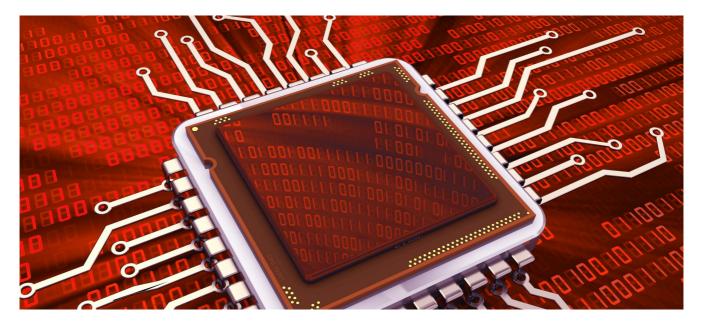


3. Getting Started with NuMaker-M032KI (pt 2)



Introduction #

This tutorial assumes that you've gone through the tutorial Getting Started with NuMaker-M032KI (pt 1) before. We will use some basics learned there in this tutorial without explaining it in detail again.

In this tutorial we will set up a project from scratch, letting the user LED on the NuMaker-M032KI blink about every one second. We will use Keil μ Vision 5 IDE and parts of CMSIS-Core for doing this.

Step 1: Open µVision 5 IDE and create a new Project

Create Project

- open µVision IDE
- Go to "Project -> new μVision Project"

Project	Flash	Debug	Peripherals	Tools	SVCS	Wind	
Nev	w µVisio	n Project					
Nev	w Multi-I	Project Wo	orkspace				
Op	en Proje	ct					
Clo	se Proje	ct					
Imp	oort						
Exp	Export						
Ma	nage						

• create a new project naming it as you like and storing it in your preferred folder

Select Device and startup files

Select the device you're using (for NuMaker-M032KI v1.0 it's M032KIAAE; you'll find it under "Nuvoton -> NuMicro M0 Family -> M031 -> M032KIAAE")



Device	
Software Packs	
Vendor: Nuvoton Device: M032KIAAE Toolset: ARM Search:	
Description: M032EC1AE M032FC1AE M032FC1AE M032KG6AE M032KG6AE M032KG6AE M032KG6AE M032KG6AE M032KG6AE M032LC2AE M032LC2	
M032LD2AE M032LE3AE M032LG6AE OK Cancel Help	_

• in the "Manage Run-Time Environment" select "Device -> Startup" to directly include the System Startup files for Nuvoton M03x Series (this is part of CMSIS-Core and is needed to get the Controller started correctly when powering up or after reset)

ftware Component	Sel.	Variant		Version	Description	
CMSIS					Cortex Microcontroller Software Interface Components	
CMSIS Driver					Unified Device Drivers compliant to CMSIS-Driver Specifications	
Compiler		ARM Compiler		1.6.0	Compiler Extensions for ARM Compiler 5 and ARM Compiler 6	
Device					Startup, System Setup	
Startup	V			0.00.001	System Startup for Nuvoton M031 Series	
File System		MDK-Plus	~	6.13.8	File Access on various storage devices	
Graphics		MDK-Plus	~	6.10.8	User Interface on graphical LCD displays	
Network		MDK-Plus	~	7.14.0	IPv4 Networking using Ethernet or Serial protocols	
PSA			-		Platform Security Architecture	
RTOS		FreeRTOS		10.3.1	FreeRTOS Real Time Kernel	
Security					Encryption for secure communication or storage	
Test					Software components for testing	
🐓 USB		MDK-Plus	~	6.14.1	USB Communication with various device classes	
mbed				-		
dation Output		Descriptic	in			
idetion Output		Descriptic	'n			

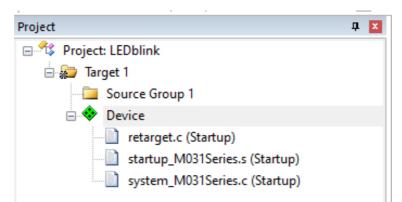
- · you can also add the startup files later, but the easiest way is doing it the way described before
- besides your application code these are the minimum needed files to get your controller up and running

You're now ready to add further files from the BSP and your own application code

Step 2: adding further files and app code to your project

Adding via project window

• in the project overview you now see:





- you can add Groups and existing files (e.g. from BSP) or your own files to the project
 - by right-clicking on a target you can add new Groups
 - by right-clicking on the certain groups you can add new files or existing files

Project	д	X
🖃 🍕 Project: LEDł	blink	
🖃 ᇶ Target 1		
- 🗁 : 🔊	Options for Target 'Target 1'	Alt+F7
÷	Add Group	
i 🗗 🗗	Manage Project Items	
	Rebuild all target files	
	Build Target	F7
\checkmark	Show Include File Dependencies	
Project	₽ X	
🖃 🎌 Project: LEDblir	· · · ·	
Project: LEDblin	ık	Ē
Project: LEDblir	ık	Alt+F7
Project: LEDblir	ik Group 1	Alt+F7
Project: LEDblir	Group 1 Options for Group 'Source Group 1'	1
Project: LEDblir	Add New Item to Group 'Source Group 1'	
Project: LEDblir	Add New Item to Group 'Source Group 1' Add New Item to Group 'Source Group 1' Add Existing Files to Group 'Source Group 1' Remove Group 'Source Group 1' and its Files	
Project: LEDblin Target 1 Source Oev	Add New Item to Group 'Source Group 1' Add New Item to Group 'Source Group 1' Add Existing Files to Group 'Source Group 1' Remove Group 'Source Group 1' and its Files Rebuild all target files	
Project: LEDblir Target 1 Source Oev	Add New Item to Group 'Source Group 1' Add New Item to Group 'Source Group 1' Add Existing Files to Group 'Source Group 1' Remove Group 'Source Group 1' and its Files Rebuild all target files	

- You also have the option to rename your Targets and Groups. Click on a target and then click on it again (no double-clicking) or use F2 on the keyboard
- You can rename "Target 1" to "M032KIAAE"
 Rename "Source Groupe 1" to "Main" (or something else) and add one new c-file (via Add New Item) which will be your main file

Project	4		
Project: LEDblink Aproject: LEDblink Aproved Main			
 Device retarget.c (Startup) startup_M031Series.c (Start system_M031Series.c (Start 	C C Hie (c) C++ File (.cp A Asm File (.s)	Create a new C source file and add it to the project. pp) reprocessed (S) (h) 1) 7	×
	Type:	C Fie (.c)	
	Name: Location:	main C:\Users\\anzen\\Wuvoton\Projects\scratch_tut	
E Project @ Books {} Functions 0. Ter	THATPS	Add Close He	p

- Add another group to your project, naming it "StdDriver"
- Add an existing file to the StdDriver Group (via right-clicking): navigate to the BSP folder on your PC, then go to "Library -> StdDriver -> src" and add gpio.c

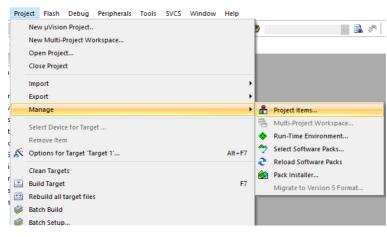


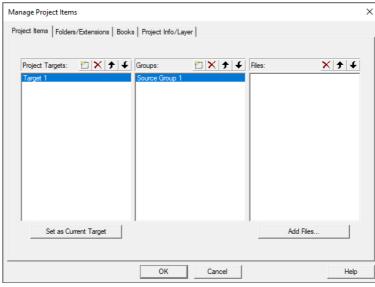
Name	Änderungsdatum	Тур	Größe
	-		
📓 acmp	03.07.2020 05:57	C-Datei	3 KB
📓 adc	03.07.2020 05:57	C-Datei	9 KB
🔐 bpwm	14.09.2020 09:59	C-Datei	29 KB
📔 clk	03.07.2020 05:57	C-Datei	35 KB
🖹 crc	03.07.2020 05:57	C-Datei	3 KB
🔛 ebi	14.09.2020 09:59	C-Datei	8 KB
📓 fmc	03.07.2020 05:57	C-Datei	15 KB
🛛 🔐 gpio	06.04.2021 14:31	C-Datei	4 KB
📔 i2c	07.09.2020 03:38	C-Datei	52 KB
🛃 pdma	07.09.2020 03:38	C-Datei	13 KB
🔐 pwm	14.09.2020 09:59	C-Datei	42 KB
📔 qspi	07.09.2020 03:38	C-Datei	29 KB
🥁 retarget	25.05.2021 08:50	C-Datei	22 KB
📔 rtc	03.07.2020 05:57	C-Datei	32 KB
🔤 eni	1/1 00 2020 00-50	C-Datei	38 K.B
			>
ateiname: gpio			Add
ateityp: C Source file (*.c)			▼ Close

- In the gpio.c and gpio.h there are some helpful definitions of macros and functions for configuring the GPIO Ports
- Using this functions and macros is the easiest way to configure your GPIOs to get the user LED on the board blinking

Adding via "Manage Project Items..."

- An alternative way for doing above mentioned steps is the "Manage Project Items..." window
- To open it you have several ways
 - Go to "Project -> Manage -> Project Items..."
 - Use the dedicated icon on the toolbar
 - Right click on the target or a group in the project tree and go to "Manage Project items..."





- By double-clicking you can rename targets and groups
- · Further you can add targets, groups and existing files in the dedicated part of this window



Manage Project Items		×
Project Items Folders/Extensions Book		
Set as Current Target	Add Files OK Cancel Help	

Step 3: main file

Header files

- Make sure to always include all necessary header files in your main file. For all Nuvoton projects you have to include at least the device specific header file. This is the device specific Peripheral Access Layer Header File, means that all peripheral header files are included here, too.
- So in this case include "M031Series.h"

#include "M031Series.h"

Initializations, functions and macros

- Now you have to make all necessary initializations of your system (additionally to those already done during startup); e.g. set the correct modes for the GPIOs
- As we want to blink the red user LED on the NuMaker-M032KI we have to set the dedicated I/O to Output Mode. The user LED is connected to Port B Pin 14 (PB14)
- You find the needed function for doing this initialization in the gpio.c driver you have added to the project before (void GPIO SetMode(GPIO T *port, uint32 t u32PinMask, uint32 t u32Mode))
- Insert following in your main function

GPI0_SetMode(PB, BIT14, GPI0_MODE_OUTPUT);

- There are different ways to write working code for blinking the LED, but the easiest one is to use the GPIO_TOGGLE function-like macro you'll find in the gpio.h #define GPIO_TOGGLE(u32Pin) ((u32Pin) ^=1)
- Use this macro in your main loop for toggling PB14 and let the LED blink

GPI0_TOGGLE(PB14);

- Now you need a delay to actually let the LED blink. In the first step you can use a "for loop" for inserting a delay
- Use a value of about 3200000 to get a delay of about one second

for(i=0; i < 3200000; i++)</pre>

- Don't forget to insert the necessary variable declaration for "i" into your main function. (You maybe need further declarations in your future projects)
- uint32_t i; (sized integer types are defined in the stdint.h which is found under C:\Keil_v5\ARM\ARMCC\include)

Final code

```
The whole main code is not more than this:
#include "M031Series.h"
int main(){
    uint32_t i;
    GPI0_SetMode(PB, BIT14, GPI0_MODE_OUTPUT);
    while(1){
        GPI0_TOGGLE(PB14);
        for(i=0; i < 3200000; i++);
        }
}
```



Step 4: Set correct "Options for Target..."

Options for Target #

- Before we can build the code and run it on the microcontroller we have to make some settings in the "Options for Target..."
- Open the "Options for Target..." via tab "Project" or via the dedicated icon

Proj	ect	Flash	Debug	Peripherals	Tools	SVCS	Window	Help	
	New µVision Project								
	New Multi-Project Workspace								
1	Open Project								
i .	Close Project								
	Import								•
	Exp	ort							•
	Manage								•
· .	Manage ,								
	Select Device for Target								
	Ren	nove Iter	n						
*	Opt	ions for	Target 'Te	emplate'					Alt+F7
	Clea	an Targe	te						
		-							
•		d Target							F7
	Reb	uild all t	target file	s					
1	Bate	h Build							

• Under tab "Target" set the ARM Compiler to "Use default compiler version 5" via the dropdown list (this is needed for making all files from the current Nuvoton BSP work together correctly)

uvoton	M032KIA/	Æ	Xtal (MHz): 1	2.0		Generation Compiler:		lt compiler vers	sion 5 💌
vstem \ M031AI	g system: Viewer File: E_v1.svd Custom Fil					se Cross-M se MicroLI	lodule Optimiza B	tion Big Endian	
Read/	Only Mem	ory Areas			Read/	Write Men	nory Areas		
default	off-chip	Start	Size	Startup	default	off-chip	Start	Size	NoInit
	ROM1:			0		RAM1:			
	ROM2:			- C		RAM2:			
	ROM3:			- c		RAM3:			
Г		,				on-chip	,		
	on-chip					IRAM1:	0x20000000	0x18000	
□ ▼	on-chip IROM1:	0×0	0x80000	•	14				

• Under tab "C/C++" you have to set all the include paths for the project (so that all necessary header files are included correctly)

rice Target Output Listing User	C/C++ Asm Linker Debug Utilities	8	
Preprocessor Symbols			
Define:			
Undefine:			
Language / Code Generation			
Execute-only Code	Strict ANSI C	Warnings: All Warr	nings 💌
Optimization: Level 0 (-00)	Enum Container always int	🗖 Thur	mb Mode
Optimize for Time	Plain Char is Signed	🗌 No A	Auto Includes
Split Load and Store Multiple	Read-Only Position Independent	C99	Mode
One ELF Section per Function	Read-Write Position Independent	🗖 GNU	J extensions
Paths	SIS_V3.03.000\Library\CMSIS\Include;\\	M031_Series_BSP	
Misc Controls			
	pcs=interworksplit_sections -l SIS_V3.03.000/Library/CMSIS/Include -l		Ŷ



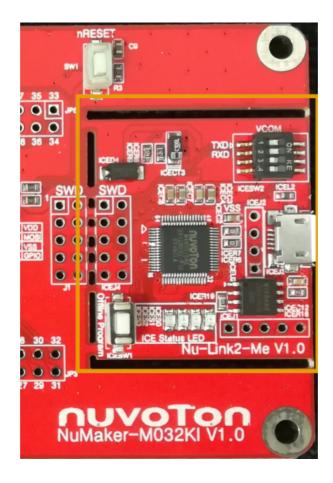
- Set the paths to include CMSIS, Device and StdDriver header files
- ...\M031_Series_BSP_CMSIS_V3.03.000\Library\CMSIS\Include
- $..\M031_Series_BSP_CMSIS_V3.03.000\Library\Device\Nuvoton\M031\Include$
- $..\.M031_Series_BSP_CMSIS_V3.03.000\Library\StdDriver\inc$

Folder Setup	?		×
Setup Compiler Include Paths:	2	X)
M031_Series_BSP_CMSIS_V3.03.000\Library\CMSIS\Include M031_Series_BSP_CMSIS_V3.03.000\Library\Device\Nuvoton\M031\Include M031_Series_BSP_CMSIS_V3.03.000\Library\StdDriver\inc			
OK Cancel			

• You maybe need to include some other header files in your future projects. Don't forget to add the correct paths here.

Connect the board and make further settings

• If you haven't connected your board via USB to your PC yet, do it now to make the last adjustments before debugging and programming the code on your board. Make sure to use the USB port on the Nu-Link2-Me part of the board





• Go to the "Debug" Tab in the "Options for Target..." and select "Nuvoton Nu-Link Debugger" from the dropdown list to use this as debugger

(A REPORT MINE CLOCK LIPTONES)	
🕅 Options for Target 'Template'	×
Device Target Output Listing User C/C++ Ast C Use Simulator with restrictions Setting Vium Speed to Real-Time	
Load Application at Startup Run to main() Initialization File: Load Application Electronic Edit.	Load Application at Startup Initialization File: Load Application Fi
Restore Debug Session Settings Image: Breakpoints Image: Toolbox Image: Watch Windows & Performance Analyzer Image: Memory Display Image: System Viewer	Restore Debug Session Settings Image: Seakpoints
CPU DLL: Parameter: [SARMCM3.DLL]	Driver DLL: Parameter: SARMCM3.DLL Dialog DLL: Parameter: TARMCM1.DLL pCM0
Warn if outdated Executable is loaded Manage Component	T Wam if outdated Executable is loaded
ок	Cancel Defaults Help

• Open the settings afterwards and set Chip Type to: "M031"

Options for Targe	et 'Target 1'		×
Nu-Link Driver Setu	р		×
Debug Trace Nu-Link	7215r Cortex-M	Chip Select Chip Type: M031 Chip Type: M031 Supporting Forum EN: http://forum.nuvoton.com/ SC: http://www.nuvoton-mcu.com/ SC: http://www.nuvoton-mcu.com/ Reset: Normal Reset: Autodetect	
Max Clock: - Power Control	1MHz •	2.5v 🕫 3.3v . C. 5v	
		OK Abbrecht	an
		OK Cancel Defaults Help	

• You should now be able to Build and run the code on your board (or make some debugging before)

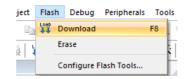
Step 5: Building, debugging and running code

Build Options (or use dedicated icons on the toolbar):



Proj	ect	Flash	Debug	Peripherals	Tools	SVCS	Window	Help			
	Nev	v µVisio	n Project								
	New Multi-Project Workspace										
	Open Project										
	Close Project										
	Imp	ort							I		
5	Exp	ort									
5	Mar	nage							I		
5	Sele	ect Devic	e for Targ	et							
Ì	Ren	nove Iter	n								
×	Opt	tions for	Target 'Te	emplate'					Alt+F7		
	Clea	an Targe	ts								
	Buil	d Target							F7		
	Reb	uild all t	target file	s							
	Bate	ch Build									
	Bate	ch Setup									
٢	Trar	nslate C:	\Users\jar	zen\Nuvoton	Projects	\Templa	te\main.c		Strg+F7		
	Sto	p build									

Download (or use dedicated icon on the toolbar):



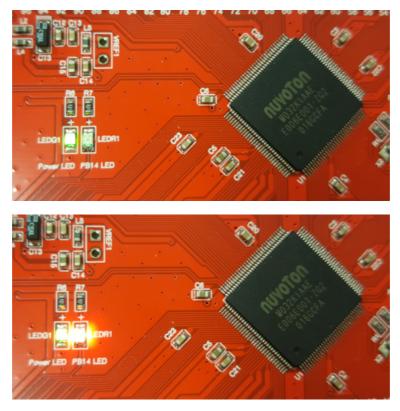
Debug Options (or use dedicated icons on the toolbar):

Deb	ug	Peripherals	Tools	SVCS	Window	Help			
<u>d</u>	Star	rt/Stop Debug		Strg+F5	5				
Ð	Energy Measurement without Debug								
e €ST	Reset CPU								
	Rur	n					F:	5	
8	Sto	р							
{·}	Ste	p		F11	1				
₿¢	Ste	p Over					F10	0	
P	Ste	p Out					Strg+F11	1	
0	Rur	n to Cursor Lin	e				Strg+F10	D	
\$	Sho	ow Next Staten	nent						
	Brea	akpoints					Strg+8	в	
•	Insert/Remove Breakpoint						F	9	
0	Enable/Disable Breakpoint					Strg+F9	9		
9	Disa	able All Breakp	oints in	current	Target				
R	Kill	All Breakpoint	s in Curr	ent Targ	et	Strg+U	mschalt+F9	9	
	OS	Support						Þ	
	Exe	cution Profilin	g					F	
	Mei	mory Map							
	Inli	ne Assembly							
	Fun	nction Editor (C	Open Ini	File)					



Conclusion #

If you've set up everything correctly and no errors arose, the red User LED (LEDR1) on your NuMaker Board should blink about every one second.



Now that you've gone through the getting started tutorials, make sure to check out also the tutorials on **Nuvoton's NuTool Suite**. This tools will help to reduce development efforts and time.