BitCloud 1.11.0 Migration Guide

1 Introduction

This document was written to help port existing applications created on top of BitCloud version 1.10.0 for use with BitCloud version 1.11.0. Changes are listed below:

- IAR project files have been moved from `[application_root]/iar/[architecture]` to `[application_root]/iar_projects`.
- IAR and GCC configuration files (`iarConfiguration.h` and `Configuration` respectively) have been merged to a single one - `configuration.h`, which has been placed to the application root directory. Changes made in this file will affect both IAR and GCC projects.
- `main()` function has been moved to high-level application's source, to make debugging process easier.
- Each supported stack configuration (any sequence of device type, security type, MCU, RF-chip, MCU frequency and compiler type, which are supported by the current release) is provided now with the corresponding low-level Makefile, stored in `[application_root]/makefiles/[platform]/`.

High-level Makefile, placed in application root directory, operates as a simple switcher between supported stack configurations. And it is still possible to adjust application behavior by changing the `configuration.h` file content.
There are two ways to run your BitCloud 1.10.0 application in BitCloud 1.11.0 environment: quick project deployment in the new environment (described in Section 2) and complete project porting (described in Section 3). Both methods would take no more than 10 minutes to migrate. Although the first approach looks simpler it is recommended to port the project completely applying the method described in Section 3. This method consumes only few additional steps but allows the application to fit the same structure as all sample applications provided with the SDK, which will make it easier to maintain the application in the future.

.

2 Quick project deployment in the new environment

This chapter describes quick BitCloud 1.10.0 project deployment in BitCloud 1.11.0 environment. Note that the complete porting of your application is more preferable, because it will make porting of your applications to future BitCloud releases much easier, and would take only few more steps in comparison with this method.

All actions should be performed in `[release_root]/Applications/[application_root]` directory of BitCloud 1.11.0 release.

1. Copy your BitCloud 1.10.0 application catalog to BitCloud 1.11.0 `[release_root]/Applications/` directory.
2. Paste `main()` function implementation, to any of your high-level application source files:

```
/*************************************************************//**
  \brief Main - C program main start function

  \param none
  \return none
*****************************************************************/
int main(void)
{
    SYS_SysInit();

    for(;;)
    {
        SYS_RunTask();
    }
}
```

3. If you use AVR Studio or command line to build your applications:
    a. Modify include paths section in the application's Makefile adding the following paths:

```
INCLUDEDIRS += -I$(MAC_PHY_PATH)/MAC_ENV/include
INCLUDEDIRS += -I$(MAC_PHY_PATH)/MAC_HWI/include
INCLUDEDIRS += -I$(MAC_PHY_PATH)/MAC_HWD_PHY/include
INCLUDEDIRS += -I$(CS_PATH)/include/private
INCLUDEDIRS += -I$(PDS_PATH)/include/private
INCLUDEDIRS += -I$(ZDO_PATH)/include/private
INCLUDEDIRS += -I$(APS_PATH)/include/private
INCLUDEDIRS += -I$(NWK_PATH)/include/private
INCLUDEDIRS += -I$(ZCL_PATH)/include/private
```

    b. Replace the Linker directive in the application Makefile

```
$(LD) $(LIBDIRS) $(LINKER_FLAGS) $(OBJS)
$(PLATFORM_SPECIFIC_OBJECTS_WITH_PATH) $(LIBS) -o $@
$(LINKER_FLAGS_TAIL)
with
```

```
$(LD) $(LIBDIRS) $(LINKER_FLAGS) -Wl,-\( $(OBJS)
$(PLATFORM_SPECIFIC_OBJECTS_WITH_PATH) $(LIBS) -Wl,-\) -o $@
$(LINKER_FLAGS_TAIL)
```

4. If you use IAR Embedded Workbench IDE to build and run your application:
    a. Extend list of included directories in your IAR configuration file with the following:

```
$PROJ_DIR$/../../../../BitCloud/Components/MAC_PHY/MAC_ENV/include
$PROJ_DIR$/../../../../BitCloud/Components/MAC_PHY/MAC_HWI/include
$PROJ_DIR$/../../../../BitCloud/Components/MAC_PHY/MAC_HWD_PHY/incl
ude
$PROJ_DIR$/../../../../BitCloud/Components/ConfigServer/include/pri
vate
$PROJ_DIR$/../../../../BitCloud/Components/PersistDataServer/includ
e/private
$PROJ_DIR$/../../../../BitCloud/Components/ZDO/include/private
$PROJ_DIR$/../../../../BitCloud/Components/APS/include/private
$PROJ_DIR$/../../../../BitCloud/Components/NWK/include/private
$PROJ_DIR$/../../../../BitCloud/Components/ZCL/include/private
```

    b. Update the contents of `Components/ConfigServer/src` directory in your project tree with actual files, placed in `release_root]/BitCloud/Components/ConfigServer/src`.
    c. Rename `WdtInit[MCU]Iar.o` library (if one linked) to `WdtInit[MCU]_Iar.o`
    d. Copy `iarConfiguration.h` file to `[application_root]/include` directory and rename it to `configuration.h`.

3 Complete porting to the new environment

This chapter presents a detailed guide to the complete porting of BitCloud 1.10.0 application to the BitCloud 1.11.0 environment.

All actions should be performed in `[release_root]/Applications/[application_root]` directory of BitCloud 1.11.0 release.

1. Place the source and the header files of your BitCloud 1.10.0 high-level application to BitCloud 1.11.0 `[application_root]/src` and `[application_root]/include` directories respectively.
2. Paste `main()` function implementation to any of your high-level application source files:

```
/************************************************************//**
  \brief Main - C program main start function

  \param none
  \return none
**************************************************************/
int main(void)
{
    SYS_SysInit();

    for(;;)
```

```
    {
        SYS_RunTask();
    }
}
```

3. In the root directory of your application create `configuration.h` file. If your use IAR Embedded Workbrench you can just move `iarConfiguration.h` file from BitCloud 1.10.0 `[application_root]/iar` to the new application root directory, rename it to configuration.h and disable the `#include <MakerulesSelector.h>` directive by commenting or removing it.

   If you don't use IAR Embedded Workbrench the easiest way to obtain "configuration.h" file is to copy it from any sample application provided with the SDK (it is better to use that application that was a base for your custom application) and modify its content to fit your needs. Note, that setting values for ConfigServer parameters in makefile is no more supported. To change default CS parameter value you should specify it with `#define` directive in `configuration.h` file. For more details, refer to BitCloud User Guide.

4. If you use AVR Studio to build your applications:

   a. In the root directory of your application create `configuration.h` file. The easiest way to obtain `configuration.h` file is to copy it from any sample application provided with the SDK (it is better to use that application that was a base for your custom application) and modify its content to fit your needs. Note, that setting values for ConfigServer parameters in makefile is no more supported. To change default CS parameter value you should specify it with `#define` directive in `configuration.h` file. For more details, refer to BitCloud User Guide.

   b. Copy the `makefiles/` directory from that application included in the new version of the BitCloud SDK that was used to create your custom application. The directory contains low-level makefiles corresponding to different available application configuration.

   c. Edit makefiles that fit your hardware configuration. Check that `APP_NAME` variable has a correct value. It should correspond to the desired firmware image files to be generated when building the application and should equal `ProjectName` parameter from AVR Studio project configuration file (.aps). Ensure also that `ObjectFile` parameter from the same .aps file equals the `APP_NAME` value followed by the file extension (.elf).

   d. Copy a high-level Makefile from the application root directory of any application included in the SDK to the root directory of your application. Edit the file: specify appropriate `PROJECT_NAME` and `CONFIG_NAME`. The first parameter is the name of the subdirectory of Makefiles directory of your application, which contains target low-level makefile. And `CONFIG_NAME` is exactly that makefile name without a "Makefile_" prefix.

   e. Open the project in AVR Studio. Open "Edit configuration options" window, check "Use external makefile" and select the high-level makefile that you created in the previous section

5. If you use IAR Embedded Workbench IDE to build your applications:

   a. In the root directory of your application create `configuration.h` file. Move `iarConfiguration.h` file from BitCloud 1.10.0 `[application_root]/iar` to the new application root directory,

rename it to `configuration.h` and disable the `#include <MakerulesSelector.h>` directive by commenting or removing it.

b. Move IAR project files from BitCloud 1.10.0 `[application_root]/iar` directory to BitCloud 1.11.0 `[application_root]/iar_projects` directory.

c. Open the .ewp file, corresponding to the required configuration for editing and change the relative paths, started with

```
$PROJ_DIR$/../../
```

to

```
$PROJ_DIR$/../
```

This operation shifts the `$PROJ_DIR$` directory one level higher in your OS directory tree.

d. Extend the list of included directories in the .ewp configuration file with the following:

```
$PROJ_DIR$/..
$PROJ_DIR$/../../../BitCloud/Components/MAC_PHY/MAC_ENV/include
$PROJ_DIR$/../../../BitCloud/Components/MAC_PHY/MAC_HWI/include
$PROJ_DIR$/../../../BitCloud/Components/MAC_PHY/MAC_HWD_PHY/include
$PROJ_DIR$/../../../BitCloud/Components/ConfigServer/include/private
$PROJ_DIR$/../../../BitCloud/Components/PersistDataServer/include/private
$PROJ_DIR$/../../../BitCloud/Components/ZDO/include/private
$PROJ_DIR$/../../../BitCloud/Components/APS/include/private
$PROJ_DIR$/../../../BitCloud/Components/NWK/include/private
$PROJ_DIR$/../../../BitCloud/Components/ZCL/include/private
```

e. Change the pre-included header file from

```
$PROJ_DIR$/../iarConfiguration.h
```

to corresponding `Makerules*.h` file, located in `[release_root]/BitCloud/lib/` and named along with the following scheme

```
Makerules[stack_type]_[device_type]_[security_type]_[MCU]_[RF-chip]_[compiler].h
```

Here are the examples of possible `Makerules*.h` filenames:

```
MakerulesBc_Coordinator_Sec_Atmega1281_Rf230_Iar.h
MakerulesZcl_All_HighSec_At91sam3s4c_Rf230B_Iar.h
MakerulesZcl_All_Atmega128rfa1_Atmega128rfa1_Gcc.h
```

f. Rename WdtInit[MCU][compiler].o library (if one linked) to WdtInit[MCU]_[compiler].o

g. Update the contents of `Components/ConfigServer/src` directory in your project tree with actual files, placed in `[release_root]/BitCloud/Components/ConfigServer/src`.