Radiy Platform Configuration Tool

Output Bitstream Files Description

[1. General Information 1](#_Toc483310804)

[2. General binary file structure 1](#_Toc483310805)

[3. Base elements description 2](#_Toc483310806)

[4. Data description items 2](#_Toc483310807)

[5. Frame binary data items 4](#_Toc483310821)

# General Information

The result of build RPCT project is a set of Output Bitstream Files. These files are uploaded to the flash memory of Logic Modules (LM). This document describes the version 5 of Output Bitstream Files.

LM has three flash memory areas: Application Logic, FSC Configuration and Tuning. Thus, there are three types of files: Application Logic data (has .alb extension), FSC Configuration data (has .mcb extension) and Tuning data (has .tub extension). Each file must be uploaded using the corresponding LM socket. Each file contains data for all modules of project subsystem.

# General binary file structure

All files are stored in JSON format. JSON is a text format for the serialization of structured data.

An example of an Output Bitstream File is shown below.

{

"buildConfig": "debug",

"buildNumber": 229,

"buildSoftware": "u7 v0.7.3683 (develop), debug, commit SHA1: 901250023929d42902d8f1309e7839a0cff93bd",

"buildTime": "23.05.2017 13:10:42",

"caption": "LM-1",

"changesetId": 709,

"fileVersion": 5,

"frameSize": 1016,

"frameSizeWithCRC": 1024,

"framesCount": 256,

"projectName": "tun01",

"subsysId": "SUBSYSID00",

"uartId": 260,

"userName": "Administrator",

"z\_description\_channel\_01": {

"desc fields": "Version;AppSignalID;CustomSignalID;Caption;Type;Default;Min;Max;Offset;BitNo",

"desc00000000": "1;#ANALOG\_001;ANALOG\_001;ANALOG\_001;AnalogFloat;100;0;100;0;0"

},

"z\_frame\_0000": {

"data0000": "0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000",

"data0010": "0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000",

"data0020": "0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000",

According to JSON format, parent and child elements are enclosed by “{” and “}” symbols. All elements’ names and values are enclosed in brackets and separated by “:” symbol. More details about the JSON data format can be found at json.org and in RFC-4627.

# Base elements description

All output files have following base elements:

– “buildConfig” - build configuration (“debug” or “release”);

– “buildNumber” – number of the build;

– “buildSoftware” – description of the software that has created the build;

– “buildTime” – time when build was made;

– “caption” - caption of the LM;

– “changesetId”– identifier of the changeset that was built;

– “fileVersion”– version of the Output Bitstream File structure;

– “frameSize”–size of data frame, in bytes;

– “frameSizeWithCRC”–size of data frame with CRC, in bytes;

– “framesCount”–number of data frames in the file;

– “projectName” – caption of the project;

– “subsysId” – subsystem identifier;

– “uartId” – UART type identifier. 257 (101h) is used for Application Logic, 258 (102h) – for FSC Configuration, 260 (104h) – for Tuning;

– “userName” – name of the user who had built the project;

– “z\_description\_channel\_CC” – array of data description items for channel CC (details are described in section 4);

– “z\_frame\_NNNN” – binary data for frame NNNN (details are described in section 5).

# Data description items

## Section *z\_description\_channel\_CC* contains description of commands and parameters stored in the Output Bitsеream File. For Application Logic it describes commands and parameters, for FSC Configuration – hardware configuration data, for Tuning – default values and ranges. Also this section contains data description arrays (*z\_description\_channel\_CC*).

## Each data description array contains:

## – a record with list of fields (*desc fields)*;

## – data description items (*descNNNNNNNN)*, where *NNNNNNNN* is a counter. *desc fields* and *descNNNNNNNN* are stored in CSV format with semicolon separator.

## Application Logic description item has following elements:

– “Version” – item format version;

– “Address” – address of the command in memory(hex in big-endian format);

– “BinCode” – binary code of the command (hex in big-endian format);

– “Comment” – comment for the command;

– “IsCommand” –tells if this item is a command (boolean);

– “MnemoCode” – mnemonic code of the command.

An example is shown below.

"desc fields": "Version;IsCommand;Address;BinCode;MnemoCode;Comment",

"desc00000000": "1;true;0000;0440000E;APPSTART 14;",

## FSC Configuration description item has following elements:

– “Version” – item format version;

– “BitNo” – number of the bit for discrete parameter (decimal format);

– “Caption” – caption of the parameter;

– “EquipmentID” – identifier of equipment this parameter belongs to;

– “Frame” – frame number (decimal format);

– “Offset” – word offset in the frame (decimal format);

– “Size” – size in bits of the parameter (decimal format);

– “Value” – value of the parameter (hexadecimal format).

An example is shown below.

"desc fields": "Version;EquipmentID;Frame;Offset;BitNo;Size;Caption;Value",

"desc00000000": "1;SYSTEMID\_RACKID\_CH00\_MD00;1;0;0;16;Marker;0xca70",

## Tuning description item has following elements:

– “Version” – item format version;

– “AppSignalID” – application signal identifier;

– “BitNo” – number of the bit for discrete parameters;

– “Caption” – caption of the parameter;

– “CustomSignalID” – custom application signal identifier;

– “Default” – default value for the parameter;

– “Max” – max value for the parameter (decimal format);

– “Min” – min value for the parameter (decimal format);

– “Offset” – offset of the parameter (decimal format);

– “Type” – type of the parameter (“AnalogFloat”/“AnalogInt”/“Discrete”).

An example is shown below.

"desc fields": "Version;AppSignalID;CustomSignalID;Caption;Type;Default;Min;Max;Offset;BitNo",

"desc00000000": "1;#ANALOG\_001;ANALOG\_001;ANALOG\_001;AnalogFloat;100;0;100;0;0"

# Frame binary data items

Frame binary data is stored in *z\_frame\_NNNN* items. An example is shown below.

"z\_frame\_0003": {

"data0000": "0440 0003 00c0 0140 d2c4 e1b6 0002 0380 d2c6 0000 0002 0140 e1b6 d2c6 0002 00c0",

"data0010": "0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000",

...

...

"data01f0": "0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000",

"frameIndex": 3

}

Frame binary data item contains data part items with identifiers *dataDDDD*, where *DDDD* is start address of this data part in frame. Data part item’s value is a string with frame data in 16-bit words in hexadecimal format, separated by space. Data is stored in big-endian format.

Number of values in one data part is stored in *frameStringWidth* element of the file, by default every data part has 16 16-bit words.

Note for file version 1-4: binary data does not contain last 4 words with checksum, it is generated later with uploader. For example, for 1024-bytes frames only 1016 bytes are stored in binary data file.

Starting from version 5, last 8 bytes of the binary data contain the 64-bit checksum of the data (CRC64), stored in big-endian. For example, all 1024 bytes of the frame are stored, 1016 bytes contain the data, and last 8 bytes - the checksum.

Also binary data item contains *frameIndex* element with zero based index of theframe.

For FSC Configuration and Tuning section *DataNNNN*is is formed according to *Section 3 of D8.21.10 FSC ED AD. Data Protocols and Packages.*