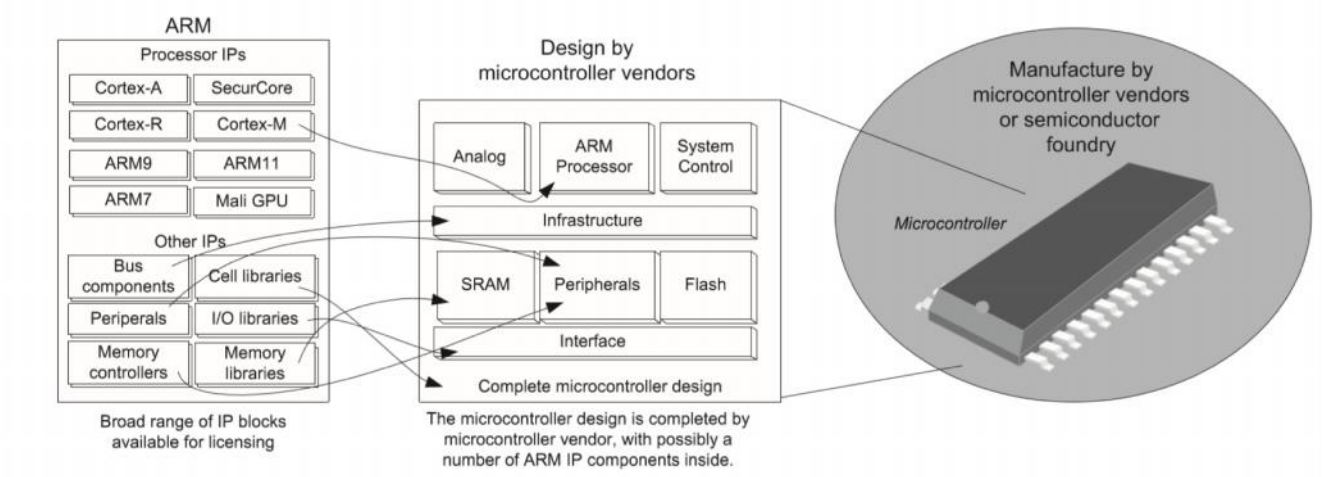
CEG3136A – Lab0
HelloWorld (C)

Objectives

To get familiar with the **Keil** Integrated Development Environment (IDE) µVision5 for ARM microcontroller. You'll write a 'Hello World' program.

Who

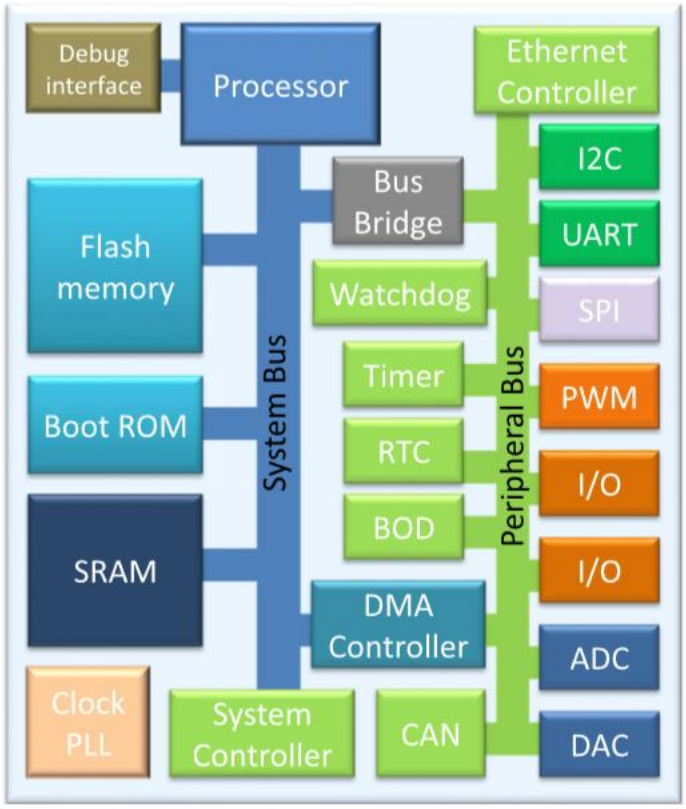
- **Keil** (www.keil.com) is an Integrated Development Environment tools developer supporting the ARM microcontroller IP core.
    - We are going to use Keil's µVision in the labs.
- **ARM** (www.arm.com) is a British semiconductor and software design company based in Cambridge, England. An Intellectual Property (IP) Developer. It develops processor IPs as well as other components essential to computer systems design. Integrated Circuits (IC) manufacturers can adopt ARM IPs as part of their IC designs.
    - We are going to use ARM Cortex –M4 processor in the labs.
- **STMicroelectronics** (www.st.com) is a Swiss-domiciled multinational electronics and semiconductor manufacturer headquartered in Geneva, Switzerland. An IC manufacturer that develops microcontroller IC and development boards that instantiate ARM IPs.
    - We are going to use STM32F417VGTx microcontroller with ARM Cortex –M4 microcontroller IP.



A microcontroller might contain multiple ARM IP products

What

A microcontroller is a System on Chip (SoC) that include at least one processor as well as other I/O and memory components that are packaged for generic or specific applications.



A microcontroller contains many different blocks

Where

You need to install the µVision5 IDE on your computer to develop small embedded applications.

When

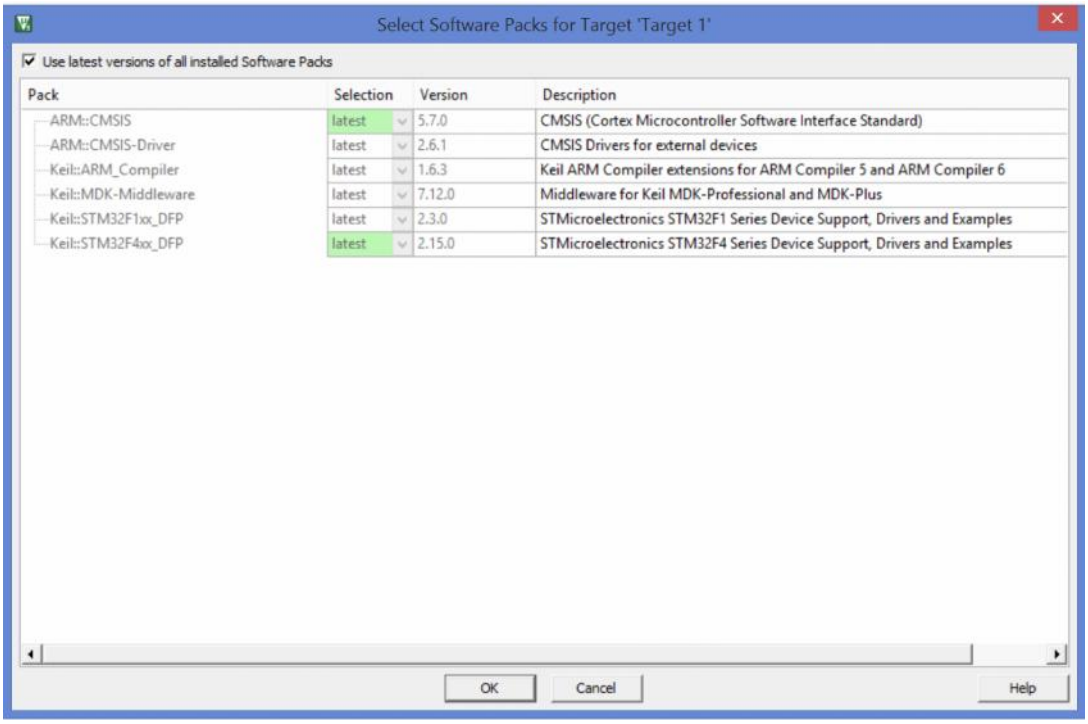
You have a weeks to set up your development environment and develop your Hello World Application

Part1 – Installation of Keil µvision5 IDE for Arm microcontroller

Go to **Keil** web site (www.keil.com) and follow the download tab to download MDK-Arm, Version 5.33 (November 2020), development environment for Cortex and Arm devices.

Note: You can also download MDK533.zip from Bright Space, unzip it and run the install exec. *Will take a long time, so be patient*!

You need to add software packs to be used. You'll need to add Keil:STM32F44xx_DFP & ARM:CMSIS.



Keil MDK-ARM contains various components including:

- µVision Integrated Development Environment (IDE)
- ARM Compilation Tools, including:
  - C/C++ Compiler
  - Assembler
  - Linker
- Debugger
- Simulator (*we'll be using simulation instead of real hardware*)
- RTX Real-time Operating System Kernel
- Startup code for various (1000s) of microcontrollers
- Flash programming algorithm
- Program Examples

Note: The µVision environment contains an instruction set simulator that allows testing of simple programs that do not require a development board. That is what we are going to use throughout the labs. You'll be able to simulate your solution on your computer.

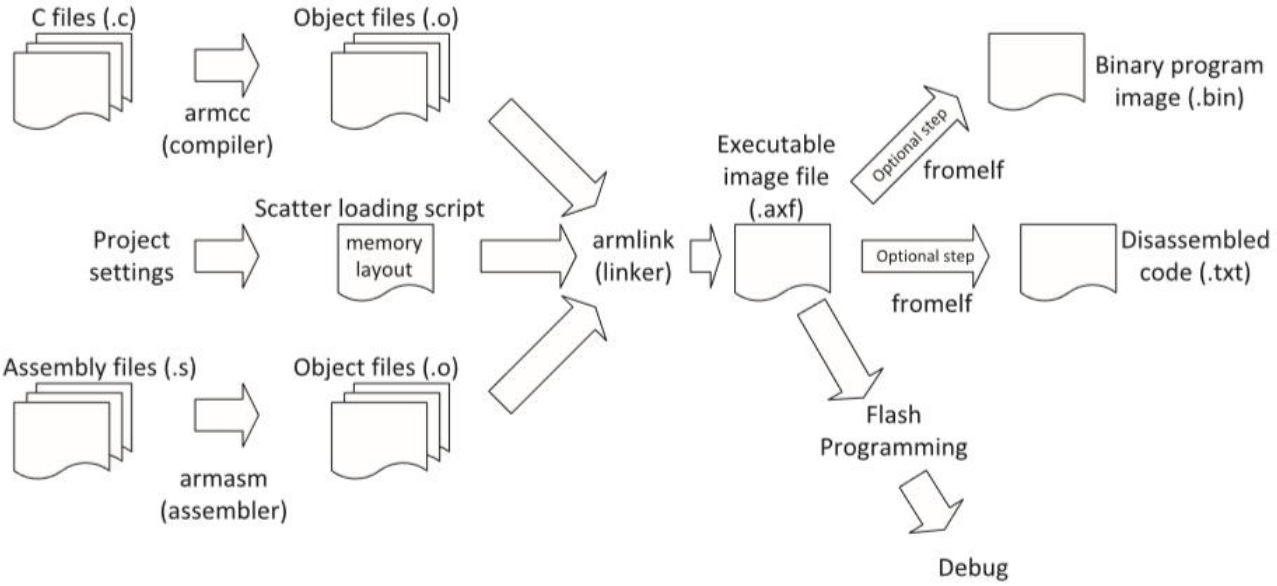
We'll be using a Lite version of the **Keil** MDK-ARM that is limited to 32KB program binary code size.

Part2 – Getting Started with Keil IDE

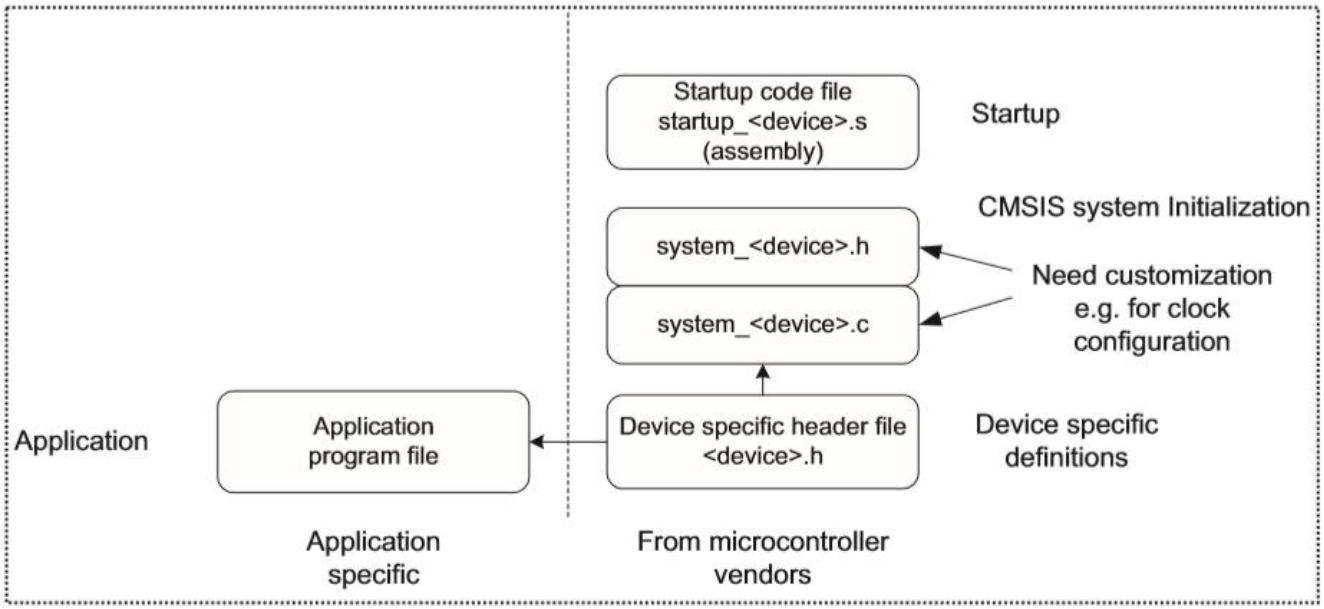
Refer to "The Definitive Guide to ARM Cortex-M3 and Cortex-M4 Processors"

- Chapter 15: Getting Started with **Keil** Microcontroller Development Kit (MDK)

Compilation Flow



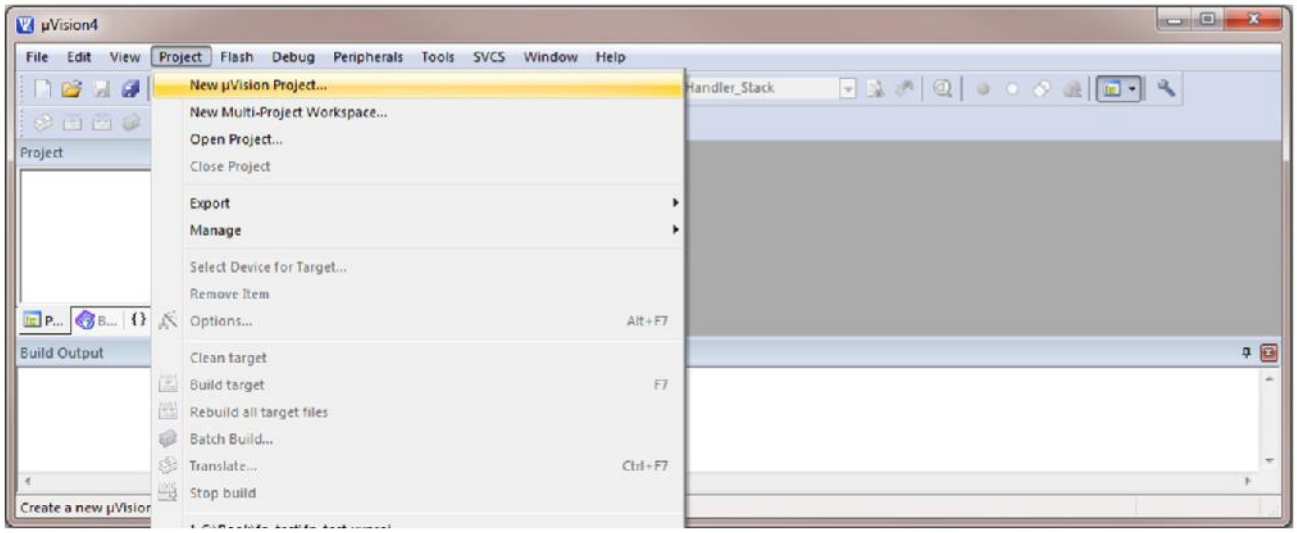
You'll write a c-program or two, a startup assembly code, and device-specific c-code are also included to support the selected device, all will be compiler and linked together to generate a binary machine code. We are not going to use flash programming tool for a physical device on test board, instead we are going to use the tool's simulator to run and debug your program.



Create a new project

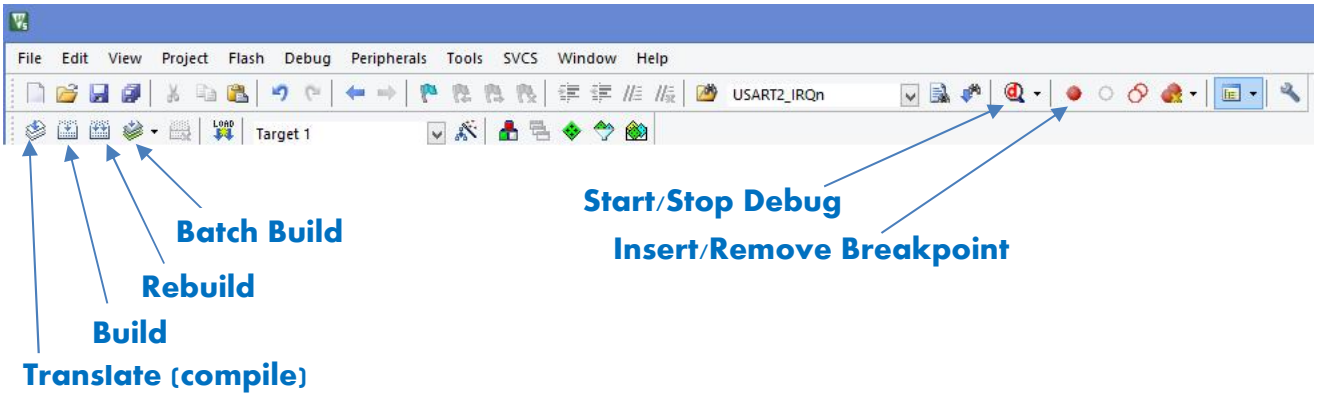
A project is already created in the attached zip file, but I would like you to go through the process by yourself to get experience or at least browse the steps.

1) Open the µVision IDE click on the project menu and select '**New** «**Vision Project**". Create a new directory (Lab1) to store the project files.

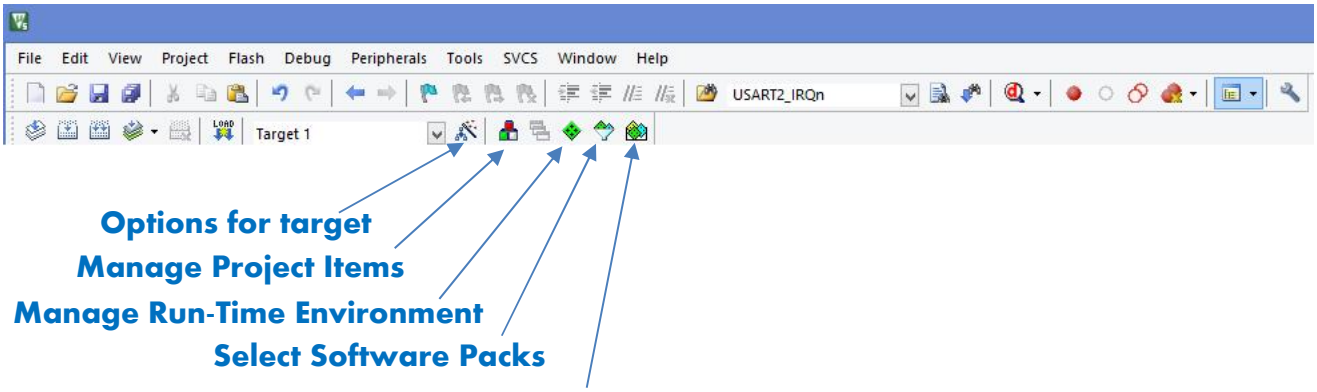


The µVision IDE starting screen, and creating a new project

2) Get familiar with the Compile buttons

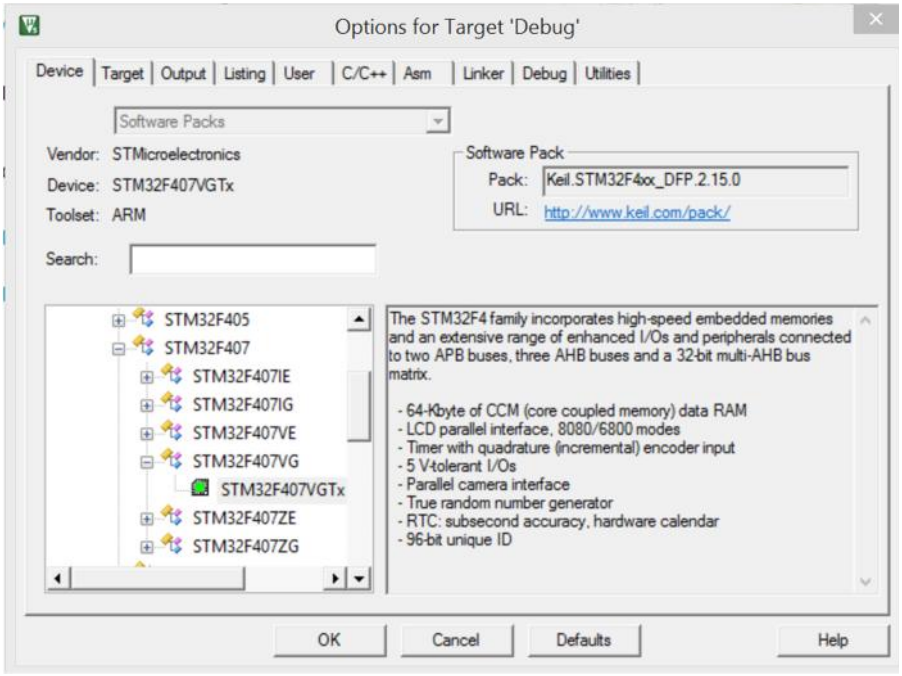


3) Get familiar with the target options and run-time buttons

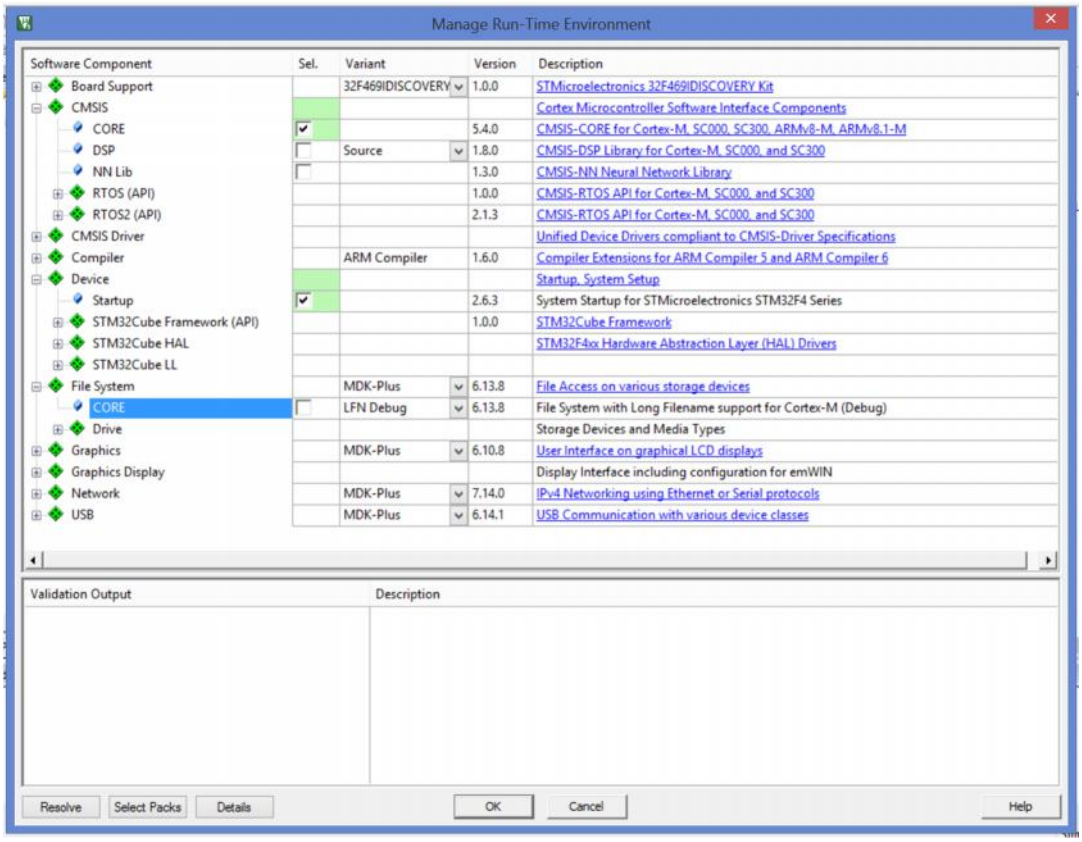


Pack Installer

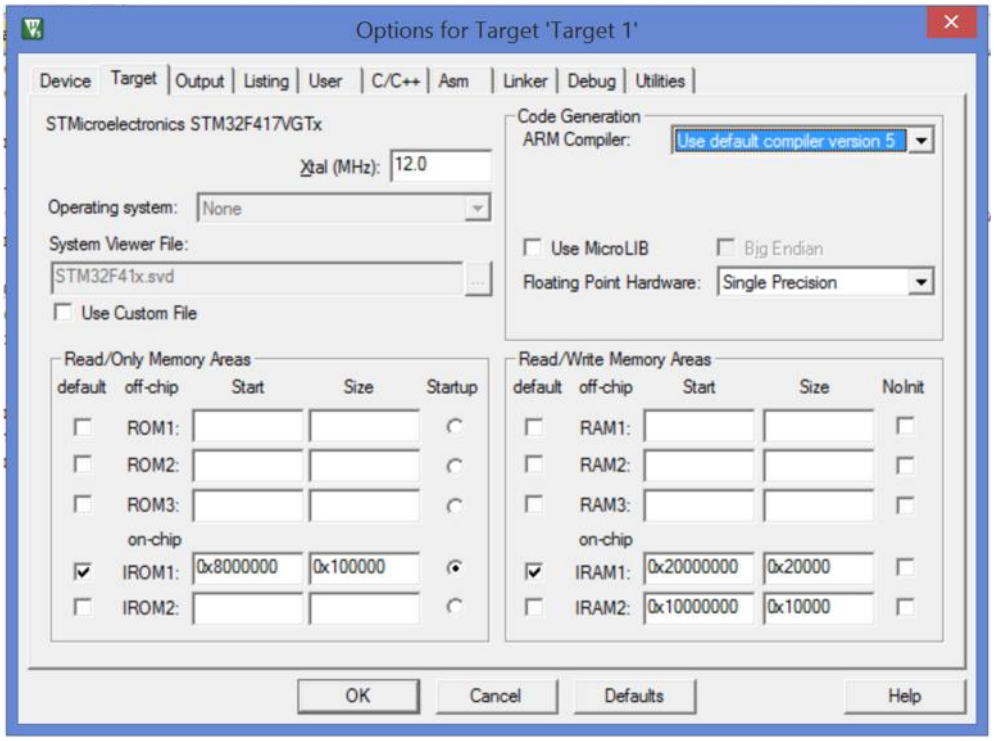
4) Select device as STMicroelectronics>STM32F4Series> STM32F417> STM32F417VG> STM32F417VGTx. This can be reached using the options for target button to open the dialog, then select the device tab.



5) Select Manage the Run-Time Environment, if it did not open automatically. Expand the CMSS components and select the CORE. Open the Device components and select startup – see below. This adds the startup code to the project.



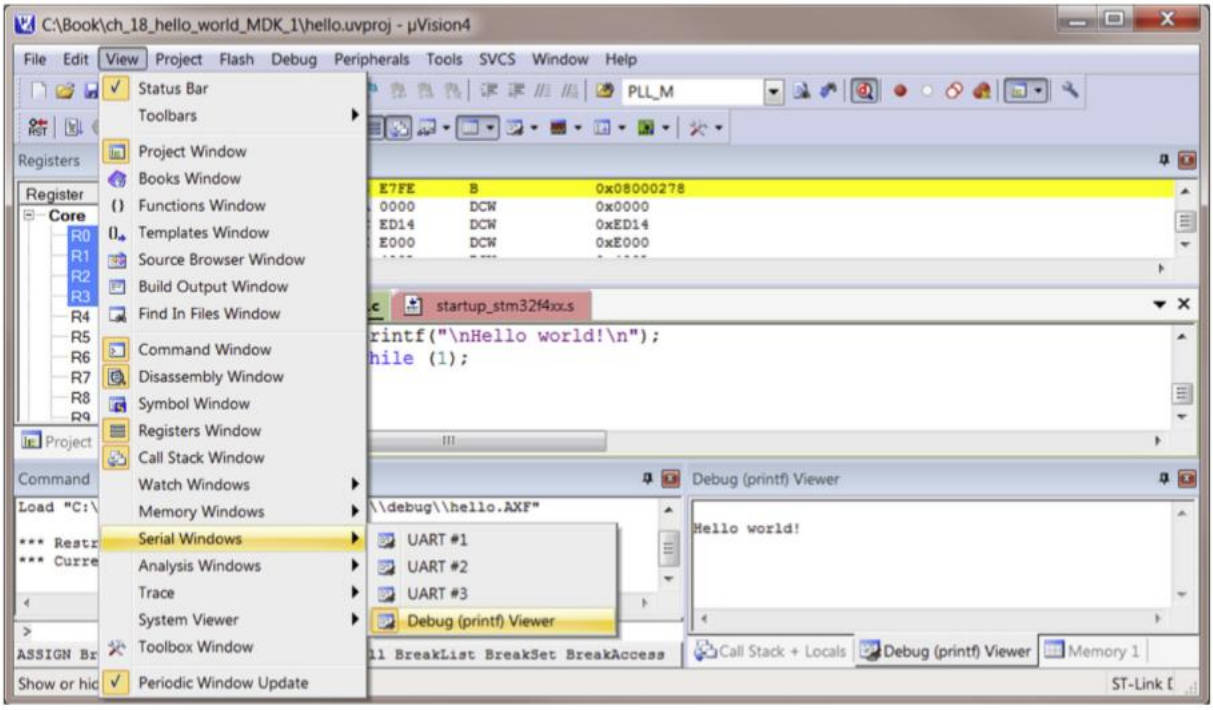
6) Select the options for the target project. To get to the project options dialog click on the project menu button and select '**options for Target XX**' or directly press the project options button. Select the Target tab and select the ARM compiler version 5



7) Create a new file in your area, copy and paste the following code in it. Call it lab0.c

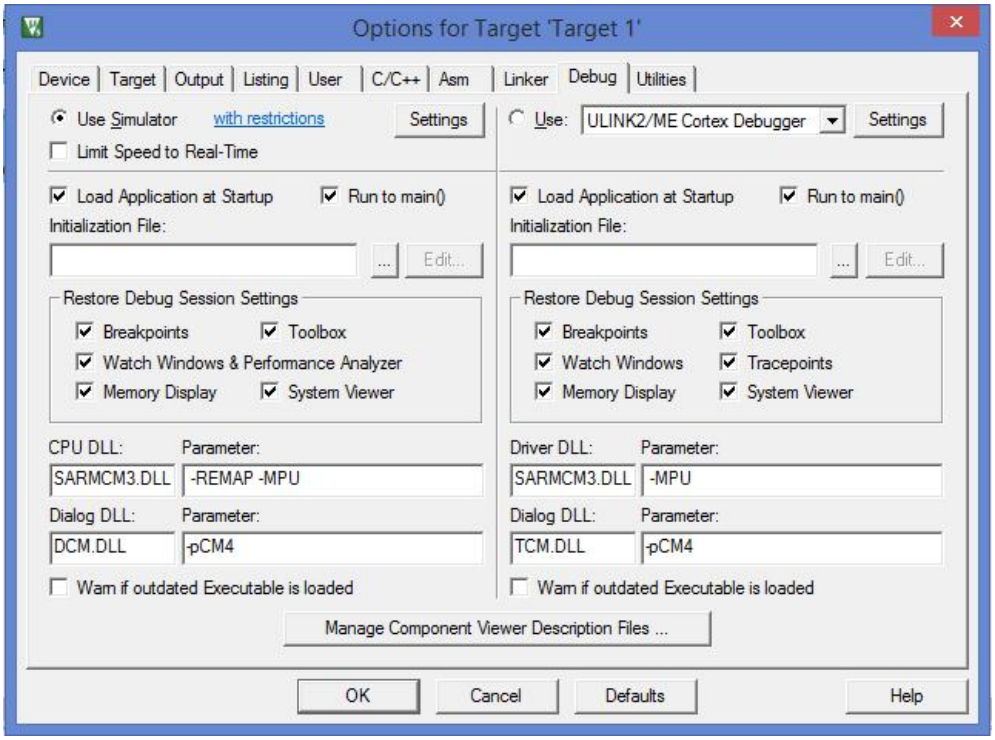
```
#include "stm32f4xx.h"
#include "stdio.h"

  char textbuffer[40] = ""; // Text buffer

int main(void)
{
  printf("\nHello world!\n");
  printf ("textbuffer address = 0x%x \n", (unsigned int) textbuffer);
  return 0;
}
```

8) Add the file to the project – use the **Manage Project Items** button
9) Add the Debug (printf) Viewer to your View

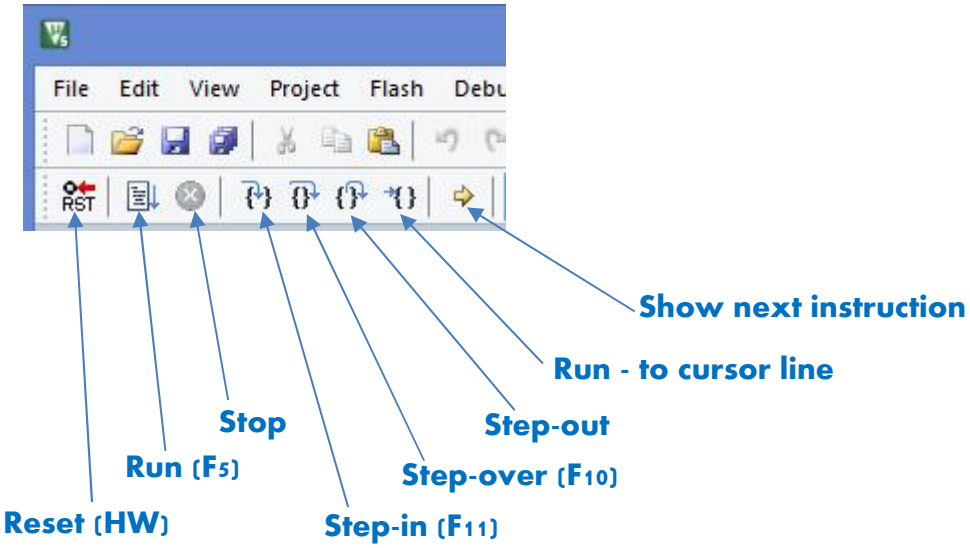


Accessing the Debug (printf) viewer and Hello world message display

10) Make sure you are debugging in simulation mode – remember we don't have hardware to use. Press Options for Target button, then select the debug tab, select the **Use Simulator** radio button.



11) Put a Break Point on the return line. This can be done by a mouse click on the curb on the left of the code, or using Insert/Remove Breakpoint button.
12) Start debug session using Start/Stop Debug Session button, or (Ctrl+F5)
13) Run the debug simulation using Run button or (F5)



14) Check the Debug Viewer, did it show the 'Hello World message'? – I expect no!
15) Strop the debugger

Part3 – Adding support for console input/output

This file is provided, the following is just for reference.

Refer to "The Definitive Guide to ARM Cortex-M3 and Cortex-M4 Processors"

- Chapter 18: Input and Output Examples
  - 18.2 Re-targeting to Instrumentation Trace Macrocell (ITM)

16) Add the following code to a file, call it retarget.c and add it to the project

```
/*****************************************************************************/
/* Retarget functions for ARM DS-5 Professional / Keil MDK                   */
/*****************************************************************************/

#include <stdio.h>
#include <time.h>
#include <rt_misc.h>
#pragma import(__use_no_semihosting_swi)
#include "stm32f4xx.h"

struct __FILE { int handle; /* Add whatever you need here */ };
FILE __stdout;
FILE __stdin;
volatile int32_t ITM_RxBuffer=0x5AA55AA5; // Initialize as EMPTY

int fputc(int ch, FILE *f) {
  return (ITM_SendChar(ch));
}
int fgetc(FILE *f) {
          char tmp;
          while (ITM_CheckChar()==0); // Wait if buffer is empty
          tmp = ITM_ReceiveChar();
          if (tmp==13) tmp = 10;
  return (ITM_SendChar(tmp));
}
int ferror(FILE *f) {
  /* Your implementation of ferror */
  return EOF;
}

void _ttywrch(int ch) {
  ITM_SendChar (ch);
}
void _sys_exit(int return_code) {
label:  goto label;  /* endless loop */
}
```

17) Modify main to include console input/output loop as shown below:

```
#include "stm32f4xx.h"
#include "stdio.h"

 char textbuffer[40] = ""; // Text buffer

int main(void)
{


 SCB->CCR |= SCB_CCR_STKALIGN_Msk; // Enable double word stack alignment
 //(recommended in Cortex-M3 r1p1, default in Cortex-M3 r2px and Cortex-M4)
 printf("\nHello world!\n");
 printf ("textbuffer address = 0x%x \n", (unsigned int) textbuffer);

 while (textbuffer[0] != 'q') {
  printf("Please enter text:");
  fgets(textbuffer, (sizeof(textbuffer)-1), stdin);
  printf("\nYou entered      :%s\n",textbuffer);
 }
}
```

18) Run the simulation, now you should be able to see printf text on the Debug Viewer, you can also input text in the debug viewer to be read by your program.
19) Enter view strings in the Debug Viewer Window and observe it is echoed back.
20) Go to memory1 viewer and use the textbuffer address from your log (0r use 0x20000000) Checkout that the ASCII code of the text you just entered is shown in the memory
21) Repeat the bove 2 steps few times
22) Enter ‘q’ to stop the program.

Note: printf tracing is very helpful in debugging the displaying your program output. I find string input intrusive, may be that’s my opinion. There is another no-intrusive method we will explain later.

Part4 – submission

23) Copy & paste the contents of the Debug Viewer to a file (Ctrl A, Ctrl C, Ctrl V). Name it logfile.txt and save it in the same directory as your code.
24) Stop the debugger. Clean all targets – remove all intermediate files from your project. Project>Clean Targets
25) Close μVision
26) Zip the project directory and submit to Bright Space