# Generating TI Hercules MCU optimized code from Simulink

## CMSIS Library

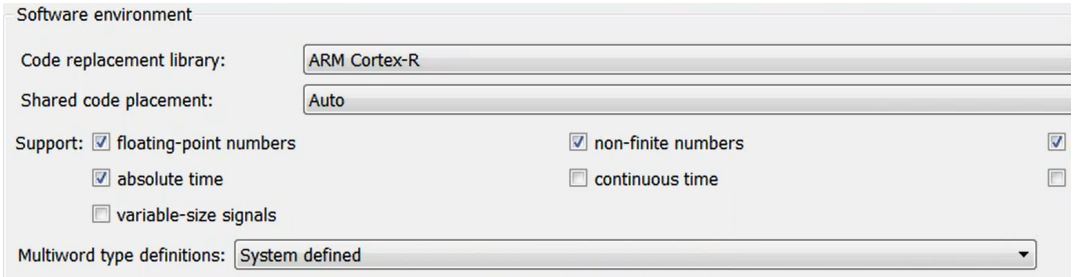
- `CMSIS` stands for `ARM's Cortex Microcontroller Software Interface Standard`
- It is used to replace equivalent function in Simullink
  - The CMSIS library is free to download as `Cortex-R4 CMSIS DSP Library` at **https://www.ti.com/tool/HERCULES-DSPLIB**
- 60+ functions covering
  - vector operations
  - matrix computing
  - complex arithmetic
  - filter functions
  - control functions
  - PID controller
  - Fourier transforms
  - many other frequently used DSP algorithms
  - ARM DSP SIMD (Single Instruction Multiple Data)

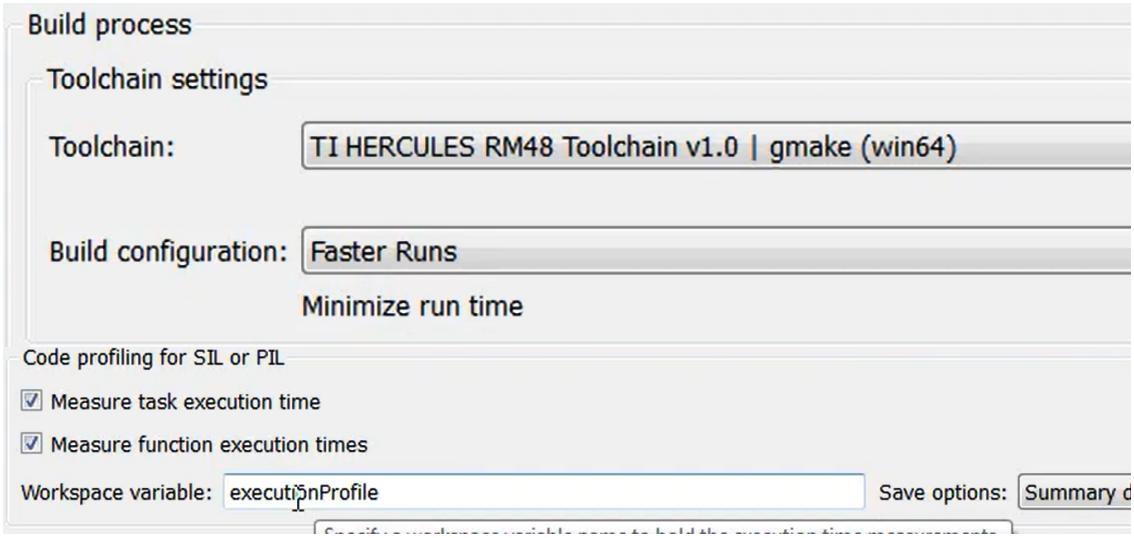
## Use of CMSIS Library with code replacement table for optimization

- **CRL**
  - Generate **Code Replacement Library (CRL)**
    - `CMSIS.m`

```
%---------- entry: sin -----------
hEnt = RTW.TflCFunctionEntry;
hEnt.setTflCFunctionEntryParameters( ...
          'Key', 'sin', ...
          'Priority', 100, ...
          'ImplementationName', 'arm_sin_f32', ...
          'ImplementationHeaderFile', 'arm_math.h', ...
          'ImplementationSourceFile', 'arm_sin_f32.c', ...
          'ImplementationHeaderPath', 'C:\ti\Hercules\Cortex-R4 CMSIS DSP Library\1.0.0\Include',
          'ImplementationSourcePath', 'C:\ti\Hercules\Cortex-R4 CMSIS DSP Library\1.0.0\Source\Fa
          'GenCallback', 'RTW.copyFileToBuildDir');
```

    - Replace sin/cos with corresponding algorithm in the SIMSIS library
  - Add `CMSIS.m` to the matlab path and do a `sl_refresh_customization` which register the CRL
  - Select the CRL for the model



- **Toolchain**
- 

- About the board
  - USB connection contains emulation connection for launching simulation and a UR there for USB virtual COM port provides serial port for communication with Simulink
  - `loadti` : launcher
- Purpose: verify the numerical equivalence the embedded target that running your algorithm and the algorithm in Simulink

# Setup for RM48

- To use the library and code generated from `HalCoGen` , I should follow this **tutorial**
- This will make help available
  - Including the required packages
  - **Making customizations** for different board in Hercules family
    - Need to take a look at ConnectivityConfig.m
      - PIL target is a ConnectivityConfig object
        - See <installdir>\TI_HERCULES_RM48\+TI_HERCULES_RM48Pil\ConnectivityConfig.m
    - I can configure following in the file
      - **Builder**
      - **Launcher**
      - **Communicator**
      - **Timer (for Profiling)**
    -

```
methods
    %% ConnectivityConfig Constructor
    function this = ConnectivityConfig(componentArgs)
        % Need to call the superclass (rtw.connectivity.Config)
        % constructor method.  It requires several parameters that
        % we need to construct first however:
        %  - builder (rtw.connectivity.MakefileBuilder)
        %  - launcher (rtw.connectivity.Launcher)
        %  - communicator (rtw.connectivity.RtIOStreamHostCommunicator)
        builder = newBuilder(componentArgs);
        launcher = newLauncher(componentArgs, builder);
        communicator = newCommunicator(componentArgs, launcher);
        this@rtw.connectivity.Config(componentArgs, ...
                                     builder,launcher,communicator);

        % Register PMU function from HalCoGen as Profiling Timer.
        % See TI_TMS570\utils\cr4pmu_crl.m
        timer = cr4pmu_crl;
        this.setTimer(timer);
    end
end
```

- **Builder**
  - The way that matlab will generate a file for you

```
function builder = newBuilder(componentArgs)
    ext = '.out';
    fw = TI_HERCULES_RM48Pil.TargetApplicationFramework(componentArgs);
    builder = rtw.connectivity.MakefileBuilder(componentArgs, fw, ext);
end
```

    All the make utility, register the tool chain that I am using with Matlab, which is CCS

  - `TI_HERCULES_RM48_toolchain.m`

```
CGTPATH = getpref('TI_HERCULES_RM48', 'CGTPATH');
CGTPATH=regexprep(CGTPATH,'\\','/');
TI_HERCULES_RM48_TARGETDIR = getpref('TI_HERCULES_RM48', 'TARGETDIR');
TARGET_SOURCE = fullfile(TI_HERCULES_RM48_TARGETDIR, ...
                         'src', ...
                         'TI_HERCULES_RM48PilSerial');
TARGET_SOURCE=regexprep(TARGET_SOURCE,'\\','/');
CCSPATH = getpref('TI_HERCULES_RM48', 'CCSPATH');
CCSPATH=regexprep(CCSPATH,'\\','/');
tc.addMacro('CGTPATH',       CGTPATH);
tc.addMacro('TARGETTOOLSMKDIR',       TI_HERCULES_RM48_TARGETDIR);
tc.addMacro('TARGET_SOURCE',       TARGET_SOURCE);
tc.addMacro('CCSROOTDIR',       CCSPATH);
tc.addMacro('TI_TOOLS',      '$(CGTPATH)/bin');
tc.addMacro('TI_INCLUDE',    '$(CGTPATH)/include');
tc.addMacro('TI_LIB',        '$(CGTPATH)/lib');
tc.addMacro('CCOUTPUTFLAG',  '--output_file=');
tc.addMacro('LDOUTPUTFLAG',  '--output_file=');
tc.addMacro('EXE_FILE_EXT',  '$(PROGRAM_FILE_EXT)')
tc.addMacro('ASAP2_PERL_PATH', '$(MATLAB_ROOT)/toolbox/target/extensions/processor/tic2000/asap2/asap2post.pl');
```

    These used to be the options in the make file

  - Instead of creating a make file or a template makefile, now you create this tool chain registration. And the actual make file is created on the flag. But it uses the option here. Eg. if I want to change big/little endian, I change compiler assemble options in this file
  - See more detail in `making customization help` page.
  - Framework

- Second part of the builder

```
function this = TargetApplicationFramework(componentArgs)

    % Call superclass constructor
    this@rtw.pil.RtIOStreamApplicationFramework(componentArgs);

    % Add the target side version of PIL main()
    this.addPILMain('target');

    % Additional sources from HalCoGen Driver Generator
    addHalCoGenSources(this);

    % rtIOStream Customizations (open, close, send, recv)
    addRTIOStreamSources(this);

end
```

  - Add HalCoGen sources and RTIO stream sources to this build
  - What is target application framework, I can't directly run main function on the target without the code to initialize the target so I have to set the clock frequency, interrupt etc. These all has to be configured before main function. And create a run time environment. And also set up the C runtime environment. `halcogen` will generate the startup code for any of the hercules product.
  - The serial port needs to be configured
  - Also I need to enable the interrupts in the VIM
  - Serial IO stream communication. We needs to provide c function for openning, closing the serial ports, sending and receiving

- **Launcher**

  -

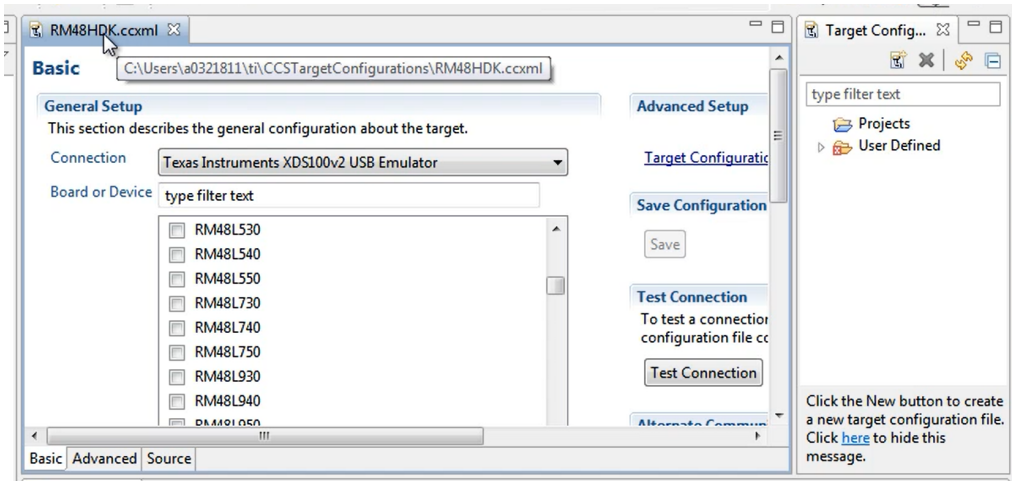
```
% Start the application
function startApplication(this)

    % get name of the executable file to download
    programFile = this.getBuilder.getApplicationExecutable;

    disp(sprintf('Begin communication with RM48'));

    manualDownload = false;
    if ( ~manualDownload )

        disp(sprintf('Launching loadti to download %s', ...
            programFile));

        % Use customized version of loadti script that ships
        % with ccs.  'tweek' to ensure the TMS570 target
        % runs freely and isn't halted after loader exits.
```

  - Here use a utility called `loadti`, which comes as a debug server scripting. And I need to know what target configuration file to use

  -

```
% Get path information from MATLAB Preferences
CCSPATH = getpref('TI_HERCULES_RM48','CCSPATH');
TARGETDIR = getpref('TI_HERCULES_RM48','TARGETDIR');
TCONF = getpref('TI_HERCULES_RM48','TCONF');
```

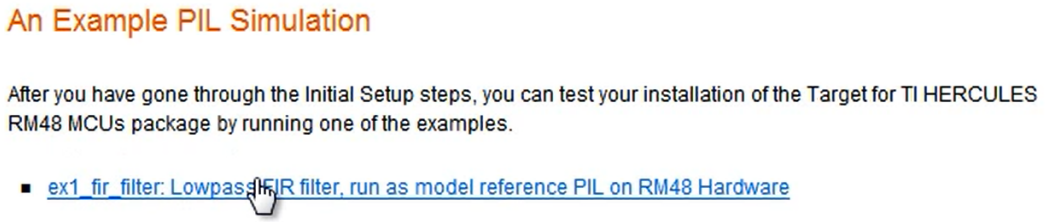
  - target configuration file -> I have to generate a CCXML file `TCONF` needs to be changed.




- **Communicator**

```
function communicator = newCommunicator(componentArgs, launcher)
    hostlib = 'libmwrtiostreamserial.dll';

    communicator = rtw.connectivity.RtIOStreamHostCommunicator(...
        componentArgs, launcher, hostlib);

    communicator.setInitCommsTimeout(20);

    COMPORT = getpref('TI_HERCULES_RM48', 'COMPORT');
    COMBAUD = getpref('TI_HERCULES_RM48', 'COMBAUD');

    communicator.setOpenRtIOStreamArgList({'-baud', COMBAUD, ...
                                           '-port', COMPORT});

end
```

- **Timer for Profiling**
  - Provide a function to read the timestamp
  - `cr4pmu_crl` : code replacement library that replace the `code_profile_read_timer` with `_pmuGetCycleCount_`
- Then get following example to run




# Appendix

## Hardware Configuration

> **ANSI C**, **ISO C** and **Standard C** are successive standards for the **C programming language** published by the **American National Standards Institute** (ANSI) and the **International Organization for Standardization** (ISO). Historically, the names referred specifically to the original and best-supported version of the standard (known as **C89** or **C90**). Software

> developers writing in C are encouraged to conform to the standards, as doing so helps **portability** between compilers.

## Processor in the loop

> In **processor-in-the-loop (PIL)** simulation, code generated from a Simulink® model runs directly on the target hardware, which means you can test models on the hardware using the same test cases as on the host. PIL tests are designed to expose problems with execution in the embedded environment.

## Profiling report

type `executionProfile.report` command in matlab

- It gives cycle time in ticks