Peripheral Configuration Guide

Objective: The objective of this document is to outline the peripheral configuration in MCC (MPLAB® Code Configurator) that are required for use with motorBench® Development Suite. Sample projects for MCHV-2 (dsPICDEM™ MCHV-2 Development Board) and MCLV-2 (dsPICDEM™ MCLV-2 Development Board) are available, but this information allows someone to utilize motorBench® Development Suite without the sample projects. motorBench® Development Suite only supports dsPIC33EP256MC506 device and cannot be guaranteed to work with other MCC configurations. One can modify the configuration outlined, if they are experienced with MCC and/or motorBench® Development Suite.

Peripherals Supported by MCC: For a peripheral to be fully supported by MCC it must contain all the functionality currently being used in the MCAF (Motor Control Application Framework) application. For example: for a UART module to be fully supported by MCC it must contain features that can transmit data, receive data, check whether a byte can be written to transmit buffer, and get the status of receive.

Project Resources: The following modules need to be added to the project resources window

Peripherals

- UART1
- TMR1
- TMR2
- DMA
- ADC1
- PWM

Libraries

- 2 Switch Modules
- 2 LED Modules

System

Interrupt Module

- Global Interrupts Enabled
- PWM Special Event Match Enabled
- TMR2 Timer 2 Interrupt Enabled
- All Enabled Interrupt priorities set to 4
- Note: If a user would like to add other peripheral interrupts, they must set the priority of those interrupts lower than 4

System Module

- Clock set to Primary Oscillator at 8 MHz
- PLL Enabled

- Fosc/2 set to 70 MHz
    - Prescaler set to 1:2
    - Feedback set to 1:70
    - Postscaler set to 1:2
        - Note: These settings result in 70 MHz but any settings resulting in 70MHz are valid
- Enable clock switching
- Emulator Pin Placement set to PGEC2 and PGED2
- Watchdog timer period set to 1ms
    - Note: You may not be able to get 1ms exactly, that is fine ±30µs is allowable
- Using the Registers tab
    - In FOSC register set pin IOL1WAY to Allow multiple configurations
    - In FOSCSEL register set pin PWMLOCK to PWM registers may be written without key sequence

Peripherals

ADC Module

- Enable Auto sampling
- Conversion clock source set to Fosc/2
- Conversion clock set to 7
- Acquisition time set to 0
- Resolution mode set to 10 bits
- CH0 Negative inputs set to AVSS
- Conversion Trigger set to PWM Primary Special Event Trigger
- Output format set to Fractional result, signed, left-justified
- Positive voltage ref set to AVDD
- Negative voltage ref set to AVSS
- Channel Select set to 4 channels
- Sampling mode set to Simultaneous
- CH123 Positive Inputs set to CH1=0A2/AN0, CH2=AN1, CH3= AN2
- CH123 Negative Inputs set to CH1=VREF-, CH2=VREF-, CH3=VREF-
- DO NOT
    - Enable the ADC Interrupt
    - Enable ISR Generation
    - Note: ADC Interrupt is enabled during runtime at a specific time to avoid early ADC triggers

DMA Module

- DMA module must be included but the user is free to configure however he/she would like

PWM Module

- 3 PWM Generators are required (settings are the same for all 3)
    - Enable PWM Special Event Match Interrupt
        - Output Postscaler set to 2
        - Compare Count set to 1

- Duty Cycle set to Primary
- Independent Time Base Mode set to Primary
- PWM Mode set to center aligned
- PWM Period set to 50µs
- Dead Time Control
  - mode set to Positive dead time for all output modes
  - set deadtime to a minimum of 2µs
    - eg: 2µs with center aligned PWM and 70MIPS Fcy will mean a dead time in counts of 280 (0x 118)
    - Note: deadtime value must also be changed in motorBench®
- Fault Control
  - mode set to PWMxH, PWMxL pins to FLTDAT values-latched
  - signal source set to FLT32
  - Fault Value set to PWMxL Low PWMxH Low
- I/O Control
  - PWMxH Control
    - Output Pin Ownership Enabled
    - Override Enable Disabled
  - PWMxL Control
    - Output Pin Ownership Enabled
    - Override Enable Disabled
  - PWM I/O Pins mode set to Complementary Output Mode
  - Override value set to PWMxL Low PWMxH Low
  - Swap disabled
- Using the Registers tab
  - In FCLCONx register Enable FLTPOL
    - Note: Fault Polarity bit for PWM Generator #, this is used to set the selected fault source to active-low
  - In PTCON register:
    - SEIEN Enabled
      - Note: Special Event Interrupt Enable bit is what triggers the ADC to start conversion
    - SYNCOEN Enabled
      - Note: Primary Time Base Sync Enable bit used to synchronize all PWM generators with each other
    - SYNCPOL Enabled
      - Note: Synchronize Input and Output Polarity bit used for troubleshooting PWM signals

TMR Module

- 2 Timers are needed
- TMR1
  - Clock source set to Fosc/2
  - Prescaler set to 1:1
  - Period Count set to 0xFFFF
  - disable TMR
  - Do not Enable Timer Interrupt

- TMR2
    - Disable TMR
    - TMR2 Interrupt Enabled
    - TMR2 set to 16-bit mode
    - TMR2 period set to 1ms

UART Module

- Select Foundation Services UART Module
- Enable UART
- Set any baud rate
- Parity set to None
- Data bits set to 8
- Stop bits is 1
- Flow control none

Libraries

Switch Modules

- Set Custom Name of one Switch module to BSP_BUTTON_GP1
- Set Custom Name of one Switch module to BSP_BUTTON_GP2
- Set both switches to active low

LED Modules

- Set Custom Name of one LED module to BSP_LED_GP1
- Set Custom Name of one LED module to BSP_LED_GP2

Pin Module and Pin Manager: Grid View

The Pin Module is a window where one can see the overall pin placement of their configuration. Here one can change the Custom Name of their pin, see what module that pin belongs to, see the function of that pin, and many other features.

The Pin Manager: Grid View is a window where one maps the pins to a specific peripheral, or other module. In this view one can also see how peripheral modules map to a GPIO, the function of that pin, and the direction of that pin (output or input). The green lock symbol represents a peripheral module "owning" that pin, and the blue unlocked symbol represents a pin that is "unowned" by a peripheral.

The table below shows how to setup the Pin Module/Pin Manager to be compatible with motorBench® Development Suite. One must map the peripheral pins to the general purpose pins (Pin Name) in the Pin Manager: Grid View and set the Custom Name in the Pin Module.

Pin Module/Pin Manager

Pin Name	Module	Custom Name	Direction
RA0	ADC1	BSP_MOTOR1_PHASEB	Input
RA1	ADC1	BSP_MOTOR1_PHASEA	Input
RA8	Pin Module	BSP_PORT_QEI1_PHASEA	Input
RA12	ADC1	BSP_MOTOR1_DCLINK	Input
RB0	ADC1	BSP_MOTOR1_SUMPHASE	Input
RB4	PWM	N/A	Input
RB5	ICD	N/A	Input

RB6	ICD	N/A	Input
RB10	PWM	N/A	Output
RB11	PWM	N/A	Output
RB12	PWM	N/A	Output
RB13	PWM	N/A	Output
RB14	PWM	N/A	Output
RB15	PWM	N/A	Output
RC5	UART1	N/A	Input
RC6	Pin Module	BSP_PORT_QEI1_PHASEB	Input
RC8	Pin Module	BSP_TESTPOINT_GP2	Output
RC9	PWM	N/A	Output
RC10	Pin Module	BSP_TESTPOINT_GP4	Output
RC12	INTERNAL OSCILLATOR	N/A	Output
RD5	OnOff_LED2	BSP_LED_GP2	Output
RD6	OnOff_LED1	BSP_LED_GP1	Output
RD8	Pin Module	BSP_TESTPOINT_GP1	Output
RE13	ADC1	BSP_MOTOR1_POT	Input
RF0	Pin Module	BSP_PORT_QEI1_INDEX	Input
RF1	UART1	N/A	Output

Pin Module/Pin Manager (MCHV-2)

Pin Name	Module	Custom Name	Direction
RB8	Switch2	BSP_BUTTON_GP2	Input
RB8	Switch1	BSP_BUTTON_GP1	Input

Note: in Pin Module view only one custom name will show per pin but, in Pin Manager: Grid View be sure to set both switches to the same pin. This will set both switches with their custom names mapped to the RB8 pin.

Pin Module/Pin Manager (MCLV-2)

Pin Name	Function	Custom Name	Direction
RG6	Switch2	BSP_BUTTON_GP2	Input
RG7	Switch1	BSP_BUTTON_GP1	Input

Required Compiler Settings

- **Optimization:**
  - *-O1* or greater; *-O0* and *-Os* will both compile without errors but do not execute fast enough to complete within the 50 microsecond ADC ISR. Note: at higher optimization levels, in-circuit debugging using MPLAB X will behave unreliably with respect to breakpoints and single-stepping through C code.
  - The "Omit frame pointer" and "Unroll loops" settings must be enabled.
- **Memory model:**
  - Large data model (handles using pointers, not direct addressing, to allow for more than 8K of program variables)
  - Small scalar model
- **Additional options:** *-Wno-volatile-register-var -finline*

- **Test harness:** In order for the test harness to be enabled, the symbols MCAF_TEST_PROFILING and MCAF_TEST_HARNESS should be defined.

MCC API List: This list contains all the MCC APIs currently being used in MCAF. Note: These APIs may change in the future.

ADC

- Typdef enum ADC1_CHANNEL
- inline static void ADC1_ChannelSelectSet(ADC1_CHANNEL channel)
- inline static uint16_t ADC1_Channel0ConversionResultGet(void)
- inline static uint16_t ADC1_Channel1ConversionResultGet(void)
- inline static uint16_t ADC1_Channel2ConversionResultGet(void)
- inline static uint16_t ADC1_Channel3ConversionResultGet(void)
- inline static void ADC1_InterruptFlagClear(void)
- inline static void ADC1_Enable(void)
- inline static void ADC1_InterruptEnable(void)
- inline static void ADC1_InterruptPrioritySet(uint16_t priorityValue)

Libraries

- static inline bool BSP_BUTTON_GP1_IsPressed()
- static inline bool BSP_BUTTON_GP2_IsPressed()
- static inline void BSP_LED_GP1_On()
- static inline void BSP_LED_GP1_Off()
- static inline void BSP_LED_GP2_On()
- static inline void BSP_LED_GP2_Off()

DMA

- inline static bool DMA_IsRequestCollision(uint16_t dmaChannel)
- inline static bool DMA_IsPeripheralWriteCollision(uint16_t dmaChannel)

Pin Manager

- BSP_TESTPOINT_GP1_SetHigh()
- BSP_TESTPOINT_GP1_SetLow()
- BSP_TESTPOINT_GP2_SetHigh()
- BSP_TESTPOINT_GP2_SetLow()

PWM

- typedef enum PWM_GENERATOR
- inline static void PWM_PeriodCenterAlignedModeSet(PWM_GENERATOR genNum,uint16_t period)
- inline static void PWM_DeadTimeCenterAlignedModeSet(PWM_GENERATOR genNum,uint16_t deadtime)
- inline static void PWM_FaultInterruptEnable(PWM_GENERATOR genNum)
- inline static void PWM_FaultInterruptStatusClear(PWM_GENERATOR genNum)
- inline static bool PWM_FaultInterruptStatusGet(PWM_GENERATOR genNum)
- inline static void PWM_TriggerDividerSet(PWM_GENERATOR genNum,uint16_t trigDivValue)
- inline static void PWM_TriggerStartDelaySet(PWM_GENERATOR genNum,uint16_t delayValue)

- inline static void PWM_TriggerCompareValueSet(PWM_GENERATOR genNum,uint16_t trigCompValue)
- inline static void PWM_TriggerInterruptEnable(PWM_GENERATOR genNum)
- inline static void PWM_OverrideLowDisable(PWM_GENERATOR genNum)
- inline static void PWM_OverrideDataSet(PWM_GENERATOR genNum,uint16_t overrideData)
- inline static void PWM_OverrideLowEnable(PWM_GENERATOR genNum)
- inline static void PWM_OverrideHighDisable(PWM_GENERATOR genNum)
- inline static void PWM_OverrideDataSet(PWM_GENERATOR genNum,uint16_t overrideData)
- inline static void PWM_OverrideHighEnable(PWM_GENERATOR genNum)
- inline static void PWM_DutyCycleSet(PWM_GENERATOR genNum,uint16_t dutyCycle)
- inline static void PWM_FaultModeLatchDisable(PWM_GENERATOR genNum)
- inline static void PWM_FaultModeLatchEnable(PWM_GENERATOR genNum)
- inline static void PWM_ModuleEnable(void)
- inline static void PWM_PrimaryMasterPeriodSet(uint16_t period)

TMR

- uint16_t TMR1_Counter16BitGet(void)
- void TMR1_Start(void)
- void TMR2_Start(void)
- TMR2_GetElapsedThenClear
- void TMR2_Stop(void)

UART

- Typedef enum UART1_STATUS
- void UART1_Initialize(void)
- void UART1_Write(uint8_t byte)
- uint8_t UART1_Read(void)
- bool UART1_IsRxReady(void)
- UART1_STATUS __attribute__((deprecated)) UART1_StatusGet (void)

Watchdog

- inline static void WATCHDOG_TimerSoftwareEnable(void)
- inline static void WATCHDOG_TimerClear(void)

MCC Notifications

- HINT level notifications and below can be ignored
- Note: These notifications may go away in a future release of MCC