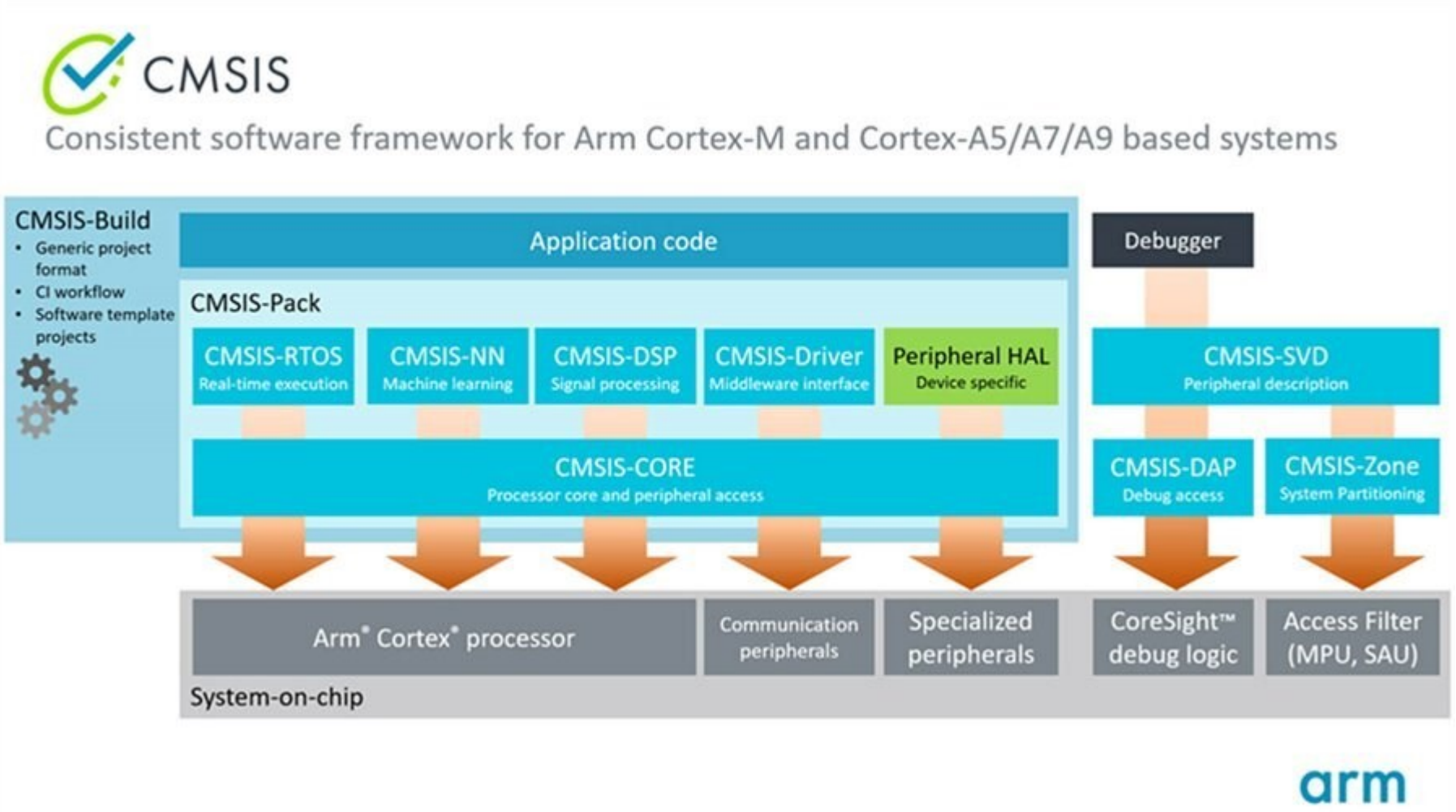
CMSIS-DSP Programming for Mbed OS

CMSIS

- The Cortex Microcontroller Software Interface Standard (CMSIS)
  - is a vendor-independent hardware abstraction layer for microcontrollers that are based on Arm Cortex processors.
  - CMSIS defines generic tool interfaces and enables consistent device support.
    - CMSIS provides interfaces to processor and peripherals, real-time operating systems, and middleware components.
  - The CMSIS software interfaces simplify software reuse, reduce the learning curve for microcontroller developers, and improve time to market for new devices.
- Open source – apache 2.0
  - Current version 5.8
  - Legacy version 4

CMSIS Components



CMSIS: support compiler

- Mainly ARM Compiler
  - ARM Compiler: current version 6
- Partial Support
  - IAR, GCC
- Mbed Studio use ARMC6 compiler
- Mbed CLI uses ARMC6 or GCC

dotproduct_example_f32

- Compute the inner product of two 1-d array

```
float32_t srcA_buf_f32[MAX_BLOCKSIZE] =
{
  -0.4325648115282207,  -1.6655843782380970,  0.1253323064748307,
   0.2876764203585489,  -1.1464713506814637,  1.1909154656429988,
   1.1891642016521031,  -0.0376332765933176,  0.3272923614086541,
   0.1746391428209245,  -0.1867085776814394,  0.7257905482933027,
  -0.5883165430141887,   2.1831858181971011, -0.1363958830865957,
   0.1139313135208096,   1.0667682113591888,  0.0592814605236053,
  -0.0956484054836690,  -0.8323494636500225,  0.2944108163926404,
  -1.3361818579378040,   0.7143245518189522,  1.6235620644462707,
  -0.6917757017022868,   0.8579966728282626,  1.2540014216025324,
  -1.5937295764474768,  -1.4409644319010200,  0.5711476236581780,
  -0.3998855777153632,   0.6899973754643451
};

/* ------------------------------------------------------------------
** Test input data of srcB for blockSize 32
** ------------------------------------------------------------------ */
float32_t srcB_buf_f32[MAX_BLOCKSIZE] =
{
   1.7491401329284098,  0.1325982188803279,   0.3252281811989881,
  -0.7938091410349637,  0.3149236145048914,  -0.5272704888029532,
   0.9322666565031119,  1.1646643544607362,  -2.0456694357357357,
  -0.6443728590041911,  1.7410657940825480,   0.4867684246821860,
   1.0488288293660140,  1.4885752747099299,   1.2705014969484090,
  -1.8561241921210170,  2.1343209047321410,   1.4358467535865909,
  -0.9173023332875400, -1.1060770780029008,   0.8105708062681296,
   0.6985430696369063, -0.4015827425012831,   1.2687512030669628,
  -0.7836083053674872,  0.2132664971465569,   0.7878984786088954,
   0.8966819356782295, -0.1869172943544062,   1.0131816724341454,
   0.2484350696132857,  0.0596083377937976
};
```

- C[i] = A[i]*B[i]

```
arm_mult_f32(srcA_buf_f32, srcB_buf_f32, multOutput, MAX_BLOCKSIZE);
```

- Compute C[0]+C[1]+.....+C[MAX_BLOCKSIZE]

```
for(i=0; i< MAX_BLOCKSIZE; i++)
{
  arm_add_f32(&testOutput, &multOutput[i], &testOutput, 1);
}
```

- Compare the result with that of matlab

```
/* absolute value of difference between ref and test */
diff = fabsf(refDotProdOut - testOutput);

/* Comparison of dot product value with reference */
status = (diff > DELTA) ? ARM_MATH_TEST_FAILURE : ARM_MATH_SUCCESS;
```

- Output

```
SUCCESS
----------------------
multOutput =
-0.756616
-0.220854
0.040762
-0.228360
-0.361051
-0.627935
1.108618
-0.043830
-0.669532
-0.112533
-0.325072
0.353292
-0.617043
3.249836
-0.173291
-0.211471
2.276826
0.085119
0.087739
0.920643
0.238641
-0.933381
-0.286860
2.059896
0.542081
0.182982
0.988026
-1.429069
0.269341
0.578676
-0.099346
0.041130
----------------------
testOutput = 5.927364
----------------------
```

CMSIS-DSP optimization

- loop-unrolling

```
while(blkCnt > 0u)
{
  /* C = A + B */
  /* Add and then store the results in the destination buffer. */

  /* read four inputs from sourceA and four inputs from sourceB */
  inA1 = *pSrcA;
  inB1 = *pSrcB;
  inA2 = *(pSrcA + 1);
  inB2 = *(pSrcB + 1);
  inA3 = *(pSrcA + 2);
  inB3 = *(pSrcB + 2);
  inA4 = *(pSrcA + 3);
  inB4 = *(pSrcB + 3);

  /* C = A + B */
  /* add and store result to destination */
  *pDst = inA1 + inB1;
  *(pDst + 1) = inA2 + inB2;
  *(pDst + 2) = inA3 + inB3;
  *(pDst + 3) = inA4 + inB4;

  /* update pointers to process next samples */
  pSrcA += 4u;
  pSrcB += 4u;
  pDst += 4u;


  /* Decrement the loop counter */
  blkCnt--;
}
```

arm_add_f32() code snippet

- SIMD

```
while(blkCnt > 0u)
{
  /* C = A + B */
  /* Add and then store the results in the destination buffer. */
  inA1 = *__SIMD32(pSrcA)++;
  inA2 = *__SIMD32(pSrcA)++;
  inB1 = *__SIMD32(pSrcB)++;
  inB2 = *__SIMD32(pSrcB)++;

  *__SIMD32(pDst)++ = __QADD16(inA1, inB1);
  *__SIMD32(pDst)++ = __QADD16(inA2, inB2);

  /* Decrement the loop counter */
  blkCnt--;
}
```

arm_add_q15() code snippet

FIR example

- Documentation
  - https://arm-software.github.io/CMSIS_5/DSP/html/group__FIRLPF.html
- Program examples
  - https://github.com/ARM-software/CMSIS_5/tree/develop/CMSIS/DSP/Examples/ARM
  - https://github.com/ARM-software/CMSIS_5/tree/develop/CMSIS/DSP/Examples/ARM/arm_fir_example

FIR Lowpass Filter Example

- Removes high frequency signal components from the input using an FIR lowpass filter.
  - The example demonstrates how to configure an FIR filter and then pass data through it in a block-by-block fashion.

Input signal and lowpass filter design

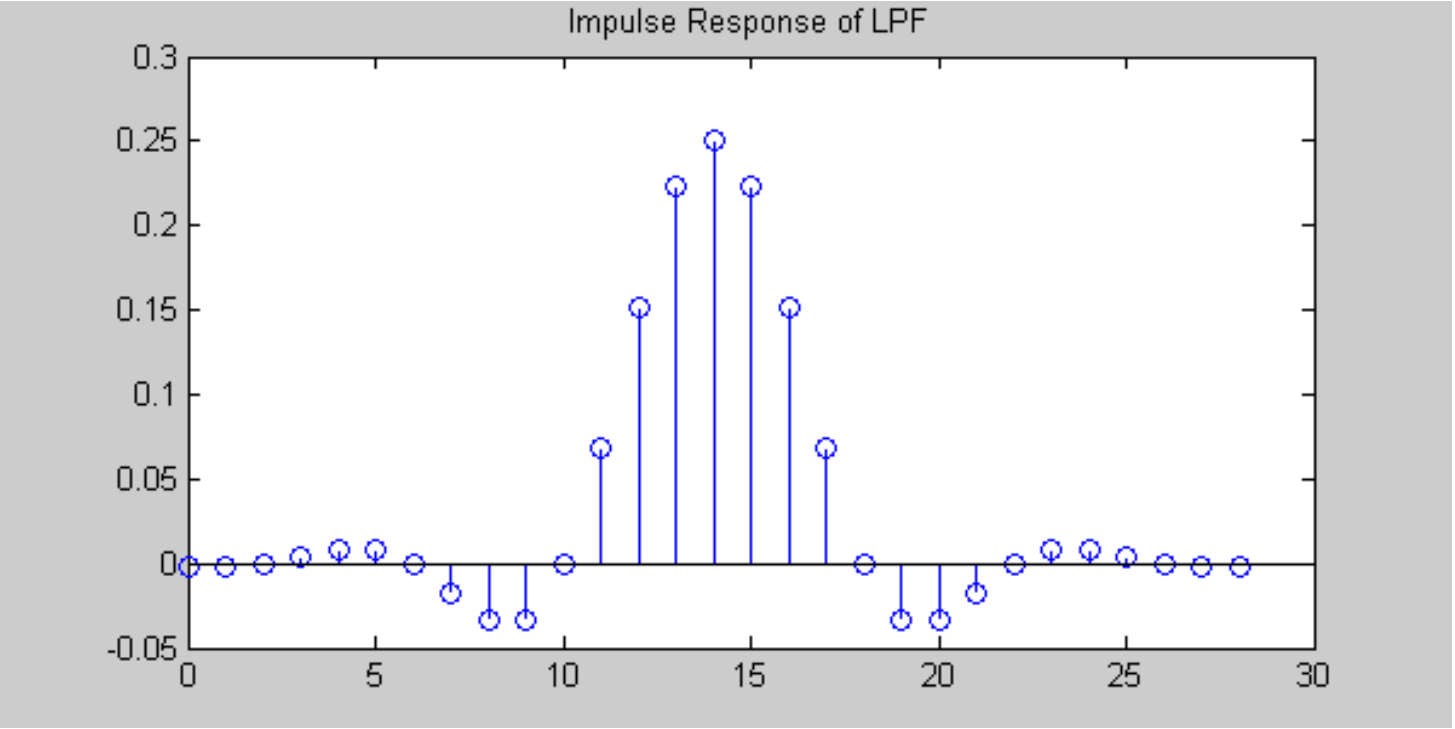
- The input signal is a sum of two sine waves: 1 kHz and 15 kHz.
  - This is processed by an FIR lowpass filter with cutoff frequency 6 kHz.
  - The lowpass filter eliminates the 15 kHz signal leaving only the 1 kHz sine wave at the output.
- The lowpass filter was designed using MATLAB with a sample rate of 48 kHz and a length of 29 points. The MATLAB code to generate the filter coefficients is shown below:

  h = fir1(28, 6/24);
  - The first argument is the "order" of the filter and is always one less than the desired length. The second argument is the normalized cutoff frequency.
  - A 6 kHz cutoff with a Nyquist frequency of 24 kHz lies at a normalized frequency of 6/24 = 0.25.
- The CMSIS FIR filter function requires the coefficients to be in time reversed order.

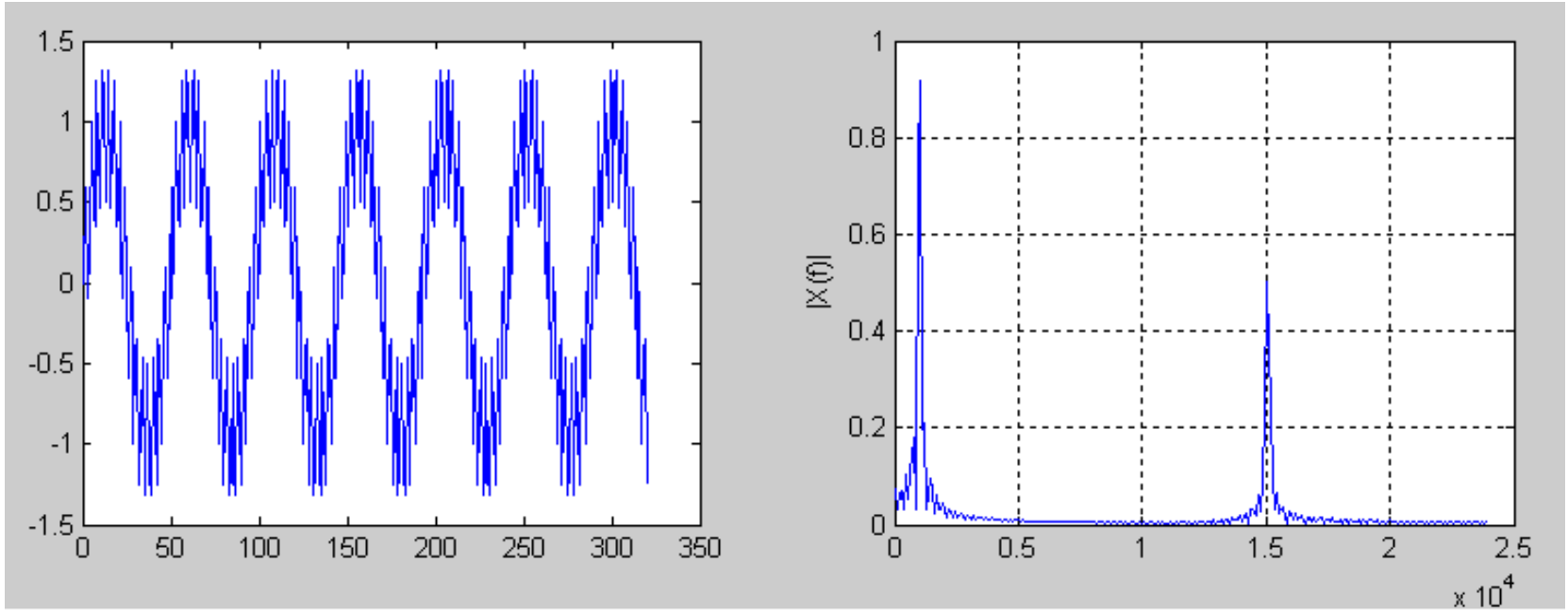
  fliplr(h)

The filter coefficients

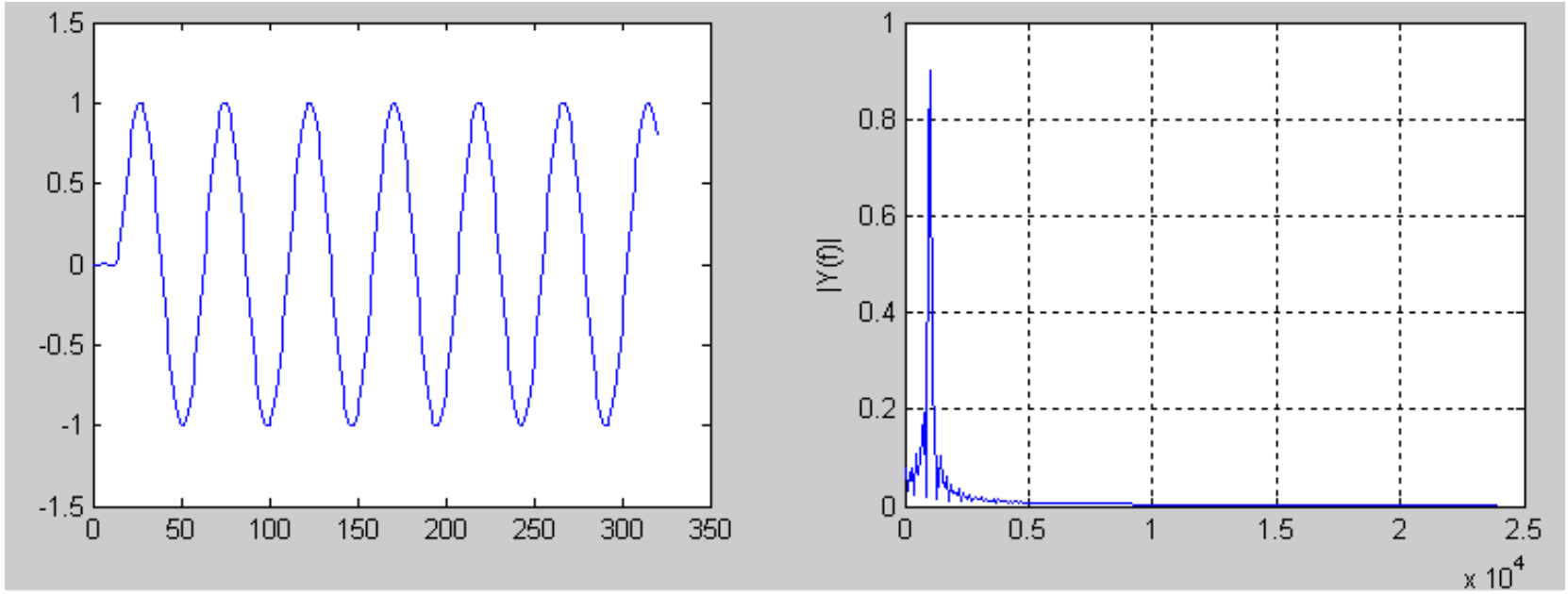
- The resulting filter coefficients and are shown below.
- Note that the filter is symmetric (a property of linear phase FIR filters) and the point of symmetry is sample 14.
- Thus the filter will have a delay of 14 samples for all frequencies.



- Input signal



- Output signal



Test FIR example at Mbed Studio: code modification and add library mbed-dsp

- File -> New Program -> empty Mbed OS program

`arm_fir_example_f32.c`

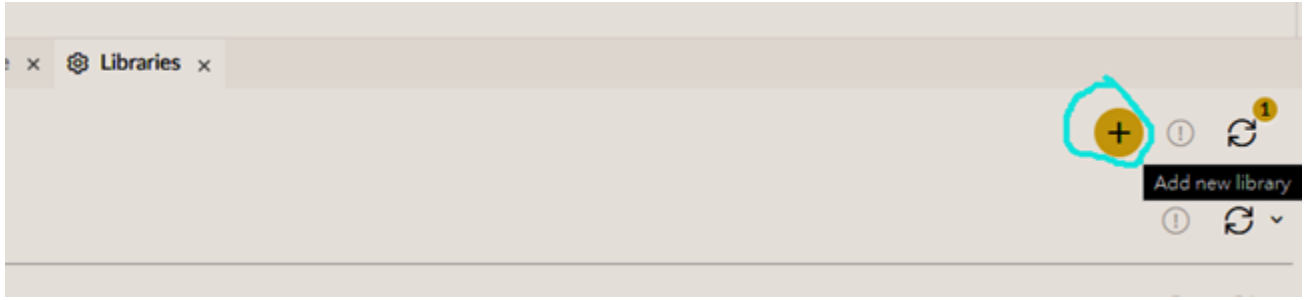
- Replace the content of **main.cpp** with the that of ~~**arm_fir_data.c**~~
  - Add the following line to **main.cpp** before the #include lines

#include "mbed.h“

- Add the file **arm_fir_data.c** to the project
- Add library **mbed-dsp** to the project:

https://os.mbed.com/teams/mbed-official/code/mbed-dsp

- Add a #define to the file **arm_bitreversal2.S** (as shown in next slide)

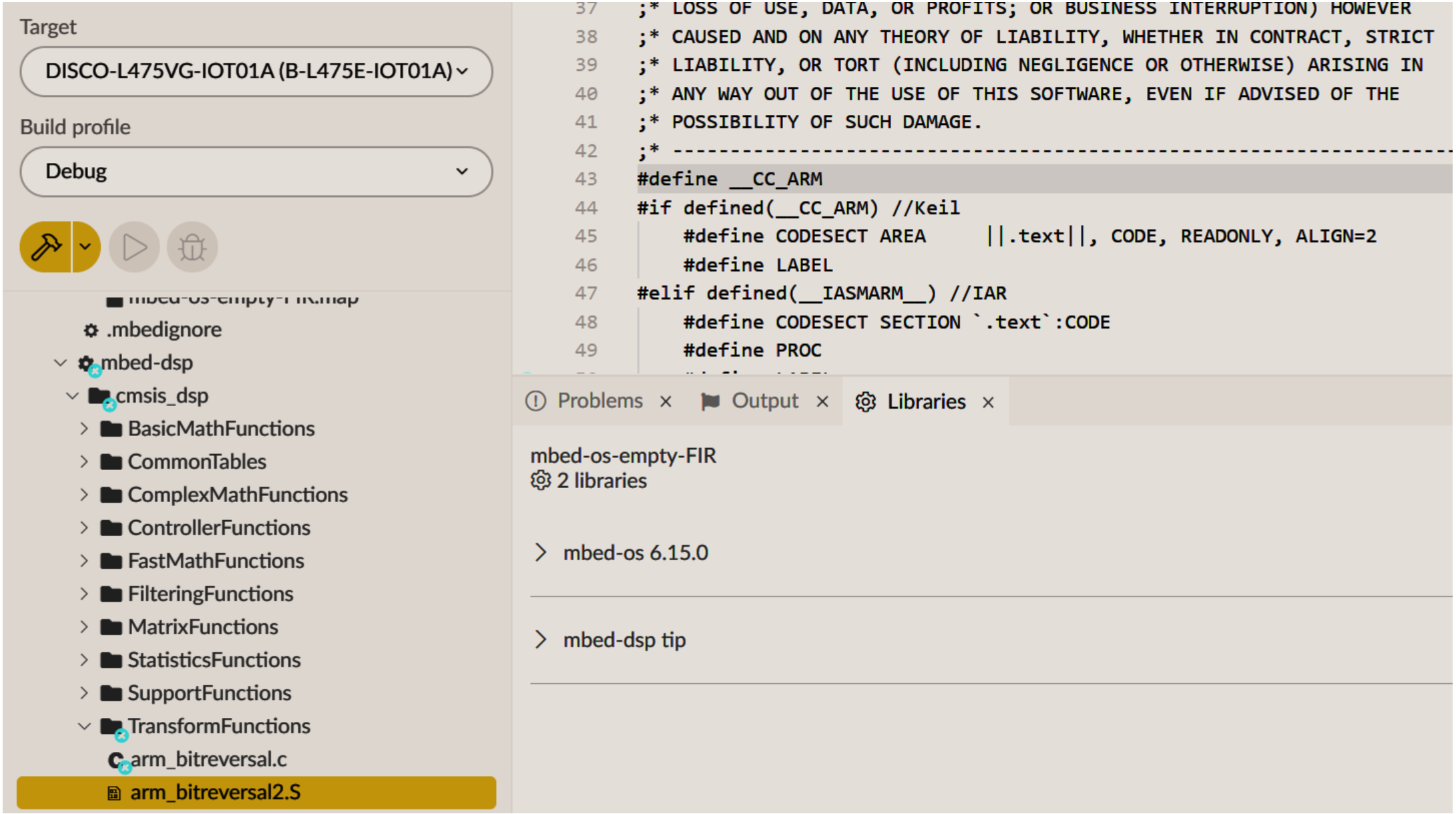


mbed-dsp/cmsis_dsp/TransformFunctions

- Before line 43, add the following line: (arm_bitreversal2.S)

#define __CC_ARM

```
43 #if defined(__CC_ARM) //Keil
44     #define CODESECT AREA     ||.text||, CODE, READONLY, ALIGN=2
45     #define LABEL
46 #elif defined(__IASMARM__) //IAR
47     #define CODESECT SECTION `.text`:CODE
48     #define PROC
49     #define LABEL
50     #define ENDP
51     #define EXPORT PUBLIC
52 #elif defined (__GNUC__) //GCC
53     .syntax unified
54     .cpu cortex-m4
55     .fpu softvfp
56     #define THUMB .thumb
57     #define CODESECT .section text
58     #define EXPORT .global
59     #define PROC :
60     #define LABEL :
61     #define ENDP
62     #define END
63 #endif
```


Target
DISCO-L475VG-IOT01A (B-L475E-IOT01A)
Build profile
Debug
.mbedignore
mbed-dsp
cmsis_dsp
BasicMathFunctions
CommonTables
ComplexMathFunctions
ControllerFunctions
FastMathFunctions
FilteringFunctions
MatrixFunctions
StatisticsFunctions
SupportFunctions
TransformFunctions
arm_bitreversal.c
arm_bitreversal2.S
;* LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
;* CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
;* LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
;* ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
;* POSSIBILITY OF SUCH DAMAGE.
#define __CC_ARM
#if defined(__CC_ARM) //Keil
#define CODESECT AREA ||.text||, CODE, READONLY, ALIGN=2
#define LABEL
#elif defined(__IASMARM__) //IAR
#define CODESECT SECTION `.text`:CODE
#define PROC
Problems
Output
Libraries
mbed-os-empty-FIR
2 libraries
mbed-os 6.15.0
mbed-dsp tip