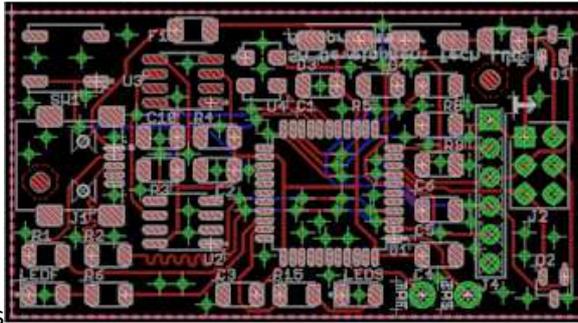


Autoprogrammer Manual

SA Development Tech LLC
Firmware 1.11 PC Application 1.12



Disclaimer:

Any electronic device can fail. To be used exclusively as industrial test equipment. SA Development makes no guarantees and is not liable for any issues that may arise from its use or malfunction.

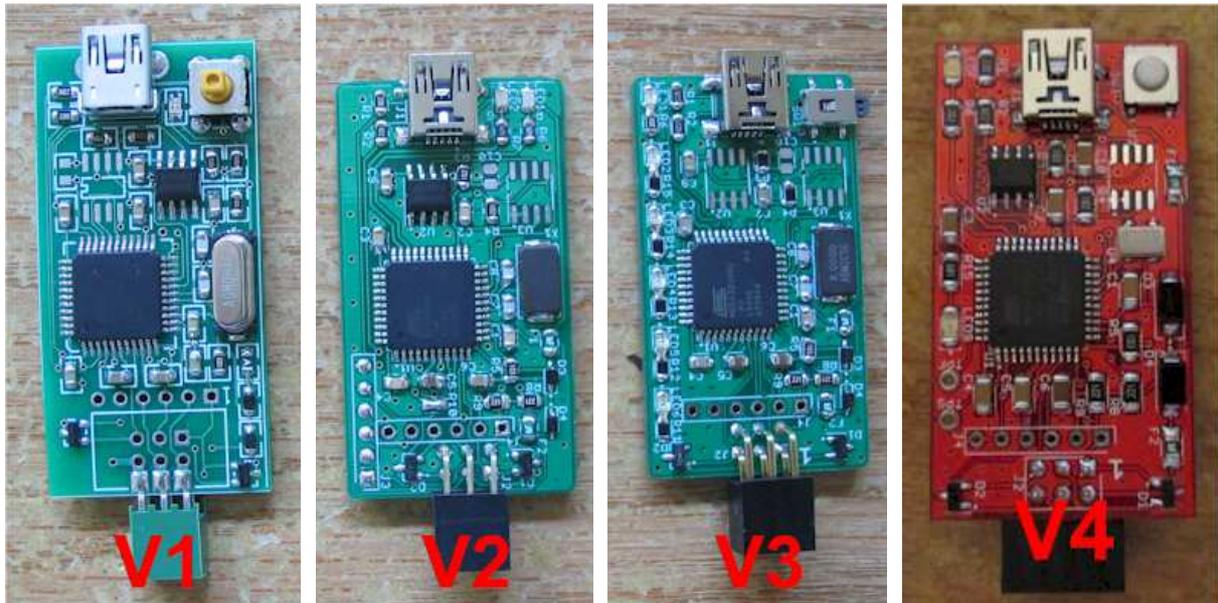
Table of Contents:

1. Introduction	2
2. Detailed Features.....	4
3. Error Codes.....	7
4. PC Application Operation	8
5. Autoprogrammer Board Operation	15
6. Programming Methodology	16
7. Simple Configuration Example.....	17
8. Multiple Device Variant Example.....	19
9. Dual Configuration Example.....	20
10. Multiple Match Select Example.....	22
11. Multiple Match Advanced Example	24
12. Version Compatibility	27
13. Release Notes	28

1. Introduction

Thank you for purchasing an Autoprogrammer. We hope you enjoy it for many years to come!

There have been four versions so far:



V1 – ISP programming only, Red LED and Green LED, Multiple Match Select Button. The yellow tipped tactile switch and silver crystal can be used to identify V1.

V2 – ISP and PDI programming, Red LED and Green LED. The lack of a right angle tactile switch can be used to identify V2.

V3 – ISP and PDI programming, Red LED and Green LED 5 segment graph, Multiple Match Select Button. The right angle tactile switch and led strip on the left edge can be used to identify V3. (The V3 in this picture doesn't have an EEPROM yet, but normally does.)

V4 – ISP and PDI programming, Red LED and Green LED, Multiple Match Select Button. The real oscillator (silver) and larger 1206 sized parts can be used to identify V4.

This tiny board is a fully automated programmer.

The original intent of this device was to update a product in the field by someone unfamiliar with programming. It requires only that someone can line up pin 1 (you could send a picture showing the right way to connected it) and turn it on. The autoprogrammer does the rest. Its

small size makes it very low cost to send out in a bubble envelope. You can even include a smaller return addressed stamped envelope to get it back when the customer is done using it.

I found that it also excels at in house production programming. It is so much more convenient to grab it and connect it to a board than it is to either take that board to a PC or bring a PC or notebook to the board. For this reason, another use is also that it can easily go where taking a PC or notebook might be difficult or impractical.

It is set apart from other self contained programmers because it has a multiple microcontroller multiple firmware configuration capability. This allows for a single autoprogammer with a single loaded configuration to handle multiple projects, different firmware versions or variants of the same project, and/or different AVR variants that need the same firmware all at the same time.

It supports flashing 2.7V-5V targets and acquires power from the programming header.

Can have up to 512KB EEPROM across two banks.

Can support either a female connector to direct plug or a male connector to use a 2x3 IDC ribbon cable.

It is so handy you won't even believe how much you use it! It is faster for me to load up the pc application and create a configuration to do programming than it is for me to drag my notebook into another room.

Ability to assign a serial number during flashing.

2. Detailed Features

Standalone programming does not require PC.

ISP protocol support from 62.5 kHz to 2 MHz.

PDI protocol support from 62.5 kHz to 2 MHz. (V2 and higher)

Fully automated programming requires no interaction from end user – you can send the device to anyone and all they have to do is plug it in and turn it on.

TAGTOOL support means that it supports a flash based method of assigning a unique serial number at the time of programming. This tool also allows protecting flash with a CRC16.

Multiple microcontroller multiple firmware configuration allows a single configuration to carry out multiple tasks. A matching process allows you to match a device by its device signature and optionally any fuse or lockbits to a particular firmware you want it to flash. One way this can be used is to have a single configuration update completely different devices. You could send one autoprogrammer to your customer and it can update 4 different products or different AVR's in the same product. Another benefit is that it can be used to handle different variants of the same basic AVR. You might have a product that was built with both ATMEGA328 and ATMEGA328P and each has its own device signature. Simply match both devices and point them to the same firmware to update and it will flash whichever it encounters. For in house production programming, I often have a need to flash two different versions of firmware for the same device. If I create a configuration where more than one match is possible, it will prompt me to select which one I wish to flash (all except V2). See the example sections at the end of this manual for various configurations you could use.

A flash limit can be set to only allow a specific number of flashes for each firmware. You can also restrict the matching to only update a device that has the expected device signature, expected fuses, and expected lockbits settings.

It keeps track of the number of successful flashes for each firmware; this information can be later retrieved using the pc application.

It can program flash and EEPROM, or program and verify flash and EEPROM. Fuse and lock its settings are always programmed and verified.

Programming settings are built in for these devices: AT90CAN128, AT90CAN32, AT90CAN64, AT90PWM1, AT90PWM161, AT90PWM216, AT90PWM2B, AT90PWM316, AT90PWM3B, AT90PWM81, AT90USB1286, AT90USB1287, AT90USB162, AT90USB646, AT90USB647, AT90USB82, ATmega128, ATmega1280, ATmega1281, ATmega1284, ATmega1284P, ATmega1284RFR2, ATmega128A, ATmega128RFA1, ATmega128RFR2, ATmega16, ATmega162, ATmega164A, ATmega164P, ATmega164PA, ATmega165A, ATmega165P, ATmega165PA, ATmega168, ATmega168A, ATmega168P, ATmega168PA, ATmega168PB, ATmega169A, ATmega169P, ATmega169PA, ATmega16A, ATmega16HVA, ATmega16HVB,

ATmega16HVBREVB, ATmega16M1, ATmega16U2, ATmega16U4, ATmega2560, ATmega2561, ATmega2564RFR2, ATmega256RFR2, ATmega32, ATmega324A, ATmega324P, ATmega324PA, ATmega324PB, ATmega325, ATmega3250, ATmega3250A, ATmega3250P, ATmega3250PA, ATmega325A, ATmega325P, ATmega325PA, ATmega328, ATmega328P, ATmega328PB, ATmega329, ATmega3290, ATmega3290A, ATmega3290P, ATmega3290PA, ATmega329A, ATmega329P, ATmega329PA, ATmega32A, ATmega32C1, ATmega32HVB, ATmega32HVBREVB, ATmega32M1, ATmega32U2, ATmega32U4, ATmega48, ATmega48A, ATmega48P, ATmega48PA, ATmega48PB, ATmega64, ATmega640, ATmega644, ATmega644A, ATmega644P, ATmega644PA, ATmega644RFR2, ATmega645, ATmega6450, ATmega6450A, ATmega6450P, ATmega645A, ATmega645P, ATmega649, ATmega6490, ATmega6490A, ATmega6490P, ATmega649A, ATmega649P, ATmega64A, ATmega64C1, ATmega64HVE2, ATmega64M1, ATmega64RFR2, ATmega8, ATmega8515, ATmega8535, ATmega88, ATmega88A, ATmega88P, ATmega88PA, ATmega88PB, ATmega8A, ATmega8HVA, ATmega8U2, ATtiny13, ATtiny13A, ATtiny1634, ATtiny167, ATtiny2313, ATtiny2313A, ATtiny24, ATtiny24A, ATtiny25, ATtiny26, ATtiny261, ATtiny261A, ATtiny4313, ATtiny43U, ATtiny44, ATtiny441, ATtiny44A, ATtiny45, ATtiny461, ATtiny461A, ATtiny48, ATtiny828, ATtiny84, ATtiny841, ATtiny84A, ATtiny85, ATtiny861, ATtiny861A, ATtiny87, ATtiny88, ATxmega128A1, ATxmega128A1U, ATxmega128A3, ATxmega128A3U, ATxmega128A4U, ATxmega128B1, ATxmega128B3, ATxmega128C3, ATxmega128D3, ATxmega128D4, ATxmega16A4, ATxmega16A4U, ATxmega16C4, ATxmega16D4, ATxmega16E5, ATxmega192A3, ATxmega192A3U, ATxmega192C3, ATxmega192D3, ATxmega256A3, ATxmega256A3B, ATxmega256A3BU, ATxmega256A3U, ATxmega256C3, ATxmega256D3, ATxmega32A4, ATxmega32A4U, ATxmega32C3, ATxmega32C4, ATxmega32D3, ATxmega32D4, ATxmega32E5, ATxmega384C3, ATxmega384D3, ATxmega64A1, ATxmega64A1U, ATxmega64A3, ATxmega64A3U, ATxmega64A4U, ATxmega64B1, ATxmega64B3, ATxmega64C3, ATxmega64D3, ATxmega64D4, ATxmega8E5

It can support for new or unknown devices by specifying a custom device signature, flash page size, and EEPROM page size. This will allow it to operate with devices that are not on the above list. New devices can be added to the above list in a PC software update without requiring a firmware update.

If a failure occurs during the programming process including the final lock bits write and verify, a chip erase will occur and leave the device unprogrammed.

Encryption is used for on board EEPROM as well as on disk configuration storage to prevent anyone from extracting code. Configurations can be uploaded to the board and tested. You can download from a board only if the board is not locked.

Everything is CRC protected to ensure data integrity and compression is supported for flash and EEPROM images to reduce configuration size.

USB design uses no PC side driver for simplicity, just plug it in and start up the PC application to manage a board. USB protocol between PC application and board is also encrypted and CRC protected.

It supports configuration storage of up to 512KB. There are two EEPROM banks for a 64KB, 128KB, or 256KB EEPROM each. Any size in any bank is valid.

Board is dual powered. For device programming it is powered by the device being programmed and supports 2.7V-5V. To be programmed with a configuration, it is USB powered when plugged into a PC. It is not intended to be plugged into both USB and a target to program at the same time and will switch to USB mode. It also has components to protect a lower voltage target from USB voltage (5V).

All boards have a Red LED which will blink out a two digit error code if something goes wrong.

V1, V2, and V4 boards have a single Green LED to indicate status. It will blink during programming and go solid when successful.

The V3 board has a 5 segment Green LED graph to indicate status. They will show a graph and blinking when programming and all be on solid when successful.

Board has extensive self test features for reliability.

PC application can load and save configurations to disk.

PC application allows to users to choose which protocols are enabled (ISP, PDI, or both) and the programming frequency for each firmware entry.

PC application has an identification string that can be used to describe the contents of a configuration, what versions are included, etc. When identifying a board it will indicate the contents of this string so the user can know what it was programmed with.

PC application shows the total configuration size. The user can use this to determine if their board has enough EEPROM to hold the configuration.

PC application supports up to 250 microcontroller entries, each one identifying a microcontroller signature and fuses or lock bits to match it by.

PC application supports up to 30 firmware entries, each one holding firmware, EEPROM, fuse, and lockbits settings.

PC application loads HEX files, ELF files, or BIN files. ELF files can import flash, EEPROM, fuse, and lockbits settings at one time.

PC application has single button tasks for board identification, program/verify/test, program, verify, test, quick erase, and secure erase. These buttons will automatically perform the action on all attached boards.

PC application has an EEPROM fill feature which allows writing a single byte value to a specific number of bytes; this can be used to erase or set EEPROM even if the EESAVE fuse is enabled.

PC application can handle multiple boards plugged in at the same time.

3. Error Codes

The Red LED will blink out a two digit error code if anything goes wrong. There will be a 0.75 second delay between digits and a 3 second delay between codes.

<code>#define ERRDIAG_UNABLE_TO_ENTER_PROGRAM_MODE</code>	11
<code>#define ERRDIAG_UNABLE_TO_REENTER_PROGRAM_MODE</code>	12
<code>#define ERRDIAG_UNABLE_TO_READ_SIGNATURE</code>	13
<code>#define ERRDIAG_UNABLE_TO_READ_FUSES_OR_LOCKBITS</code>	14
<code>#define ERRDIAG_UNABLE_TO_ERASE</code>	15
<code>#define ERRDIAG_UNABLE_TO_WRITE_FLASH</code>	16
<code>#define ERRDIAG_UNABLE_TO_WRITE_EEPROM</code>	17
<code>#define ERRDIAG_UNABLE_TO_WRITE_FUSE</code>	18
<code>#define ERRDIAG_UNABLE_TO_WRITE_LOCKBITS</code>	19
<code>#define ERRDIAG_FLASH_VERIFY_FAILURE</code>	21
<code>#define ERRDIAG_EEPROM_VERIFY_FAILURE</code>	22
<code>#define ERRDIAG_COMMUNICATION_FAILURE</code>	23
<code>#define ERRDIAG_NO_SUPPORTED_PROTOCOLS_ARE_ENABLED</code>	24
<code>#define ERRDIAG_NO_MICROCONTROLLER_MATCH_FOUND_IN_STRUCTURE</code>	25
<code>#define ERRDIAG_FLASH_LIMIT_REACHED</code>	26
<code>#define ERRDIAG_CONFIG_NO_SIGNATURE</code>	27
<code>#define ERRDIAG_CONFIG_VERSION_MISMATCH</code>	28
<code>#define ERRDIAG_CONFIG_FAILS_CRC</code>	29
<code>#define ERRDIAG_EEPROM_FAILURE</code>	31
<code>#define ERRDIAG_SELFTEST_SIGNATURE_FAILURE</code>	32
<code>#define ERRDIAG_SELFTEST_FUSE_FAILURE</code>	33
<code>#define ERRDIAG_SELFTEST_LOCKBITS_FAILURE</code>	34
<code>#define ERRDIAG_SELFTEST_CRC_FAILURE</code>	35
<code>#define ERRDIAG_PORT_CONFLICT_DETECTED</code>	36
<code>#define ERRDIAG_DEVICE_CHANGED</code>	37

4. PC Application Operation

The PC application is where the user will build a configuration to program to the autoprogrammer board via USB. A single Green LED will flash when it is trying to connect to USB and go solid when connected.

It has 5 main areas – each area is covered in greater detail below.

Configuration – This section allows you to load and save configurations to disk.

Main – This section allows you to specify which protocols are enabled and the identification string.

Microcontroller Entries – This section allows you to specify which microcontrollers will be programmed.

Firmware Entries – This section contains the flash, EEPROM, fuse, and lockbits to program along with programming settings.

Board – This section allows you to do operations on one or more connected autoprogrammer boards.

SA Development Autoprogrammer 1.10b

Configuration
New Load Save Save As File: Untitled
Configuration Size 150

Main
Enabled Protocols: ISP and PDI Identification String:
 Lock Configuration (Disables Retrieve and Set Serial No.)

Microcontroller Entries

uc0	Device Name	(unknown or custom)	Select Device	<input type="checkbox"/> Match Fuse 0 / Low	00
	Flash Size	(unknown)		<input type="checkbox"/> Match Fuse 1 / High	00
	EEPROM Size	(unknown)	Edit Device	<input type="checkbox"/> Match Fuse 2 / Extended	00
	Match Signature	0x00 0x00 0x00		<input type="checkbox"/> Match Fuse 3	00
	Flash Page Size	0		<input type="checkbox"/> Match Fuse 4	00
	EEPROM Page Size	0		<input type="checkbox"/> Match Fuse 5	00
	Poll Method	Command		<input type="checkbox"/> Match Fuse 6	00
	Program Firmware	prg0		<input type="checkbox"/> Match Lockbits	00

Firmware Entries

prg0	Flash	0 bytes	Load HEX	Load BIN	Clear	Load ELF	<input type="checkbox"/> Program Fuse 0 / Low	00
	Compression	None					<input type="checkbox"/> Program Fuse 1 / High	00
	CRC	0x0000					<input type="checkbox"/> Program Fuse 2 / Extended	00
	EEPROM	0 bytes	Load HEX	Load BIN	Clear	Fill	<input type="checkbox"/> Program Fuse 3	00
	Compression	None					<input type="checkbox"/> Program Fuse 4	00
	CRC	0x0000					<input type="checkbox"/> Program Fuse 5	00
	Program Frequency	125 kHz					<input type="checkbox"/> Program Fuse 6	00
	Program Method	Program and Verify					<input type="checkbox"/> Program Lockbits	00
	<input checked="" type="checkbox"/> Auto Select							
	<input type="checkbox"/> Do Not Attempt Compression When Loading							
	<input type="checkbox"/> Limit Flashes							
	<input type="checkbox"/> Set Serial No.							

Board
Identify Program and Test Program Test Quick Erase Secure Erase Retrieve Set Serial No.

Configuration Section



A configuration contains everything the autoprogrammer board needs to do its job. It has everything from which protocols are enabled to which microcontrollers will be matched to the actual flash, EEPROM, and fuse/lockbits settings to program.

It also shows the current filename and the configuration size in bytes.

New allows you to clear out the current configuration and start with an empty configuration.

Load allows you to load a configuration from a disk file.

Save allows you to save a configuration to a disk file. If no filename has been provided, it will prompt for one.

Save As always prompts for a filename so you can save to a different filename.

Main Section



You can select which protocols it attempts. You can leave it on both ISP and PDI, but if you only require one protocol or the other, you can set it to only attempt that protocol.

The identification string helps you identify a configuration, it is a great place to put the description of the configuration along with version numbers, etc. The identify command will show this string even on a locked board.

Lock configuration will prevent the use of the retrieve button or setting the serial number.

Microcontroller Entries Section

The screenshot shows a window titled "Microcontroller Entries". On the left, there is a list box containing "uc0". Below the list are "Add" and "Del" buttons. To the right of the list are up and down arrow buttons. The main area contains the following fields:

- Device Name: (unknown or custom)
- Flash Size: (unknown)
- EEPROM Size: (unknown)
- Match Signature: 0x00 0x00 0x00
- Flash Page Size: 0
- EEPROM Page Size: 0
- Poll Method: Command
- Program Firmware: prg0 (dropdown menu)

On the right side, there are two buttons: "Select Device" and "Edit Device". Below these are seven checkboxes, each followed by a "00" value field:

- Match Fuse 0 / Low
- Match Fuse 1 / High
- Match Fuse 2 / Extended
- Match Fuse 3
- Match Fuse 4
- Match Fuse 5
- Match Fuse 6
- Match Lockbits

These entries are used to match a device to a specific firmware to be programmed.

On the left side is an Add button for add and a Del button for delete. It is worth noting that deleting a microcontroller will cause the list to renumber. For example if you have uc0, uc1, and uc2 and you delete uc0, it will renumber uc1 to uc0 and uc2 to uc1.

There are also arrows to the right that allow you to move configurations up and down to set which one is matched first.

To the right there is a group of six informational fields for Device Name, Flash Size, EEPROM Size, Match Signature, Flash Page Size, and EEPROM Page Size. The top three fields are always looked up from the signature in the Match Signature field. If the signature is known, the top fields are filled in; otherwise they will say (Unknown).

The Match Signature field is **always used** to match a device. If a device signature is not the same, this microcontroller entry will be skipped and others that might match the signature will be evaluated.

The Select Device button allows the user to select a device from the supported device list. You can speed type the letters of the device you want and it will scroll through to find it for you. Clicking select will set all six fields quickly.

The Edit Device button allows the user to specify a signature and page sizes for a device that is new or not on the list. The page sizes are important and these values are used with the appropriate programming commands to program flash and EEPROM properly. If they are incorrect, the device will not be programmed correctly and will probably fail verification. It is worth noting that the flash page size given in most Atmel datasheets is in words and all fields in this application are given in bytes, so if the datasheet says 256 words, then enter 512 bytes.

Off to the right are more match fields for the fuses and lockbits. Some devices such as the ATTINY or ATMEGA series only use the top three fuse fields, but ATXMEGA devices use more of them. The final field is the lockbits field. Enabling these restricts which devices will match further than the device signature alone. For example, checking Match Fuse 0 / Low and setting a value of E2 means that only a device with the matching device signature that **also** matches a low fuse of E2 will use this microcontroller entry.

The second most important field here is Program Firmware value. It points to the firmware entry to program if this microcontroller entry is matched.

It is easier to understand the complexity of this section by looking through the examples in at the end of this manual. It can be as simple as a single microcontroller entry matching a device pointing to a single firmware to update to as complicated as multiple microcontroller entries matching by device and fuse settings flashing different versions of firmware including conditions where the user will be prompted during production programming, but not for subsequent programming.

A feature called Multiple Match Selection allows the user to select a specific match when more than one match occurs (all except V2). There is more information about this in the Board Operation and Example sections.

Poll method shows the method used to find out when an operation has been completed. For most AVR's it should be command, but for some older ones that do not support the command, it should be delay.

Firmware Entries Section

The screenshot shows the 'Firmware Entries' window. On the left, a list contains 'prog0'. Below the list are 'Add' and 'Del' buttons, and a checked 'Auto Select' checkbox. The main area is divided into two sections: Flash and EEPROM. Each section shows '0 bytes', 'None' compression, and '0x0000' CRC. The Flash section has buttons for 'Load HEX', 'Load BIN', 'Clear', and 'Load ELF'. The EEPROM section has buttons for 'Load HEX', 'Load BIN', 'Clear', and 'Fill'. Below these are settings for 'Program Frequency' (125 kHz), 'Program Method' (Program and Verify), 'Flash Count' (0), 'Limit Flashes' (1), and 'Set Serial No.' (0). On the right, there are checkboxes for 'Program Fuse 0 / Low', 'Program Fuse 1 / High', 'Program Fuse 2 / Extended', 'Program Fuse 3', 'Program Fuse 4', 'Program Fuse 5', 'Program Fuse 6', and 'Program Lockbits', each with a corresponding '00' value field.

These entries contain what needs to be programmed to a device. Each one includes the flash image, EEPROM image, fuses, lockbits, and programming settings.

Program frequency sets the ISP or PDI frequency for the flash and EEPROM programming and verification.

Program Method can be either Program Only or Program and Verify.

Limit Flashes allows you to limit the number of times a firmware entry can be programmed. Flash Count will show the number of times it has been flashed (for unlocked configurations that were retrieved from a board).

Set Serial No. works with the autoincrementing serial number assignment feature.

The same thing applies as did in the microcontroller entries section, the Add and Del buttons will add and delete entries and they will always be renumbered. The Program Firmware setting in each microcontroller is automatically adjusted if renumbering occurs.

The top section is for flash and it will show the number of bytes, and whether or not the flash image is compressed or not and if so how much compression occurred.

You can use the buttons to clear, or load an image from a HEX file, a BIN file, or an ELF file. Loading from an ELF file will not only load the flash image, but it will also load the EEPROM image, fuses, and lockbits if they are present in the ELF file.

The EEPROM section and works shows the same information for EEPROM.

It has an additional button for Fill. This allows you to fill the EEPROM area with a specific value. If you have the EESAVE fuse programmed, but you need to deploy an update that clears the EEPROM, you can use this to force the contents of the EEPROM to be 0xFF again.

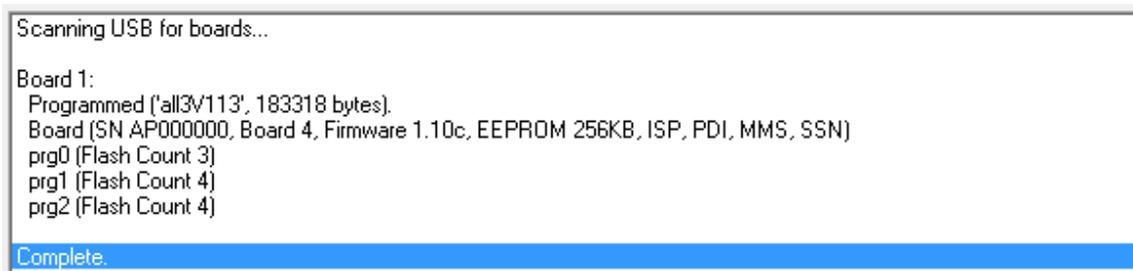
The right area allows you to specify the fuses and lockbits to program. Make sure you are programming the correct ones so you do not brick a device by accident.

Board Section



One thing to note about the buttons in the board section is that they apply to **all** connected boards. If you click Quick Erase and have 3 boards attached, they will all be quick erased.

Here is an example of the Identify button:



Again, if more than one board was plugged in, there would be a Board 2, and so on. They are enumerated in USB device order so there is no way to identify them from each other, but that is not an issue because all attached boards will be treated the same anyway.

The first line shows this board state is Programmed with an identification string of 'all3V113' that is 183318 bytes in size.

The second line shows the board features including serial number, board version, firmware version, EEPROM size, and supported features. In this case that includes the ISP and PDI protocols, the multiple match select (MMS), and set serial number (SSN).

Finally it shows programming counts. prg0 has been programmed 3 times, prg1=4 times, prg2=4 times. It will keep track of how many times each firmware has been programmed **successfully**.

Program and Test combines two buttons into one button.



Programming is actually writing the configuration to the board's EEPROM memory. The flash counts are cleared at the same time.

Testing is the device itself performing CRC and integrity tests on the configuration.

Quick Erase will erase a configuration from a board (just the first 256 bytes):

```
Board 1:  
Quick Erase successful.
```

```
Complete.
```

Secure Erase is more thorough and erases all EEPROM:

```
Board 1:  
Secure Erase (100%)..  
Secure Erase successful.
```

```
Complete.
```

5. Autoprogrammer Board Operation

Normally all you need to do is connect the autoprogrammer board to a device and give it power. It will start out with a blinking Green LED as it attempts to connect, read signature, fuse, and lockbits information, and evaluate its configuration to see if it can flash the device.

It will attempt at power on to connect over any enabled protocols for 5 seconds.

If anything goes wrong the Green LED will go out and the Red LED will blink out an error code (see section 3 above for a list of error codes).

The Green LED(s) will show the status as the flashing occurs. The V1, V2, and V4 versions have a single LED that will blink when busy and go solid when done. The V3 version has five segments so each segment represents 20%. If a segment is started it will blink, when it is completed it will go solid.

It is **important to understand** that flashing is complete when the Green LED(s) are solid. For V1, V2, and V4 this means the one Green LED is solid, for V3 this means all 5 segments are solid. Only that is the case you can be sure that everything is done. If you have the program and verify option set for the firmware entry being programmed, you can be sure that it verified correctly as well.

A feature called Multiple Match Selection (all except V2) will change the automatic behavior if the user creates a configuration where more than one microcontroller entry match is possible. For example, if uc0 and uc1 both have the same device signature selected and no other match restrictions and the board encounters this device signature, it has more than one microcontroller entry and will need to prompt for which one. This allows the user to select which of up to 5 microcontroller entries is used.

When this feature is activated, it will blink out the current number of matches that are available to choose from. Tapping the button will rotate through them. Holding the button for 2 seconds will flash the currently selected one. For example, it might blink twice. You now know that you have two microcontroller entry matches and must pick 1 or 2. Tapping the button once will roll the 2 over to 1. Again and it goes back to 2. Holding it for 2 seconds and it will flash the selected match.

It is important to note that the number **does not match** any particular microcontroller entry number. You are selecting between the first or second match in this example and they could be uc2 and uc5.

6. Programming Methodology

The autoprogrammer follows these steps when trying to program a device:

Test Configuration. It verifies that the main structure, microcontroller entries, and firmware entries have CRC integrity.

Identify. It tries to connect to a device at its slowest speed (62.5 kHz) using all enabled protocols. It will attempt to read the device signature, fuses, and lockbits.

Find matches. It goes through the microcontroller entries to find which ones match the device. If there is more than one match, it will release the device and prompt the user for which to install (all except V2, V2 will always select the first match). Once a firmware is selected, it will reconnect and verify that the device has not changed so it can proceed. It will also verify that the firmware entry has not reached its limit if one is set.

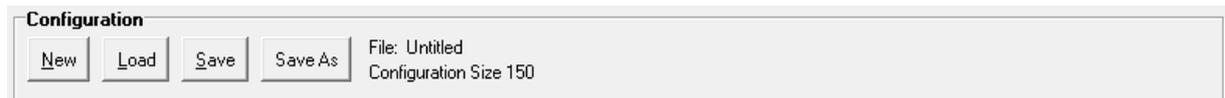
Program fuses. If the firmware is configured to program fuses and any fuses do not match, it will erase the device, and then program and verify any fuses that do not matching. It will then disconnect to make sure that the fuse changes will take effect. It does this at the slowest speed (62.5 kHz).

Program flash, EEPROM, and lockbits. It will change speeds to the program frequency and program flash and EEPROM, and if verify is enabled, verify flash and EEPROM, and finally program and finally verify lockbits.

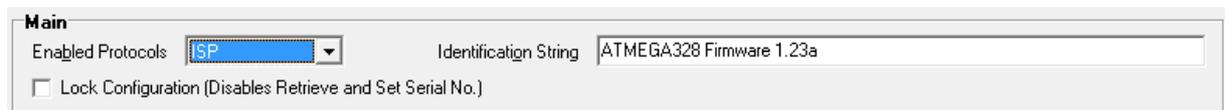
7. Simple Configuration Example

This is a basic simple how can I get it to program configuration walkthrough.

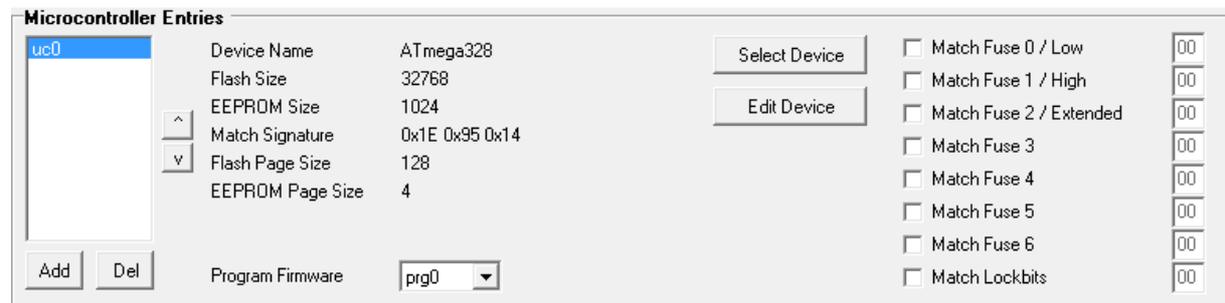
Start off by clicking the New Button. You should see the top area look like this:



Configure the Main section first. Set the enabled protocols to the ones you will need. I am going to support an ATMEGA328, so I set it to ISP. I filled in an identification string so if I identify a board with configuration later, I'll recognize what I've put onto it.



In the Microcontroller Entries section I just need one. Click Select Device and choose ATMEGA328. It defaults to prg0 for the firmware to flash which is fine because that is the section I will do next.



In the Firmware Entries section I am going to take the easy route and just load my ELF file for this project. It has the correct fuse and lockbits settings already so with one click and picking the file, it updates this section to the following. My project does not have any EEPROM flash data so it shows as empty.

Firmware Entries

Flash	26272 bytes (compressed 25.8% to 19487 bytes)	<input checked="" type="checkbox"/> Program Fuse 0 / Low	D0		
<input type="button" value="Load HEX"/>	<input type="button" value="Load BIN"/>	<input checked="" type="checkbox"/> Program Fuse 1 / High	D1		
<input type="button" value="Load ELF"/>	<input type="button" value="Clear"/>	<input checked="" type="checkbox"/> Program Fuse 2 / Extended	FC		
EEPROM	1024 bytes (compressed 97.6% to 25 bytes)	<input type="checkbox"/> Program Fuse 3	00		
<input type="button" value="Load HEX"/>	<input type="button" value="Load BIN"/>	<input type="button" value="Fill"/>	<input type="button" value="Clear"/>	<input type="checkbox"/> Program Fuse 4	00
Program Frequency	2 MHz	<input type="checkbox"/> Program Fuse 5	00		
Program Method	Program and Verify	<input type="checkbox"/> Program Fuse 6	00		
<input type="checkbox"/> Limit Flashes	1	<input checked="" type="checkbox"/> Program Lockbits	FC		
<input type="checkbox"/> Do Not Attempt Compression When Loading Images					

This device has a clock speed of 16 MHz, so I know it can handle being programmed at 2 MHz.

That is a complete configuration!

Click save in the top Configuration Section and give it a filename if you ever want to load it from disk.

Click Program/Verify/Test in the bottom Board Section when you want to program it to a board.

Unplug the board from USB and take it to the project to test it out!

8. Multiple Device Variant Example

This example adds to the previous example, so load the configuration from the previous example or start new and recreate it.

One issue that can be a pain to deal with is that you may have ATMEGA328 and ATMEGA328P devices that you want to program. Atmel treats each independently and each has its own independent device signature.

The autoprogrammer makes dealing with this very easy because all you have to do is create a second Microcontroller Entry for the ATMEGA328P and point it to the same Firmware Entry that the ATMEGA328 was using. First click the Add button in the Microcontroller Entry section to create a uc1. It will default to an (Unknown) device and the EXCEEDS SIZE OR UNKNOWN warning will appear in red. Click the Select Device button and select the ATMEGA328P. The warning will go away and you should see this:

Microcontroller Entries	
uc0	Device Name: ATmega328P
uc1	Flash Size: 32768
	EEPROM Size: 1024
	Match Signature: 0x1E 0x95 0x0F
	Flash Page Size: 128
	EEPROM Page Size: 4
	Program Firmware: prg0

Buttons: Add, Del, Select Device, Edit Device

Match options (checkboxes):

- Match Fuse 0 / Low: 00
- Match Fuse 1 / High: 00
- Match Fuse 2 / Extended: 00
- Match Fuse 3: 00
- Match Fuse 4: 00
- Match Fuse 5: 00
- Match Fuse 6: 00
- Match Lockbits: 00

Now it will match an ATMEGA328 on uc0 or an ATMEGA328P on uc1 and both will point it to the Firmware Entry prg0. The configuration size only increased slightly for the extra Microcontroller Entry because the Firmware Entry at prg0 was reused.

9. Dual Configuration Example

Here is an example of two different projects being updated, one that has an ATMEGA328 using ISP and one that has an ATXMEGA256A3BU using PDI. Both protocols are enabled in the main section:

Main
 Enabled Protocols: ISP and PDI
 Identification String:

Here is a screen shot of the ATMEGA328 configuration:

Microcontroller Entries

uc0	Device Name	ATmega328	<input type="button" value="Select Device"/>	<input type="checkbox"/> Match Fuse 0 / Low	<input type="text" value="00"/>
uc1	Flash Size	32768	<input type="button" value="Edit Device"/>	<input type="checkbox"/> Match Fuse 1 / High	<input type="text" value="00"/>
	EEPROM Size	1024		<input type="checkbox"/> Match Fuse 2 / Extended	<input type="text" value="00"/>
	Match Signature	0x1E 0x95 0x14		<input type="checkbox"/> Match Fuse 3	<input type="text" value="00"/>
	Flash Page Size	128		<input type="checkbox"/> Match Fuse 4	<input type="text" value="00"/>
	EEPROM Page Size	4		<input type="checkbox"/> Match Fuse 5	<input type="text" value="00"/>
	Program Firmware	prg0		<input type="checkbox"/> Match Fuse 6	<input type="text" value="00"/>
				<input type="checkbox"/> Match Lockbits	<input type="text" value="00"/>

Firmware Entries

prg0	Flash	26272 bytes (compressed 25.8% to 19485 bytes)	<input checked="" type="checkbox"/> Program Fuse 0 / Low	<input type="text" value="D0"/>
prg1	<input type="button" value="Load HEX"/> <input type="button" value="Load BIN"/> <input type="button" value="Load ELF"/> <input type="button" value="Clear"/>		<input checked="" type="checkbox"/> Program Fuse 1 / High	<input type="text" value="D1"/>
	EEPROM (empty)		<input checked="" type="checkbox"/> Program Fuse 2 / Extended	<input type="text" value="FC"/>
	<input type="button" value="Load HEX"/> <input type="button" value="Load BIN"/> <input type="button" value="Fill"/> <input type="button" value="Clear"/>		<input type="checkbox"/> Program Fuse 3	<input type="text" value="00"/>
	Program Frequency	1 MHz	<input type="checkbox"/> Program Fuse 4	<input type="text" value="00"/>
	Program Method	Program and Verify	<input type="checkbox"/> Program Fuse 5	<input type="text" value="00"/>
	<input type="checkbox"/> Limit Flashes	1	<input type="checkbox"/> Program Fuse 6	<input type="text" value="00"/>
	<input type="checkbox"/> Do Not Attempt Compression When Loading Images		<input checked="" type="checkbox"/> Program Lockbits	<input type="text" value="FC"/>

Here is a screen shot of the ATXMEGA256A3BU configuration:

The screenshot displays the configuration interface for the ATXMEGA256A3BU microcontroller. It is divided into two main sections: **Microcontroller Entries** and **Firmware Entries**.

Microcontroller Entries: This section contains a list of microcontroller entries on the left, with 'uc0' and 'uc1' visible. 'uc1' is currently selected. To the right of the list are fields for device properties: Device Name (ATxmega256A3B/ATxmega256A3BU), Flash Size (270336), EEPROM Size (4096), Match Signature (0x1E 0x98 0x43), Flash Page Size (512), and EEPROM Page Size (32). There are 'Select Device' and 'Edit Device' buttons. Below the list are 'Add' and 'Del' buttons. The 'Program Firmware' dropdown is set to 'prg1'. On the right side, there are checkboxes for 'Match Fuse 0 / Low', 'Match Fuse 1 / High', 'Match Fuse 2 / Extended', 'Match Fuse 3', 'Match Fuse 4', 'Match Fuse 5', 'Match Fuse 6', and 'Match Lockbits', each with a corresponding '00' value field.

Firmware Entries: This section contains a list of firmware entries on the left, with 'prg0' and 'prg1' visible. 'prg1' is currently selected. To the right of the list are fields for firmware properties: Flash (1810 bytes, compressed 22.2% to 1409 bytes), EEPROM (empty), Program Frequency (2 MHz), Program Method (Program and Verify), Limit Flashes (1), and Do Not Attempt Compression When Loading Images (unchecked). There are 'Load HEX', 'Load BIN', 'Load ELF', and 'Clear' buttons for both Flash and EEPROM. On the right side, there are checkboxes for 'Program Fuse 0 / Low', 'Program Fuse 1 / High', 'Program Fuse 2 / Extended', 'Program Fuse 3', 'Program Fuse 4', 'Program Fuse 5', 'Program Fuse 6', and 'Program Lockbits', each with a corresponding '00' value field.

Note that uc0 points to prg0 and uc1 points to prg1.

As both protocols are enabled, at startup the autoprogrammer will first try to connect over ISP for 50 milliseconds and if that fails, then try PDI for 50 milliseconds. It will round robin trying both protocols for up to 5 seconds before failing if a device is not connected.

10. Multiple Match Select Example

This configuration requires a board with a button (all except V2). A project of mine has two versions of firmware. A basic version and then a “more” version that has more features. I flash either version to the same device, an ATTINY1634. I load the basic version into Firmware Entry prg0 and the “more” version into Firmware Entry prg1. Then I have to create a two Microcontroller Entries because I can only point one entry to one firmware. By putting the same information in each Microcontroller Entry, when the autoprogrammer obtains the device signature, it will match both entries and this will enable the Multiple Match Select feature. Then I can choose which one I want, the first match (uc0) with basic firmware, or the second match (uc1) with “more” firmware. Here is a screen shot of both Microcontroller Entries and both Firmware Entries:

Microcontroller Entries				
uc0	Device Name ATtiny1634	Select Device	<input type="checkbox"/> Match Fuse 0 / Low	00
uc1	Flash Size 16384	Edit Device	<input type="checkbox"/> Match Fuse 1 / High	00
	EEPROM Size 256		<input type="checkbox"/> Match Fuse 2 / Extended	00
	Match Signature 0x1E 0x94 0x12		<input type="checkbox"/> Match Fuse 3	00
	Flash Page Size 32		<input type="checkbox"/> Match Fuse 4	00
	EEPROM Page Size 4		<input type="checkbox"/> Match Fuse 5	00
			<input type="checkbox"/> Match Fuse 6	00
			<input type="checkbox"/> Match Lockbits	00
Add Del	Program Firmware	prg0		

Microcontroller Entries				
uc0	Device Name ATtiny1634	Select Device	<input type="checkbox"/> Match Fuse 0 / Low	00
uc1	Flash Size 16384	Edit Device	<input type="checkbox"/> Match Fuse 1 / High	00
	EEPROM Size 256		<input type="checkbox"/> Match Fuse 2 / Extended	00
	Match Signature 0x1E 0x94 0x12		<input type="checkbox"/> Match Fuse 3	00
	Flash Page Size 32		<input type="checkbox"/> Match Fuse 4	00
	EEPROM Page Size 4		<input type="checkbox"/> Match Fuse 5	00
			<input type="checkbox"/> Match Fuse 6	00
			<input type="checkbox"/> Match Lockbits	00
Add Del	Program Firmware	prg1		

Firmware Entries

prg0	Flash	16384 bytes (compressed 57.1% to 7028 bytes)	<input checked="" type="checkbox"/> Program Fuse 0 / Low	E2
prg1			<input checked="" type="checkbox"/> Program Fuse 1 / High	D4
			<input checked="" type="checkbox"/> Program Fuse 2 / Extended	F5
			<input type="checkbox"/> Program Fuse 3	00
			<input type="checkbox"/> Program Fuse 4	00
			<input type="checkbox"/> Program Fuse 5	00
			<input type="checkbox"/> Program Fuse 6	00
			<input checked="" type="checkbox"/> Program Lockbits	FC

Flash 16384 bytes (compressed 57.1% to 7028 bytes)
 Load HEX Load BIN Load ELF Clear

EEPROM (empty)
 Load HEX Load BIN Fill Clear

Program Frequency 125 kHz
 Program Method Program and Verify
 Limit Flashes 1
 Do Not Attempt Compression When Loading Images

Add Del

Firmware Entries

prg0	Flash	16384 bytes (compressed 51.0% to 8023 bytes)	<input checked="" type="checkbox"/> Program Fuse 0 / Low	E2
prg1			<input checked="" type="checkbox"/> Program Fuse 1 / High	D4
			<input checked="" type="checkbox"/> Program Fuse 2 / Extended	F5
			<input type="checkbox"/> Program Fuse 3	00
			<input type="checkbox"/> Program Fuse 4	00
			<input type="checkbox"/> Program Fuse 5	00
			<input type="checkbox"/> Program Fuse 6	00
			<input checked="" type="checkbox"/> Program Lockbits	FC

Flash 16384 bytes (compressed 51.0% to 8023 bytes)
 Load HEX Load BIN Load ELF Clear

EEPROM (empty)
 Load HEX Load BIN Fill Clear

Program Frequency 125 kHz
 Program Method Program and Verify
 Limit Flashes 1
 Do Not Attempt Compression When Loading Images

Add Del

Note that both show 16384 bytes used. This was from a time when I signed a CRC in program code at the very end on the entire flash size and an unfortunate byproduct of that was a flash size being the full size of the device flash. I use a newer method now that signs the CRC in the flash image itself. The compression percentages are different because the flash images are different, again prg0 is basic firmware and prg1 is the “more” firmware.

When I try to flash a device, it goes into Multiple Match Select mode and I can easily select which one I want.

11. Multiple Match Advanced Example

This is the most advanced example! This configuration requires a board with a button (all except V2).

I have a friend who has a project with 3 microcontrollers. Two of the three are the same device and the third one is different. All three devices need their own specific firmware. That is no problem because we can use the Multiple Match Select feature to choose which one. That is fine at production where we know what we are doing, but not fine if we need to send a device to someone to upgrade our product and they don't know what they are doing. Here is a configuration that allows us to select at production, but after that becomes a fully automatic with no prompting update device for someone in the field. Obviously the person in the field will need to plug the device into each of the programming headers, but they won't have to do anything past that.

The first part is easy, let's just knock out the single device on its own. We'll use uc0 and prg0 for it. I don't recall exactly what it is, but I'm going to go with an ATXMEGA64A3. Here is its configuration.

The screenshot displays the AVR Studio configuration interface, divided into two main sections: Microcontroller Entries and Firmware Entries.

Microcontroller Entries: This section shows a list of microcontroller entries. The entry 'uc0' is selected. The configuration for 'uc0' is as follows:

Device Name	ATxmega64A3/ATxmega64A3U	Select Device	<input type="checkbox"/> Match Fuse 0 / Low	00
Flash Size	69632	Edit Device	<input type="checkbox"/> Match Fuse 1 / High	00
EEPROM Size	2048		<input type="checkbox"/> Match Fuse 2 / Extended	00
Match Signature	0x1E 0x96 0x42		<input type="checkbox"/> Match Fuse 3	00
Flash Page Size	256		<input type="checkbox"/> Match Fuse 4	00
EEPROM Page Size	32		<input type="checkbox"/> Match Fuse 5	00
			<input type="checkbox"/> Match Fuse 6	00
			<input type="checkbox"/> Match Lockbits	00

Additional controls include 'Add', 'Del', and 'Program Firmware' (set to 'prg0').

Firmware Entries: This section shows a list of firmware entries. The entry 'prg0' is selected. The configuration for 'prg0' is as follows:

Flash	34548 bytes (compressed 71.7% to 9790 bytes)	<input type="checkbox"/> Program Fuse 0 / Low	00		
Load HEX	Load BIN	Load ELF	Clear	<input type="checkbox"/> Program Fuse 1 / High	00
EEPROM	(empty)	<input type="checkbox"/> Program Fuse 2 / Extended	00	<input type="checkbox"/> Program Fuse 3	00
Load HEX	Load BIN	Fill	Clear	<input type="checkbox"/> Program Fuse 4	00
Program Frequency	125 kHz	<input type="checkbox"/> Program Fuse 5	00	<input type="checkbox"/> Program Fuse 6	00
Program Method	Program and Verify	<input type="checkbox"/> Program Lockbits	00		
<input type="checkbox"/> Limit Flashes	1				
<input type="checkbox"/> Do Not Attempt Compression When Loading Images					

Additional controls include 'Add', 'Del', and 'Program Fuse' options.

For the other two devices, a pair of ATXMEGA384C3's, we will need 4 Microcontroller Entries and 2 Firmware Entries to pull off selecting it the first time, fully automatic the second time.

Some thought needs to be put into how can we identify these two from each other after they are programmed? As they both have the same device signature, what we have left to work with is the fuses and lockbits. Assuming we want both locked with the same lockbits such as 0xFC that leaves us with the fuses.

This method uses the FUSEBYTE0 fuse on XMEGA which is the JTAG USER ID. An unprogrammed device will have this value unprogrammed (0xFF) and we can change it to 0x01 for our first device and 0x02 for our second device.

The first two Microcontroller Entries both are set for ATXMEGA384C3 and Match Fuse 0 as 0xFF. Because they are the same Multiple Match Select will see both matches and prompt us for which one we want. This will occur at production. Once prg1 or prg2 are programmed, each will change the Fuse 0 to a new value, 0x01 for the first one and 0x02 for the second one.

The second two Microcontroller Entries are set for ATXMEGA384C3 and Match Fuse 0 as 0x01 for the first and Match Fuse 0 as 0x02 for the second. Now each device will be recognized as one of these entries, there will be no multiple matching, and it will flash the correct firmware.

Here are the screenshots for this configuration:

Microcontroller Entries

uc0	Device Name	ATxmega384C3	Select Device	<input checked="" type="checkbox"/> Match Fuse 0 / Low	FF
uc1	Flash Size	401408	Edit Device	<input type="checkbox"/> Match Fuse 1 / High	00
uc2	EEPROM Size	4096		<input type="checkbox"/> Match Fuse 2 / Extended	00
uc3	Match Signature	0x1E 0x98 0x45		<input type="checkbox"/> Match Fuse 3	00
uc4	Flash Page Size	512		<input type="checkbox"/> Match Fuse 4	00
	EEPROM Page Size	32		<input type="checkbox"/> Match Fuse 5	00
	Program Firmware	prg1		<input type="checkbox"/> Match Fuse 6	00
				<input type="checkbox"/> Match Lockbits	00

Microcontroller Entries

uc0	Device Name	ATxmega384C3	Select Device	<input checked="" type="checkbox"/> Match Fuse 0 / Low	FF
uc1	Flash Size	401408	Edit Device	<input type="checkbox"/> Match Fuse 1 / High	00
uc2	EEPROM Size	4096		<input type="checkbox"/> Match Fuse 2 / Extended	00
uc3	Match Signature	0x1E 0x98 0x45		<input type="checkbox"/> Match Fuse 3	00
uc4	Flash Page Size	512		<input type="checkbox"/> Match Fuse 4	00
	EEPROM Page Size	32		<input type="checkbox"/> Match Fuse 5	00
	Program Firmware	prg2		<input type="checkbox"/> Match Fuse 6	00
				<input type="checkbox"/> Match Lockbits	00

Microcontroller Entries

uc0	Device Name	ATxmega384C3	Select Device	<input checked="" type="checkbox"/> Match Fuse 0 / Low	01
uc1	Flash Size	401408	Edit Device	<input type="checkbox"/> Match Fuse 1 / High	00
uc2	EEPROM Size	4096		<input type="checkbox"/> Match Fuse 2 / Extended	00
uc3	Match Signature	0x1E 0x98 0x45		<input type="checkbox"/> Match Fuse 3	00
uc4	Flash Page Size	512		<input type="checkbox"/> Match Fuse 4	00
	EEPROM Page Size	32		<input type="checkbox"/> Match Fuse 5	00
	Program Firmware	prg1		<input type="checkbox"/> Match Fuse 6	00
				<input type="checkbox"/> Match Lockbits	00

Microcontroller Entries

uc0	Device Name	ATxmega384C3	Select Device	<input checked="" type="checkbox"/> Match Fuse 0 / Low	02
uc1	Flash Size	401408	Edit Device	<input type="checkbox"/> Match Fuse 1 / High	00
uc2	EEPROM Size	4096		<input type="checkbox"/> Match Fuse 2 / Extended	00
uc3	Match Signature	0x1E 0x98 0x45	<input type="checkbox"/> Match Fuse 3	00	
uc4	Flash Page Size	512	<input type="checkbox"/> Match Fuse 4	00	
	EEPROM Page Size	32	<input type="checkbox"/> Match Fuse 5	00	
	Program Firmware	prg2	<input type="checkbox"/> Match Fuse 6	00	
			<input type="checkbox"/> Match Lockbits	00	

Firmware Entries

prg0	Flash	102784 bytes (compressed 23.4% to 78686 bytes)	<input checked="" type="checkbox"/> Program Fuse 0 / Low	01		
prg1	Load HEX	Load BIN	Load ELF	Clear	<input checked="" type="checkbox"/> Program Fuse 1 / High	FF
prg2	EEPROM	(empty)	<input type="checkbox"/> Program Fuse 2 / Extended	00	<input checked="" type="checkbox"/> Program Fuse 3	EE
	Load HEX	Load BIN	Fill	Clear	<input checked="" type="checkbox"/> Program Fuse 4	DD
	Program Frequency	2 MHz	<input type="checkbox"/> Program Fuse 5	00	<input type="checkbox"/> Program Fuse 6	00
	Program Method	Program and Verify	<input checked="" type="checkbox"/> Program Lockbits	FC		
	<input type="checkbox"/> Limit Flashes	1				
	<input type="checkbox"/> Do Not Attempt Compression When Loading Images					

Firmware Entries

prg0	Flash	25088 bytes (not compressed)	<input checked="" type="checkbox"/> Program Fuse 0 / Low	02		
prg1	Load HEX	Load BIN	Load ELF	Clear	<input checked="" type="checkbox"/> Program Fuse 1 / High	FF
prg2	EEPROM	(empty)	<input type="checkbox"/> Program Fuse 2 / Extended	00	<input checked="" type="checkbox"/> Program Fuse 3	EE
	Load HEX	Load BIN	Fill	Clear	<input checked="" type="checkbox"/> Program Fuse 4	DD
	Program Frequency	2 MHz	<input type="checkbox"/> Program Fuse 5	00	<input type="checkbox"/> Program Fuse 6	00
	Program Method	Program and Verify	<input checked="" type="checkbox"/> Program Lockbits	FC		
	<input type="checkbox"/> Limit Flashes	1				
	<input type="checkbox"/> Do Not Attempt Compression When Loading Images					

12. Version Compatibility

Almost always before version 1.11, the versions for the board firmware and pc application were kept in sync with each other. If you have firmware version 1.04 for example, you would need pc application 1.04 to go with it. Starting at version 1.11, the board firmware and PC Application will no longer be kept in sync. The deciding factor between whether a particular board and particular PC Application can talk to each other is called the protocol version. This table will show the relationships:

Protocol Version	Compatible Components
1.06	Firmware 1.11 PC Application 1.11 PC Application 1.10e PC Application 1.10d PC Application 1.10c Firmware 1.10c PC Application 1.10b Firmware 1.10b Firmware 1.10 PC Application 1.10
1.05	Firmware 1.06 PC Application 1.06
1.04	Firmware 1.05 Firmware 1.04 PC Application 1.04
1.03	Firmware 1.03 PC Application 1.03
1.02	Firmware 1.02 PC Application 1.02
1.01	Firmware 1.01 PC Application 1.01
1.00	Firmware 1.00 PC Application 1.00

13. Release Notes

Firmware 1.11 PC Application 1.12 (4/4/2019):

The ATmega324PB and ATmega328PB have been added to the device list.

Firmware 1.11 PC Application 1.11 (1/9/2018):

The application has been split into two applications/installs: autoprogammer.exe which is the full feature variant, and autoprogammerm.exe which is the minimal variant. There is an independent setup application for each. The main application no longer supports the -simplemode option.

A new merge feature has been added that allows merging hex files together. You can begin by loading a HEX or ELF file and then use Merge HEX to merge in a bootloader for example.

USB device scanning has been improved. The previous behavior was to exit if a USB device was present that it could not get attributes for. The result was that the software could not see a board even though one was present. The new code will skip any boards it can't get attributes for and still return the ones it can.

The autoprogammer was identifying itself as a PID Controller the operating system in error. This has been corrected and it will now properly identify itself as an Autoprogammer.

When using the MMS feature, the startup blinks and the first blink to indicate the number of MMS matches there are was running together. A delay has been added to make it more clear.

Firmware 1.10c PC Application 1.10e (5/31/2017):

The PC Application has been enhanced to have a simple mode. To use it simply run it with a -simplemode option specified.

Some of the other command line options did not work in the first Unicode version of this tool; this has been fixed.

Firmware 1.10c PC Application 1.10d (1/11/2017):

The PC Application has been enhanced to ignore an Intel type 3 hex command which is not necessary, but was causing an error. This error no longer occurs.

Firmware 1.10c PC Application 1.10c (9/27/2016):

The PC Application has been enhanced to ignore an Intel type 5 hex command which is not necessary, but was causing an error. This error no longer occurs.

Firmware 1.10c PC Application 1.10b (6/14/2016):

Supports board versions 1, 2, 3, and 4. Board version 4 is a new version that has the following changes: A real oscillator instead of a crystal, single green LED for status, move to 1206 sized parts, top side button, polarity changed for a top mount male shrouded header or bottom right angle female, added terminals for the programming status (TPS=success, TPF=fail).

Pin 1 of the AVR now indicates successful programming status, while pin 41 now indicates failed programming status. Both will be low unless programming has succeeded or failed, in which case the appropriate one will go high.

Firmware 1.10b PC Application 1.10b (4/20/2016):

The PC Application ELF loader would only load the .data and .text sections before. It has been enhanced to now load other sections properly as well.

Firmware 1.10b PC Application 1.10 (2/2/2016):

Supports board versions 1, 2, and 3.

Pin 1 of the AVR now indicates the programming status. It will always be low unless programming is complete, in which case it will be high.

An issue was discovered on PDI/XMEGA devices that would cause a fuse verification failure (error 18), particularly when changing FUSE BYTE 4. This version corrects the issue by writing the fuse, disconnecting and reconnecting to the PDI, and then verifying it was written properly. (A workaround for users on firmware earlier than 1.10b is to simply program the device a second time. The first time will set the fuse and the second time it will not need to be set and will verify properly.)

Firmware 1.10 PC Application 1.10 (12/31/2015):

Supports board versions 1, 2, and 3.

Full support for CRCSNTOOL has been added. CRCSNTOOL allows a user to crc protect flash and assign a serial number in flash without recompiling. The programmer can embed the serial number when programming and automatically increment it. The

user can change the next serial number using the PC Application without reprogramming the board if the board is not locked.

The ability to retrieve a configuration from a board has been added. To prevent this, the user can check on the Lock option before programming. A locked board will not allow retrieval or the next serial no. for any firmware's to be changed.

Some older AVR devices would fail programming with ISP. This was caused because these models do not support the isp command that polls whether a flash operation is complete. A new poll method option allows the user to select between command polling (newer devices) and delay polling (older devices). The device table has been modified to automatically select the correct type of polling by device.

The flash and EEPROM CRC values are now visible in the PC application.

The multiple match select has been expanded from a maximum of 5 to 10.

EEPROM read caching performance has been improved.

Support for a version subletter has been added to reflect minor changes or fixes.

The PC application will report the board features based on the board version. ISP=ISP programming, PDI=PDI programming, MMS=Multiple match selection, and SSN=Set serial no.

The file extension (and file format) have changes from iap (ISP Auto Programmer) to apc (Auto Programmer Configuration). The best way to migrate any old iap files to the new format is to open them in the old PC application and recreate them manually in the new PC application.

Firmware 1.06 PC Application 1.06 (9/18/2015):

Supports board versions 1, 2, and 3.

Expanded firmware entries to 255, but only the first 30 allow limits and will keep a count of the number of times flashed.

Auto select feature will automatically select the firmware entry when a microcontroller entry is selected or the program firmware combo box is changed. You can disable this by unchecking the auto select combo box.

Secure erase will also verify that the data has been erased.

Device list changed to match the way Atmel Studio lists devices. PC Application will remember which specific device is selected even in cases where multiple devices share the same device signature.

Support added for ATmega168PB, ATmega2564RFR2, ATmega256RFR2, ATmega48PB, ATmega88PB, ATtiny441, ATtiny841.

The status of operations is now indicated in percent instead of bytes.

Identify and Verify operations are faster.

Test now also decompresses and validates the CRC of all flash and EEPROM images and has also been optimized for more performance.

A configuration file named "All AVR Erase and Fuse Defaults.apc" can be used to easily erase and return fuses to their factory settings for many AVR's.

Speed letters (for use with the ALT key) have been added to the PC Application.

Firmware 1.05 PC Application 1.04 (9/15/2015):

Supports board versions 1, 2, and 3.

Some PDI devices require more than 25ms to be ready so the timeout was changed to 250ms.

Version 1.04 (8/29/2015):

Supports board versions 1, 2, and 3.

The frequency selection has been moved from the main area to each firmware entry as program frequency. Device identification and fuse programming are always done at 62.5 kHz, but flash and EEPROM programming are done according to the program frequency setting. If a fuse change is made, the autoprogrammer will disconnect and reconnect to make sure they are effective before attempting the program frequency.

The program method also seemed more suited for the firmware entry area and was moved there as well.

The identification string has been expanded to 54 bytes.

Version 1.03 (8/28/2015):

Supports board versions 1, 2, and 3.

The multiple match selection feature was added to allow the user to select between multiple matches (board versions 1 and 3).

Added support for 5 segment led graph (board version 3).

The board version has been added to identify.

Firmware improved to support board versions more cleanly.

Version 1.02 (8/26/2015)

Supports board versions 1 and 2.

Bug in ISP code found and corrected.

PDI connection code has additional tests to verify connection is good.

Version 1.01 (8/24/2015):

Supports board versions 1 and 2.

The PDI protocol was added (board version 2).

Structure performance improved greatly.

Support for ATXMEGA devices added.

Compression is done only one time when loading a flash or EEPROM image.

The user interface was improved.

Version 1.00 (3/21/2015):

Supports board version 1.

This version is the initial release (ISP protocol only).