

# **Using Multi-Project Feature for Dataman Universal Device Programmers**

**Application Note**

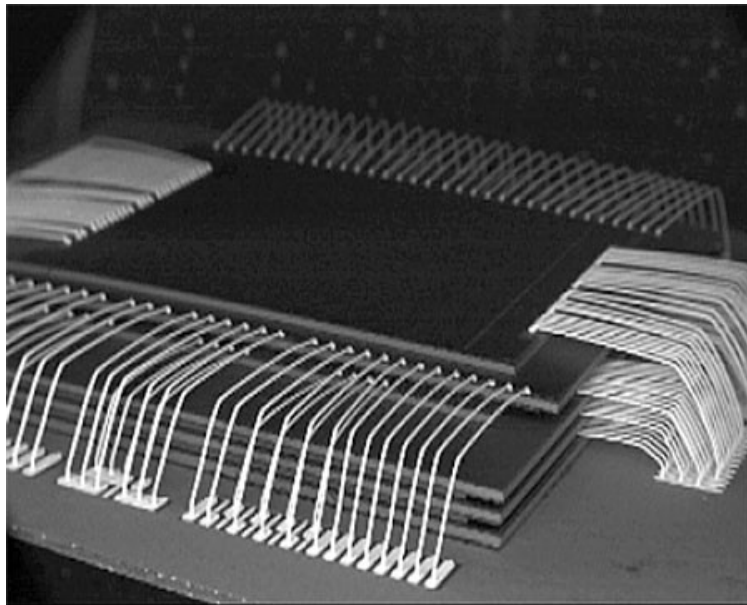
July 2008  
an\_Dataman\_multi\_prj, version 1.00

## Table of contents

Table of contents.....	2
Preface.....	3
Introducing the Multi-Project feature.....	4
Terminology.....	4
Convention.....	5
Using Multi-Project Wizard.....	6
Building-up new Multi-Project file.....	8
Loading existing Multi-Project file.....	12
Running the multi-chip device operation.....	14
Limitations of Multi-Project feature.....	18
Using Multi-Project feature for various programming tasks.....	19
Using Multi-Project feature for programming multi-chip devices.....	19
Using Multi-Project feature for programming multiply partitions into NAND flash devices.....	23
Using Multi-Project feature for programming multiply daisy-chained JTAG devices.....	26
Version history.....	30
Place for your comments.....	31

## Preface

Small form, fast speed and low power consumption – these are three most demanding requests from mobile devices manufacturers. The chip makers respond in simple way – by mixing various memories into single package, so called multi-chip device. There is a wide variety of mixtures of memory types (NOR, NAND, OneNAND, SRAM, DRAM,...) and capacities (from several mega-bytes up to giga-bytes) available on the market.



*Figure 1: Inside a multi-chip device.*

These multi-chip devices put specific requirements on programming process. The programmer must be able to conduct various memory types and switch between them again and again. There must be a high level of understanding between programmer manufacturer and user / technician in programming centre, regarding the issues such as buffer organization, operation settings and similar.

Dataman has discovered the workaround for those obstructions. From now on, you can simple use your memories in your favourite way, no matter whether they are single or multi-chips. From now on, you can use Multi-Project feature for Dataman universal device programmers.

## Introducing the Multi-Project feature

Multi-Project feature was released with version 2.50 of Pg4uw control software. This chapter gives you the introduction to the topic and shows to you how to work with dedicated tool – Multi-Project Wizard.

### *Terminology*

There are several new terms that should be explained before starting any work using Multi-Project feature:

**Multi-Project** – special feature designed to simplify the programming tasks for multi-chip devices.

**Multi-chip device** – (memory) device with two or more independent chips (of the same or various types) in single package.

**Sub-device** – an individual part of multi-chip device. Sub-device is selectable from Pg4uw device list. Once selected, you can work with respective chip in fully manner. You can define, test and save the Project file for the particular chip.

**Master-device** – a multi-chip device unit, consists of Sub-devices. Master-device is selectable from Pg4uw device list, too. Once selected, you can use Multi-Project Wizard for building-up the Multi-Project file from individual Project files and save/load/execute it.

**Project file** – a special file that combines buffer data, device operation options, special options and some level of safety features. It completely defines the way how to treat the device. Once saved, it can be reloaded at any time so the operation can be repeated exactly. Typically, Project files use .eprj file extension. Probably, you are familiar with Project files already from your previous experience with Pg4uw – Dataman's universal device programmers control software.

**Multi-Project file** – a special file that contains user selected Project files. Multi-Project file can include one or more Project files for individual Sub-devices. Typically, Multi-Project files use .epj-m file extension.

**Multi-Project Wizard** – a special aid for building-up the Multi-Project file from individual Project files. The wizard allows user to select Project files that have to be included in Multi-Project file and save them into single Multi-Project file. The wizard also allows to run the multi-chip device operation according to Project files included in Multi-Project file. More information about using the Multi-Project Wizard can be found in following chapters.

## ***Part names convention***

Overview of convention used for Master-device and Sub-device part names in Pg4uw device list:

Master-device: Multi-chip\_device\_original\_part\_name [package\_type]

Sub-devices: Multi-chip\_device\_original\_part\_name [package\_type] (part\_1)

Multi-chip\_device\_original\_part\_name [package\_type] (part\_2)

...

Multi-chip\_device\_original\_part\_name [package\_type] (part\_n)

Examples:

Master-device: TV0057A00CAGD [FBGA107]

Sub-devices: TV0057A00CAGD [FBGA107] (NAND)

TV0057A00CAGD [FBGA107] (NOR)

Master-device: M30W0R6500T0 [LFBGA88]

Sub-devices: M30W0R6500T0 [LFBGA88] (Flash1)

M30W0R6500T0 [LFBGA88] (Flash2)

## Using Multi-Project Wizard

Multi-chip device operation requires the Multi-Project file, which contains partial Project files for all accessed Sub-devices of Master-device. The Multi-Project file can be created only using Multi-Project Wizard (see Figure 2). The Wizard can be accessed by selecting the Master-device from Pg4uw device list, or from Pg4uw menu **Options | Multi-Project Wizard**, or using a key short-cut **<Ctrl+M>**.

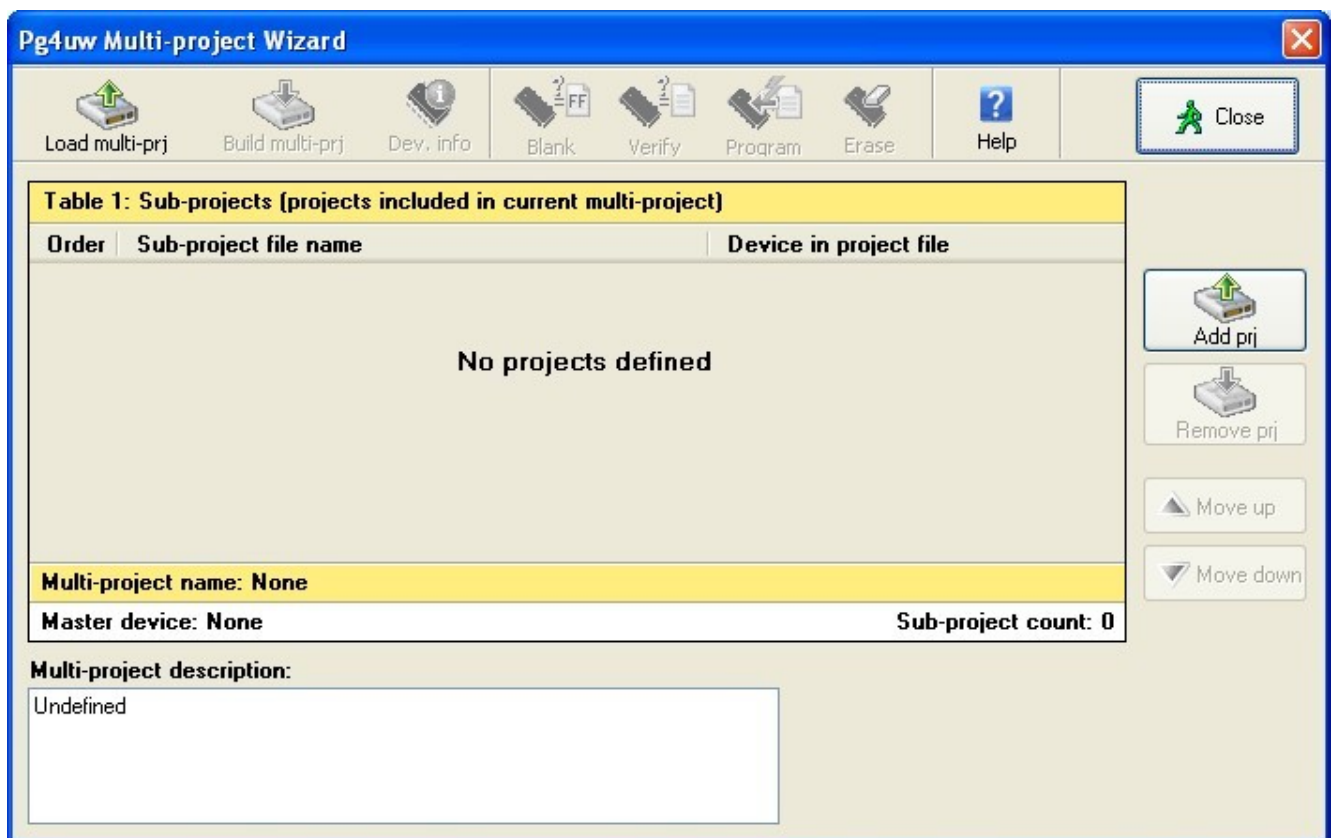


Figure 2: Pg4uw Multi-Project Wizard window

The Wizard allows following main features:

1. Building-up new Multi-Project file
2. Loading existing Multi-Project file
3. Running the multi-chip device operation, regarding to actual Multi-Project file

The Multi-Project Wizard window contains following controls:



Button **Load Multi-Project file**

The button is used for loading existing Multi-Project file.



Button **Build Multi-Project file**

The button is used for building-up new Multi-Project file from selected Project files.



Button **Device info**

The button is used for viewing short device info (if available). See Figure 3 for device info window example.



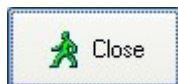
Buttons for device operations – **Blank, Verify, Program, Erase**

These buttons are used for running the selected multi-chip device operation regarding to Project files.



Button **Help**

The button displays the Help window with brief assistance on using Multi-Project Wizard.



Button **Close**

The button terminates the Multi-Project Wizard. After closing, the “unselected” device is automatically selected in Pg4uw software.



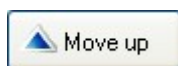
Button **Add project**

The button for adding existing Project file into selected Project files list, making it ready for later Multi-Project file building-up.



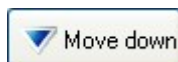
Button **Remove project**

The button for removing the Project file from selected Project files list.



Button **Move up**

Button for moving the Project file one position up in selected Project files list. When running the Multi-Project file, individual Project files are processed in order according to selected Project files list.



Button **Move down**

Button for moving the Project file one position down in selected Project files list.



*Figure 3: Example of Device info window for Master-device*

## Building-up new Multi-Project file

Following steps are recommended when building-up new Multi-Project file:

### 1. Creating Project files

Projects are created in the same way like projects for generic devices:

- select Sub-device from Pg4uw device list
- set device parameters, settings, and load required data into buffer using **Load file** command in Pg4uw
- optionally perform test of device operation by running the device operation on real device
- if everything is OK, the Project file can be created using **Save project** command in Pg4uw

Creating the Project files for various tasks is in more details discussed in further chapters.

### 2. Selecting Master-device

From Pg4uw device list, select the Master-device the Multi-Project file has to be used for. After device selection, Multi-Project Wizard (see Figure 2) is launched automatically.

### 3. Adding Project files

In Multi-Project Wizard, add required Project files using **Add project** button.



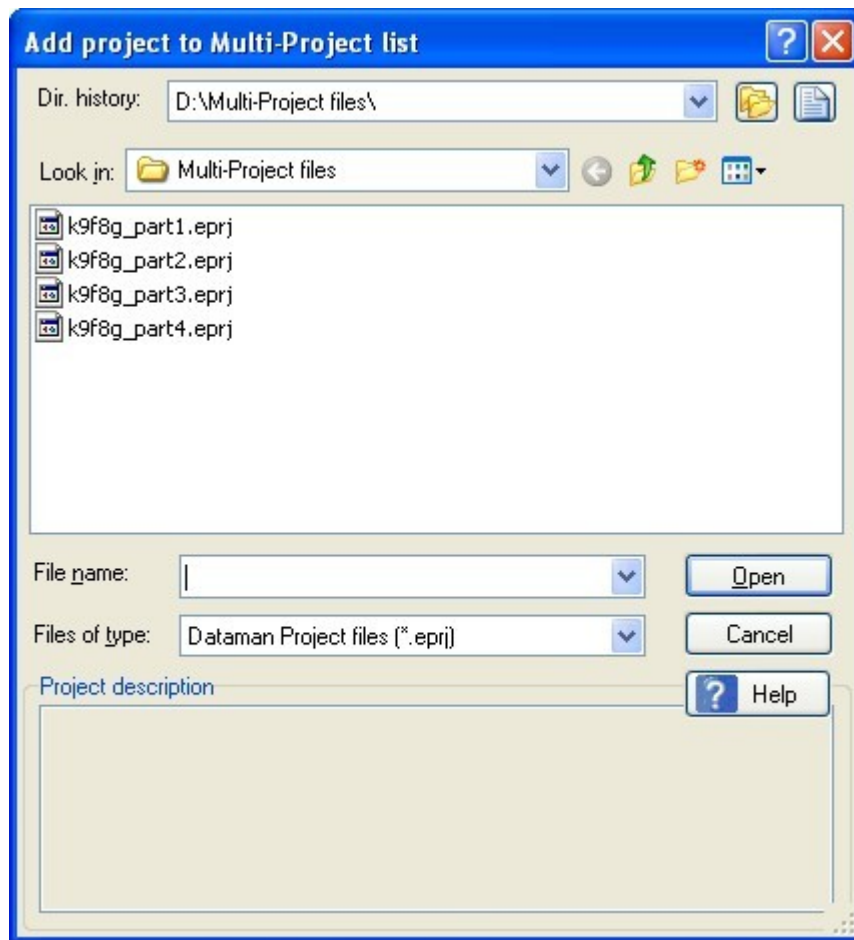


Figure 4: Add Project to Multi-Project list dialog window

4. Building-up final Multi-Project file  
After completion of Project files selection, use **Build Multi-Project file** button to start the building process. A short warning window appears (see Figure 5), click OK to continue.

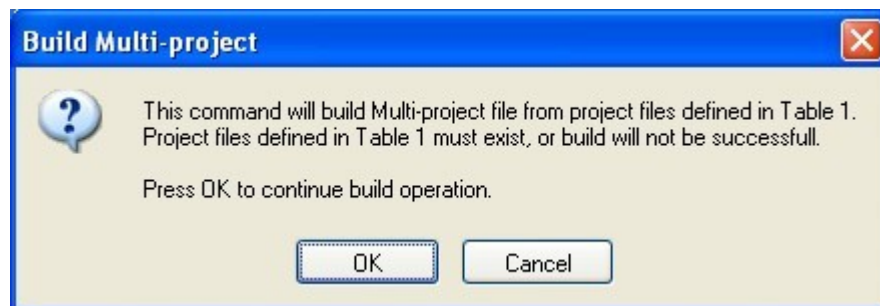


Figure 5: Build Multi-Project file warning window

A Build Multi-Project file As dialog window appears (see Figure 6), similar to the one for Save project command.

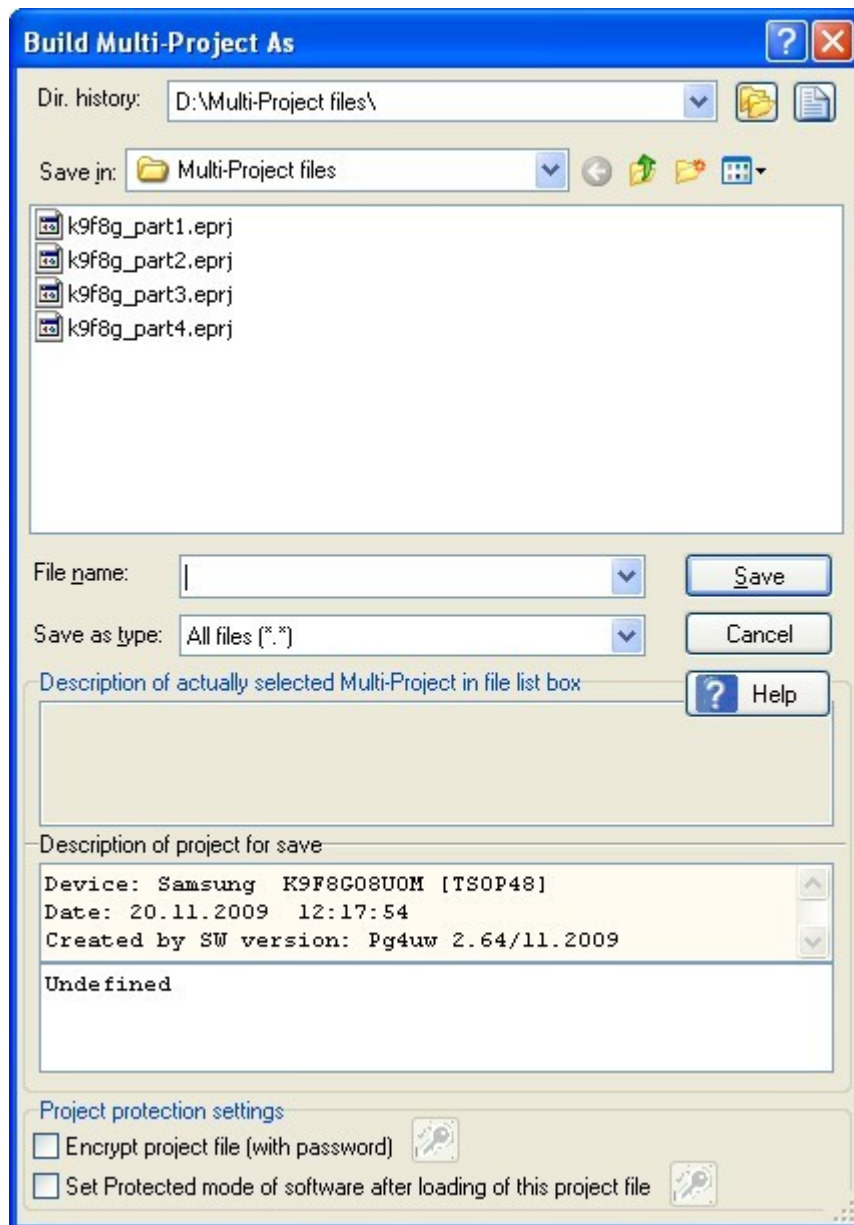


Figure 6: Build Multi-Project file As dialog window

Enter the Multi-Project file name and click Save button. The building process will start. During building, the individual Project files are compacted into single Multi-Project file. A small progress window is displayed on the top of Pg4uw windows stack (see Figure 7). After the building process is finished, the Multi-Project Wizard window is shown again, ready for running the multi-chip device operations (see Figure 8).

New Multi-project file carries all data specified in particular Project files. Those files are not further necessary for correct multi-chip device operation. Only Multi-Project file should be used / shared with other programmers.

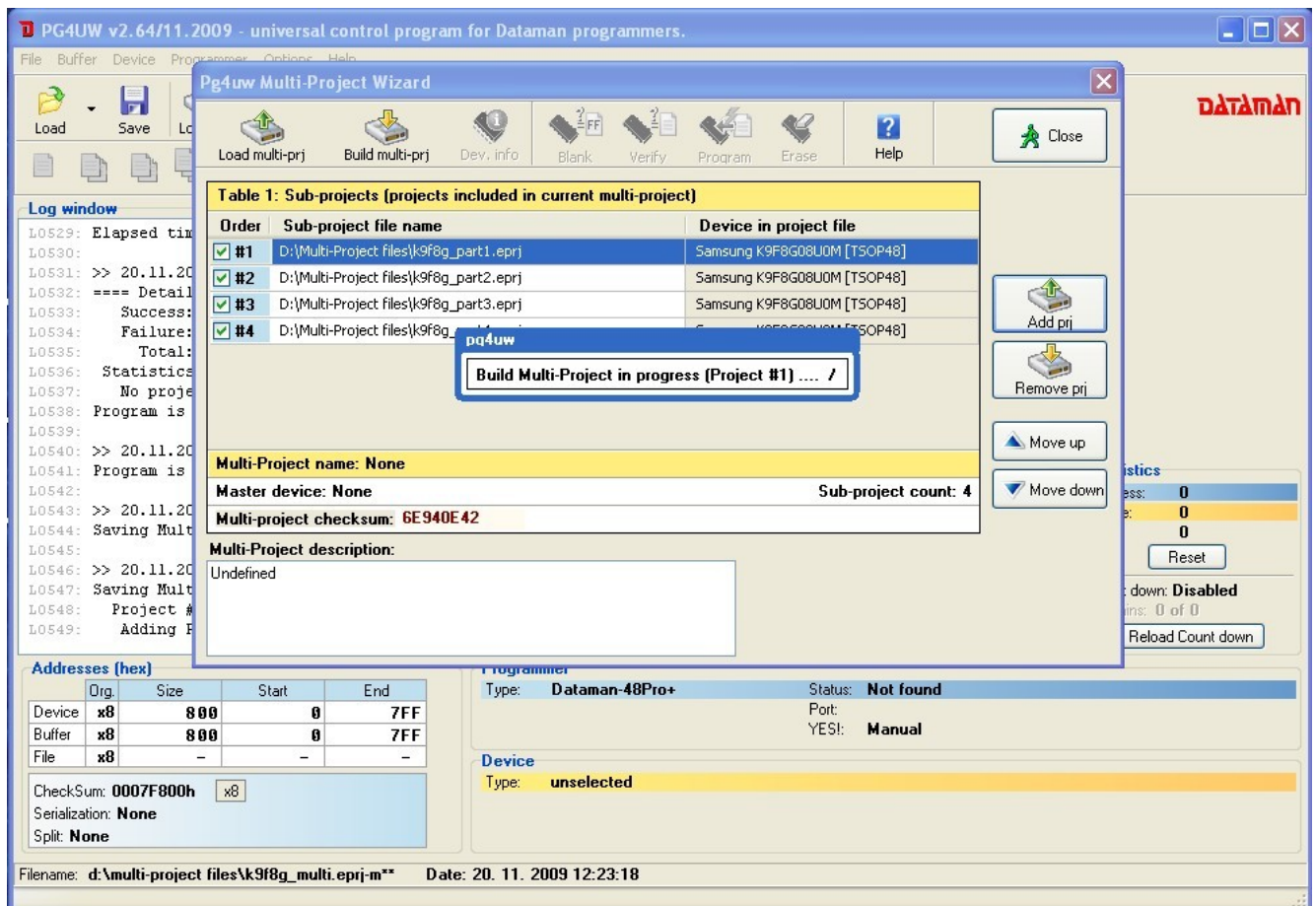


Figure 7: Pg4uw windows during building-up the Multi-Project file

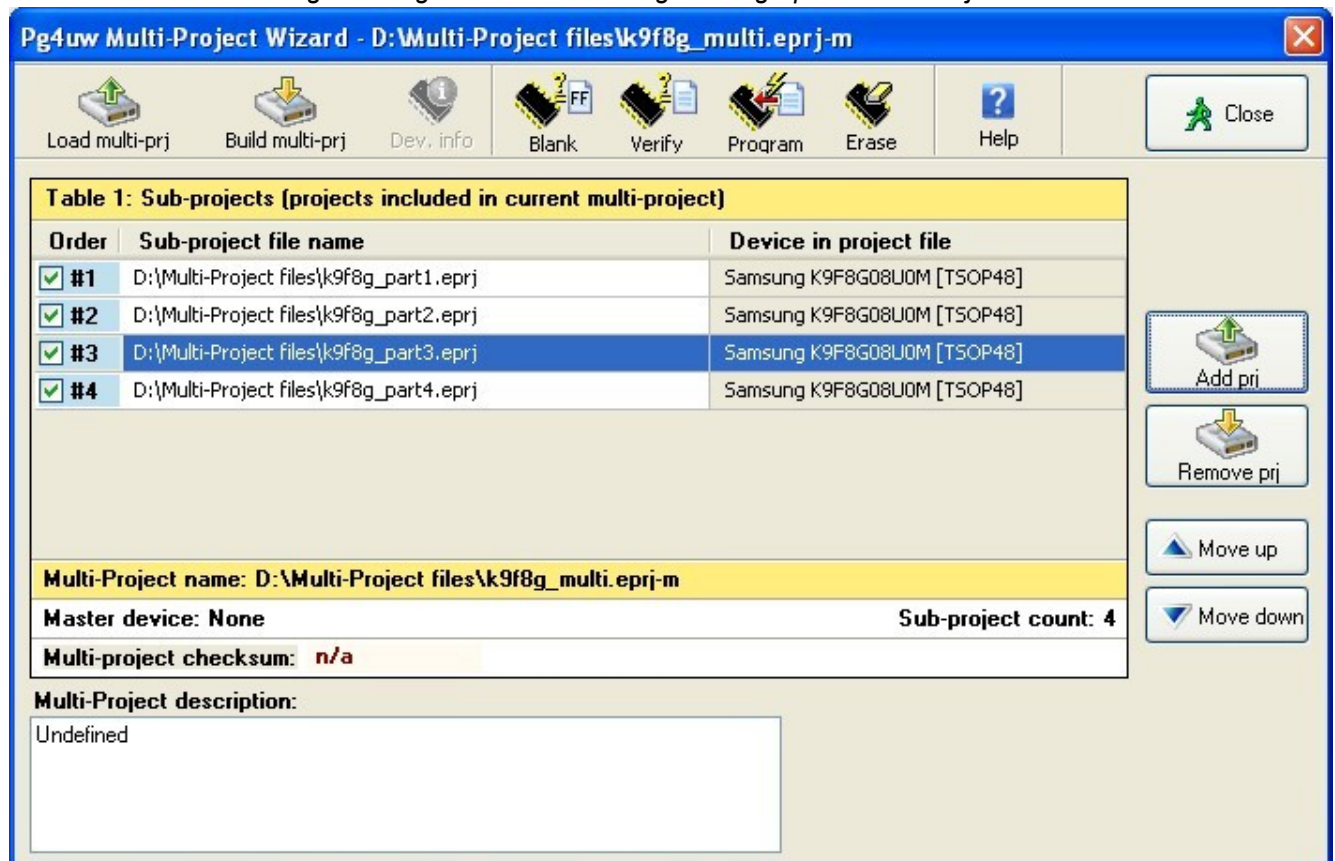
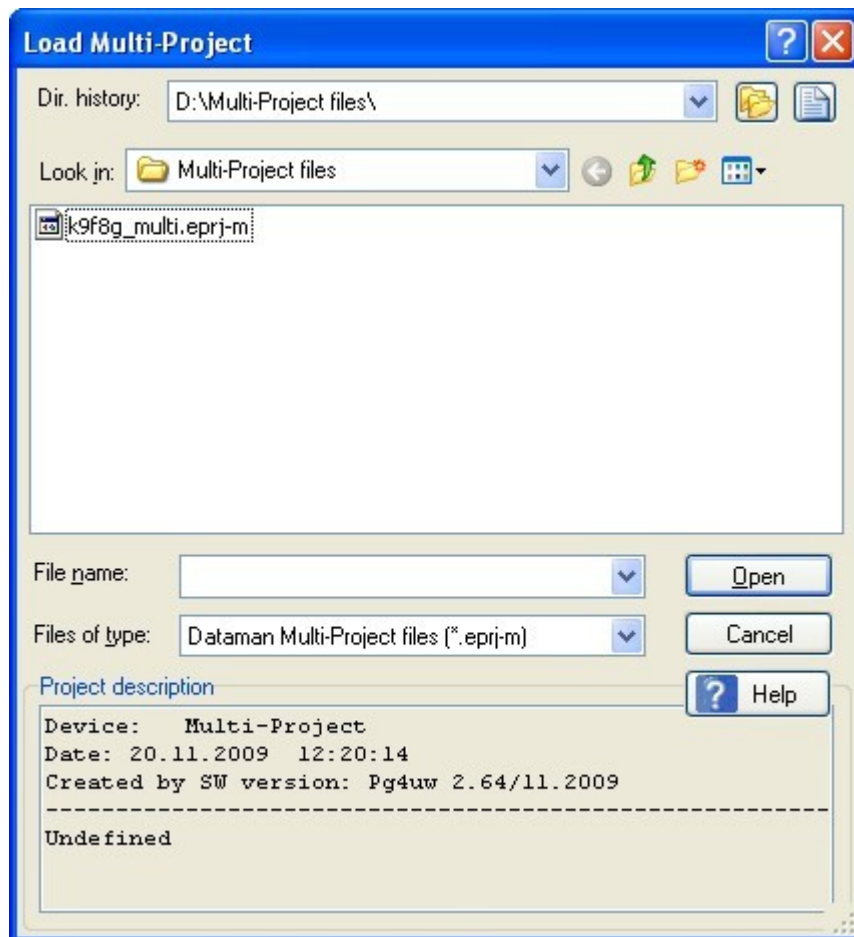


Figure 8: Multi-Project Wizard window with Multi-Project file loaded

## Loading existing Multi-Project file

Simply click the **Load Multi-Project file** button. A Load Multi-Project file dialog window appears, as shown on Figure 9.



*Figure 9: Load Multi-Project file dialog window*

Select the required Multi-Project file and click the **Open** button to load the file. During Multi-Project file loading, all included Project files are pre-loaded and internal Pg4uw environment is set-up for running multi-chip device operation. During the loading, various messages can appear in Pg4uw Log window (see Figure 10). After loading completion, the Multi-Project Wizard window appears again (see Figure 8) and the programmer is ready to run the multi-chip device operation.

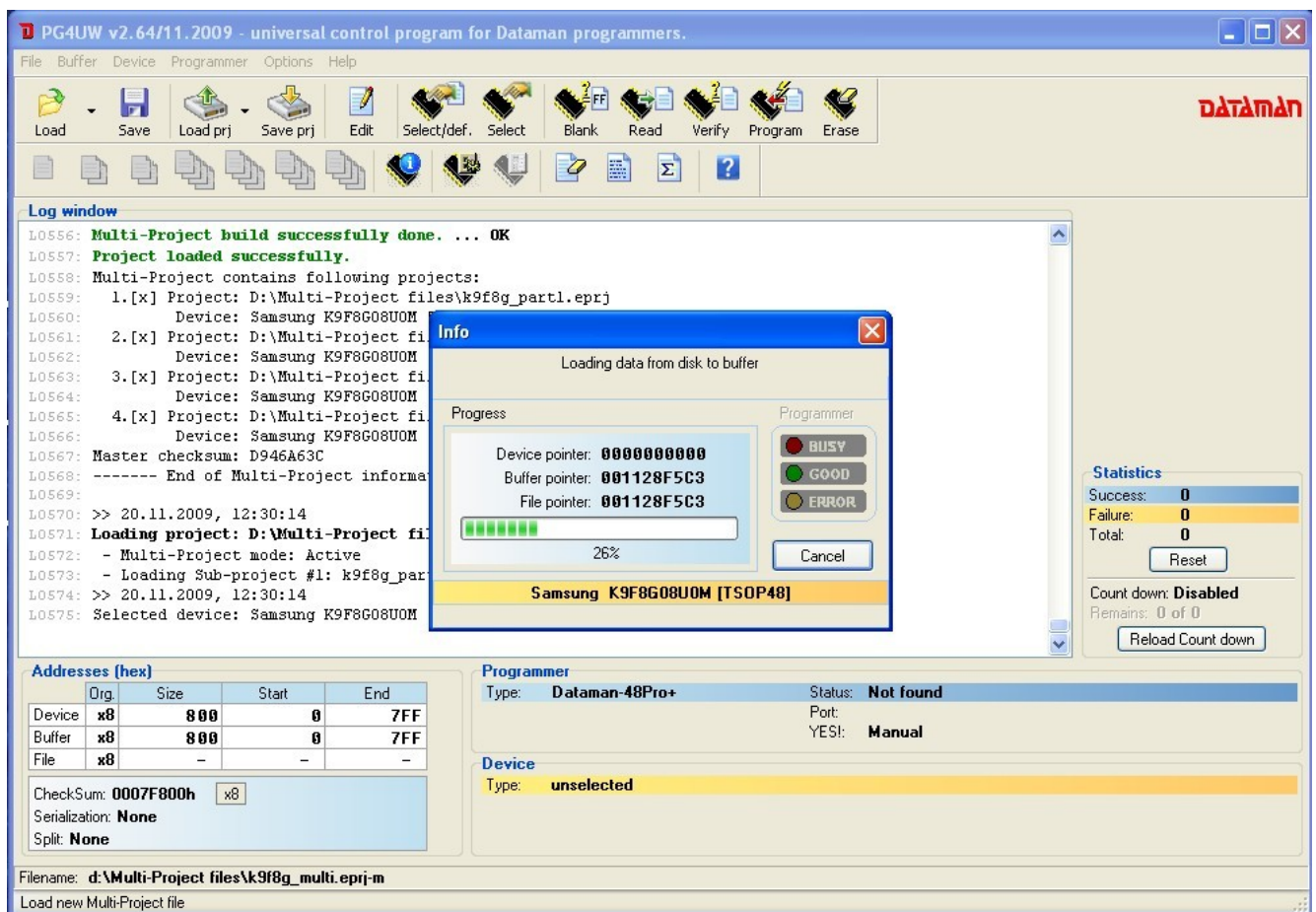


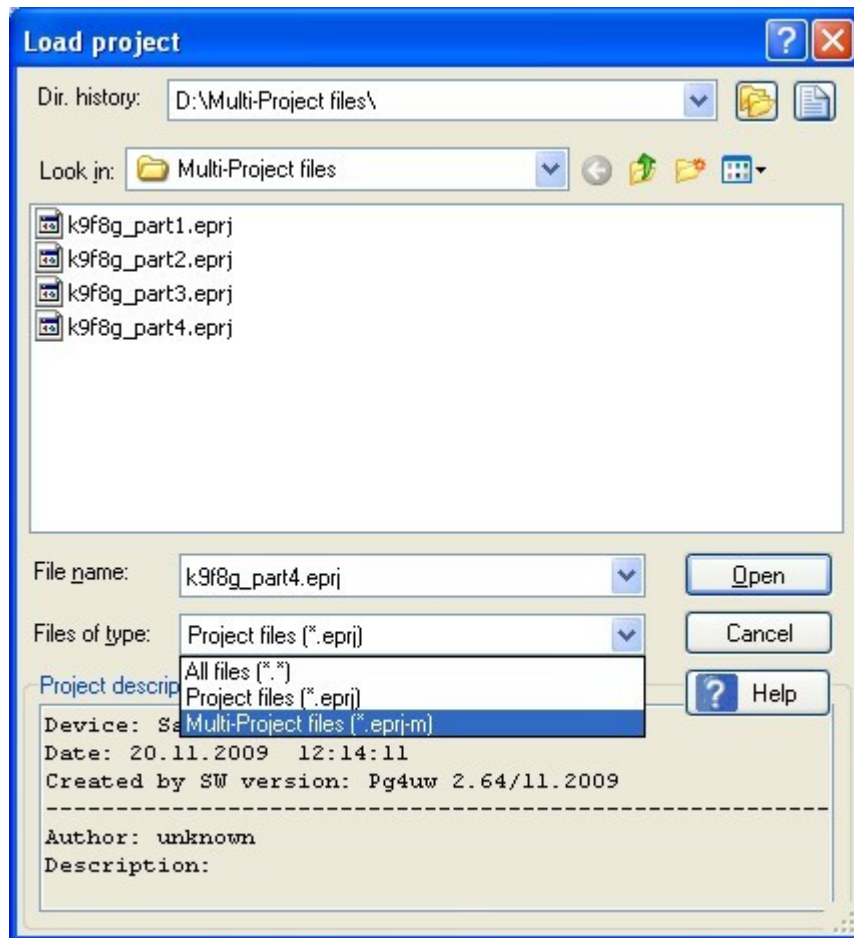
Figure 10: Pg4uw windows during loading the Multi-Project file

Alternatively, existing Multi-Project file can be loaded from Pg4uw software main menu using command **File | Load project**. In Load project dialog window, select **Multi-Project files (\*.eprj-m)** option in **Files of type** drop-down menu to view Multi-Project files instead of (single) Project files (see Figure 11).

Then, select the desired Multi-Project file and click the **Open** button to load the file. A loading process is similar to that one using Multi-Project Wizard. After loading the Multi-Project file is finished, the Multi-Project Wizard window is shown (see Figure 8) and the programmer is ready to run the multi-chip device operation.

After loading the Multi-Project file, you can edit the Multi-Project settings, if necessary. You can change the Project files order (determines the Project files execution order), add new Project files or remove existing Project files. After any change is made, the Multi-Project file must be built-up again to save performed changes. See previous pages for more information about building-up the Multi-Project file.





*Figure 11: Load Project dialog window – select Multi-Project files (\*.eprj-m)*

## Running the multi-chip device operation

Multi-Project feature is intended (but not restricted) to allow programming the multi-chip devices in single button click. Different techniques of usage the Multi-Project feature for running various device programming tasks are explained in next chapters. However, the difference is concerned just to Multi-Project file preparation stage. After successful building-up the Multi-Project file, the procedure is always the same. The only difference on using the Multi-Project file for running the device operations is determined by programmer / control software used for performing the programming tasks.

**For single programming using Pg4uw:**

1. Load existing Multi-Project file using **File | Load project** menu command in Pg4uw main window or **Load Multi-Project file** button in the Multi-Project Wizard window. After the Multi-Project file is successfully loaded, the Multi-Project Wizard window is opened automatically.
2. In the Multi-Project Wizard window, run desired multi-chip device operation using one of available device operation buttons (Blank, Verify, Program, Erase). Mostly the programming operation is used.

Selected device operation is executed as sequence of Project file loading and consequent Sub-device operation for each Sub-device specified in the Multi-Project file (see Figure 12 and Figure 13).

This is the main purpose of Multi-Project feature – to automate sequence of device operations for each chip in the multi-chip device.

The side effect of used approach is that device operation progress indicators are reset to 0 at beginning of each Sub-device operation. Therefore, it looks like progress-bar is jumping to 0 several times during the multi-chip device operation execution.

3. After programming (verifying, ...) of all Sub-devices is finished (or error occurs), standard Repeat dialog window is displayed (see Figure 14). Finished device can be removed from programmer socket and new device can be inserted. Pressing **Yes** button in Repeat dialog window or **YES!** button on programmer will start multi-chip device operation sequence again.

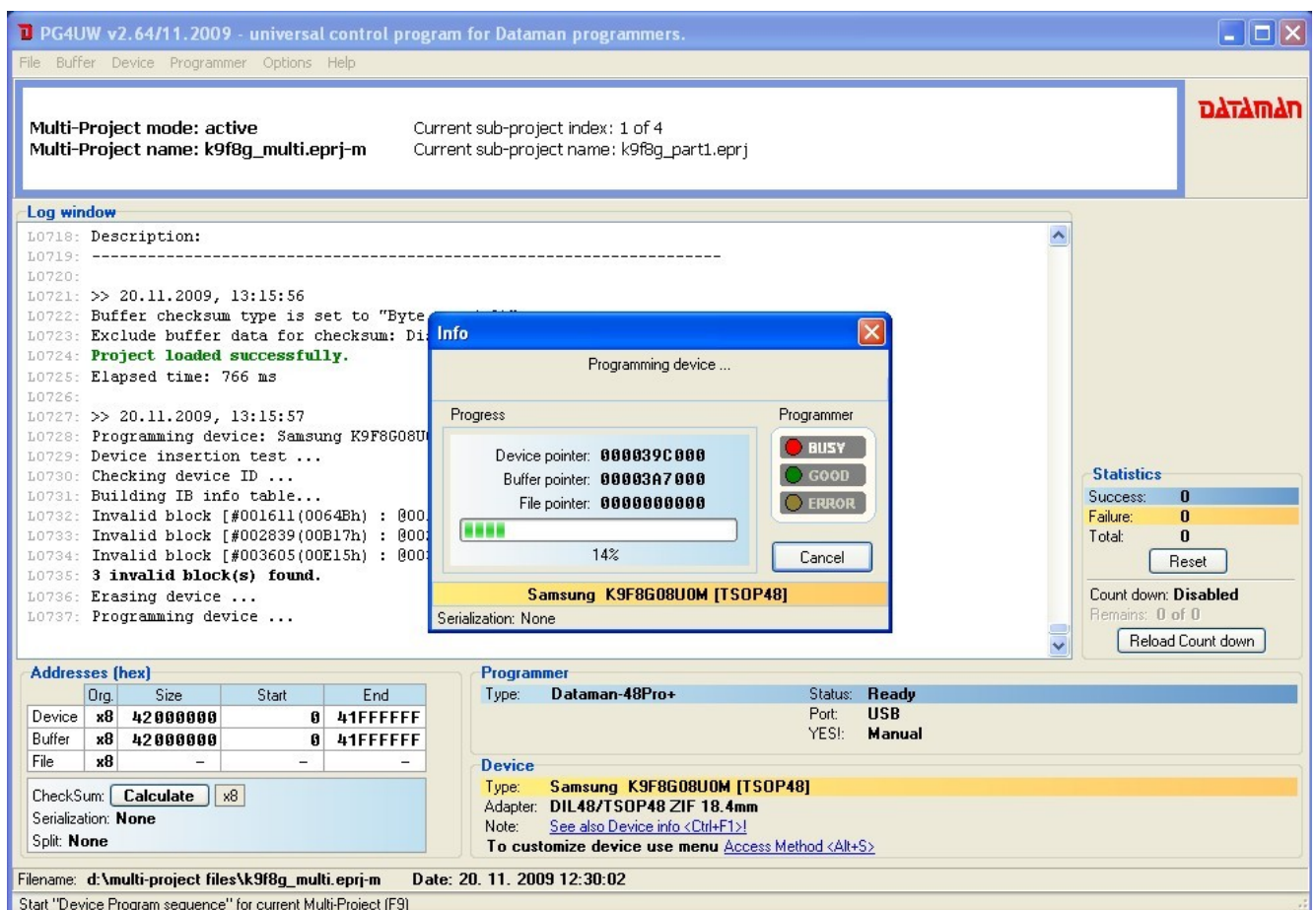


Figure 12: Processing the Multi-Project file - Project #1 of 4

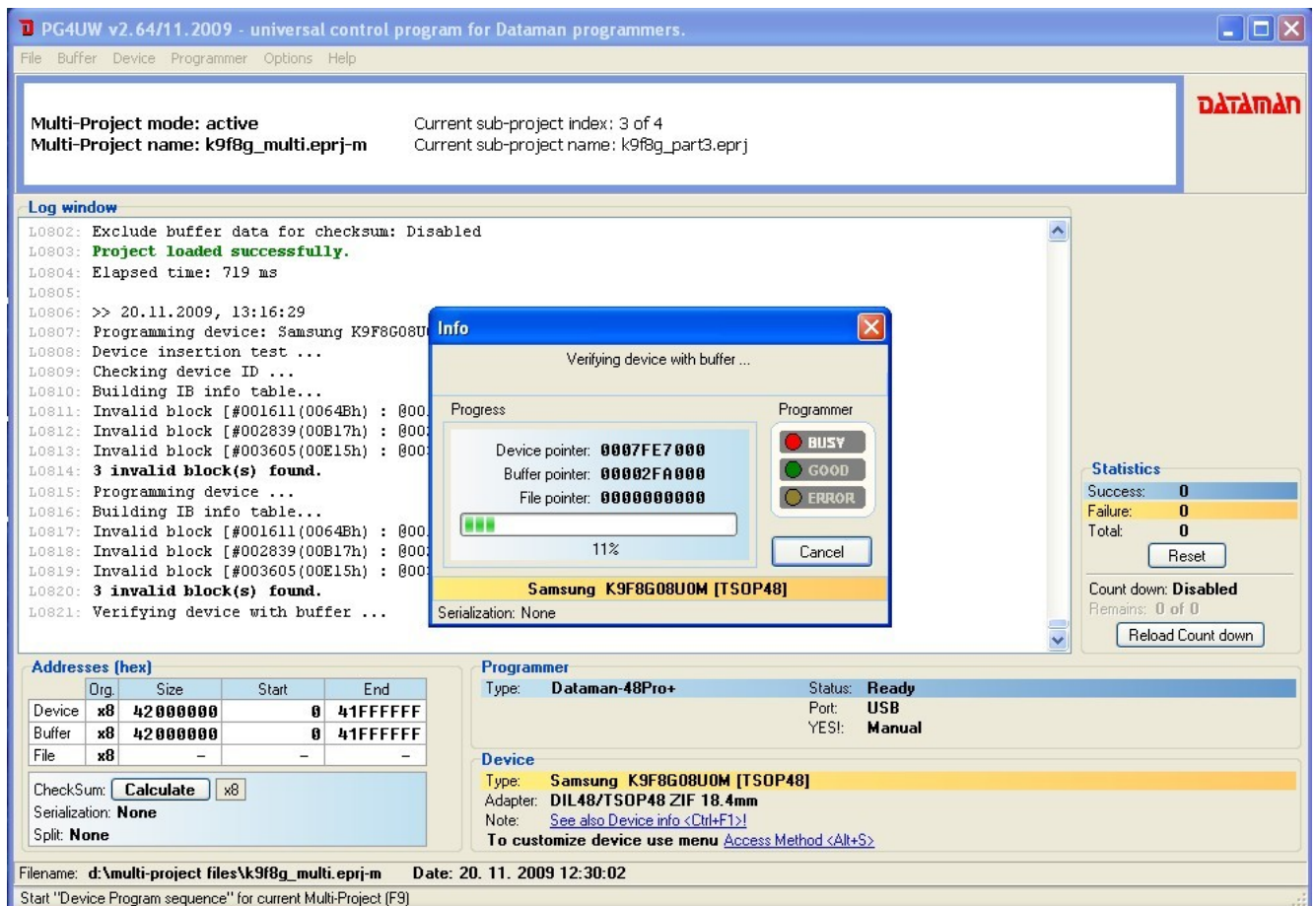


Figure 13: Processing the Multi-Project file - Project #3 of 4

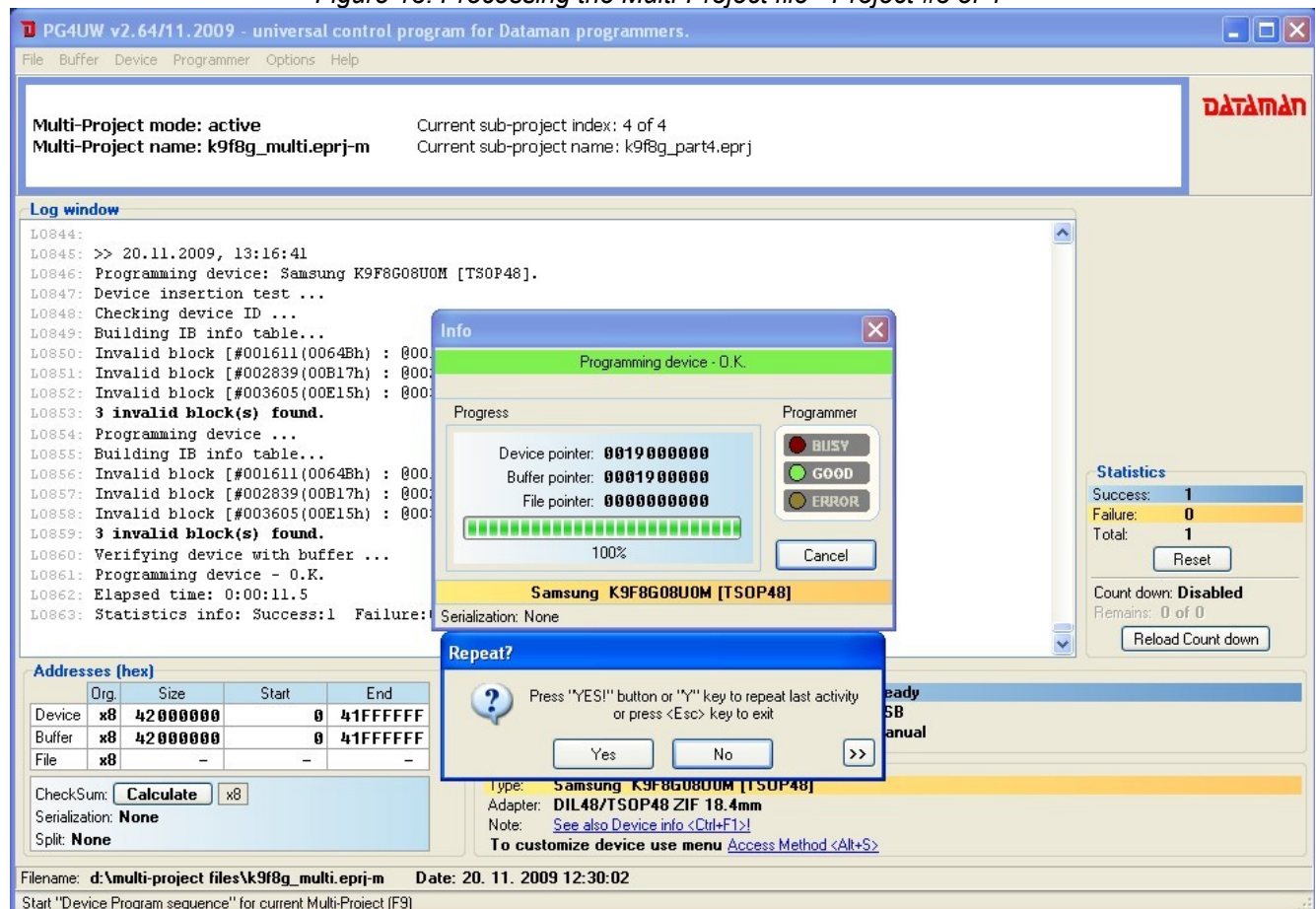


Figure 14: Processing the Multi-Project file - successful completion with Repeat dialog window



Note: If **Automatic YES!** feature is active, no Repeat dialog window is displayed after the completion of multi-chip device operation. Automatic YES! dialog window is displayed instead. The window shows the status of programmer socket and notices to remove finished device from programmer socket and insert the new one. After the new device is inserted, multi-chip device operation sequence will start automatically again. For more information on Automatic YES! feature consult your programmer manual or **Programmer | Automatic YES!** paragraph in Pg4uw help.

**For multi-programming using Pg4uwMC or stand-alone programmer:**

1. Load existing Multi-Project file using **Load project** menu.
2. Run desired multi-chip device operation using one of available device operation buttons (Blank, Verify, Program, Erase). Mostly the programming operation is used.

Selected device operation is executed as sequence of Project file loading and consequent Sub-device operation for each Sub-device specified in the Multi-Project file (see Figure 15).

This is the main purpose of Multi-Project feature – to automate sequence of device operations for each chip in multi-chip device.

The side effect of used approach is that device operation progress indicators are reset to 0 at beginning of each Sub-device operation. Therefore, it looks like progress-bar is jumping to 0 several times during the multi-chip device operation execution.

3. After programming (verifying, ...) of all Sub-devices is finished (or error occurs), information with result of device operation is displayed in Pg4uwMC. Finished device can be removed from programmer socket new device can be inserted. Pressing the operation button for the Site or **YES!** Button on the Site will start multi-chip device operation sequence again.

Note: If **Automatic YES!** feature is active, the multi-chip device operation sequence is automatically started again after removing the finished device from programmer socket and inserting the new one. For more information on Automatic YES! feature consult your programmer manual or **Programmer | Automatic YES!** paragraph in Pg4uwMC help.

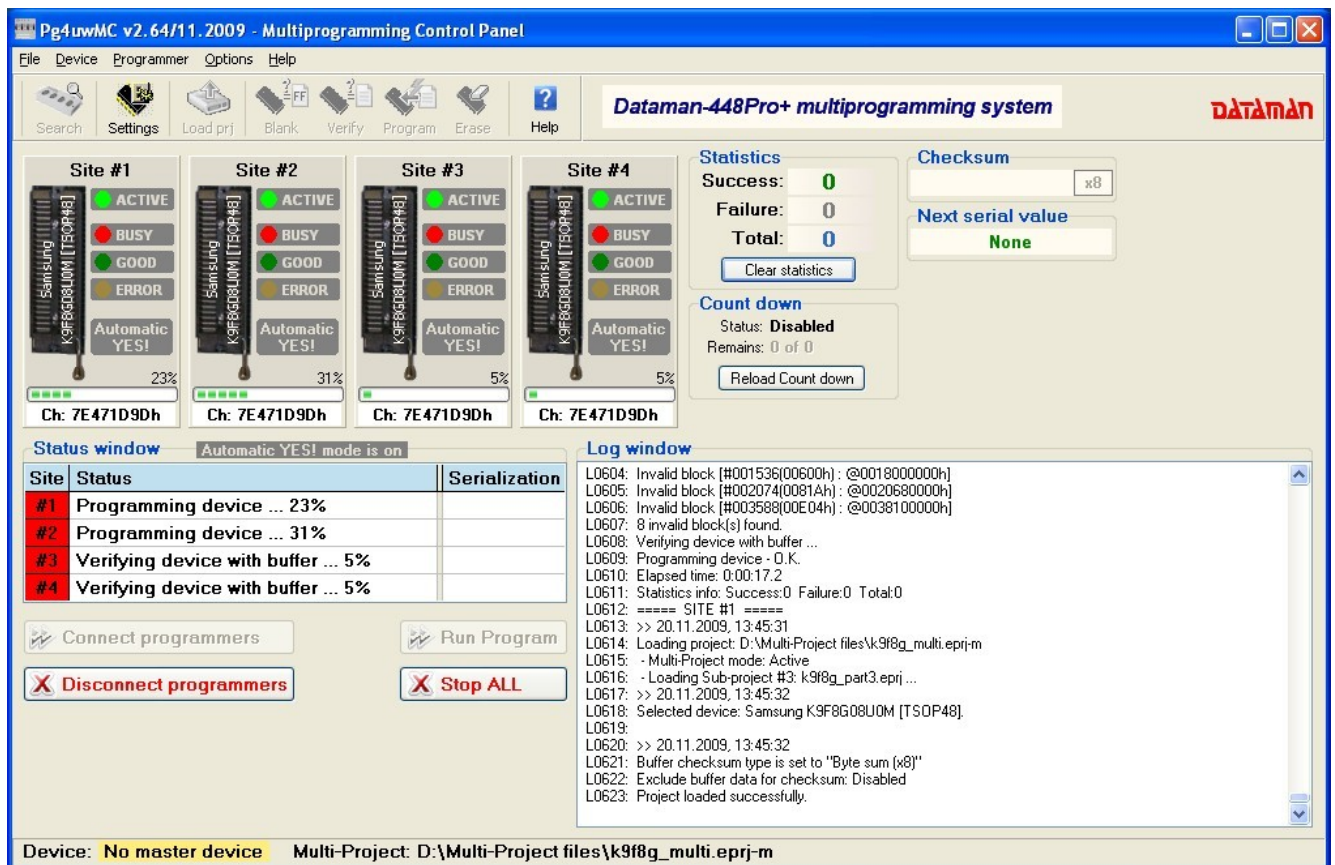


Figure 15: Processing the Multi-Project file in Pg4uwMC

## Limitations of Multi-Project feature

- serialization is not supported in multi-programming mode (only single programming supports serialization)
- count-down function is actually not supported
- actually, maximally 8 Project files in single Multi-Project file are supported

We are working on improving of the Multi-Project feature, so missing features will be supported in near future. Please, check for the newest version of control software and this application note.

## Using Multi-Project feature for various programming tasks

Multi-Project feature is very versatile tool. Its primary designation is to simplify the processing of multi-chip devices. However, it can be used for various other kinds of complex programming tasks. The following chapters will provide more details on this topic.

### *Using Multi-Project feature for programming multi-chip devices*

Imagine you have a multi-chip device that incorporates several memories. For example – NAND flash, NOR flash and PSRAM memories in single package. Generally, RAM memories are out of the scope, since they cannot be programmed permanently. So there are two independent chips remaining that need to be programmed – NAND a NOR flash memory. Having Multi-Project feature available, you can program both memories on single button click. Read further on how to proceed.

Notes:

There are several devices in our support with above mentioned multiply chips in single package. We will use TV0057A002CAGD [FBGA107] from Toshiba in this example.

#### **Step 1 – Defining the Project file for NAND part**

1. Firstly, display the Select device dialog window and list for the relevant Sub-device, i.e. for TV0057A002CAGD [FBGA107] (NAND) in this example (see Figure 16).
2. Select the device and configure all settings (Device operation option - <Alt+O> as well as Access Method - <Alt+S>). For more information about programming NAND flash memories using Dataman's universal device programmers consult our application note freely available from our web-site <http://www.Dataman.com>.
3. Load the data file into buffer.
4. Save the Project file for NAND flash part.
5. Test your Project file.

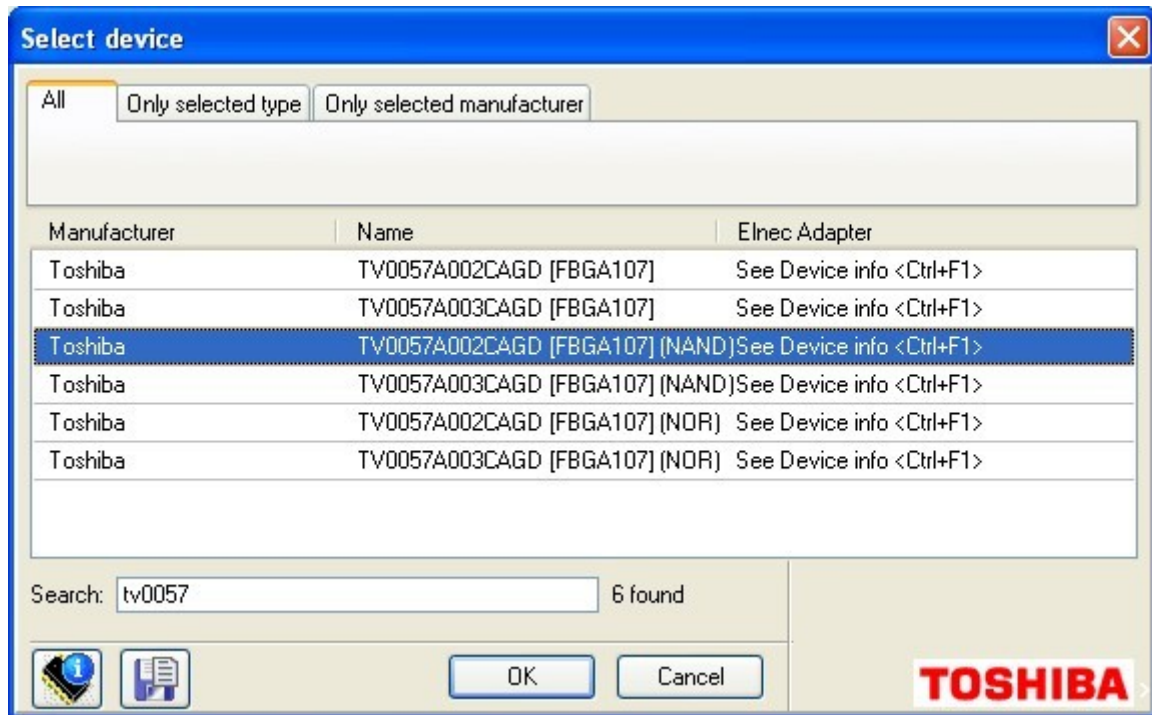


Figure 16: Selecting NAND part of multi-chip device

## Step 2 – Defining the project for NOR part

1. Then, display Select device dialog window again and list for other Sub-device, i.e. TV0057A002CAGD [FBGA107] (NOR) in this example (see Figure 17).
2. Select the device and configure all settings (Device operation options - <Alt+O> as well as Device settings - <Alt+S>).
3. Load the data file into buffer.
4. Save the Project file for NOR flash part.
5. Test your Project file.

In this way, configure the Project files for all Sub-devices included in desired multi-chip device. After doing so, you can finally build the Multi-Project file.

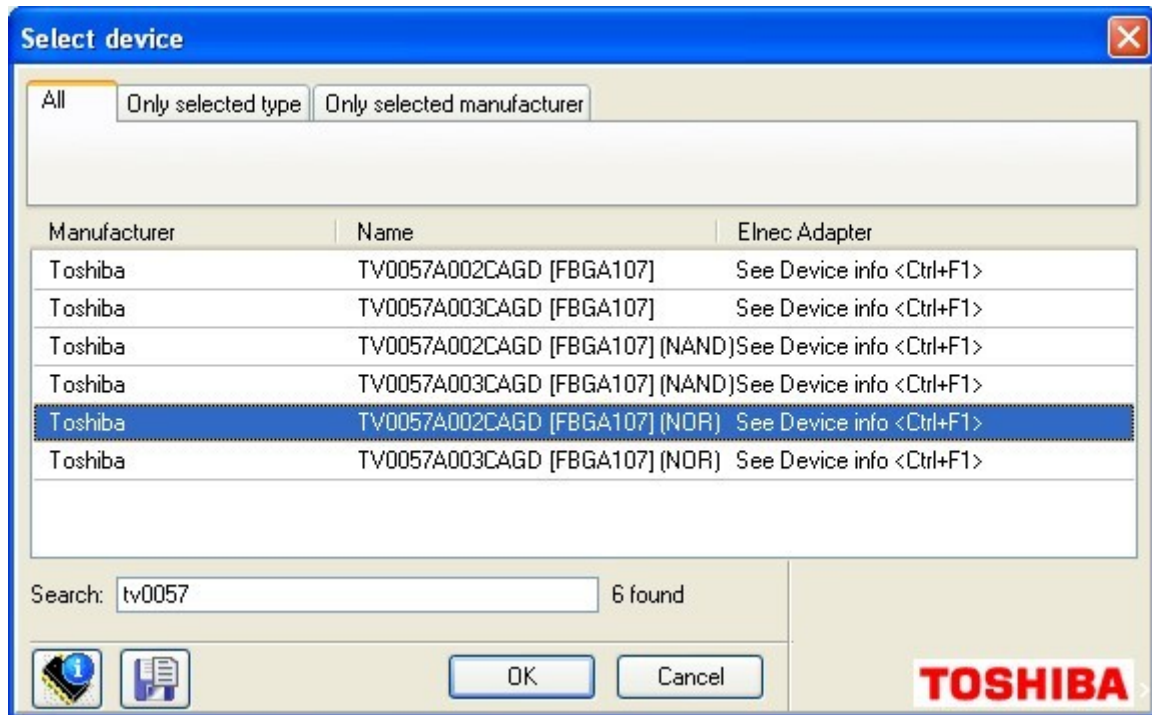


Figure 17: Selecting NOR part of multi-chip device

### Step 3 – Building the Multi-Project file

1. Having Project files available for all Sub-devices, you can finally build-up the Multi-Project file. Display Select device dialog window again and select the Master-Device, i.e. TV0057A002CAGD [FBGA107] in this example (see Figure 18). Empty Multi-Project Wizard window displays, see Figure 19.

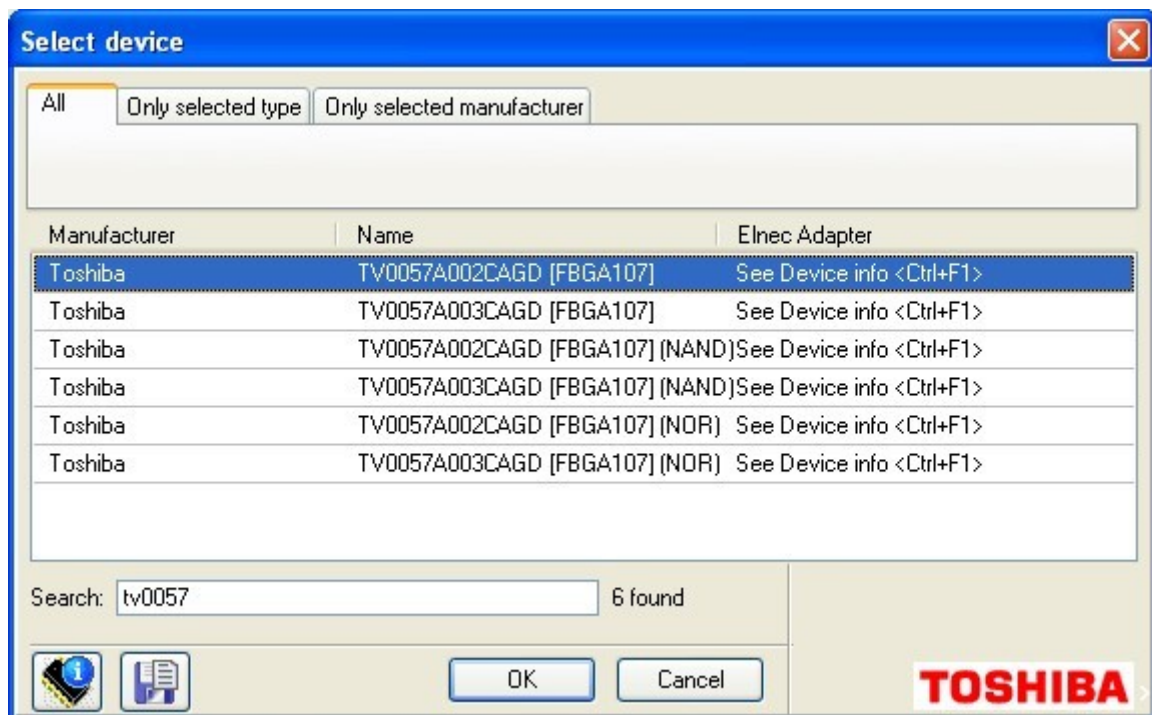


Figure 18: Selecting Master-device

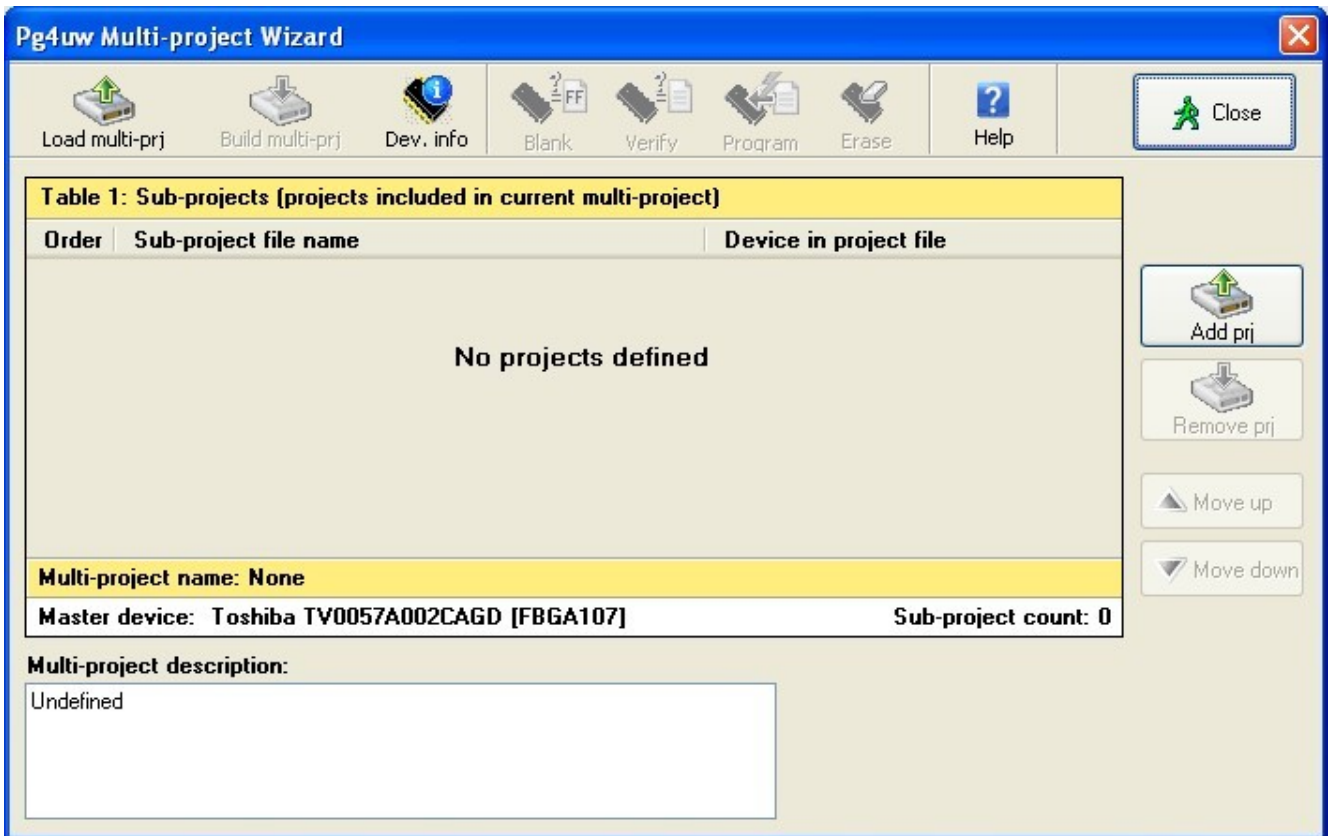


Figure 19: Empty Multi-Project Wizard window for Master-device

2. Add Project files saved in previous steps and build-up the Multi-Project file. The procedure was described in chapter **Building-up new Multi-Project file**.
3. Test your Multi-Project file.

#### Known limitations:

All general limitations are effective in this mode.

## ***Using Multi-Project feature for programming multiply partitions into NAND flash devices***

Imagine your project need to arrange various data into single NAND flash memory. For example – boot record image, operating system image and file system image. Each image need to be placed at exactly specified address / block of NAND flash memory. It is not such easy to guarantee the exact block mapping in NAND flash device, due to a possibility of invalid block(s) occurrence. Having Multi-Project feature available, your images can start at exact block on single button click. Read further on how to proceed.

### **Notes:**

This Multi-Project feature operation mode is primarily designated (but not limited to) for programming of multiply partitions into single NAND flash device.

If you need to program multiply partitions into NAND flash chip that stands for a Sub-device in multi-chip device, you can use this technique to save the Project files for NAND flash part. Then you can use common multi-chip approach described in previous chapter – just use multiply Project files for NAND part saved here instead of single Project file defined in previous chapter.

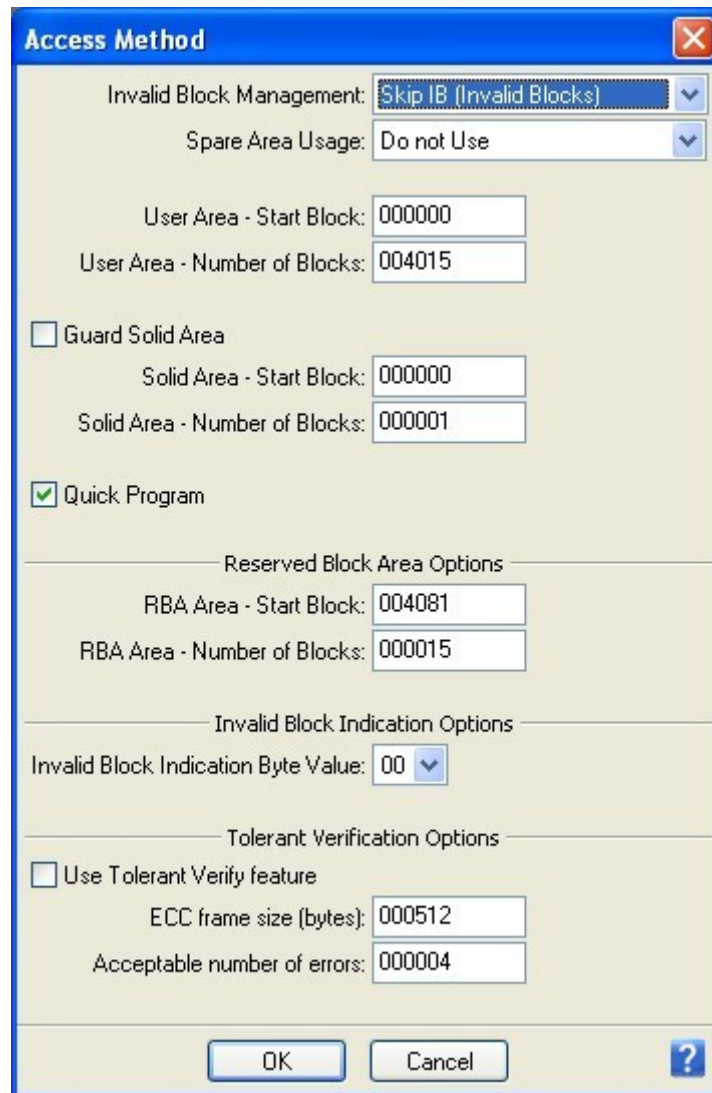
There are plenty of NAND flash devices in our support today. We will use K9F1208U0C [TSOP48] from Samsung in this example. The device has 512 Mbit / 64 Mbyte capacity and consists of 4096 blocks.

### **Step 1 – Defining the Project files for individual data images**

1. Firstly, display the Select device dialog window and list for desired NAND flash device, i.e. for K9F1208U0C [TSOP48] in this example. Select the device.
2. Display Access Method dialog window, key short-cut <Alt+S> (see Figure 20).
3. Set appropriate Invalid Block Management method, Spare Area Usage method, User Area – Start Block (equivalent to partition start) and User Area - Number of Blocks (equivalent to used partition size) and other options as necessary for the first data image. For more information about programming NAND flash memories using Dataman's universal device programmers consult our application note available at <http://www.Dataman.com>.
4. Display Device operation options dialog window, key short-cut <Alt+O>. Select the proper settings.
5. Save the Project file for the first data image / partition.

In this way, save the Project files for all your data images. Once all Project files are saved, you can build-up the Multi-Project file.





The 'Access Method' dialog window for a NAND flash device contains the following settings:

- Invalid Block Management:** Skip IB (Invalid Blocks)
- Spare Area Usage:** Do not Use
- User Area - Start Block:** 000000
- User Area - Number of Blocks:** 004015
- ☐ **Guard Solid Area**
  - Solid Area - Start Block:** 000000
  - Solid Area - Number of Blocks:** 000001
- ☒ **Quick Program**
- Reserved Block Area Options**
  - RBA Area - Start Block:** 004081
  - RBA Area - Number of Blocks:** 000015
- Invalid Block Indication Options**
  - Invalid Block Indication Byte Value:** 00
- Tolerant Verification Options**
  - ☐ **Use Tolerant Verify feature**
    - ECC frame size (bytes):** 000512
    - Acceptable number of errors:** 000004

Buttons: OK, Cancel, and a help icon (?) are at the bottom.

Figure 20: Access Method dialog window for NAND flash device

## Step 2 – Building-up the Multi-Project file

1. Having Project files available for all data images, you can finally build-up the Multi-Project file. Open the Multi-Project Wizard window (see Figure 2) using **Options | Multi-Project Wizard** Pg4uw menu command or key short-cut **<Ctrl-M>**, add individual Project files and build-up the Multi-Project file, as described in chapter **Building-up new Multi-Project file**.
2. Test your Multi-Project file.

If you need to program multiply partitions into NAND flash chip that stands for a Sub-device in multi-chip device, use individual Project files saved in Step 1 together with Project file for NOR part for building-up the final Multi-Project file. The Multi-Project file built here in Step 2 can be used for processing only the NAND part of multi-chip device.



**Known limitations:**

All general limitations are effective in this mode.

Special limitations for Access Method settings:

- At this time, there is none possibility to set the reserved partition size. You can set only the partition start (User Area – Start Block) and used partition size (User Area – Number of Blocks). As the result, if there are too many invalid blocks in partition, the relevant image data may be programmed into / after the block that is specified as User Area – Start Block for subsequent partition.

Workaround: This error will be found later, during the verification of subsequent partition. Make sure you have enabled option **Verify after programming** in Device operation options - <Alt+O> for all partitions.

- If **Spare Area Usage** setting is set to **User Data** for certain partition, a plenty of invalid blocks will be found by the programmer when processing all subsequent partitions. The only effect will be a long listing in control software log window. This will not affect the quality of the programming process for subsequent partitions, since invalid blocks before User Area – Start Blocks are ignored, anyway. None workaround is necessary.

Special limitations for Device operation options settings:

- Each time the **Erase before programming** option is enabled, a whole NAND flash device will be erased. In this way, data programmed for preceding partitions will be destroyed.

Workaround: Deal carefully with this option. Make sure that you have enabled **Erase before programming** only for the first Project file in sequence.

## Using Multi-Project feature for programming multiply daisy-chained JTAG devices

Imagine you have a data acquisition board that incorporates several (e.g. 8) low-cost microcontrollers, that act as intelligent A/D converters. If these microcontrollers are provided with JTAG interface, you can connect them into single chain. Probably, you have already did so because of debug purposes. Having Multi-Project feature available, you can program all your chained microcontrollers on single button click. Read further on how to proceed.

### Notes:

Please, test the feature with your board before you decide to start mass-production. This mode was not tested in a responsible manner due to a lack of suitable board. The maximum supported chain length is unknown. If you observe any problems using Multi-Project feature for programming multiply daisy-chained JTAG devices, please, contact your distributor. Sending your board to our lab may be necessary.

Although all microcontrollers used in following example are the same and use the same data file, this is not a limitation. From Multi-Project feature point of view, it is possible to combine any JTAG devices and an individual data image can be used for each device.

Actually, there are several ARM based microcontroller families from ST Microelectronics in our support, that can be used as target device if connected in JTAG daisy chain. They are differentiable by part name extension of (ISP-JTAG CHAIN). We will use STR712FR0 (ISP-JTAG CHAIN) in this example.

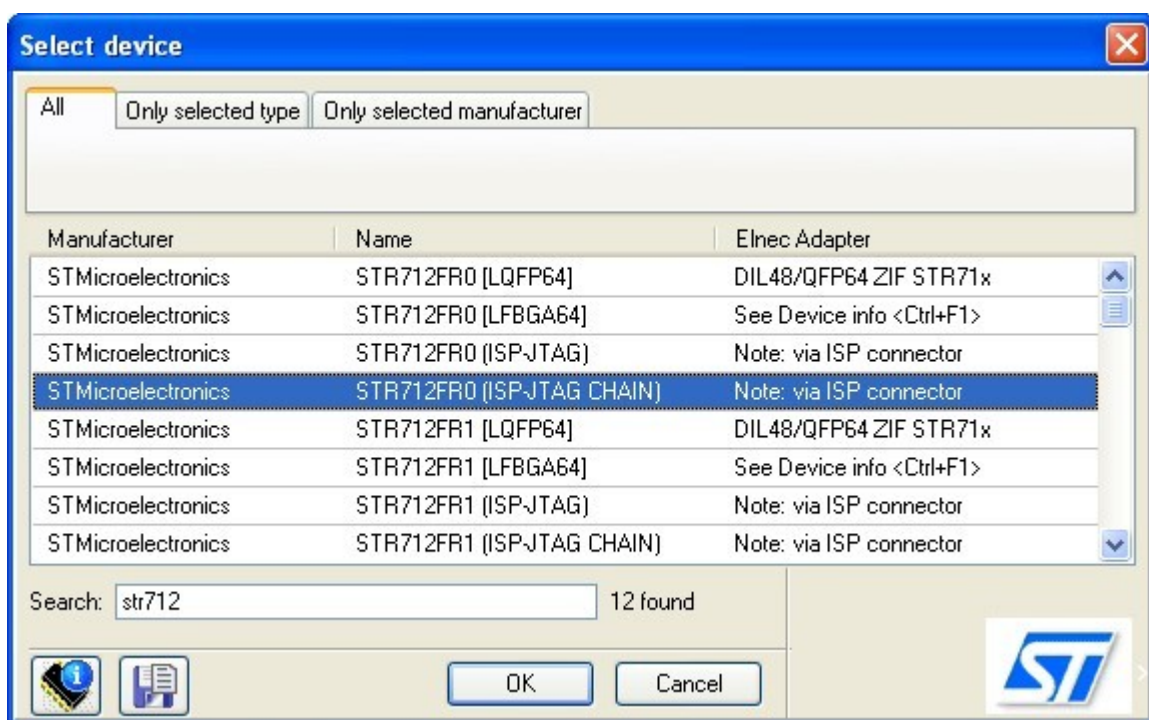


Figure 21: JTAG device selection - must be supported in ISP-JTAG CHAIN mode

**Step 1 – Defining the Project files for individual devices**

1. Firstly, display the Select device dialog window and list for desired JTAG device supported in ISP-JTAG CHAIN mode, i.e. for STR712FR0 (ISP-JTAG CHAIN) in this example (see Figure 21).
2. Display Device access configuration dialog window, key short-cut <Alt+S>. Set desired device configuration options.
3. Display Device operation options dialog window, key shortcut <Alt+O> (see Figure 22).

**Device operation options**

Insertion test

Device ID check error terminates the operation: **Enable**

Command execution

Erase before programming: **Disable**

Verify after reading: **Enable**

Verify after programming: **Enable**

Target system power supply parameters

☐ Enable target system power supply

Voltage (2000..6000 mV): 3300

Max. current (0..300 mA): 250

Voltage rise time (us): 10

Target supply settle time (us): 10000

Voltage fall time (us): 10

Power down time (us): 10000

Target system parameters

☒ Disable supply voltage test

Supply voltage (mV): 3300

☐ Keep ISP signals at defined level after operation

Inactive level of all ISP signals: **Pull-down**

JTAG Chain Parameters

Max. JTAG frequency (kHz): 1000

☒ Single device in chain

Number of devices before target: 0

Number of devices after target: 0

Sum of IR registers lengths of the devices before target: 0

Sum of IR registers lengths of the devices after target: 0

OK Cancel ?

Figure 22: Device operation options for device supported in ISP-JTAG CHAIN mode

The dialog consists of several sections of the settings, that you probably are familiarized with from your previous experience with in-circuit programming using Dataman's universal device programmers. The sections Insertion test, Command execution, Target system power supply parameters and Target system parameters are out of the scope of this application note. If you need more information about those, consult your programmer manual or Pg4uw Help.

4. The section JTAG Chain Parameters is critical for this mode of operation. There are several settings available:

**Max. JTAG frequency (kHz)** – determines maximum clock frequency on TCK pin allowed for use. It must comply with the slowest device in the chain. Consult the datasheets of connected devices for maximum allowed JTAG speed.

**Single device in chain** – if checked (default state), only one device (no chain) is considered and the following settings are ignored (equivalent to simple ISP-JTAG mode). Uncheck the check-box.

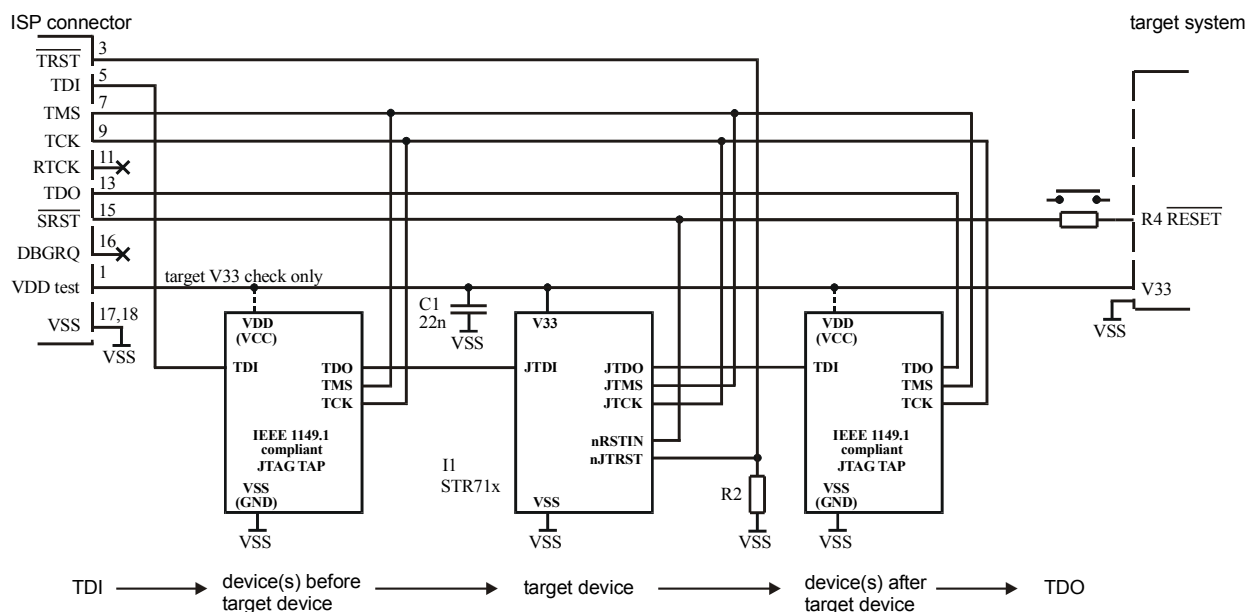


Figure 23: Example of JTAG chain connection

**Number of devices before target** – see Figure 23 for definition of the target device (STR712FR0 in this example) and devices before the target device (devices between the programmer and the target device TDI pin) and after the target device (devices between the target device TDO pin and the programmer). Specify the number of devices before the target device in this field.

**Number of devices after target** – similarly to previous field, specify the number of devices after the target device here.

**Sum of IR registers lengths of the devices before target** – each JTAG device has at least three registers – instruction register (IR), data register (DR) and bypass register (BYPASS). Typically, if device is not the target one, bypass instruction is written into IR. In consequence, the bypass register is

connected to JTAG chain instead of DR one. The length of bypass register is known and compulsory – 1 bit. Also the bypass instruction is exactly defined – all ones (1). In this way, all devices in chain other than the target one are simply switchable to bypass mode, so the programmer affects only the target device. Consult the datasheets of connected devices for IR lengths. Compute the sum of the IR lengths for devices that are connected before the target one and specify it in this field.

**Sum of IR registers lengths of the devices after target** – similarly to previous field, specify the sum of the IR lengths for devices that are connected after the target one here.

Note:

Some devices consist of several independent units that are chained internally – e.g. CPU core, flash memory, trace unit, etc. Although they are in single package, consider them being independent devices before / after the target device, having their own instruction registers, and reflect them in the chain settings. However, the target device should be considered as single device, internal units should not be included into the sums calculations.

5. Having all settings properly set, load the data image into buffer and save the Project file for this device.

In similar way, configure and save the Project files for all desired devices in the chain.

## Step 2 – Building-up the Multi-Project file

1. Having Project files available for all desired devices connected in the chain, you can finally build-up the Multi-Project file. Open the Multi-Project Wizard window (see Figure 2) using **Options | Multi-Project Wizard Pg4uw** menu command or key short-cut **<Ctrl+M>**, add individual Project files and build-up the Multi-Project file, as described in chapter **Building-up new Multi-Project file**.
2. Test your Multi-Project file.

### Known limitations:

All general limitations are effective in this mode.

### Special limitations:

- At this time, the maximum chain length for reliable operation is not specified due to a lack of suitable board. The maximum value estimated based on test results is 64 bits – represents the sum of lengths of all instruction registers connected in the chain (devices before / after the target device plus target device).

## Version history

Version 1.0 – July 2008

- initial release

**Place for your comments**

