Short howto create lab1 'h3\_lab1\_samd21' used for FHdortmund-session Nov.2022

# TOC

Contents

[TOC 1](#_Toc117668213)

[History 1](#_Toc117668214)

[Goal 2](#_Toc117668215)

[Steps 2](#_Toc117668216)

[Expected result 2](#_Toc117668217)

[#eof 2](#_Toc117668218)

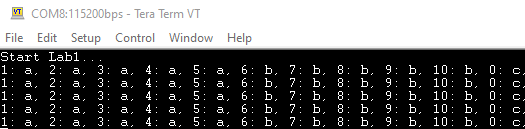
# History

v1.0: first version (SL, 24.10.22)

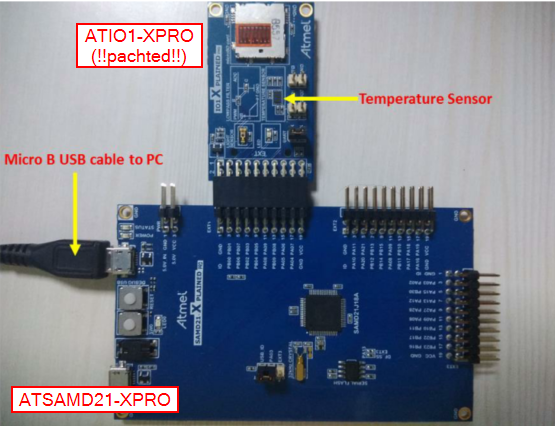
# Goal & Needs

We have 2 goals for this Lab:

1. toggleLED regularly
2. Output msg to Terminal (eg to check state)



**2**



***Not needed***

**1**

Toggle onboard LED at 1/250ms

Output formatted msg on terminal

Needs to fulfil goal

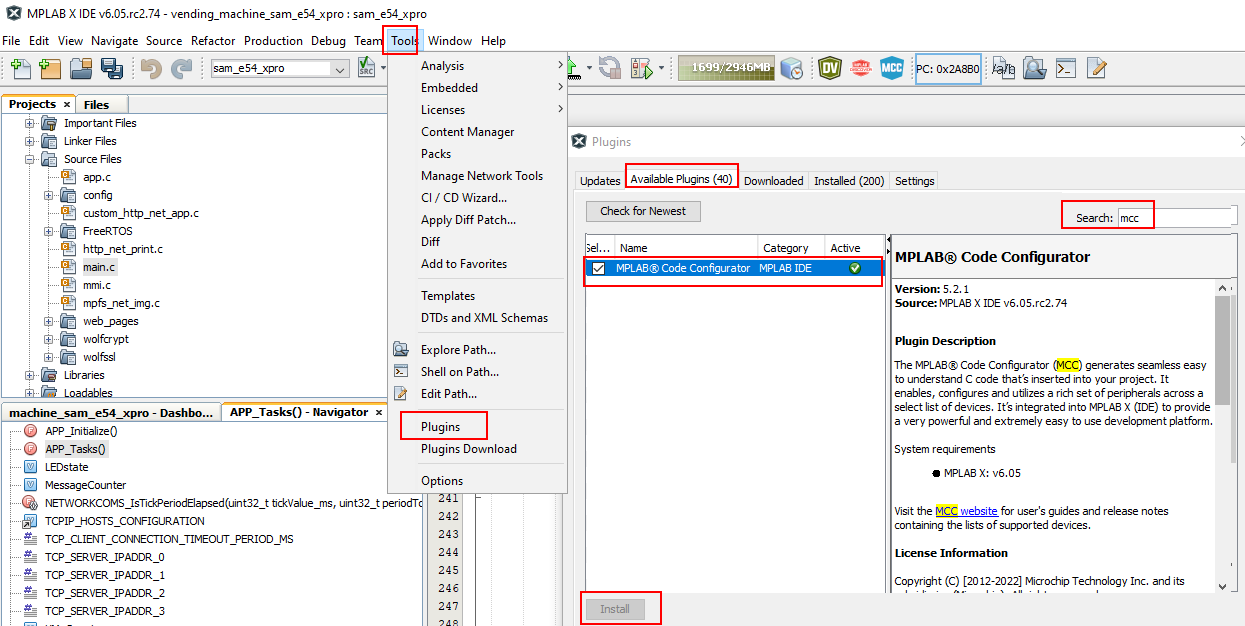
1. LED:
   1. Create ***regular event*** (->timer)
   2. @event toggle GPIO ***connected*** to LED
2. msg on terminal
3. print msg ***formatted*** -> printf()
4. ***Communicate*** with terminal -> UART
5. ***Connection*** to terminal -> pinning

# HW/SW needs

HW needed

* [ATSAMD21-xpro](https://www.microchip.com/en-us/development-tool/atsamd21-xpro)

SW needed

* MPLABX-v6.00(-> <http://www.microchip.com/mplabx>) - direct for Windows  
   [MPLABX-v6.00-windows-installer.exe](https://ww1.microchip.com/downloads/aemDocuments/documents/DEV/ProductDocuments/SoftwareTools/MPLABX-v6.00-windows-installer.exe)
* XC32-v4.10 (-> <http://www.microchip.com/compilers>) - direct for Windows  
  [xc32-v4.10-windows-x64-installer.exe](https://ww1.microchip.com/downloads/aemDocuments/documents/DEV/ProductDocuments/SoftwareTools/xc32-v4.10-windows-x64-installer.exe)
* MCC -> install plugin from 'list of available plugins' here:

# Steps

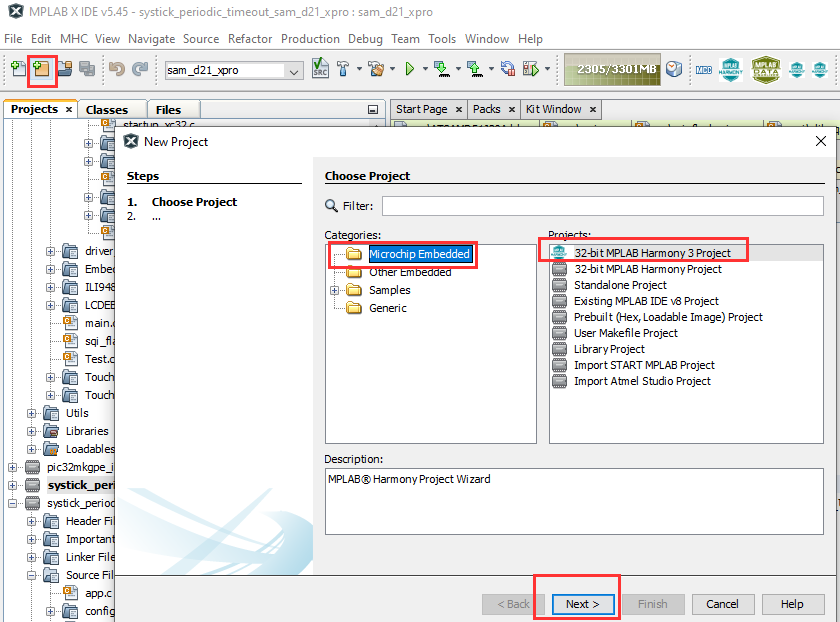
Steps to reach above 2 goals

1. Preparation
   1. Create project and set device-to-use
   2. Download H3-framework
2. Add modules in H3-UI 'MHC3'
   1. For timer event use systemTimer (already there -> ***System***)
   2. Add ***STDIO*** (=printf()) and connect to ***UART/SERCOM*** connected to host-terminal
3. Configure modules
   1. Configure sysTmr
   2. Configure UART/SERCOM
   3. Configure pins
4. Create code
   1. Create code and verify what was created
5. Add application code
   1. Add code to toggleLED@sysTmr-wrap and uartOut@4xsysTmrWrap
6. download and test
   1. Connect HW and download+run FW

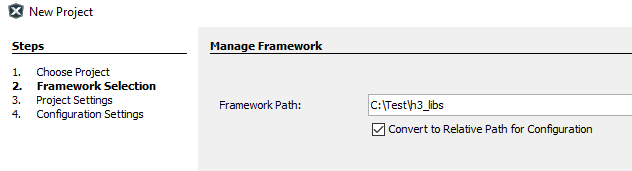
## Steps: Preparation

Create new project and download needed H3-framework

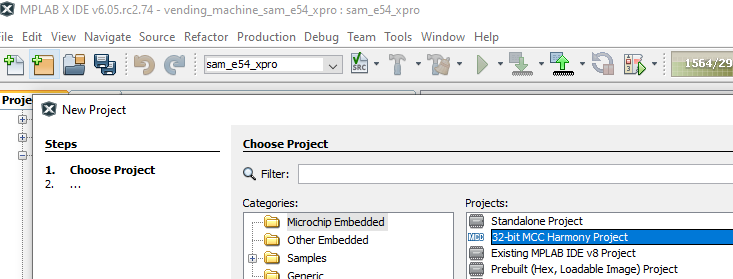
1. Create new project
2. Select type 'MCC-Harmony'
3. Specify path where H3-framework data will be download into



**1**

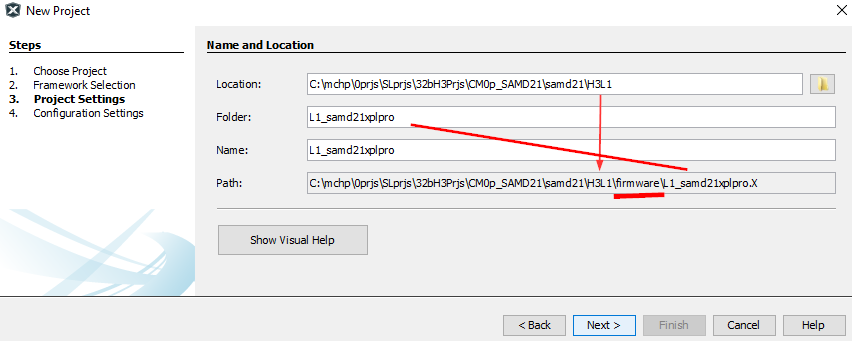


**3**

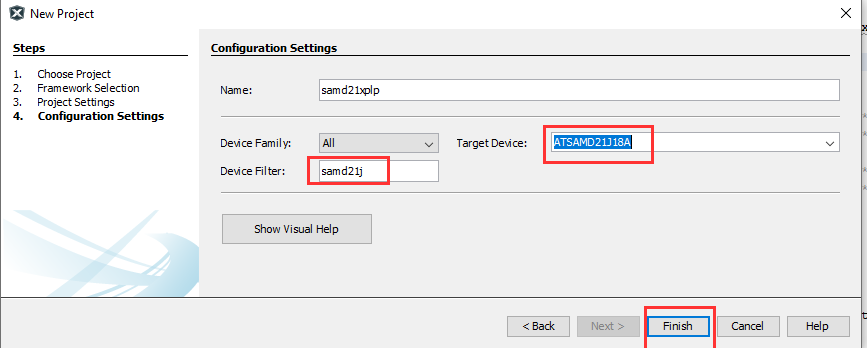


**2**

1. Specify project location/path, folder name, name in IDE
2. Finally provide name in H3-UI and select device (use filter to shorten device list)

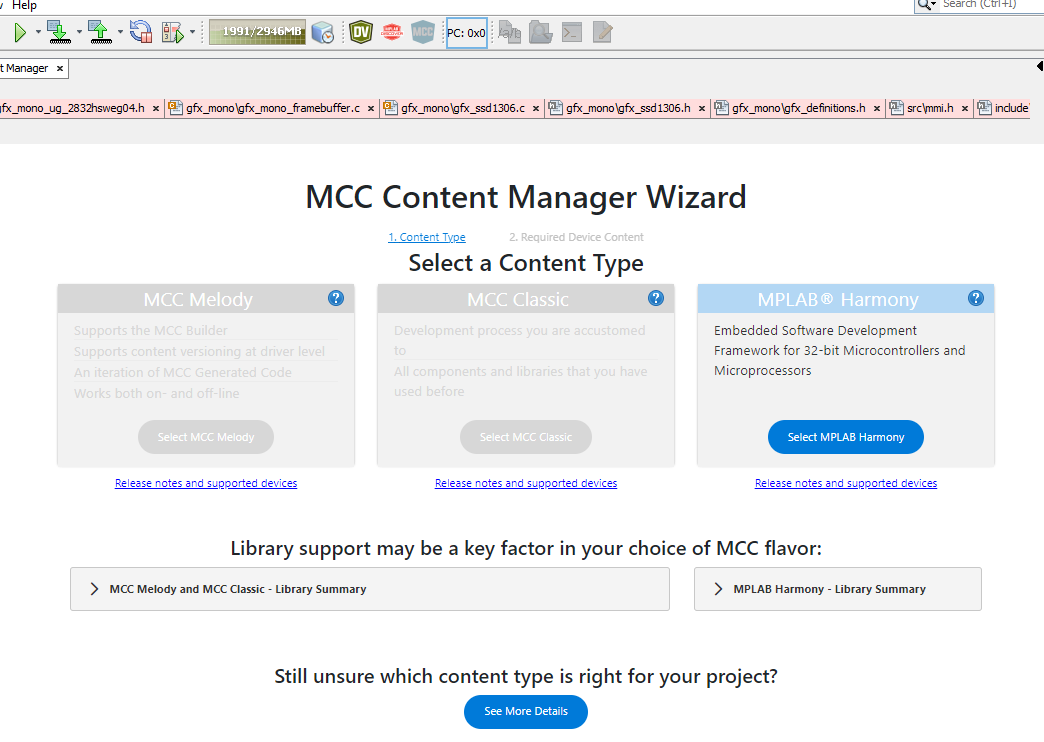


**4**

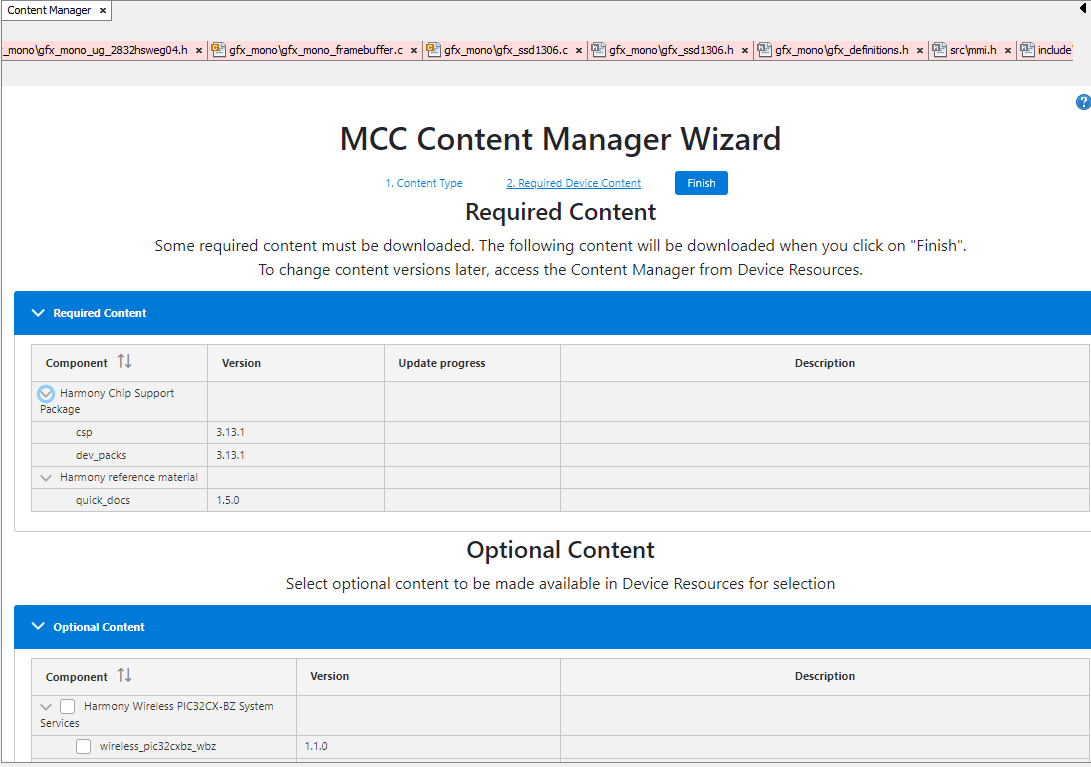


**5**

1. Download H3-framework



**6a**

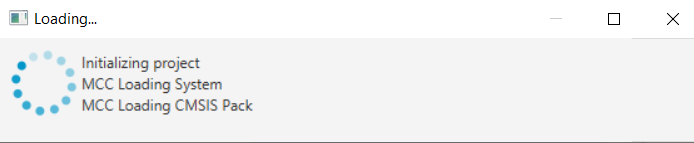


**6b**

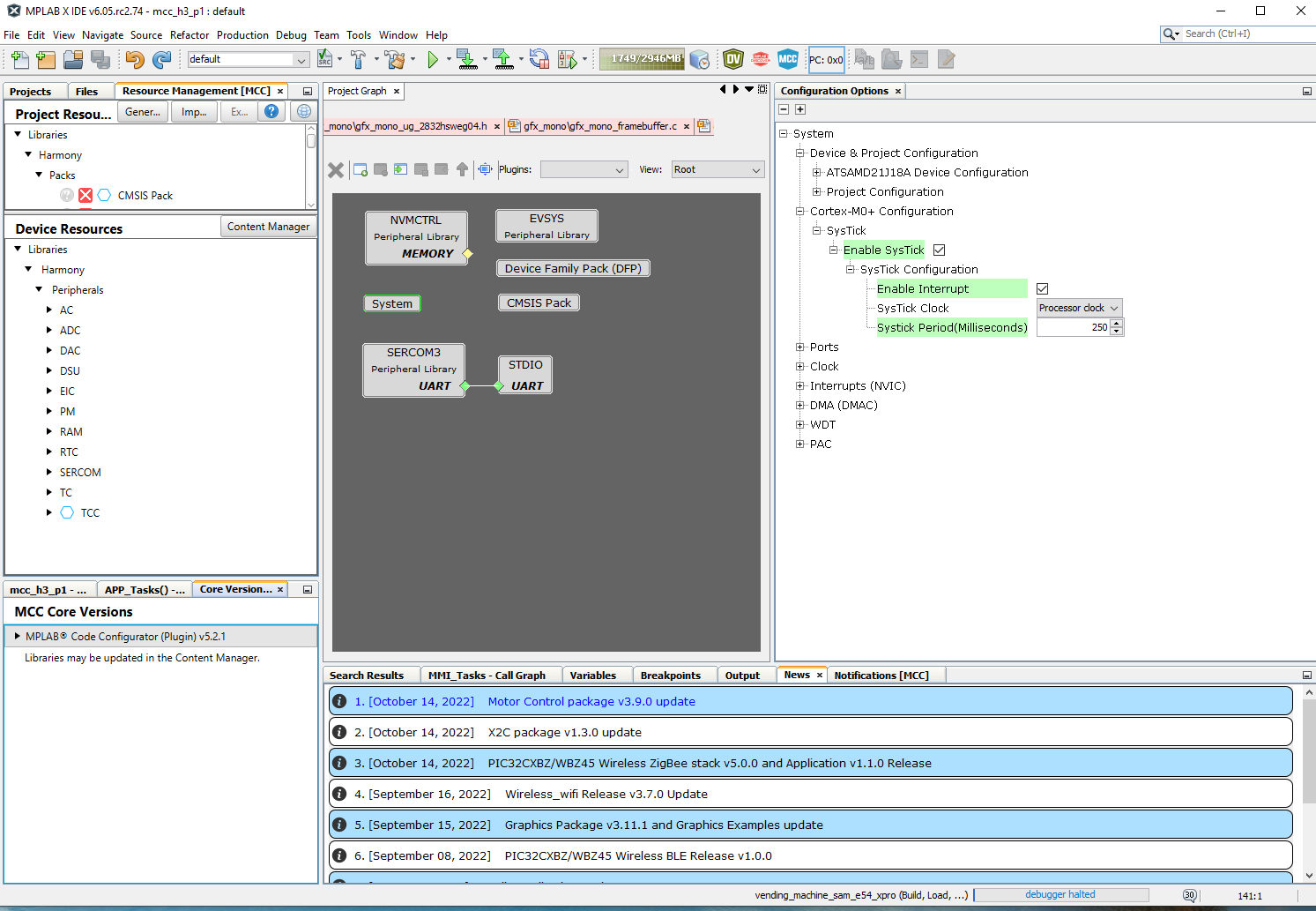


**6c**

1. Download H3-framework



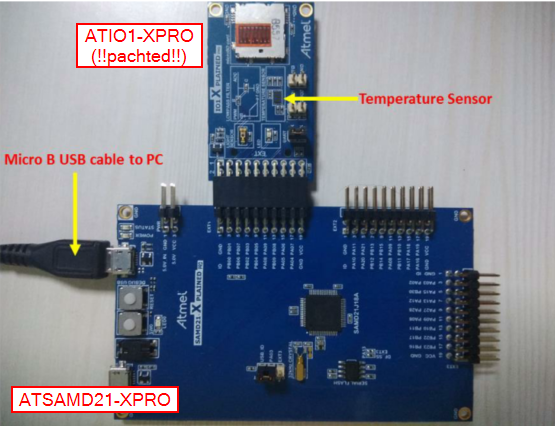
**7a**



**7b**

## Steps: use MHC to add functionality/modules and configure them

1. Infos from datasheet

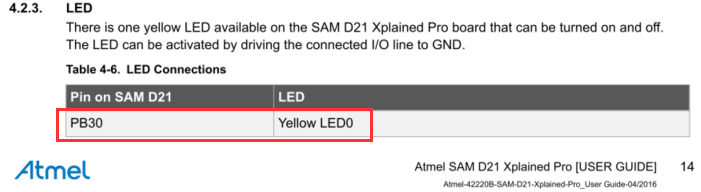


What we know from the Datasheet

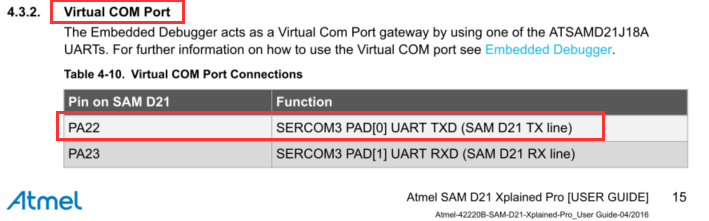
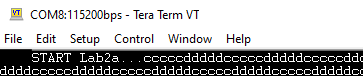
**Function MCU-pin**

LED0 PB30

Console/UTX PA22

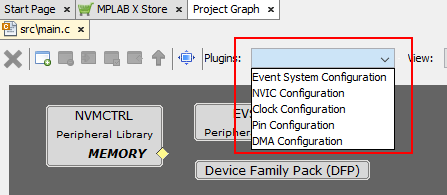
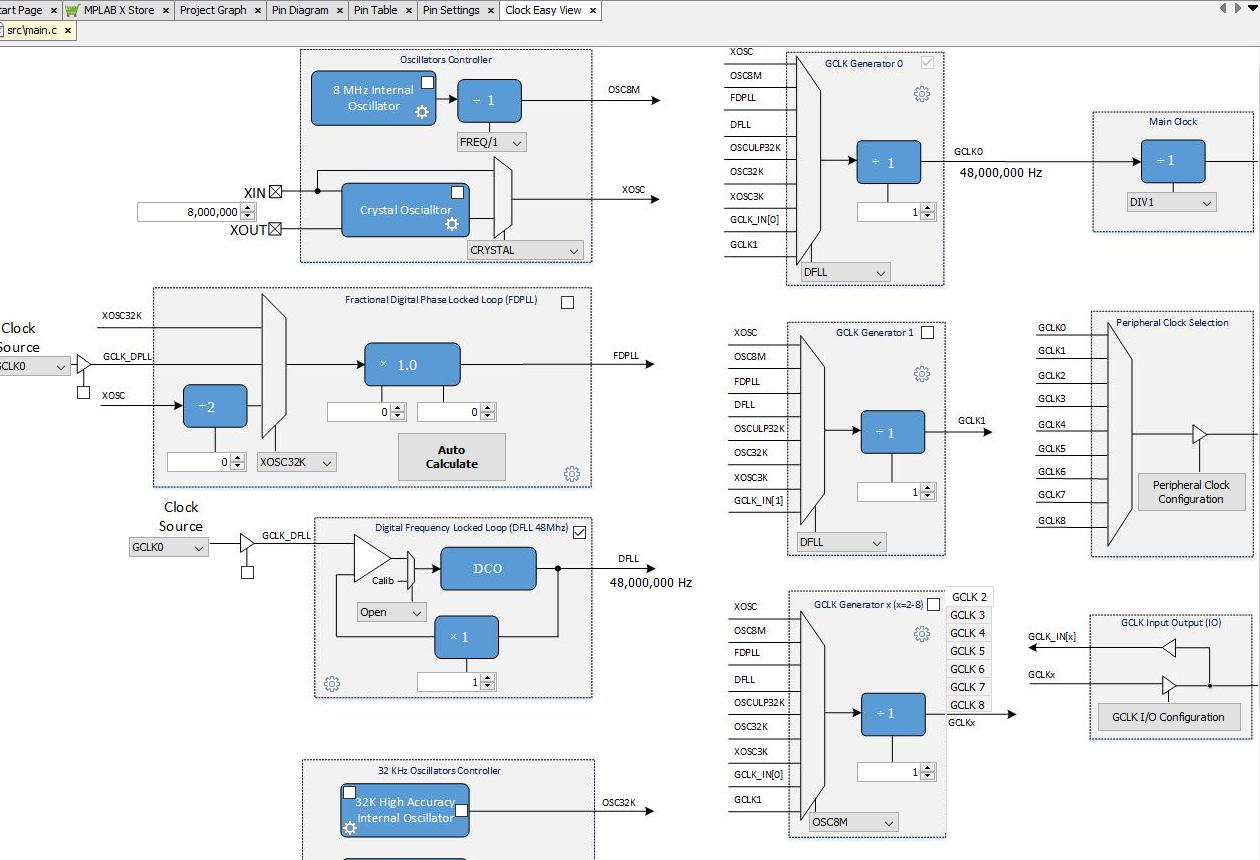
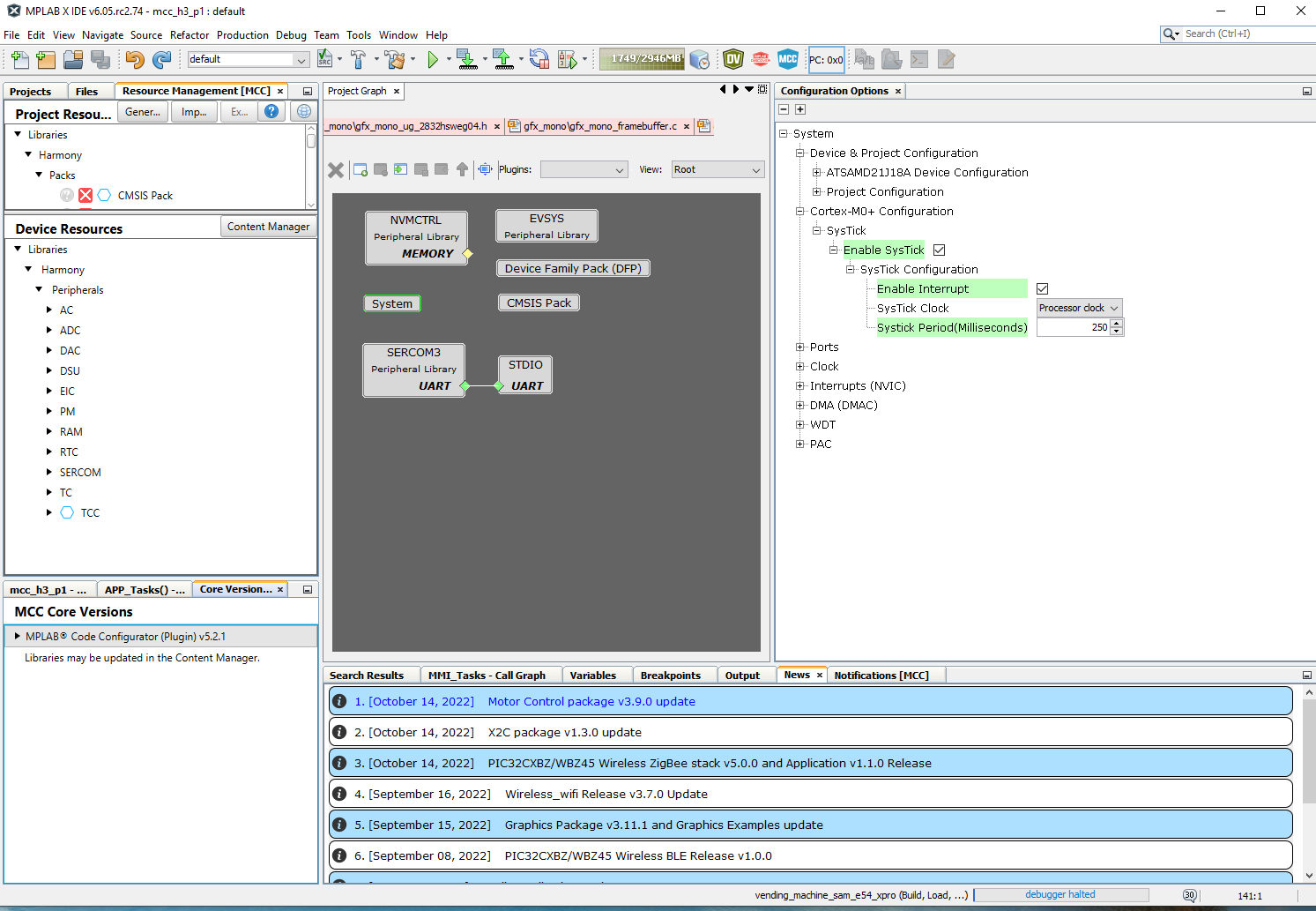


to toggleLED

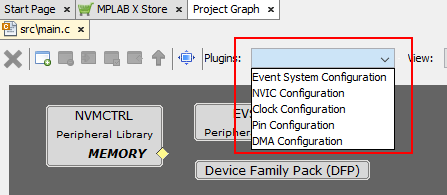
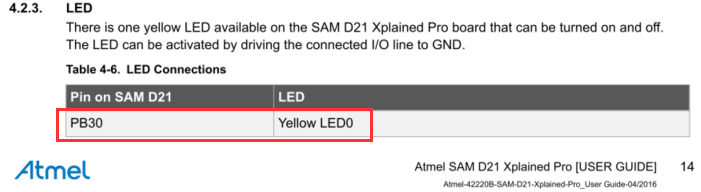
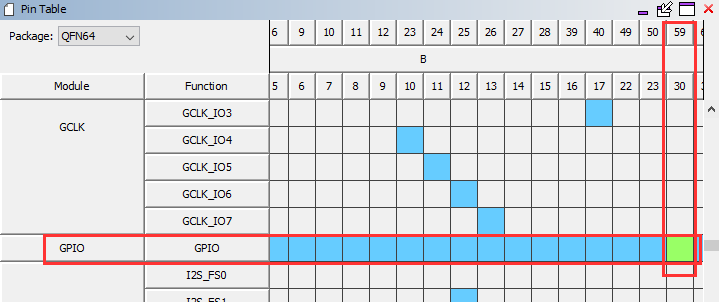
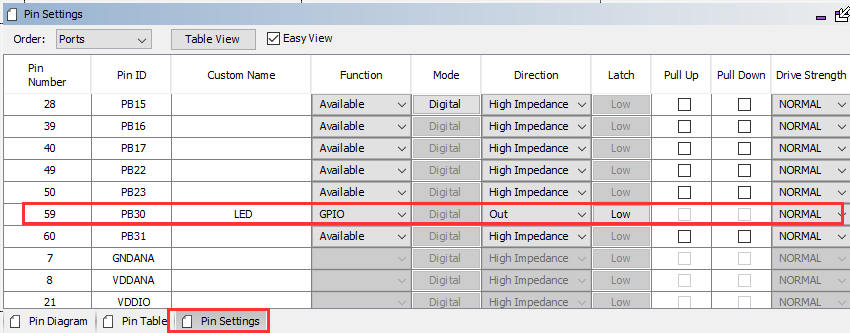


to send msg to terminal

1. Add H3-modules/-functionality: configure sysTmr to wrap @250ms



1. Add H3-modules/-functionality: enable GPIO, name 'LED' and assign to PB30

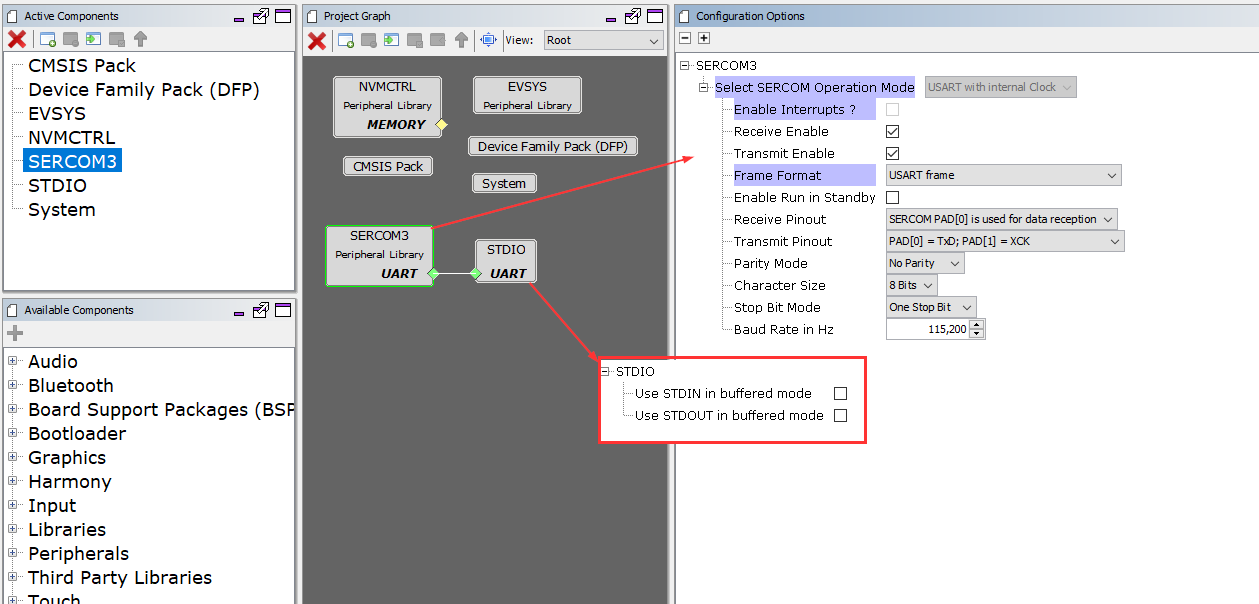
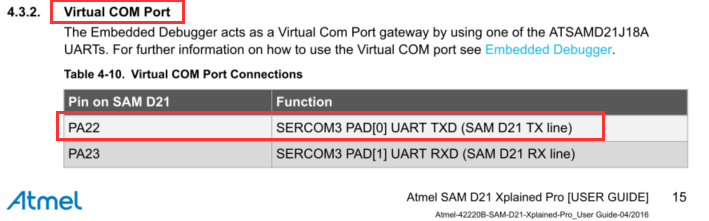
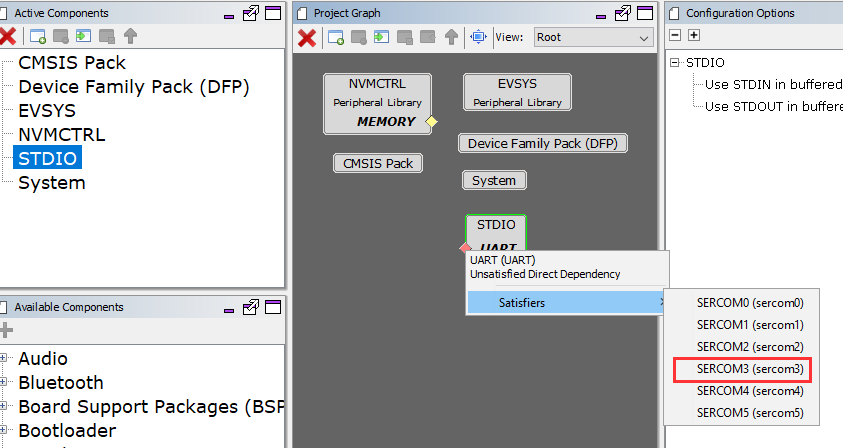
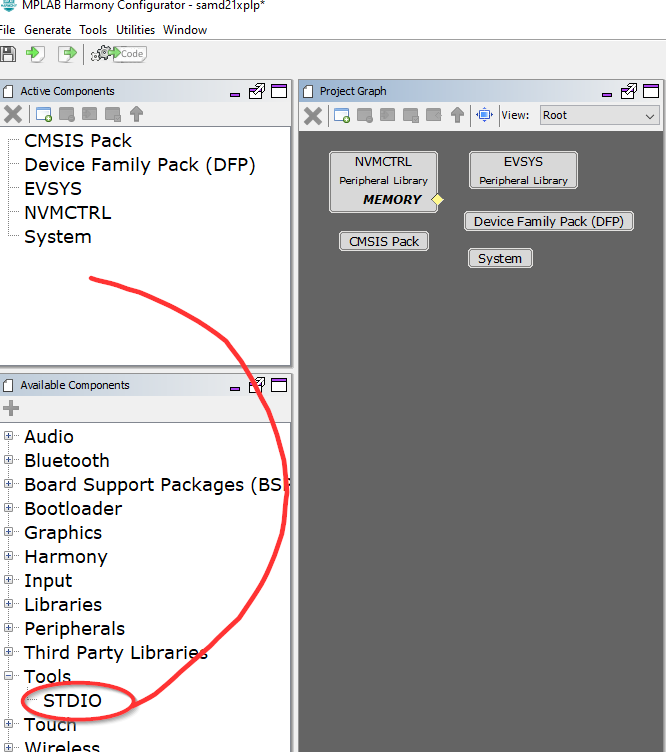


customName '***LED***', so in code you can use sth like ***LED***\_toggle()

Open PinManager

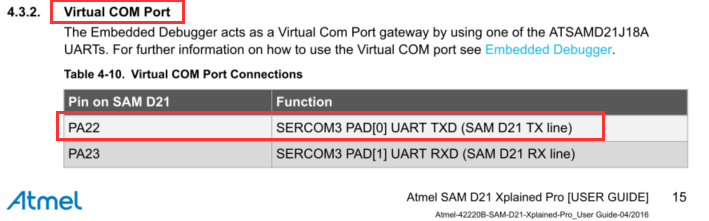
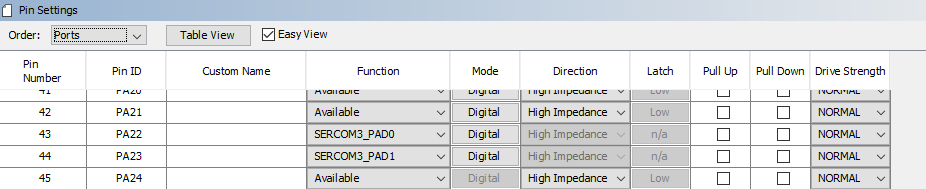
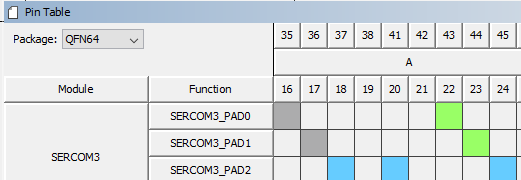
1. Add H3-modules/-functionality: add printf() functionality contained in STDIO+UART

uartOut: for printf() add H3-module STDIO and assign SERCOM3 to its UART-IF (=Satisfier) -> from Datasheet know that EDBG-UART = SERCOM3



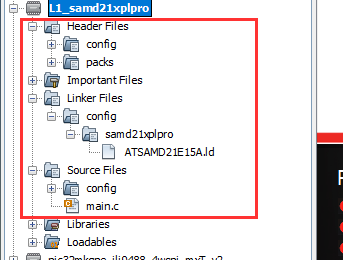
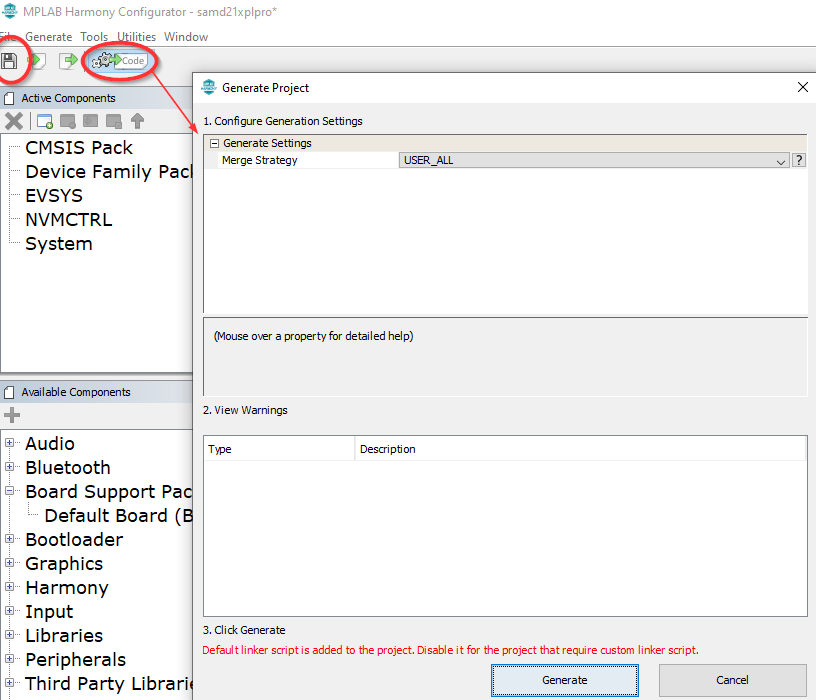
1. Add H3-modules/-functionality: uart/sercom pinning

SERCOM3 pinning to use vCOM from datasheet



1. Ready to generateCode

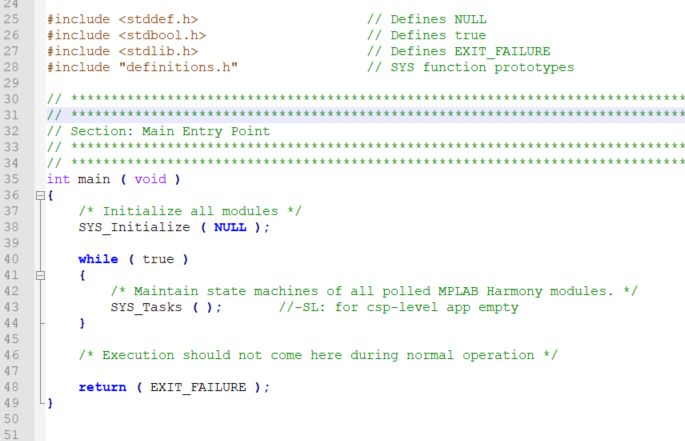
In prj view noticed how much code is created



## Steps: Code creation and correlation MHC-UI<->createCode

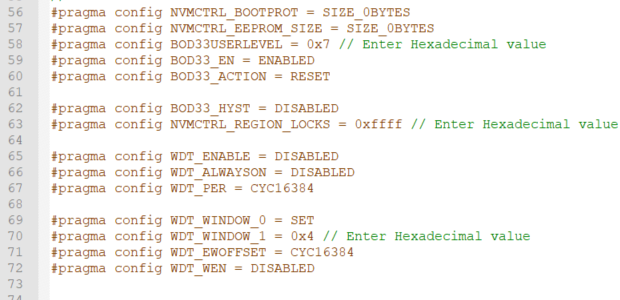
Goal next is to create the actual application, so what your FW should actually do

1. Let's first review the created code



main.c

Initialization.c



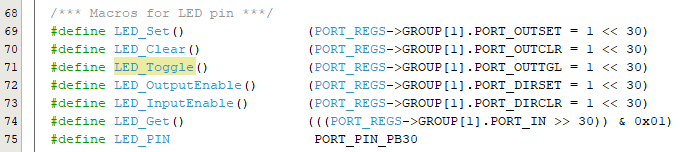
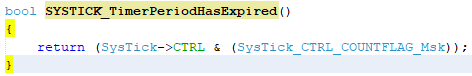
1. Lots of helpful code created which reflects datasheet

**Remember goal**

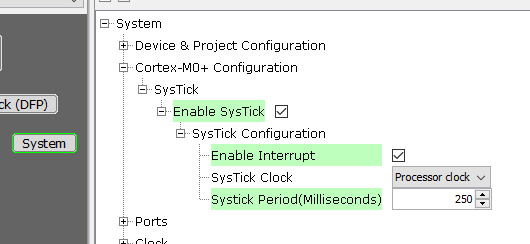
* + toggleLED @250ms -> sysTmr + LEDpin=out pinning
  + msg on console @1s -> stdio=printf() + uart/sercom3 + pinning to edbgUart

**Functionality**

* + LEDblink:
    - check when sysTmr=wraps (polled/callback)
    - toggleLED on systTmr=wrap ->   
      Port-lib (->GPIO-pin-assignment)   
      created fcts like 'portToggle()'
  + msg
    - On sysTmr-wrap count++ until count=4 (sysTmr=250ms, but charOut=1s)
    - On count=4 printf('a')  
      ->printf() is connected to SERCOM3 and putc() - can you find it?



1. find correlation MHC-UI <-> created code



As example MHC3-setting 'Systick Period(msec) = 250' set but where is to find in the code??

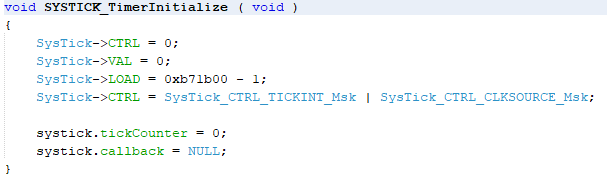
How does a TMR work? 'load some tmrReg and decrement with tmrClk -> set wrapFlag if loadReg=0' -> In SYSTICK\_TimerInitialize() you find this:

SysTick->LOAD = 0xb71b00 - 1;

The SAMD21/CM0+ runs @48MHz -> so 1xcycle=1/48MHz=20usec

-> 250ms/20us = 12.000.000 ticks -> 0xB7.1B00

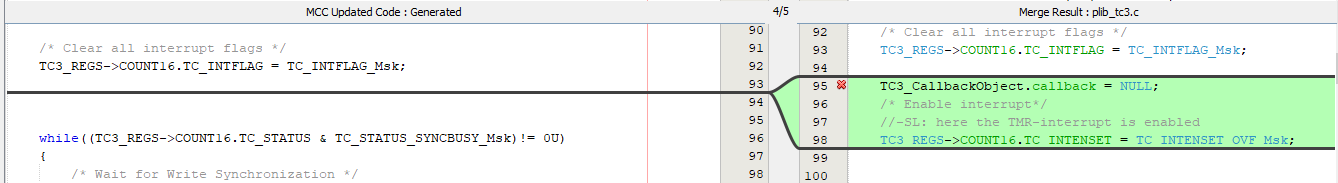
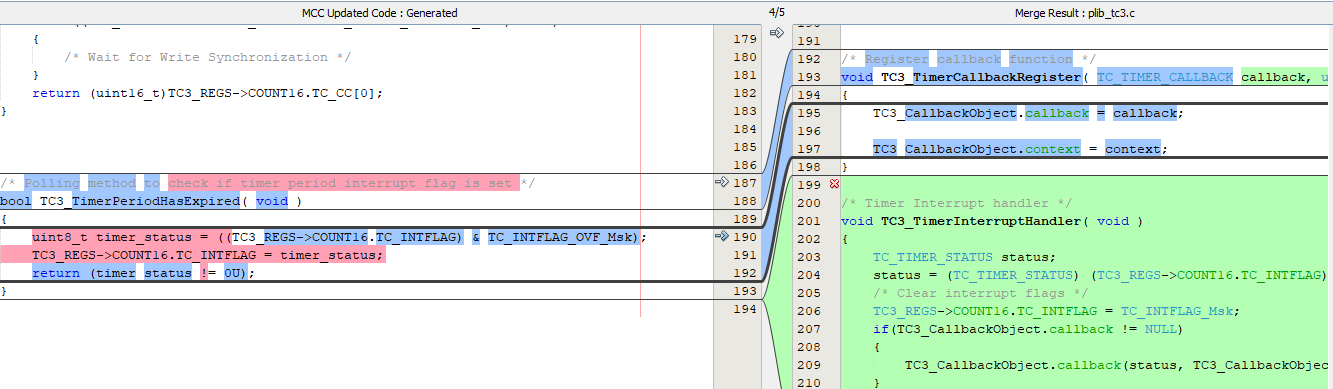
and this value is loaded into the SysTick->LOAD register



In MHC-UI

Created code

1. MHC-integrated merge-tool



left: TMR-Interrupt=***dis***nabled

-)noINT -> no callback-fct needed

-)noINT -> need polling-fct to find event=TMRwrap

-)event=TMRwrap still happens, BUT doesNOT trigger INT!)

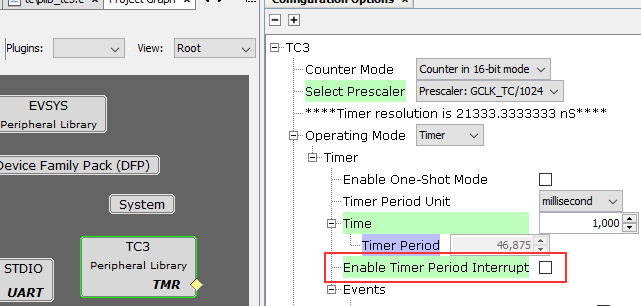
right: TMR-Interrupt=***e***nabled

-)INT needs callback, attached at the end of ISR

-)INT needs ISR

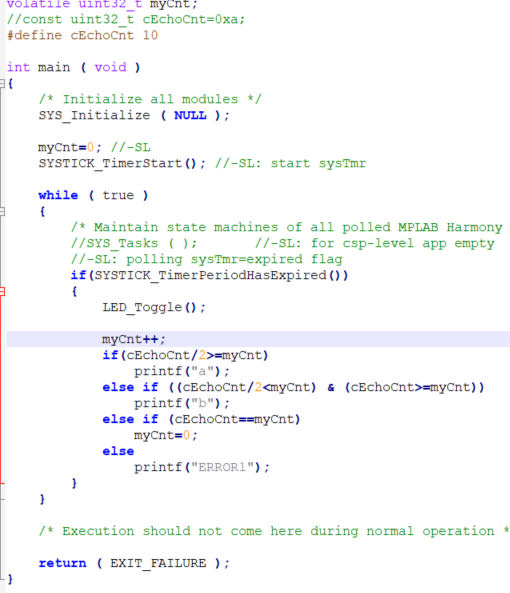
-)withINT -> need to enable INT-bit in configReg

En/disable Interrupt



## Steps: application

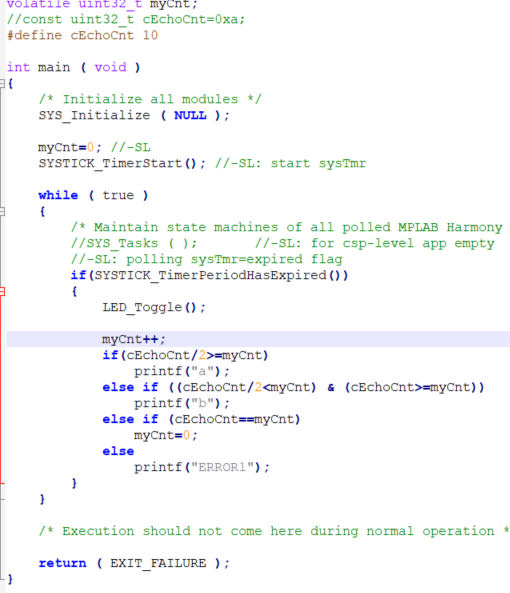
### Polled implementation



### callback implementation



v2/callback

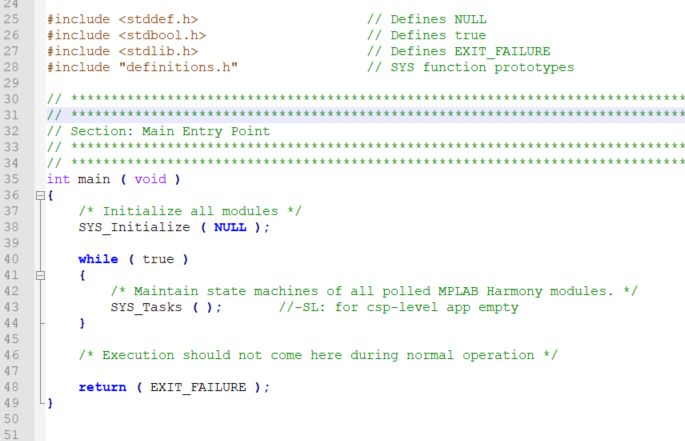
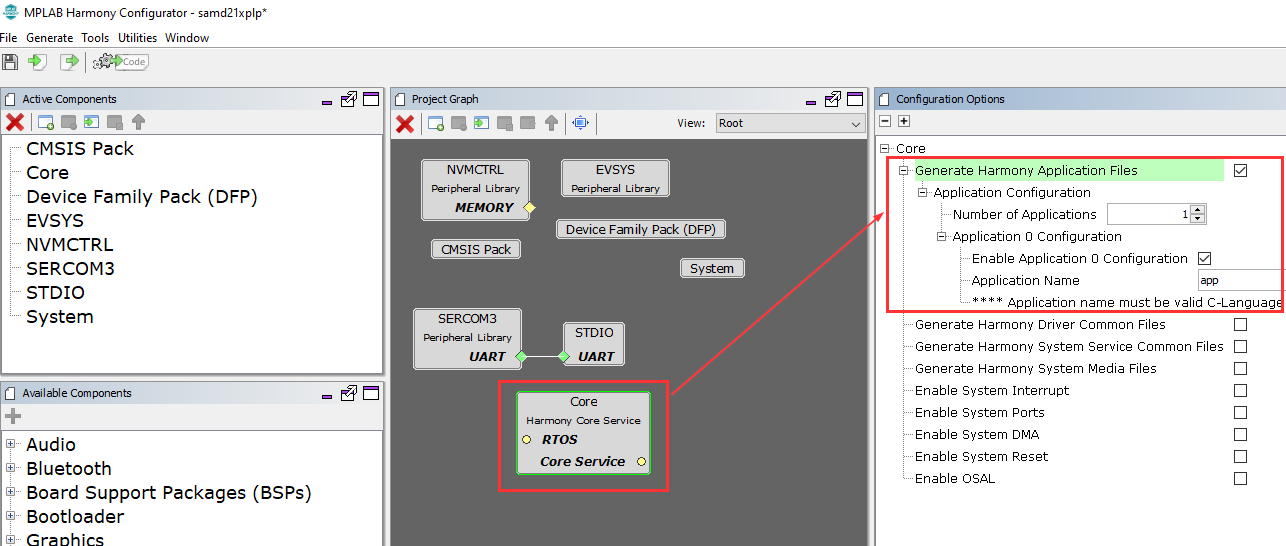


v1/polled

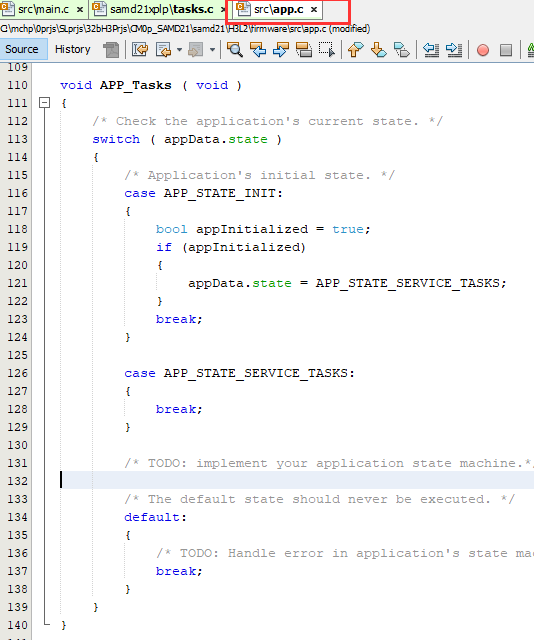
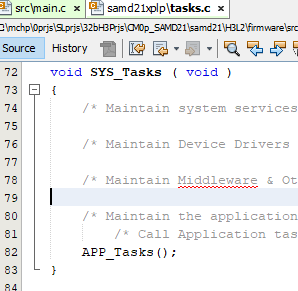
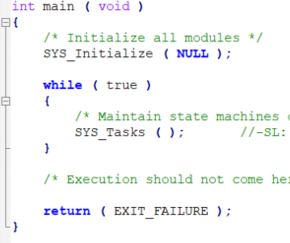
### statemachine implementation

first what changed in the code-structure is

To get Statemachine first need to add module CORE -> main() stays, but   
now SYS\_Tasks() has content



main.c



from main.c:main()/SYS\_Tasks() **->** tasks.c:SYS\_Tasks(){…} **->** app.c/APP\_Tasks()

x



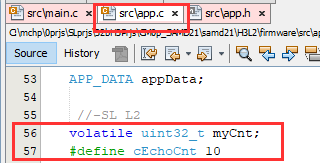
***move code from main()   
into APP\_Tasks()***

***mode1/polledMode***

***mode2/driverLevel***



**== while(1)**



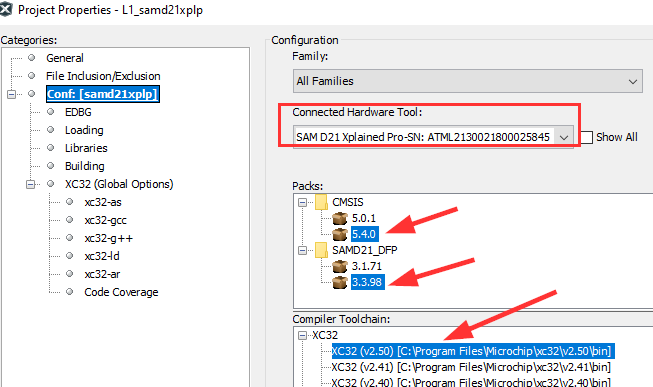
## Terminal/console

Graphical user interface

Description automatically generatedOn the host-PC you need a terminal program like putty or terraterm

## Download&Run

**Configure project-properties**

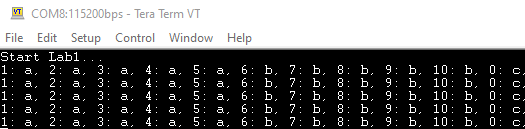
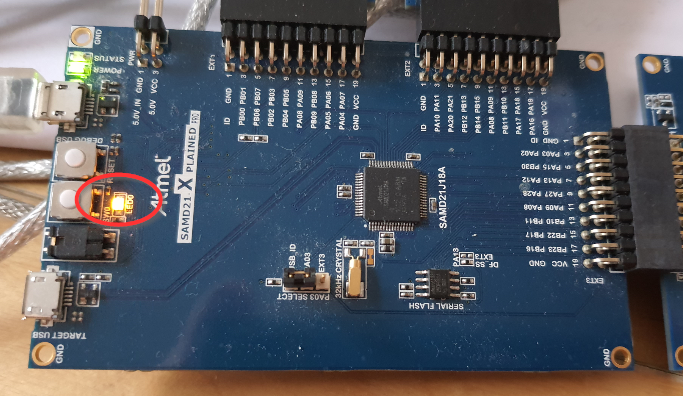


**compile&download**



# Expected result

**Expected result**



*goal1: LED toggling @1/250ms*

*goal2: msg on terminal*

# #eof