Radiy Platform Configuration Tool

Output Bitstream File (\*.bts) Description

**Version 2**

Contents

[1. General Information 2](#_Toc500852724)

[2. General binary file structure 2](#_Toc500852725)

[3. Base elements description 5](#_Toc500852726)

[4. Subsystem information 5](#_Toc500852727)

[5. Logic module information 5](#_Toc500852728)

[6. Subsystem data 6](#_Toc500852729)

[7. Firmware data items 6](#_Toc500852730)

[8. Data description items 6](#_Toc500852731)

[9. Frame binary data items 8](#_Toc500852732)

# General Information

The result of build RPCT project is an Output Bitstream File that has .bts extension. Data from this file is uploaded to the flash memory of Logic Modules (LM).

The file contains data for all flash memory areas for all logic modules of every subsystem of a of project.

# 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": 118,

"buildSoftware": "u7 v0.7.4618 (rpct-1925), debug, commit SHA1: 9db3cc02b1eca1ce0a8d36d0d70a716e9025e72",

"buildTime": "12.12.2017 08:57:31",

"changesetId": 0,

"fileVersion": 2,

"projectName": "bts",

"userName": "Administrator",

"z\_i\_subsystemsInfo": [

{

"lmDescriptionFile": "LM1\_SF00.xml",

"lmDescriptionNumber": 0,

"subsystemId": "SUBSYSID00",

"subsystemKey": 5,

"z\_modules": [

{

"channel": "A",

"customModuleFamily": 0,

"equipmentId": "SYSTEMID\_RACKID\_CH00\_MD00",

"lmNumber": 1,

"moduleFamily": 4352,

"moduleType": 4352,

"moduleVersion": 0

}

]

},

{

"lmDescriptionFile": "LM1\_SR01.xml",

"lmDescriptionNumber": 1,

"subsystemId": "SUBSYSID01",

"subsystemKey": 0,

"z\_modules": [

{

"channel": "A",

" customModuleFamily ": 0,

"equipmentId": "SYSTEMID\_RACKID\_CH01\_MD00",

"lmNumber": 1,

"moduleFamily": 4352,

"moduleType": 4512,

"moduleVersion": 160

},

{

"channel": "A",

" customModuleFamily ": 0,

"equipmentId": "SYSTEMID\_RACKID\_CH02\_MD00",

"lmNumber": 2,

"moduleFamily": 4352,

"moduleType": 4512,

"moduleVersion": 160

}

]

}

],

"z\_s\_subsystemsData": [

{

"subsystemId": "SUBSYSID01",

"z\_firmwareData": [

{

"eepromFrameCount": 256,

"eepromFramePayloadSize": 1016,

"eepromFrameSize": 1024,

"uartId": 257,

"uartType": "AppLogic",

"z\_description\_channel\_01": {

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

"desc00000000": "1;true;0000;CC400003;APPSTART 3;",

"desc00000001": "1;false;;;;FB's initialization code",

"desc00000002": "1;false;;;;End of FB's initialization code section",

"desc00000003": "1;true;0002;60C0;STOP ;set address of application logic code start",

"desc00000004": "1;false;;;;Start of application logic code",

"desc00000005": "1;true;0003;61C0B40200000000;MOVBC 46082[0], #0;init const bit 0",

"desc00000006": "1;true;0007;89C0B40200010001;MOVBC 46082[1], #1;init const bit 1",

"desc00000007": "1;false;;;;Copy acquired raw data",

"desc00000008": "1;false;;;;Application logic code",

"desc00000009": "1;false;;;;Copy discrete output signals to output modules memory",

"desc00000010": "1;true;000B;A180E1B60000;MOVC 57782, #0;write #0 to LM's outputs area",

"desc00000011": "1;false;;;;End of application logic code",

"desc00000012": "1;true;000E;60C0;STOP ;"

},

"z\_description\_channel\_02": {

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

"desc00000000": "1;true;0000;CC400003;APPSTART 3;",

"desc00000001": "1;false;;;;FB's initialization code",

"desc00000002": "1;false;;;;End of FB's initialization code section",

"desc00000003": "1;true;0002;60C0;STOP ;set address of application logic code start",

"desc00000004": "1;false;;;;Start of application logic code",

"desc00000005": "1;true;0003;61C0B40200000000;MOVBC 46082[0], #0;init const bit 0",

"desc00000006": "1;true;0007;89C0B40200010001;MOVBC 46082[1], #1;init const bit 1",

"desc00000007": "1;false;;;;Copy acquired raw data",

"desc00000008": "1;false;;;;Application logic code",

"desc00000009": "1;false;;;;Copy discrete output signals to output modules memory",

"desc00000010": "1;true;000B;A180E1B60000;MOVC 57782, #0;write #0 to LM's outputs area",

"desc00000011": "1;false;;;;End of application logic code",

"desc00000012": "1;true;000E;60C0;STOP ;"

},

"z\_frame\_0000": {

"data01f0": "0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0bd0 4916 115f ffff",

"frameIndex": 0

},

"z\_frame\_0001": {

"data0000": "ca70 0001 0000 0076 0001 0000 0000 0002 0002 0000 0000 0004 0000 0000 0000 0000",

"data01f0": "0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 a811 b0a0 9e66 2b01",

"frameIndex": 1

},

...

"z\_frame\_0255": {

"data01f0": "0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0bd0 49e9 21ac 0ccf",

"frameIndex": 255

}

},

{

"eepromFrameCount": 256,

"eepromFramePayloadSize": 1016,

"eepromFrameSize": 1024,

"uartId": 258,

"uartType": "Configuration",

"z\_description\_channel\_01": {

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

"desc00000000": "1;SYSTEMID\_RACKID\_CH01\_MD00;1;0;0;16;Marker;0xca70",

"desc00000001": "1;SYSTEMID\_RACKID\_CH01\_MD00;1;2;0;16;Version;0x1",

...

},

"z\_description\_channel\_02": {

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

"desc00000000": "1;SYSTEMID\_RACKID\_CH02\_MD00;1;0;0;16;Marker;0xca70",

"desc00000001": "1;SYSTEMID\_RACKID\_CH02\_MD00;1;2;0;16;Version;0x1",

...

},

"z\_frame\_0000": {

"data01f0": "0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0bd0 4916 115f ffff",

"frameIndex": 0

},

"z\_frame\_0001": {

"data0000": "ca70 0001 0000 0076 0001 0000 0000 0002 0002 0000 0000 0015 0000 0000 0000 0000",

"data01f0": "0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 ebc5 8de3 4beb 050e",

"frameIndex": 1

},

...

"z\_frame\_0255": {

"data01f0": "0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0bd0 49e9 21ac 0ccf",

"frameIndex": 255

}

},

{

"eepromFrameCount": 256,

"eepromFramePayloadSize": 1016,

"eepromFrameSize": 1024,

"uartId": 260,

"uartType": "Tuning",

"z\_frame\_0000": {

"data01f0": "0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0bd0 4916 115f ffff",

"frameIndex": 0

},

"z\_frame\_0001": {

"data0000": "ca70 0001 0000 0076 0001 0000 0000 0002 0002 0000 0000 0003 0000 0000 0000 0000",

"data01f0": "0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 8646 008e c993 4101",

"frameIndex": 1

},

...

"z\_frame\_0255": {

"data01f0": "0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0bd0 49e9 21ac 0ccf",

"frameIndex": 255

}

}

]

}

]

}

According to JSON format, parent and child elements are enclosed by “{” and “}” symbols. Arrays 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;

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

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

– “*projectName*” – caption of the project;

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

– “*z\_i\_subsystemsInfo*” – array of items that contain information about subsystems and modules;

– “*z\_s\_subsystemsData*” – array of items that contain binary data for different flash memory areas for each module of each subsystem.

# Subsystem information

This section contains information about every subsystem in a project and a list of logic modules for every subsystem.

The informaton is stored in elements of *z\_i\_subsystemsInfo* array. Every item of the array has following elements:

– “*lmDescriptionFile*” – name of description file of logic module in RPCT;

– “*lmDescriptionNumber*” – version number of logic module’s description;

– “*subsystemId*” – a string identifier of the subsystem;

– “*subsystemKey*” – a subsystem’s key number;

– “*z\_modules*” – an array that describes logic modules of the subsystem.

# Logic module information

Every module description record contains following elements:

– "*channel*" – a code of subsystem’s logical channel;

– "*customModuleFamily*" – family for non-platform modules;

– "*equipmentId*" – equipment identifier for logic module;

–"*lmNumber*" – number of logic module in the subsystem;

– "*moduleFamily*" – family of the module;

– "*moduleType*" – type of the module;

– "*moduleVersion*" – version of the module.

# Subsystem data

This section contains binary data for flash memory of logic modules. Data is stored in elements of *z\_s\_subsystemsData* aray. Every element contains following items:

– "*subsystemId*" – a string identifier of the subsystem;

– "*z\_firmwareData*" – an array of records with binary data for all types of flash memory.

# Firmware data items

*z\_firmwareData* itemcontains an array of firmware data items for certain type of flash memory.

Firmware data item has following elements:

– “*eepromFramePayloadSize*”–size of data frame, in bytes;

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

– “*eepromFrameCount*”–number of data frames in the file;

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

– “*uartType*” – UART type text description;

– “*z\_description\_channel\_CC*” – array of data description items for channel CC (details are described in section 8);

– “*z\_frame\_NNNN*” – binary data for frame NNNN (details are described in section 9).

# 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": "cc40 0003 60c0 61c0 b402 0000 0000 89c0 b402 0001 0001 7900 d2c4 b501 a180 e1b6",

"data0010": "0000 60c0 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 67e7 5082 96c3 3a2d",

"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.

Every data part string contains 16 16-bit words.

If all values in a data part contains zeroes, this data is not written to the file. For example, data parts “data0020”, “data0030” are skipped in sample data above.

Last 8 bytes of the binary data contain the 64-bit checksum of the data including frame number (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 the frame.

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