

# **Application Note: NAE-AN-001**

## **Loading or Reloading the LoonBoard Unified Bootloader LUB code into the LoonBoard**

## Loading the LoonBoard Unified Bootloader onto the Target

The LoonBoard should have come preloaded with the lub system, but for instance if you have replaced the AVR on the LoonBoard the files will no longer be loaded. It is very simple to load these files, provided nothing else goes wrong.

This example assumes you have avrdude installed, which if you do not is a prerequisite. If you are using Windows this is accomplished by installing WinAVR from

<http://winavr.sourceforge.net>,

and if you are using Linux you will have to compile and install avrdude as described in the source for the package.



Figure 1: Hookup for Programming

As well you need a working AVR programmer, the Atmel avr-isp is used here. Finally you need a communication cable that works with the LoonBoard – if you have the NewAE lb-cable then you are all set!

### Getting the AVR Files

You will need two files to program into the AVR. The first is the part of the bootloader that communicates with the computer. The second is the “tinyloader” that is always resident in the AVR's FLASH space.

On the NewAE website in the lubloader section you can download the two files at once. If you want to make these files then you will need to download the sources for the lub. In it you will find an AVR directory, within that is two directories “lub” and “tinyloader”. Run make with WinAVR installed on Windows or the avr-gcc toolchain installed in linux as such:

```
cd directory/to/lub/sources
cd AVR
cd lub
make
    ***LOTS OF OUTPUT, NO ERRORS HOPEFULLY***
cd ..
cd tinyloader
make
    ***LOTS OF OUTPUT, NO ERRORS HOPEFULLY***
cd ..
```

The two hex files are main.hex in the lub directory, and tinyloader.hex in the tinyloader directory.

### Setting up the Hardware

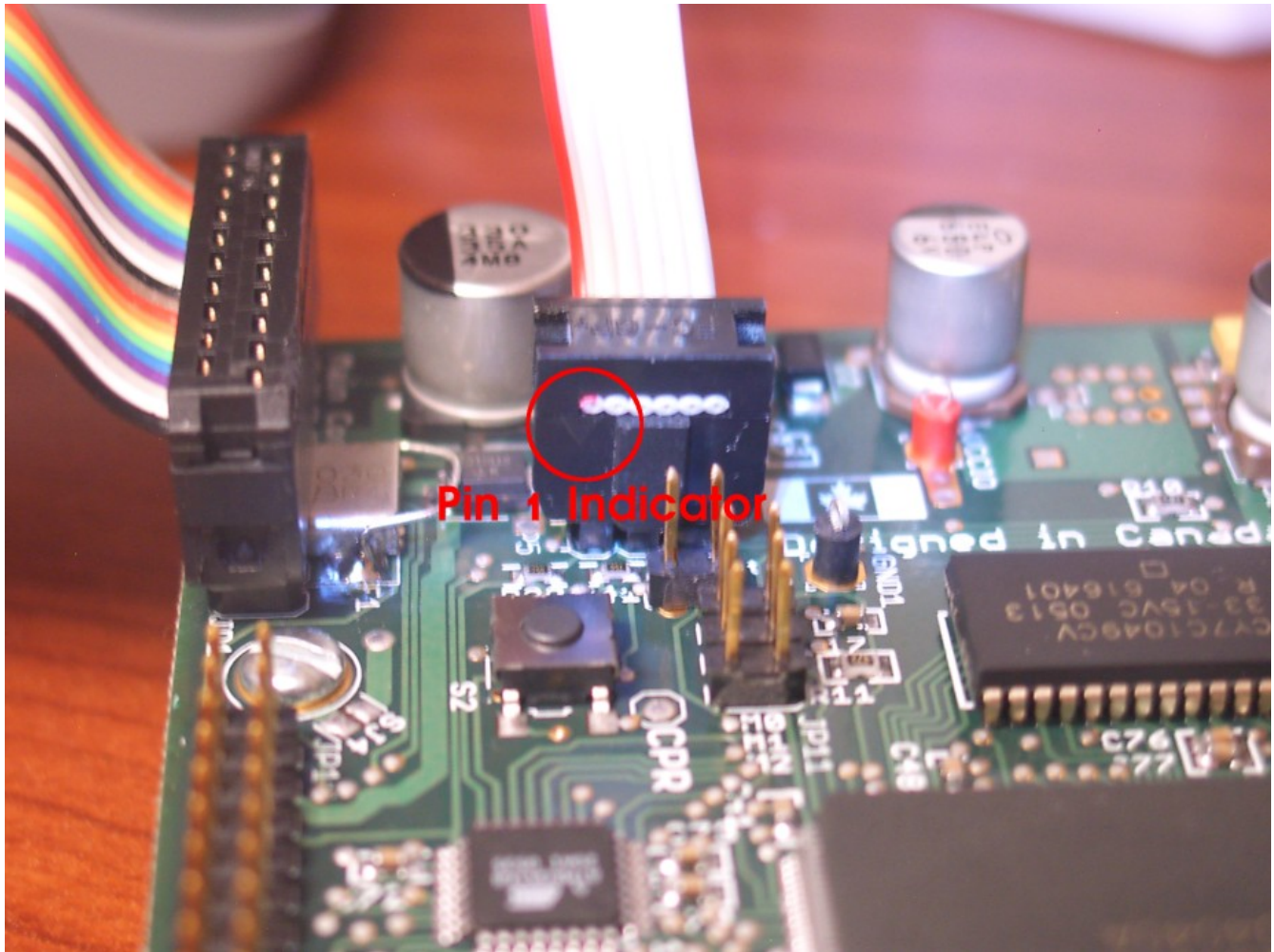


Figure 2: Pin 1 Indicator on Atmel AVRISP, a small triangle

Connect the programmer to the 6-pin connector on the LoonBoard. This connector has the same pinout as the AVR-ISP, and is marked with the pin numbers. Be sure that pin 1 on the programming cable lines up with pin 1 on the connector. Figure 2 shows this pin-1 indicator, a small triangle in the IDC connector.

You will also need power and a serial connection as mentioned. See the LoonBoard manual for more information on this pinout.

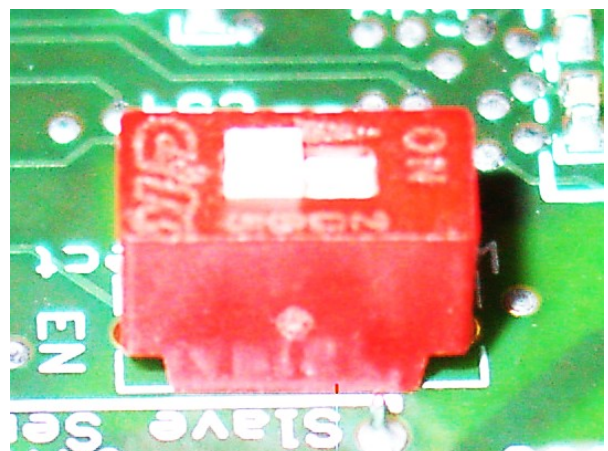


Figure 3: - Write protect is off

On the bottom of the LoonBoard find the “Write Protect” switch, and be sure it is in the “off” position. Base it on the markings directly on the switch as shown in figure 3.

## Programming

Now that you have the hardware set up, program the main.hex file into the AVR. As well make sure the “divide clock by 8” fuse is unprogrammed, by default it is programmed.

The avrdude command to run to accomplish all this is:

```
cd lub
avrdude -c avrispv2 -p m168 -P /dev/ttyS0 -U
flash:w:main.hex -U lfuse:w:0xd2:m -U
hfuse:w:0xdf:m -U efuse:w:0xf9:m
```

Where you will have to change avrispv2 to your programmer and /dev/ttyS0 to your serial port. The /dev/ttyS0 format is for Linux, if you are using Windows then this would just be com1 likely.

You now have to store the data in the permanent DataFLASH location. Unplug the AVR programmer from the LoonBoard, it may interfere with the DataFLASH lines. You will need to have the lubloader program installed as well. If you are not still in it change to the lub directory that should have the main.hex file. Run as follows:

```
lubloader -q main.hex -P /dev/ttyUSB0
```

You will need to reset the AVR on the LoonBoard by cycling power to it. You should see the file sent and it is verified OK. The serial port is again the /dev/ttyUSB0 and if you are using Windows this will be com2/com3/etc.

Send the “safemode” bootloader now:

```
lubloader -w main.hex -P /dev/ttyUSB0
```

Now you need to set the AVR to jump to the bootloader on start-up. To do this set the “bootloader size 256 words / 512 bytes” fuse and the “jump to bootloader at reset” fuse. You also need to program the tinyloader.hex file into the AVR device now.

```
cd ..
cd tinyloader
avrdude -c avrispv2 -p m168 -P /dev/ttyS0 -U
efuse:w:0xfc:m -U flash:w:tinyloader.hex
```

Now you should have a working bootloader! To test it run the command:

```
lubloader -P /dev/ttyUSB0 -v
```

And you should see some output that indicates it communicated OK and was able to reset the AVR. If you don't see the troubleshooting section.

## Troubleshooting

There is lots of potential problems. They are separated into area:

### Unable to program AVR

This problem is most likely an error with your programmer. Make sure it is connected the correct way and to the proper header. If you plug in an AVRISP backwards it will still get powered and look like it is working, but it won't be able to communicate with the target AVR.

If you are using a simple parallel port programmer try to find an AVRISP, the parallel port programmers can be unreliable.

If you have been programming away and then the AVR stops responding, you may have programmed the clock source fuse to use an external source which is not present. Get a logic level (0-3.3V) clock and apply it to the "CPR" pad on the LoonBoard above the pushbutton. You can solder a wire to this pad, it connects to the line that would need an external clock source on it (XTAL1).

If you have no other source, you can make an oscillator using an inverter chip such as a 74HC04 or similar. Note that it is being used below its rated voltage in most cases as this is the more common 5V device. If possible use a logic probe to make sure it is oscillating, as there is little guarantee it will work. Try soldering directly to the pins and don't use a breadboard, as the breadboard could add too much capacitance.

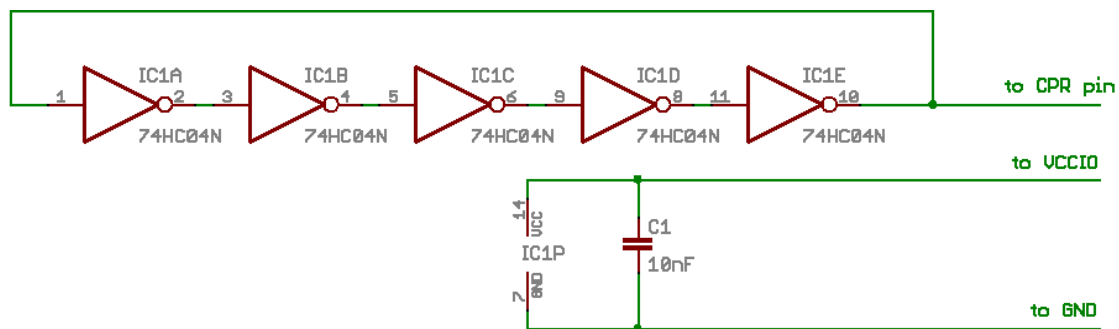


Figure 4: NOT gate Oscillator

Once you are able to communicate with the AVR set the fuses back to default, lfuse is 0xd2, hfuse is 0xdf, and efuse is 0xf9.

### Unable to communicate with LUB after programming main.hex

This is likely a problem with the serial cable. Open a terminal emulator and connect to the com port you are using. If this is a real serial cable and not a lb-cable, unplug the cable from the LoonBoard adapter you made and short pins 2 and 3. Typing on the terminal emulator should see an echo, removing the short should make the echo stop.

You have now confirmed that you know which serial port you are using and it works. Next short the TX and RX outputs from your RS232-logic level adapter. You should see an echo now, and the echo should stop when you remove the short. If the echo does not stop it means it was a “local echo”, and in fact you should be seeing two echo's when you type anything and TX is shorted to RX.

If you did not see this then there is a problem with your RS232 adapter, not the LoonBoard. The final test is to see if the signals made it all the way onto the LoonBoard. Compile the sample program shown in the listing below and program it into the AVR.

```
#include <avr/io.h>

int main(void)
{
    DDRD |= 1<<1;

    while(1==1)
    {
        PORTD = (PIND & 0x01) << 1;
    }

    return 1;
}
```

This forms a software loopback – you should again check for an echo that exists only when the LoonBoard is connected. If you see this then the serial link is fine.

When running lubloader append a number of -v's onto the end of it. So for example:

```
lubloader -P com3 -q main.hex -vvvvvvv
```

will provide lots of debugging output. Adding only -v provides the least, and the more v's the more output. So at the start you might want lot's of v's. However it will print each byte sent and received, so eventually this will become too much information as you progress.

If you are never able to get reliable communications it may be that the oscillator is unable to calibrate itself. This could be the case if the 32.768 KHz crystal is damaged, the TWI bus is loaded down, or the X1226 is damaged.

### Unable to communicate with LUB after programming Tinyloader

If you can communicate normally but not when the Tinyloader is setup the problem is likely a hardware one, assuming the program for the tinyloader is OK. The tinyloader needs the DataFLASH working properly, as it loads the constant from it. If you are using a parallel port programmer or other device that might not release the lines properly the DataFLASH could have corrupted data written to it.

If you are unable to get the device working contact [support@newae.com](mailto:support@newae.com). As well you can use the lubloader mailing list on the Sourceforge website, or the support forums on the NewAE website.

## ***Document Revision History***

November 8, 2005:

- Initial document release

**Disclaimer:** NewAE assumes no liability whatsoever and disclaims any express, implied, or statutory warranty related to its products including, but not limited to, the implied warranty of merchantability, fitness for a particular purpose, or non-infringement. In no event shall NewAE be liable for any direct, indirect, consequential, punitive, special or incidental damages (including, without limitation, damages for loss of profits, business, interruption, or loss of information) arising out of the use or inability to use this document, even if NewAE has been advised of the possibility of such damages. NewAE makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to the specifications and product descriptions at any time without notice. NewAE does not make any commitment to update the information contained herein. NewAE's products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

© NewAE 2005. All rights reserved.