NAME

p2bin - convert code files into hex files

SYNTAX

p2bin [option(s)] <name(s)> [further options/names]

DESCRIPTION

P2BIN is a tool to convert the contents of one or several code files generated by AS into binary files. A binary file is a 1:1 memory image of the processor's memory and is especially suited for EPROM programmers and emulators.

Arguments to P2BIN may be either command line parameters or file name specifications. Any argument that starts with the charactes +, - or / is regarded as a comand line parameter (which may take an additional command line argument); any other argument is regarded as a file name. Generally, P2BIN needs at least two file names: An input code file and the name of the binary output file. If multiple file names are given, P2BIN will always take the last name as the output file's name. If an input file name does not have an extension, the extension '.p' is added automatically. Similarly, the extension '.bin' is added automatically to the target file's name. A special case occurs when only one file name is given: P2BIN will then take its name as the source (possibly extended with '.p'), and the same name as target (with '.bin' as additional or replaced extension).

COMMAND-LINE PARAMETERS

If a command-line parameter starts with a slash(/) or minus sign(-), it turns an option on; if a command-line parameter starts with a plus sign(+), it turns a specific option off. Numeric arguments to parameters can be either written in decimal or hexadecimal notation. For hexadecimal notation, prefix the number with a dollar($) sign. In the following list, all options will be shown in the form that is needed to change the default behaviour, which might be a plus or minus sign, depening on wether the option is on or off by default.

p2bin accepts the following command-line parameters:

-f <number>[,<further numbers>]

Add <number> to the list of record header IDs that allow a record from a source file to be written to the target file. A certain header ID marks code for a certain target processor family; thus, this filter allows to distill code for a certain processor out of a source file that contains code for different processor families. Negation of this parameter removes certain header IDs from P2BIN's list. See the user manual of AS for a list of all possible header ID values. If P2BIN's list of header IDs is empty, no filtering will take place, i.e. all records from a source file will make it into the target file.

-l <number>

Set the value that should be used to fill memory areas in the

binary image that are unused by the input code file(s). The default for this is to fill unused areas with the value 255 (0xff), which is the best choice for EPROMs as erased EPROM cells carry this value and an intelligent EPROM burner can skip these cells, speeding up the programming process and reducing stress for the EPROM. However, there are specialized EPROMs that have zeros in their cells after erasure, and you might want to fill unused areas with a code that executes as a NOP or BREAK statement.

-m <all|even|odd|byte<0|1|2|3>|word<0|1>>

Set the mask of bytes to be filtered out. If your target processor has a 16- or 32-bit data path, but your EPROMs are only 8- or 16-bits wide, the code has to be spread over the EPROMs in an alternating fashion. This option allows you to do the necesary splitting, however you have to run P2BIN two or four times with different masks. The possible arguments have the following meanings:

all does not do any filtering, i.e. all bytes of your code will show up in the resulting image. This is the default.

even or **odd** will take only those bytes whose addresses are in the form 2*n or 2*n+1. They are useful if you have a 16-bit data path and two 8-bit EPROMs.

byte0, **byte1**, **byte2** or **byte3** will take only those bytes whose addresses are in the form 4*n ... 4*n+3. They are useful if you have a 32-bit data path and four 8-bit EPROMs.

word0 or **word1** will take only those bytes whose addresses are in the form 4*n+0 / 4*n+1 or 4*n+2 / 4*n+3. They are useful if you have a 32-bit data path and two 16-bit EPROMs.

When using one of these filters, the resulting images will automatically become smaller by a factor of 2 or 4. Beware that this does not influence address specifications given with the **-r** command-line parameter! See also the examples section below for correct usage.

-r < <start>-<stop> >

Set a certain address range to be filtered out of the input file(s). Code that lies outside this range does not appear in the output file. The default for the address filter is the 0-$7fff, which might create confusion in some cases. As a special option, **<start>** and **<stop>** may consist of just a single dollar sign (escape this in UNIX shells!) to signify the lowest resp. highest address that occurs in the input file(s). Using this option will implicitly enable a second pass over all input files to find the minimum and maximum values before conversion starts, reducing the speed of P2BIN slightly.

-segment <CODE|DATA|....>

Select the address space hex data is created from. By default,

only records for the CODE segment (plus DATA for TI DSK) will be considered. Use this option with different arguments if the source file contains data from other address spaces. This way, multiple HEX files (one per address space) can be produced.

-e <address>

Set an entry address or modify an existing one. P2BIN can optionally prepend the start address to the binary image to tell a program loader where to jump after the image has been loaded (see the '-S' option). Normally, this address is generated by AS if the program's END statement has a label as argument, but this options allows to change the entry point or add one if it was forgotten in the program itself.

-S [L|B]<n>

Instruct P2BIN to prepend the program entry address to the image. 'n' is the length in bytes the address should have and has an allowed range from 1 to 4. The number may be prefixed by a 'L' or 'B' letter that sets the endianess of the address. If no letter is used, little endian is assumed.

-s

Tell P2BIN to include a checksum into the image. A checksum is a byte value entered into the image's last byte that is the two's complement of the sum of all previous bytes. Therefore, the sum of all bytes modulus 256 will become zero. This option is useful if you want to check the ROM contents in your program as part of a power-on self-test, but keep in mind that you must not use the last byte for your own purposes any more!

-k

Instruct P2BIN to erase the program source files after conversion.

-q or -quiet

Enable quiet operation mode, suppressing copyright and purely informative messages. Only errors will be displayed.

PRESETTING PARAMETERS

Parameters need not neccessarily be given in the command line itself. Before processing of command line parameters starts, P2BIN will look if the **P2BINCMD** environment variable is defined. If it exists, its contents will be treated as additional command line paramters whose syntax is absolutely equal to normal command line parameters. As exception is made if the variable's contents start with a '@' sign; in such a case, the string after the '@' sign is treated as the name of a file that contains the options. Such a file (also called a 'key file') has the advantage that it allows the options to be written in different lines, and it does not have a size limit. Some operating systems (like MS-DOS) do have a length limit on command lines and environment variable contents, so the key file may be your only option if you have a lot of lengthy parameters for P2BIN.

RETURN CODES

p2bin may return with the following codes:

0 no errors.

1 incorrect command line parameters.

2 I/O-error.

3 An input file had an incorrect format.

EXAMPLES

To convert a file **file1.p** fully into its binary representation, use

p2bin -r \$-\$ file1

If you have a processor with a 64 KByte address space and a 16-bit data path and you want to assure that the memory image always starts at address 0, regardless of address layout in the code file, use

p2bin -r 0-\$ffff -m even file1 evenfile

p2bin -r 0-\$ffff -m odd file1 oddfile

to get images for two 27256 EPROMs.

NATIONAL LANGUAGE SUPPORT

p2bin supports national languages in the same way as AS. See the manual page for asl(1) for more information about this.

TIPS

Calling P2BIN without any arguments will print a short help listing all command line parameters.

SEE ALSO

asl(1), plist(1), pbind(1), p2hex(1)

HISTORY

P2BIN originally appeared as an AS tool in 1992, written in Borland-Pascal, and was ported to C and UNIX in 1996.

BUGS

Command line interpreters of some operating systems reserve some characters for their own use, so it might be necessary to give command line parameters with certain tricks (e.g., with the help of escape characters).

P2BIN does not have so far an opportunity to filter records by target segment. Instead, records that contain data for any other segment than CODE are completely ignored.

AUTHOR(S)

Alfred Arnold (alfred@ccac.rwth-aachen.de)