	/g24c08		
NM24C16	DIP8	/g24c16		
NM24C17	DIP8	/g24c16		
NM27C010	DIP32	/gn27c010		
	PLCC32	/gn27c010	PLCC32	
NM27C020	DIP32	/gn27c020		
	PLCC32	/gn27c020	PLCC32	
NM27C040	DIP32	/gn27c040		
	PLCC32	/gn27c040	PLCC32	
NM27C128	DIP28	/gn27c128		
NM27C16B	DIP24	/gn27c16	DIPMEM	
NM27C210	DIP40	/geprom16_t1	MEM16_DIP40	Set jumper
NM27C220	DIP40	/geprom16_t1	MEM16_DIP40	Set jumper
NM27C240	DIP40	/geprom16_t1	MEM16_DIP40	Set jumper
NM27C256	DIP28	/gn27c256		
	PLCC32	/gn27c256	PLCC32_28	
NM27C32	DIP24	/gn27c32_2		
NM27C512	DIP28	/gn27c512_2		
	PLCC32	/gn27c512_2	PLCC32_28	
NM27C64	DIP28	/gn27c64		
NM27LC256	DIP28	/gn27lc256		
NM27LC64	DIP28	/gn27c64		
NM27LV010	DIP32	/gn27c010		
NM27P040	DIP32	/gn27c040		
NM27P210	DIP40	/geprom16_t1	MEM16_DIP40	Set jumper
NM27P220	DIP40	/geprom16_t1	MEM16_DIP40	Set jumper
NM27P240	DIP40	/geprom16_t1	MEM16_DIP40	Set jumper
NM93C06	DIP8	/g93c06	SERMEM	
NM93C46	DIP8	/g93c46	SERMEM	
NM93C56	DIP8	/g93c56	SERMEM	
NM93C66	DIP8	/g93c66	SERMEM	
NM93C86B	DIP8	/g93c86	SERMEM	
NMC27C2048	DIP40	/geprom16_t1	MEM16_DIP40	Set jumper
NMC27C4096	DIP40	/geprom16_t1	MEM16_DIP40	Set jumper
NMC87C257	DIP28	/gn27c256		
	PLCC32	/gn27c256	PLCC32_28	

Manufacturer	National Semiconductor (NSC)			
Device	**Package**	**Mnemonic**	**Adapter**	**Comment**
NMX27C1024	DIP40	/geprom16_t1	MEM16_DIP40	Set jumper

Manufacturer	Philips			
Device	**Package**	**Mnemonic**	**Adapter**	**Comment**
27C210	DIP40	/geprom16_t1	MEM16_DIP40	Set jumper
27C240	DIP40	/geprom16_t1	MEM16_DIP40	Set jumper
87C504	DIP40	/gp87c5x		
	PLCC44	/gp87c5x	PLCC44	
87C51/FA/FB/FC	DIP40	/gp87c5x		
	PLCC44	/gp87c5x	PLCC44	
87C51/RA/RB/RC/RD+	DIP40	/gp87c5x		
	PLCC44	/gp87c5x	PLCC44	
87C52	DIP40	/gp87c5x		
	PLCC44	/gp87c5x	PLCC44	
87C524	DIP40	/gp87c5x		
	PLCC44	/gp87c5x	PLCC44	
87C528	DIP40	/gp87c5x		
	PLCC44	/gp87c5x	PLCC44	
87C54	DIP40	/gp87c5x		
	PLCC44	/gp87c5x	PLCC44	
87C550	DIP40	/gp87c5x		
	PLCC44	/gp87c5x	PLCC44	
87C552	PLCC68	/gp87c5x	PLCC68_40	
87C575	DIP40	/gp87c5x		
	PLCC44	/gp87c5x	PLCC44	
87C576	DIP40	/gp87c5x		
	PLCC44	/gp87c5x	PLCC44	
87C58	DIP40	/gp87c5x		
	PLCC44	/gp87c5x	PLCC44	
87C652	PLCC68	/gp87c5x	PLCC68_40	
87C654	DIP40	/gp87c5x		
	PLCC44	/gp87c5x	PLCC44	
87C748	DIP24	/gp87c7xx		
87C749	DIP28	/gp87c7xx	DIP752	
87C750	DIP24	/gp87c750		
87C751	DIP24	/gp87c7xx		
87C752	DIP28	/gp87c7xx	DIP752	
P87C51MA2	PLCC44	/gp87c51mx2	PLCC44	
P87C51MB2	PLCC44	/gp87c51mx2	PLCC44	
P87C51MC2	PLCC44	/gp87c51mx2	PLCC44	
P89C51RB2	DIP40	/gp89c5x		
	PLCC44	/gp89c5x	PLCC44	
P89C51RC+	DIP40	/gp89c5x		
	PLCC44	/gp89c5x	PLCC44	
P89C51RC2	DIP40	/gp89c5x		
	PLCC44	/gp89c5x	PLCC44	
P89C51RD+	DIP40	/gp89c5x		
	PLCC44	/gp89c5x	PLCC44	
P89C51RD2	DIP40	/gp89c5x		
	PLCC44	/gp89c5x	PLCC44	
P89C51Uxxx	DIP40	/gp89c5x		
	PLCC44	/gp89c5x	PLCC44	

Manufacturer	Philips			
Device	**Package**	**Mnemonic**	**Adapter**	**Comment**
P89C52Uxxx	DIP40	/gp89c5x		
	PLCC44	/gp89c5x	PLCC44	
P89C54Uxxx	DIP40	/gp89c5x		
	PLCC44	/gp89c5x	PLCC44	
P89C58Uxxx	DIP40	/gp89c5x		
	PLCC44	/gp89c5x	PLCC44	
P89C660	PLCC44	/gp89c5x	PLCC44	
P89C662	PLCC44	/gp89c5x	PLCC44	
P89C664	PLCC44	/gp89c5x	PLCC44	
P89C668	PLCC44	/gp89c5x	PLCC44	
PCF8582C-2	DIP8	/gpcf8582		
PCF8594C-2	DIP8	/gpcf8594		
PCF8598C-2	DIP8	/gpcf8598		

Manufacturer	PMC Flash			
Device	**Package**	**Mnemonic**	**Adapter**	**Comment**
Pm29F002B/T	DIP32	/g29fxxx		
	PLCC32	/g29fxxx	PLCC32	
Pm39F010	DIP32	/g29fxxx		
	PLCC32	/g29fxxx	PLCC32	
Pm39LV010	PLCC32	/g29fxxx	PLCC32	
	VSOP32	/g29fxxx	VSOP32	
Pm39LV512	PLCC32	/g29fxxx	PLCC32	
	VSOP32	/g29fxxx	VSOP32	
Pm49FL002	PLCC32	/g49lf00xa	PLCC32	
	VSOP32	/g49lf00xa	VSOP32	
Pm49FL004	PLCC32	/g49lf00xa	PLCC32	
	VSOP32	/g49lf00xa	VSOP32	
Pm49FL008	PLCC32	/g49lf00xa	PLCC32	
	VSOP32	/g49lf00xa	VSOP32	

Manufacturer	SGS Thomson			
Device	**Package**	**Mnemonic**	**Adapter**	**Comment**
M24C01	DIP8	/g24c01a		
M24C02	DIP8	/g24c02		
M24C04	DIP8	/g24c04		
M24C08	DIP8	/g24c08		
M24C128	DIP8	/g24xc128		
M24C16	DIP8	/g24c16		
M24C256	DIP8	/g24xc256		
M24C32	DIP8	/g24xc32		
M24C64	DIP8	/g24xc64		
M27128A	DIP28	/gs27128		
M27256	DIP28	/gs27256		
M27512	DIP28	/gs27512_2		
M2764A	DIP28	/gs2764		
M27C1000	DIP32	/gs27c1000		
M27C1001	DIP32	/gs27c1001		
	PLCC32	/gs27c1001	PLCC32	

Manufacturer	SGS Thomson			
Device	**Package**	**Mnemonic**	**Adapter**	**Comment**
M27C1024	DIP40	/geprom16_t1	MEM16_DIP40	Set jumper
M27C128A	DIP28	/gs27c128		
M27C2001	DIP32	/gs27c2001		
	PLCC32	/gs27c2001	PLCC32	
M27C202	DIP40	/geprom16_t1	MEM16_DIP40	Set jumper
M27C256B	DIP28	/gs27c256		
	PLCC32	/gs27c256	PLCC32_28	
M27C4001	DIP32	/gs27c4001		
	PLCC32	/gs27c4001	PLCC32	
M27C4002	DIP40	/geprom16_t1	MEM16_DIP40	Set jumper
M27C512	DIP28	/gs27c512_2		
	PLCC32	/gs27c512_2	PLCC32_28	
M27C64A	DIP28	/gs27c64		
M27C801	DIP32	/gs27c801_2		
M27V101	DIP32	/gs27c1001		
	PLCC32	/gs27c1001	PLCC32	
M27V201	DIP32	/gs27c2001		
	PLCC32	/gs27c2001	PLCC32	
M27V401	DIP32	/gs27c4001		
	PLCC32	/gs27c4001	PLCC32	
M27W101	DIP32	/gs27c1001		
	PLCC32	/gs27c1001	PLCC32	
M27W102	DIP40	/geprom16_t1	MEM16_DIP40	Set jumper
M27W201	DIP32	/gs27c2001		
	PLCC32	/gs27c2001	PLCC32	
M27W401	DIP32	/gs27c4001		
	PLCC32	/gs27c4001	PLCC32	
M27W402	DIP40	/geprom16_t1	MEM16_DIP40	Set jumper
M28F101	DIP32	/g28fxxx		
	PLCC32	/g28fxxx	PLCC32	
M28F201	DIP32	/g28fxxx		
	PLCC32	/g28fxxx	PLCC32	
M28F256	DIP32	/g28fxxx		
	PLCC32	/g28fxxx	PLCC32	
M28F512	DIP32	/g28fxxx		
	PLCC32	/g28fxxx	PLCC32	
M29F002B(B/T)	DIP32	/g29f00x		Sector protection not programmable
	PLCC32	/g29f00x	PLCC32	
	TSOP32	/g29f00x	TSOP32	
M29F002BN(B/T)	DIP32	/g29fxxx		Sector protection not programmable
	PLCC32	/g29fxxx	PLCC32	
	TSOP32	/g29fxxx	TSOP32	
M29F010B	DIP32	/g29fxxx		Sector protection not programmable
	PLCC32	/g29fxxx	PLCC32	
M29F040	DIP32	/g29fxxx		Sector protection not programmable
	PLCC32	/g29fxxx	PLCC32	
M29F512B	PLCC32	/g29fxxx	PLCC32	Sector protection not programmable
M29W010B	PLCC32	/g29lvxxx	PLCC32	Sector protection not programmable
M29W022B	PLCC32	/g29lvxxx	PLCC32	Sector protection not programmable
M29W040B	PLCC32	/g29lvxxx	PLCC32	Sector protection not

Manufacturer	SGS Thomson			
Device	**Package**	**Mnemonic**	**Adapter**	**Comment**
				programmable
M48T02	DIP24	/g28c16		
M48T08	DIP28	/gsram64		
M48T12	DIP24	/g28c16		
M48T18	DIP28	/gsram64		
M87C257	DIP28	/gs27c256		
	PLCC32	/gs27c256	PLCC32_28	
M93C06	DIP8	/g93c06	SERMEM	
M93C46	DIP8	/g93c46	SERMEM	
M93C56	DIP8	/g93c56	SERMEM	
M93C66	DIP8	/g93c66	SERMEM	
M93C76	DIP8	/g93c76	SERMEM	
M93C86	DIP8	/g93c86	SERMEM	
M95128	DIP8	/g25128		
M95256	DIP8	/g25256		
ST24E32	DIP8	/g24xc32		
ST24E64	DIP8	/g24xc64		
ST25E32	DIP8	/g24xc32		
ST25E64	DIP8	/g24xc64		

Manufacturer	Siemens			
Device	**Package**	**Mnemonic**	**Adapter**	**Comment**
SAB-C513A-H	PLCC44	/gc513a	PLCC44	
SAB-C501-1E	DIP40	/gc501		
	PLCC44	/gc501	PLCC44	
SAB-C505A-4E	PQFP44	/gc505a	PQFP44_C505	
SAB-C505CA-4E	PQFP44	/gc505a	PQFP44_C505	

Manufacturer	Silicon Storage Technology SST			
Device	**Package**	**Mnemonic**	**Adapter**	**Comment**
27SF010	DIP32	/gss27sf010		Programming only,
	PLCC32	/gss27sf010	PLCC32	erasure not possible
27SF020	DIP32	/gss27sf020		Programming only,
	PLCC32	/gss27sf020	PLCC32	erasure not possible
27SF256	DIP32	/gss27sf256		Programming only,
	PLCC32	/gss27sf256	PLCC32_28	erasure not possible
27SF512	DIP28	/gss27sf512_2		Programming only,
	PLCC32	/gss27sf512_2	PLCC32_28	erasure not possible
28VF040A	DIP32	/gss28vf040		Low-Voltage
	PLCC32	/gss28vf040	PLCC32	
28xF040(A)	DIP32	/gss28xf040		Not for 28VF040 !
	PLCC32	/gss28xf040	PLCC32	
29EE010	DIP32	/gss29eexxx		
	PLCC32	/gss29eexxx	PLCC32	
29EE020	DIP32	/gss29eexxx		
	PLCC32	/gss29eexxx	PLCC32	
29EE512	DIP32	/gss29eexxx		
	PLCC32	/gss29eexxx	PLCC32	
39LF010	DIP32	/g29lvxxx		Low Voltage
	PLCC32	/g29lvxxx	PLCC32	

Manufacturer	Silicon Storage Technology SST			
Device	**Package**	**Mnemonic**	**Adapter**	**Comment**
39LF020	DIP32	/g29lvxxx		Low Voltage
	PLCC32	/g29lvxxx	PLCC32	
39LF040	DIP32	/g29lvxxx		Low Voltage
	PLCC32	/g29lvxxx	PLCC32	
39LF512	DIP32	/g29lvxxx		Low Voltage
	PLCC32	/g29lvxxx	PLCC32	
39SF010A	DIP32	/g29fxxx		
	PLCC32	/g29fxxx	PLCC32	
39SF020A	DIP32	/g29fxxx		
	PLCC32	/g29fxxx	PLCC32	
39SF040	DIP32	/g29fxxx		
	PLCC32	/g29fxxx	PLCC32	
39SF512	DIP32	/g29fxxx		
	PLCC32	/g29fxxx	PLCC32	
39VF010	DIP32	/g29lvxxx		Low Voltage
	PLCC32	/g29lvxxx	PLCC32	
39VF020	DIP32	/g29lvxxx		Low Voltage
	PLCC32	/g29lvxxx	PLCC32	
39VF040	DIP32	/g29lvxxx		Low Voltage
	PLCC32	/g29lvxxx	PLCC32	
39VF512	DIP32	/g29lvxxx		Low Voltage
	PLCC32	/g29lvxxx	PLCC32	
49LF002A	PLCC32	/g49lf00xa	PLCC32	
	VSOP32	/g49lf00xa	VSOP32	
49LF003A	PLCC32	/g49lf00xa	PLCC32	
	VSOP32	/g49lf00xa	VSOP32	
49LF004A	PLCC32	/g49lf00xa	PLCC32	
	VSOP32	/g49lf00xa	VSOP32	
49LF004B	PLCC32	/g49lf00xa	PLCC32	
	VSOP32	/g49lf00xa	VSOP32	
49LF008A	PLCC32	/g49lf00xa	PLCC32	
	VSOP32	/g49lf00xa	VSOP32	
49LF020	PLCC32	/g49lf00xa	PLCC32	
	TSOP32	/g49lf00xa	TSOP32	
49LF040	PLCC32	/g49lf00xa	PLCC32	
	TSOP32	/g49lf00xa	TSOP32	
89F54	DIP40	/gss89f5x_0		FLASH Block 0
	PLCC44	/gss89f5x_0	PLCC44	
89F54	DIP40	/gss89f5x_1		FLASH Block 1
	PLCC44	/gss89f5x_1	PLCC44	
89F58	DIP40	/gss89f5x_0		FLASH Block 0
	PLCC44	/gss89f5x_0	PLCC44	
89F58	DIP40	/gss89f5x_1		FLASH Block 1
	PLCC44	/gss89f5x_1	PLCC44	

Manufacturer	Temic Semiconductors			
Device	**Package**	**Mnemonic**	**Adapter**	**Comment**
TSC87C51	DIP40	/gt87c5x		has no lock bits
	PLCC44	/gt87c5x	PLCC44	

More MCS51 Microcontroller: see Atmel

Manufacturer	Texas Instruments			
Device	**Package**	**Mnemonic**	**Adapter**	**Comment**
TMS27C010A	DIP32	/gt27c010		
	PLCC32	/gt27c010	PLCC32	
TMS27C020	DIP32	/gt27c020		
	PLCC32			on request
TMS27C040	DIP32	/gt27c040		
	PLCC32	/gt27c040	PLCC32	
TMS27C128	DIP28	/gt27c128		
TMS27C210	DIP40	/geprom16_t1	MEM16_DIP40	Set jumper
TMS27C240	DIP40	/geprom16_t1	MEM16_DIP40	Set jumper
TMS27C256	DIP28	/gt27c256		
	PLCC32	/gt27c256	PLCC32_28	
TMS27C510	DIP32	/gt27c510		
	PLCC32	/gt27c510	PLCC32	
TMS27C512	DIP28	/gt27c512_2		
	PLCC32	/gt27c512_2	PLCC32_28	
TMS27PC210	DIP40	/geprom16_t1	MEM16_DIP40	Set jumper
TMS27PC240	DIP40	/geprom16_t1	MEM16_DIP40	Set jumper

Manufacturer	Toshiba			
Device	**Package**	**Mnemonic**	**Adapter**	**Comment**
TC571000AD	DIP32	/gam27c010		
TC571001AD	DIP32	/gam27c010		
TMP88PH40M/N	DIP28	/gtmp88ph40	Toshiba BM11196	Use special adapter
	SOIC28	/gtmp88ph40	Toshiba BM11195	made by Toshiba
TMP88PS43F	QFP80	/gtmp88ps43		Use special adapter made by Toshiba

Manufacturer	Winbond			
Device	**Package**	**Mnemonic**	**Adapter**	**Comment**
W27C020	DIP32	/gw27c020		programming only, erasure not possible.
	PLCC32	/gw27c020	PLCC32	
W27C040	DIP32	/gw27c040		programming only, erasure not possible.
	PLCC32	/gw27c040	PLCC32	
W27C4096	DIP40	/geprom16_t1	MEM16_DIP40	Set jumper
W27C512	DIP28	/gw27e512_2		programming only, erasure not possible
	PLCC32	/gw27e512_2	PLCC32_28	
W27E010	DIP32	/gw27e010		programming only, erasure not possible. Also: 27c010
	PLCC32	/gw27e010	PLCC32	
W27E257	DIP28	/gw27e257		programming only, erasure not possible
	PLCC32	/gw27e257	PLCC32_28	
W27E512	DIP28	/gw27e512_2		programming only, erasure not possible
	PLCC32	/gw27e512_2	PLCC32_28	
W29C011A	DIP32	/gw29eexxx		
	PLCC32	/gw29eexxx	PLCC32	
W29C020/C	DIP32	/gw29eexxx		
	PLCC32	/gw29eexxx	PLCC32	
W29C040	DIP32	/gw29eexxx		
	PLCC32	/gw29eexxx	PLCC32	
W29EE011	DIP32	/gw29eexxx		

Manufacturer	Winbond			
Device	**Package**	**Mnemonic**	**Adapter**	**Comment**
	PLCC32	/gw29eexxx	PLCC32	
W29EE512	DIP32	/gw29eexxx		
	PLCC32	/gw29eexxx	PLCC32	
W39L040	PLCC32	/g29lvxxx	PLCC32	
W39L040A	PLCC32	/g29lvxxx	PLCC32	
W39V040AP	PLCC32	/g49lf00xa	PLCC32	
W39V040FAP	PLCC32	/g49lf00xa	PLCC32	
W49F002	DIP32	/g49f00x		Bottom Boot Block Protection and Reset-Pin
	PLCC32	/g49f00x	PLCC32	
W49F002B	DIP32	/g29fxxx		Bottom Boot Block Protection, without Reset-Pin
	PLCC32	/g29fxxx	PLCC32	
W49F002N	DIP32	/g29fxxx		Top Boot Block Protection, without Reset-Pin
	PLCC32	/g29fxxx	PLCC32	
W49F002U	DIP32	/g49f00x		Top Boot Block Protection and Reset-Pin
	PLCC32	/g49f00x	PLCC32	
W49F020	DIP32	/g29fxxx		Boot Block Protection
	PLCC32	/g29fxxx	PLCC32	
W49V002AP	PLCC32	/g49lf00xa	PLCC32	
W49V002FAP	PLCC32	/g49lf00xa	PLCC32	
W77E58/P	DIP40	/gw78e58		
	PLCC44	/gw78e58	PLCC44	
W78E516B/P EPROM	DIP40	/gw78e516eprom		4 kB EPROM Loader Memory (LDROM)
	PLCC44	/gw78e516eprom	PLCC44	
W78E516B/P FLASH	DIP40	/gw78e516flash		64 kB FLASH Program Memory (APROM)
	PLCC44	/gw78e516flash	PLCC44	
W78E51B/P	DIP40	/gw78e51b		
	PLCC44	/gw78e51b	PLCC44	
W78E52B/P	DIP40	/gw78e52b		
	PLCC44	/gw78e52b	PLCC44	
W78E54/B/P/M	DIP40	/gw78e54		
	PLCC44	/gw78e54	PLCC44	
	TQFP44	/gw78e54	TQFP44	
W78E58/P	DIP40	/gw78e58		
	PLCC44	/gw78e58	PLCC44	
W78LE812/P	DIP40	/gw78e52b		
	PLCC44	/gw78e52b	PLCC44	

Manufacturer	Xicor			
Device	**Package**	**Mnemonic**	**Adapter**	**Comment**
X25020	DIP8	/gx25020		
X25040	DIP8	/gx25040		
X25080	DIP8	/g25080		
X25128	DIP8	/g25128		
X25160	DIP8	/g25160		
X25320	DIP8	/g25320		
X25640	DIP8	/g25640		
X25642	DIP8	/g25640		
X25F008	DIP8	/g25080		
X25F016	DIP8	/g25160		
X25F032	DIP8	/g25320		
X25F064	DIP8	/g25640		
X25F128	DIP8	/g25128		

Manufacturer	Xicor			
Device	**Package**	**Mnemonic**	**Adapter**	**Comment**
X28HC64	DIP28	/g28c64b		
	PLCC32	/g28c64bplcc	PLCC32	