jc2pmon Documentation

Version: 03.08.2014

wmmnpr@gmail.com

Inhalt

[Preface 2](#_Toc396194883)

[Welcome 2](#_Toc396194884)

[Architecture 4](#_Toc396194885)

[Running the Test Program 5](#_Toc396194886)

[Understanding the Test Program 6](#_Toc396194887)

[Understanding CSafe Commands 7](#_Toc396194888)

[The Natives Libraries 9](#_Toc396194889)

[Gluegen 10](#_Toc396194890)

[Building the installer 11](#_Toc396194891)

[Appendix 11](#_Toc396194892)

# Preface

This document describes the important aspects of jc2pmon project. For comments, please contact [wmmnpr@gmail.com](mailto:wmmnpr@gmail.com).

Herein will PM refer to both the PM3 and PM4 Performances Monitors from Concept2.

# Welcome

jc2pmon is a Java library, which enables Java applications to communicate with the performance monitor (PM) of the Concept2 Indoor Rower using the runtime C++ libraries provided by the [Concept2 SDK](http://www.concept2.com/service/software/software-development-kit).



All the methods, except for the asynchronous I/O functions, provided by the PM3CsafeCP.lib and PM3DDICP.lib, are made available through JNI.

The methods of the Java library can be understood by referencing the [Communication Interface Definition](https://bitbucket.org/wmmnpr/jc2pmon/downloads/Concept2%20PM%20Communication%20Interface%20Definition%20014.pdf) provided by Concept2.

The jc2pmon project contains both the native and Java code in its Git repository. The code was generated using [Gluegen (v2.1.5)](http://jogamp.org/gluegen/www/).



In order to determine if the jc2pmon library might useful to you, see “Running the Test Program”.

# Architecture

The blue rectangles are components that belong to the jc2pmon and the others belong to the concept2 SDK. The jc2pmon is a Java library, jar file, whic loads the two native libraries, which inturn depend of those libraries in the lower part of the diagramm.

|  |
| --- |
| jc2pmon  PM3DDICP\_JNI  PM3CsafeCP\_JNI  PM3USBCP  PM3DDICP  PM3CsafeCP |

Library Hierarchy

To learn more about the PM3CsafeCP\_JNI and PM3DDICP\_JNI, see The Natives Libraries.

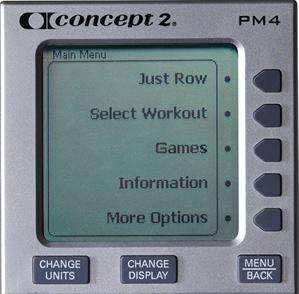
# Running the Test Program

The test program logs all output to the console. To run the program:

1. Download the appropriate archive file from the website and unzip it.
2. Connect the PC and the PM3/4 using a USB A male- B male cable.



1. Make sure that the PM3/4 boots and that the Main Menu is shown (see below).

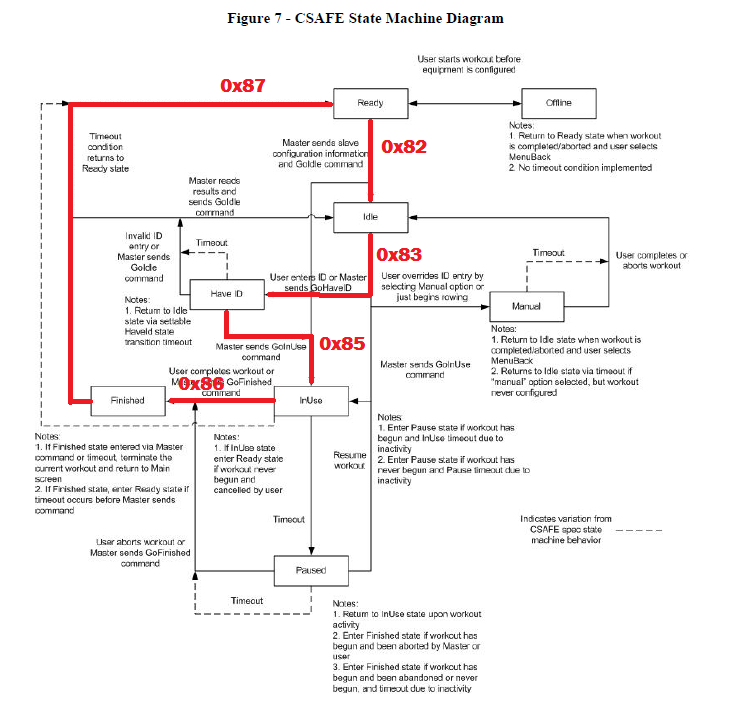


1. Double click on the jar file and wait.
2. Eventually, a screen for a fix distance100 meter workout will be shown.
3. Start rowing.
4. The test / workout will end either when the 100 meters has been rowed or 1 minute has elapsed.
5. The program will exit.

If the above executes to completion, then you have found the proper libraries of your architecture.

# Understanding the Test Program

To understand the test program (org.wmmnpr.junit.PM3CSafeRowTest), it’s best to first take a look at the state diagram of the PM computer. Below is a copy of the state diagram, which was taken from the PM documentation. The test program relevant transitions have been marked in red. Next to each transition is the hexadecimal number of the corresponding command that initiates the transition.



Before setting up the workout, the test program makes sure that the PM is in the *Ready* state before it starts configuring the workout with the *CSAFE\_SETHORIZONTAL\_*CMD command. The entries in the following array are executed sequentially during the setup phase. The program will abort if any of the steps fails.

|  |
| --- |
| **int** cmd\_data[][] = {  {*CSAFE\_GOFINISHED\_CMD*},  {*CSAFE\_GOREADY\_CMD*},  {*CSAFE\_SETHORIZONTAL\_CMD*, 0x03, 0x01, 0x00, 0x22},  {*CSAFE\_SETUSERCFG1\_CMD*, 0x07, *CSAFE\_PM\_SET\_SPLITDURATION*, 0x05, 0x80, 0x64, 0x00, 0x00, 0x00},  {*CSAFE\_SETPROGRAM\_CMD*, 0x02, 0x00, 0x00},  {*CSAFE\_GOIDLE\_CMD*},  {*CSAFE\_GOHAVEID\_CMD*},  {*CSAFE\_GOINUSE\_CMD*}  }; |

PM Configuration Command Array

In the table below, are the hexadecimal numbers corresponding to the constant literals used in the command data array.

|  |
| --- |
| **public** **final** **static** **int** *CSAFE\_SETHORIZONTAL\_CMD* = 0x21;  **public** **final** **static** **int** *CSAFE\_SETPROGRAM\_CMD* = 0x24;  **public** **final** **static** **int** *CSAFE\_SETUSERCFG1\_CMD* = 0x1A;  **public** **final** **static** **int** *CSAFE\_PM\_GET\_WORKTIME* = 0xA0;  **public** **final** **static** **int** *CSAFE\_PM\_GET\_WORKDISTANCE* = 0xA3;    **public** **final** **static** **int** *CSAFE\_RESET\_CMD* = 0x81;  **public** **final** **static** **int** *CSAFE\_GOIDLE\_CMD* = 0x82;  **public** **final** **static** **int** *CSAFE\_GOHAVEID\_CMD* = 0x83;  **public** **final** **static** **int** *CSAFE\_GOINUSE\_CMD* = 0x85;  **public** **final** **static** **int** *CSAFE\_GOFINISHED\_CMD* = 0x86;  **public** **final** **static** **int** *CSAFE\_GOREADY\_CMD* = 0x87; |

Hexadecimal – Command Literal Mapping

# **Understanding** CSafe Commands

CSAFE\_SETHORIZONTAL\_CMD (0x24)

|  |  |  |
| --- | --- | --- |
| Byte | Example | Description |
| 0 | 0x03 | Length of bytes in command |
| 1 | 0x01 | LSB of number 100m units. |
| 2 | 0x00 | MSB of number 100m units. |
| 3 | 0x22 | 0.1 km. See Appendix CSafe Units (34) |

Example:

{*CSAFE\_SETHORIZONTAL\_CMD*, 0x03, 0xE0, 0x06, 0x21}

Enter, from right to left, the hexadecimal numbers into the Windows’ calculator, change to the decimal view and then multiply by the units.

|  |  |
| --- | --- |
|  |  |

CSAFE\_PM\_GET\_WORKDISTANCE (0xA3)

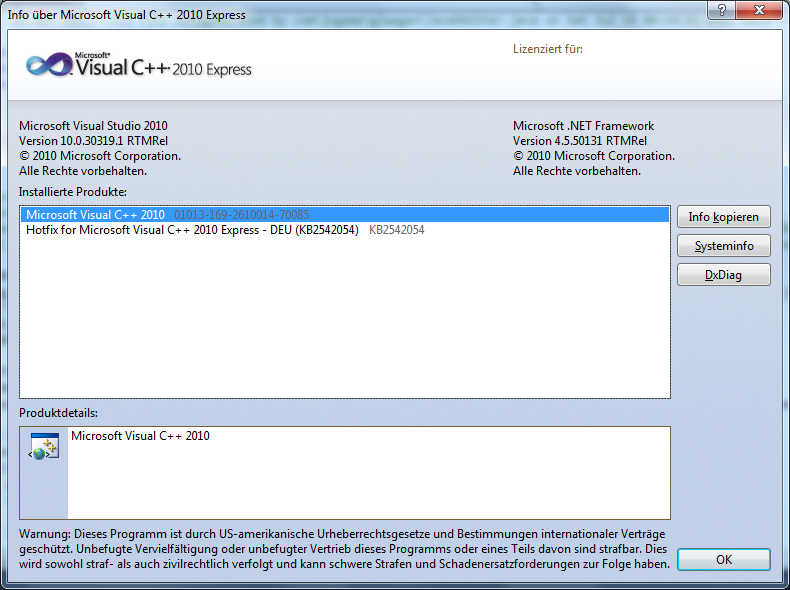
The program also shows how to convert the distance to response to a floating point decimal.

|  |
| --- |
| //[1A:0E:A0:05:00:00:00:00:00:A3:05:E8:03:00:00:00]  **float** distance = (rsp\_data[11] + rsp\_data[12] \* 256 + rsp\_data[13] \* 65536  + rsp\_data[14] \* 16777216) \* 10 + rsp\_data[15];  distance /= 100.0; |

# The Natives Libraries

Basic Information

The generated C code from the Gluegen program was compiled with Microsoft Visual C++ compiler (see below) for the Windows’ native libraries. The project can be found in the native/C2PM3\_JNI directory of the project’s Git repository.



See [here](http://jiggermast.blogspot.de/2014/07/creating-java-libraries-from-dlls-for.html) to learn about changes made to the code.

Debugging the native libraries

The jc2pmon native libraries can be configured to output additional debugging information by set the environment variable pm3.trace to true.

Windows:

|  |
| --- |
| >set pm3.trace=true |

# Gluegen

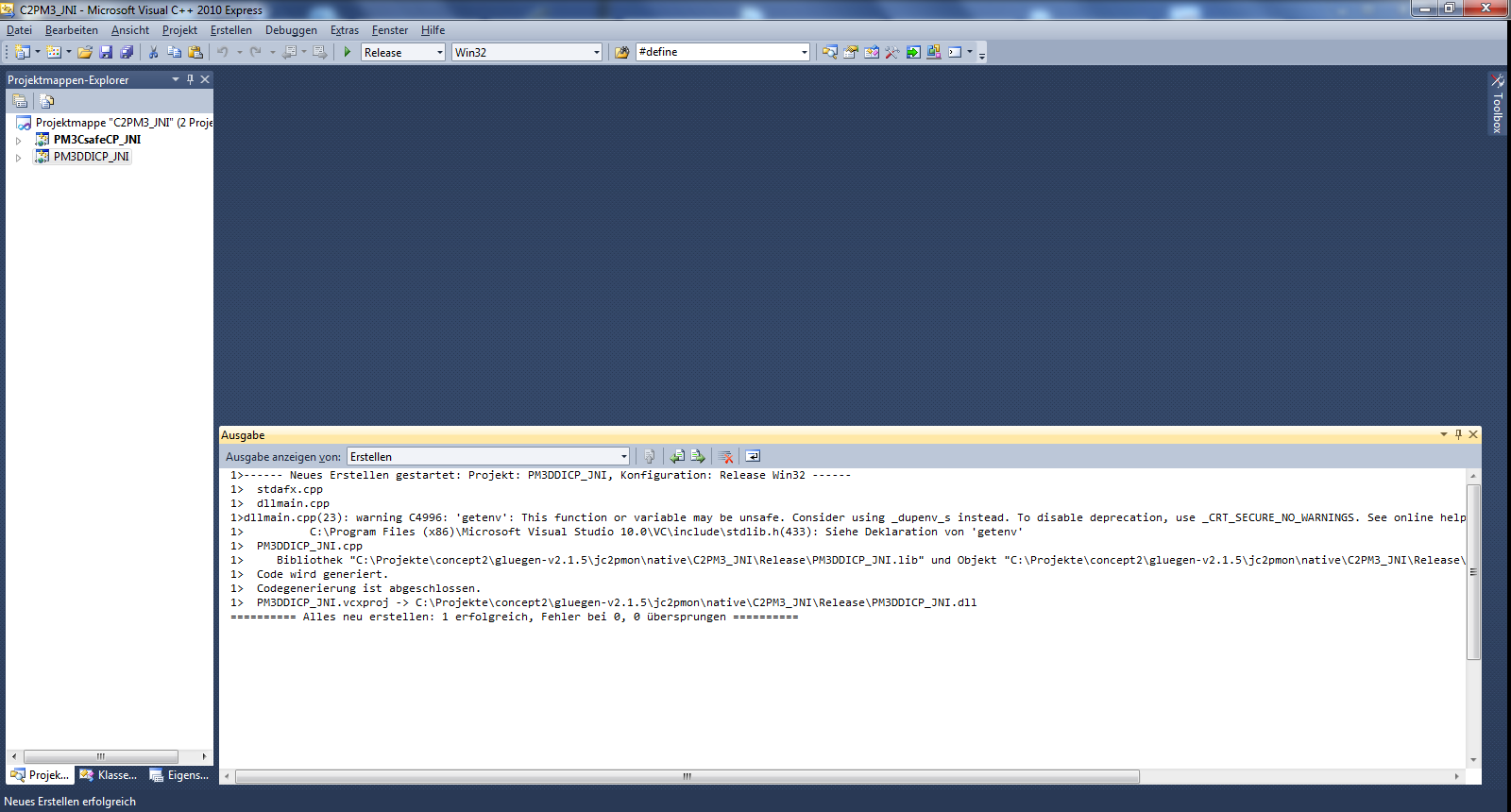
TODO

Configuration files.

See %JC2PMON\_HOME%/gluegen/ gluegen-run.bat

# Building the installer

Open %JC2PMON\_HOME%\native\C2PM3\_JNI\C2PM3\_JNI.sln and build the “Release” configuration of both projects.



>cd %JC2PMON\_HOME%/javaws/jc2pmon

>mvn install -DskipTests=true

>build-launcher.bat

>cd %JC2PMON\_HOME%

>%WIX\_HOME%\candle.exe jc2pmon.wxs

>%WIX\_HOME%\light.exe -ext WixUIExtension -ext WixUtilExtension jc2pmon.wixobj

# Appendix

***CSafe Units***

Source: <http://www.fitlinxx.com/csafe/Commands.htm> (3.2.2 Units)

Some of the commands and responses described above contain quantities whose meaning depends on the units being used. When there is more than one unit that could be used to express a quantity a extra byte has been added to the command or response structure to specify the correct unit. This method handles the case of the unit being changed during a workout. The following table gives the unit code for several standard units:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | mile | 33 | Km | 65 | floors | 97 | Amperes |
| 2 | 0.1mile | 34 | 0.1km | 66 | 0.1 floors | 98 | 0.001 Amperes |
| 3 | 0.01 mile | 35 | 0.01km | 67 | steps | 99 | Volts |
| 4 | 0.001 mile | 36 | Meter | 68 | revolutions | 100 | 0.001 Volts |
| 5 | ft | 37 | 0.1 meter | 69 | strides | 101 |  |
| 6 | inch | 38 | Cm | 70 | strokes | 102 |  |
| 7 | lbs. | 39 | Kg | 71 | beats | 103 |  |
| 8 | 0.1 lbs. | 40 | 0.1 kg | 72 | calories | 104 |  |
| 9 |  | 41 |  | 73 | Kp | 105 |  |
| 10 | 10 ft | 42 |  | 74 | % grade | 106 |  |
| 11 |  | 43 |  | 75 | 0.01 % grade | 107 |  |
| 12 |  | 44 |  | 76 | 0.1 % grade | 108 |  |
| 13 |  | 45 |  | 77 |  | 109 |  |
| 14 |  | 46 |  | 78 |  | 110 |  |
| 15 |  | 47 |  | 79 | 0.1 floors/minute | 111 |  |
| 16 | mile/hour | 48 | Km/hour | 80 | floors/minute | 112 |  |
| 17 | 0.1 mile/hour | 49 | 0.1Km/hour | 81 | steps/minute | 113 |  |
| 18 | 0.01 mile/hour | 50 | 0.01 Km/hour | 82 | revs/minute | 114 |  |
| 19 | ft/minute | 51 | Meter/minute | 83 | strides/minute | 115 |  |
| 20 |  | 52 |  | 84 | stokes/minute | 116 |  |
| 21 |  | 53 |  | 85 | beats/minute | 117 |  |
| 22 |  | 54 |  | 86 | calories/minute | 118 |  |
| 23 |  | 55 | Minutes/mile | 87 | calories/hour | 119 |  |
| 24 |  | 56 | Minutes/km | 88 | Watts | 120 |  |
| 25 |  | 57 | Seconds/km | 89 | Kpm | 121 |  |
| 26 |  | 58 | Seconds/mile | 90 | Inch-Lb | 122 |  |
| 27 |  | 59 |  | 91 | Foot-Lb | 123 |  |
| 28 |  | 60 |  | 92 | Newton-Meters | 124 |  |
| 29 |  | 61 |  | 93 |  | 125 |  |
| 30 |  | 62 |  | 94 |  | 126 |  |
| 31 |  | 63 |  | 95 |  | 127 |  |
| 32 |  | 64 |  | 96 |  | 128 |  |