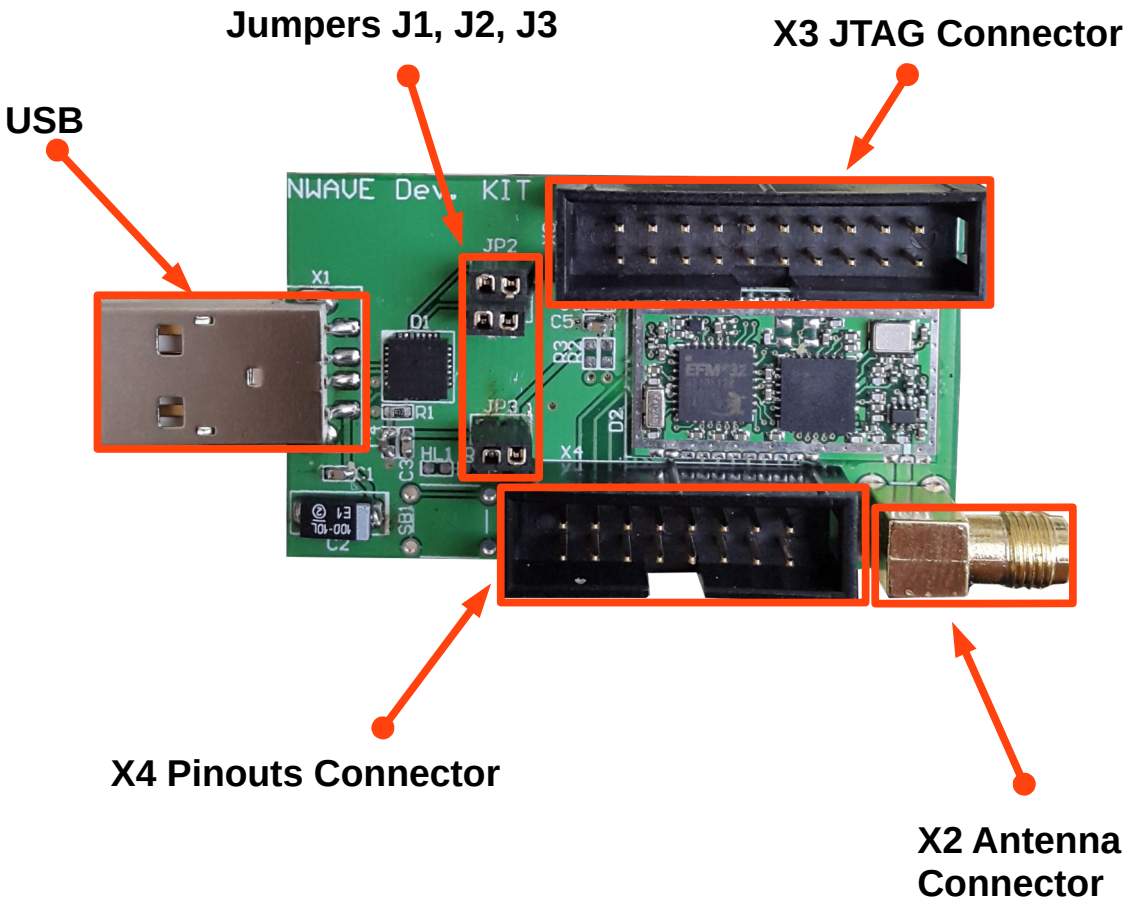
NWAVE DEVELOPMENT KIT V3
USER MANUAL

Table of Contents

1. Main features of this Development Kit

This Development Kit is purposed to give user a capability to evaluate and use RM3 radio module and a software library in a variety of different applications. It is typically used in such application field as telemetry, but can find a wider applying, e.g. radio networks with ultra low power consumption.



- Easy USB connection to PC or direct UART interface
- AT-commands set via UART / USB-Serial to work in the modem mode
- Capability to embed up to 128 KB* of user firmware on MCU flash including communication API supplied as a library
- Access to all Cortex-M3 features and the EFM32G210F128 or EFM32TG210F32 MCU peripherals
- Programming templates and examples of simple telemetry applications
- Support for ARM Cortex Microcontroller Software Interface Standard (CMSIS)

* The firmware code size can be up to 128 KB according to this MCU flash size and NWAVE library size is about 9 Kb only. There is a limitation of code size caused by IAR compiler free license which is 32Kb as maximum of compilable code. The full license has no such a limitation and can be purchased from IAR. However, there is no code size limitation when project is built in Simplicity Studio using GNU C Compiler (*gcc*). The *gcc* is not limited by code size. In the current version of the Development Kit user package both IAR EW and Simplicity Studio with *gcc* are supported.

A bootloader can be flashed to the device in order to allow user to program the device via serial port without Jlink adapter. When the bootloader is used it occupies 8 Kbytes of MCU flash.

2. Getting Started

Here is a Quick Start instruction describing how to send Nwave messages.

2. 1. IAR Embedded Workbench (IAR EW) Installation

For getting an evaluation software follow the next link:

http://supp.iar.com/Download/SW/?item=EWARM-EVAL

After starting IAR EW you will be offered to select a license type: *a 30-day time limitation* or *a 32Kbytes code size limitation*.

2. 2. Simplicity Studio installation

Download Simplicity Studio from the next link:

http://www.silabs.com/products/mcu/Pages/simplicity-studio.aspx

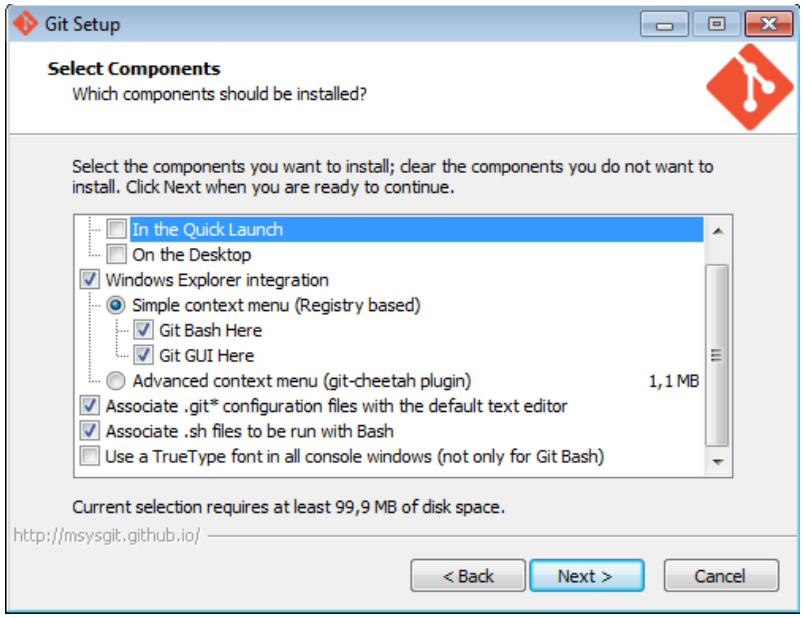
After installation and first start let the software complete the updates downloading.

2. 3. Getting Git

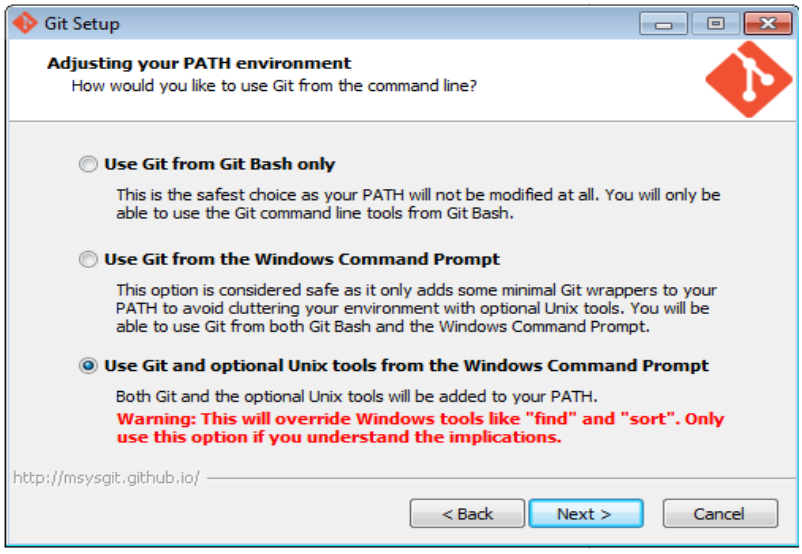
http://git-scm.com/download/win

Run the installation.

Tick *Git Bash Here* and *Git GUI Here*.



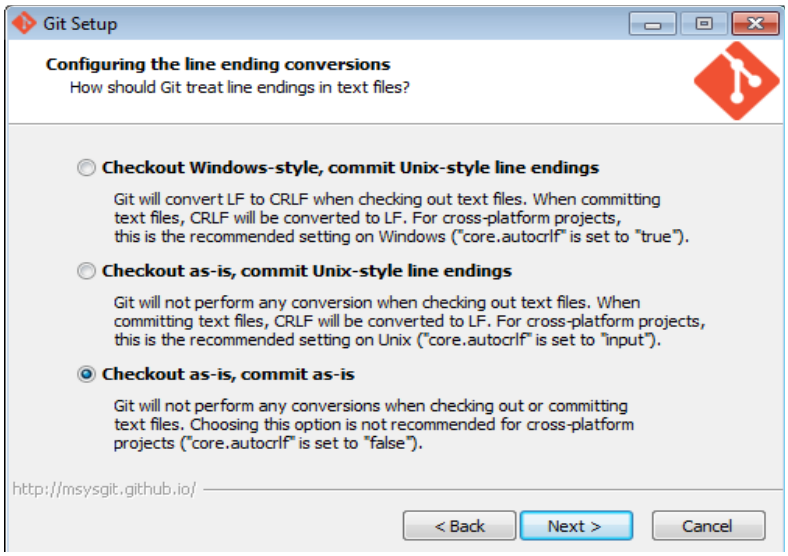
Select the command line option.



Note: if you select the last option it is going to override Windows tools like “find”

and “sort”

Select the line ending conversions.

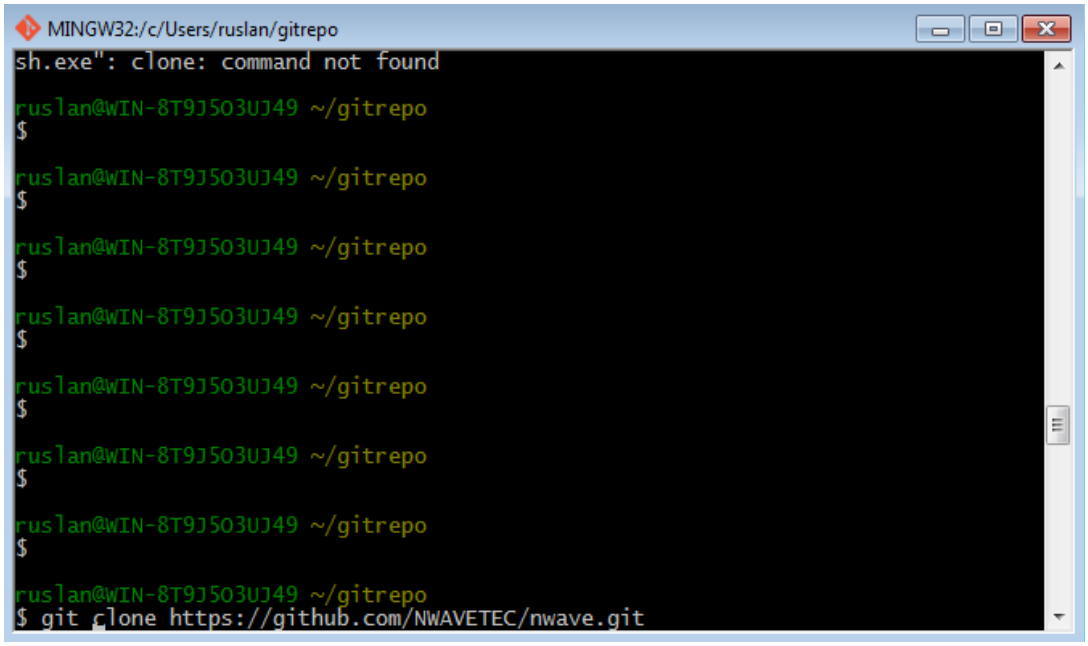


2. 4. Getting Nwave example for IAE EW from Git repository

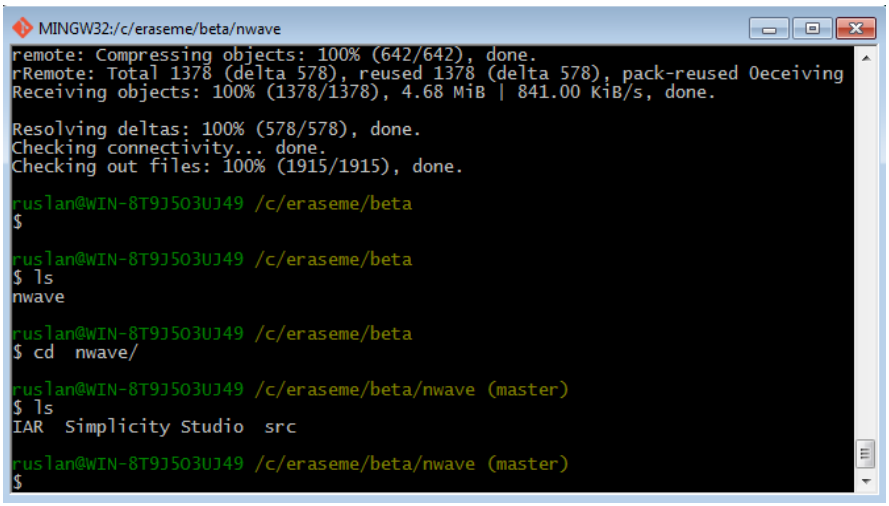
Git has two interfaces: Command Line and GUI.

Getting a repository from the Command Line Interface run *Git Bash*

Type: ***`git clone https://github.com/nwavetec/nwave.git`***

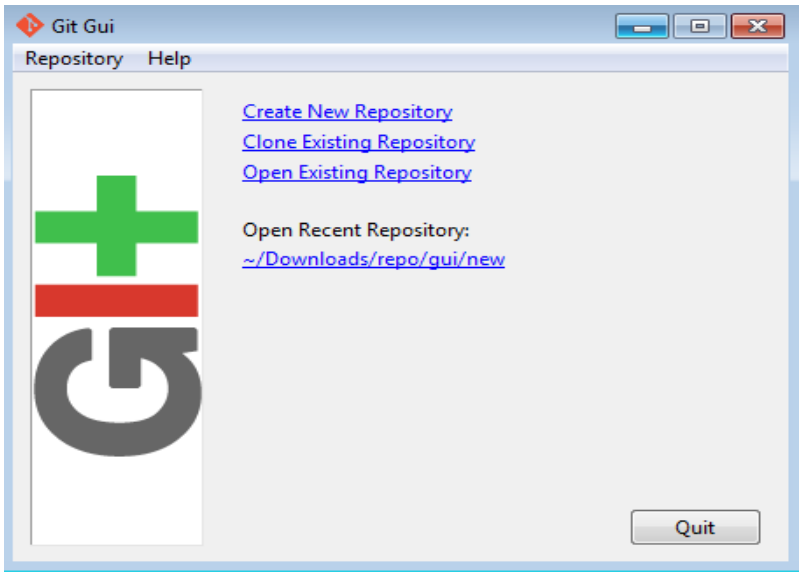


Check the content of this directory (run **`ls`**) it should show files:

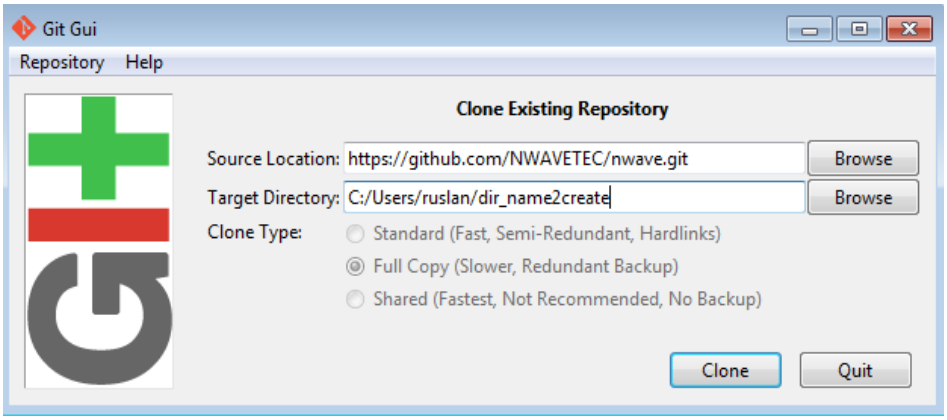


Getting a repository form the GUI run *Git GUI.*

At the first start select *Clone Existing Repository*

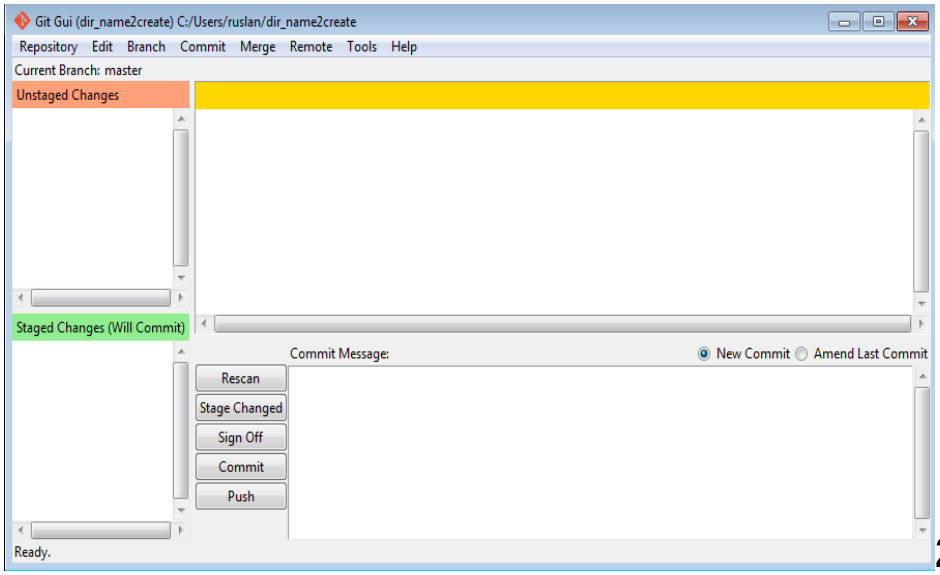


Type in the repository name and the directory name which will be created after this step.



Press *Clone* and it will show the process of getting a repository.

Then it finishes with showing this window below, now the project files are placed in the given directory:



2. 5. Open Workspace in IAR EW

Open a workspace with file ***nwave_rm3_templates.eww*** which is placed at the

top level of the cloned ***nwave*** directory.

2. 6. Open Workspace in Simplicity Studio

Run Simplicity IDE and click menu *File->Switch to Workspace->other* and select subfolder called *Simplicity Studio* under clonned *nwave* folder.

2. 7. Open ***main.c*** file and select an example

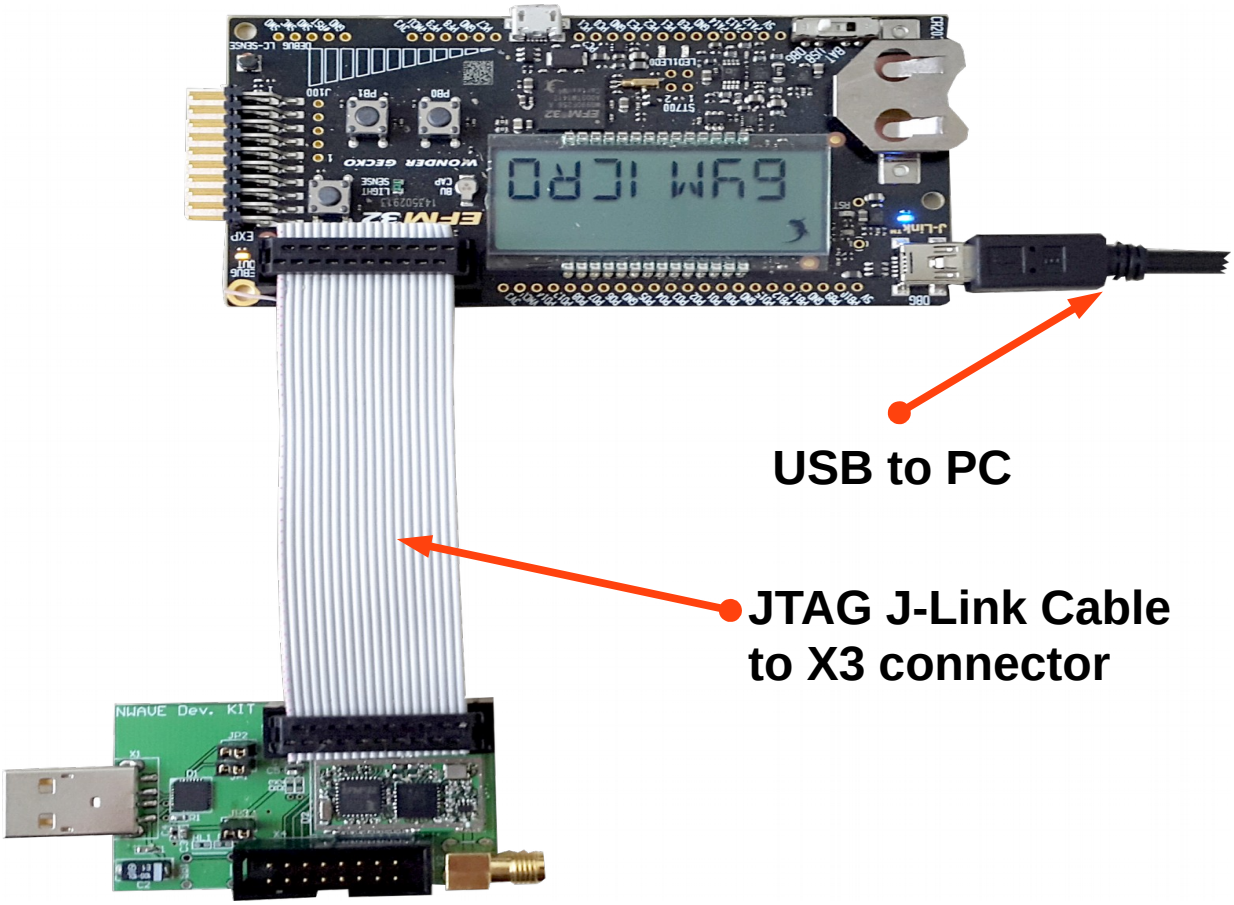
Select an example by editing this string:

```
#define EXAMPLE_CODE  UART_2_RM
```

UART_2_RM means to include a simple example code which demonstrates how to use Nwave API, in particular sending upcoming to UART data directly to the radio module transmitter.

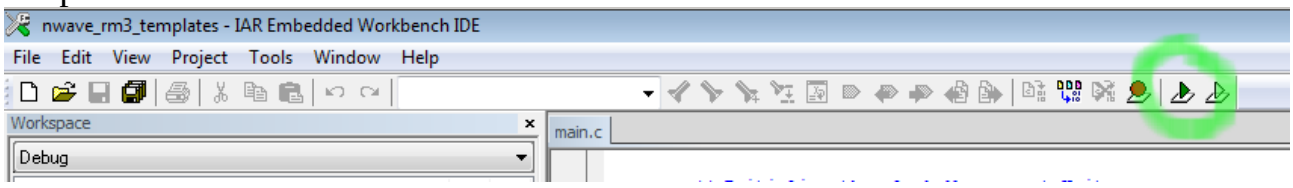
AT_PARSER means to include an example code of the parser of AT-commands which allows user to operate with the transmitter by AT-commands. The parser listens to the specific AT-commands on the serial port. These commands are described below in AT-Commands section.

2. 8. Connection J-Link programmer to the Development Kit



2. 9. Compile and Flash the device in IAR EW

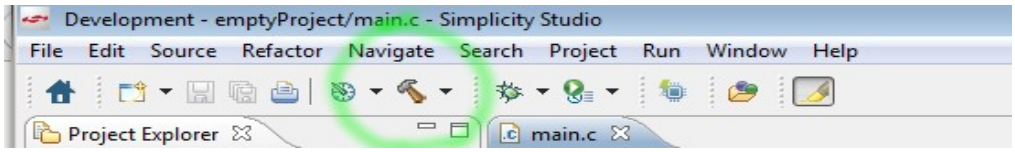
In open project press F7 to compile it or CTRL+D to compile, flash and debug the device *. The same action can be performed via *Download and Debug* button, see at the picture:



* To *Download and Debug* in hardware you need to connect JTAG J-Link adapter (or any starter kit, for instance, STK3200 or STK3300) to the X3 connector. Otherwise it is possible to trace the firmware code in *Simulator* which can be alternatively chosen in *Debug* option of the project. By default it is set to JTAG J-Link adapter.

2. 10. Compile project in Simplicity Studio

Having a project open click at the *build* button to compile as it is shown at the picture below/



2. 11. Create a connection in serial terminal programme

Connect the Development Kit to the USB port of your computer. The Development Kit has a USB-to-Serial converter so it should appear as an additional serial port in the device list on your system.

In your terminal programme* create a new connection, select this new serial port and and set the parameters for the connection as below:

Baud rate: 9600
Data bits: 8
Stop bits: 1
Parity: none
Handshake(Flow Control) Hardware: none
Handshake(Flow Control) Software: none

* Tera Term or Putty can be used as terminal programmes free of charge.

2. 12. Sending Nwave messages

The API of Nwave Library is described in the section Nwave Library API.

Depending on selected example code in *main.c* file by defines *AT_PARSER* or *UART_2_RM* user can input either AT-Commands or data bytes directly to the radio transmitter. Both examples demonstrate using an API of *nwave_unb* library and send Nwave messages.

Note: In *AT_PARSER* **example user basically needs to set c*arrying frequency*** **and *bandwidth* in Hz by** *«AT+FREQ=...»* **command and send data by** *«AT+SEND=...»* **command. In** *UART_2_RM* **example user needs to care about setting a proper value** *carrying frequency* **by calling** *NWAVE_Set_Frequency(...)* **function (see section Nwave Library API).**

2. 13. User Application

The provided source code is a template which allows simply embed a user application. Its structure is organized as two functions *user_setup()* and *user_loop()* called from main.

The *user_setup()* is supposed to include all initialization and it is called once.

The *user_loop()* is supposed to contain user application load.

An example code given below shows how simple transmission of user data can be done. In this example each 4 bytes coming to UART are sent to the radio module transmitter.

```
void uart_2_rm (void)
{
  int watch_c;
  while ((watch_c = NWRM_UART_GetChar()) >= 0) {
  send_buf_len.send_buffer[i++] = watch_c;
   if (i >= 4) {
     unsigned char packetRec[256];
#warning "Set a proper frequency value for your environment"
     /* NWAVE_Set_Frequency(866500000, 50000, 100); */
     NWAVE_Set_Frequency(868800000, 50000, 100);
     NWAVE_send(send_buf_len.send_buffer, 4 /*send_buf_len.len*/ ,
    packetRec, PROTOCOL_B);
     tfp_printf("Data sent.\n");
     i = 0;
   }
  }
}
void user_loop (void)
{
  uart_2_rm();
 }
int main(void)
{
```

```
  user_setup();
  while (1) {
    NWRM_RTC_Sleep();
    user_loop();
  }
}
```

3. AT-Commands

3. 1. Setting carrying frequency and bandwidth

Syntax:

«AT+FREQ=[Carrying_frequency],[Bandwidth]»

Carrying_frequency - central carrying frequency in Hz

Bandwidth - half-range which is added to and subtracted from central carrier frequency giving a full range in a result.

Example: *AT+FREQ=868000000,25000*

If success it returns «Ok», if failed - «Error»

3. 2. Sending messages by Nwave protocol

Syntax:

«AT+SEND=[&hex_byte1][&hex_byte2]..[[&hex_byteN]]»

hex_byte - hexadecimal representation of the byte. Before each hexadecimal byte the symbol «$» is placed.

Example: *AT+SEND=$23$0A$5D$5F*

If success it returns «Ok», if failed - «Error»

3. 3. Getting a serial number

Syntax:

«AT+SERIAL=?»

Example: *AT+SERIAL=?*

If success it returns a serial number of the device, e.g. *«0000F556»*, if failed - «Error»

4. Nwave Library API

The Nwave Library is provided as a compiled library and it is stored in the *nwave_unb.lib* file. It is linked to the project. The *unb.h* is a header file with definitions for C functions of the library.
The Nwave library supports two protocols: Protocol-B for only sending messages to base station and Protocol-E for sending and receiving messages from base station.
The *unb.h* contains next function definitions:

4. 1. Setting carrying frequency and bandwidth

```
int NWAVE_Frequency_Set(unsigned long carrying_frequency,
   unsigned long bandwidth, int channels);
```

carrying_frequency – frequency of the low end of a range
bandwidth – frequency of the high end of a range
channels – channels number

4. 2. Sending messages by Nwave protocol

```
unsigned char NWAVE_send(unsigned char *packet, unsigned
   char length, unsigned char* packetRec, unsigned char
   protocol);
```

packet – pointer to the buffer for transmission
length – lengeh of the transmitted buffer
packetRec – pointer to the reading back buffer of Protocol-E; a format of data structure is described below
protocol – protocol type, is chosen by defined constants*:* *PROTOCOL_B* or *PROTOCOL_E*

Return: 0 is always returned when Protocol-B is used; *Number* of received bytes returned or *0* (if nothing received) when Protocol-E is used.

Here is a data structure format of reading back buffer of Protocol-E:
Modem Identifier: 3 Bytes
Message Identifier: 1 Byte
Message Body: 124 Bytes

NOTE: The *packetRec* is recommened to be a char array of at least 128 Bytes. This char array is for receiving base station reply (when Protocol-E is used).

NOTE: When Protocol-E is used this function waits as maximum for one minute to receive data, if data is not received during this timeout then the function returns *0* (to return variable, not to the buffer). If it happens it is recommended to send again after some delay and increase the delay each time user re-sends until a reply is received.

NOTE: User should not interrupt the execution of this function for more than 200us.

4. 3. Receiving messages by Nwave protocol

Receiving happens when Protocol-E is used. User can send data and receive the answer from base station using ***NWAVE_send(…)*** function described in 4.2

5. Peripheral Devices Sharing

On this Development Kit the *USART 0* is used for communication with the radio device.

The CP2102 is the USB to Serial converter which is connected to the Low Energy UART of the MCU. It also converts 5V level of USB to the 3.3V level for the board power supply. This power supply line can be disconnected by removing *J3* jumper. The *RX* and *TX* lines can be disconnected by removing *J1, J2* jumpers.

Table 1 shows a pinout assignments on X4.

Table 1. The Device Pinout on X4

X4 Pin # and Name	Pin Functionality Alternatives	MCU Pin # and

			Name	
Pin #	Pin Name		Pin #	Pin Name
1	GND			
2	VDD			
3	PC0	ACMP0_CH0, PCNT0_S0IN, US1_TX	5	PC0
4	PA1	TIM0_CC1, I2C0_SCL, CMU_OUT1	2	PA1
5	PA0	TIM0_CC0, I2C0_SDA	1	PA0
6	PF1	DBG_SWDIO, LETIM0_OUT1	26	PF1
7	PF0	DBG_SWCLK, LETIM0_OUT0	25	PF0
8	PC14	ACMP1_CH6, TIM0_CDTI1, TIM1_CC1, PCNT0_S1IN	23	PC14
9	PC1	ACMP0_CH1, PCNT0_S1IN, US1_RX0	6	PC1
10	PB11	DAC0_OUT0, LETIM0_OUT0	10	PB11
11	PB13	HFXTAL_P, LEU0_TX	12	PB13
12	PC15	ACMP1_CH7, DBG_SWV, TIM0_CDTI2, TIM1_CC2	24	PC15
13	PD4	ADC0_CH4, LEU0_TX	16	PD4
14	PD5	ADC0_CH5, LEU0_RX	17	PD5
15	PD6	ADC0_CH6, LETIM0_OUT0, I2C0_SDA	18	PD6
16	PD7	ADC0_CH7, LETIM0_OUT1, I2C0_SCL	19	PD7

Table 2 shows a pinout assignments on X3.

Table 2. The Device Pinout on X3

X3 Pin # and Name		Pin Functionality Alternatives
Pin #	Pin Name	
1	VDD	
2	GND	
3		
4	GND	
5		

6	GND	
7	DB2	DBG_SWDIO, LETIM0_OUT1
8	GND	
9	DB3	DBG_SWCLK, LETIM0_OUT0
10	GND	
11		
12	GND	
13		
14	GND	
15	RST	
16	GND	
17		
18		
19		
20	GND	

6. Bootloader

A device that you use may be prepared with a bootloader which means it was flashed to the MCU. The bootloader allows user to program the device via serial port (in fact, USB-Serial convertor) without J-Link adapter. When the bootloader is flashed to the device it occupies 8 Kbytes of MCU flash.

To flash the device by means of the bootloader you need to use an Nwave loader tool on the computer side. The Nwave loader is a part of entire bootloader git repository. To get it do next:

Run *Git Bash*

Type ***`git clone https://github.com/nwavetec/boot.git`***

6. 1. Loading a firmware by means of the bootloader

Plug the developmint kit into the USB port of computer or connect it with the USB extension cable. After device power cycle happens there is a timeout interval of 2 seconds within which the entering to the bootloader mode is available. To do this user should run the Nwave loader tool right after device started during the specified time

period (2s). The usage of the command for this is like:

winloader.exe firmware.bin portnumber

or

./linux_loader firmware.bin /dev/ttyUSBX

for example:

cd Tool/Win

winloader.exe at_cmds.bin 4

or

cd Tool/Linux

./linux_loader at_cmds.bin /dev/ttyUSB0

NOTE: The firmware binary file may be of two kinds of flash offset for program start entry and vectors: zero offset and 8 Kbytes offset (the bootloader size). This depends on linker file. To produce an application which will keep alive the bootloader user should use customized *at_cmds.icf* **linker file (applied and set as default in Nwave project linker options) which puts an application to 8 Kbytes boundary. If user needs to use all flash size for an apllication including first 8 Kbytes then the default chip linker file should be used.**

ATTENTION: The bootloader will be erased if an application is built with zero offset (by using default chip linker file). If this happened then the only way to program the device is a J-Link adapter. By means of the J-Link adapter user can restore the bootloader back again.

6. 2. Flashing bootloader into device

If your device does not have a bootloader or you have deleted/overwritten the MCU flash then you can put it into the device back again. You will need to have the J-Link adapter.

- Getting bin2hex utility

  Download bin2hex utility. Extract zip-archive.

- Converting binary file to an Intel Hex file.

  Run ***bin2hex.exe nwave_boot.bin nwave_boot.hex***

- Getting the Simplicity Studio Production Programmer

  Download Simplicity Studio Production Programmer . Extract zip-archive.

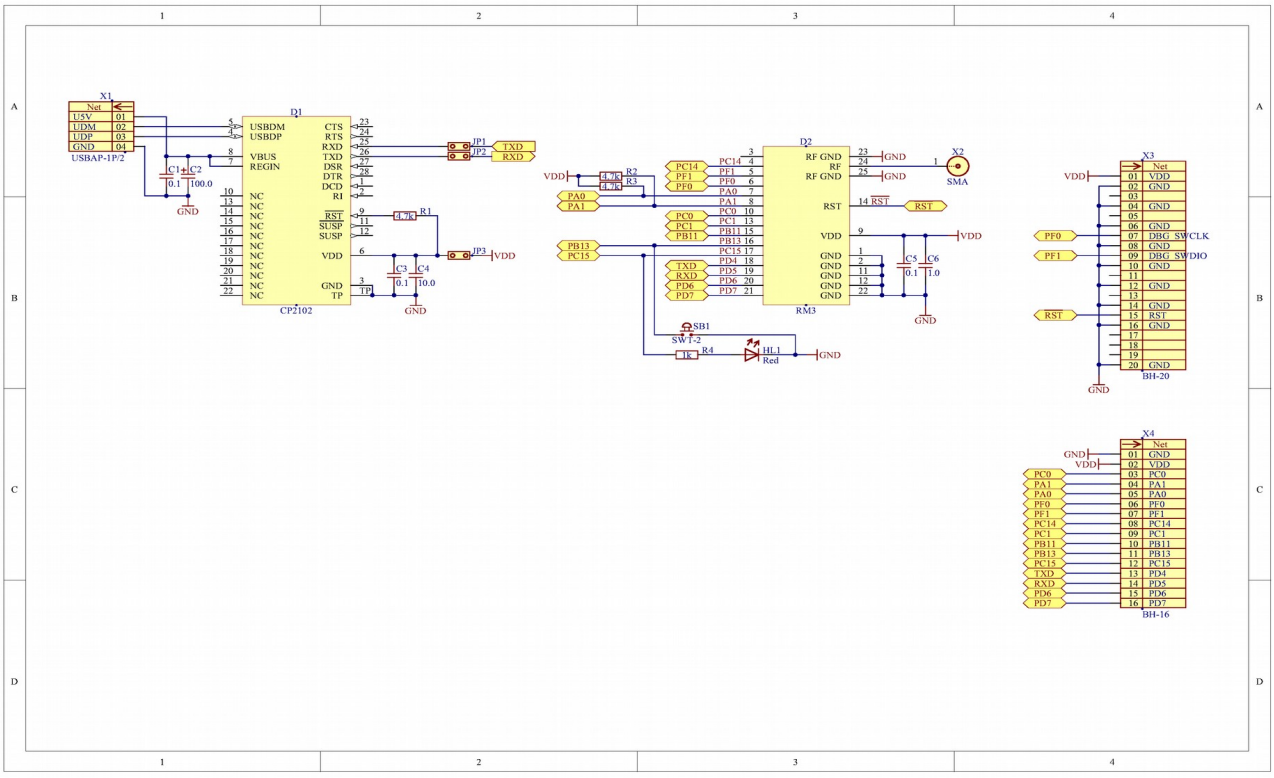
- Uploading a firmware to the MCU flash

  Run *programmer.exe*

  Press *Browse* button and select *nwave_boot.hex* file. Then press ***ProgramDevice*** button.

7. Electrical Scheme

The figure below is an electric circuit diagram of the NWAVE development kit.



8. Bill of Materials

The table below is a list of used mechanical and electronic components and their parameters.

Designator	Description	Value	Tolerance	Parameters	Package	Part Number	Manufacturer	Quantity	Pins
C1, C3, C5	Capacitor	0.1	10%	X7R, 16V	0603	GRM188R71C104K	Murata	3	2
C2	Polar Capacitor	100.0	10%	10V	Case-C	TPSC107K010R0150	AVX	1	2
C4	Capacitor	10.0	10%	X5R, 10V	0603	GRM188R61A106K	Murata	1	2
C6	Capacitor	1.0	10%	X7R, 10V	0603	GRM188R71A105K	Murata	1	2
D1	UART to USB	CP2102			QFN-28	CP2102-GM	Silabs	1	29
D2	RF ISM TS	RM3		868MHz	12.7x25.4mm	RM3	NWAVETEC	1	25
HL1	LED	Red		1.8V	0603	KP-1608SRC	KB	1	2
JP1, JP2, JP3	Jumper	PLS-2				PLS-2	BM	3	2
R1, R2, R3	Resistor	4.7k	1%		0603	CRCW06034K70F	Vishay	3	2
R4	Resistor	1k	1%		0603	CRCW06031K00F	Vishay	1	2
SB1	Tact Switch	SWT-2				FSM4JH	TE	1	2
X1	Connector Male	USBAP-1P/2				1734028-1	TE	1	5
X2	RF Connector	SMA		Z50	SMA	5-1814832-1	TE	1	2
X3	Connector Male	BH-20				BH-20	BM	1	20
X4	Connector Male	BH-16				BH-16	BM	1	16

9. Revision History

Here is a revision history for this document.

Author	Description of Changes	Date	Revision Number
Ruslan Gerasimov	Created	15-04-2015	1.1
Ruslan Gerasimov	Bootloader and Protocol-E notes added	01-05-2015	1.2
