



# **FIT SDK**

## Introductory Guide

## Copyright Information and Usage Notice

This information disclosed herein is the exclusive property of Dynastream Innovations Inc. The recipient and user of this document must be an ANT+ Adopter pursuant to the ANT+ Adopter's Agreement and must use the information in this document according to the terms and conditions of the Adopter's Agreement and the following:

- a) You agree that any products or applications that you create using the ANT+ Documents and ANT+ Design Tools will comply with the minimum requirements for interoperability as defined in the ANT+ Documents and will not deviate from the standards described therein.
- b) You agree not to modify in any way the ANT+ Documents provided to you under this Agreement.
- c) You agree not to distribute, transfer, or provide any part of the ANT+ Documents or ANT+ Design Tools to any person or entity other than employees of your organization with a need to know.
- d) You agree to not claim any intellectual property rights or other rights in or to the ANT+ Documents, ANT+ Design Tools, or any other associated documentation and source code provided to you under this Agreement. Dynastream retains all right, title and interest in and to the ANT+ Documents, ANT+ Design Tools, associated documentation, and source code and you are not granted any rights in or to any of the foregoing except as expressly set forth in this Agreement.
- e) DYNASTREAM MAKES NO CONDITIONS, WARRANTIES OR REPRESENTATIONS ABOUT THE SUITABILITY, RELIABILITY, USABILITY, SECURITY, QUALITY, CAPACITY, PERFORMANCE, AVAILABILITY, TIMELINESS OR ACCURACY OF THE ANT+ DOCUMENTS, ANT+ DESIGN TOOLS OR ANY OTHER PRODUCTS OR SERVICES SUPPLIED UNDER THIS AGREEMENT OR THE NETWORKS OF THIRD PARTIES. DYNASTREAM EXPRESSLY DISCLAIMS ALL CONDITIONS, WARRANTIES AND REPRESENTATIONS, EXPRESS, IMPLIED OR STATUTORY INCLUDING, BUT NOT LIMITED TO, IMPLIED CONDITIONS OR WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, DURABILITY, TITLE AND NON-INFRINGEMENT, WHETHER ARISING BY USAGE OF TRADE, COURSE OF DEALING, COURSE OF PERFORMANCE OR OTHERWISE.
- f) You agree to indemnify and hold harmless Dynastream for claims, whether arising in tort or contract, against Dynastream, including legal fees, expenses, settlement amounts, and costs, arising out of the application, use or sale of your designs and/or products that use ANT, ANT+, ANT+ Documents, ANT+ Design Tools, or any other products or services supplied under this Agreement.

If you are not an ANT+ Adopter, please visit our website at [www.thisisant.com](http://www.thisisant.com) to become an ANT+ Adopter. Otherwise you must destroy this document immediately and have no right to use this document or any information included in this document.

The information contained in this document is subject to change without notice and should not be construed as a commitment by Dynastream Innovations Inc.

Products sold by DYNASTREAM are not designed for use in life support and/or safety equipment where malfunction of the Product can reasonably be expected to result in injury or death. Your use or sell such products for use in life support and/or safety applications at your own risk and agree to defend, indemnify and hold harmless DYNASTREAM from any and all damages, claims, suits or expense resulting from such use.

©2011 Dynastream Innovations Inc. All Rights Reserved.

## Revision History

Revision	Effective Date	Description
1.0	May 2010	Initial release
1.1	March 2011	Updated License
1.2	November 2011	Updated format Added "alternate message configurations" section

## Table of Contents

<b>1</b>	<b>Overview of the FIT SDK .....</b>	<b>5</b>
1.1	SDK Package Contents.....	5
<b>2</b>	<b>Customizing C code for a specific product .....</b>	<b>5</b>
2.1	Generating code for multiple products .....	5
2.2	Data Structure Alignment.....	5
2.3	Selecting messages and fields .....	5
2.4	File Structures .....	6
2.5	Capabilities.....	6
2.6	Alternate Message Configurations.....	7
<b>3</b>	<b>Using FitCSVTool .....</b>	<b>8</b>

## 1 Overview of the FIT SDK

### 1.1 SDK Package Contents

The FIT SDK includes:

- FitGen.exe -
- config.csv -
- c\ - C code
  - examples\ - Example encode and decode applications
- cpp\ - C++ code
  - examples\ - Example encode and decode applications
- java\ - Java code
  - fit.jar - .FIT java package
  - FitCSVTool.jar - Executable CSV conversion tool.
  - FitTestTool.jar - Executable test tool.
  - com\garmin\fit\ - Source code.
  - doc\ - Java API documentation.

## 2 Customizing C code for a specific product

The C code can be customized to a product specific set of .FIT messages and fields. This allows the code to be optimized for the product to minimize RAM and ROM resource requirements.

1. Modify compile options in fit\_config.h to suit product requirements.
2. Modify config.csv to customize the .FIT profile for a product(s).
3. Run FitGen.exe to regenerate the C code with the new configuration.

### 2.1 Generating code for multiple products

Additional columns can be added to config.csv to specify configurations for multiple products. The default product is 'SDK' which includes a possible messages and fields. fit\_product.c/h allow the product configuration to be selected at compile time by defining FIT\_PRODUCT\_<PRODUCT\_NAME> in fit\_config.h.

### 2.2 Data Structure Alignment

Message data is accessed directly through auto generated C structures. See FIT\_\*\_MSG type definitions. The order of fields is optimized to minimize the structure padding requirements. The default alignment is 4 bytes and can be configured in config.csv by product.

### 2.3 Selecting messages and fields

Messages and fields are selected by specifying the number of field elements in the product column of config.csv. A field is not included if the cell is 0 or blank. If no fields are included, then the message is also not included in the source.

For example, the following configuration includes a user profile message with a friendly name with maximum length of 16, gender, age and weight. Language is not included in the message because it is set to 0.

Message Name	Field Name	Product
user_profile		
	friendly_name	16
	gender	1
	age	1
	weight	1
	language	0

## 2.4 File Structures

Access to some files such as settings can be setup as a fixed structure of message definitions and data. Defining a fixed file structure allows efficient random access of message data. A message can be included in a file structure by adding the following option to the message row of config.csv:

`f=<file type>,<number of messages in file>`

If file type is "all", then the message is included in all file types. The file\_id message is required in every file so it is always configured as `f=all,1`.

For example, the following configuration includes 3 user profile messages in a settings file:

Message Name	Field Name	Product
user_profile		f=settings,3
	friendly_name	16
	gender	1
	age	1
	weight	1
	language	0

The FIT\_MESG\_OFFSET macro can be used to compute byte offset of a message in a file. For example, the offset of the 3rd user profile in a settings file:

`FIT_MESG_OFFSET(user_profile_mesg, 2, USER_PROFILE_MESG_SIZE, FIT_SETTINGS_FILE)`

The number of files can specified in the "Files:" row of config.csv.

## 2.5 Capabilities

Message and field capabilities can be auto generated. See device file type for more information on device capabilities.

A capability is configured by added the following option to the message or field row of config.csv:

`c=<file type>,<capability type>,<number of messages>`

The options for capability type are: num\_per\_file, max\_per\_file, or max\_per\_file\_type. Each message and field can have multiple capability options (for different file types).

## 2.6 Alternate Message Configurations

A message can be configured to have multiple definitions with different numbers of fields included. Alternate definitions must first be named by adding the following option to the message row of config.csv:

d=<alt-def 1 name>,<alt-def 2 name>,<alt-def 3 name>,...

If  $n$  names are specified, then  $n$  alternate message definitions are created **in addition to** the main message definition. Names can only include letters, numbers and the underscore character. To specify the number of elements for a particular field, the field row of config.csv must be formatted as follows:

<main-def # elements> d=<alt-def 1 # elements>,<alt-def 2 # elements>,<alt-def 3 # elements>,...

For example, the following configuration includes three user profile message definitions. The main definition includes a friendly name with maximum length of 16, gender, age and weight. The alternate definition "long\_name" includes a friendly name with a maximum length of 32. The alternate definition "language\_only" includes only the language field.

Message Name	Field Name	Product
user_profile		d=long_name,language_only
	friendly_name	16 d=32,0
	gender	1 d=0,0
	age	1 d= 0,0
	weight	1 d=0,0
	language	0 d=0,1

Alternate messages can be selected when setting up files and RAM. To set up a file with an alternate message, add the following option to the message row of config.csv:

f=<file\_type>,<number of messages in file>,<alt-def name>

To set up a static copy of a message in RAM, add the following option to the message row of config.csv:

r=<number of messages in RAM>, <alt-def name>

If the alternate definition name is not specified, the main definition is used. For example, the following configuration selects the "long\_name" definition for the settings file but the main definition in the device file.

Message Name	Field Name	Product
user_profile		d=long_name,language_only f=settings,3,long_name f=device,3
	friendly_name	16 d=32,0
	gender	1 d=0,0
	age	1 d= 0,0
	weight	1 d=0,0
	language	0 d=0,1

Note that functions in the SDK that access definitions and sizes through the global message number (for example, Fit\_GetMesgDef) will have return values corresponding only to the *main* definition. To access alternate message definitions, use the fit\_mesg\_defs array using the appropriate FIT\_MESG enumerator as the index.

### 3 Using FitCSVTool

The FitCSVTool is a command line utility that converts FIT information stored in csv files, to binary FIT files, and vice versa. It's usage is described below:

```
.FIT CSV Tool 1.0.0.70
Usage: java -jar FitCSVTool.jar -b|-c <INPUT FILE> <OUTPUT FILE>
       -b <FIT FILE> <CSV FILE>  FIT binary to CSV.
       -c <CSV FILE> <FIT FILE>  CSV to FIT binary.
       -t Enable file verification tests.
       -d Enable debug output (also enables file verification tests).
```

Refer to the \examples directory in the FIT SDK for examples of how to correctly define the .csv file. Note, the .csv files may be created with or without specifying definition messages and local message types. If the local message type is not defined in the .csv file, the fitCSVTool will redefine each message with local message type 0. Refer to the workout .csv files for an example of defining csv's without defining the local message type; and refer to the blood pressure multiuser csv file for an example of defining FIT messages and their respective local message types.