

PACMAN Programming User Manual

Pack Management System

Last Updated 12/19/15

Engineer: John Gehrig

Overview

This document describes the methods and procures to build and upload code to the Rev 0.3 and later PACMAN boards.

Prerequisites

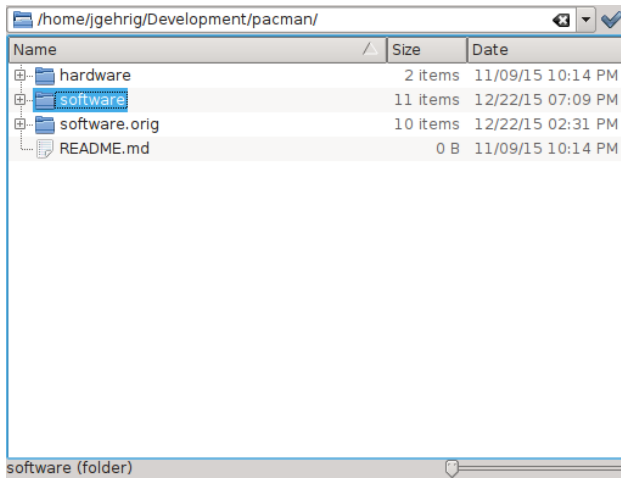
You will need the following programs installed on the development machine. The procedure is designed for a Linux environment, however Windows should also work with minimal modification.

- AVR-GCC (<http://www.nongnu.org/avr-libc/>)
(For details see http://avr-eclipse.sourceforge.net/wiki/index.php/The_AVR_GCC_Toolchain)
- GNU Make (<https://www.gnu.org/software/make/>)
- AVRDUDE (<http://www.nongnu.org/avrdude/>)
- Atmel ICE Basic Programmer

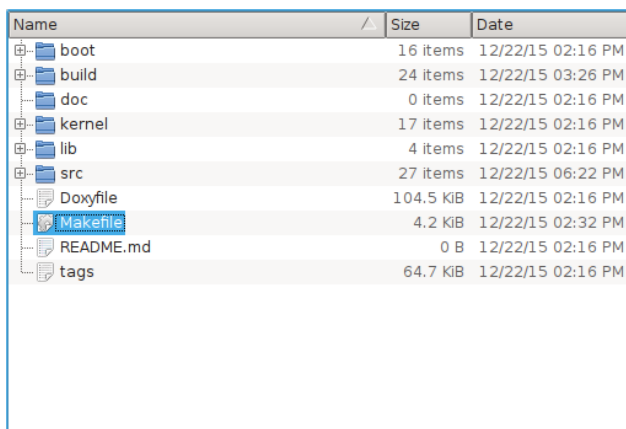


Software Build Procedure

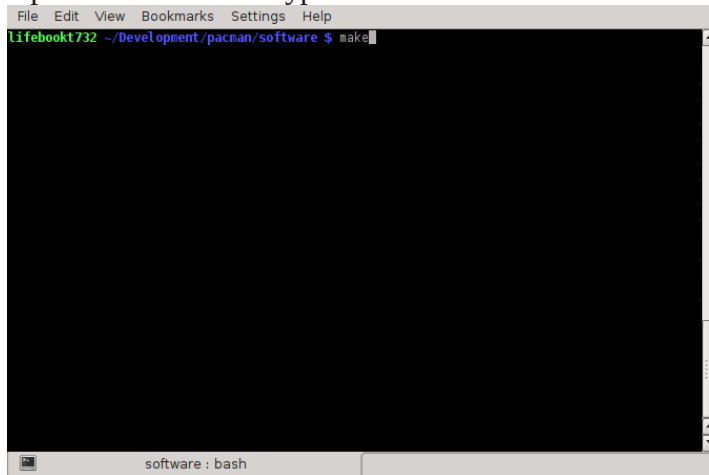
All source code is located in the 'software' folder of the PACMAN repository



Navigate to the folder containing 'Makefile'



Open a terminal and type make



Here is an example of “good” make output

```
avr-gcc -c -g -Os -mmcu=at90can32 -Wall -Werror -std=c99 -Isrc -Ikernel -Ilib src/main.c -o build/main.o
avr-gcc -c -g -Os -mmcu=at90can32 -Wall -Werror -std=c99 -Isrc -Ikernel -Ilib src/i2c.c -o build/i2c.o
avr-gcc -c -g -Os -mmcu=at90can32 -Wall -Werror -std=c99 -Isrc -Ikernel -Ilib src/task_gui.c -o build/task_gui.o
avr-gcc -c -g -Os -mmcu=at90can32 -Wall -Werror -std=c99 -Isrc -Ikernel -Ilib src/task_watchdog.c -o build/task_watchdog.o
avr-gcc -c -g -Os -mmcu=at90can32 -Wall -Werror -std=c99 -Isrc -Ikernel -Ilib src/task_control.c -o build/task_control.o
avr-gcc -c -g -Os -mmcu=at90can32 -Wall -Werror -std=c99 -Isrc -Ikernel -Ilib src/task_safety.c -o build/task_safety.o
avr-gcc -c -g -Os -mmcu=at90can32 -Wall -Werror -std=c99 -Isrc -Ikernel -Ilib src/task_config.c -o build/task_config.o
avr-gcc -c -g -Os -mmcu=at90can32 -Wall -Werror -std=c99 -Isrc -Ikernel -Ilib src/task_canbus.c -o build/task_canbus.o
avr-gcc -c -g -Os -mmcu=at90can32 -Wall -Werror -std=c99 -Isrc -Ikernel -Ilib src/tasklist.c -o build/tasklist.o
avr-gcc -c -g -Os -mmcu=at90can32 -Wall -Werror -std=c99 -x assembler-with-cpp -Isrc -Ikernel kernel/atomport-asm.s -o
build/atomport-asm.o
avr-gcc -c -g -Os -mmcu=at90can32 -Wall -Werror -std=c99 -I. -Isrc kernel/atomkernel.c -o build/atomkernel.o
avr-gcc -c -g -Os -mmcu=at90can32 -Wall -Werror -std=c99 -I. -Isrc kernel/atomsem.c -o build/atomsem.o
avr-gcc -c -g -Os -mmcu=at90can32 -Wall -Werror -std=c99 -I. -Isrc kernel/atommutex.c -o build/atommutex.o
avr-gcc -c -g -Os -mmcu=at90can32 -Wall -Werror -std=c99 -I. -Isrc kernel/atomtimer.c -o build/atomtimer.o
avr-gcc -c -g -Os -mmcu=at90can32 -Wall -Werror -std=c99 -I. -Isrc kernel/atomqueue.c -o build/atomqueue.o
avr-gcc -c -g -Os -mmcu=at90can32 -Wall -Werror -std=c99 -I. -Isrc kernel/atomport.c -o build/atomport.o
avr-gcc -c -g -Os -mmcu=at90can32 -Wall -Werror -std=c99 -Isrc -Ikernel -Ilib lib/uart/uart.c -o build/uart.o
avr-gcc -c -g -Os -mmcu=at90can32 -Wall -Werror -std=c99 -Isrc -Ikernel -Ilib lib/atmel/TWI_Master.c -o build/TWI_Master.o
avr-gcc -c -g -Os -mmcu=at90can32 -Wall -Werror -std=c99 -Isrc -Ikernel -Ilib lib/atmel/can_drv.c -o build/can_drv.o
avr-gcc -c -g -Os -mmcu=at90can32 -Wall -Werror -std=c99 -Isrc -Ikernel -Ilib lib/atmel/can_lib.c -o build/can_lib.o
avr-gcc -c -g -Os -mmcu=at90can32 -Wall -Werror -std=c99 -Isrc -Ikernel -Ilib lib/lcd/lcd.c -o build/lcd.o
#avr-gcc -g -Os -mmcu=at90can32 -Wall -Werror -std=c99 build/main.o build/main.o build/i2c.o build/task_gui.o build/task_watchdog.o
build/task_control.o build/task_safety.o build/task_config.o build/task_canbus.o build/tasklist.o build/atomport-asm.o
build/atomkernel.o build/atomsem.o build/atommutex.o build/atomtimer.o build/atomqueue.o build/atomport.o build/uart.o
build/TWI_Master.o build/can_drv.o build/can_lib.o build/lcd.o -o build/pacman.elf -Wl,-Map,build/pacman.map
echo build/main.o build/i2c.o build/task_gui.o build/task_watchdog.o build/task_control.o build/task_safety.o build/task_config.o
build/task_canbus.o build/tasklist.o build/atomport-asm.o build/atomkernel.o build/atomsem.o build/atommutex.o build/atomtimer.o
build/atomqueue.o build/atomport.o build/uart.o build/TWI_Master.o build/can_drv.o build/can_lib.o build/lcd.o
build/main.o build/i2c.o build/task_gui.o build/task_watchdog.o build/task_control.o build/task_safety.o build/task_config.o
build/task_canbus.o build/tasklist.o build/atomport-asm.o build/atomkernel.o build/atomsem.o build/atommutex.o build/atomtimer.o
build/atomqueue.o build/atomport.o build/uart.o build/TWI_Master.o build/can_drv.o build/can_lib.o build/lcd.o
avr-gcc -g -Os -mmcu=at90can32 -Wall -Werror -std=c99 build/main.o build/i2c.o build/task_gui.o build/task_watchdog.o
build/task_control.o build/task_safety.o build/task_config.o build/task_canbus.o build/tasklist.o build/atomport-asm.o
build/atomkernel.o build/atomsem.o build/atommutex.o build/atomtimer.o build/atomqueue.o build/atomport.o build/uart.o
build/TWI_Master.o build/can_drv.o build/can_lib.o build/lcd.o -o build/pacman.elf -Wl,-Map,build/pacman.map
Building pacman.hex
avr-objcopy -j .text -j .data -O ihex build/pacman.elf build/pacman.hex
avr-size -C --mcu=at90can32 build/pacman.elf
AVR Memory Usage
-----
Device: at90can32

Program:   11636 bytes (35.5% Full)
(.text + .data + .bootloader)

Data:      994 bytes (48.5% Full)
(.data + .bss + .noinit)
```

Make will output a file in the build directory based on the variable 'TARGET_NAME'. The default output is to build/pacman.hex. This file can be manually uploaded through avrdude, or automatically from make.

NOTE: Device resource usage is listed at the bottom of the make file.

Adding Files to the Project

SRC_DIR – Directory containing all source files.

LIB_DIR – Directory containing all library files.

BUILD_DIR – The directory to which program output should be directed.

APP_OBJECTS – A list of individual source files, must have a .o extension

LIB_OBJECTS – A list of library source files, must have a .o extension

To add a file to the project, you must first determine if it should be treated as a library, or as a source file. Library files should be added to the LIB_OBJECTS variable with the .c extension replaced with .o, folder path should be included. Source files should be added to the APP_OBJECTS variable, no folder path is required, but the extension should also be converted from .c to .o.

For example, to add the source file src/foo.c to the project, the following code would be added to the make file.

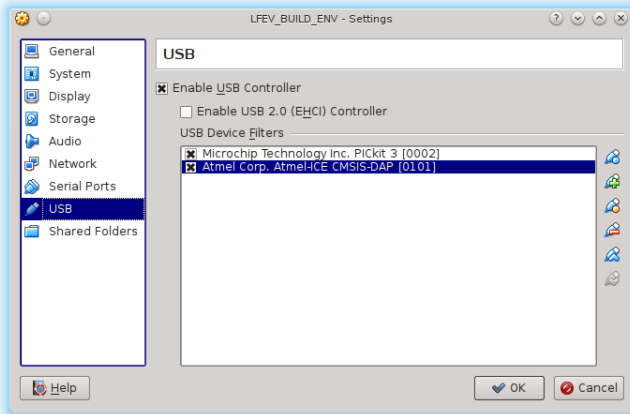
```
APP_OBJECTS = {all other source files} foo.o
```

A library file named baz/bar.c would be added as follows

```
LIB_OBJECTS = {all other source files} baz/bar.o
```

Programming the Target Device

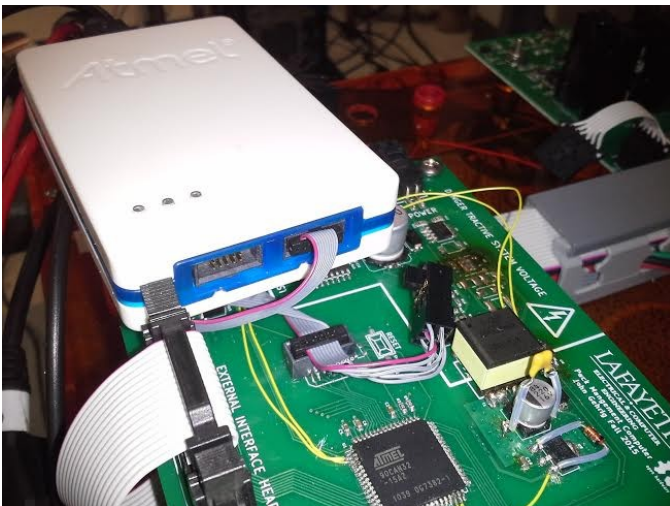
Ensure the Atmel ICE device is attached to your computer. If you are using a virtual machine, ensure a filter rule is present so the device can be recognized by the guest machine.



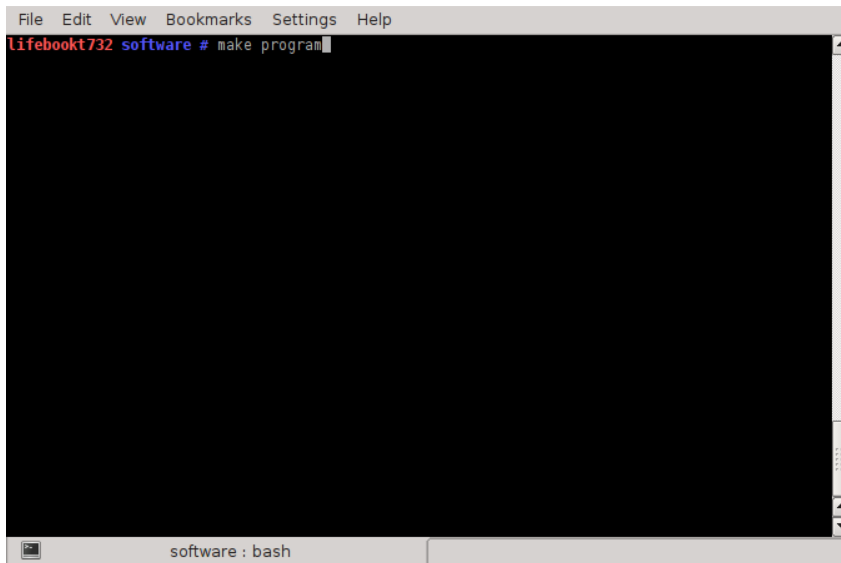
Place the JTAG connector in the AVR receptacle on the Atmel ICE



Attach the provided cable to the JTAG port of the PACMAN target board.



Open a terminal and type the following command:

A screenshot of a terminal window. The window has a title bar with menu items: File, Edit, View, Bookmarks, Settings, Help. The terminal text shows the prompt 'lifebookt732 software # make program' followed by a cursor. The window's status bar at the bottom indicates 'software : bash'.

NOTE: You will have to run this command as root!

Here is successful programming output:

```
lifebookt732 software # make program
avr-size -C build/pacman.hex

AVR Memory Usage
-----
Device: Unknown
Program:      0 bytes
(.text + .data + .bootloader)

Data:         0 bytes
(.data + .bss + .noinit)

avrdude -F -V -c atmelice -p c32 -U flash:w:build/pacman.hex
avrdude: usbdev_open(): WARNING: failed to set configuration 1: Device or resource busy
avrdude: AVR device initialized and ready to accept instructions
Reading | ##### | 100% 0.07s
avrdude: Device signature = 0x1e9581
avrdude: NOTE: "flash" memory has been specified, an erase cycle will be performed
        To disable this feature, specify the -D option.
avrdude: erasing chip
avrdude: reading input file "build/pacman.hex"
avrdude: input file build/pacman.hex auto detected as Intel Hex
avrdude: writing flash (11636 bytes):
Writing | ##### | 100% 1.13s
avrdude: 11636 bytes of flash written
avrdude: safemode: Fuses OK (E:FF, H:99, L:CF)
avrdude done. Thank you.
```