Final Project: PIC Brushless DC Motor Controller DRAFT V2

Nate Hoffman and DJ Stahlberger
Rowan University

December 22, 2018

1 Design Overview

The goal of the final project was to design a system that took in real world data, analyzed that data, controlled something as a result of that data, and communicate with either a computer or through wireless means. The system that was created was a Motor Controller for a brushless DC (BLDC) motor capable of communicating over a Controller Area Network (CAN) using Microchip's dsPIC33EV256GM102 and CAN transceivers.

On a top level view, the system reads a potentiometer value and the motor will spin at a speed consistent with that value. This works with the motor under load so that it can actually be used. This can be achieved due to the nature of the closed loop system under which the motor is operating. The motor reports its speed back to the microcontroller and there it is determined whether the motor needs to spin faster or slower. This will be explained in more detail later in the report.

1.1 Design Features

Due to CAN and closed loop system feedback control, here are the design features:

- Closed loop speed control of a BLDC motor
- CAN bus communication
- Variable speed control with a potentiometer

1.2 Featured Applications

- Electric Vehicles
- Industrial Motor control

- CAN communication in automotive use
- Potential Wireless Communication

1.3 Design Resources

Here are a few links to the GitHub repository and a few useful documents:

- GitHub Code Repository
- dsPIC33EV256GM102 Microcontroller DataSheet
- TC4422A MOSFET Gate Driver DataSheet
- FQP27P06 PMOS Datasheet
- FQP30N06L NMOS Datasheet
- MCP2561 CAN Transceiver Datasheet
- Microchip AN885 - Brushless DC (BLDC) Motor Fundamentals
- Microchip AN898 - Determining MOSFET Driver Needs for Motor Drive Applications

Here are the emails of the authors:

- DJ Stahlberger: stahlberd8@students.rowan.edu
- Nate Hoffman: hoffmann4@students.rowan.edu

1.4 Design Inspiration

The inspiration for this project stems from the Rowan University Formula Electric Clinic. That clinic is student run with the goal of building a fully-functional race ready formula style electric race car. The timeline is 2 years starting in the Fall of 2018. The clinic is split into two parts, a mechanical side and an electrical side. This design was created with research intent towards the electrical side. Since the car is being built from the ground up, some of the parts and components need to be tested in house. This design is one of those test models and attempts to see if a motor controller could be created in house by the members of the team. Hopefully this design will provide useful information that can be used by members of the Formula Electric team.

1.5 Block Diagram

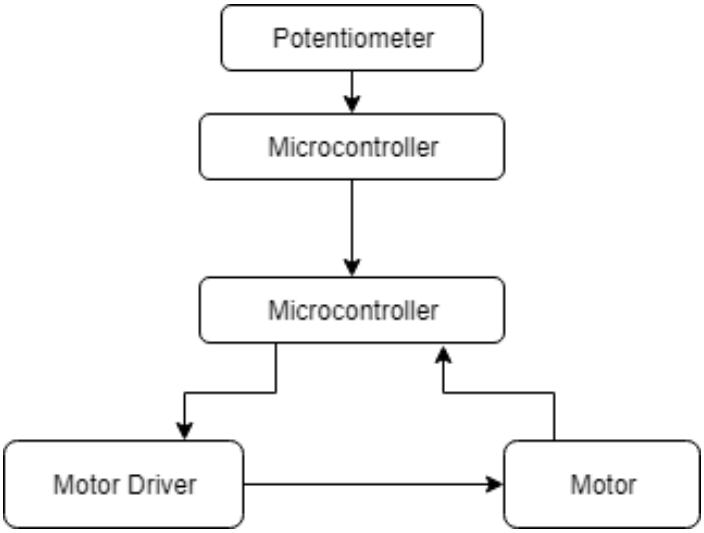

Figure 1: Top level block diagram of the system

1.6 Board Image

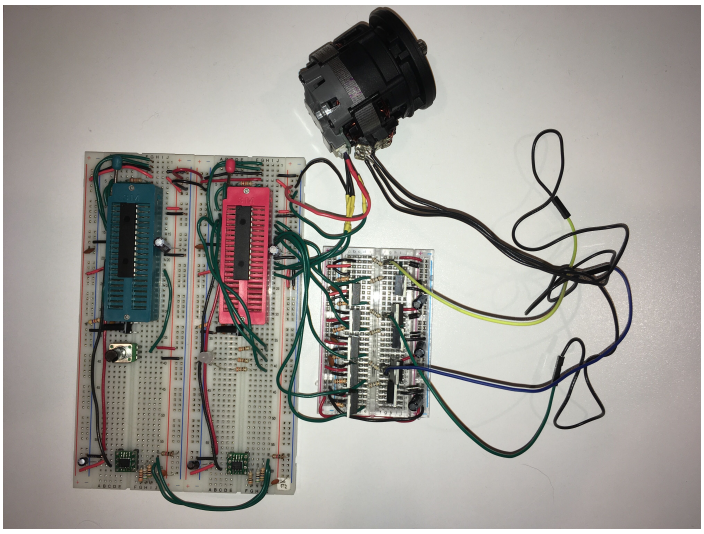

Figure 2: The entire closed loop system

2 Key System Specifications

Table 1: dsPIC33EV256GM102 Specifications

Symbol	Characteristic	Min.	Typ.	Max.	Units
V_{DD}	Voltage Range	4.5	-	5.5	V
I_{DD}	Operating Current	-	28.3	31	mA
I_{Idle}	Idle Current	-	7.25	8.6	mA
I_{IL}	Input Low Voltage	V_{SS}	-	0.2 V_{DD}	V
I_{IH}	Input High Voltage	0.75 V_{DD}	-	5.5	V

Table 2: TC4422A Specifications

Symbol	Characteristic	Min.	Typ.	Max.	Units
V_{DD}	Voltage Range	4.5	-	18	V
V_{IL}	Input Low Voltage	-	1.3	0.8	V
V_{IH}	Input High Voltage	2.4	1.8	-	V
V_{OL}	Output Low Voltage	-	-	0.025	V
V_{OH}	Output High Voltage	V_{DD} - 0.025	-	-	V
I_{DC}	Continuous Output Current	2	-	-	A
I_{PK}	Peak Output Current	-	10	-	A

Table 3: MCP2561 Specifications

Symbol	Characteristic	Min.	Typ.	Max.	Units
V_{DD}	Voltage Range	4.5	-	5.5	V
I_{DDR}	Recessive Supply Current	-	5	10	mA
I_{DDD}	Dominant Supply Current	-	45	70	mA
I_{DDS}	Standby Current	-	5	15	uA
V_{IL}	TXD Input Low Voltage	-0.3	-	0.3 V_{DD}	V
V_{IH}	TXD Input High Voltage	0.7 V_{DD}	-	V_{DD} + 0.3	V
V_{OL}	RXD Output Low Voltage	-	-	0.4	V
V_{OH}	RXD Output High Voltage	V_{DD} - 0.4	-	-	V

3 System Description

The system that was created is a motor controller that communicates over CAN. There are two dsPIC33EV256GM102 microcontrollers set up to communicate over CAN. One reads, analyzes, and transmits data from a potentiometer. The other reads the data from the first microcontroller as well as data from the hall effect sensor in the motor to control the speed of the motor. The second PIC will then determine whether the motor needs to speed up or slow down using an algorithm and transmit data in correspondence to the data received and send it to the gate drivers and motor drivers. These drivers will then spin the motor appropriately.

3.1 Detailed Block Diagram

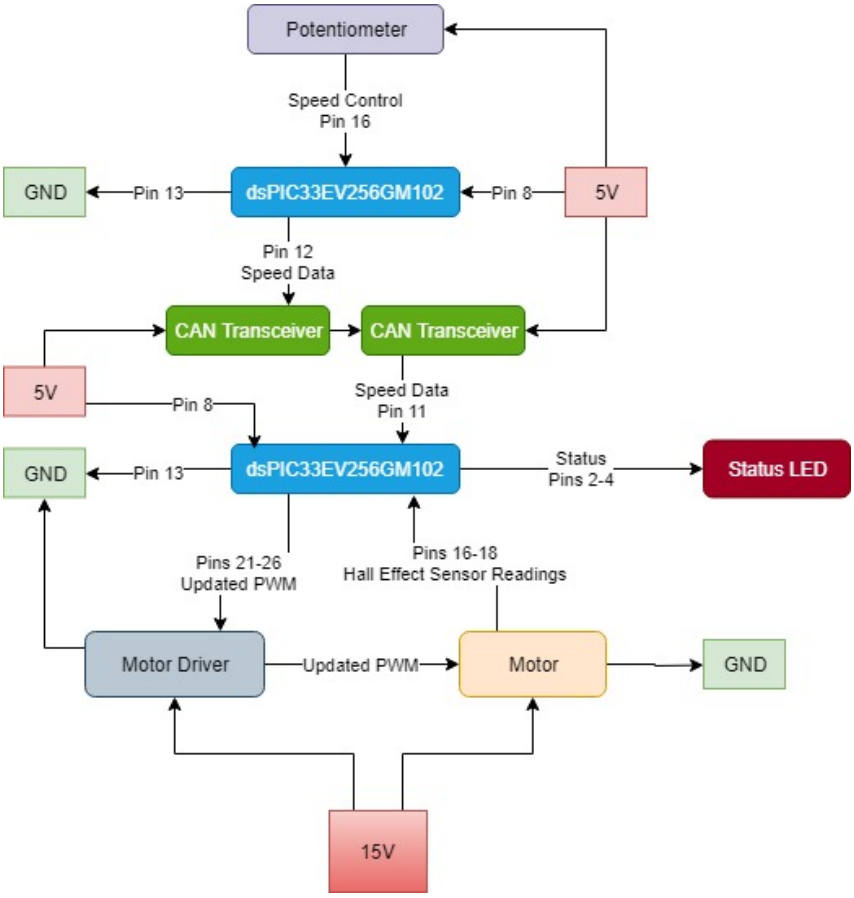

Figure 3: Detailed block diagram of the system

3.2 Highlighted Devices

- dsPIC33EV256GM102 Microcontroller
- Brushless DC Motor
- TC4422A MOSFET Gate Driver
- FQP27P06 PMOS
- FQP30N06L NMOS
- MCP2561 CAN Transceiver

3.3 dsPIC33EV256GM102

A 16-bit microcontroller that can operate at up to 70 MIPS. As this project is in conjunction with the Rowan University Formula Electric Clinic, a versatile microcontroller was chosen that has many more features than were needed in this motor controller. Notable features are:

- Internal 7.37 MHz clock accurate to within 1%.
- Built-in clock switching with PLL
- Low-power modes, operating down to 50 μA
- 6 PWM outputs with 7.14 ns precision
- 12-bit ADC up to 500 ksps
- 5 16-bit timers, which can be paired to create 2 32-bit timers
- 2 UART up to 6.25 Mbps
- 2 SPI up to 15 MHz
- 1 IIC up to 1 Mbaud
- 2 SENT modules
- 1 CAN module with 32 buffers, 16 filters, and 3 masks

The three dual-output PWM modules are designed to be used to control complex motor systems like BLDC motors. This PWM module has a resolution of 7.14 ns, the chosen operation mode was a 10-bit output at 133 kHz. If a higher frequency was desired, then there would be a loss of precision.

The Enhanced CAN (ECAN) peripheral provides 32 receive buffers, of which 8 can be used for transmission. The different filters and masks can be used to sort the incoming messages into different buffers for easy processing.

3.4 Brushless DC Motor

Taken from a DeWalt electric drill, this motor is powered by 3 wires. Internally these wires are connected in a Y configuration, as shown in figure 4. These lines can be driven in a certain sequence that produces a rotation in the desired direction. Figure 5 shows an example sequence of powering the coils. The more current flowing through the coils produces a higher torque, while a higher voltage is needed for a higher speed.

Over time, the amount of current flowing through an inductor increases while voltage is applied. By controlling the PWM at 133 kHz, the current fluctuates less than if the current was slowly toggled. A higher frequency could be used at the loss of resolution.

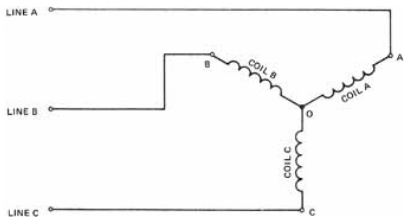

Figure 4: BLDC Motor Y configuration

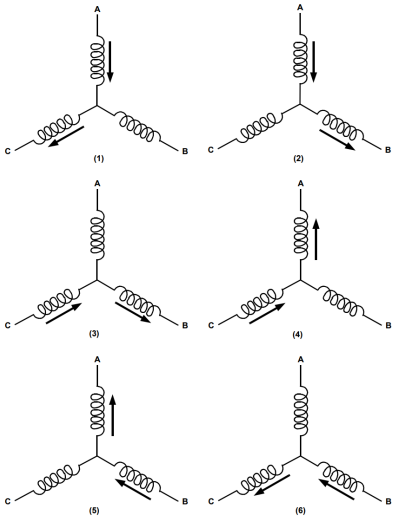

Figure 5: BLDC Motor Winding Energization

Built onto the back (opposite from the drive shaft output) is the hall effect sensor array. These allow a microcontroller to read the rotational position of the output shaft so the correct signals can be driven to the 3 input phases.

3.5 TC4422A MOSFET Gate Driver

To achieve the full amperage of the MOSFETs for driving the motor, a higher voltage than the microcontroller can output must be used. The gate driver performs this action by stepping up the voltage from the microcontroller to the 15 V supply for the motors. It is also able to source up to 10 A of current, which provides fast (¡20 ns) switch times for MOSFETs, as typically the larger the MOSFET, the higher the gate capacitance. It is not good to leave a MOSFET partially on, as that increases its internal resistance while it is still conducting, which generates excess heat.

There is a limitation of max 18 V supply to the gate drivers. Should higher motor voltages be needed in the future (which require higher gate voltages to be applied to the MOSFETs) an alternate circuit can be created. This alternate circuit uses inductors to control the voltages at the MOSFET gates. Figure 6 shows an example of what this circuit could look like.

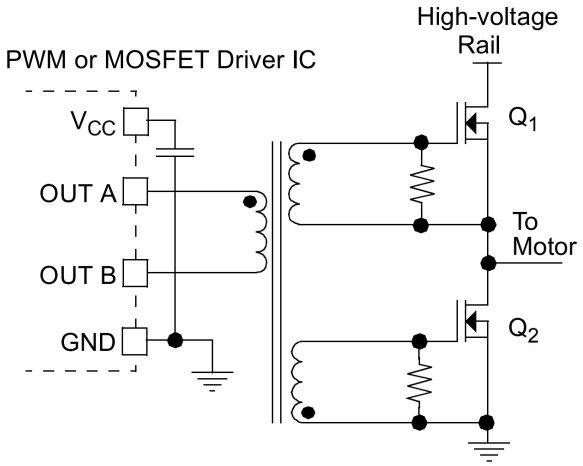

Figure 6: Double-Ended Gate Drive Transformer

3.6 FQP27P06 PMOS

Allows a motor phase to get driven to 15 V. The source is connected to 15 V, the gate is connected to the output of a gate driver, and the drain is connected to a single wire of the motor. When not activated, the drain floats. To activate the MOSFET, the V_{GS} must be less than -4 V, but for the full amperage rating of the MOSFET, -7 V is recommended.

The drain is also connected to the drain of the NMOS. The idea is that either the PMOS or the NMOS is activated at once.

3.7 FQP30N06L NMOS

Allows a motor phase to get driven to ground. The source is connected to ground, the gate is connected to the output of a gate driver, and the drain is connected to a single wire of the motor. When not activated, the drain floats. To activate the MOSFET, the V_{GS} must be greater than 2.5 V, but for the full amperage rating of the MOSFET, -7 V is recommended.

3.8 MCP2561 CAN Transceiver

While the microcontroller contains a CAN peripheral, it does not have the ability to drive the differential voltage. This chip is designed to take the input TTL signal from the microcontroller and convert it to CAN. It also performs the reverse action by sending the current value from the CAN bus to the microcontroller. This separates the transmit/receive buffers, input filters, and input masks in the microcontroller from the actual noisy signals of the CAN bus.

Features like not driving the CAN lines low when not powered and a split output that allows the bus to still operate correctly even if one of the lines is shorted to high or low make this chip a valuable asset to the system. In addition, it is designed to buffer the sensitive microcontroller from the noisy CAN bus that could provide voltages outside of the microcontroller's operating range. To achieve low power usage, an input pin controls whether the CAN transceiver is disabled. Should the microcontroller error and drive a line to an active 0 output for too long, the CAN transceiver times this out and resets its output to a recessive state. This stops a microcontroller from locking the line, preventing any other data from being sent.

4 Circuit Setup

A 15 Volt power supply is used to power both the motor and the motor driver. A 5 Volt power supply is used to power the Microcontrollers, transceivers, and the potentiometer. These are all hooked up to a common ground. The potentiometer is hooked up to the first PIC. The first potentiometer is then connected to a CAN transceiver which is connected to another transceiver that transmits to the second PIC. This PIC has several purposes. It receives data from the first PIC as well as the motor, and it also transmits data to the motor driver. Going down the line, the second pic is connected to the motor driver using 6 pins for 3 phase PWM. Each phase has a high and a low, thus the need for 6 wires. The motor driver consists of MOSFETS and MOSFET gate drivers. These components transmit a 3 phase PWM signal to the motor. The motor takes in this data and spins at a certain speed. The hall effect sensor tracks every time the motor completes a half a revolution. This data is reported back into the second PIC where it can be compared to the desired speed and the mismatch can be corrected if there is one. The second PIC also transmits to an LED that is used to determine the status of the entire system.

5 Test Setup

To confirm the accuracy of the system, the circuit setup was followed. The range of speeds was swept through and the hall effect sensor was monitored to make sure the desired speed was being recorded. A load was put on the motor as well to make sure that the speed of the motor was adjusted to make up for the load.

5.1 Test Data

Figure 7 shows a measurement of a BLDC motor's phase voltage with respect to ground. The motor was set to run at 83.3 rps, or 5000 rpm.

It should be noted that due to a broken MOSFET gate driver, only 2.5 of the 3 phases were working, so 1 out of every 6 different states was exclusively driven to ground. This negatively affected the performance of the motor, but it was able to maintain running due to the momentum of the output shaft. Once the replacement part came in, the full 3 phases worked correctly.

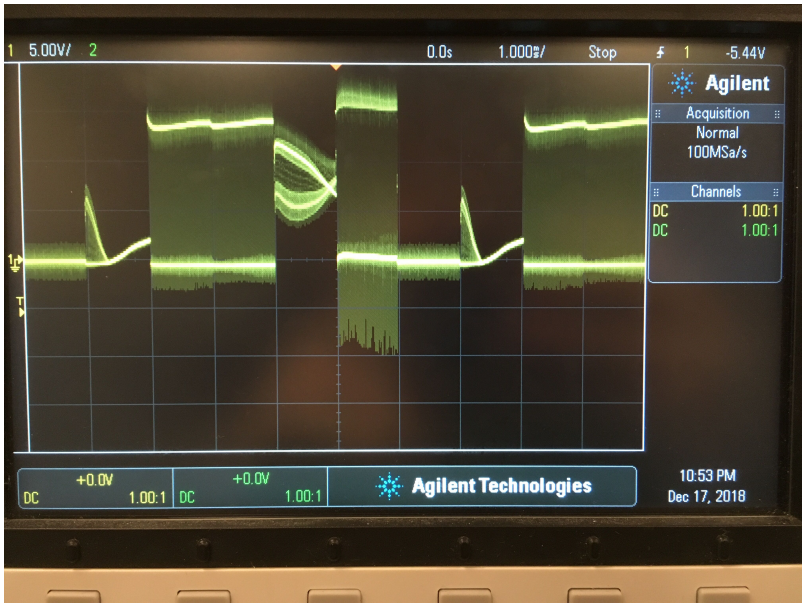

Figure 7: Single phase voltage with respect to ground

Figure 8 is a zoomed-in view of the phases. This shows the PWM cycling on and off to limit the current drawn by the motor and the applied voltage to the motor.

These figures demonstrate the correct operation of the BLDC motor. Motor speed was maintained even though various loads were placed on the output shaft. The slowest speed recorded was 8 rps, and the fastest speed was 150 rps.

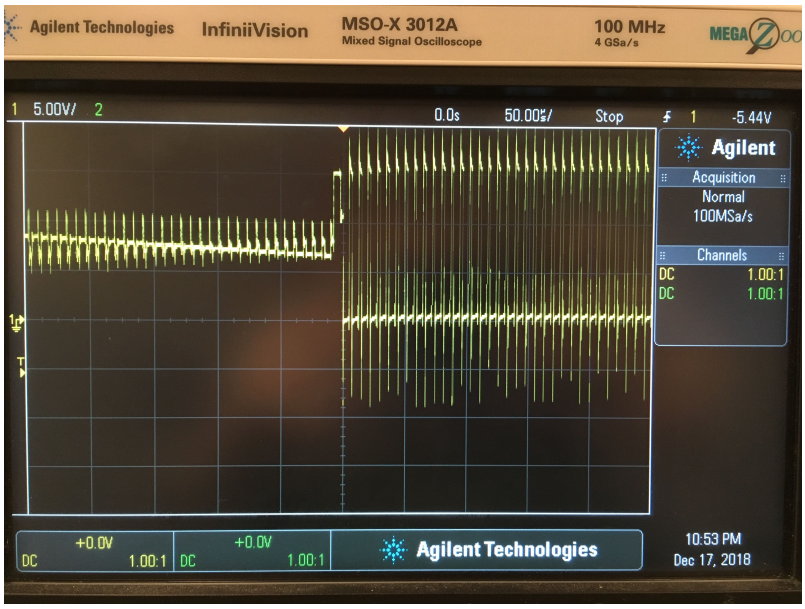

Figure 8: PWM of a single phase with respect to ground

6 Conclusion and Future Work

A BLDC motor driver was successfully prototyped. Control through a wide range of speeds was demonstrated. This project focused on having the motor maintain a certain speed by increasing or decreasing the power applied to the motor as the load changed. Use of a CAN bus was also demonstrated, both receiving and sending messages over the bus.

Future work will include increasing the maximum power safely delivered by the motor driver. This is the main effort as the Formula Electric Clinic needs a high-powered driver to control the motors. Once more power is able to be delivered to the DeWalt motor, steps will be taken to optimize the signals sent to the motor. Rather than sending three square waves to the motor, a continuously variable duty cycle will be used to produce a 3-phase AC signal. Advancing or delaying the signals may also produce better performance. There are two potential ways to tune the controller, either for maximum power or maximum efficiency. It is not expected that these two goals will yield the same tune, but it is an ideal possibility.

To handle the high-powered motor that the Formula Electric Clinic is using, the circuit will need a small redesign to both isolate the high and low voltage electrical systems and also step up the MOSFET gate signals to the 400 V system that is planned. This alternate circuit would use inductors to control the gates of the MOSFETs, with a resistor to discharge the gates when not activated.

7 Design

7.1 Schematics

7.1.1 3x Motor Driver

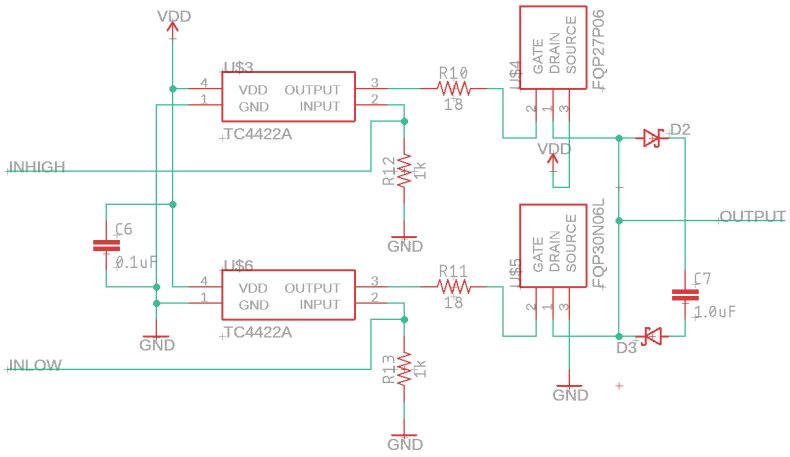

Figure 9: Motor Driver Schematic

7.1.2 2x Microcontroller

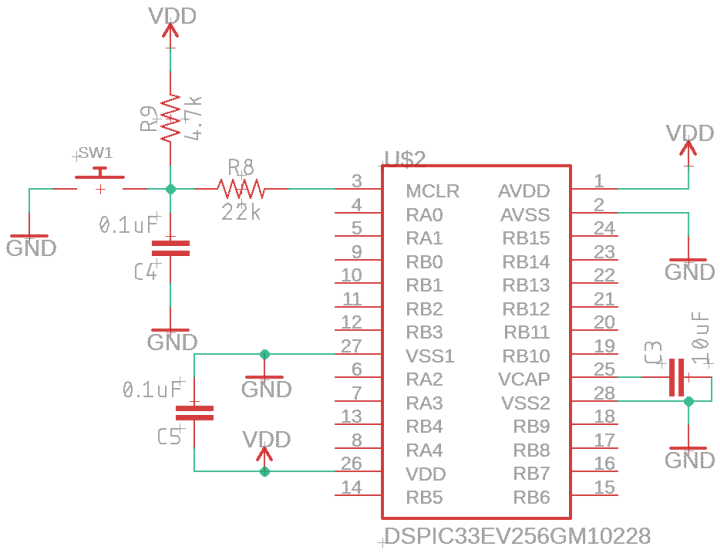

Figure 10: Microcontroller schematic

7.1.3 Motor Controller

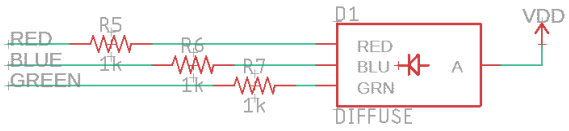

Figure 11: Motor controller schematic

7.1.4 Speed Input

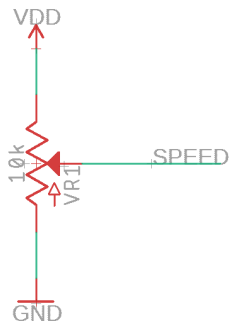

Figure 12: Speed Input Schematic

7.1.5 2x CAN Transceiver

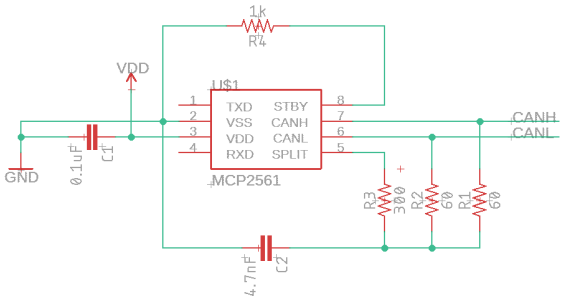

Figure 13: CAN Schematic

7.2 Bill of Materials

The parts list is broken down into multiple lists to categorize the parts used for easy grouping. These lists go with the corresponding schematics.

7.2.1 Motor Driver

- 6x TC4422A MOSFET Gate Driver
- 3x FQP27P06 PMOS

- 3x FQP30N06L NMOS
- 6x 1N5817 Schottky Diode
- 6x 18 Ω Resistor
- 6x 10 kΩ Resistor
- 3x 1 uF Capacitor
- 3x 0.1 uF Capacitor

7.2.2 2x Microcontroller

- 1x dsPIC33EV256GM102 Microcontroller
- 1x 22 kΩ Resistor
- 1x 4.7 kΩ Resistor
- 1x Pushbutton
- 1x 10 uF Capacitor
- 2x 0.1 uF Capacitor

7.2.3 Motor Controller

- 1x 3-Phase Brushless DC Motor with Hall Effect Sensors
- 1x RGB LED
- 3x 330 Ω Resistor

7.2.4 Speed Input

- 1x 10 kΩ Linear Potentiometer

7.2.5 2x CAN Transceiver

- 1x MCP2561 CAN Transceiver
- 2x 60 Ω Resistor
- 1x 300 Ω Resistor
- 1x 1 kΩ Resistor
- 1x 0.1 uF Capacitor
- 1x 4.7 nF Capacitor

7.3 Code Appendix

Please see the full code listing on the GitHub page here. The code is also shown below. It should be noted that two code projects were created, BLDC Input and BLDC Controller. This was written using MPLAB X IDE v5.10 with help from the MPLAB Code Configurator (MCC). Assuming the main.c files are in a folder, the remaining files should be placed in a subfolder titled mcc_generated_files.

7.3.1 BLDC Input main.c File

```
/**
  Generated main.c file from MPLAB Code Configurator

  @Company
    Microchip Technology Inc.

  @File Name
    main.c

  @Summary
    This is the generated main.c using PIC24 / dsPIC33 / PIC32MM MCUs.

  @Description
    This source file provides main entry point for system intialization and
    application code development.
    Generation Information :
        Product Revision  :  PIC24 / dsPIC33 / PIC32MM MCUs - 1.75.1
        Device            :  dsPIC33EV256GM102
    The generated drivers are tested against the following:
        Compiler          :  XC16 v1.35
        MPLAB             :  MPLAB X v5.05
*/

/*
    (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this
    software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
    EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY
    IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
    PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS,
    COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
    INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
    BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS
    IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF
    ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

```

```
41      MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF
        THESE
42      TERMS.
43  */
44
45  /**
46    Section: Included Files
47  */
48  #include "mcc_generated_files/system.h"
49  #include "mcc_generated_files/clock.h"
50  #include "mcc_generated_files/adc1.h"
51  #include "mcc_generated_files/ecan1.h"
52  #define FCY _XTAL_FREQ/2
53  #include <libpic30.h>
54
55  /*
56                          Main application
57   */
58  int main(void)
59  {
60      // initialize the device
61      SYSTEM_Initialize();
62
63      // Create CAN message
64      uCAN1_MSG msg;
65      msg.frame.id = 8;
66      msg.frame.idType = CAN1_FRAME_STD;
67      msg.frame.msgtype = CAN1_MSG_DATA;
68      msg.frame.dlc = 1;
69
70      // Enable transmitting
71      ECAN1_TransmitEnable();
72
73      // Start sampling the ADC
74      ADC1_ChannelSelectSet(ADC1_SPEED);
75      ADC1_SamplingStart();
76
77      while (1)
78      {
79          __delay_ms(10);
80
81          // Get a new speed value
82          uint16_t newSpeed = ADC1_Channel0ConversionResultGet();
83          uint8_t desiredRPS = newSpeed >> 4;
84          ADC1_SamplingStart();
85
86          // Send the new speed value
87          msg.frame.data0 = desiredRPS;
88          ECAN1_transmit(ECAN1_PRIORITY_HIGH, &msg);
89      }
90      return 1;
91  }
92  /**
93   End of File
94  */
```

7.3.2 BLDC Controller main.c File

```
/**
  Generated main.c file from MPLAB Code Configurator

  @Company
    Microchip Technology Inc.

  @File Name
    main.c

  @Summary
    This is the generated main.c using PIC24 / dsPIC33 / PIC32MM MCUs.

  @Description
    This source file provides main entry point for system intialization and
    application code development.
    Generation Information :
        Product Revision  :  PIC24 / dsPIC33 / PIC32MM MCUs - 1.75.1
        Device            :  dsPIC33EV256GM102
    The generated drivers are tested against the following:
        Compiler          :  XC16 v1.35
        MPLAB             :  MPLAB X v5.05
*/

/*
    (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this
    software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
    EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY
    IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
    PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS,
    COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
    INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
    BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS
    IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF
    ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF
    THESE
    TERMS.
*/

/**
  Section: Included Files
*/
#include "mcc_generated_files/system.h"
#include "mcc_generated_files/pwm.h"
```

```
#include "mcc_generated_files/pin_manager.h"
#include "mcc_generated_files/tmr2.h"
#include "mcc_generated_files/adc1.h"
#include "mcc_generated_files/ecan1.h"

/*
                         Main application
 */
int main(void)
{
    CNPDBbits.CNPDB4 = 1;
    // initialize the device
    SYSTEM_Initialize();

    // Set the status LEDs
    Stat1_SetLow();
    Stat2_SetHigh();
    Stat3_SetHigh();

    // Start the PWM
    MDC = 0x2FF;
    PWM_FaultInterruptStatusClear(PWM_GENERATOR_1);
    PWM_FaultInterruptStatusClear(PWM_GENERATOR_2);
    PWM_FaultInterruptStatusClear(PWM_GENERATOR_3);

    // Track the rotation of the motor
    int graycode = 0;
    uint16_t h1, h2, h3;

    // Track the RPS of the motor
    bool first = false;
    double timerPeriod = 1000; // ns
    double desiredRPS = 100;

    // Initialize CAN
    uCAN1_MSG msg;
    ECAN1_ReceiveEnable();

    while (1)
    {
        // Read any available CAN message
        bool received = ECAN1_receive(&msg);
        if (received) {
            desiredRPS = msg.frame.data0;
        }

        // Read the hall effect sensor
        h1 = Hall1_GetValue();
        h2 = Hall2_GetValue();
        h3 = Hall3_GetValue();
        graycode = (h3 << 2) | (h2 << 1) | h1;

        // Enable the PWM lines based on the position of the motor
        switch (graycode) {
            case 1:
                // Recalculate PWM duty cycle to run at a desired speed
                if (first) {
```

```
107                     first = false;
108
109                     // Set the speed based on the time taken to loop once
110                     double period = TMR2_SoftwareCounterGet();
111                     TMR2_SoftwareCounterClear();
112                     // Twice as fast due to two cycles of the hall effect
    sensor per revolution
113                     double calculatedRPS = 5000.0 / period;
114                     if (calculatedRPS < 1) {
115                         calculatedRPS = desiredRPS;
116                     }
117                     double differencePercent = ((desiredRPS * 1.0) /
    calculatedRPS) - 1.0;
118                     uint32_t newValue = MDC + ((MDC * 0.01) *
    differencePercent);
119                     if (newValue > 0x37F) {
120                         newValue = 0x2FF;
121                     } else if (newValue < 0x0FF) {
122                         newValue = 0x0FF;
123                     }
124                     MDC = newValue;
125                 }
126                 Stat2_SetLow();
127                 Stat3_SetHigh();
128                 PWM_OverrideHighEnable(PWM_GENERATOR_1); // Yellow float
129                 PWM_OverrideLowEnable(PWM_GENERATOR_1);
130                 PWM_OverrideHighEnable(PWM_GENERATOR_2); // Green low
131                 PWM_OverrideLowDisable(PWM_GENERATOR_2);
132                 PWM_OverrideHighDisable(PWM_GENERATOR_3); // Blue high
133                 PWM_OverrideLowEnable(PWM_GENERATOR_3);
134                 break;
135             case 5:
136                 Stat2_SetHigh();
137                 Stat3_SetHigh();
138                 PWM_OverrideHighEnable(PWM_GENERATOR_1); // Yellow low
139                 PWM_OverrideLowDisable(PWM_GENERATOR_1);
140                 PWM_OverrideHighEnable(PWM_GENERATOR_2); // Green float
141                 PWM_OverrideLowEnable(PWM_GENERATOR_2);
142                 PWM_OverrideHighDisable(PWM_GENERATOR_3); // Blue high
143                 PWM_OverrideLowEnable(PWM_GENERATOR_3);
144                 break;
145             case 4:
146                 Stat2_SetHigh();
147                 Stat3_SetHigh();
148                 PWM_OverrideHighEnable(PWM_GENERATOR_1); // Yellow low
149                 PWM_OverrideLowDisable(PWM_GENERATOR_1);
150                 PWM_OverrideHighDisable(PWM_GENERATOR_2); // Green high
151                 PWM_OverrideLowEnable(PWM_GENERATOR_2);
152                 PWM_OverrideHighEnable(PWM_GENERATOR_3); // Blue float
153                 PWM_OverrideLowEnable(PWM_GENERATOR_3);
154                 break;
155             case 6:
156                 first = true;
157                 ADC1_SamplingStop();
158                 Stat2_SetHigh();
159                 Stat3_SetHigh();
160                 PWM_OverrideHighEnable(PWM_GENERATOR_1); // Yellow float
```

```
161                     PWM_OverrideLowEnable(PWM_GENERATOR_1);
162                     PWM_OverrideHighDisable(PWM_GENERATOR_2); // Green high
163                     PWM_OverrideLowEnable(PWM_GENERATOR_2);
164                     PWM_OverrideHighEnable(PWM_GENERATOR_3); // Blue low
165                     PWM_OverrideLowDisable(PWM_GENERATOR_3);
166                     break;
167                 case 2:
168                     Stat2_SetHigh();
169                     Stat3_SetHigh();
170                     PWM_OverrideHighDisable(PWM_GENERATOR_1); // Yellow high
171                     PWM_OverrideLowEnable(PWM_GENERATOR_1);
172                     PWM_OverrideHighEnable(PWM_GENERATOR_2); // Green float
173                     PWM_OverrideLowEnable(PWM_GENERATOR_2);
174                     PWM_OverrideHighEnable(PWM_GENERATOR_3); // Blue low
175                     PWM_OverrideLowDisable(PWM_GENERATOR_3);
176                     break;
177                 case 3:
178                     Stat2_SetHigh();
179                     Stat3_SetHigh();
180                     PWM_OverrideHighDisable(PWM_GENERATOR_1); // Yellow high
181                     PWM_OverrideLowEnable(PWM_GENERATOR_1);
182                     PWM_OverrideHighEnable(PWM_GENERATOR_2); // Green low
183                     PWM_OverrideLowDisable(PWM_GENERATOR_2);
184                     PWM_OverrideHighEnable(PWM_GENERATOR_3); // Blue float
185                     PWM_OverrideLowEnable(PWM_GENERATOR_3);
186                     break;
187                 default:
188                     first = false;
189                     MDC = 0x2FF;
190                     Stat2_SetHigh();
191                     Stat3_SetLow();
192                     PWM_OverrideHighEnable(PWM_GENERATOR_1); // All float
193                     PWM_OverrideLowEnable(PWM_GENERATOR_1);
194                     PWM_OverrideHighEnable(PWM_GENERATOR_2);
195                     PWM_OverrideLowEnable(PWM_GENERATOR_2);
196                     PWM_OverrideHighEnable(PWM_GENERATOR_3);
197                     PWM_OverrideLowEnable(PWM_GENERATOR_3);
198             }
199         }
200         return 1;
201 }
202 /**
203  End of File
204 */
```

7.3.3 BLDC Input MCC Generated Files

adc1.h file

```
/**
  ADC1 Generated Driver API Header File

  @Company
    Microchip Technology Inc.

  @File Name
    adc1.h

  @Summary
    This is the generated header file for the ADC1 driver using PIC24 /
    dsPIC33 / PIC32MM MCUs

  @Description
    This header file provides APIs for driver for ADC1.
    Generation Information :
        Product Revision  :  PIC24 / dsPIC33 / PIC32MM MCUs - pic24-dspic-
    pic32mm : 1.75.1
        Device            :  dsPIC33EV256GM102
    The generated drivers are tested against the following:
        Compiler          :  XC16 v1.35
        MPLAB             :  MPLAB X v5.05
*/

/*
    (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this
    software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
    EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY
    IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
    PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS,
    COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
    INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
    BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS
    IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF
    ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF
    THESE
    TERMS.
*/

#ifndef _ADC1_H
#define _ADC1_H

```

```
48 /**
49   Section: Included Files
50 */
51
52 #include <stdbool.h>
53 #include <stdint.h>
54 #include <stdlib.h>
55
56 #ifdef __cplusplus  // Provide C++ Compatibility
57
58     extern "C" {
59
60 #endif
61 /**
62   Section: ISR Helper Macros
63 */
64
65
66 /**
67   Section: Data Types
68 */
69
70 /** ADC Channel Definition
71
72  @Summary
73    Defines the channels available for conversion
74
75  @Description
76    This routine defines the channels that are available conversion.
77
78  Remarks:
79    None
80  */
81 typedef enum
82 {
83     ADC1_SPEED =  0x19,
84     ADC1_CHANNEL_INTERNAL_BAND_GAP_REFERENCE =  0x3D,
85     ADC1_CHANNEL_CTMU =  0x3E,
86     ADC1_MAX_CHANNEL_COUNT = 3
87 } ADC1_CHANNEL;
88
89 /** ADC Positive 123 Channels Definition
90
91  @Summary
92    Defines the positive 123 channels available for conversion
93
94  @Description
95    This routine defines the positive 123 channels that are available for the
96    module to use.
97
98  Remarks:
99    None
100  */
101 typedef enum
102 {
103     ADC1_POS_123_CHANNEL_0 = 0x0,
104     ADC1_POS_123_CHANNEL_1 = 0x1,
```

```
105     ADC1_POS_123_CHANNEL_2 = 0x8,
106     ADC1_POS_123_CHANNEL_3 = 0x9,
107     ADC1_POS_123_CHANNEL_4 = 0x10
108 } ADC1_POS_123_CHANNEL;
109
110 /** ADC Negative 123 Channels Definition
111
112  @Summary
113    Defines the negative 123 channels available for conversion
114
115  @Description
116    This routine defines the negative 123 channels that are available for the
117    module to use.
118
119  Remarks:
120    None
121  */
122 typedef enum
123 {
124    ADC1_NEG_123_CHANNEL_0 = 0x0,
125    ADC1_NEG_123_CHANNEL_1 = 0x2,
126    ADC1_NEG_123_CHANNEL_2 = 0x3
127 } ADC1_NEG_123_CHANNEL;
128
129 /** ADC Data Format Type Definition
130
131  @Summary
132    Defines the data format types available
133
134  @Description
135    This routine defines the data format types that are available for the
       module
136    to use.
137
138  Remarks:
139    None
140  */
141 typedef enum
142 {
143     ADC1_FORM_UNSIGNED_INT    = 0, /* Unsigned Integer */
144     ADC1_FORM_SIGNED_INT      = 1, /* Signed Integer */
145     ADC1_FORM_UNSIGNED_FRACT = 2, /* Unsigned Fraction */
146     ADC1_FORM_SIGNED_FRACT    = 3  /* Signed Integer */
147 } ADC1_FORM_TYPE;
148
149 /** ADC Resolution Type Definition
150
151  @Summary
152    Defines the resolution types available
153
154  @Description
155    This routine defines the resolution types that are available for the module
156    to use.
157
158  Remarks:
159    None
160  */
```

```
161 typedef enum
162 {
163     ADC1_RESOLUTION_10_BIT    = 0, /* 10-bit, 4-channel ADC operation */
164     ADC1_RESOLUTION_12_BIT    = 1  /* 12-bit, 1-channel ADC operation */
165 } ADC1_RESOLUTION_TYPE;
166
167 /** ADC Sampling Source Definition
168
169  @Summary
170    Defines the sampling sources available
171
172  @Description
173    This routine defines the sampling sources that are available for the module
174    to use.
175
176  Remarks:
177    None
178  */
179 typedef enum
180 {
181     ADC1_SAMPLING_SOURCE_PWM2  =  0x1,
182     ADC1_SAMPLING_SOURCE_PWM  =  0x3,
183     ADC1_SAMPLING_SOURCE_MANUAL  =  0x0,
184     ADC1_SAMPLING_SOURCE_AUTO  =  0x7,
185     ADC1_SAMPLING_SOURCE_PWM3  =  0x2,
186     ADC1_SAMPLING_SOURCE_TMR5  =  0x4,
187     ADC1_SAMPLING_SOURCE_TMR3  =  0x2,
188     ADC1_SAMPLING_SOURCE_INT0  =  0x1,
189     ADC1_SAMPLING_SOURCE_CTMU  =  0x6,
190     ADC1_SAMPLING_SOURCE_PWM1  =  0x0,
191 } ADC1_SAMPLING_SOURCE;
192
193 /** ADC Conversion Channel Type Definition
194
195  @Summary
196    Defines the conversion channel types available
197
198  @Description
199    This routine defines the conversion channels types that are available for
       the
200    module to use.
201
202  Remarks:
203    None
204  */
205 typedef enum
206 {
207     ADC1_CONVERSION_CHANNELS_CH0     = 0, /* Converts only CH0 */
208     ADC1_CONVERSION_CHANNELS_CH01    = 1, /* Converts CH0 and CH1 */
209     ADC1_CONVERSION_CHANNELS_CH0123 = 2  /* Converts CH0, CH1, CH2 and CH3 */
210 } ADC1_CONVERSION_CHANNELS_TYPE;
211
212 /**
213   Section: Interface Routines
214 */
215
216
```

```
/**
  @Summary
    This function initializes ADC instance : 1

  @Description
    This routine initializes the ADC driver instance for : 1
    index, making it ready for clients to open and use it. It also initializes
     any
    internal data structures.
    This routine must be called before any other ADC routine is called.

  @Preconditions
    None.

  @Param
    None.

  @Returns
    None.

  @Comment


  @Example
    <code>
        int conversion;
        ADC1_Initialize();
        ADC1_ChannelSelect(AN1_Channel);
        ADC1_SamplingStart();
        //Provide Delay
        for(int i=0;i <1000;i++)
        {
        }
        ADC1_SamplingStop();
        while(!ADC1_IsConversionComplete())
        {
            ADC1_Tasks();
        }
        conversion = ADC1_ConversionResultGet();
    </code>

*/

void ADC1_Initialize (void);

/**
  @Summary
    Clears interrupt flag

  @Description
    This routine is used to clear the interrupt flag manually.

  @Preconditions
    None.

  @Param
    None.
```

```
273
274   @Returns
275     None.
276
277   @Example
278     Refer to ADC1_Initialize() for an example
279
280 */
281
282 inline static void ADC1_InterruptFlagClear(void)
283 {
284    IFS0bits.AD1IF = 0;
285 }
286 /**
287   @Summary
288     Enables interrupts.
289
290   @Description
291     This routine is used to enable the ADC1 interrupt manually.
292
293   @Preconditions
294     None.
295
296   @Param
297     None.
298
299   @Returns
300     None.
301
302   @Example
303     Refer to ADC1_Initialize() for an example
304
305 */
306 inline static void ADC1_InterruptEnable(void)
307 {
308    IEC0bits.AD1IE = 1;
309 }
310 /**
311   @Summary
312     Disables interrupts
313
314   @Description
315     This routine is used to disable the ADC1 interrupt manually.
316
317   @Preconditions
318     None.
319
320   @Param
321     None.
322
323   @Returns
324     None.
325
326   @Example
327     Refer to ADC1_Initialize() for an example
328
329 */
```

```
330
331 inline static void ADC1_InterruptDisable(void)
332 {
333    IEC0bits.AD1IE = 0;
334 }
335 /**
336   @Summary
337     Starts sampling manually.
338
339   @Description
340     This routine is used to start the sampling manually.
341
342   @Preconditions
343     ADC1_Initialize() function should have been called
344     before calling this function.
345
346   @Param
347     None.
348
349   @Returns
350     None.
351
352   @Example
353     <code>
354         int conversion;
355         ADC1__Initialize();
356         ADC1_ChannelSelect(AN1_Channel);
357         ADC1_SamplingStart();
358         //Provide Delay
359         for(int i=0;i <1000;i++)
360         {
361         }
362         ADC1_SamplingStop();
363         while(!ADC1_IsConversionComplete())
364         {
365             ADC1_Tasks();
366         }
367         conversion = ADC1_ConversionResultGet();
368     </code>
369
370 */
371
372 inline static void ADC1_SamplingStart(void)
373 {
374    AD1CON1bits.SAMP = 1;
375 }
376 /**
377   @Summary
378     Stops sampling manually.
379
380   @Description
381     This routine is used to stop the sampling manually before conversion
382     is triggered.
383
384   @Preconditions
385     ADC1_Initialize() function should have been
386     called before calling this function.
```

```
387
388   @Param
389     None.
390
391   @Returns
392     None.
393
394   @Example
395     <code>
396         int conversion;
397         ADC1__Initialize();
398         ADC1_ChannelSelect(AN1_Channel);
399         ADC1_SamplingStart();
400         //Provide Delay
401         for(int i=0;i <1000;i++)
402         {
403         }
404         ADC1_SamplingStop();
405         while(!ADC1_IsConversionComplete())
406         {
407             ADC1_Tasks();
408         }
409         conversion = ADC1_ConversionResultGet();
410     </code>
411 */
412
413 inline static void ADC1_SamplingStop(void)
414 {
415    AD1CON1bits.SAMP = 0;
416 }
417 /**
418   @Summary
419     Gets the buffer loaded with conversion results.
420
421   @Description
422     This routine is used to get the analog to digital converted values in a
423     buffer. This routine gets converted values from multiple channels.
424
425   @Preconditions
426     This routine returns the buffer containing the conversion values only
        after
427     the conversion is complete. Completion status conversion can be checked
        using
428     ADC1_IsConversionComplete() routine.
429
430   @Param
431     None.
432
433   @Returns
434     Returns the count of the buffer containing the conversion values.
435
436   @Example
437     <code>
438         int count;
439         //Initialize for channel scanning
440         ADC1_Initialize();
441         ADC1_SamplingStart();
```

```
442         //Provide Delay
443         for(int i=0;i <1000;i++)
444         {
445         }
446         ADC1_SamplingStop();
447         while(!ADC1_IsConversionComplete())
448         {
449             count = ADC1_ConversionResultBufferGet();
450         }
451     </code>
452 */
453
454 uint16_t ADC1_ConversionResultBufferGet(uint16_t *buffer);
455
456 /**
457   @Summary
458     Returns the ADC1 conversion value for Channel 0.
459
460   @Description
461     This routine is used to get the analog to digital converted value. This
462     routine gets converted values from the channel specified.
463
464   @Preconditions
465     The channel required must be selected before calling this routine using
466     ADC1_ChannelSelect(channel). This routine returns the
467     conversion value only after the conversion is complete. Completion status
468     conversion can be checked using ADC1_IsConversionComplete()
469     routine.
470
471   @Returns
472     Returns the buffer containing the conversion value.
473
474   @Param
475     Buffer address
476
477   @Example
478     Refer to ADC1_Initialize(); for an example
479  */
480
481 inline static uint16_t ADC1_Channel0ConversionResultGet(void)
482 {
483     return ADC1BUF0;
484 }
485 /**
486   @Summary
487     Returns the ADC1 conversion value from Channel 1.
488
489   @Description
490     This routine is used to get the analog to digital converted value. This
491     routine gets converted values from the channel specified.
492
493   @Preconditions
494     The channel required must be selected before calling this routine using
495     ADC1_ChannelSelect(channel). This routine returns the
496     conversion value only after the conversion is complete. Completion status
497     conversion can be checked using ADC1_IsConversionComplete()
498     routine.
```

```
499
500   @Returns
501     Returns the buffer containing the conversion value.
502
503   @Param
504     Buffer address
505
506   @Example
507     Refer to ADC1_Initialize(); for an example
508  */
509
510 inline static uint16_t ADC1_Channel1ConversionResultGet(void)
511 {
512     return ADC1BUF1;
513 }
514 /**
515   @Summary
516     Returns the ADC1 conversion value from Channel 2.
517
518   @Description
519     This routine is used to get the analog to digital converted value. This
520     routine gets converted values from the channel specified.
521
522   @Preconditions
523     The channel required must be selected before calling this routine using
524     ADC1_ChannelSelect(channel). This routine returns the
525     conversion value only after the conversion is complete. Completion status
526     conversion can be checked using ADC1_IsConversionComplete()
527     routine.
528
529   @Returns
530     Returns the buffer containing the conversion value.
531
532   @Param
533     Buffer address
534
535   @Example
536     Refer to ADC1_Initialize(); for an example
537  */
538
539 inline static uint16_t ADC1_Channel2ConversionResultGet(void)
540 {
541     return ADC1BUF2;
542 }
543
544 /**
545   @Summary
546     Returns the ADC1 conversion value from Channel 3.
547
548   @Description
549     This routine is used to get the analog to digital converted value. This
550     routine gets converted values from the channel specified.
551
552   @Preconditions
553     The channel required must be selected before calling this routine using
554     ADC1_ChannelSelect(channel). This routine returns the
555     conversion value only after the conversion is complete. Completion status
```

```
556     conversion can be checked using ADC1_IsConversionComplete()
557     routine.
558
559   @Returns
560     Returns the buffer containing the conversion value.
561
562   @Param
563     Buffer address
564
565   @Example
566     Refer to ADC1_Initialize(); for an example
567  */
568
569 inline static uint16_t ADC1_Channel3ConversionResultGet(void)
570 {
571     return ADC1BUF3;
572 }
573 /**
574   @Summary
575     Returns true when the conversion is completed
576
577   @Description
578     This routine is used to determine if conversion is completed. This routine
579     returns the value of the DONE bit. When conversion is complete the routine
580     returns 1. It returns 0 otherwise.
581
582   @Preconditions
583     ADC1_Initialize() function should have been
584     called before calling this function.
585
586   @Returns
587     Returns true if conversion is completed
588
589   @Param
590     None
591
592   @Example
593     Refer to ADC1_Initialize(); for an example
594  */
595
596 inline static bool ADC1_IsConversionComplete( void )
597 {
598     return AD1CON1bits.DONE; //Wait for conversion to complete
599 }
600
601 /**
602   @Summary
603     Allows selection of a channel for conversion
604
605   @Description
606     This routine is used to select desired channel for conversion.
607
608   @Preconditions
609     ADC1_Initialize() function should have been
610     called before calling this function.
611
612   @Returns
```

```
613     None
614
615   @Param
616     Pass in required channel from the ADC1_CHANNEL list
617
618   @Example
619     Refer to ADC1_Initialize(); for an example
620
621 */
622
623 inline static void ADC1_ChannelSelectSet( ADC1_CHANNEL channel )
624 {
625     AD1CHS0bits.CH0SA = channel;
626 }
627 /**
628   @Summary
629     Returns the channel selected for conversion
630
631   @Description
632     This routine is used to return the channel selected for conversion.
633
634   @Preconditions
635     ADC1_Initialize() function should have been
636     called before calling this function.
637
638   @Returns
639     The value of the Channel Conversion register
640
641   @Param
642     None
643
644   @Example
645     Refer to ADC1_Initialize(); for an example
646
647 */
648
649 inline static uint16_t ADC1_ChannelSelectGet( void )
650 {
651     return AD1CHS0bits.CH0SA ;
652 }
653 /**
654   @Summary
655     Allows selection of a data format type for conversion
656
657   @Description
658     This routine is used to select desired data format for conversion.
659
660   @Preconditions
661     ADC1_Initialize() function should have been
662     called before calling this function.
663
664   @Returns
665     None
666
667   @Param
668     Pass in required data format type from the ADC1_FORM_TYPE list
669
```

```
670   @Example
671     Refer to ADC1_Initialize(); for an example
672 */
673
674 inline static void ADC1_FormatDataSet( ADC1_FORM_TYPE form )
675 {
676     AD1CON1bits.FORM = form;
677 }
678 /**
679   @Summary
680     Allows selection of a resolution mode for conversion
681
682   @Description
683     This routine is used to select desired resolution mode for conversion.
684
685   @Preconditions
686     ADC1_Initialize() function should have been
687     called before calling this function.
688
689   @Returns
690     None
691
692   @Param
693     Pass in required resolution mode from the ADC1_RESOLUTION_TYPE list
694
695   @Example
696     Refer to ADC1_Initialize(); for an example
697 */
698
699 inline static void ADC1_ResolutionModeSet( ADC1_RESOLUTION_TYPE resolution )
700 {
701     AD1CON1bits.AD12B = resolution;
702 }
703 /**
704   @Summary
705     Allows simultaneous sampling to be enabled manually
706
707   @Description
708     This routine is used to enable simultaneous sampling of channels manually
709
710   @Preconditions
711     ADC1_Initialize() function should have been
712     called before calling this function.
713
714   @Returns
715     None
716
717   @Param
718     None.
719
720   @Example
721     Refer to ADC1_Initialize(); for an example
722
723 */
724
725 inline static void ADC1_SimultaneousSamplingEnable(void)
726 {
```

```
727     AD1CON1bits.SIMSAM = 1;
728 }
729 /**
730   @Summary
731     Allows simultaneous sampling to be disabled manually
732 
733   @Description
734     This routine is used to disable simultaneous sampling of channels manually
735 
736   @Preconditions
737     ADC1_Initialize() function should have been
738     called before calling this function.
739 
740   @Returns
741     None
742 
743   @Param
744     None.
745 
746   @Example
747     Refer to ADC1_Initialize(); for an example
748 */
749 
750 inline static void ADC1_SimultaneousSamplingDisble(void)
751 {
752     AD1CON1bits.SIMSAM = 0;
753 }
754 /**
755 @Summary
756     Allows sutomatic sampling to be enabled manually
757 
758   @Description
759     This routine is used to enable automatic sampling of channels manually
760 
761   @Preconditions
762     ADC1_Initialize() function should have been
763     called before calling this function.
764 
765   @Returns
766     None
767 
768   @Param
769     None.
770 
771   @Example
772     Refer to ADC1_Initialize(); for an example
773 */
774 
775 inline static void ADC1_AutomaticSamplingEnable(void)
776 {
777     AD1CON1bits.ASAM = 1;
778 }
779 /**
780   @Summary
781     Allows automatic sampling to be disabled manually
782 
783   @Description
```

```
784     This routine is used to disable automatic sampling of channels manually
785
786   @Preconditions
787     ADC1_Initialize() function should have been
788     called before calling this function.
789
790   @Returns
791     None
792
793   @Param
794     None.
795
796   @Example
797     Refer to ADC1_Initialize(); for an example
798 */
799
800 inline static void ADC1_AutomaticSamplingDisable(void)
801 {
802     AD1CON1bits.ASAM = 0;
803 }
804 /**
805   @Summary
806     Allows conversion clock prescaler value to be set
807
808   @Description
809     This routine is used to allow conversion clock prescaler value to be set
        manually
810
811   @Preconditions
812     ADC1_Initialize() function should have been
813     called before calling this function.
814
815   @Returns
816     None
817
818   @Param
819     Pass in required prescaler integer value
820
821   @Example
822     Refer to ADC1_Initialize(); for an example
823
824 */
825
826 inline static void ADC1_ConversionClockPrescalerSet(uint8_t prescaler)
827 {
828     AD1CON3bits.ADCS = prescaler - 1;
829 }
830 /**
831   @Summary
832     Allows module to be enabled manually
833
834   @Description
835     This routine is used to enable the ADC1 module manually
836
837   @Preconditions
838     ADC1_Initialize() function should have been
839     called before calling this function.
```

```
840
841   @Returns
842     None
843
844   @Param
845     None
846
847   @Example
848 */
849
850 inline static void ADC1_Enable(void)
851 {
852     AD1CON1bits.ADON = 1;
853 }
854 /**
855   @Summary
856     Allows module to be disabled manually
857
858   @Description
859     This routine is used to disable the ADC1 module manually
860
861   @Preconditions
862     ADC1_Initialize() function should have been
863     called before calling this function.
864
865   @Returns
866     None
867
868   @Param
869     None
870
871   @Example
872 */
873
874 inline static void ADC1_Disable(void)
875 {
876     AD1CON1bits.ADON = 0;
877 }
878
879 /**
880   @Summary
881     Allows selection of a positive 123 channel for conversion
882
883   @Description
884     This routine is used to select desired positive 123 channel for conversion
      .
885
886   @Preconditions
887     ADC1_Initialize() function should have been
888     called before calling this function.
889
890   @Returns
891     None
892
893   @Param
894     Pass in required channel from the ADC1_POS_123_CHANNEL list
895
```

```
896   @Example
897     Refer to ADC1_Initialize(); for an example
898
899 */
900
901 inline static void ADC1_Positive123ChannelSelect( ADC1_POS_123_CHANNEL channel
        )
902 {
903     AD1CHS123 = (AD1CHS123 & 0xFF06) | channel;
904 }
905 /**
906   @Summary
907     Allows selection of a negative 123 channel for conversion
908
909   @Description
910     This routine is used to select desired negative 123 channel for conversion
        .
911
912   @Preconditions
913     ADC1_Initialize() function should have been
914     called before calling this function.
915
916   @Returns
917     None
918
919   @Param
920     Pass in required channel from the ADC1_NEG_123_CHANNEL list
921
922   @Example
923     Refer to ADC1_Initialize(); for an example
924
925 */
926
927 inline static void ADC1_Negative123ChannelSelect( ADC1_NEG_123_CHANNEL channel
        )
928 {
929     AD1CHS123bits.CH123NA = channel;
930 }
931 /**
932   @Summary
933     Allows selection of conversion channels
934
935   @Description
936     This routine is used to select conversion channel for conversion.
937
938   @Preconditions
939     ADC1_Initialize() function should have been
940     called before calling this function.
941
942   @Returns
943     None
944
945   @Param
946     Pass in required channel from the ADC1_CONVERSION_CHANNELS_TYPE list
947
948   @Example
949     Refer to ADC1_Initialize(); for an example
```

```
950
951 */
952
953 inline static void ADC1_ConversionChannelsSet( ADC1_CONVERSION_CHANNELS_TYPE
        channel )
954 {
955     AD1CON2bits.CHPS = channel;
956 }
957 /**
958   @Summary
959     Allows selection of a priority for interrupt
960
961   @Description
962     This routine is used to select desired priority for interrupt.
963
964   @Preconditions
965     ADC1_Initialize() function should have been
966     called before calling this function.
967
968   @Returns
969     None
970
971   @Param
972     Pass in required integer priority value
973
974   @Example
975     Refer to ADC1_Initialize(); for an example
976
977 */
978
979 inline static void ADC1_InterruptPrioritySet( uint16_t priorityValue )
980 {
981     _AD1IP = 0x7 & priorityValue;
982 }
983 /**
984   @Summary
985     Polled implementation
986
987   @Description
988     This routine is used to implement the tasks for polled implementations.
989
990   @Preconditions
991     ADC1_Initialize() function should have been
992     called before calling this function.
993
994   @Returns
995     None
996
997   @Param
998     None
999
1000  @Example
1001    Refer to ADC1_Initialize(); for an example
1002
1003 */
1004 void ADC1_Tasks(void);
1005
```

```
1006
1007 #ifdef __cplusplus  // Provide C++ Compatibility
1008
1009     }
1010
1011 #endif
1012
1013 #endif // _ADC1_H
1014
1015 /**
1016  End of File
1017 */
```

adc1.c file

```
/**
  ADC1 Generated Driver File

  @Company
    Microchip Technology Inc.

  @File Name
    adc1.c

  @Summary
    This is the generated header file for the ADC1 driver using PIC24 /
    dsPIC33 / PIC32MM MCUs

  @Description
    This header file provides APIs for driver for ADC1.
    Generation Information :
        Product Revision  :  PIC24 / dsPIC33 / PIC32MM MCUs - pic24-dspic-
    pic32mm : 1.75.1
        Device            :  dsPIC33EV256GM102
    The generated drivers are tested against the following:
        Compiler          :  XC16 v1.35
        MPLAB             :  MPLAB X v5.05
*/

/*
    (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this
    software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
    EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY
    IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
    PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS,
    COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
    INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
    BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS
    IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF
    ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF
    THESE
    TERMS.
*/

/**
  Section: Included Files
*/

#include <xc.h>
```

```
50 #include "adc1.h"
51
52 /**
53   Section: Data Type Definitions
54 */
55
56 /* ADC Driver Hardware Instance Object
57
58   @Summary
59     Defines the object required for the maintenance of the hardware instance.
60
61   @Description
62     This defines the object required for the maintenance of the hardware
63     instance. This object exists once per hardware instance of the peripheral.
64
65  */
66 typedef struct
67 {
68   uint8_t intSample;
69 }
70
71 ADC_OBJECT;
72
73 static ADC_OBJECT adc1_obj;
74
75 /**
76   Section: Driver Interface
77 */
78
79
80 void ADC1_Initialize (void)
81 {
82     // ASAM disabled; ADDMABM disabled; ADSIDL disabled; DONE disabled; SIMSAM
       Sequential; FORM Absolute decimal result, unsigned, right-justified;
      SAMP disabled; SSRC Internal counter ends sampling and starts conversion;
       AD12B 12-bit; ADON enabled; SSRCG disabled;
83
84    AD1CON1 = 0x84E0;
85
86     // CSCNA disabled; VCFG0 AVDD; VCFG1 AVSS; ALTS disabled; BUFM disabled;
      SMPI Generates interrupt after completion of every sample/conversion
      operation; CHPS 1 Channel;
87
88    AD1CON2 = 0x0;
89
90     // SAMC 0; ADRC FOSC/2; ADCS 99;
91
92    AD1CON3 = 0x63;
93
94     // CH0SA OA2/AN0; CH0SB OA2/AN0; CH0NB VREFL; CH0NA VREFL;
95
96    AD1CHS0 = 0x0;
97
98     // CSS26 disabled; CSS25 disabled; CSS24 disabled; CSS27 disabled;
99
100   AD1CSSH = 0x0;
101
```

```
102     // CSS2 disabled; CSS1 disabled; CSS0 disabled; CSS5 disabled; CSS4
        disabled; CSS3 disabled;
103
104    AD1CSSL = 0x0;
105
106     // DMABL Allocates 1 word of buffer to each analog input; ADDMAEN disabled
        ;
107
108    AD1CON4 = 0x0;
109
110     // CH123SA2 disabled; CH123SB2 CH1=OA2/AN0,CH2=AN1,CH3=AN2; CH123NA
        disabled; CH123NB CH1=VREF-,CH2=VREF-,CH3=VREF-;
111
112    AD1CHS123 = 0x0;
113
114
115    adc1_obj.intSample = AD1CON2bits.SMPI;
116
117 }
118
119
120
121 void ADC1_Tasks ( void )
122 {
123     // clear the ADC interrupt flag
124     IFS0bits.AD1IF = false;
125 }
126
127
128
129 /**
130   End of File
131 */
```

clock.h file

```
/**
  @Generated PIC24 / dsPIC33 / PIC32MM MCUs Source File

  @Company:
    Microchip Technology Inc.

  @File Name:
    clock.h

  @Summary:
    This is the clock.h file generated using PIC24 / dsPIC33 / PIC32MM MCUs

  @Description:
    This header file provides implementations for driver APIs for all modules selected in the GUI.
    Generation Information :
        Product Revision  :  PIC24 / dsPIC33 / PIC32MM MCUs - pic24-dspic-pic32mm : 1.75.1
        Device            :  dsPIC33EV256GM102
    The generated drivers are tested against the following:
        Compiler          :  XC16 v1.35
        MPLAB             :  MPLAB X v5.05
*/

/*
    (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this
    software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
    EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
    PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS, COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
    INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
    BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF THESE
    TERMS.
*/

#ifndef CLOCK_H
#define CLOCK_H

#define _XTAL_FREQ  138187500UL
/**
```

```
50  * @Param
51      none
52  * @Returns
53      none
54  * @Description
55      Initializes the oscillator to the default states configured in the
56  *                   MCC GUI
57  * @Example
58      CLOCK_Initialize(void);
59  */
60 void CLOCK_Initialize(void);
61 #endif  /* CLOCK_H */
62 /**
63  End of File
64 */
```

clock.c file

```
/**
  @Generated PIC24 / dsPIC33 / PIC32MM MCUs Source File

  @Company:
    Microchip Technology Inc.

  @File Name:
    clock.c

  @Summary:
    This is the clock.c file generated using PIC24 / dsPIC33 / PIC32MM MCUs

  @Description:
    This header file provides implementations for driver APIs for all modules selected in the GUI.
    Generation Information :
        Product Revision  :  PIC24 / dsPIC33 / PIC32MM MCUs - pic24-dspic-pic32mm : 1.75.1
        Device            :  dsPIC33EV256GM102
    The generated drivers are tested against the following:
        Compiler          :  XC16 v1.35
        MPLAB             :  MPLAB X v5.05
*/

/*
    (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this
    software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
    EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
    PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS, COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
    INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
    BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF THESE
    TERMS.
*/

#include "clock.h"
#include "stdint.h"
#include "xc.h"

void CLOCK_Initialize(void)
```

```
50 {
51     // FRCDIV FRC/1; PLLPRE 2; DOZE 1:8; PLLPOST 1:2; DOZEN disabled; ROI
       disabled;
52     CLKDIV = 0x3000;
53     // TUN Center frequency;
54     OSCTUN = 0x0;
55     // ROON disabled; ROSEL disabled; RODIV Base clock value; ROSSLP disabled;
56     REFOCON = 0x0;
57     // PLLDIV 73;
58     PLLFBD = 0x49;
59     // RND disabled; SATB disabled; SATA disabled; ACCSAT disabled;
60   CORCONbits.RND = 0;
61   CORCONbits.SATB = 0;
62   CORCONbits.SATA = 0;
63   CORCONbits.ACCSAT = 0;
64     // CF no clock failure; NOSC FRCPLL; CLKLOCK unlocked; OSWEN Switch is
       Complete;
65     __builtin_write_OSCCONH((uint8_t) ((0x100 >> _OSCCON_NOSC_POSITION) & 0
       x00FF));
66     __builtin_write_OSCCONL((uint8_t) ((0x100 | _OSCCON_OSWEN_MASK) & 0xFF));
67     // Wait for Clock switch to occur
68     while (OSCCONbits.OSWEN != 0);
69     while (OSCCONbits.LOCK != 1);
70 }
```

dma.h file

```
/*
    ***************************************************************************

  DMA Generated Driver API Header File

  Company:
    Microchip Technology Inc.

  File Name:
    dma.h

  Summary:
    This is the generated header file for the DMA driver using PIC24 / dsPIC33
     / PIC32MM MCUs

  Description:
    This header file provides APIs for driver for DMA.
    Generation Information :
        Product Revision  :  PIC24 / dsPIC33 / PIC32MM MCUs - pic24-dspic-
    pic32mm : 1.75.1
        Device            :  dsPIC33EV256GM102
    The generated drivers are tested against the following:
        Compiler          :  XC16 v1.35
        MPLAB             :  MPLAB X v5.05

    ***************************************************************************
    */

/*
    (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this
    software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
    EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY
    IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
    PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS,
    COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
    INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
    BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS
    IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF
    ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF
    THESE
    TERMS.
*/

#ifndef DMA_H
```

```
46 #define DMA_H
47
48 #include <xc.h>
49 #include <stdbool.h>
50 #include <stdint.h>
51 #include <stdlib.h>
52
53 #ifdef __cplusplus  // Provide C++ Compatibility
54
55     extern "C" {
56
57 #endif
58
59 /**
60   Section: Data Types
61 */
62
63 /** DMA Channel Definition
64
65  @Summary
66    Defines the channels available for DMA
67
68  @Description
69    This routine defines the channels that are available for the module to use.
70
71  Remarks:
72    None
73  */
74 typedef enum
75 {
76     DMA_CHANNEL_0 =  0,
77     DMA_CHANNEL_1 =  1,
78     DMA_CHANNEL_2 =  2,
79     DMA_CHANNEL_3 =  3,
80     DMA_NUMBER_OF_CHANNELS = 4
81 } DMA_CHANNEL;
82 /**
83   Section: Interface Routines
84 */
85
86 /**
87   @Summary
88     This function initializes DMA instance : 1
89
90   @Description
91     This routine initializes the DMA driver instance for : 1
92     index, making it ready for clients to open and use it. It also initializes
       any
93     internal data structures.
94     This routine must be called before any other DMA routine is called.
95
96   @Preconditions
97     None.
98
99   @Param
100    None.
101
```

```
102   @Returns
103     None.
104
105   @Comment
106
107
108   @Example
109     <code>
110         unsigned short int srcArray[100];
111         unsigned short int dstArray[100];
112         int i;
113         int count;
114         for (i=0; i<100; i++)
115         {
116             srcArray[i] = i+1;
117             dstArray[i] = 0;
118         }
119
120         DMA_Initialize();
121         DMA_SoftwareTriggerEnable(CHANNEL1);
122
123         count = DMA_TransferCountGet;
124         while(count > 0)
125         {
126         while(
127         DMA_SoftwareTriggerEnable(CHANNEL1);
128         }
129     </code>
130
131 */
132 void DMA_Initialize(void);
133
134 /**
135   @Summary
136     Clears the interrupt request flag.
137
138   @Description
139     This routine is used to clear the interrupt request flag. This routine
      sets the value of the DMAIF bit to 0.
140
141   @Preconditions
142     DMA_Initializer() function should have been
143     called before calling this function.
144
145   @Returns
146     None
147
148   @Param
149     None
150
151   @Example
152     Refer to DMA_Initializer(); for an example
153  */
154 inline static void DMA_FlagInterruptClear(DMA_CHANNEL  channel)
155 {
156     switch(channel) {
157         case DMA_CHANNEL_0:
```

```
158                 IFS0bits.DMA0IF = 0;
159                 break;
160         case DMA_CHANNEL_1:
161                 IFS0bits.DMA1IF = 0;
162                 break;
163         case DMA_CHANNEL_2:
164                 IFS1bits.DMA2IF = 0;
165                 break;
166         case DMA_CHANNEL_3:
167                 IFS2bits.DMA3IF = 0;
168                 break;
169         default:break;
170
171     }
172 }
173
174 /**
175 @Summary
176     Enables the interrupt for a DMA channel
177
178   @Description
179     This routine is used to enable an interrupt for a DMA channel. This
        routine
180     sets the value of the DMAIE bit to 1.
181
182   @Preconditions
183     DMA_Initializer() function should have been
184     called before calling this function.
185
186   @Returns
187     None
188
189   @Param
190     None
191
192   @Example
193     Refer to DMA_Initializer(); for an example
194  */
195 inline static void DMA_InterruptEnable(DMA_CHANNEL  channel)
196 {
197     switch(channel) {
198         case DMA_CHANNEL_0:
199                 IEC0bits.DMA0IE = 1;
200                 break;
201         case DMA_CHANNEL_1:
202                 IEC0bits.DMA1IE = 1;
203                 break;
204         case DMA_CHANNEL_2:
205                 IEC1bits.DMA2IE = 1;
206                 break;
207         case DMA_CHANNEL_3:
208                 IEC2bits.DMA3IE = 1;
209                 break;
210         default:break;
211
212     }
213 }
```

```
/**
@Summary
 Disables the interrupt for a DMA channel

@Description
 This routine is used to disable an interrupt for a DMA channel. This routine
 sets the value of the DMAIE bit to 0.

@Preconditions
  DMA_Initializer() function should have been
  called before calling this function.

@Returns
 None

@Param
 None

@Example
 Refer to DMA_Initializer(); for an example
*/
inline static void DMA_InterruptDisable(DMA_CHANNEL  channel)
{
    switch(channel) {
        case DMA_CHANNEL_0:
                IEC0bits.DMA0IE = 0;
                break;
        case DMA_CHANNEL_1:
                IEC0bits.DMA1IE = 0;
                break;
        case DMA_CHANNEL_2:
                IEC1bits.DMA2IE = 0;
                break;
        case DMA_CHANNEL_3:
                IEC2bits.DMA3IE = 0;
                break;
        default:break;
    }
}

/**
 @Summary
  Enables the channel in the DMA

 @Description
  This routine is used to enable a channel in the DMA. This routine
  sets the value of the CHEN bit to 1.

 @Preconditions
  DMA_Initializer() function should have been
  called before calling this function.

 @Returns
  None

 @Param
  None
```

```
271
272  @Example
273  Refer to DMA_Initializer(); for an example
274  */
275 inline static void DMA_ChannelEnable(DMA_CHANNEL  channel)
276 {
277     switch(channel) {
278         case DMA_CHANNEL_0:
279                 DMA0CONbits.CHEN = 1;
280                 break;
281         case DMA_CHANNEL_1:
282                 DMA1CONbits.CHEN = 1;
283                 break;
284         case DMA_CHANNEL_2:
285                 DMA2CONbits.CHEN = 1;
286                 break;
287         case DMA_CHANNEL_3:
288                 DMA3CONbits.CHEN = 1;
289                 break;
290         default: break;
291     }
292 }
293
294 /**
295  @Summary
296  Disables the channel in the DMA
297
298  @Description
299   This routine is used to disable a channel in the DMA. This routine
300   sets the value of the CHEN bit to 0.
301
302 @Preconditions
303  DMA_Initializer() function should have been
304  called before calling this function.
305
306 @Returns
307  None
308
309 @Param
310  None
311
312 @Example
313  Refer to DMA_Initializer(); for an example
314 */
315 inline static void DMA_ChannelDisable(DMA_CHANNEL  channel)
316 {
317     switch(channel) {
318         case DMA_CHANNEL_0:
319                 DMA0CONbits.CHEN = 0;
320                 break;
321         case DMA_CHANNEL_1:
322                 DMA1CONbits.CHEN = 0;
323                 break;
324         case DMA_CHANNEL_2:
325                 DMA2CONbits.CHEN = 0;
326                 break;
327         case DMA_CHANNEL_3:
```

```
328                 DMA3CONbits.CHEN = 0;
329                 break;
330         default: break;
331     }
332 }
333 /**
334   @Summary
335     Sets the transfer count of the DMA
336 
337   @Description
338     This routine is used to set the DMA transfer count. This routine sets the
339     value of the DMACNT register.
340 
341   @Preconditions
342     DMA_Initializer() function should have been
343     called before calling this function.
344 
345   @Returns
346     None
347 
348   @Param
349     None
350 
351   @Example
352     Refer to DMA_Initializer(); for an example
353  */
354 inline static void DMA_TransferCountSet(DMA_CHANNEL channel, uint16_t
      transferCount)
355 {
356     switch(channel) {
357         case DMA_CHANNEL_0:
358                 DMA0CNT = transferCount;
359                 break;
360         case DMA_CHANNEL_1:
361                 DMA1CNT = transferCount;
362                 break;
363         case DMA_CHANNEL_2:
364                 DMA2CNT = transferCount;
365                 break;
366         case DMA_CHANNEL_3:
367                 DMA3CNT = transferCount;
368                 break;
369         default: break;
370     }
371 }
372 /**
373  @Summary
374   Returns the transfer count of the DMA
375 
376 @Description
377  This routine is used to determine the DMA transfer count. This routine
378  returns the value of the DMACNT register.
379 
380 @Preconditions
381   DMA_Initializer() function should have been
382   called before calling this function.
383 
```

```
384 @Returns
385  Returns the transfer count of the DMA
386
387 @Param
388  None
389
390 @Example
391  Refer to DMA_Initializer(); for an example
392 */
393 inline static uint16_t DMA_TransferCountGet(DMA_CHANNEL channel)
394     {
395     switch(channel) {
396             case DMA_CHANNEL_0:
397                 return (DMA0CNT);
398             case DMA_CHANNEL_1:
399                 return (DMA1CNT);
400             case DMA_CHANNEL_2:
401                 return (DMA2CNT);
402             case DMA_CHANNEL_3:
403                 return (DMA3CNT);
404         default: return 0;
405     }
406 }
407
408 /**
409  @Summary
410   Initiates a transfer on the requested DMA channel.
411
412  @Description
413   This routine is used to initiate a transfer on the requested DMA channel.
        When a transfer on the requested channel is initiated the routine
414   returns the value of the FORCE bit. It returns 0 otherwise.
415
416  @Preconditions
417    DMA_Initializer() function should have been
418    called before calling this function.
419
420  @Returns
421   Returns true if the transfer on the requested channel is initiated
422
423  @Param
424   None
425
426  @Example
427   Refer to DMA_Initializer(); for an example
428  */
429 inline static void DMA_SoftwareTriggerEnable(DMA_CHANNEL channel )
430 {
431     switch(channel) {
432         case DMA_CHANNEL_0:
433                 DMA0REQbits.FORCE = 1;
434         case DMA_CHANNEL_1:
435                 DMA1REQbits.FORCE = 1;
436         case DMA_CHANNEL_2:
437                 DMA2REQbits.FORCE = 1;
438         case DMA_CHANNEL_3:
439                 DMA3REQbits.FORCE = 1;
```

```
440         default: break;
441     }
442 }
443 
444 /**
445   @Summary
446     Sets the address for register A in the DMA
447 
448   @Description
449     This routine is used to set the address in register A for a DMA channel.
450 
451   @Preconditions
452     DMA_Initializer() function should have been
453     called before calling this function.
454 
455   @Returns
456     None
457 
458   @Param
459     None
460 
461   @Example
462     Refer to DMA_Initializer(); for an example
463  */
464 inline static void DMA_StartAddressASet(DMA_CHANNEL  channel, uint16_t address
        )
465 {
466     switch(channel) {
467         case DMA_CHANNEL_0:
468                 DMA0STAL = address;
469                 DMA0STAH = 0;
470                 break;
471         case DMA_CHANNEL_1:
472                 DMA1STAL = address;
473                 DMA1STAH = 0;
474                 break;
475         case DMA_CHANNEL_2:
476                 DMA2STAL = address;
477                 DMA2STAH = 0;
478                 break;
479         case DMA_CHANNEL_3:
480                 DMA3STAL = address;
481                 DMA3STAH = 0;
482                 break;
483         default: break;
484     }
485     }
486 
487 /**
488   @Summary
489     Sets the address for register B in the DMA
490 
491   @Description
492     This routine is used to set the address in register B for a DMA channel.
493 
494   @Preconditions
495     DMA_Initializer() function should have been
```

```
496     called before calling this function.
497
498   @Returns
499     None
500
501   @Param
502     None
503
504   @Example
505     Refer to DMA_Initializer(); for an example
506  */
507
508 inline static void DMA_StartAddressBSet(DMA_CHANNEL  channel, uint16_t address
        )
509 {
510     switch(channel) {
511         case DMA_CHANNEL_0:
512                 DMA0STBL = address;
513                 DMA0STBH = 0;
514                 break;
515         case DMA_CHANNEL_1:
516                 DMA1STBL = address;
517                 DMA1STBH = 0;
518                 break;
519         case DMA_CHANNEL_2:
520                 DMA2STBL = address;
521                 DMA2STBH = 0;
522                 break;
523         case DMA_CHANNEL_3:
524                 DMA3STBL = address;
525                 DMA3STBH = 0;
526                 break;
527         default: break;
528     }
529 }
530     /**
531   @Summary
532     Gets the address for register A in the DMA
533
534   @Description
535     This routine is used to get the address in register A for a DMA channel.
536
537   @Preconditions
538     DMA_Initializer() function should have been
539     called before calling this function.
540
541   @Returns
542     None
543
544   @Param
545     None
546
547   @Example
548     Refer to DMA_Initializer(); for an example
549  */
550 inline static uint16_t DMA_StartAddressAGet(DMA_CHANNEL  channel, uint16_t
        address)
```

```
551 {
552     switch(channel) {
553         case DMA_CHANNEL_0:
554                 address= DMA0STAL;
555                 break;
556         case DMA_CHANNEL_1:
557                 address= DMA1STAL;
558                 break;
559         case DMA_CHANNEL_2:
560                 address= DMA2STAL;
561                 break;
562         case DMA_CHANNEL_3:
563                 address= DMA3STAL;
564                 break;
565         default:
566                 address = 0;
567                 break;
568     }
569         return address;
570
571 }
572     /**
573   @Summary
574     Sets the address for register B in the DMA
575
576   @Description
577     This routine is used to set the address in register B for a DMA channel.
578
579   @Preconditions
580     DMA_Initializer() function should have been
581     called before calling this function.
582
583   @Returns
584     None
585
586   @Param
587     None
588
589   @Example
590     Refer to DMA_Initializer(); for an example
591  */
592
593 inline static uint16_t DMA_StartAddressBGet(DMA_CHANNEL  channel, uint16_t
      address) {
594
595     switch(channel) {
596         case DMA_CHANNEL_0:
597                 address= DMA0STBL;
598                 break;
599         case DMA_CHANNEL_1:
600                 address= DMA1STBL;
601                 break;
602         case DMA_CHANNEL_2:
603                 address= DMA2STBL;
604                 break;
605         case DMA_CHANNEL_3:
606                 address= DMA3STBL;
```

```
607                     break;
608             default:
609                     address = 0;
610                     break;
611         }
612             return address;
613 }
614         /**
615         @Summary
616         Sets the peripheral address in the DMA
617 
618       @Description
619         This routine is used to set the peripheral address for a DMA channel.
620 
621       @Preconditions
622         DMA_Initializer() function should have been
623         called before calling this function.
624 
625       @Returns
626         None
627 
628       @Param
629         None
630 
631       @Example
632         Refer to DMA_Initializer(); for an example
633     */
634     inline static void DMA_PeripheralAddressSet(DMA_CHANNEL channel, volatile
        unsigned int * address)
635         {
636         switch(channel) {
637             case DMA_CHANNEL_0:
638                     DMA0PAD = (int)address;
639                     break;
640             case DMA_CHANNEL_1:
641                     DMA1PAD = (int)address;
642                     break;
643             case DMA_CHANNEL_2:
644                     DMA2PAD = (int)address;
645                     break;
646             case DMA_CHANNEL_3:
647                     DMA3PAD = (int)address;
648                     break;
649             default: break;
650         }
651 }
652 /**
653       @Summary
654         Returns true when there is a Peripheral Write Collision Event
655 
656       @Description
657         This routine is used to determine if there is a Peripheral Write Collision
            Event. This routine
658         returns the value of the PWCOL bit in DMAPWC register. When there is a
            Peripheral Write Collision Event, the routine
659         returns 1. It returns 0 otherwise.
660 
```

```
661   @Preconditions
662     DMA_Initializer() function should have been
663     called before calling this function.
664
665   @Returns
666     Returns true if there is a Peripheral Write Collision Event
667
668   @Param
669     None
670
671   @Example
672     Refer to DMA_Initializer(); for an example
673  */
674 inline static bool DMA_IsPeripheralWriteCollision(uint16_t dmaChannel )
675 {
676     return DMAPWC & (1 << dmaChannel);
677 }
678
679 /**
680   @Summary
681     Returns true when there is a Request Collision Event
682
683   @Description
684     This routine is used to determine if there is a Request Collision Event.
        This routine
685     returns the value of the RQCOL bit in DMARQC register. When there is a
        Request Collision Event, the routine
686     returns 1. It returns 0 otherwise.
687
688   @Preconditions
689     DMA_Initializer() function should have been
690     called before calling this function.
691
692   @Returns
693     Returns true if there is a Request Collision Event
694
695   @Param
696     None
697
698   @Example
699     Refer to DMA_Initializer(); for an example
700  */
701 inline static bool DMA_IsRequestCollision(uint16_t dmaChannel )
702 {
703     return DMARQC & (1 << dmaChannel);
704 }
705
706 /**
707   @Summary
708     Sets the requested DMA Channel IRQ Select register with the requested
        peripheral IRQ number.
709
710   @Description
711     This routine is used to set the requested DMA Channel IRQ Select register
        with the requested peripheral IRQ number.
712
713   @Preconditions
```

```
714     DMA_Initializer() function should have been
715     called before calling this function.
716
717   @Returns
718     None
719
720   @Param
721     None
722
723   @Example
724     Refer to DMA_Initializer(); for an example
725  */
726 inline static void DMA_PeripheralIrqNumberSet(DMA_CHANNEL channel, uint8_t
      irqNumber)
727 {
728     switch(channel) {
729         case DMA_CHANNEL_0:
730                 DMA0REQ = irqNumber;
731                 break;
732         case DMA_CHANNEL_1:
733                 DMA1REQ = irqNumber;
734                 break;
735         case DMA_CHANNEL_2:
736                 DMA2REQ = irqNumber;
737                 break;
738         case DMA_CHANNEL_3:
739                 DMA3REQ = irqNumber;
740                 break;
741         default: break;
742     }
743 }
744
745 #ifdef __cplusplus  // Provide C++ Compatibility
746
747     }
748
749 #endif
750
751 #endif  // DMA.h
752
753
754 /**
755   End of File
756 */
```

dma.c file

```
/*
    ***************************************************************************

  DMA Generated Driver File

Company:
Microchip Technology Inc.

File Name:
    dma.c

Summary:
This is the generated driver implementation file for the DMA driver using
    PIC24 / dsPIC33 / PIC32MM MCUs

Description:
This source file provides implementations for driver APIs for DMA.
Generation Information :
Product Revision  :  PIC24 / dsPIC33 / PIC32MM MCUs - pic24-dspic-pic32mm :
    1.75.1
Device            :  dsPIC33EV256GM102
The generated drivers are tested against the following:
Compiler          :  XC16 v1.35
MPLAB             :  MPLAB X v5.05
***************************************************************************
    */

/*
    (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this
    software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
    EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY
    IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
    PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS,
    COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
    INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
    BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS
    IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF
    ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF
    THESE
    TERMS.
*/

#include <xc.h>
#include "dma.h"
```

```
47 void DMA_Initialize(void)
48 {
49     // Initialize channels which are enabled
50 
51     // AMODE Peripheral Indirect Addressing mode; CHEN disabled; DIR Reads
       from RAM address, writes to peripheral address; HALF Initiates interrupt
       when all of the data has been moved; SIZE 16 bit; NULLW disabled; MODE
       Continuous, Ping-Pong modes are disabled;
52     DMA0CON= 0x2020 & 0x7FFF; //Enable DMA Channel later;
53     // FORCE disabled; IRQSEL ECAN1 TX;
54     DMA0REQ= 0x46;
55     // CNT 7;
56     DMA0CNT= 0x7;
57     // STA 4096;
58     DMA0STAL= 0x1000;
59     // STA 0;
60     DMA0STAH= 0x0;
61     // Clearing Channel 0 Interrupt Flag;
62     IFS0bits.DMA0IF = false;
63     // Enabling Channel 0 Interrupt
64 
65 
66     // AMODE Peripheral Indirect Addressing mode; CHEN disabled; SIZE 16 bit;
       DIR Reads from peripheral address, writes to RAM address; NULLW disabled;
        HALF Initiates interrupt when all of the data has been moved; MODE
       Continuous, Ping-Pong modes are disabled;
67     DMA1CON= 0x20 & 0x7FFF; //Enable DMA Channel later;
68     // FORCE disabled; IRQSEL ECAN1 RX;
69     DMA1REQ= 0x22;
70     // CNT 7;
71     DMA1CNT= 0x7;
72     // STA 4096;
73     DMA1STAL= 0x1000;
74     // STA 0;
75     DMA1STAH= 0x0;
76     // Clearing Channel 1 Interrupt Flag;
77     IFS0bits.DMA1IF = false;
78     // Enabling Channel 1 Interrupt
79 
80 
81     // AMODE Register Indirect with Post-Increment mode; CHEN disabled; SIZE
       16 bit; DIR Reads from peripheral address, writes to RAM address; NULLW
       disabled; HALF Initiates interrupt when all of the data has been moved;
       MODE Continuous, Ping-Pong modes are disabled;
82     DMA2CON= 0x0 & 0x7FFF; //Enable DMA Channel later;
83     // IRQSEL INT0; FORCE disabled;
84     DMA2REQ= 0x0;
85     // CNT 0;
86     DMA2CNT= 0x0;
87     // STA 4096;
88     DMA2STAL= 0x1000;
89     // STA 0;
90     DMA2STAH= 0x0;
91     // Clearing Channel 2 Interrupt Flag;
92     IFS1bits.DMA2IF = false;
93     // Enabling Channel 2 Interrupt
94 
```

```
95
96      // MODE Continuous, Ping-Pong modes are disabled; AMODE Register Indirect
        with Post-Increment mode; CHEN disabled; HALF Initiates interrupt when
        all of the data has been moved; SIZE 16 bit; DIR Reads from peripheral
        address, writes to RAM address; NULLW disabled;
97      DMA3CON= 0x0 & 0x7FFF; //Enable DMA Channel later;
98      // IRQSEL INT0; FORCE disabled;
99      DMA3REQ= 0x0;
100     // CNT 0;
101     DMA3CNT= 0x0;
102     // STA 4096;
103     DMA3STAL= 0x1000;
104     // STA 0;
105     DMA3STAH= 0x0;
106     // Clearing Channel 3 Interrupt Flag;
107     IFS2bits.DMA3IF = false;
108     // Enabling Channel 3 Interrupt
109
110
111 }
112
113 /**
114   End of File
115 */
```

ecan1.h file

```
/**
  ECAN1 Generated Driver API Header File

  @Company
    Microchip Technology Inc.

  @File Name
    ecan1.h

  @Summary
    This is the generated header file for the ECAN1 driver using PIC24 /
    dsPIC33 / PIC32MM MCUs

  @Description
    This header file provides APIs for driver for ECAN1.
    Generation Information :
        Product Revision  :  PIC24 / dsPIC33 / PIC32MM MCUs - 1.75.1
        Device            :  dsPIC33EV256GM102
        Driver Version    :  1.00
    The generated drivers are tested against the following:
        Compiler          :  XC16 v1.35
        MPLAB             :  MPLAB X v5.05
*/

/*
    (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this
    software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
    EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY
    IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
    PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS,
    COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
    INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
    BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS
    IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF
    ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF
    THESE
    TERMS.
*/

#ifndef _ECAN1_H
#define _ECAN1_H

/**
  Section: Included Files
```

```
51 */
52
53 #include <xc.h>
54 #include <stdint.h>
55 #include <stdbool.h>
56
57 /* ECAN message type identifiers */
58 #define CAN1_MSG_DATA     0x01
59 #define CAN1_MSG_RTR      0x02
60 #define CAN1_FRAME_EXT    0x03
61 #define CAN1_FRAME_STD    0x04
62 #define CAN1_BUF_FULL 0x05
63 #define CAN1_BUF_EMPTY    0x06
64
65 typedef union {
66     struct {
67         uint32_t id;
68         uint8_t idType;
69         uint8_t msgtype;
70         uint8_t dlc;
71         uint8_t data0;
72         uint8_t data1;
73         uint8_t data2;
74         uint8_t data3;
75         uint8_t data4;
76         uint8_t data5;
77         uint8_t data6;
78         uint8_t data7;
79     } frame;
80     unsigned char array[16];
81 } uCAN1_MSG;
82
83 /* Operation modes */
84 typedef enum
85 {
86   CAN1_NORMAL_OPERATION_MODE = 0,
87   CAN1_DISABLE_MODE = 1,
88   CAN1_LOOPBACK_MODE = 2,
89   CAN1_LISTEN_ONLY_MODE = 3,
90   CAN1_CONFIGURATION_MODE = 4,
91   CAN1_LISTEN_ALL_MESSAGES_MODE = 7
92 }ECAN1_OP_MODES;
93
94 typedef enum{
95     ECAN1_PRIORITY_HIGH = 0b11,
96     ECAN1_PRIORITY_MEDIUM = 0b10,
97     ECAN1_PRIORITY_LOW = 0b01,
98     ECAN1_PRIORITY_NONE = 0b00
99 } ECAN1_TX_PRIOIRTY;
100
101 #ifdef __cplusplus  // Provide C++ Compatibility
102
103     extern "C" {
104
105 #endif
106
107 /**
```

```
108   Section: ECAN1 Module APIs
109 */
110
111 /**
112   @Summary
113     Initializes the ECAN1_Initialize.
114
115   @Description
116     This routine initializes the ECAN1_Initialize.
117     This routine must be called before any other ECAN1 routine is called.
118     This routine should only be called once during system initialization.
119
120   @Preconditions
121     None
122
123   @Param
124     None
125
126   @Returns
127     None
128
129   @Comment
130
131
132  @Example
133     <code>
134     ECAN1_Initialize();
135     </code>
136  */
137 void ECAN1_Initialize(void);
138
139 /*
        ***************************************************************************

140 *
141 *     Function:     ECAN1_receive
142 *     Description:        Receives the message from CAN buffer to user buffer
143 *     Arguments:    recCanMsg: pointer to the message object
144 *     Return Value:       true - Receive successful
145 *                         false - Receive failure
146 ***************************************************************************
        */
147 bool ECAN1_receive(uCAN1_MSG *recCanMsg);
148
149 /*
        ***************************************************************************

150 *
151 *     Function:     ECAN1_transmit
152 *     Description:        Transmits the message from user buffer to CAN buffer
153 *     Arguments:    priority: priority of the message to be transmitted
154 *                         sendCanMsg: pointer to the message object
155 *     Return Value:       true - Transmit successful
156 *                         false - Transmit failure
157 ***************************************************************************
        */
158 bool ECAN1_transmit(ECAN1_TX_PRIOIRTY priority,
```

```
159                                     uCAN1_MSG *sendCanMsg);
160
161 /*
        ********************************************************************************
162 *     Function:           ECAN1_isBusOff
163 *     Description:        Checks whether the transmitter in Bus off state
164 *     Return Value:       true - Transmitter in Bus Off state
165 *                         false - Transmitter not in Bus Off state
166 ********************************************************************************
        */
167 bool ECAN1_isBusOff();
168
169 /*
        ********************************************************************************
170 *     Function:     ECAN1_isRXErrorPassive
171 *     Description:        Checks whether the receive in error passive state
172 *     Return Value:       true - Receiver in Error Passive state
173 *                         false - Receiver not in Error Passive state
174 ********************************************************************************
        */
175 bool ECAN1_isRXErrorPassive();
176
177 /*
        ********************************************************************************
178 *     Function:     ECAN1_isTXErrorPassive
179 *     Description:        Checks whether the transmitter in error passive state
180 *     Return Value:       true - Transmitter in Error Passive state
181 *                         false - Transmitter not in Error Passive state
182 ********************************************************************************
        */
183 bool ECAN1_isTXErrorPassive();
184
185 /*
        ********************************************************************************
186 *     Function:     ECAN1_messagesInBuffer
187 *     Description:        returns the number of messages that are received
188 *     Return Value:       Number of message received
189 ********************************************************************************
        */
190 uint8_t ECAN1_messagesInBuffer();
191
192 /*
        ********************************************************************************
193 *
194 *     Function:     ECAN1_sleep
195 *     Description:        Puts ECAN1 module in disable mode.
196 *     Return Value:       None
197 *
198 ********************************************************************************
        */
199 void ECAN1_sleep();
200
```

```
201 /*
        ********************************************************************************
202 *     Function:     ECAN1_TransmitEnable
203 *     Description:        Enables Transmit for ECAN1
204 *     Return Value:       None
205 ********************************************************************************
        */
206 void ECAN1_TransmitEnable();
207
208 /*
        ********************************************************************************
209 *     Function:     ECAN1_ReceiveEnable
210 *     Description:        Enables Receive for ECAN1
211 *     Return Value:       None
212 ********************************************************************************
        */
213 void ECAN1_ReceiveEnable();
214
215 #ifdef __cplusplus  // Provide C++ Compatibility
216
217     }
218
219 #endif
220
221 #endif  //_ECAN1_H
222 /**
223  End of File
224 */
```

ecan1.c file

```
/**
  ECAN1 Generated Driver File

  @Company
    Microchip Technology Inc.

  @File Name
    ecan1.c

  @Summary
    This is the generated driver implementation file for the ECAN1 driver
    using PIC24 / dsPIC33 / PIC32MM MCUs

  @Description
    This source file provides APIs for ECAN1.
    Generation Information :
        Product Revision  :  PIC24 / dsPIC33 / PIC32MM MCUs - 1.75.1
        Device            :  dsPIC33EV256GM102
        Driver Version    :  1.00
    The generated drivers are tested against the following:
        Compiler          :  XC16 v1.35
        MPLAB             :  MPLAB X v5.05
*/

/*
    (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this
    software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
    EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY
    IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
    PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS,
    COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
    INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
    BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS
    IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF
    ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF
    THESE
    TERMS.
*/

/**
  Section: Included Files
*/

#include <xc.h>
```

```
51 #include "ecan1.h"
52 #include "dma.h"
53
54 #define ECAN1_TX_DMA_CHANNEL DMA_CHANNEL_0
55 #define ECAN1_RX_DMA_CHANNEL DMA_CHANNEL_1
56
57 /* Valid options are 4, 6, 8, 12, 16, 24, or 32. */
58 #define CAN1_MESSAGE_BUFFERS          32
59
60 #define CAN1_TX_BUFFER_COUNT 1
61
62 /*****************************************************************************
     */
63
64 /*****************************************************************************
     */
65 /* Private type definitions                                                  */
66 /*****************************************************************************
     */
67 typedef struct __attribute__((packed))
68 {
69     unsigned priority                   :2;
70     unsigned remote_transmit_enable     :1;
71     unsigned send_request               :1;
72     unsigned error                      :1;
73     unsigned lost_arbitration           :1;
74     unsigned message_aborted            :1;
75     unsigned transmit_enabled           :1;
76 } CAN1_TX_CONTROLS;
77
78 /*****************************************************************************
     */
79 /* Private variable definitions
     */
80 /*****************************************************************************
     */
81 /* This alignment is required because of the DMA's peripheral indirect
82  * addressing mode. */
83 static unsigned int ecan1msgBuf [CAN1_MESSAGE_BUFFERS][8] __attribute__((
     aligned(32 * 8 * 2)));
84
85 /*****************************************************************************
     */
86 /* Private function prototypes
     */
87 /*****************************************************************************
     */
88 static void ECAN1_DMACopy(uint8_t buffer_number, uCAN1_MSG *message);
89 static void ECAN1_MessageToBuffer(uint16_t* buffer, uCAN1_MSG* message);
90
91 /* Null weak implementations of callback functions. */
92 void __attribute__((weak)) ECAN1_CallbackBusOff(void){}
93 void __attribute__((weak)) ECAN1_CallbackTxErrorPassive(void){}
94 void __attribute__((weak)) ECAN1_CallbackRxErrorPassive(void){}
95 void __attribute__((weak)) ECAN1_CallbackMessageReceived(void){}
96
97 /**
```

```
98   Section: ECAN1 APIs
99 ******************************************************************************************
       */
100
101 void ECAN1_Initialize(void)
102 {
103     // Disable interrupts before the Initialization
104     IEC2bits.C1IE = 0;
105     C1INTE = 0;
106
107     // set the CAN_initialize module to the options selected in the User
        Interface
108
109     /* put the module in configuration mode */
110     C1CTRL1bits.REQOP = CAN1_CONFIGURATION_MODE;
111     while(C1CTRL1bits.OPMODE != CAN1_CONFIGURATION_MODE);
112
113     /* Set up the baud rate*/
114     C1CFG1 = 0x03;  //BRP TQ = (2 x 4)/FCAN; SJW 1 x TQ;
115     C1CFG2 = 0x1D8; //WAKFIL disabled; SEG2PHTS Freely programmable; SEG2PH 2
        x TQ; SEG1PH 4 x TQ; PRSEG 1 x TQ; SAM Three times at the sample point;
116     C1FCTRL = 0xC001; //FSA Transmit/Receive Buffer TRB1; DMABS 32;
117     C1FEN1 = 0x00;  //FLTEN8 disabled; FLTEN7 disabled; FLTEN9 disabled;
        FLTEN0 disabled; FLTEN2 disabled; FLTEN10 disabled; FLTEN1 disabled;
        FLTEN11 disabled; FLTEN4 disabled; FLTEN3 disabled; FLTEN6 disabled;
        FLTEN5 disabled; FLTEN12 disabled; FLTEN13 disabled; FLTEN14 disabled;
        FLTEN15 disabled;
118     C1CTRL1 = 0x00; //CANCKS FOSC/2; CSIDL disabled; ABAT disabled; REQOP Sets
         Normal Operation Mode; WIN Uses buffer window; CANCAP disabled;
119
120     /* Filter configuration */
121     /* enable window to access the filter configuration registers */
122     /* use filter window*/
123     C1CTRL1bits.WIN=1;
124
125     /* select acceptance masks for filters */
126
127     /* Configure the masks */
128     C1RXM0SIDbits.SID = 0x0;
129     C1RXM1SIDbits.SID = 0x0;
130     C1RXM2SIDbits.SID = 0x0;
131
132     C1RXM0SIDbits.EID = 0x0;
133     C1RXM1SIDbits.EID = 0x0;
134     C1RXM2SIDbits.EID = 0x0;
135
136     C1RXM0EID = 0x00;
137     C1RXM1EID = 0x00;
138     C1RXM2EID = 0x00;
139
140     C1RXM0SIDbits.MIDE = 0x0;
141     C1RXM1SIDbits.MIDE = 0x0;
142     C1RXM2SIDbits.MIDE = 0x0;
143
144     /* Configure the filters */
145
146
```

```
147
148
149     /* Non FIFO Mode */
150
151     /* clear window bit to access ECAN control registers */
152     C1CTRL1bits.WIN=0;
153
154     /* ECAN1, Buffer 0 is a Transmit Buffer */
155     C1TR01CONbits.TXEN0 = 0x1; // Buffer 0 is a Transmit Buffer
156     C1TR01CONbits.TXEN1 = 0x0; // Buffer 1 is a Receive Buffer
157     C1TR23CONbits.TXEN2 = 0x0; // Buffer 2 is a Receive Buffer
158     C1TR23CONbits.TXEN3 = 0x0; // Buffer 3 is a Receive Buffer
159     C1TR45CONbits.TXEN4 = 0x0; // Buffer 4 is a Receive Buffer
160     C1TR45CONbits.TXEN5 = 0x0; // Buffer 5 is a Receive Buffer
161     C1TR67CONbits.TXEN6 = 0x0; // Buffer 6 is a Receive Buffer
162     C1TR67CONbits.TXEN7 = 0x0; // Buffer 7 is a Receive Buffer
163
164     C1TR01CONbits.TX0PRI = 0x0; // Message Buffer 0 Priority Level
165     C1TR01CONbits.TX1PRI = 0x0; // Message Buffer 1 Priority Level
166     C1TR23CONbits.TX2PRI = 0x0; // Message Buffer 2 Priority Level
167     C1TR23CONbits.TX3PRI = 0x0; // Message Buffer 3 Priority Level
168     C1TR45CONbits.TX4PRI = 0x0; // Message Buffer 4 Priority Level
169     C1TR45CONbits.TX5PRI = 0x0; // Message Buffer 5 Priority Level
170     C1TR67CONbits.TX6PRI = 0x0; // Message Buffer 6 Priority Level
171     C1TR67CONbits.TX7PRI = 0x0; // Message Buffer 7 Priority Level
172
173     /* clear the buffer and overflow flags */
174     C1RXFUL1 = 0x0000;
175     C1RXFUL2 = 0x0000;
176     C1RXOVF1 = 0x0000;
177     C1RXOVF2 = 0x0000;
178
179     /* configure the device to interrupt on the receive buffer full flag */
180     /* clear the buffer full flags */
181     C1INTFbits.RBIF = 0;
182
183     /* put the module in normal mode */
184     C1CTRL1bits.REQOP = CAN1_NORMAL_OPERATION_MODE;
185     while(C1CTRL1bits.OPMODE != CAN1_NORMAL_OPERATION_MODE);
186
187     /* Enable ECAN1 Interrupt */
188     IEC2bits.C1IE = 1;
189
190     /* Enable Receive interrupt */
191     C1INTEbits.RBIE = 1;
192
193     /* Enable Error interrupt*/
194     C1INTEbits.ERRIE = 1;
195
196
197 }
198
199 /*
        ******************************************************************************
200 *
201 *     Function:     ECAN1_TransmitEnable
```

```
202 *     Description:      Setup the DMA for Transmit from the ECAN module.  The
203 *                       relevant DMA module APIs are grouped in this function
204 *                       and this API needs to be called after DMA_Initialize
205 *                       and CAN_Initialize
206 *
207 ******************************************************************************
    */
208 void ECAN1_TransmitEnable()
209 {
210     /* setup channel 0 for peripheral indirect addressing mode
211     normal operation, word operation and select as Tx to peripheral */
212
213     /* DMA_PeripheralIrqNumberSet and DMA_TransferCountSet would be done in
        the
214     DMA */
215
216     /* setup the address of the peripheral ECAN1 (C1TXD) */
217     DMA_PeripheralAddressSet(ECAN1_TX_DMA_CHANNEL, &C1TXD);
218
219     /* DPSRAM start address offset value */
220     /* DPSRAM start address offset value */
221     DMA_StartAddressASet(ECAN1_TX_DMA_CHANNEL, (uint16_t)(&ecan1msgBuf));
222
223     /* enable the channel */
224     DMA_ChannelEnable(ECAN1_TX_DMA_CHANNEL);
225 }
226
227 /*
        ******************************************************************************

228 *
229 *     Function:     ECAN1_ReceiveEnable
230 *     Description:      Setup the DMA for Receive on the ECAN module.  The
231 *                       relevant DMA module APIs are grouped in this function
232 *                       and this API needs to be called after DMA_Initialize
233 *                       and CAN_Initialize
234 *
235 ******************************************************************************
    */
236 void ECAN1_ReceiveEnable()
237 {
238     /* setup DMA channel for peripheral indirect addressing mode
239     normal operation, word operation and select as Rx to peripheral */
240
241     /* setup the address of the peripheral ECAN1 (C1RXD) */
242     /* DMA_TransferCountSet and DMA_PeripheralIrqNumberSet would be set in
243     the DMA_Initialize function */
244
245     DMA_PeripheralAddressSet(ECAN1_RX_DMA_CHANNEL, &C1RXD);
246
247     /* DPSRAM start address offset value */
248     DMA_StartAddressASet(ECAN1_RX_DMA_CHANNEL, (uint16_t)(&ecan1msgBuf) );
249
250     /* enable the channel */
251     DMA_ChannelEnable(ECAN1_RX_DMA_CHANNEL);
252 }
253
```

```
254 /*
    ******************************************************************************
255 *
256 *    Function:    ECAN1_transmit
257 *    Description:       Transmits the message from user buffer to CAN buffer
258 *                       as per the buffer number allocated.
259 *                       Allocation of the buffer number is done by user
260 *
261 *    Arguments:   priority : priority of the message to be transmitted
262 *                       sendCanMsg: pointer to the message object
263 *
264 *    Return Value:      true - Transmit successful
265 *                       false - Transmit failure
266 ******************************************************************************
    */
267 bool ECAN1_transmit(ECAN1_TX_PRIOIRTY priority, uCAN1_MSG *sendCanMsg)
268 {
269     CAN1_TX_CONTROLS * pTxControls = (CAN1_TX_CONTROLS*)&C1TR01CON;
270     uint_fast8_t i;
271     bool messageSent = false;
272
273     for(i=0; i<CAN1_TX_BUFFER_COUNT; i++)
274     {
275         if(pTxControls->transmit_enabled == 1)
276         {
277             if (pTxControls->send_request == 0)
278             {
279                 ECAN1_MessageToBuffer( &ecan1msgBuf[i][0], sendCanMsg );
280
281                 pTxControls->priority = priority;
282
283                 /* set the message for transmission */
284                 pTxControls->send_request = 1;
285
286                 messageSent = true;
287                 break;
288             }
289         }
290
291         pTxControls++;
292     }
293
294     return messageSent;
295 }
296
297 /*
    ******************************************************************************
298 *
299 *    Function:    ECAN1_receive
300 *    Description:       Receives the message from CAN buffer to user buffer
301 *
302 *    Arguments:   recCanMsg: pointer to the message object
303 *
304 *    Return Value:      true - Receive successful
305 *                       false - Receive failure
```

```
306 ********************************************************************************
    */
307 bool ECAN1_receive(uCAN1_MSG *recCanMsg)
308 {
309     /* We use a static buffer counter so we don't always check buffer 0 first
310      * resulting in potential starvation of later buffers.
311      */
312     static uint_fast8_t currentDedicatedBuffer = 0;
313     uint_fast8_t i;
314     bool messageReceived = false;
315     uint16_t receptionFlags;
316 
317     receptionFlags = C1RXFUL1;
318 
319     if (receptionFlags != 0)
320     {
321         /* check which message buffer is free */
322         for (i=0 ; i < 16; i++)
323         {
324             if (((receptionFlags >> currentDedicatedBuffer ) & 0x1) == 0x1)
325             {
326                 ECAN1_DMACopy(currentDedicatedBuffer, recCanMsg);
327 
328                 C1RXFUL1 &= ~(1 << currentDedicatedBuffer);
329 
330                 messageReceived = true;
331             }
332 
333             currentDedicatedBuffer++;
334 
335             if(currentDedicatedBuffer >= 16)
336             {
337                 currentDedicatedBuffer = 0;
338             }
339 
340             if(messageReceived == true)
341             {
342                 break;
343             }
344         }
345     }
346 
347     return (messageReceived);
348 }
349 
350 
351 /*
    ********************************************************************************
352 *
353 *     Function:     ECAN1_isBusOff
354 *     Description:        Checks whether the transmitter in Bus off state
355 *
356 
357 *     Return Value:       true – Transmitter in Bus Off state
358 *                         false – Transmitter not in Bus Off state
359 ********************************************************************************
```

```
    */
bool ECAN1_isBusOff()
{
    return C1INTFbits.TXBO;
}

/*
    ******************************************************************************

*
*    Function:    ECAN1_isRXErrorPassive
*    Description:       Checks whether the receive in error passive state
*
*    Return Value:      true - Receiver in Error Passive state
*                       false - Receiver not in Error Passive state
*******************************************************************************
    */
bool ECAN1_isRXErrorPassive()
{
    return C1INTFbits.RXBP;
}

/*
    ******************************************************************************

*
*    Function:    ECAN1_isTXErrorPassive
*    Description:       Checks whether the transmitter in error passive state
*
*    Return Value:      true - Transmitter in Error Passive state
*                       false - Transmitter not in Error Passive state
*******************************************************************************
    */
bool ECAN1_isTXErrorPassive()
{
    return (C1INTFbits.TXBP);
}

/*
    ******************************************************************************

*
*    Function:    ECAN1_messagesInBuffer
*    Description:       returns the number of messages that are received
*
*    Return Value:      Number of message received
*******************************************************************************
    */
uint8_t ECAN1_messagesInBuffer()
{
    uint_fast8_t messageCount;
    uint_fast8_t currentBuffer;
    uint16_t receptionFlags;

    messageCount = 0;

    /* Check any message in buffer 0 to buffer 15*/
```

```
407     receptionFlags = C1RXFUL1;
408     if (receptionFlags != 0)
409     {
410         /* check whether a message is received */
411         for (currentBuffer=0 ; currentBuffer < 16; currentBuffer++)
412         {
413             if (((receptionFlags >> currentBuffer ) & 0x1) == 0x1)
414             {
415                 messageCount++;
416             }
417         }
418     }
419 
420     return (messageCount);
421 }
422 
423 /*
        ********************************************************************************

424 *
425 *     Function:     ECAN1_sleep
426 *     Description:        Puts ECAN1 module in disable mode.
427 *
428 ********************************************************************************
        */
429 void ECAN1_sleep(void) {
430     C1INTFbits.WAKIF = 0;
431     C1INTEbits.WAKIE = 1;
432 
433     /* put the module in disable mode */
434     C1CTRL1bits.REQOP=CAN1_DISABLE_MODE;
435     while(C1CTRL1bits.OPMODE != CAN1_DISABLE_MODE);
436 
437     //Wake up from sleep should set the CAN module straight into Normal mode
438 }
439 
440 /*
        ********************************************************************************

441  * PRIVATE FUNCTIONS
442  ********************************************************************************
        */
443 
444 /*
        ********************************************************************************

445 *
446 *     Function:     ECAN1_DMACopy
447 *     Description:        moves the message from the DMA memory to RAM
448 *
449 *     Arguments:    *message: a pointer to the message structure in RAM
450 *      that will store the message.
451 *
452 *
453 ********************************************************************************
        */
454 static void ECAN1_DMACopy(uint8_t buffer_number, uCAN1_MSG *message)
```

```
455 {
456     uint16_t ide=0;
457     uint16_t rtr=0;
458     uint32_t id=0;
459 
460     /* read word 0 to see the message type */
461     ide=ecan1msgBuf[buffer_number][0] & 0x0001U;
462 
463     /* check to see what type of message it is */
464     /* message is standard identifier */
465     if(ide==0U)
466     {
467         message->frame.id=(ecan1msgBuf[buffer_number][0] & 0x1FFCU) >> 2U;
468         message->frame.idType = CAN1_FRAME_STD;
469         rtr=ecan1msgBuf[buffer_number][0] & 0x0002U;
470     }
471     /* message is extended identifier */
472     else
473     {
474         id=ecan1msgBuf[buffer_number][0] & 0x1FFCU;
475         message->frame.id = id << 16U;
476         message->frame.id += ( ((uint32_t)ecan1msgBuf[buffer_number][1] & (
    uint32_t)0x0FFF) << 6U );
477         message->frame.id += ( ((uint32_t)ecan1msgBuf[buffer_number][2] & (
    uint32_t)0xFC00U) >> 10U );
478         message->frame.idType = CAN1_FRAME_EXT;
479         rtr=ecan1msgBuf[buffer_number][2] & 0x0200;
480     }
481     /* check to see what type of message it is */
482     /* RTR message */
483     if(rtr != 0U)
484     {
485         /* to be defined ?*/
486         message->frame.msgtype = CAN1_MSG_RTR;
487     }
488     /* normal message */
489     else
490     {
491         message->frame.msgtype = CAN1_MSG_DATA;
492         message->frame.data0 =(unsigned char)ecan1msgBuf[buffer_number][3];
493         message->frame.data1 =(unsigned char)((ecan1msgBuf[buffer_number][3] &
     0xFF00U) >> 8U);
494         message->frame.data2 =(unsigned char)ecan1msgBuf[buffer_number][4];
495         message->frame.data3 =(unsigned char)((ecan1msgBuf[buffer_number][4] &
     0xFF00U) >> 8U);
496         message->frame.data4 =(unsigned char)ecan1msgBuf[buffer_number][5];
497         message->frame.data5 =(unsigned char)((ecan1msgBuf[buffer_number][5] &
     0xFF00U) >> 8U);
498         message->frame.data6 =(unsigned char)ecan1msgBuf[buffer_number][6];
499         message->frame.data7 =(unsigned char)((ecan1msgBuf[buffer_number][6] &
     0xFF00U) >> 8U);
500         message->frame.dlc =(unsigned char)(ecan1msgBuf[buffer_number][2] & 0
    x000FU);
501     }
502 }
503 
504 /*
```

```
    ********************************************************************************
505 *
506 *    Function:    ECAN1_MessageToBuffer
507 *    Description:       This function takes the input message, reformats it,
508 *                       and copies it to the specified CAN module buffer
509 *
510 *    Arguments:   *buffer: a pointer to the buffer where the message
511 *                       would be stored
512 *                       *message: pointer to the input message that is
    received
513 *                       by the CAN module
514 *
515 ********************************************************************************
    */
516 static void ECAN1_MessageToBuffer(uint16_t* buffer, uCAN1_MSG* message)
517 {
518     if(message->frame.idType == CAN1_FRAME_STD)
519     {
520         buffer[0]= (message->frame.id & 0x000007FF) << 2;
521         buffer[1]= 0;
522         buffer[2]= message->frame.dlc & 0x0F;
523     }
524     else
525     {
526         buffer[0]= ( ( (uint16_t)(message->frame.id >> 16 ) & 0x1FFC ) ) | 0b1
    ;
527         buffer[1]= (uint16_t)(message->frame.id >> 6) & 0x0FFF;
528         buffer[2]= (message->frame.dlc & 0x0F) + ( (uint16_t)(message->frame.
    id << 10) & 0xFC00);
529     }
530
531     buffer[3]= ((message->frame.data1)<<8) + message->frame.data0;
532     buffer[4]= ((message->frame.data3)<<8) + message->frame.data2;
533     buffer[5]= ((message->frame.data5)<<8) + message->frame.data4;
534     buffer[6]= ((message->frame.data7)<<8) + message->frame.data6;
535 }
536
537
538 void __attribute__((__interrupt__, no_auto_psv)) _C1Interrupt(void)
539 {
540
541     if (C1INTFbits.ERRIF)
542     {
543
544         if (C1INTFbits.TXBO == 1)
545         {
546             ECAN1_CallbackBusOff();
547             C1INTFbits.TXBO = 0;
548         }
549
550         if (C1INTFbits.TXBP == 1)
551         {
552             ECAN1_CallbackTxErrorPassive();
553             C1INTFbits.TXBP = 0;
554         }
555
```

```
556         if (C1INTFbits.RXBP == 1)
557         {
558             ECAN1_CallbackRxErrorPassive();
559             C1INTFbits.RXBP = 0;
560         }
561 
562         /* Call error notification function */
563         C1INTFbits.ERRIF = 0;
564 
565     }
566 
567     if(C1INTFbits.RBIF)
568     {
569         C1INTFbits.RBIF = 0;
570 
571         /* Notification function */
572         ECAN1_CallbackMessageReceived();
573     }
574 
575 
576     IFS2bits.C1IF = 0;
577 }
578 
579 
580 
581 /**
582  End of File
583 */
```

interrupt_manager.h file

```
/**
  System Interrupts Generated Driver File

  @Company:
    Microchip Technology Inc.

  @File Name:
    interrupt_manager.h

  @Summary:
    This is the generated driver implementation file for setting up the
    interrupts using PIC24 / dsPIC33 / PIC32MM MCUs

  @Description:
    This source file provides implementations for PIC24 / dsPIC33 / PIC32MM
    MCUs interrupts.
    Generation Information :
        Product Revision  :  PIC24 / dsPIC33 / PIC32MM MCUs - pic24-dspic-
    pic32mm : 1.75.1
        Device            :  dsPIC33EV256GM102
    The generated drivers are tested against the following:
        Compiler          :  XC16 v1.35
        MPLAB             :  MPLAB X v5.05
*/
/*
    (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this
    software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
    EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY
    IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
    PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS,
    COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
    INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
    BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS
    IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF
    ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF
    THESE
    TERMS.
*/

#ifndef _INTERRUPT_MANAGER_H
#define _INTERRUPT_MANAGER_H

/**
  @Summary
```

```
50      Initializes the interrupt priorities of the dsPIC33EV256GM102
51
52    @Description
53      This routine sets the interrupt priorities of the modules that have been
        configured
54      for the dsPIC33EV256GM102
55
56    @Preconditions
57      None.
58
59    @Returns
60      None.
61
62    @Param
63      None.
64
65    @Example
66      <code>
67      void SYSTEM_Initialize(void)
68      {
69          // Other initializers are called from this function
70          INTERRUPT_Initialize ();
71      }
72      </code>
73
74  */
75  void INTERRUPT_Initialize(void);
76
77  /**
78    @Summary
79      Enables global interrupts of the dsPIC33EV256GM102
80
81    @Description
82      This routine enables the global interrupt bit for the dsPIC33EV256GM102
83
84    @Preconditions
85      None.
86
87    @Returns
88      None.
89
90    @Param
91      None.
92
93    @Example
94      <code>
95      void SYSTEM_Initialize(void)
96      {
97          // Other initializers are called from this function
98          INTERRUPT_GlobalEnable ();
99      }
100     </code>
101
102 */
103 inline static void INTERRUPT_GlobalEnable(void)
104 {
105     __builtin_enable_interrupts();
```

```
106 }
107
108 /**
109   @Summary
110     Disables global interrupts of the dsPIC33EV256GM102
111
112   @Description
113     This routine disables the global interrupt bit for the dsPIC33EV256GM102
114
115   @Preconditions
116     None.
117
118   @Returns
119     None.
120
121   @Param
122     None.
123
124   @Example
125     <code>
126     void SYSTEM_Initialize(void)
127     {
128         // Other initializers are called from this function
129         INTERRUPT_GlobalDisable ();
130     }
131     </code>
132
133 */
134 inline static void INTERRUPT_GlobalDisable(void)
135 {
136     __builtin_disable_interrupts();
137 }
138
139
140 #endif
```

interrupt_manager.c file

```
/**
  System Interrupts Generated Driver File

  @Company:
    Microchip Technology Inc.

  @File Name:
    interrupt_manager.h

  @Summary:
    This is the generated driver implementation file for setting up the
    interrupts using PIC24 / dsPIC33 / PIC32MM MCUs

  @Description:
    This source file provides implementations for PIC24 / dsPIC33 / PIC32MM
    MCUs interrupts.
    Generation Information :
        Product Revision  :  PIC24 / dsPIC33 / PIC32MM MCUs - pic24-dspic-
    pic32mm : 1.75.1
        Device            :  dsPIC33EV256GM102
    The generated drivers are tested against the following:
        Compiler          :  XC16 v1.35
        MPLAB             :  MPLAB X v5.05
*/
/*
    (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this
    software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
    EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY
    IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
    PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS,
    COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
    INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
    BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS
    IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF
    ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF
    THESE
    TERMS.
*/

/**
    Section: Includes
*/
#include <xc.h>

```

```
50 /**
51     void INTERRUPT_Initialize (void)
52 */
53 void INTERRUPT_Initialize (void)
54 {
55     //     CI: ECAN1 Event
56     //     Priority: 1
57         IPC8bits.C1IP = 1;
58 }
```

mcc.h file

```
/**
  @Generated PIC24 / dsPIC33 / PIC32MM MCUs Header File

  @Company:
    Microchip Technology Inc.

  @File Name:
    mcc.h

  @Summary:
    This is the mcc.h file generated using PIC24 / dsPIC33 / PIC32MM MCUs

  @Description:
    This header file provides implementations for driver APIs for all modules selected in the GUI.
    Generation Information :
        Product Revision  :  PIC24 / dsPIC33 / PIC32MM MCUs - pic24-dspic-pic32mm : 1.75.1
        Device            :  dsPIC33EV256GM102
    The generated drivers are tested against the following:
        Compiler          :  XC16 v1.35
        MPLAB             :  MPLAB X v5.05
*/

/*
    (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS, COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF THESE
    TERMS.
*/

#ifndef MCC_H
#define MCC_H
#include <xc.h>
#include "reset.h"
#include "system.h"
```

```
50 #include "system_types.h"
51 #include "clock.h"
52 #include "pin_manager.h"
53 #include <stdint.h>
54 #include <stdbool.h>
55 #include "ecan1.h"
56 #include "adc1.h"
57 #include "dma.h"
58 #include "reset.h"
59 #include "watchdog.h"
60 #include "interrupt_manager.h"
61 #include "traps.h"
62
63 #define _XTAL_FREQ  138187500UL
64
65 /**
66  * @Param
67     none
68  * @Returns
69     none
70  * @Description
71     Initializes the oscillator to the default states configured in the
72  *                  MCC GUI
73  * @Example
74     OSCILLATOR_Initialize(void);
75  */
76 void OSCILLATOR_Initialize(void) __attribute__((deprecated ("\nThis will be
        removed in future MCC releases. \nUse CLOCK_Initialize (void) instead. ")
        ));
77
78 /**
79 * Checks reset cause, flashes UI with an error code as a result.
80 *
81 * Note: this function should be called before any use of CLRWDT
82 * since it has a side-effect of clearing the appropriate bits in the
83 * register showing reset cause (see DS70602B page 8-10)
84 */
85 uint16_t SYSTEM_GetResetCause(void) __attribute__((deprecated ("\nThis will be
         removed in future MCC releases. \nUse RESET_GetCause(void) (void)
        instead. ")));
86
87
88 /**
89  * Enables Watch Dog Timer (WDT) using the software bit.
90  * @example
91  * <code>
92  * WDT_WatchdogtimerSoftwareEnable();
93  * </code>
94  */
95 __attribute__((deprecated ("\nThis will be removed in future MCC releases. \
        nUse WATCHDOG_TimerSoftwareEnable (void) instead. ")))
96 inline static void WDT_WatchdogtimerSoftwareEnable(void)
97 {
98     RCONbits.SWDTEN = 1;
99 }
100
101 /**
```

```
102  * Disables Watch Dog Timer (WDT) using the software bit.
103  * @example
104  * <code>
105  * WDT_WatchdogtimerSoftwareDisable();
106  * </code>
107  */
108 __attribute__((deprecated ("\nThis will be removed in future MCC releases. \
         nUse WATCHDOG_TimerSoftwareDisable(void) instead.")))
109 inline static void WDT_WatchdogtimerSoftwareDisable(void)
110 {
111     RCONbits.SWDTEN = 0;
112 }
113
114 /**
115  * Clears the Watch Dog Timer (WDT).
116  * @example
117  * <code>
118  * WDT_WatchdogTimerClear();
119  * </code>
120  */
121 __attribute__((deprecated ("\nThis will be removed in future MCC releases. \
         nUse WATCHDOG_TimerClear(void) instead.")))
122 inline static void WDT_WatchdogTimerClear(void)
123 {
124     ClrWdt();
125 }
126
127 /**
128  * Gets the base address of the DEVID register for the currently selected
         device
129  * @return base address of the DEVID register
130  * @example
131  * <code>
132  * uint32_t devIdAddress;
133  * devIdAddress = DEVICE_DeviceIdRegisterAddressGet();
134  * </code>
135  */
136 __attribute__((deprecated ("\nThis will be removed in future MCC releases. \
         nUse SYSTEM_DeviceIdRegisterAddressGet(void) instead.")))
137 inline static uint32_t DEVICE_DeviceIdRegisterAddressGet(void)
138 {
139     return __DEVID_BASE;
140 }
141
142 /**
143  * Initializes the CPU core control register.
144  * @example
145  * <code>
146  * CORCON_Initialize();
147  * </code>
148  */
149 __attribute__((deprecated ("\nThis will be removed in future MCC releases. \
         nUse SYSTEM_CORCONInitialize() instead.")))
150 inline static void CORCON_Initialize()
151 {
152     CORCON = (CORCON & 0x00F2) | CORCON_MODE_PORVALUES;     // POR value
153 }
```

```
154
155 /**
156  * Sets the CPU core control register operating mode to a value that is
        decided by the
157  * SYSTEM_CORCON_MODES argument.
158  * @param modeValue SYSTEM_CORCON_MODES initialization mode specifier
159  * @example
160  * <code>
161  * CORCON_ModeOperatingSet(CORCON_MODE_ENABLEALLSATNORMAL_ROUNDUNBIASED);
162  * </code>
163  */
164 __attribute__((deprecated ("\nThis will be removed in future MCC releases. \
        nUse SYSTEM_CORCONModeOperatingSet(SYSTEM_CORCON_MODES modeValue) instead
        . ")))
165 inline static void CORCON_ModeOperatingSet(SYSTEM_CORCON_MODES modeValue)
166 {
167     CORCON = (CORCON & 0x00F2) | modeValue;
168 }
169
170 /**
171  * Sets the value of CPU core control register.
172  * @param value value that needs to be written to the CPU core control
        register
173  * @example
174  * <code>
175  * CORCON_RegisterValueSet(0x00E2);
176  * </code>
177  */
178 __attribute__((deprecated ("\nThis will be removed in future MCC releases. \
        nUse SYSTEM_CORCONRegisterValueSet(uint16_t value) instead. ")))
179 inline static void CORCON_RegisterValueSet(uint16_t value)
180 {
181     CORCON = value;
182 }
183
184 /**
185  * Gets the value of CPU core control register.
186  * @return value of the CPU core control register
187  * @example
188  * <code>
189  * corconSave = CORCON_RegisterValueGet();
190  * </code>
191  */
192 __attribute__((deprecated ("\nThis will be removed in future MCC releases. \
        nUse SYSTEM_CORCONRegisterValueGet (void) instead. ")))
193 inline static uint16_t CORCON_RegisterValueGet(void)
194 {
195     return CORCON;
196 }
197
198 /**
199  * It handles the reset cause by clearing the cause register values.
200  * Its a weak function user can override this function.
201  * @return None
202  * @example
203  * <code>
204  * SYSTEM_ResetCauseHandler();
```

```
205  * </code>
206  */
207 void __attribute__ ((weak)) SYSTEM_ResetCauseHandler(void) __attribute__((
        deprecated ("\nThis will be removed in future MCC releases. \nUse
        RESET_CauseHandler(void) (void) instead. ")));
208
209 /**
210  * This function resets the reset cause register.
211  * @return None
212  * @example
213  * <code>
214  * SYSTEM_ResetCauseClearAll();
215  * </code>
216  */
217 void SYSTEM_ResetCauseClearAll() __attribute__((deprecated ("\nThis will be
        removed in future MCC releases. \nUse RESET_CauseClearAll(void) (void)
        instead. ")));
218
219 #endif  /* MCC_H */
220 /**
221  End of File
222 */
```

mcc.c file

```
/**
  @Generated PIC24 / dsPIC33 / PIC32MM MCUs Source File

  @Company:
    Microchip Technology Inc.

  @File Name:
    mcc.c

  @Summary:
    This is the mcc.c file generated using PIC24 / dsPIC33 / PIC32MM MCUs

  @Description:
    This header file provides implementations for driver APIs for all modules selected in the GUI.
    Generation Information :
        Product Revision  :  PIC24 / dsPIC33 / PIC32MM MCUs - pic24-dspic-pic32mm : 1.75.1
        Device            :  dsPIC33EV256GM102
    The generated drivers are tested against the following:
        Compiler          :  XC16 v1.35
        MPLAB             :  MPLAB X v5.05
*/

/*
    (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS, COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF THESE
    TERMS.
*/

// Configuration bits: selected in the GUI

// FSEC
#pragma config BWRP = OFF    // Boot Segment Write-Protect Bit->Boot Segment may be written
```

```
49 #pragma config BSS = DISABLED     // Boot Segment Code-Protect Level bits->No
       Protection (other than BWRP)
50 #pragma config BSS2 = OFF     // Boot Segment Control Bit->No Boot Segment
51 #pragma config GWRP = OFF     // General Segment Write-Protect Bit->General
       Segment may be written
52 #pragma config GSS = DISABLED     // General Segment Code-Protect Level bits->
       No Protection (other than GWRP)
53 #pragma config CWRP = OFF     // Configuration Segment Write-Protect Bit->
       Configuration Segment may be written
54 #pragma config CSS = DISABLED     // Configuration Segment Code-Protect Level
       bits->No Protection (other than CWRP)
55 #pragma config AIVTDIS = DISABLE     // Alternate Interrupt Vector Table
       Disable Bit->Disable Alternate Vector Table
56
57 // FBSLIM
58 #pragma config BSLIM = 8191     // Boot Segment Code Flash Page Address Limit
       Bits->
59
60 // FOSCSEL
61 #pragma config FNOSC = FRC     // Initial oscillator Source Selection Bits->FRC
62 #pragma config IESO = ON     // Two Speed Oscillator Start-Up Bit->Start up
       device with FRC, then automatically switch to user selected oscillator
       source
63
64 // FOSC
65 #pragma config POSCMD = NONE     // Primary Oscillator Mode Select Bits->
       Primary Oscillator disabled
66 #pragma config OSCIOFNC = ON     // OSC2 Pin I/O Function Enable Bit->OSC2 is
       general purpose digital I/O pin
67 #pragma config IOL1WAY = ON     // Peripheral Pin Select Configuration Bit->
       Allow Only One reconfiguration
68 #pragma config FCKSM = CSECMD     // Clock Switching Mode Bits->Clock Switching
        is enabled, Fail-safe Clock Monitor is disabled
69 #pragma config PLLKEN = ON     // PLL Lock Enable Bit->Clock switch to PLL
       source will wait until the PLL lock signal is valid
70
71 // FWDT
72 #pragma config WDTPOST = PS32768     // Watchdog Timer Postscaler Bits->1:32768
73 #pragma config WDTPRE = PR128     // Watchdog Timer Prescaler Bit->1:128
74 #pragma config FWDTEN = OFF     // Watchdog Timer Enable Bits->WDT and SWDTEN
       Disabled
75 #pragma config WINDIS = OFF     // Watchdog Timer Window Enable Bit->Watchdog
       timer in Non-Window Mode
76 #pragma config WDTWIN = WIN25     // Watchdog Window Select Bits->WDT Window is
        25% of WDT period
77
78 // FPOR
79 #pragma config BOREN0 = OFF     // Brown Out Reset Detection Bit->BOR is
       Disabled
80
81 // FICD
82 #pragma config ICS = PGD1     // ICD Communication Channel Select Bits->
       Communicate on PGEC1 and PGED1
83
84 // FDMTINTVL
85 #pragma config DMTIVTL = 0     // Lower 16 Bits of 32 Bit DMT Window Interval->
86
```

```
87 // FDMTINTVH
88 #pragma config DMTIVTH = 0     // Upper 16 Bits of 32 Bit DMT Window Interval->
89
90 // FDMTCNTL
91 #pragma config DMTCNTL = 0     // Lower 16 Bits of 32 Bit DMT Instruction Count
       Time-Out Value->
92
93 // FDMTCNTH
94 #pragma config DMTCNTH = 0     // Upper 16 Bits of 32 Bit DMT Instruction Count
       Time-Out Value->
95
96 // FDMT
97 #pragma config DMTEN = DISABLE     // Dead Man Timer Enable Bit->Dead Man Timer
       is Disabled and can be enabled by software
98
99 // FDEVOPT
100 #pragma config PWMLOCK = ON     // PWM Lock Enable Bit->Certain PWM registers
      may only be written after key sequence
101 #pragma config ALTI2C1 = OFF     // Alternate I2C1 Pins Selection Bit->I2C1
      mapped to SDA1/SCL1 pins
102
103 // FALTREG
104 #pragma config CTXT1 = NONE     // Interrupt Priority Level (IPL) Selection
      Bits For Alternate Working Register Set 1->Not Assigned
105 #pragma config CTXT2 = NONE     // Interrupt Priority Level (IPL) Selection
      Bits For Alternate Working Register Set 2->Not Assigned
106
107 #include "reset.h"
108 #include "mcc.h"
109
110 /**
111  Section: Local Variables
112 */
113
114 /**
115  Section: Function prototypes
116 */
117
118 /**
119 * a private place to store the error code if we run into a severe error
120 */
121
122 void OSCILLATOR_Initialize(void)
123 {
124     CLOCK_Initialize();
125 }
126
127 uint16_t SYSTEM_GetResetCause(void)
128 {
129     return RESET_GetCause();
130 }
131
132 void __attribute__ ((weak)) SYSTEM_ResetCauseHandler(void)
133 {
134     RESET_CauseHandler();
135 }
136
```

```
137 void SYSTEM_ResetCauseClearAll()
138 {
139     RESET_CauseClearAll();
140 }
141 /**
142  End of File
143 */
```

pin_manager.h file

```
/**
  System Interrupts Generated Driver File

  @Company:
    Microchip Technology Inc.

  @File Name:
    pin_manager.h

  @Summary:
    This is the generated manager file for the MPLAB(c) Code Configurator device.  This manager
    configures the pins direction, initial state, analog setting.
    The peripheral pin select, PPS, configuration is also handled by this manager.

  @Description:
    This source file provides implementations for MPLAB(c) Code Configurator interrupts.
    Generation Information :
        Product Revision  :  MPLAB(c) Code Configurator - pic24-dspic-pic32mm : 1.75.1
        Device            :  dsPIC33EV256GM102
    The generated drivers are tested against the following:
        Compiler          :  XC16 v1.35
        MPLAB             :  MPLAB X v5.05

    Copyright (c) 2013 - 2015 released Microchip Technology Inc.  All rights reserved.

    Microchip licenses to you the right to use, modify, copy and distribute
    Software only when embedded on a Microchip microcontroller or digital signal
    controller that is integrated into your product or third party product
    (pursuant to the sublicense terms in the accompanying license agreement).

    You should refer to the license agreement accompanying this Software for additional information regarding your rights and obligations.

    SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND,
    EITHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF
    MERCHANTABILITY, TITLE, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE.
    IN NO EVENT SHALL MICROCHIP OR ITS LICENSORS BE LIABLE OR OBLIGATED UNDER
    CONTRACT, NEGLIGENCE, STRICT LIABILITY, CONTRIBUTION, BREACH OF WARRANTY, OR
    OTHER LEGAL EQUITABLE THEORY ANY DIRECT OR INDIRECT DAMAGES OR EXPENSES
    INCLUDING BUT NOT LIMITED TO ANY INCIDENTAL, SPECIAL, INDIRECT, PUNITIVE OR
    CONSEQUENTIAL DAMAGES, LOST PROFITS OR LOST DATA, COST OF PROCUREMENT OF
    SUBSTITUTE GOODS, TECHNOLOGY, SERVICES, OR ANY CLAIMS BY THIRD PARTIES
    (INCLUDING BUT NOT LIMITED TO ANY DEFENSE THEREOF), OR OTHER SIMILAR COSTS.

*/
```

```
46
47 #ifndef _PIN_MANAGER_H
48 #define _PIN_MANAGER_H
49 /**
50     Section: Includes
51 */
52 #include <xc.h>
53 /**
54     Section: Device Pin Macros
55 */
56 /**
57   @Summary
58     Sets the GPIO pin, RB7, high using LATB7.
59
60   @Description
61     Sets the GPIO pin, RB7, high using LATB7.
62
63   @Preconditions
64     The RB7 must be set to an output.
65
66   @Returns
67     None.
68
69   @Param
70     None.
71
72   @Example
73     <code>
74     // Set RB7 high (1)
75     Speed_SetHigh();
76     </code>
77
78 */
79 #define Speed_SetHigh()           _LATB7 = 1
80 /**
81   @Summary
82     Sets the GPIO pin, RB7, low using LATB7.
83
84   @Description
85     Sets the GPIO pin, RB7, low using LATB7.
86
87   @Preconditions
88     The RB7 must be set to an output.
89
90   @Returns
91     None.
92
93   @Param
94     None.
95
96   @Example
97     <code>
98     // Set RB7 low (0)
99     Speed_SetLow();
100    </code>
101
102 */
```

```
103 #define Speed_SetLow()            _LATB7 = 0
104 /**
105   @Summary
106     Toggles the GPIO pin, RB7, using LATB7.
107
108   @Description
109     Toggles the GPIO pin, RB7, using LATB7.
110
111   @Preconditions
112     The RB7 must be set to an output.
113
114   @Returns
115     None.
116
117   @Param
118     None.
119
120   @Example
121     <code>
122     // Toggle RB7
123     Speed_Toggle();
124     </code>
125
126 */
127 #define Speed_Toggle()            _LATB7 ^= 1
128 /**
129   @Summary
130     Reads the value of the GPIO pin, RB7.
131
132   @Description
133     Reads the value of the GPIO pin, RB7.
134
135   @Preconditions
136     None.
137
138   @Returns
139     None.
140
141   @Param
142     None.
143
144   @Example
145     <code>
146     uint16_t portValue;
147
148     // Read RB7
149     postValue = Speed_GetValue();
150     </code>
151
152 */
153 #define Speed_GetValue()          _RB7
154 /**
155   @Summary
156     Configures the GPIO pin, RB7, as an input.
157
158   @Description
159     Configures the GPIO pin, RB7, as an input.
```

```
160
161   @Preconditions
162     None.
163
164   @Returns
165     None.
166
167   @Param
168     None.
169
170   @Example
171     <code>
172     // Sets the RB7 as an input
173     Speed_SetDigitalInput();
174     </code>
175
176 */
177 #define Speed_SetDigitalInput()   _TRISB7 = 1
178 /**
179   @Summary
180     Configures the GPIO pin, RB7, as an output.
181
182   @Description
183     Configures the GPIO pin, RB7, as an output.
184
185   @Preconditions
186     None.
187
188   @Returns
189     None.
190
191   @Param
192     None.
193
194   @Example
195     <code>
196     // Sets the RB7 as an output
197     Speed_SetDigitalOutput();
198     </code>
199
200 */
201 #define Speed_SetDigitalOutput()  _TRISB7 = 0
202
203 /**
204     Section: Function Prototypes
205 */
206 /**
207   @Summary
208     Configures the pin settings of the dsPIC33EV256GM102
209     The peripheral pin select, PPS, configuration is also handled by this
      manager.
210
211   @Description
212     This is the generated manager file for the MPLAB(c) Code Configurator
      device.  This manager
213     configures the pins direction, initial state, analog setting.
214     The peripheral pin select, PPS, configuration is also handled by this
```

```
    manager.
215
216  @Preconditions
217    None.
218
219  @Returns
220    None.
221
222  @Param
223    None.
224
225  @Example
226    <code>
227    void SYSTEM_Initialize(void)
228    {
229        // Other initializers are called from this function
230        PIN_MANAGER_Initialize();
231    }
232    </code>
233
234 */
235 void PIN_MANAGER_Initialize(void);
236
237 #endif
```

pin_manager.c file

```
/**
  System Interrupts Generated Driver File

  @Company:
    Microchip Technology Inc.

  @File Name:
    pin_manager.c

  @Summary:
    This is the generated manager file for the MPLAB(c) Code Configurator device.  This manager
    configures the pins direction, initial state, analog setting.
    The peripheral pin select, PPS, configuration is also handled by this manager.

  @Description:
    This source file provides implementations for MPLAB(c) Code Configurator interrupts.
    Generation Information :
        Product Revision  :  MPLAB(c) Code Configurator - pic24-dspic-pic32mm : 1.75.1
        Device            :  dsPIC33EV256GM102
    The generated drivers are tested against the following:
        Compiler          :  XC16 v1.35
        MPLAB             :  MPLAB X v5.05

    Copyright (c) 2013 - 2015 released Microchip Technology Inc.  All rights reserved.

    Microchip licenses to you the right to use, modify, copy and distribute
    Software only when embedded on a Microchip microcontroller or digital signal
    controller that is integrated into your product or third party product
    (pursuant to the sublicense terms in the accompanying license agreement).

    You should refer to the license agreement accompanying this Software for additional information regarding your rights and obligations.

    SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND,
    EITHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF
    MERCHANTABILITY, TITLE, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE.
    IN NO EVENT SHALL MICROCHIP OR ITS LICENSORS BE LIABLE OR OBLIGATED UNDER
    CONTRACT, NEGLIGENCE, STRICT LIABILITY, CONTRIBUTION, BREACH OF WARRANTY, OR
    OTHER LEGAL EQUITABLE THEORY ANY DIRECT OR INDIRECT DAMAGES OR EXPENSES
    INCLUDING BUT NOT LIMITED TO ANY INCIDENTAL, SPECIAL, INDIRECT, PUNITIVE OR
    CONSEQUENTIAL DAMAGES, LOST PROFITS OR LOST DATA, COST OF PROCUREMENT OF
    SUBSTITUTE GOODS, TECHNOLOGY, SERVICES, OR ANY CLAIMS BY THIRD PARTIES
    (INCLUDING BUT NOT LIMITED TO ANY DEFENSE THEREOF), OR OTHER SIMILAR COSTS.

*/
```

```
46
47
48  /**
49      Section: Includes
50  */
51  #include <xc.h>
52  #include "pin_manager.h"
53
54  /**
55      void PIN_MANAGER_Initialize(void)
56  */
57  void PIN_MANAGER_Initialize(void)
58  {
59      /*
        ****************************************************************************
60       * Setting the Output Latch SFR(s)
61
        ****************************************************************************
        */
62      LATA = 0x0000;
63      LATB = 0x0000;
64
65      /*
        ****************************************************************************
66       * Setting the GPIO Direction SFR(s)
67
        ****************************************************************************
        */
68      TRISA = 0x000F;
69      TRISB = 0xFFFF;
70
71      /*
        ****************************************************************************
72       * Setting the Weak Pull Up and Weak Pull Down SFR(s)
73
        ****************************************************************************
        */
74      CNPDA = 0x0000;
75      CNPDB = 0x0000;
76      CNPUA = 0x0000;
77      CNPUB = 0x0000;
78
79      /*
        ****************************************************************************
80       * Setting the Open Drain SFR(s)
81
        ****************************************************************************
        */
82      ODCA = 0x0000;
83      ODCB = 0x0000;
84
85      /*
        ****************************************************************************
```

```
86      * Setting the Analog/Digital Configuration SFR(s)
87
        *********************************************************************
        */
88      ANSELA = 0x0007;
89      ANSELB = 0x0383;
90
91
92      /*
        *********************************************************************
93       * Set the PPS
94
        *********************************************************************
        */
95      __builtin_write_OSCCONL(OSCCON & 0xbf); // unlock PPS
96
97      RPOR0bits.RP20R = 0x000E;     //RA4->ECAN1:C1TX;
98      RPINR26bits.C1RXR = 0x0024;     //RB4->ECAN1:C1RX;
99
100     __builtin_write_OSCCONL(OSCCON | 0x40); // lock   PPS
101
102 }
```

reset.h file

```
/**
  RESET Generated Driver File

  @Company
    Microchip Technology Inc.

  @File Name
    reset.c

  @Summary
    This is the generated driver implementation file for the RESET driver
    using PIC24 / dsPIC33 / PIC32MM MCUs

  @Description
    This header file provides implementations for driver APIs for RESET.
    Generation Information :
        Product Revision  :  PIC24 / dsPIC33 / PIC32MM MCUs - 1.75.1
        Device            :  dsPIC33EV256GM102
    The generated drivers are tested against the following:
        Compiler          :  XC16 v1.35
        MPLAB             :  MPLAB X v5.05
*/

/*
    (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this
    software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
    EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY
    IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
    PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS,
    COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
    INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
    BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS
    IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF
    ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF
    THESE
    TERMS.
*/

#ifndef RESET_H
#define RESET_H

#include <stdint.h>
#include "reset_types.h"

```

```
51 /**
52 * Checks reset cause, flashes UI with an error code as a result.
53 *
54 * Note: this function should be called before any use of CLRWDT
55 * since it has a side-effect of clearing the appropriate bits in the
56 * register showing reset cause (see DS70602B page 8-10)
57 */
58 uint16_t RESET_GetCause(void);
59
60 /**
61  * It handles the reset cause by clearing the cause register values.
62  * Its a weak function user can override this function.
63  * @return None
64  * @example
65  * <code>
66  * RESET_CauseHandler();
67  * </code>
68  */
69 void __attribute__ ((weak)) RESET_CauseHandler(void);
70
71 /**
72  * This function resets the reset cause register.
73  * @return None
74  * @example
75  * <code>
76  * RESET_CauseClearAll();
77  * </code>
78  */
79 void RESET_CauseClearAll();
80
81 #endif  /* RESET_H */
82 /**
83  End of File
84 */
```

reset_types.h file

```
/**
  RESET Generated Driver File

  @Company
    Microchip Technology Inc.

  @File Name
    reset_types.h

  @Summary
    This is the generated driver implementation file for the RESET driver
    using PIC24 / dsPIC33 / PIC32MM MCUs

  @Description
    This header file provides implementations for driver APIs for RESET.
    Generation Information :
        Product Revision  :  PIC24 / dsPIC33 / PIC32MM MCUs - 1.75.1
        Device            :  dsPIC33EV256GM102
    The generated drivers are tested against the following:
        Compiler          :  XC16 v1.35
        MPLAB             :  MPLAB X v5.05
*/

/*
    (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this
    software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
    EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY
    IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
    PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS,
    COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
    INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
    BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS
    IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF
    ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF
    THESE
    TERMS.
*/

#ifndef RESET_TYPES_H
#define RESET_TYPES_H

/**
  Section: Type defines
 */
```

```
51
52 /**
53  * RCON error type enumerator. Supported types:
54  * ERR_RCON_TRAPR
55  * ERR_RCON_IOPUWR
56  * ERR_RCON_CM
57  * ERR_RCON_WDTO_ISR
58  */
59 typedef enum tagERROR_TYPE
60 {
61     ERR_RCON_TRAPR      = 1, /** A Trap Conflict Reset has occurred **/
62     ERR_RCON_IOPUWR     = 2, /** An illegal opcode detection, an illegal
    address mode or Uninitialized W register used as an
63                              *   Address Pointer caused a Reset **/
64     ERR_RCON_CM         = 3, /** A Configuration Mismatch Reset has occurred *
    */
65     ERR_RCON_WDTO_ISR   = 4  /** WDT time-out has occurred **/
66 }RESET_TYPES;
67
68 /**
69  * RESET CAUSE Masks. Supported masks:
70  * RESET_MASK_WDTO
71  * RESET_MASK_SWR
72  * REST_MASK_EXTR
73  * RESET_MASK_CM
74  * RESET_MASK_IOPUWR
75  * RESET_MASK_TRAPR
76  */
77 typedef enum tagRESET_MASKS
78 {
79   RESET_MASK_WDTO = 0x0010,
80   RESET_MASK_SWR = 0x0040,
81   RESET_MASK_EXTR = 0x0080,
82   RESET_MASK_CM = 0x0200,
83   RESET_MASK_IOPUWR = 0x4000,
84   RESET_MASK_TRAPR = 0x8000,
85 } RESET_MASKS;
86
87 #endif  /* RESET_TYPES_H */
88 /**
89  End of File
90 */
```

reset.c file

```
/**
  RESET Generated Driver File

  @Company
    Microchip Technology Inc.

  @File Name
    reset.c

  @Summary
    This is the generated driver implementation file for the RESET driver
    using PIC24 / dsPIC33 / PIC32MM MCUs

  @Description
    This header file provides implementations for driver APIs for RESET.
    Generation Information :
        Product Revision  :  PIC24 / dsPIC33 / PIC32MM MCUs - 1.75.1
        Device            :  dsPIC33EV256GM102
    The generated drivers are tested against the following:
        Compiler          :  XC16 v1.35
        MPLAB             :  MPLAB X v5.05
*/

/*
    (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this
    software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
    EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY
    IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
    PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS,
    COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
    INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
    BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS
    IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF
    ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF
    THESE
    TERMS.
*/

#include <stdbool.h>
#include <stdint.h>
#include "xc.h"
#include "reset.h"

/**
```

```
51  Section: Local Variables
52 */
53
54 /**
55  Section: Function prototypes
56 */
57 static bool RESET_CauseFromSoftware(uint16_t resetCause);
58 static bool RESET_CauseFromWatchdogTimer(uint16_t resetCause);
59 static bool RESET_CauseFromConfigurationMismatch(uint16_t resetCause);
60 static bool RESET_CauseFromIllegalOpcode(uint16_t resetCause);
61 static bool RESET_CauseFromExternal(uint16_t resetCause);
62 static bool RESET_CauseFromTrap(uint16_t resetCause);
63 static void RESET_CauseClear(RESET_MASKS resetFlagMask);
64
65 uint16_t RESET_GetCause(void)
66 {
67     return RCON;
68 }
69
70 void __attribute__ ((weak)) RESET_CauseHandler(void)
71 {
72     uint16_t resetCause = RESET_GetCause();
73     if(RESET_CauseFromTrap(resetCause))
74     {
75       RESET_CauseClear(RESET_MASK_TRAPR);
76       //Do something
77     }
78     if(RESET_CauseFromIllegalOpcode(resetCause))
79     {
80       RESET_CauseClear(RESET_MASK_IOPUWR);
81       //Do something
82     }
83     if(RESET_CauseFromConfigurationMismatch(resetCause))
84     {
85       RESET_CauseClear(RESET_MASK_CM);
86       //Do something
87     }
88     if(RESET_CauseFromExternal(resetCause))
89     {
90       RESET_CauseClear(RESET_MASK_EXTR);
91       //Do something
92     }
93     if(RESET_CauseFromSoftware(resetCause))
94     {
95       RESET_CauseClear(RESET_MASK_SWR);
96       //Do something
97     }
98     if(RESET_CauseFromWatchdogTimer(resetCause))
99     {
100      RESET_CauseClear(RESET_MASK_WDTO);
101      //Do something
102    }
103 }
104
105 static bool RESET_CauseFromTrap(uint16_t resetCause)
106 {
107     bool resetStatus = false;
```

```
108     if(resetCause & RESET_MASK_TRAPR)
109     {
110       resetStatus = true;
111     }
112     return resetStatus;
113 }
114 
115 static bool RESET_CauseFromIllegalOpcode(uint16_t resetCause)
116 {
117     bool resetStatus = false;
118     if(resetCause & RESET_MASK_IOPUWR)
119     {
120       resetStatus = true;
121     }
122     return resetStatus;
123 }
124 
125 static bool RESET_CauseFromConfigurationMismatch(uint16_t resetCause)
126 {
127     bool resetStatus = false;
128     if(resetCause & RESET_MASK_CM)
129     {
130       resetStatus = true;
131     }
132     return resetStatus;
133 }
134 
135 static bool RESET_CauseFromExternal(uint16_t resetCause)
136 {
137     bool resetStatus = false;
138     if(resetCause & RESET_MASK_EXTR)
139     {
140       resetStatus = true;
141     }
142     return resetStatus;
143 }
144 
145 static bool RESET_CauseFromSoftware(uint16_t resetCause)
146 {
147     bool resetStatus = false;
148     if(resetCause & RESET_MASK_SWR)
149     {
150       resetStatus = true;
151     }
152     return resetStatus;
153 }
154 
155 static bool RESET_CauseFromWatchdogTimer(uint16_t resetCause)
156 {
157     bool resetStatus = false;
158     if(resetCause & RESET_MASK_WDTO)
159     {
160       resetStatus = true;
161     }
162     return resetStatus;
163 }
164 
```

```
165 static void RESET_CauseClear(RESET_MASKS resetFlagMask)
166 {
167     RCON = RCON & (~resetFlagMask);
168 }
169
170 void RESET_CauseClearAll()
171 {
172     RCON = 0x00;
173 }
174 /**
175  End of File
176 */
```

system.h file

```
/**
  @Generated PIC24 / dsPIC33 / PIC32MM MCUs Source File

  @Company:
    Microchip Technology Inc.

  @File Name:
    system.h

  @Summary:
    This is the system.h file generated using PIC24 / dsPIC33 / PIC32MM MCUs

  @Description:
    This header file provides implementations for driver APIs for all modules selected in the GUI.
    Generation Information :
        Product Revision  :  PIC24 / dsPIC33 / PIC32MM MCUs - pic24-dspic-pic32mm : 1.75.1
        Device            :  dsPIC33EV256GM102
    The generated drivers are tested against the following:
        Compiler          :  XC16 v1.35
        MPLAB             :  MPLAB X v5.05
*/

/*
    (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this
    software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
    EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
    PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS, COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
    INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
    BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF THESE
    TERMS.
*/

#include "xc.h"
#include "stdint.h"
#include "system_types.h"

#ifndef SYSTEM_H
```

```
50 #define SYSTEM_H
51 /**
52  * Initializes the CPU core control register.
53  * @example
54  * <code>
55  * SYSTEM_CORCONInitialize();
56  * </code>
57  */
58 inline static void SYSTEM_CORCONInitialize()
59 {
60     CORCON = (CORCON & 0x00F2) | CORCON_MODE_PORVALUES;     // POR value
61 }
62
63 /**
64  * Sets the CPU core control register operating mode to a value that is
       decided by the
65  * SYSTEM_CORCON_MODES argument.
66  * @param modeValue SYSTEM_CORCON_MODES initialization mode specifier
67  * @example
68  * <code>
69  * SYSTEM_CORCONModeOperatingSet(CORCON_MODE_ENABLEALLSATNORMAL_ROUNDUNBIASED)
       ;
70  * </code>
71  */
72 inline static void SYSTEM_CORCONModeOperatingSet(SYSTEM_CORCON_MODES modeValue
       )
73 {
74     CORCON = (CORCON & 0x00F2) | modeValue;
75 }
76
77 /**
78  * Sets the value of CPU core control register.
79  * @param value value that needs to be written to the CPU core control
       register
80  * @example
81  * <code>
82  *  SYSTEM_CORCONRegisterValueSet(0x00E2);
83  * </code>
84  */
85 inline static void SYSTEM_CORCONRegisterValueSet(uint16_t value)
86 {
87     CORCON = value;
88 }
89
90 /**
91  * Gets the value of CPU core control register.
92  * @return value of the CPU core control register
93  * @example
94  * <code>
95  * corconSave = SYSTEM_CORCONRegisterValueGet();
96  * </code>
97  */
98 inline static uint16_t SYSTEM_CORCONRegisterValueGet(void)
99 {
100     return CORCON;
101 }
102
```

```
103
104 /**
105  * Gets the base address of the DEVID register for the currently selected
        device
106  * @return base address of the DEVID register
107  * @example
108  * <code>
109  * uint32_t devIdAddress;
110  * devIdAddress = SYSTEM_DeviceIdRegisterAddressGet();
111  * </code>
112  */
113 inline static uint32_t SYSTEM_DeviceIdRegisterAddressGet(void)
114 {
115     return __DEVID_BASE;
116 }
117
118 /**
119  * @Param
120     none
121  * @Returns
122     none
123  * @Description
124     Initializes the device to the default states configured in the
125  *                  MCC GUI
126  * @Example
127     SYSTEM_Initialize(void);
128  */
129 void SYSTEM_Initialize(void);
130 #endif  /* SYSTEM_H */
131 /**
132  End of File
133 */
```

system_types.h file

```
/**
  @Generated PIC24 / dsPIC33 / PIC32MM MCUs Source File

  @Company:
    Microchip Technology Inc.

  @File Name:
    system_types.h

  @Summary:
    This is the system_types.h file generated using PIC24 / dsPIC33 / PIC32MM
    MCUs

  @Description:
    This header file provides implementations for driver APIs for all modules
    selected in the GUI.
    Generation Information :
        Product Revision  :  PIC24 / dsPIC33 / PIC32MM MCUs - pic24-dspic-
    pic32mm : 1.75.1
        Device            :  dsPIC33EV256GM102
    The generated drivers are tested against the following:
        Compiler          :  XC16 v1.35
        MPLAB             :  MPLAB X v5.05
*/

/*
    (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this
    software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
    EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY
    IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
    PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS,
    COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
    INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
    BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS
    IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF
    ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF
    THESE
    TERMS.
*/

#ifndef SYSTEM_TYPES_H
#define SYSTEM_TYPES_H

/**
```

```
49   Section: Type defines
50  */
51
52 /**
53  * CORCON initialization type enumerator. Supported types:
54  * CORCON_MODE_PORVALUES
55  * CORCON_MODE_ENABLEALLSATNORMAL_ROUNDBIASED
56  * CORCON_MODE_ENABLEALLSATNORMAL_ROUNDUNBIASED
57  * CORCON_MODE_DISABLEALLSAT_ROUNDBIASED
58  * CORCON_MODE_DISABLEALLSAT_ROUNDUNBIASED
59  * CORCON_MODE_ENABLEALLSATSUPER_ROUNDBIASED
60  * CORCON_MODE_ENABLEALLSATSUPER_ROUNDUNBIASED
61  */
62 typedef enum tagCORCON_MODE_TYPE
63 {
64     CORCON_MODE_PORVALUES    = 0x0020,                    /** Use POR values
      of CORCON */
65     CORCON_MODE_ENABLEALLSATNORMAL_ROUNDBIASED = 0x00E2,  /** Enable
     saturation for ACCA, ACCB
66                                                           *  and Dataspace
     write, enable normal
67                                                           *  ACCA/ACCB
     saturation mode and set
68                                                           *  rounding to
     Biased (conventional)
69                                                           *  mode. Rest of
     CORCON settings are
70                                                           *  set to the
     default POR values.
71                                                           *  */
72     CORCON_MODE_ENABLEALLSATNORMAL_ROUNDUNBIASED = 0x00E0, /** Enable
     saturation for ACCA, ACCB
73                                                           *  and Dataspace
     write, enable normal
74                                                           *  ACCA/ACCB
     saturation mode and set
75                                                           *  rounding to
     Unbiased (convergent)
76                                                           *  mode. Rest of
     CORCON settings are
77                                                           *  set to the
     default POR values.
78                                                           *  */
79     CORCON_MODE_DISABLEALLSAT_ROUNDBIASED = 0x0022,       /** Disable
     saturation for ACCA, ACCB
80                                                           *  and Dataspace
     write and set
81                                                           *  rounding to
     Biased (conventional)
82                                                           *  mode. Rest of
     CORCON settings are
83                                                           *  set to the
     default POR values.
84                                                           *  */
85     CORCON_MODE_DISABLEALLSAT_ROUNDUNBIASED = 0x0020,     /** Disable
     saturation for ACCA, ACCB
86                                                           *  and Dataspace
```

```
87      write and set                                                   *  rounding to
        Unbiased (convergent)
88                                                                      *  mode. Rest of
        CORCON settings are
89                                                                      *  set to the
        default POR values.
90                                                                      *  */
91      CORCON_MODE_ENABLEALLSATSUPER_ROUNDBIASED = 0x00F2,     /** Enable
        saturation for ACCA, ACCB
92                                                                      *  and Dataspace
        write, enable super
93                                                                      *  ACCA/ACCB
        saturation mode and set
94                                                                      *  rounding to
        Biased (conventional)
95                                                                      *  mode. Rest of
        CORCON settings are
96                                                                      *  set to the
        default POR values.
97                                                                      *  */
98      CORCON_MODE_ENABLEALLSATSUPER_ROUNDUNBIASED = 0x00F0,   /** Enable
        saturation for ACCA, ACCB
99                                                                      *  and Dataspace
        write, enable super
100                                                                     *  ACCA/ACCB
        saturation mode and set
101                                                                     *  rounding to
        Unbiased (convergent)
102                                                                     *  mode. Rest of
        CORCON settings are
103                                                                     *  set to the
        default POR values.
104                                                                     *  */
105 } SYSTEM_CORCON_MODES;
106
107 #endif  /* SYSTEM_TYPES_H */
108 /**
109  End of File
110 */
```

system.c file

```
/**
  @Generated PIC24 / dsPIC33 / PIC32MM MCUs Source File

  @Company:
    Microchip Technology Inc.

  @File Name:
    system.h

  @Summary:
    This is the sysetm.h file generated using PIC24 / dsPIC33 / PIC32MM MCUs

  @Description:
    This header file provides implementations for driver APIs for all modules selected in the GUI.
    Generation Information :
        Product Revision  :  PIC24 / dsPIC33 / PIC32MM MCUs - pic24-dspic-pic32mm : 1.75.1
        Device            :  dsPIC33EV256GM102
    The generated drivers are tested against the following:
        Compiler          :  XC16 v1.35
        MPLAB             :  MPLAB X v5.05
*/

/*
    (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS, COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF THESE
    TERMS.
*/

#include "pin_manager.h"
#include "clock.h"
#include "system.h"
#include "stdint.h"
#include "system_types.h"
```

```
50 #include "ecan1.h"
51 #include "adc1.h"
52 #include "dma.h"
53 #include "interrupt_manager.h"
54 #include "traps.h"
55
56 void SYSTEM_Initialize(void)
57 {
58     PIN_MANAGER_Initialize();
59     CLOCK_Initialize();
60     INTERRUPT_Initialize();
61     ADC1_Initialize();
62     DMA_Initialize();
63     ECAN1_Initialize();
64     INTERRUPT_GlobalEnable();
65     SYSTEM_CORCONModeOperatingSet(CORCON_MODE_PORVALUES);
66 }
67
68 /**
69  End of File
70 */
```

traps.h file

```
/**
  System Traps Generated Driver File

  @Company:
    Microchip Technology Inc.

  @File Name:
    traps.h

  @Summary:
    This is the generated driver implementation file for handling traps
    using PIC24 / dsPIC33 / PIC32MM MCUs

  @Description:
    This source file provides implementations for PIC24 / dsPIC33 / PIC32MM
    MCUs traps.
    Generation Information :
        Product Revision  :  PIC24 / dsPIC33 / PIC32MM MCUs - pic24-dspic-
    pic32mm : 1.75.1
        Device            :  dsPIC33EV256GM102
    The generated drivers are tested against the following:
        Compiler          :  XC16 v1.35
        MPLAB             :  MPLAB X v5.05
*/
/*
    (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this
    software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
    EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY
    IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
    PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS,
    COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
    INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
    BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS
    IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF
    ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF
    THESE
    TERMS.
*/

#ifndef _TRAPS_H
#define _TRAPS_H

#include <stdint.h>

```

```
50 /**
51  * Error codes
52  */
53 typedef enum
54 {
55     /* ------ Traps ------ */
56     TRAPS_OSC_FAIL = 0, /** Oscillator Fail Trap vector */
57     TRAPS_STACK_ERR = 1, /** Stack Error Trap Vector */
58     TRAPS_ADDRESS_ERR = 2, /** Address error Trap vector */
59     TRAPS_MATH_ERR = 3, /** Math Error Trap vector */
60     TRAPS_DMAC_ERR = 4, /** DMAC Error Trap vector */
61     TRAPS_HARD_ERR = 7, /** Generic Hard Trap vector */
62     TRAPS_DOOVR_ERR = 10, /** Generic Soft Trap vector */
63 } TRAPS_ERROR_CODE;
64 /**
65   @Summary
66     Default handler for the traps
67
68   @Description
69     This routine will be called whenever a trap happens. It stores the trap
70     error code and waits forever.
71     This routine has a weak attribute and can be over written.
72
73   @Preconditions
74     None.
75
76   @Returns
77     None.
78
79   @Param
80     None.
81
82   @Example
83     None.
84
85 */
86 void __attribute__((naked, noreturn, weak)) TRAPS_halt_on_error(uint16_t code)
       ;
87
88 #endif
```

traps.c file

```
/**
  System Traps Generated Driver File

  @Company:
    Microchip Technology Inc.

  @File Name:
    traps.h

  @Summary:
    This is the generated driver implementation file for handling traps
    using PIC24 / dsPIC33 / PIC32MM MCUs

  @Description:
    This source file provides implementations for PIC24 / dsPIC33 / PIC32MM
    MCUs traps.
    Generation Information :
        Product Revision  :  PIC24 / dsPIC33 / PIC32MM MCUs - pic24-dspic-
    pic32mm : 1.75.1
        Device            :  dsPIC33EV256GM102
    The generated drivers are tested against the following:
        Compiler          :  XC16 v1.35
        MPLAB             :  MPLAB X v5.05
*/
/*
    (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this
    software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
    EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY
    IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
    PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS,
    COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
    INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
    BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS
    IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF
    ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF
    THESE
    TERMS.
*/

/**
    Section: Includes
*/
#include <xc.h>
#include "traps.h"
```

```
50
51 #define ERROR_HANDLER __attribute__((interrupt, no_auto_psv, keep, section("
       error_handler")))
52 #define ERROR_HANDLER_NORETURN ERROR_HANDLER __attribute__((noreturn))
53 #define FAILSAFE_STACK_GUARDSIZE 8
54
55 /**
56  * a private place to store the error code if we run into a severe error
57  */
58 static uint16_t TRAPS_error_code = -1;
59
60 /**
61  * Halts
62  *
63  * @param code error code
64  */
65 void __attribute__((naked, noreturn, weak)) TRAPS_halt_on_error(uint16_t code)
66 {
67     TRAPS_error_code = code;
68 #ifdef __DEBUG
69     __builtin_software_breakpoint();
70     /* If we are in debug mode, cause a software breakpoint in the debugger */
71 #endif
72     while(1);
73
74 }
75
76 /**
77  * Sets the stack pointer to a backup area of memory, in case we run into
78  * a stack error (in which case we can't really trust the stack pointer)
79  */
80 inline static void use_failsafe_stack(void)
81 {
82     static uint8_t failsafe_stack[32];
83     asm volatile (
84         "   mov    %[pstack], W15\n"
85         :
86         : [pstack]"r"(failsafe_stack)
87     );
88 /* Controls where the stack pointer limit is, relative to the end of the
89  * failsafe stack
90  */
91     SPLIM = (uint16_t)(((uint8_t *)failsafe_stack) + sizeof(failsafe_stack)
92             - FAILSAFE_STACK_GUARDSIZE);
93 }
94
95
96 /** Oscillator Fail Trap vector**/
97 void ERROR_HANDLER_NORETURN _OscillatorFail(void)
98 {
99     INTCON1bits.OSCFAIL = 0;  //Clear the trap flag
100     TRAPS_halt_on_error(TRAPS_OSC_FAIL);
101 }
102 /** Stack Error Trap Vector**/
103 void ERROR_HANDLER_NORETURN _StackError(void)
104 {
105     /* We use a failsafe stack: the presence of a stack-pointer error
```

```
106      * means that we cannot trust the stack to operate correctly unless
107      * we set the stack pointer to a safe place.
108      */
109     use_failsafe_stack();
110     INTCON1bits.STKERR = 0;  //Clear the trap flag
111     TRAPS_halt_on_error(TRAPS_STACK_ERR);
112 }
113 /** Address error Trap vector**/
114 void ERROR_HANDLER_NORETURN _AddressError(void)
115 {
116     INTCON1bits.ADDRERR = 0;  //Clear the trap flag
117     TRAPS_halt_on_error(TRAPS_ADDRESS_ERR);
118 }
119 /** Math Error Trap vector**/
120 void ERROR_HANDLER_NORETURN _MathError(void)
121 {
122     INTCON1bits.MATHERR = 0;  //Clear the trap flag
123     TRAPS_halt_on_error(TRAPS_MATH_ERR);
124 }
125 /** DMAC Error Trap vector**/
126 void ERROR_HANDLER_NORETURN _DMACError(void)
127 {
128     INTCON1bits.DMACERR = 0;  //Clear the trap flag
129     TRAPS_halt_on_error(TRAPS_DMAC_ERR);
130 }
131 /** Generic Hard Trap vector**/
132 void ERROR_HANDLER_NORETURN _HardTrapError(void)
133 {
134     INTCON4bits.SGHT = 0;  //Clear the trap flag
135     TRAPS_halt_on_error(TRAPS_HARD_ERR);
136 }
137 /** Generic Soft Trap vector**/
138 void ERROR_HANDLER_NORETURN _SoftTrapError(void)
139 {
140     INTCON3bits.DOOVR = 0;  //Clear the trap flag
141     TRAPS_halt_on_error(TRAPS_DOOVR_ERR);
142 }
```

watchdog.h file

```
/**
  WATCHDOG Generated Driver File

  @Company
    Microchip Technology Inc.

  @File Name
    watchdog.h

  @Summary
    This is the generated driver implementation file for the WATCHDOG driver
    using PIC24 / dsPIC33 / PIC32MM MCUs

  @Description
    This header file provides implementations for driver APIs for WATCHDOG.
    Generation Information :
        Product Revision  :  PIC24 / dsPIC33 / PIC32MM MCUs - 1.75.1
        Device            :  dsPIC33EV256GM102
    The generated drivers are tested against the following:
        Compiler          :  XC16 v1.35
        MPLAB             :  MPLAB X v5.05
*/

/*
    (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this
    software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
    EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY
    IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
    PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS,
    COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
    INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
    BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS
    IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF
    ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF
    THESE
    TERMS.
*/

#ifndef WATCHDOG_H
#define WATCHDOG_H

/**
  Section: Type defines
 */
```

```
51
52 /**
53  * Enables Watch Dog Timer (WDT) using the software bit.
54  * @example
55  * <code>
56  * WATCHDOG_TimerSoftwareEnable();
57  * </code>
58  */
59 inline static void WATCHDOG_TimerSoftwareEnable(void)
60 {
61     RCONbits.SWDTEN = 1;
62 }
63
64 /**
65  * Disables Watch Dog Timer (WDT) using the software bit.
66  * @example
67  * <code>
68  * WATCHDOG_TimerSoftwareDisable();
69  * </code>
70  */
71 inline static void WATCHDOG_TimerSoftwareDisable(void)
72 {
73     RCONbits.SWDTEN = 0;
74 }
75
76 /**
77  * Clears the Watch Dog Timer (WDT).
78  * @example
79  * <code>
80  * WATCHDOG_TimerClear();
81  * </code>
82  */
83 inline static void WATCHDOG_TimerClear(void)
84 {
85     ClrWdt();
86 }
87
88 #endif  /* WATCHDOG_H */
89 /**
90  End of File
91 */
```

7.4 BLDC Controller MCC Generated Files

adc1.h file

```
/**
  ADC1 Generated Driver API Header File

  @Company
    Microchip Technology Inc.

  @File Name
    adc1.h

  @Summary
    This is the generated header file for the ADC1 driver using PIC24 /
    dsPIC33 / PIC32MM MCUs

  @Description
    This header file provides APIs for driver for ADC1.
    Generation Information :
        Product Revision  :  PIC24 / dsPIC33 / PIC32MM MCUs - pic24-dspic-
    pic32mm : 1.75.1
        Device            :  dsPIC33EV256GM102
    The generated drivers are tested against the following:
        Compiler          :  XC16 v1.35
        MPLAB             :  MPLAB X v5.05
*/

/*
    (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this
    software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
    EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY
    IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
    PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS,
    COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
    INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
    BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS
    IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF
    ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF
    THESE
    TERMS.
*/

#ifndef _ADC1_H
#define _ADC1_H

```

```
48 /**
49   Section: Included Files
50 */
51
52 #include <stdbool.h>
53 #include <stdint.h>
54 #include <stdlib.h>
55
56 #ifdef __cplusplus  // Provide C++ Compatibility
57
58     extern "C" {
59
60 #endif
61 /**
62   Section: ISR Helper Macros
63 */
64
65
66 /**
67   Section: Data Types
68 */
69
70 /** ADC Channel Definition
71
72  @Summary
73    Defines the channels available for conversion
74
75  @Description
76    This routine defines the channels that are available conversion.
77
78  Remarks:
79    None
80  */
81 typedef enum
82 {
83     ADC1_CHANNEL_INTERNAL_BAND_GAP_REFERENCE =  0x3D,
84     ADC1_CHANNEL_CTMU =  0x3E,
85     ADC1_MAX_CHANNEL_COUNT = 2
86 } ADC1_CHANNEL;
87
88 /** ADC Positive 123 Channels Definition
89
90  @Summary
91    Defines the positive 123 channels available for conversion
92
93  @Description
94    This routine defines the positive 123 channels that are available for the
95    module to use.
96
97  Remarks:
98    None
99  */
100 typedef enum
101 {
102     ADC1_POS_123_CHANNEL_0 = 0x0,
103     ADC1_POS_123_CHANNEL_1 = 0x1,
104     ADC1_POS_123_CHANNEL_2 = 0x8,
```

```
105     ADC1_POS_123_CHANNEL_3 = 0x9,
106     ADC1_POS_123_CHANNEL_4 = 0x10
107 } ADC1_POS_123_CHANNEL;
108
109 /** ADC Negative 123 Channels Definition
110
111  @Summary
112    Defines the negative 123 channels available for conversion
113
114  @Description
115    This routine defines the negative 123 channels that are available for the
116    module to use.
117
118  Remarks:
119    None
120  */
121 typedef enum
122 {
123    ADC1_NEG_123_CHANNEL_0 = 0x0,
124    ADC1_NEG_123_CHANNEL_1 = 0x2,
125    ADC1_NEG_123_CHANNEL_2 = 0x3
126 } ADC1_NEG_123_CHANNEL;
127
128 /** ADC Data Format Type Definition
129
130  @Summary
131    Defines the data format types available
132
133  @Description
134    This routine defines the data format types that are available for the
       module
135    to use.
136
137  Remarks:
138    None
139  */
140 typedef enum
141 {
142     ADC1_FORM_UNSIGNED_INT    = 0, /* Unsigned Integer */
143     ADC1_FORM_SIGNED_INT      = 1, /* Signed Integer */
144     ADC1_FORM_UNSIGNED_FRACT = 2, /* Unsigned Fraction */
145     ADC1_FORM_SIGNED_FRACT    = 3  /* Signed Integer */
146 } ADC1_FORM_TYPE;
147
148 /** ADC Resolution Type Definition
149
150  @Summary
151    Defines the resolution types available
152
153  @Description
154    This routine defines the resolution types that are available for the module
155    to use.
156
157  Remarks:
158    None
159  */
160 typedef enum
```

```
{
    ADC1_RESOLUTION_10_BIT    = 0, /* 10-bit, 4-channel ADC operation */
    ADC1_RESOLUTION_12_BIT    = 1  /* 12-bit, 1-channel ADC operation */
} ADC1_RESOLUTION_TYPE;

/** ADC Sampling Source Definition

 @Summary
   Defines the sampling sources available

 @Description
   This routine defines the sampling sources that are available for the module
   to use.

 Remarks:
   None
 */
typedef enum
{
    ADC1_SAMPLING_SOURCE_CTMU   =  0x6,
    ADC1_SAMPLING_SOURCE_MANUAL  =  0x0,
    ADC1_SAMPLING_SOURCE_TMR5   =  0x4,
    ADC1_SAMPLING_SOURCE_TMR3   =  0x2,
    ADC1_SAMPLING_SOURCE_INT0   =  0x1,
    ADC1_SAMPLING_SOURCE_PWM2   =  0x1,
    ADC1_SAMPLING_SOURCE_PWM3   =  0x2,
    ADC1_SAMPLING_SOURCE_AUTO   =  0x7,
    ADC1_SAMPLING_SOURCE_PWM   =  0x3,
    ADC1_SAMPLING_SOURCE_PWM1   =  0x0,
} ADC1_SAMPLING_SOURCE;

/** ADC Conversion Channel Type Definition

 @Summary
   Defines the conversion channel types available

 @Description
   This routine defines the conversion channels types that are available for
    the
   module to use.

 Remarks:
   None
 */
typedef enum
{
    ADC1_CONVERSION_CHANNELS_CH0     = 0, /* Converts only CH0 */
    ADC1_CONVERSION_CHANNELS_CH01    = 1, /* Converts CH0 and CH1 */
    ADC1_CONVERSION_CHANNELS_CH0123 = 2  /* Converts CH0, CH1, CH2 and CH3 */
} ADC1_CONVERSION_CHANNELS_TYPE;

/**
  Section: Interface Routines
*/


/**
```

```
217   @Summary
218     This function initializes ADC instance : 1
219
220   @Description
221     This routine initializes the ADC driver instance for : 1
222     index, making it ready for clients to open and use it. It also initializes
         any
223     internal data structures.
224     This routine must be called before any other ADC routine is called.
225
226   @Preconditions
227     None.
228
229   @Param
230     None.
231
232   @Returns
233     None.
234
235   @Comment
236
237
238   @Example
239     <code>
240         int conversion;
241         ADC1_Initialize();
242         ADC1_ChannelSelect(AN1_Channel);
243         ADC1_SamplingStart();
244         //Provide Delay
245         for(int i=0;i <1000;i++)
246         {
247         }
248         ADC1_SamplingStop();
249         while(!ADC1_IsConversionComplete())
250         {
251             ADC1_Tasks();
252         }
253         conversion = ADC1_ConversionResultGet();
254     </code>
255
256 */
257
258 void ADC1_Initialize (void);
259
260 /**
261   @Summary
262     Clears interrupt flag
263
264   @Description
265     This routine is used to clear the interrupt flag manually.
266
267   @Preconditions
268     None.
269
270   @Param
271     None.
272
```

```
273   @Returns
274     None.
275
276   @Example
277     Refer to ADC1_Initialize() for an example
278
279 */
280
281 inline static void ADC1_InterruptFlagClear(void)
282 {
283    IFS0bits.AD1IF = 0;
284 }
285 /**
286   @Summary
287     Enables interrupts.
288
289   @Description
290     This routine is used to enable the ADC1 interrupt manually.
291
292   @Preconditions
293     None.
294
295   @Param
296     None.
297
298   @Returns
299     None.
300
301   @Example
302     Refer to ADC1_Initialize() for an example
303
304 */
305 inline static void ADC1_InterruptEnable(void)
306 {
307    IEC0bits.AD1IE = 1;
308 }
309 /**
310   @Summary
311     Disables interrupts
312
313   @Description
314     This routine is used to disable the ADC1 interrupt manually.
315
316   @Preconditions
317     None.
318
319   @Param
320     None.
321
322   @Returns
323     None.
324
325   @Example
326     Refer to ADC1_Initialize() for an example
327
328 */
329
```

```
330 inline static void ADC1_InterruptDisable(void)
331 {
332    IEC0bits.AD1IE = 0;
333 }
334 /**
335   @Summary
336     Starts sampling manually.
337
338   @Description
339     This routine is used to start the sampling manually.
340
341   @Preconditions
342     ADC1_Initialize() function should have been called
343     before calling this function.
344
345   @Param
346     None.
347
348   @Returns
349     None.
350
351   @Example
352     <code>
353         int conversion;
354         ADC1__Initialize();
355         ADC1_ChannelSelect(AN1_Channel);
356         ADC1_SamplingStart();
357         //Provide Delay
358         for(int i=0;i <1000;i++)
359         {
360         }
361         ADC1_SamplingStop();
362         while(!ADC1_IsConversionComplete())
363         {
364             ADC1_Tasks();
365         }
366         conversion = ADC1_ConversionResultGet();
367     </code>
368
369 */
370
371 inline static void ADC1_SamplingStart(void)
372 {
373    AD1CON1bits.SAMP = 1;
374 }
375 /**
376   @Summary
377     Stops sampling manually.
378
379   @Description
380     This routine is used to stop the sampling manually before conversion
381     is triggered.
382
383   @Preconditions
384     ADC1_Initialize() function should have been
385     called before calling this function.
386
```

```
387   @Param
388     None.
389
390   @Returns
391     None.
392
393   @Example
394     <code>
395         int conversion;
396         ADC1__Initialize();
397         ADC1_ChannelSelect(AN1_Channel);
398         ADC1_SamplingStart();
399         //Provide Delay
400         for(int i=0;i <1000;i++)
401         {
402         }
403         ADC1_SamplingStop();
404         while(!ADC1_IsConversionComplete())
405         {
406             ADC1_Tasks();
407         }
408         conversion = ADC1_ConversionResultGet();
409     </code>
410 */
411
412 inline static void ADC1_SamplingStop(void)
413 {
414    AD1CON1bits.SAMP = 0;
415 }
416 /**
417   @Summary
418     Gets the buffer loaded with conversion results.
419
420   @Description
421     This routine is used to get the analog to digital converted values in a
422     buffer. This routine gets converted values from multiple channels.
423
424   @Preconditions
425     This routine returns the buffer containing the conversion values only
        after
426     the conversion is complete. Completion status conversion can be checked
        using
427     ADC1_IsConversionComplete() routine.
428
429   @Param
430     None.
431
432   @Returns
433     Returns the count of the buffer containing the conversion values.
434
435   @Example
436     <code>
437         int count;
438         //Initialize for channel scanning
439         ADC1_Initialize();
440         ADC1_SamplingStart();
441         //Provide Delay
```

```
442         for(int i=0;i <1000;i++)
443         {
444         }
445         ADC1_SamplingStop();
446         while(!ADC1_IsConversionComplete())
447         {
448             count = ADC1_ConversionResultBufferGet();
449         }
450     </code>
451 */
452
453 uint16_t ADC1_ConversionResultBufferGet(uint16_t *buffer);
454
455 /**
456   @Summary
457     Returns the ADC1 conversion value for Channel 0.
458
459   @Description
460     This routine is used to get the analog to digital converted value. This
461     routine gets converted values from the channel specified.
462
463   @Preconditions
464     The channel required must be selected before calling this routine using
465     ADC1_ChannelSelect(channel). This routine returns the
466     conversion value only after the conversion is complete. Completion status
467     conversion can be checked using ADC1_IsConversionComplete()
468     routine.
469
470   @Returns
471     Returns the buffer containing the conversion value.
472
473   @Param
474     Buffer address
475
476   @Example
477     Refer to ADC1_Initialize(); for an example
478  */
479
480 inline static uint16_t ADC1_Channel0ConversionResultGet(void)
481 {
482     return ADC1BUF0;
483 }
484 /**
485   @Summary
486     Returns the ADC1 conversion value from Channel 1.
487
488   @Description
489     This routine is used to get the analog to digital converted value. This
490     routine gets converted values from the channel specified.
491
492   @Preconditions
493     The channel required must be selected before calling this routine using
494     ADC1_ChannelSelect(channel). This routine returns the
495     conversion value only after the conversion is complete. Completion status
496     conversion can be checked using ADC1_IsConversionComplete()
497     routine.
498
```

```
499   @Returns
500     Returns the buffer containing the conversion value.
501
502   @Param
503     Buffer address
504
505   @Example
506     Refer to ADC1_Initialize(); for an example
507  */
508
509 inline static uint16_t ADC1_Channel1ConversionResultGet(void)
510 {
511     return ADC1BUF1;
512 }
513 /**
514   @Summary
515     Returns the ADC1 conversion value from Channel 2.
516
517   @Description
518     This routine is used to get the analog to digital converted value. This
519     routine gets converted values from the channel specified.
520
521   @Preconditions
522     The channel required must be selected before calling this routine using
523     ADC1_ChannelSelect(channel). This routine returns the
524     conversion value only after the conversion is complete. Completion status
525     conversion can be checked using ADC1_IsConversionComplete()
526     routine.
527
528   @Returns
529     Returns the buffer containing the conversion value.
530
531   @Param
532     Buffer address
533
534   @Example
535     Refer to ADC1_Initialize(); for an example
536  */
537
538 inline static uint16_t ADC1_Channel2ConversionResultGet(void)
539 {
540     return ADC1BUF2;
541 }
542
543 /**
544   @Summary
545     Returns the ADC1 conversion value from Channel 3.
546
547   @Description
548     This routine is used to get the analog to digital converted value. This
549     routine gets converted values from the channel specified.
550
551   @Preconditions
552     The channel required must be selected before calling this routine using
553     ADC1_ChannelSelect(channel). This routine returns the
554     conversion value only after the conversion is complete. Completion status
555     conversion can be checked using ADC1_IsConversionComplete()
```

```
556     routine.
557
558   @Returns
559     Returns the buffer containing the conversion value.
560
561   @Param
562     Buffer address
563
564   @Example
565     Refer to ADC1_Initialize(); for an example
566  */
567
568 inline static uint16_t ADC1_Channel3ConversionResultGet(void)
569 {
570     return ADC1BUF3;
571 }
572 /**
573   @Summary
574     Returns true when the conversion is completed
575
576   @Description
577     This routine is used to determine if conversion is completed. This routine
578     returns the value of the DONE bit. When conversion is complete the routine
579     returns 1. It returns 0 otherwise.
580
581   @Preconditions
582     ADC1_Initialize() function should have been
583     called before calling this function.
584
585   @Returns
586     Returns true if conversion is completed
587
588   @Param
589     None
590
591   @Example
592     Refer to ADC1_Initialize(); for an example
593  */
594
595 inline static bool ADC1_IsConversionComplete( void )
596 {
597     return AD1CON1bits.DONE; //Wait for conversion to complete
598 }
599
600 /**
601   @Summary
602     Allows selection of a channel for conversion
603
604   @Description
605     This routine is used to select desired channel for conversion.
606
607   @Preconditions
608     ADC1_Initialize() function should have been
609     called before calling this function.
610
611   @Returns
612     None
```

```
613
614   @Param
615     Pass in required channel from the ADC1_CHANNEL list
616
617   @Example
618     Refer to ADC1_Initialize(); for an example
619
620 */
621
622 inline static void ADC1_ChannelSelectSet( ADC1_CHANNEL channel )
623 {
624     AD1CHS0bits.CH0SA = channel;
625 }
626 /**
627   @Summary
628     Returns the channel selected for conversion
629
630   @Description
631     This routine is used to return the channel selected for conversion.
632
633   @Preconditions
634     ADC1_Initialize() function should have been
635     called before calling this function.
636
637   @Returns
638     The value of the Channel Conversion register
639
640   @Param
641     None
642
643   @Example
644     Refer to ADC1_Initialize(); for an example
645
646 */
647
648 inline static uint16_t ADC1_ChannelSelectGet( void )
649 {
650     return AD1CHS0bits.CH0SA ;
651 }
652 /**
653   @Summary
654     Allows selection of a data format type for conversion
655
656   @Description
657     This routine is used to select desired data format for conversion.
658
659   @Preconditions
660     ADC1_Initialize() function should have been
661     called before calling this function.
662
663   @Returns
664     None
665
666   @Param
667     Pass in required data format type from the ADC1_FORM_TYPE list
668
669   @Example
```

```
670     Refer to ADC1_Initialize(); for an example
671 */
672
673 inline static void ADC1_FormatDataSet( ADC1_FORM_TYPE form )
674 {
675     AD1CON1bits.FORM = form;
676 }
677 /**
678   @Summary
679     Allows selection of a resolution mode for conversion
680
681   @Description
682     This routine is used to select desired resolution mode for conversion.
683
684   @Preconditions
685     ADC1_Initialize() function should have been
686     called before calling this function.
687
688   @Returns
689     None
690
691   @Param
692     Pass in required resolution mode from the ADC1_RESOLUTION_TYPE list
693
694   @Example
695     Refer to ADC1_Initialize(); for an example
696 */
697
698 inline static void ADC1_ResolutionModeSet( ADC1_RESOLUTION_TYPE resolution )
699 {
700     AD1CON1bits.AD12B = resolution;
701 }
702 /**
703   @Summary
704     Allows simultaneous sampling to be enabled manually
705
706   @Description
707     This routine is used to enable simultaneous sampling of channels manually
708
709   @Preconditions
710     ADC1_Initialize() function should have been
711     called before calling this function.
712
713   @Returns
714     None
715
716   @Param
717     None.
718
719   @Example
720     Refer to ADC1_Initialize(); for an example
721
722 */
723
724 inline static void ADC1_SimultaneousSamplingEnable(void)
725 {
726     AD1CON1bits.SIMSAM = 1;
```

```
727 }
728 /**
729   @Summary
730     Allows simultaneous sampling to be disabled manually
731
732   @Description
733     This routine is used to disable simultaneous sampling of channels manually
734
735   @Preconditions
736     ADC1_Initialize() function should have been
737     called before calling this function.
738
739   @Returns
740     None
741
742   @Param
743     None.
744
745   @Example
746     Refer to ADC1_Initialize(); for an example
747 */
748
749 inline static void ADC1_SimultaneousSamplingDisble(void)
750 {
751     AD1CON1bits.SIMSAM = 0;
752 }
753 /**
754 @Summary
755     Allows sutomatic sampling to be enabled manually
756
757   @Description
758     This routine is used to enable automatic sampling of channels manually
759
760   @Preconditions
761     ADC1_Initialize() function should have been
762     called before calling this function.
763
764   @Returns
765     None
766
767   @Param
768     None.
769
770   @Example
771     Refer to ADC1_Initialize(); for an example
772 */
773
774 inline static void ADC1_AutomaticSamplingEnable(void)
775 {
776     AD1CON1bits.ASAM = 1;
777 }
778 /**
779   @Summary
780     Allows automatic sampling to be disabled manually
781
782   @Description
783     This routine is used to disable automatic sampling of channels manually
```

```
784
785   @Preconditions
786     ADC1_Initialize() function should have been
787     called before calling this function.
788
789   @Returns
790     None
791
792   @Param
793     None.
794
795   @Example
796     Refer to ADC1_Initialize(); for an example
797 */
798
799 inline static void ADC1_AutomaticSamplingDisable(void)
800 {
801     AD1CON1bits.ASAM = 0;
802 }
803 /**
804   @Summary
805     Allows conversion clock prescaler value to be set
806
807   @Description
808     This routine is used to allow conversion clock prescaler value to be set
      manually
809
810   @Preconditions
811     ADC1_Initialize() function should have been
812     called before calling this function.
813
814   @Returns
815     None
816
817   @Param
818     Pass in required prescaler integer value
819
820   @Example
821     Refer to ADC1_Initialize(); for an example
822
823 */
824
825 inline static void ADC1_ConversionClockPrescalerSet(uint8_t prescaler)
826 {
827     AD1CON3bits.ADCS = prescaler - 1;
828 }
829 /**
830   @Summary
831     Allows module to be enabled manually
832
833   @Description
834     This routine is used to enable the ADC1 module manually
835
836   @Preconditions
837     ADC1_Initialize() function should have been
838     called before calling this function.
839
```

```
840   @Returns
841     None
842
843   @Param
844     None
845
846   @Example
847 */
848
849 inline static void ADC1_Enable(void)
850 {
851     AD1CON1bits.ADON = 1;
852 }
853 /**
854   @Summary
855     Allows module to be disabled manually
856
857   @Description
858     This routine is used to disable the ADC1 module manually
859
860   @Preconditions
861     ADC1_Initialize() function should have been
862     called before calling this function.
863
864   @Returns
865     None
866
867   @Param
868     None
869
870   @Example
871 */
872
873 inline static void ADC1_Disable(void)
874 {
875     AD1CON1bits.ADON = 0;
876 }
877
878 /**
879   @Summary
880     Allows selection of a positive 123 channel for conversion
881
882   @Description
883     This routine is used to select desired positive 123 channel for conversion
      .
884
885   @Preconditions
886     ADC1_Initialize() function should have been
887     called before calling this function.
888
889   @Returns
890     None
891
892   @Param
893     Pass in required channel from the ADC1_POS_123_CHANNEL list
894
895   @Example
```

```
896     Refer to ADC1_Initialize(); for an example
897
898 */
899
900 inline static void ADC1_Positive123ChannelSelect( ADC1_POS_123_CHANNEL channel
        )
901 {
902     AD1CHS123 = (AD1CHS123 & 0xFF06) | channel;
903 }
904 /**
905   @Summary
906     Allows selection of a negative 123 channel for conversion
907
908   @Description
909     This routine is used to select desired negative 123 channel for conversion
        .
910
911   @Preconditions
912     ADC1_Initialize() function should have been
913     called before calling this function.
914
915   @Returns
916     None
917
918   @Param
919     Pass in required channel from the ADC1_NEG_123_CHANNEL list
920
921   @Example
922     Refer to ADC1_Initialize(); for an example
923
924 */
925
926 inline static void ADC1_Negative123ChannelSelect( ADC1_NEG_123_CHANNEL channel
        )
927 {
928     AD1CHS123bits.CH123NA = channel;
929 }
930 /**
931   @Summary
932     Allows selection of conversion channels
933
934   @Description
935     This routine is used to select conversion channel for conversion.
936
937   @Preconditions
938     ADC1_Initialize() function should have been
939     called before calling this function.
940
941   @Returns
942     None
943
944   @Param
945     Pass in required channel from the ADC1_CONVERSION_CHANNELS_TYPE list
946
947   @Example
948     Refer to ADC1_Initialize(); for an example
949
```

```
950 */
951
952 inline static void ADC1_ConversionChannelsSet( ADC1_CONVERSION_CHANNELS_TYPE
        channel )
953 {
954     AD1CON2bits.CHPS = channel;
955 }
956 /**
957   @Summary
958     Allows selection of a priority for interrupt
959
960   @Description
961     This routine is used to select desired priority for interrupt.
962
963   @Preconditions
964     ADC1_Initialize() function should have been
965     called before calling this function.
966
967   @Returns
968     None
969
970   @Param
971     Pass in required integer priority value
972
973   @Example
974     Refer to ADC1_Initialize(); for an example
975
976 */
977
978 inline static void ADC1_InterruptPrioritySet( uint16_t priorityValue )
979 {
980     _AD1IP = 0x7 & priorityValue;
981 }
982 /**
983   @Summary
984     Polled implementation
985
986   @Description
987     This routine is used to implement the tasks for polled implementations.
988
989   @Preconditions
990     ADC1_Initialize() function should have been
991     called before calling this function.
992
993   @Returns
994     None
995
996   @Param
997     None
998
999   @Example
1000    Refer to ADC1_Initialize(); for an example
1001
1002 */
1003 void ADC1_Tasks(void);
1004
1005
```

```
1006 #ifdef __cplusplus  // Provide C++ Compatibility
1007
1008     }
1009
1010 #endif
1011
1012 #endif // _ADC1_H
1013
1014 /**
1015  End of File
1016 */
```

adc1.c file

```
1
2  /**
3    ADC1 Generated Driver File
4
5    @Company
6      Microchip Technology Inc.
7
8    @File Name
9      adc1.c
10
11   @Summary
12     This is the generated header file for the ADC1 driver using PIC24 /
       dsPIC33 / PIC32MM MCUs
13
14   @Description
15     This header file provides APIs for driver for ADC1.
16     Generation Information :
17         Product Revision  :  PIC24 / dsPIC33 / PIC32MM MCUs - pic24-dspic-
       pic32mm : 1.75.1
18         Device            :  dsPIC33EV256GM102
19     The generated drivers are tested against the following:
20         Compiler          :  XC16 v1.35
21         MPLAB             :  MPLAB X v5.05
22 */
23
24 /*
25     (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this
26     software and any derivatives exclusively with Microchip products.
27
28     THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
29     EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY
       IMPLIED
30     WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
31     PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS,
       COMBINATION
32     WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.
33
34     IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
35     INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
36     WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
37     BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
38     FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS
       IN
39     ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF
       ANY,
40     THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.
41
42     MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF
       THESE
43     TERMS.
44 */
45
46 /**
47   Section: Included Files
48 */
49
```

```
50 #include <xc.h>
51 #include "adc1.h"
52
53 /**
54   Section: Data Type Definitions
55 */
56
57 /* ADC Driver Hardware Instance Object
58
59   @Summary
60     Defines the object required for the maintenance of the hardware instance.
61
62   @Description
63     This defines the object required for the maintenance of the hardware
64     instance. This object exists once per hardware instance of the peripheral.
65
66  */
67 typedef struct
68 {
69   uint8_t intSample;
70 }
71
72 ADC_OBJECT;
73
74 static ADC_OBJECT adc1_obj;
75
76 /**
77   Section: Driver Interface
78 */
79
80
81 void ADC1_Initialize (void)
82 {
83     // ASAM enabled; ADDMABM disabled; ADSIDL disabled; DONE disabled; SIMSAM
       Sequential; FORM Absolute decimal result, unsigned, right-justified; SAMP
        disabled; SSRC Internal counter ends sampling and starts conversion;
       AD12B 12-bit; ADON enabled; SSRCG disabled;
84
85     AD1CON1 = 0x84E4;
86
87     // CSCNA disabled; VCFG0 AVDD; VCFG1 AVSS; ALTS disabled; BUFM disabled;
       SMPI Generates interrupt after completion of every sample/conversion
       operation; CHPS 1 Channel;
88
89     AD1CON2 = 0x0;
90
91     // SAMC 0; ADRC FOSC/2; ADCS 9;
92
93     AD1CON3 = 0x9;
94
95     // CH0SA OA2/AN0; CH0SB OA2/AN0; CH0NB VREFL; CH0NA VREFL;
96
97     AD1CHS0 = 0x0;
98
99     // CSS26 disabled; CSS25 disabled; CSS24 disabled; CSS27 disabled;
100
101    AD1CSSH = 0x0;
```

```
102
103     // CSS2 disabled; CSS1 disabled; CSS0 disabled; CSS5 disabled; CSS4
        disabled; CSS3 disabled;
104
105    AD1CSSL = 0x0;
106
107     // DMABL Allocates 1 word of buffer to each analog input; ADDMAEN disabled
        ;
108
109    AD1CON4 = 0x0;
110
111     // CH123SA2 disabled; CH123SB2 CH1=OA2/AN0,CH2=AN1,CH3=AN2; CH123NA
        disabled; CH123NB CH1=VREF-,CH2=VREF-,CH3=VREF-;
112
113    AD1CHS123 = 0x0;
114
115
116    adc1_obj.intSample = AD1CON2bits.SMPI;
117
118 }
119
120
121
122 void ADC1_Tasks ( void )
123 {
124     // clear the ADC interrupt flag
125     IFS0bits.AD1IF = false;
126 }
127
128
129
130 /**
131   End of File
132 */
```

clock.h file

```
/**
  @Generated PIC24 / dsPIC33 / PIC32MM MCUs Source File

  @Company:
    Microchip Technology Inc.

  @File Name:
    clock.h

  @Summary:
    This is the clock.h file generated using PIC24 / dsPIC33 / PIC32MM MCUs

  @Description:
    This header file provides implementations for driver APIs for all modules selected in the GUI.
    Generation Information :
        Product Revision  :  PIC24 / dsPIC33 / PIC32MM MCUs - pic24-dspic-pic32mm : 1.75.1
        Device            :  dsPIC33EV256GM102
    The generated drivers are tested against the following:
        Compiler          :  XC16 v1.35
        MPLAB             :  MPLAB X v5.05
*/

/*
    (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this
    software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
    EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
    PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS, COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
    INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
    BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF THESE
    TERMS.
*/

#ifndef CLOCK_H
#define CLOCK_H

#define _XTAL_FREQ  138187500UL
/**
```

```
50  * @Param
51      none
52  * @Returns
53      none
54  * @Description
55      Initializes the oscillator to the default states configured in the
56  *                   MCC GUI
57  * @Example
58      CLOCK_Initialize(void);
59  */
60 void CLOCK_Initialize(void);
61 #endif  /* CLOCK_H */
62 /**
63  End of File
64 */
```

clock.c file

```
/**
  @Generated PIC24 / dsPIC33 / PIC32MM MCUs Source File

  @Company:
    Microchip Technology Inc.

  @File Name:
    clock.c

  @Summary:
    This is the clock.c file generated using PIC24 / dsPIC33 / PIC32MM MCUs

  @Description:
    This header file provides implementations for driver APIs for all modules selected in the GUI.
    Generation Information :
        Product Revision  :  PIC24 / dsPIC33 / PIC32MM MCUs - pic24-dspic-pic32mm : 1.75.1
        Device            :  dsPIC33EV256GM102
    The generated drivers are tested against the following:
        Compiler          :  XC16 v1.35
        MPLAB             :  MPLAB X v5.05
*/

/*
    (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this
    software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
    EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
    PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS, COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
    INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
    BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF THESE
    TERMS.
*/

#include "clock.h"
#include "stdint.h"
#include "xc.h"

void CLOCK_Initialize(void)
```

```
50 {
51     // FRCDIV FRC/1; PLLPRE 2; DOZE 1:8; PLLPOST 1:2; DOZEN disabled; ROI
       disabled;
52     CLKDIV = 0x3000;
53     // TUN Center frequency;
54     OSCTUN = 0x0;
55     // ROON disabled; ROSEL disabled; RODIV Base clock value; ROSSLP disabled;
56     REFOCON = 0x0;
57     // PLLDIV 73;
58     PLLFBD = 0x49;
59     // RND disabled; SATB disabled; SATA disabled; ACCSAT disabled;
60   CORCONbits.RND = 0;
61   CORCONbits.SATB = 0;
62   CORCONbits.SATA = 0;
63   CORCONbits.ACCSAT = 0;
64     // CF no clock failure; NOSC FRCPLL; CLKLOCK unlocked; OSWEN Switch is
       Complete;
65     __builtin_write_OSCCONH((uint8_t) ((0x100 >> _OSCCON_NOSC_POSITION) & 0
       x00FF));
66     __builtin_write_OSCCONL((uint8_t) ((0x100 | _OSCCON_OSWEN_MASK) & 0xFF));
67     // Wait for Clock switch to occur
68     while (OSCCONbits.OSWEN != 0);
69     while (OSCCONbits.LOCK != 1);
70 }
```

dma.h file

```
/*
    ***************************************************************************

  DMA Generated Driver API Header File

  Company:
    Microchip Technology Inc.

  File Name:
    dma.h

  Summary:
    This is the generated header file for the DMA driver using PIC24 / dsPIC33
     / PIC32MM MCUs

  Description:
    This header file provides APIs for driver for DMA.
    Generation Information :
        Product Revision  :  PIC24 / dsPIC33 / PIC32MM MCUs - pic24-dspic-
    pic32mm : 1.75.1
        Device            :  dsPIC33EV256GM102
    The generated drivers are tested against the following:
        Compiler          :  XC16 v1.35
        MPLAB             :  MPLAB X v5.05

    ***************************************************************************
    */

/*
    (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this
    software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
    EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY
    IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
    PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS,
    COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
    INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
    BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS
    IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF
    ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF
    THESE
    TERMS.
*/

#ifndef DMA_H
```

```
46 #define DMA_H
47
48 #include <xc.h>
49 #include <stdbool.h>
50 #include <stdint.h>
51 #include <stdlib.h>
52
53 #ifdef __cplusplus  // Provide C++ Compatibility
54
55     extern "C" {
56
57 #endif
58
59 /**
60   Section: Data Types
61 */
62
63 /** DMA Channel Definition
64
65  @Summary
66    Defines the channels available for DMA
67
68  @Description
69    This routine defines the channels that are available for the module to use.
70
71  Remarks:
72    None
73  */
74 typedef enum
75 {
76     DMA_CHANNEL_0 =  0,
77     DMA_CHANNEL_1 =  1,
78     DMA_CHANNEL_2 =  2,
79     DMA_CHANNEL_3 =  3,
80     DMA_NUMBER_OF_CHANNELS = 4
81 } DMA_CHANNEL;
82 /**
83   Section: Interface Routines
84 */
85
86 /**
87   @Summary
88     This function initializes DMA instance : 1
89
90   @Description
91     This routine initializes the DMA driver instance for : 1
92     index, making it ready for clients to open and use it. It also initializes
       any
93     internal data structures.
94     This routine must be called before any other DMA routine is called.
95
96   @Preconditions
97     None.
98
99   @Param
100    None.
101
```

```
102   @Returns
103     None.
104
105   @Comment
106
107
108   @Example
109     <code>
110         unsigned short int srcArray[100];
111         unsigned short int dstArray[100];
112         int i;
113         int count;
114         for (i=0; i<100; i++)
115         {
116             srcArray[i] = i+1;
117             dstArray[i] = 0;
118         }
119
120         DMA_Initialize();
121         DMA_SoftwareTriggerEnable(CHANNEL1);
122
123         count = DMA_TransferCountGet;
124         while(count > 0)
125         {
126         while(
127         DMA_SoftwareTriggerEnable(CHANNEL1);
128         }
129     </code>
130
131 */
132 void DMA_Initialize(void);
133
134 /**
135   @Summary
136     Clears the interrupt request flag.
137
138   @Description
139     This routine is used to clear the interrupt request flag. This routine
      sets the value of the DMAIF bit to 0.
140
141   @Preconditions
142     DMA_Initializer() function should have been
143     called before calling this function.
144
145   @Returns
146     None
147
148   @Param
149     None
150
151   @Example
152     Refer to DMA_Initializer(); for an example
153  */
154 inline static void DMA_FlagInterruptClear(DMA_CHANNEL  channel)
155 {
156     switch(channel) {
157         case DMA_CHANNEL_0:
```

```
158                     IFS0bits.DMA0IF = 0;
159                     break;
160             case DMA_CHANNEL_1:
161                     IFS0bits.DMA1IF = 0;
162                     break;
163             case DMA_CHANNEL_2:
164                     IFS1bits.DMA2IF = 0;
165                     break;
166             case DMA_CHANNEL_3:
167                     IFS2bits.DMA3IF = 0;
168                     break;
169             default:break;
170
171     }
172 }
173
174 /**
175 @Summary
176     Enables the interrupt for a DMA channel
177
178   @Description
179     This routine is used to enable an interrupt for a DMA channel. This
        routine
180     sets the value of the DMAIE bit to 1.
181
182   @Preconditions
183     DMA_Initializer() function should have been
184     called before calling this function.
185
186   @Returns
187     None
188
189   @Param
190     None
191
192   @Example
193     Refer to DMA_Initializer(); for an example
194  */
195 inline static void DMA_InterruptEnable(DMA_CHANNEL  channel)
196 {
197     switch(channel) {
198         case DMA_CHANNEL_0:
199                 IEC0bits.DMA0IE = 1;
200                 break;
201         case DMA_CHANNEL_1:
202                 IEC0bits.DMA1IE = 1;
203                 break;
204         case DMA_CHANNEL_2:
205                 IEC1bits.DMA2IE = 1;
206                 break;
207         case DMA_CHANNEL_3:
208                 IEC2bits.DMA3IE = 1;
209                 break;
210         default:break;
211
212     }
213 }
```

```
/**
@Summary
 Disables the interrupt for a DMA channel

@Description
 This routine is used to disable an interrupt for a DMA channel. This routine
 sets the value of the DMAIE bit to 0.

@Preconditions
  DMA_Initializer() function should have been
  called before calling this function.

@Returns
 None

@Param
 None

@Example
 Refer to DMA_Initializer(); for an example
*/
inline static void DMA_InterruptDisable(DMA_CHANNEL  channel)
{
    switch(channel) {
        case DMA_CHANNEL_0:
                IEC0bits.DMA0IE = 0;
                break;
        case DMA_CHANNEL_1:
                IEC0bits.DMA1IE = 0;
                break;
        case DMA_CHANNEL_2:
                IEC1bits.DMA2IE = 0;
                break;
        case DMA_CHANNEL_3:
                IEC2bits.DMA3IE = 0;
                break;
        default:break;
    }
}

/**
 @Summary
  Enables the channel in the DMA

 @Description
  This routine is used to enable a channel in the DMA. This routine
  sets the value of the CHEN bit to 1.

 @Preconditions
  DMA_Initializer() function should have been
  called before calling this function.

 @Returns
  None

 @Param
  None
```

```
271
272  @Example
273  Refer to DMA_Initializer(); for an example
274  */
275 inline static void DMA_ChannelEnable(DMA_CHANNEL  channel)
276 {
277     switch(channel) {
278         case DMA_CHANNEL_0:
279                 DMA0CONbits.CHEN = 1;
280                 break;
281         case DMA_CHANNEL_1:
282                 DMA1CONbits.CHEN = 1;
283                 break;
284         case DMA_CHANNEL_2:
285                 DMA2CONbits.CHEN = 1;
286                 break;
287         case DMA_CHANNEL_3:
288                 DMA3CONbits.CHEN = 1;
289                 break;
290         default: break;
291     }
292 }
293
294 /**
295  @Summary
296  Disables the channel in the DMA
297
298  @Description
299   This routine is used to disable a channel in the DMA. This routine
300   sets the value of the CHEN bit to 0.
301
302 @Preconditions
303  DMA_Initializer() function should have been
304  called before calling this function.
305
306 @Returns
307  None
308
309 @Param
310  None
311
312 @Example
313  Refer to DMA_Initializer(); for an example
314 */
315 inline static void DMA_ChannelDisable(DMA_CHANNEL  channel)
316 {
317     switch(channel) {
318         case DMA_CHANNEL_0:
319                 DMA0CONbits.CHEN = 0;
320                 break;
321         case DMA_CHANNEL_1:
322                 DMA1CONbits.CHEN = 0;
323                 break;
324         case DMA_CHANNEL_2:
325                 DMA2CONbits.CHEN = 0;
326                 break;
327         case DMA_CHANNEL_3:
```

```
328                 DMA3CONbits.CHEN = 0;
329                 break;
330         default: break;
331     }
332 }
333 /**
334   @Summary
335     Sets the transfer count of the DMA
336 
337   @Description
338     This routine is used to set the DMA transfer count. This routine sets the
339     value of the DMACNT register.
340 
341   @Preconditions
342     DMA_Initializer() function should have been
343     called before calling this function.
344 
345   @Returns
346     None
347 
348   @Param
349     None
350 
351   @Example
352     Refer to DMA_Initializer(); for an example
353  */
354 inline static void DMA_TransferCountSet(DMA_CHANNEL channel, uint16_t
      transferCount)
355 {
356     switch(channel) {
357         case DMA_CHANNEL_0:
358                 DMA0CNT = transferCount;
359                 break;
360         case DMA_CHANNEL_1:
361                 DMA1CNT = transferCount;
362                 break;
363         case DMA_CHANNEL_2:
364                 DMA2CNT = transferCount;
365                 break;
366         case DMA_CHANNEL_3:
367                 DMA3CNT = transferCount;
368                 break;
369         default: break;
370     }
371 }
372 /**
373  @Summary
374   Returns the transfer count of the DMA
375 
376 @Description
377  This routine is used to determine the DMA transfer count. This routine
378  returns the value of the DMACNT register.
379 
380 @Preconditions
381   DMA_Initializer() function should have been
382   called before calling this function.
383 
```

```
384 @Returns
385  Returns the transfer count of the DMA
386
387 @Param
388  None
389
390 @Example
391  Refer to DMA_Initializer(); for an example
392 */
393 inline static uint16_t DMA_TransferCountGet(DMA_CHANNEL channel)
394     {
395     switch(channel) {
396             case DMA_CHANNEL_0:
397                 return (DMA0CNT);
398             case DMA_CHANNEL_1:
399                 return (DMA1CNT);
400             case DMA_CHANNEL_2:
401                 return (DMA2CNT);
402             case DMA_CHANNEL_3:
403                 return (DMA3CNT);
404         default: return 0;
405     }
406 }
407
408 /**
409  @Summary
410   Initiates a transfer on the requested DMA channel.
411
412  @Description
413   This routine is used to initiate a transfer on the requested DMA channel.
        When a transfer on the requested channel is initiated the routine
414   returns the value of the FORCE bit. It returns 0 otherwise.
415
416  @Preconditions
417    DMA_Initializer() function should have been
418    called before calling this function.
419
420  @Returns
421   Returns true if the transfer on the requested channel is initiated
422
423  @Param
424   None
425
426  @Example
427   Refer to DMA_Initializer(); for an example
428  */
429 inline static void DMA_SoftwareTriggerEnable(DMA_CHANNEL channel )
430 {
431     switch(channel) {
432         case DMA_CHANNEL_0:
433                 DMA0REQbits.FORCE = 1;
434         case DMA_CHANNEL_1:
435                 DMA1REQbits.FORCE = 1;
436         case DMA_CHANNEL_2:
437                 DMA2REQbits.FORCE = 1;
438         case DMA_CHANNEL_3:
439                 DMA3REQbits.FORCE = 1;
```

```
440         default: break;
441     }
442 }
443 
444 /**
445   @Summary
446     Sets the address for register A in the DMA
447 
448   @Description
449     This routine is used to set the address in register A for a DMA channel.
450 
451   @Preconditions
452     DMA_Initializer() function should have been
453     called before calling this function.
454 
455   @Returns
456     None
457 
458   @Param
459     None
460 
461   @Example
462     Refer to DMA_Initializer(); for an example
463  */
464 inline static void DMA_StartAddressASet(DMA_CHANNEL  channel, uint16_t address
        )
465 {
466     switch(channel) {
467         case DMA_CHANNEL_0:
468                 DMA0STAL = address;
469                 DMA0STAH = 0;
470                 break;
471         case DMA_CHANNEL_1:
472                 DMA1STAL = address;
473                 DMA1STAH = 0;
474                 break;
475         case DMA_CHANNEL_2:
476                 DMA2STAL = address;
477                 DMA2STAH = 0;
478                 break;
479         case DMA_CHANNEL_3:
480                 DMA3STAL = address;
481                 DMA3STAH = 0;
482                 break;
483         default: break;
484     }
485     }
486 
487 /**
488   @Summary
489     Sets the address for register B in the DMA
490 
491   @Description
492     This routine is used to set the address in register B for a DMA channel.
493 
494   @Preconditions
495     DMA_Initializer() function should have been
```

```
496     called before calling this function.
497
498   @Returns
499     None
500
501   @Param
502     None
503
504   @Example
505     Refer to DMA_Initializer(); for an example
506  */
507
508 inline static void DMA_StartAddressBSet(DMA_CHANNEL  channel, uint16_t address
        )
509 {
510     switch(channel) {
511         case DMA_CHANNEL_0:
512                 DMA0STBL = address;
513                 DMA0STBH = 0;
514                 break;
515         case DMA_CHANNEL_1:
516                 DMA1STBL = address;
517                 DMA1STBH = 0;
518                 break;
519         case DMA_CHANNEL_2:
520                 DMA2STBL = address;
521                 DMA2STBH = 0;
522                 break;
523         case DMA_CHANNEL_3:
524                 DMA3STBL = address;
525                 DMA3STBH = 0;
526                 break;
527         default: break;
528     }
529 }
530     /**
531   @Summary
532     Gets the address for register A in the DMA
533
534   @Description
535     This routine is used to get the address in register A for a DMA channel.
536
537   @Preconditions
538     DMA_Initializer() function should have been
539     called before calling this function.
540
541   @Returns
542     None
543
544   @Param
545     None
546
547   @Example
548     Refer to DMA_Initializer(); for an example
549  */
550 inline static uint16_t DMA_StartAddressAGet(DMA_CHANNEL  channel, uint16_t
        address)
```

```
551 {
552     switch(channel) {
553         case DMA_CHANNEL_0:
554                 address= DMA0STAL;
555                 break;
556         case DMA_CHANNEL_1:
557                 address= DMA1STAL;
558                 break;
559         case DMA_CHANNEL_2:
560                 address= DMA2STAL;
561                 break;
562         case DMA_CHANNEL_3:
563                 address= DMA3STAL;
564                 break;
565         default:
566                 address = 0;
567                 break;
568     }
569         return address;
570
571 }
572     /**
573   @Summary
574     Sets the address for register B in the DMA
575
576   @Description
577     This routine is used to set the address in register B for a DMA channel.
578
579   @Preconditions
580     DMA_Initializer() function should have been
581     called before calling this function.
582
583   @Returns
584     None
585
586   @Param
587     None
588
589   @Example
590     Refer to DMA_Initializer(); for an example
591  */
592
593 inline static uint16_t DMA_StartAddressBGet(DMA_CHANNEL  channel, uint16_t
      address) {
594
595     switch(channel) {
596         case DMA_CHANNEL_0:
597                 address= DMA0STBL;
598                 break;
599         case DMA_CHANNEL_1:
600                 address= DMA1STBL;
601                 break;
602         case DMA_CHANNEL_2:
603                 address= DMA2STBL;
604                 break;
605         case DMA_CHANNEL_3:
606                 address= DMA3STBL;
```

```
607                 break;
608         default:
609                 address = 0;
610                 break;
611     }
612         return address;
613 }
614     /**
615     @Summary
616     Sets the peripheral address in the DMA
617 
618   @Description
619     This routine is used to set the peripheral address for a DMA channel.
620 
621   @Preconditions
622     DMA_Initializer() function should have been
623     called before calling this function.
624 
625   @Returns
626     None
627 
628   @Param
629     None
630 
631   @Example
632     Refer to DMA_Initializer(); for an example
633  */
634  inline static void DMA_PeripheralAddressSet(DMA_CHANNEL channel, volatile
       unsigned int * address)
635     {
636     switch(channel) {
637         case DMA_CHANNEL_0:
638                 DMA0PAD = (int)address;
639                 break;
640         case DMA_CHANNEL_1:
641                 DMA1PAD = (int)address;
642                 break;
643         case DMA_CHANNEL_2:
644                 DMA2PAD = (int)address;
645                 break;
646         case DMA_CHANNEL_3:
647                 DMA3PAD = (int)address;
648                 break;
649         default: break;
650     }
651 }
652 /**
653   @Summary
654     Returns true when there is a Peripheral Write Collision Event
655 
656   @Description
657     This routine is used to determine if there is a Peripheral Write Collision
        Event. This routine
658     returns the value of the PWCOL bit in DMAPWC register. When there is a
        Peripheral Write Collision Event, the routine
659     returns 1. It returns 0 otherwise.
660 
```

```
661   @Preconditions
662     DMA_Initializer() function should have been
663     called before calling this function.
664
665   @Returns
666     Returns true if there is a Peripheral Write Collision Event
667
668   @Param
669     None
670
671   @Example
672     Refer to DMA_Initializer(); for an example
673  */
674 inline static bool DMA_IsPeripheralWriteCollision(uint16_t dmaChannel )
675 {
676     return DMAPWC & (1 << dmaChannel);
677 }
678
679 /**
680   @Summary
681     Returns true when there is a Request Collision Event
682
683   @Description
684     This routine is used to determine if there is a Request Collision Event.
        This routine
685     returns the value of the RQCOL bit in DMARQC register. When there is a
        Request Collision Event, the routine
686     returns 1. It returns 0 otherwise.
687
688   @Preconditions
689     DMA_Initializer() function should have been
690     called before calling this function.
691
692   @Returns
693     Returns true if there is a Request Collision Event
694
695   @Param
696     None
697
698   @Example
699     Refer to DMA_Initializer(); for an example
700  */
701 inline static bool DMA_IsRequestCollision(uint16_t dmaChannel )
702 {
703     return DMARQC & (1 << dmaChannel);
704 }
705
706 /**
707   @Summary
708     Sets the requested DMA Channel IRQ Select register with the requested
        peripheral IRQ number.
709
710   @Description
711     This routine is used to set the requested DMA Channel IRQ Select register
        with the requested peripheral IRQ number.
712
713   @Preconditions
```

```
714     DMA_Initializer() function should have been
715     called before calling this function.
716
717   @Returns
718     None
719
720   @Param
721     None
722
723   @Example
724     Refer to DMA_Initializer(); for an example
725  */
726 inline static void DMA_PeripheralIrqNumberSet(DMA_CHANNEL channel, uint8_t
      irqNumber)
727 {
728     switch(channel) {
729         case DMA_CHANNEL_0:
730                 DMA0REQ = irqNumber;
731                 break;
732         case DMA_CHANNEL_1:
733                 DMA1REQ = irqNumber;
734                 break;
735         case DMA_CHANNEL_2:
736                 DMA2REQ = irqNumber;
737                 break;
738         case DMA_CHANNEL_3:
739                 DMA3REQ = irqNumber;
740                 break;
741         default: break;
742     }
743 }
744
745 #ifdef __cplusplus  // Provide C++ Compatibility
746
747     }
748
749 #endif
750
751 #endif  // DMA.h
752
753
754 /**
755   End of File
756 */
```

dma.c file

```
/*
    ********************************************************************************

  DMA Generated Driver File

Company:
Microchip Technology Inc.

File Name:
    dma.c

Summary:
This is the generated driver implementation file for the DMA driver using
    PIC24 / dsPIC33 / PIC32MM MCUs

Description:
This source file provides implementations for driver APIs for DMA.
Generation Information :
Product Revision  :  PIC24 / dsPIC33 / PIC32MM MCUs - pic24-dspic-pic32mm :
    1.75.1
Device            :  dsPIC33EV256GM102
The generated drivers are tested against the following:
Compiler          :  XC16 v1.35
MPLAB             :  MPLAB X v5.05
********************************************************************************
    */

/*
    (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this
    software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
    EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY
    IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
    PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS,
    COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
    INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
    BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS
    IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF
    ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF
    THESE
    TERMS.
*/

#include <xc.h>
#include "dma.h"
```

```
47 void DMA_Initialize(void)
48 {
49     // Initialize channels which are enabled
50 
51     // AMODE Peripheral Indirect Addressing mode; CHEN disabled; DIR Reads
       from RAM address, writes to peripheral address; HALF Initiates interrupt
       when all of the data has been moved; SIZE 16 bit; NULLW disabled; MODE
       Continuous, Ping-Pong modes are disabled;
52     DMA0CON= 0x2020 & 0x7FFF; //Enable DMA Channel later;
53     // FORCE disabled; IRQSEL ECAN1 TX;
54     DMA0REQ= 0x46;
55     // CNT 7;
56     DMA0CNT= 0x7;
57     // STA 4096;
58     DMA0STAL= 0x1000;
59     // STA 0;
60     DMA0STAH= 0x0;
61     // Clearing Channel 0 Interrupt Flag;
62     IFS0bits.DMA0IF = false;
63     // Enabling Channel 0 Interrupt
64 
65 
66     // AMODE Peripheral Indirect Addressing mode; CHEN disabled; SIZE 16 bit;
       DIR Reads from peripheral address, writes to RAM address; NULLW disabled;
        HALF Initiates interrupt when all of the data has been moved; MODE
       Continuous, Ping-Pong modes are disabled;
67     DMA1CON= 0x20 & 0x7FFF; //Enable DMA Channel later;
68     // FORCE disabled; IRQSEL ECAN1 RX;
69     DMA1REQ= 0x22;
70     // CNT 7;
71     DMA1CNT= 0x7;
72     // STA 4096;
73     DMA1STAL= 0x1000;
74     // STA 0;
75     DMA1STAH= 0x0;
76     // Clearing Channel 1 Interrupt Flag;
77     IFS0bits.DMA1IF = false;
78     // Enabling Channel 1 Interrupt
79 
80 
81     // AMODE Register Indirect with Post-Increment mode; CHEN disabled; SIZE
       16 bit; DIR Reads from peripheral address, writes to RAM address; NULLW
       disabled; HALF Initiates interrupt when all of the data has been moved;
       MODE Continuous, Ping-Pong modes are disabled;
82     DMA2CON= 0x0 & 0x7FFF; //Enable DMA Channel later;
83     // IRQSEL INT0; FORCE disabled;
84     DMA2REQ= 0x0;
85     // CNT 0;
86     DMA2CNT= 0x0;
87     // STA 4096;
88     DMA2STAL= 0x1000;
89     // STA 0;
90     DMA2STAH= 0x0;
91     // Clearing Channel 2 Interrupt Flag;
92     IFS1bits.DMA2IF = false;
93     // Enabling Channel 2 Interrupt
94 
```

```
95
96      // MODE Continuous, Ping-Pong modes are disabled; AMODE Register Indirect
        with Post-Increment mode; CHEN disabled; HALF Initiates interrupt when
        all of the data has been moved; SIZE 16 bit; DIR Reads from peripheral
        address, writes to RAM address; NULLW disabled;
97      DMA3CON= 0x0 & 0x7FFF; //Enable DMA Channel later;
98      // IRQSEL INT0; FORCE disabled;
99      DMA3REQ= 0x0;
100     // CNT 0;
101     DMA3CNT= 0x0;
102     // STA 4096;
103     DMA3STAL= 0x1000;
104     // STA 0;
105     DMA3STAH= 0x0;
106     // Clearing Channel 3 Interrupt Flag;
107     IFS2bits.DMA3IF = false;
108     // Enabling Channel 3 Interrupt
109
110
111 }
112
113 /**
114   End of File
115 */
```

ecan1.h file

```
/**
  ECAN1 Generated Driver API Header File

  @Company
    Microchip Technology Inc.

  @File Name
    ecan1.h

  @Summary
    This is the generated header file for the ECAN1 driver using PIC24 /
    dsPIC33 / PIC32MM MCUs

  @Description
    This header file provides APIs for driver for ECAN1.
    Generation Information :
        Product Revision  :  PIC24 / dsPIC33 / PIC32MM MCUs - 1.75.1
        Device            :  dsPIC33EV256GM102
        Driver Version    :  1.00
    The generated drivers are tested against the following:
        Compiler          :  XC16 v1.35
        MPLAB             :  MPLAB X v5.05
*/

/*
    (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this
    software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
    EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY
    IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
    PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS,
    COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
    INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
    BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS
    IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF
    ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF
    THESE
    TERMS.
*/

#ifndef _ECAN1_H
#define _ECAN1_H

/**
  Section: Included Files
```

```
51 */
52
53 #include <xc.h>
54 #include <stdint.h>
55 #include <stdbool.h>
56
57 /* ECAN message type identifiers */
58 #define CAN1_MSG_DATA     0x01
59 #define CAN1_MSG_RTR      0x02
60 #define CAN1_FRAME_EXT    0x03
61 #define CAN1_FRAME_STD    0x04
62 #define CAN1_BUF_FULL 0x05
63 #define CAN1_BUF_EMPTY    0x06
64
65 typedef union {
66     struct {
67         uint32_t id;
68         uint8_t idType;
69         uint8_t msgtype;
70         uint8_t dlc;
71         uint8_t data0;
72         uint8_t data1;
73         uint8_t data2;
74         uint8_t data3;
75         uint8_t data4;
76         uint8_t data5;
77         uint8_t data6;
78         uint8_t data7;
79     } frame;
80     unsigned char array[16];
81 } uCAN1_MSG;
82
83 /* Operation modes */
84 typedef enum
85 {
86   CAN1_NORMAL_OPERATION_MODE = 0,
87   CAN1_DISABLE_MODE = 1,
88   CAN1_LOOPBACK_MODE = 2,
89   CAN1_LISTEN_ONLY_MODE = 3,
90   CAN1_CONFIGURATION_MODE = 4,
91   CAN1_LISTEN_ALL_MESSAGES_MODE = 7
92 }ECAN1_OP_MODES;
93
94 typedef enum{
95     ECAN1_PRIORITY_HIGH = 0b11,
96     ECAN1_PRIORITY_MEDIUM = 0b10,
97     ECAN1_PRIORITY_LOW = 0b01,
98     ECAN1_PRIORITY_NONE = 0b00
99 } ECAN1_TX_PRIOIRTY;
100
101 #ifdef __cplusplus  // Provide C++ Compatibility
102
103     extern "C" {
104
105 #endif
106
107 /**
```

```
108   Section: ECAN1 Module APIs
109 */
110
111 /**
112   @Summary
113     Initializes the ECAN1_Initialize.
114
115   @Description
116     This routine initializes the ECAN1_Initialize.
117     This routine must be called before any other ECAN1 routine is called.
118     This routine should only be called once during system initialization.
119
120   @Preconditions
121     None
122
123   @Param
124     None
125
126   @Returns
127     None
128
129   @Comment
130
131
132  @Example
133     <code>
134     ECAN1_Initialize();
135     </code>
136  */
137 void ECAN1_Initialize(void);
138
139 /*
        ***************************************************************************

140 *
141 *     Function:     ECAN1_receive
142 *     Description:        Receives the message from CAN buffer to user buffer
143 *     Arguments:    recCanMsg: pointer to the message object
144 *     Return Value:       true - Receive successful
145 *                         false - Receive failure
146 ***************************************************************************
        */
147 bool ECAN1_receive(uCAN1_MSG *recCanMsg);
148
149 /*
        ***************************************************************************

150 *
151 *     Function:     ECAN1_transmit
152 *     Description:        Transmits the message from user buffer to CAN buffer
153 *     Arguments:    priority: priority of the message to be transmitted
154 *                         sendCanMsg: pointer to the message object
155 *     Return Value:       true - Transmit successful
156 *                         false - Transmit failure
157 ***************************************************************************
        */
158 bool ECAN1_transmit(ECAN1_TX_PRIOIRTY priority,
```

```
159                                     uCAN1_MSG *sendCanMsg);
160
161 /*
        ******************************************************************************
162 *     Function:           ECAN1_isBusOff
163 *     Description:        Checks whether the transmitter in Bus off state
164 *     Return Value:       true - Transmitter in Bus Off state
165 *                         false - Transmitter not in Bus Off state
166 ******************************************************************************
        */
167 bool ECAN1_isBusOff();
168
169 /*
        ******************************************************************************
170 *     Function:     ECAN1_isRXErrorPassive
171 *     Description:        Checks whether the receive in error passive state
172 *     Return Value:       true - Receiver in Error Passive state
173 *                         false - Receiver not in Error Passive state
174 ******************************************************************************
        */
175 bool ECAN1_isRXErrorPassive();
176
177 /*
        ******************************************************************************
178 *     Function:     ECAN1_isTXErrorPassive
179 *     Description:        Checks whether the transmitter in error passive state
180 *     Return Value:       true - Transmitter in Error Passive state
181 *                         false - Transmitter not in Error Passive state
182 ******************************************************************************
        */
183 bool ECAN1_isTXErrorPassive();
184
185 /*
        ******************************************************************************
186 *     Function:     ECAN1_messagesInBuffer
187 *     Description:        returns the number of messages that are received
188 *     Return Value:       Number of message received
189 ******************************************************************************
        */
190 uint8_t ECAN1_messagesInBuffer();
191
192 /*
        ******************************************************************************
193 *
194 *     Function:     ECAN1_sleep
195 *     Description:        Puts ECAN1 module in disable mode.
196 *     Return Value:       None
197 *
198 ******************************************************************************
        */
199 void ECAN1_sleep();
200
```

```
201 /*
        ********************************************************************************
202 *     Function:     ECAN1_TransmitEnable
203 *     Description:        Enables Transmit for ECAN1
204 *     Return Value:       None
205 ********************************************************************************
        */
206 void ECAN1_TransmitEnable();
207
208 /*
        ********************************************************************************
209 *     Function:     ECAN1_ReceiveEnable
210 *     Description:        Enables Receive for ECAN1
211 *     Return Value:       None
212 ********************************************************************************
        */
213 void ECAN1_ReceiveEnable();
214
215 #ifdef __cplusplus  // Provide C++ Compatibility
216
217     }
218
219 #endif
220
221 #endif  //_ECAN1_H
222 /**
223  End of File
224 */
```

ecan1.c file

```
/**
  ECAN1 Generated Driver File

  @Company
    Microchip Technology Inc.

  @File Name
    ecan1.c

  @Summary
    This is the generated driver implementation file for the ECAN1 driver
    using PIC24 / dsPIC33 / PIC32MM MCUs

  @Description
    This source file provides APIs for ECAN1.
    Generation Information :
        Product Revision  :  PIC24 / dsPIC33 / PIC32MM MCUs - 1.75.1
        Device            :  dsPIC33EV256GM102
        Driver Version    :  1.00
    The generated drivers are tested against the following:
        Compiler          :  XC16 v1.35
        MPLAB             :  MPLAB X v5.05
*/

/*
    (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this
    software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
    EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY
    IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
    PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS,
    COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
    INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
    BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS
    IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF
    ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF
    THESE
    TERMS.
*/

/**
  Section: Included Files
*/

#include <xc.h>
```

```
51 #include "ecan1.h"
52 #include "dma.h"
53
54 #define ECAN1_TX_DMA_CHANNEL DMA_CHANNEL_0
55 #define ECAN1_RX_DMA_CHANNEL DMA_CHANNEL_1
56
57 /* Valid options are 4, 6, 8, 12, 16, 24, or 32. */
58 #define CAN1_MESSAGE_BUFFERS          32
59
60 #define CAN1_TX_BUFFER_COUNT 1
61
62 /*****************************************************************************
       */
63
64 /*****************************************************************************
       */
65 /* Private type definitions                                                  */
66 /*****************************************************************************
       */
67 typedef struct __attribute__((packed))
68 {
69     unsigned priority                   :2;
70     unsigned remote_transmit_enable     :1;
71     unsigned send_request               :1;
72     unsigned error                      :1;
73     unsigned lost_arbitration           :1;
74     unsigned message_aborted            :1;
75     unsigned transmit_enabled           :1;
76 } CAN1_TX_CONTROLS;
77
78 /*****************************************************************************
       */
79 /* Private variable definitions
       */
80 /*****************************************************************************
       */
81 /* This alignment is required because of the DMA's peripheral indirect
82  * addressing mode. */
83 static unsigned int ecan1msgBuf [CAN1_MESSAGE_BUFFERS][8] __attribute__((
       aligned(32 * 8 * 2)));
84
85 /*****************************************************************************
       */
86 /* Private function prototypes
       */
87 /*****************************************************************************
       */
88 static void ECAN1_DMACopy(uint8_t buffer_number, uCAN1_MSG *message);
89 static void ECAN1_MessageToBuffer(uint16_t* buffer, uCAN1_MSG* message);
90
91 /* Null weak implementations of callback functions. */
92 void __attribute__((weak)) ECAN1_CallbackBusOff(void){}
93 void __attribute__((weak)) ECAN1_CallbackTxErrorPassive(void){}
94 void __attribute__((weak)) ECAN1_CallbackRxErrorPassive(void){}
95 void __attribute__((weak)) ECAN1_CallbackMessageReceived(void){}
96
97 /**
```

```
98    Section: ECAN1 APIs
99  ******************************************************************************
        */
100
101 void ECAN1_Initialize(void)
102 {
103     // Disable interrupts before the Initialization
104     IEC2bits.C1IE = 0;
105     C1INTE = 0;
106
107     // set the CAN_initialize module to the options selected in the User
        Interface
108
109     /* put the module in configuration mode */
110     C1CTRL1bits.REQOP = CAN1_CONFIGURATION_MODE;
111     while(C1CTRL1bits.OPMODE != CAN1_CONFIGURATION_MODE);
112
113     /* Set up the baud rate*/
114     C1CFG1 = 0x03;  //BRP TQ = (2 x 4)/FCAN; SJW 1 x TQ;
115     C1CFG2 = 0x1D8; //WAKFIL disabled; SEG2PHTS Freely programmable; SEG2PH 2
        x TQ; SEG1PH 4 x TQ; PRSEG 1 x TQ; SAM Three times at the sample point;
116     C1FCTRL = 0xC001; //FSA Transmit/Receive Buffer TRB1; DMABS 32;
117     C1FEN1 = 0x01;  //FLTEN8 disabled; FLTEN7 disabled; FLTEN9 disabled;
        FLTEN0 enabled; FLTEN2 disabled; FLTEN10 disabled; FLTEN1 disabled;
        FLTEN11 disabled; FLTEN4 disabled; FLTEN3 disabled; FLTEN6 disabled;
        FLTEN5 disabled; FLTEN12 disabled; FLTEN13 disabled; FLTEN14 disabled;
        FLTEN15 disabled;
118     C1CTRL1 = 0x00; //CANCKS FOSC/2; CSIDL disabled; ABAT disabled; REQOP Sets
         Normal Operation Mode; WIN Uses buffer window; CANCAP disabled;
119
120     /* Filter configuration */
121     /* enable window to access the filter configuration registers */
122     /* use filter window*/
123     C1CTRL1bits.WIN=1;
124
125     /* select acceptance masks for filters */
126     C1FMSKSEL1bits.F0MSK = 0x0; //Select Mask 0 for Filter 0
127
128     /* Configure the masks */
129     C1RXM0SIDbits.SID = 0x7ff;
130     C1RXM1SIDbits.SID = 0x0;
131     C1RXM2SIDbits.SID = 0x0;
132
133     C1RXM0SIDbits.EID = 0x0;
134     C1RXM1SIDbits.EID = 0x0;
135     C1RXM2SIDbits.EID = 0x0;
136
137     C1RXM0EID = 0x00;
138     C1RXM1EID = 0x00;
139     C1RXM2EID = 0x00;
140
141     C1RXM0SIDbits.MIDE = 0x0;
142     C1RXM1SIDbits.MIDE = 0x0;
143     C1RXM2SIDbits.MIDE = 0x0;
144
145     /* Configure the filters */
146     C1RXF0SIDbits.SID = 0x8;
```

```
147
148     C1RXF0SIDbits.EID = 0x0;
149
150     C1RXF0EID = 0x00;
151
152     C1RXF0SIDbits.EXIDE = 0x0;
153
154     /* Non FIFO Mode */
155     C1BUFPNT1bits.F0BP = 0x1; //Filter 0 uses Buffer1
156
157     /* clear window bit to access ECAN control registers */
158     C1CTRL1bits.WIN=0;
159
160     /* ECAN1, Buffer 0 is a Transmit Buffer */
161     C1TR01CONbits.TXEN0 = 0x1; // Buffer 0 is a Transmit Buffer
162     C1TR01CONbits.TXEN1 = 0x0; // Buffer 1 is a Receive Buffer
163     C1TR23CONbits.TXEN2 = 0x0; // Buffer 2 is a Receive Buffer
164     C1TR23CONbits.TXEN3 = 0x0; // Buffer 3 is a Receive Buffer
165     C1TR45CONbits.TXEN4 = 0x0; // Buffer 4 is a Receive Buffer
166     C1TR45CONbits.TXEN5 = 0x0; // Buffer 5 is a Receive Buffer
167     C1TR67CONbits.TXEN6 = 0x0; // Buffer 6 is a Receive Buffer
168     C1TR67CONbits.TXEN7 = 0x0; // Buffer 7 is a Receive Buffer
169
170     C1TR01CONbits.TX0PRI = 0x0; // Message Buffer 0 Priority Level
171     C1TR01CONbits.TX1PRI = 0x0; // Message Buffer 1 Priority Level
172     C1TR23CONbits.TX2PRI = 0x0; // Message Buffer 2 Priority Level
173     C1TR23CONbits.TX3PRI = 0x0; // Message Buffer 3 Priority Level
174     C1TR45CONbits.TX4PRI = 0x0; // Message Buffer 4 Priority Level
175     C1TR45CONbits.TX5PRI = 0x0; // Message Buffer 5 Priority Level
176     C1TR67CONbits.TX6PRI = 0x0; // Message Buffer 6 Priority Level
177     C1TR67CONbits.TX7PRI = 0x0; // Message Buffer 7 Priority Level
178
179     /* clear the buffer and overflow flags */
180     C1RXFUL1 = 0x0000;
181     C1RXFUL2 = 0x0000;
182     C1RXOVF1 = 0x0000;
183     C1RXOVF2 = 0x0000;
184
185     /* configure the device to interrupt on the receive buffer full flag */
186     /* clear the buffer full flags */
187     C1INTFbits.RBIF = 0;
188
189     /* put the module in normal mode */
190     C1CTRL1bits.REQOP = CAN1_NORMAL_OPERATION_MODE;
191     while(C1CTRL1bits.OPMODE != CAN1_NORMAL_OPERATION_MODE);
192
193     /* Enable ECAN1 Interrupt */
194     IEC2bits.C1IE = 1;
195
196     /* Enable Receive interrupt */
197     C1INTEbits.RBIE = 1;
198
199     /* Enable Error interrupt*/
200     C1INTEbits.ERRIE = 1;
201
202
203 }
```

```
204
205 /*
    ********************************************************************************
206 *
207 *    Function:    ECAN1_TransmitEnable
208 *    Description:       Setup the DMA for Transmit from the ECAN module.  The
209 *                       relevant DMA module APIs are grouped in this function
210 *                       and this API needs to be called after DMA_Initialize
211 *                       and CAN_Initialize
212 *
213 ********************************************************************************
    */
214 void ECAN1_TransmitEnable()
215 {
216     /* setup channel 0 for peripheral indirect addressing mode
217     normal operation, word operation and select as Tx to peripheral */
218
219     /* DMA_PeripheralIrqNumberSet and DMA_TransferCountSet would be done in
    the
220     DMA */
221
222     /* setup the address of the peripheral ECAN1 (C1TXD) */
223     DMA_PeripheralAddressSet(ECAN1_TX_DMA_CHANNEL, &C1TXD);
224
225     /* DPSRAM start address offset value */
226     /* DPSRAM start address offset value */
227     DMA_StartAddressASet(ECAN1_TX_DMA_CHANNEL, (uint16_t)(&ecan1msgBuf));
228
229     /* enable the channel */
230     DMA_ChannelEnable(ECAN1_TX_DMA_CHANNEL);
231 }
232
233 /*
    ********************************************************************************
234 *
235 *    Function:    ECAN1_ReceiveEnable
236 *    Description:       Setup the DMA for Receive on the ECAN module.  The
237 *                       relevant DMA module APIs are grouped in this function
238 *                       and this API needs to be called after DMA_Initialize
239 *                       and CAN_Initialize
240 *
241 ********************************************************************************
    */
242 void ECAN1_ReceiveEnable()
243 {
244     /* setup DMA channel for peripheral indirect addressing mode
245     normal operation, word operation and select as Rx to peripheral */
246
247     /* setup the address of the peripheral ECAN1 (C1RXD) */
248     /* DMA_TransferCountSet and DMA_PeripheralIrqNumberSet would be set in
249     the DMA_Initialize function */
250
251     DMA_PeripheralAddressSet(ECAN1_RX_DMA_CHANNEL, &C1RXD);
252
253     /* DPSRAM start address offset value */
```

```
254     DMA_StartAddressASet(ECAN1_RX_DMA_CHANNEL, (uint16_t)(&ecan1msgBuf) );
255
256     /* enable the channel */
257     DMA_ChannelEnable(ECAN1_RX_DMA_CHANNEL);
258 }
259
260 /*
        ******************************************************************************
261 *
262 *     Function:     ECAN1_transmit
263 *     Description:        Transmits the message from user buffer to CAN buffer
264 *                         as per the buffer number allocated.
265 *                         Allocation of the buffer number is done by user
266 *
267 *     Arguments:    priority : priority of the message to be transmitted
268 *                         sendCanMsg: pointer to the message object
269 *
270 *     Return Value:       true - Transmit successful
271 *                         false - Transmit failure
272 ******************************************************************************
        */
273 bool ECAN1_transmit(ECAN1_TX_PRIOIRTY priority, uCAN1_MSG *sendCanMsg)
274 {
275     CAN1_TX_CONTROLS * pTxControls = (CAN1_TX_CONTROLS*)&C1TR01CON;
276     uint_fast8_t i;
277     bool messageSent = false;
278
279     for(i=0; i<CAN1_TX_BUFFER_COUNT; i++)
280     {
281         if(pTxControls->transmit_enabled == 1)
282         {
283             if (pTxControls->send_request == 0)
284             {
285                 ECAN1_MessageToBuffer( &ecan1msgBuf[i][0], sendCanMsg );
286
287                 pTxControls->priority = priority;
288
289                 /* set the message for transmission */
290                 pTxControls->send_request = 1;
291
292                 messageSent = true;
293                 break;
294             }
295         }
296
297         pTxControls++;
298     }
299
300     return messageSent;
301 }
302
303 /*
        ******************************************************************************
304 *
305 *     Function:     ECAN1_receive
```

```
306 *     Description:       Receives the message from CAN buffer to user buffer
307 *
308 *     Arguments:    recCanMsg: pointer to the message object
309 *
310 *     Return Value:      true – Receive successful
311 *                        false – Receive failure
312 ********************************************************************************
    */
313 bool ECAN1_receive(uCAN1_MSG *recCanMsg)
314 {
315     /* We use a static buffer counter so we don't always check buffer 0 first
316      * resulting in potential starvation of later buffers.
317      */
318     static uint_fast8_t currentDedicatedBuffer = 0;
319     uint_fast8_t i;
320     bool messageReceived = false;
321     uint16_t receptionFlags;
322
323     receptionFlags = C1RXFUL1;
324
325     if (receptionFlags != 0)
326     {
327         /* check which message buffer is free */
328         for (i=0 ; i < 16; i++)
329         {
330             if (((receptionFlags >> currentDedicatedBuffer ) & 0x1) == 0x1)
331             {
332                 ECAN1_DMACopy(currentDedicatedBuffer, recCanMsg);
333
334                 C1RXFUL1 &= ~(1 << currentDedicatedBuffer);
335
336                 messageReceived = true;
337             }
338
339             currentDedicatedBuffer++;
340
341             if(currentDedicatedBuffer >= 16)
342             {
343                 currentDedicatedBuffer = 0;
344             }
345
346             if(messageReceived == true)
347             {
348                 break;
349             }
350         }
351     }
352
353     return (messageReceived);
354 }
355
356
357 /*
    ********************************************************************************

358 *
359 *     Function:     ECAN1_isBusOff
```

```
360 *     Description:      Checks whether the transmitter in Bus off state
361 *
362
363 *     Return Value:     true - Transmitter in Bus Off state
364 *                       false - Transmitter not in Bus Off state
365 ********************************************************************
    */
366 bool ECAN1_isBusOff()
367 {
368     return C1INTFbits.TXBO;
369 }
370
371 /*
    ********************************************************************

372 *
373 *     Function:     ECAN1_isRXErrorPassive
374 *     Description:      Checks whether the receive in error passive state
375 *
376 *     Return Value:     true - Receiver in Error Passive state
377 *                       false - Receiver not in Error Passive state
378 ********************************************************************
    */
379 bool ECAN1_isRXErrorPassive()
380 {
381     return C1INTFbits.RXBP;
382 }
383
384 /*
    ********************************************************************

385 *
386 *     Function:     ECAN1_isTXErrorPassive
387 *     Description:      Checks whether the transmitter in error passive state
388 *
389 *     Return Value:     true - Transmitter in Error Passive state
390 *                       false - Transmitter not in Error Passive state
391 ********************************************************************
    */
392 bool ECAN1_isTXErrorPassive()
393 {
394     return (C1INTFbits.TXBP);
395 }
396
397 /*
    ********************************************************************

398 *
399 *     Function:     ECAN1_messagesInBuffer
400 *     Description:      returns the number of messages that are received
401 *
402 *     Return Value:     Number of message received
403 ********************************************************************
    */
404 uint8_t ECAN1_messagesInBuffer()
405 {
406     uint_fast8_t messageCount;
```

```
407     uint_fast8_t currentBuffer;
408     uint16_t receptionFlags;
409 
410     messageCount = 0;
411 
412     /* Check any message in buffer 0 to buffer 15*/
413     receptionFlags = C1RXFUL1;
414     if (receptionFlags != 0)
415     {
416         /* check whether a message is received */
417         for (currentBuffer=0 ; currentBuffer < 16; currentBuffer++)
418         {
419             if (((receptionFlags >> currentBuffer ) & 0x1) == 0x1)
420             {
421                 messageCount++;
422             }
423         }
424     }
425 
426     return (messageCount);
427 }
428 
429 /*
      ******************************************************************************
430 *
431 *     Function:     ECAN1_sleep
432 *     Description:        Puts ECAN1 module in disable mode.
433 *
434 ******************************************************************************
      */
435 void ECAN1_sleep(void) {
436     C1INTFbits.WAKIF = 0;
437     C1INTEbits.WAKIE = 1;
438 
439     /* put the module in disable mode */
440     C1CTRL1bits.REQOP=CAN1_DISABLE_MODE;
441     while(C1CTRL1bits.OPMODE != CAN1_DISABLE_MODE);
442 
443     //Wake up from sleep should set the CAN module straight into Normal mode
444 }
445 
446 /*
      ******************************************************************************
447  * PRIVATE FUNCTIONS
448  ******************************************************************************
      */
449 
450 /*
      ******************************************************************************
451 *
452 *     Function:     ECAN1_DMACopy
453 *     Description:        moves the message from the DMA memory to RAM
454 *
455 *     Arguments:    *message: a pointer to the message structure in RAM
```

```
456 *     that will store the message.
457 *
458 *
459 ********************************************************************************
    */
460 static void ECAN1_DMACopy(uint8_t buffer_number, uCAN1_MSG *message)
461 {
462     uint16_t ide=0;
463     uint16_t rtr=0;
464     uint32_t id=0;
465
466     /* read word 0 to see the message type */
467     ide=ecan1msgBuf[buffer_number][0] & 0x0001U;
468
469     /* check to see what type of message it is */
470     /* message is standard identifier */
471     if(ide==0U)
472     {
473         message->frame.id=(ecan1msgBuf[buffer_number][0] & 0x1FFCU) >> 2U;
474         message->frame.idType = CAN1_FRAME_STD;
475         rtr=ecan1msgBuf[buffer_number][0] & 0x0002U;
476     }
477     /* message is extended identifier */
478     else
479     {
480         id=ecan1msgBuf[buffer_number][0] & 0x1FFCU;
481         message->frame.id = id << 16U;
482         message->frame.id += ( ((uint32_t)ecan1msgBuf[buffer_number][1] & (
    uint32_t)0x0FFF) << 6U );
483         message->frame.id += ( ((uint32_t)ecan1msgBuf[buffer_number][2] & (
    uint32_t)0xFC00U) >> 10U );
484         message->frame.idType = CAN1_FRAME_EXT;
485         rtr=ecan1msgBuf[buffer_number][2] & 0x0200;
486     }
487     /* check to see what type of message it is */
488     /* RTR message */
489     if(rtr != 0U)
490     {
491         /* to be defined ?*/
492         message->frame.msgtype = CAN1_MSG_RTR;
493     }
494     /* normal message */
495     else
496     {
497         message->frame.msgtype = CAN1_MSG_DATA;
498         message->frame.data0 =(unsigned char)ecan1msgBuf[buffer_number][3];
499         message->frame.data1 =(unsigned char)((ecan1msgBuf[buffer_number][3] &
     0xFF00U) >> 8U);
500         message->frame.data2 =(unsigned char)ecan1msgBuf[buffer_number][4];
501         message->frame.data3 =(unsigned char)((ecan1msgBuf[buffer_number][4] &
     0xFF00U) >> 8U);
502         message->frame.data4 =(unsigned char)ecan1msgBuf[buffer_number][5];
503         message->frame.data5 =(unsigned char)((ecan1msgBuf[buffer_number][5] &
     0xFF00U) >> 8U);
504         message->frame.data6 =(unsigned char)ecan1msgBuf[buffer_number][6];
505         message->frame.data7 =(unsigned char)((ecan1msgBuf[buffer_number][6] &
     0xFF00U) >> 8U);
```

```
506         message->frame.dlc =(unsigned char)(ecan1msgBuf[buffer_number][2] & 0
        x000FU);
507     }
508 }
509
510 /*
        ********************************************************************************
511 *
512 *     Function:     ECAN1_MessageToBuffer
513 *     Description:        This function takes the input message, reformats it,
514 *                         and copies it to the specified CAN module buffer
515 *
516 *     Arguments:    *buffer: a pointer to the buffer where the message
517 *                         would be stored
518 *                         *message: pointer to the input message that is
        received
519 *                         by the CAN module
520 *
521 ********************************************************************************
        */
522 static void ECAN1_MessageToBuffer(uint16_t* buffer, uCAN1_MSG* message)
523 {
524     if(message->frame.idType == CAN1_FRAME_STD)
525     {
526         buffer[0]= (message->frame.id & 0x000007FF) << 2;
527         buffer[1]= 0;
528         buffer[2]= message->frame.dlc & 0x0F;
529     }
530     else
531     {
532         buffer[0]= ( ( (uint16_t)(message->frame.id >> 16 ) & 0x1FFC ) ) | 0b1
        ;
533         buffer[1]= (uint16_t)(message->frame.id >> 6) & 0x0FFF;
534         buffer[2]= (message->frame.dlc & 0x0F) + ( (uint16_t)(message->frame.
        id << 10) & 0xFC00);
535     }
536
537     buffer[3]= ((message->frame.data1)<<8) + message->frame.data0;
538     buffer[4]= ((message->frame.data3)<<8) + message->frame.data2;
539     buffer[5]= ((message->frame.data5)<<8) + message->frame.data4;
540     buffer[6]= ((message->frame.data7)<<8) + message->frame.data6;
541 }
542
543
544 void __attribute__((__interrupt__, no_auto_psv)) _C1Interrupt(void)
545 {
546
547     if (C1INTFbits.ERRIF)
548     {
549
550         if (C1INTFbits.TXBO == 1)
551         {
552             ECAN1_CallbackBusOff();
553             C1INTFbits.TXBO = 0;
554         }
555
```

```
556         if (C1INTFbits.TXBP == 1)
557         {
558             ECAN1_CallbackTxErrorPassive();
559             C1INTFbits.TXBP = 0;
560         }
561 
562         if (C1INTFbits.RXBP == 1)
563         {
564             ECAN1_CallbackRxErrorPassive();
565             C1INTFbits.RXBP = 0;
566         }
567 
568         /* Call error notification function */
569         C1INTFbits.ERRIF = 0;
570 
571     }
572 
573     if(C1INTFbits.RBIF)
574     {
575         C1INTFbits.RBIF = 0;
576 
577         /* Notification function */
578         ECAN1_CallbackMessageReceived();
579     }
580 
581 
582     IFS2bits.C1IF = 0;
583 }
584 
585 
586 
587 /**
588  End of File
589 */
```

interrupt_manager.h file

```
/**
  System Interrupts Generated Driver File

  @Company:
    Microchip Technology Inc.

  @File Name:
    interrupt_manager.h

  @Summary:
    This is the generated driver implementation file for setting up the
    interrupts using PIC24 / dsPIC33 / PIC32MM MCUs

  @Description:
    This source file provides implementations for PIC24 / dsPIC33 / PIC32MM
    MCUs interrupts.
    Generation Information :
        Product Revision  :  PIC24 / dsPIC33 / PIC32MM MCUs - pic24-dspic-
    pic32mm : 1.75.1
        Device            :  dsPIC33EV256GM102
    The generated drivers are tested against the following:
        Compiler          :  XC16 v1.35
        MPLAB             :  MPLAB X v5.05
*/
/*
    (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this
    software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
    EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY
    IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
    PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS,
    COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
    INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
    BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS
    IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF
    ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF
    THESE
    TERMS.
*/

#ifndef _INTERRUPT_MANAGER_H
#define _INTERRUPT_MANAGER_H

/**
  @Summary
```

```
50      Initializes the interrupt priorities of the dsPIC33EV256GM102
51
52    @Description
53      This routine sets the interrupt priorities of the modules that have been
        configured
54      for the dsPIC33EV256GM102
55
56    @Preconditions
57      None.
58
59    @Returns
60      None.
61
62    @Param
63      None.
64
65    @Example
66      <code>
67      void SYSTEM_Initialize(void)
68      {
69          // Other initializers are called from this function
70          INTERRUPT_Initialize ();
71      }
72      </code>
73
74  */
75  void INTERRUPT_Initialize(void);
76
77  /**
78    @Summary
79      Enables global interrupts of the dsPIC33EV256GM102
80
81    @Description
82      This routine enables the global interrupt bit for the dsPIC33EV256GM102
83
84    @Preconditions
85      None.
86
87    @Returns
88      None.
89
90    @Param
91      None.
92
93    @Example
94      <code>
95      void SYSTEM_Initialize(void)
96      {
97          // Other initializers are called from this function
98          INTERRUPT_GlobalEnable ();
99      }
100     </code>
101
102 */
103 inline static void INTERRUPT_GlobalEnable(void)
104 {
105     __builtin_enable_interrupts();
```

```
106 }
107
108 /**
109   @Summary
110     Disables global interrupts of the dsPIC33EV256GM102
111
112   @Description
113     This routine disables the global interrupt bit for the dsPIC33EV256GM102
114
115   @Preconditions
116     None.
117
118   @Returns
119     None.
120
121   @Param
122     None.
123
124   @Example
125     <code>
126     void SYSTEM_Initialize(void)
127     {
128         // Other initializers are called from this function
129         INTERRUPT_GlobalDisable ();
130     }
131     </code>
132
133 */
134 inline static void INTERRUPT_GlobalDisable(void)
135 {
136     __builtin_disable_interrupts();
137 }
138
139
140 #endif
```

interrupt_manager.c file

```
/**
  System Interrupts Generated Driver File

  @Company:
    Microchip Technology Inc.

  @File Name:
    interrupt_manager.h

  @Summary:
    This is the generated driver implementation file for setting up the
    interrupts using PIC24 / dsPIC33 / PIC32MM MCUs

  @Description:
    This source file provides implementations for PIC24 / dsPIC33 / PIC32MM
    MCUs interrupts.
    Generation Information :
        Product Revision  :  PIC24 / dsPIC33 / PIC32MM MCUs - pic24-dspic-
    pic32mm : 1.75.1
        Device            :  dsPIC33EV256GM102
    The generated drivers are tested against the following:
        Compiler          :  XC16 v1.35
        MPLAB             :  MPLAB X v5.05
*/
/*
    (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this
    software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
    EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY
    IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
    PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS,
    COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
    INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
    BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS
    IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF
    ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF
    THESE
    TERMS.
*/

/**
    Section: Includes
*/
#include <xc.h>

```

```
50 /**
51     void INTERRUPT_Initialize (void)
52 */
53 void INTERRUPT_Initialize (void)
54 {
55     //     CI: ECAN1 Event
56     //     Priority: 1
57         IPC8bits.C1IP = 1;
58     //     TI: Timer 2
59     //     Priority: 1
60         IPC1bits.T2IP = 1;
61 }
```

mcc.h file

```
/**
  @Generated PIC24 / dsPIC33 / PIC32MM MCUs Header File

  @Company:
    Microchip Technology Inc.

  @File Name:
    mcc.h

  @Summary:
    This is the mcc.h file generated using PIC24 / dsPIC33 / PIC32MM MCUs

  @Description:
    This header file provides implementations for driver APIs for all modules selected in the GUI.
    Generation Information :
        Product Revision  :  PIC24 / dsPIC33 / PIC32MM MCUs - pic24-dspic-pic32mm : 1.75.1
        Device            :  dsPIC33EV256GM102
    The generated drivers are tested against the following:
        Compiler          :  XC16 v1.35
        MPLAB             :  MPLAB X v5.05
*/

/*
    (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS, COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF THESE
    TERMS.
*/

#ifndef MCC_H
#define MCC_H
#include <xc.h>
#include "reset.h"
#include "system.h"
```

```
50 #include "system_types.h"
51 #include "clock.h"
52 #include "pin_manager.h"
53 #include <stdint.h>
54 #include <stdbool.h>
55 #include "dma.h"
56 #include "pwm.h"
57 #include "ecan1.h"
58 #include "adc1.h"
59 #include "tmr2.h"
60 #include "tmr4.h"
61 #include "reset.h"
62 #include "watchdog.h"
63 #include "interrupt_manager.h"
64 #include "traps.h"
65
66 #define _XTAL_FREQ  138187500UL
67
68 /**
69  * @Param
70     none
71  * @Returns
72     none
73  * @Description
74     Initializes the oscillator to the default states configured in the
75  *                  MCC GUI
76  * @Example
77     OSCILLATOR_Initialize(void);
78  */
79 void OSCILLATOR_Initialize(void) __attribute__((deprecated ("\nThis will be
    removed in future MCC releases. \nUse CLOCK_Initialize (void) instead. ")
    ));
80
81 /**
82 * Checks reset cause, flashes UI with an error code as a result.
83 *
84 * Note: this function should be called before any use of CLRWDT
85 * since it has a side-effect of clearing the appropriate bits in the
86 * register showing reset cause (see DS70602B page 8-10)
87 */
88 uint16_t SYSTEM_GetResetCause(void) __attribute__((deprecated ("\nThis will be
     removed in future MCC releases. \nUse RESET_GetCause(void) (void)
    instead. ")));
89
90
91 /**
92  * Enables Watch Dog Timer (WDT) using the software bit.
93  * @example
94  * <code>
95  * WDT_WatchdogtimerSoftwareEnable();
96  * </code>
97  */
98 __attribute__((deprecated ("\nThis will be removed in future MCC releases. \
    nUse WATCHDOG_TimerSoftwareEnable (void) instead. ")))
99 inline static void WDT_WatchdogtimerSoftwareEnable(void)
100 {
101     RCONbits.SWDTEN = 1;
```

```
102 }
103
104 /**
105  * Disables Watch Dog Timer (WDT) using the software bit.
106  * @example
107  * <code>
108  * WDT_WatchdogtimerSoftwareDisable();
109  * </code>
110  */
111 __attribute__((deprecated ("\nThis will be removed in future MCC releases. \
        nUse WATCHDOG_TimerSoftwareDisable (void) instead. ")))
112 inline static void WDT_WatchdogtimerSoftwareDisable(void)
113 {
114     RCONbits.SWDTEN = 0;
115 }
116
117 /**
118  * Clears the Watch Dog Timer (WDT).
119  * @example
120  * <code>
121  * WDT_WatchdogTimerClear();
122  * </code>
123  */
124 __attribute__((deprecated ("\nThis will be removed in future MCC releases. \
        nUse WATCHDOG_TimerClear (void) instead. ")))
125 inline static void WDT_WatchdogTimerClear(void)
126 {
127     ClrWdt();
128 }
129
130 /**
131  * Gets the base address of the DEVID register for the currently selected
        device
132  * @return base address of the DEVID register
133  * @example
134  * <code>
135  * uint32_t devIdAddress;
136  * devIdAddress = DEVICE_DeviceIdRegisterAddressGet();
137  * </code>
138  */
139 __attribute__((deprecated ("\nThis will be removed in future MCC releases. \
        nUse SYSTEM_DeviceIdRegisterAddressGet (void) instead. ")))
140 inline static uint32_t DEVICE_DeviceIdRegisterAddressGet(void)
141 {
142     return __DEVID_BASE;
143 }
144
145 /**
146  * Initializes the CPU core control register.
147  * @example
148  * <code>
149  * CORCON_Initialize();
150  * </code>
151  */
152 __attribute__((deprecated ("\nThis will be removed in future MCC releases. \
        nUse SYSTEM_CORCONInitialize() instead. ")))
153 inline static void CORCON_Initialize()
```

```
154 {
155     CORCON = (CORCON & 0x00F2) | CORCON_MODE_PORVALUES;     // POR value
156 }
157
158 /**
159  * Sets the CPU core control register operating mode to a value that is
        decided by the
160  * SYSTEM_CORCON_MODES argument.
161  * @param modeValue SYSTEM_CORCON_MODES initialization mode specifier
162  * @example
163  * <code>
164  * CORCON_ModeOperatingSet(CORCON_MODE_ENABLEALLSATNORMAL_ROUNDUNBIASED);
165  * </code>
166  */
167 __attribute__((deprecated ("\nThis will be removed in future MCC releases. \
        nUse SYSTEM_CORCONModeOperatingSet(SYSTEM_CORCON_MODES modeValue) instead
        . ")))
168 inline static void CORCON_ModeOperatingSet(SYSTEM_CORCON_MODES modeValue)
169 {
170     CORCON = (CORCON & 0x00F2) | modeValue;
171 }
172
173 /**
174  * Sets the value of CPU core control register.
175  * @param value value that needs to be written to the CPU core control
        register
176  * @example
177  * <code>
178  * CORCON_RegisterValueSet(0x00E2);
179  * </code>
180  */
181 __attribute__((deprecated ("\nThis will be removed in future MCC releases. \
        nUse SYSTEM_CORCONRegisterValueSet(uint16_t value) instead. ")))
182 inline static void CORCON_RegisterValueSet(uint16_t value)
183 {
184     CORCON = value;
185 }
186
187 /**
188  * Gets the value of CPU core control register.
189  * @return value of the CPU core control register
190  * @example
191  * <code>
192  * corconSave = CORCON_RegisterValueGet();
193  * </code>
194  */
195 __attribute__((deprecated ("\nThis will be removed in future MCC releases. \
        nUse SYSTEM_CORCONRegisterValueGet (void) instead. ")))
196 inline static uint16_t CORCON_RegisterValueGet(void)
197 {
198     return CORCON;
199 }
200
201 /**
202  * It handles the reset cause by clearing the cause register values.
203  * Its a weak function user can override this function.
204  * @return None
```

```
205  * @example
206  * <code>
207  * SYSTEM_ResetCauseHandler();
208  * </code>
209  */
210 void __attribute__ ((weak)) SYSTEM_ResetCauseHandler(void) __attribute__((
        deprecated ("\nThis will be removed in future MCC releases. \nUse
        RESET_CauseHandler(void) (void) instead. ")));
211
212 /**
213  * This function resets the reset cause register.
214  * @return None
215  * @example
216  * <code>
217  * SYSTEM_ResetCauseClearAll();
218  * </code>
219  */
220 void SYSTEM_ResetCauseClearAll() __attribute__((deprecated ("\nThis will be
        removed in future MCC releases. \nUse RESET_CauseClearAll(void) (void)
        instead. ")));
221
222 #endif  /* MCC_H */
223 /**
224  End of File
225 */
```

mcc.c file

```
/**
  @Generated PIC24 / dsPIC33 / PIC32MM MCUs Source File

  @Company:
    Microchip Technology Inc.

  @File Name:
    mcc.c

  @Summary:
    This is the mcc.c file generated using PIC24 / dsPIC33 / PIC32MM MCUs

  @Description:
    This header file provides implementations for driver APIs for all modules selected in the GUI.
    Generation Information :
        Product Revision  :  PIC24 / dsPIC33 / PIC32MM MCUs - pic24-dspic-pic32mm : 1.75.1
        Device            :  dsPIC33EV256GM102
    The generated drivers are tested against the following:
        Compiler          :  XC16 v1.35
        MPLAB             :  MPLAB X v5.05
*/

/*
    (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS, COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF THESE
    TERMS.
*/

// Configuration bits: selected in the GUI

// FSEC
#pragma config BWRP = OFF     // Boot Segment Write-Protect Bit->Boot Segment may be written
```

```
49 #pragma config BSS = DISABLED     // Boot Segment Code-Protect Level bits->No
       Protection (other than BWRP)
50 #pragma config BSS2 = OFF     // Boot Segment Control Bit->No Boot Segment
51 #pragma config GWRP = OFF     // General Segment Write-Protect Bit->General
       Segment may be written
52 #pragma config GSS = DISABLED     // General Segment Code-Protect Level bits->
       No Protection (other than GWRP)
53 #pragma config CWRP = OFF     // Configuration Segment Write-Protect Bit->
       Configuration Segment may be written
54 #pragma config CSS = DISABLED     // Configuration Segment Code-Protect Level
       bits->No Protection (other than CWRP)
55 #pragma config AIVTDIS = DISABLE     // Alternate Interrupt Vector Table
       Disable Bit->Disable Alternate Vector Table
56
57 // FBSLIM
58 #pragma config BSLIM = 8191     // Boot Segment Code Flash Page Address Limit
       Bits->
59
60 // FOSCSEL
61 #pragma config FNOSC = FRC     // Initial oscillator Source Selection Bits->FRC
62 #pragma config IESO = ON     // Two Speed Oscillator Start-Up Bit->Start up
       device with FRC, then automatically switch to user selected oscillator
       source
63
64 // FOSC
65 #pragma config POSCMD = NONE     // Primary Oscillator Mode Select Bits->
       Primary Oscillator disabled
66 #pragma config OSCIOFNC = ON     // OSC2 Pin I/O Function Enable Bit->OSC2 is
       general purpose digital I/O pin
67 #pragma config IOL1WAY = ON     // Peripheral Pin Select Configuration Bit->
       Allow Only One reconfiguration
68 #pragma config FCKSM = CSECMD     // Clock Switching Mode Bits->Clock Switching
        is enabled, Fail-safe Clock Monitor is disabled
69 #pragma config PLLKEN = ON     // PLL Lock Enable Bit->Clock switch to PLL
       source will wait until the PLL lock signal is valid
70
71 // FWDT
72 #pragma config WDTPOST = PS32768     // Watchdog Timer Postscaler Bits->1:32768
73 #pragma config WDTPRE = PR128     // Watchdog Timer Prescaler Bit->1:128
74 #pragma config FWDTEN = OFF     // Watchdog Timer Enable Bits->WDT and SWDTEN
       Disabled
75 #pragma config WINDIS = OFF     // Watchdog Timer Window Enable Bit->Watchdog
       timer in Non-Window Mode
76 #pragma config WDTWIN = WIN25     // Watchdog Window Select Bits->WDT Window is
        25% of WDT period
77
78 // FPOR
79 #pragma config BOREN0 = ON     // Brown Out Reset Detection Bit->BOR is Enabled
80
81 // FICD
82 #pragma config ICS = PGD1     // ICD Communication Channel Select Bits->
       Communicate on PGEC1 and PGED1
83
84 // FDMTINTVL
85 #pragma config DMTIVTL = 0     // Lower 16 Bits of 32 Bit DMT Window Interval->
86
87 // FDMTINTVH
```

```
88 #pragma config DMTIVTH = 0    // Upper 16 Bits of 32 Bit DMT Window Interval->
89
90 // FDMTCNTL
91 #pragma config DMTCNTL = 0    // Lower 16 Bits of 32 Bit DMT Instruction Count
       Time-Out Value->
92
93 // FDMTCNTH
94 #pragma config DMTCNTH = 0    // Upper 16 Bits of 32 Bit DMT Instruction Count
       Time-Out Value->
95
96 // FDMT
97 #pragma config DMTEN = DISABLE    // Dead Man Timer Enable Bit->Dead Man Timer
       is Disabled and can be enabled by software
98
99 // FDEVOPT
100 #pragma config PWMLOCK = ON    // PWM Lock Enable Bit->Certain PWM registers
      may only be written after key sequence
101 #pragma config ALTI2C1 = OFF    // Alternate I2C1 Pins Selection Bit->I2C1
      mapped to SDA1/SCL1 pins
102
103 // FALTREG
104 #pragma config CTXT1 = NONE    // Interrupt Priority Level (IPL) Selection
      Bits For Alternate Working Register Set 1->Not Assigned
105 #pragma config CTXT2 = NONE    // Interrupt Priority Level (IPL) Selection
      Bits For Alternate Working Register Set 2->Not Assigned
106
107 #include "reset.h"
108 #include "mcc.h"
109
110 /**
111  Section: Local Variables
112 */
113
114 /**
115  Section: Function prototypes
116 */
117
118 /**
119 * a private place to store the error code if we run into a severe error
120 */
121
122 void OSCILLATOR_Initialize(void)
123 {
124     CLOCK_Initialize();
125 }
126
127 uint16_t SYSTEM_GetResetCause(void)
128 {
129     return RESET_GetCause();
130 }
131
132 void __attribute__ ((weak)) SYSTEM_ResetCauseHandler(void)
133 {
134     RESET_CauseHandler();
135 }
136
137 void SYSTEM_ResetCauseClearAll()
```

```
138 {
139     RESET_CauseClearAll();
140 }
141 /**
142  End of File
143 */
```

pin_manager.h file

```
/**
  System Interrupts Generated Driver File

  @Company:
    Microchip Technology Inc.

  @File Name:
    pin_manager.h

  @Summary:
    This is the generated manager file for the MPLAB(c) Code Configurator device.  This manager
    configures the pins direction, initial state, analog setting.
    The peripheral pin select, PPS, configuration is also handled by this manager.

  @Description:
    This source file provides implementations for MPLAB(c) Code Configurator interrupts.
    Generation Information :
        Product Revision  :  MPLAB(c) Code Configurator - pic24-dspic-pic32mm : 1.75.1
        Device            :  dsPIC33EV256GM102
    The generated drivers are tested against the following:
        Compiler          :  XC16 v1.35
        MPLAB             :  MPLAB X v5.05

    Copyright (c) 2013 - 2015 released Microchip Technology Inc.  All rights reserved.

    Microchip licenses to you the right to use, modify, copy and distribute
    Software only when embedded on a Microchip microcontroller or digital signal
    controller that is integrated into your product or third party product
    (pursuant to the sublicense terms in the accompanying license agreement).

    You should refer to the license agreement accompanying this Software for additional information regarding your rights and obligations.

    SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND,
    EITHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF
    MERCHANTABILITY, TITLE, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE.
    IN NO EVENT SHALL MICROCHIP OR ITS LICENSORS BE LIABLE OR OBLIGATED UNDER
    CONTRACT, NEGLIGENCE, STRICT LIABILITY, CONTRIBUTION, BREACH OF WARRANTY, OR
    OTHER LEGAL EQUITABLE THEORY ANY DIRECT OR INDIRECT DAMAGES OR EXPENSES
    INCLUDING BUT NOT LIMITED TO ANY INCIDENTAL, SPECIAL, INDIRECT, PUNITIVE OR
    CONSEQUENTIAL DAMAGES, LOST PROFITS OR LOST DATA, COST OF PROCUREMENT OF
    SUBSTITUTE GOODS, TECHNOLOGY, SERVICES, OR ANY CLAIMS BY THIRD PARTIES
    (INCLUDING BUT NOT LIMITED TO ANY DEFENSE THEREOF), OR OTHER SIMILAR COSTS.

*/
```

```
46
47 #ifndef _PIN_MANAGER_H
48 #define _PIN_MANAGER_H
49 /**
50     Section: Includes
51 */
52 #include <xc.h>
53 /**
54     Section: Device Pin Macros
55 */
56 /**
57   @Summary
58     Sets the GPIO pin, RA0, high using LATA0.
59
60   @Description
61     Sets the GPIO pin, RA0, high using LATA0.
62
63   @Preconditions
64     The RA0 must be set to an output.
65
66   @Returns
67     None.
68
69   @Param
70     None.
71
72   @Example
73     <code>
74     // Set RA0 high (1)
75     Stat1_SetHigh();
76     </code>
77
78 */
79 #define Stat1_SetHigh()          _LATA0 = 1
80 /**
81   @Summary
82     Sets the GPIO pin, RA0, low using LATA0.
83
84   @Description
85     Sets the GPIO pin, RA0, low using LATA0.
86
87   @Preconditions
88     The RA0 must be set to an output.
89
90   @Returns
91     None.
92
93   @Param
94     None.
95
96   @Example
97     <code>
98     // Set RA0 low (0)
99     Stat1_SetLow();
100    </code>
101
102 */
```

```
103 #define Stat1_SetLow()           _LATA0 = 0
104 /**
105   @Summary
106     Toggles the GPIO pin, RA0, using LATA0.
107
108   @Description
109     Toggles the GPIO pin, RA0, using LATA0.
110
111   @Preconditions
112     The RA0 must be set to an output.
113
114   @Returns
115     None.
116
117   @Param
118     None.
119
120   @Example
121     <code>
122     // Toggle RA0
123     Stat1_Toggle();
124     </code>
125
126 */
127 #define Stat1_Toggle()           _LATA0 ^= 1
128 /**
129   @Summary
130     Reads the value of the GPIO pin, RA0.
131
132   @Description
133     Reads the value of the GPIO pin, RA0.
134
135   @Preconditions
136     None.
137
138   @Returns
139     None.
140
141   @Param
142     None.
143
144   @Example
145     <code>
146     uint16_t portValue;
147
148     // Read RA0
149     postValue = Stat1_GetValue();
150     </code>
151
152 */
153 #define Stat1_GetValue()         _RA0
154 /**
155   @Summary
156     Configures the GPIO pin, RA0, as an input.
157
158   @Description
159     Configures the GPIO pin, RA0, as an input.
```

```
160
161   @Preconditions
162     None.
163
164   @Returns
165     None.
166
167   @Param
168     None.
169
170   @Example
171     <code>
172     // Sets the RA0 as an input
173     Stat1_SetDigitalInput();
174     </code>
175
176 */
177 #define Stat1_SetDigitalInput()   _TRISA0 = 1
178 /**
179   @Summary
180     Configures the GPIO pin, RA0, as an output.
181
182   @Description
183     Configures the GPIO pin, RA0, as an output.
184
185   @Preconditions
186     None.
187
188   @Returns
189     None.
190
191   @Param
192     None.
193
194   @Example
195     <code>
196     // Sets the RA0 as an output
197     Stat1_SetDigitalOutput();
198     </code>
199
200 */
201 #define Stat1_SetDigitalOutput() _TRISA0 = 0
202 /**
203   @Summary
204     Sets the GPIO pin, RA1, high using LATA1.
205
206   @Description
207     Sets the GPIO pin, RA1, high using LATA1.
208
209   @Preconditions
210     The RA1 must be set to an output.
211
212   @Returns
213     None.
214
215   @Param
216     None.
```

```
217
218   @Example
219     <code>
220     // Set RA1 high (1)
221     Stat2_SetHigh();
222     </code>
223
224 */
225 #define Stat2_SetHigh()           _LATA1 = 1
226 /**
227   @Summary
228     Sets the GPIO pin, RA1, low using LATA1.
229
230   @Description
231     Sets the GPIO pin, RA1, low using LATA1.
232
233   @Preconditions
234     The RA1 must be set to an output.
235
236   @Returns
237     None.
238
239   @Param
240     None.
241
242   @Example
243     <code>
244     // Set RA1 low (0)
245     Stat2_SetLow();
246     </code>
247
248 */
249 #define Stat2_SetLow()            _LATA1 = 0
250 /**
251   @Summary
252     Toggles the GPIO pin, RA1, using LATA1.
253
254   @Description
255     Toggles the GPIO pin, RA1, using LATA1.
256
257   @Preconditions
258     The RA1 must be set to an output.
259
260   @Returns
261     None.
262
263   @Param
264     None.
265
266   @Example
267     <code>
268     // Toggle RA1
269     Stat2_Toggle();
270     </code>
271
272 */
273 #define Stat2_Toggle()            _LATA1 ^= 1
```

```
/**
  @Summary
    Reads the value of the GPIO pin, RA1.

  @Description
    Reads the value of the GPIO pin, RA1.

  @Preconditions
    None.

  @Returns
    None.

  @Param
    None.

  @Example
    <code>
    uint16_t portValue;

    // Read RA1
    postValue = Stat2_GetValue();
    </code>

*/
#define Stat2_GetValue()           _RA1
/**
  @Summary
    Configures the GPIO pin, RA1, as an input.

  @Description
    Configures the GPIO pin, RA1, as an input.

  @Preconditions
    None.

  @Returns
    None.

  @Param
    None.

  @Example
    <code>
    // Sets the RA1 as an input
    Stat2_SetDigitalInput();
    </code>

*/
#define Stat2_SetDigitalInput()   _TRISA1 = 1
/**
  @Summary
    Configures the GPIO pin, RA1, as an output.

  @Description
    Configures the GPIO pin, RA1, as an output.

```

```
331   @Preconditions
332     None.
333
334   @Returns
335     None.
336
337   @Param
338     None.
339
340   @Example
341     <code>
342     // Sets the RA1 as an output
343     Stat2_SetDigitalOutput();
344     </code>
345
346 */
347 #define Stat2_SetDigitalOutput() _TRISA1 = 0
348 /**
349   @Summary
350     Sets the GPIO pin, RB0, high using LATB0.
351
352   @Description
353     Sets the GPIO pin, RB0, high using LATB0.
354
355   @Preconditions
356     The RB0 must be set to an output.
357
358   @Returns
359     None.
360
361   @Param
362     None.
363
364   @Example
365     <code>
366     // Set RB0 high (1)
367     Stat3_SetHigh();
368     </code>
369
370 */
371 #define Stat3_SetHigh()          _LATB0 = 1
372 /**
373   @Summary
374     Sets the GPIO pin, RB0, low using LATB0.
375
376   @Description
377     Sets the GPIO pin, RB0, low using LATB0.
378
379   @Preconditions
380     The RB0 must be set to an output.
381
382   @Returns
383     None.
384
385   @Param
386     None.
387
```

```
388   @Example
389     <code>
390     // Set RB0 low (0)
391     Stat3_SetLow();
392     </code>
393
394 */
395 #define Stat3_SetLow()            _LATB0 = 0
396 /**
397   @Summary
398     Toggles the GPIO pin, RB0, using LATB0.
399
400   @Description
401     Toggles the GPIO pin, RB0, using LATB0.
402
403   @Preconditions
404     The RB0 must be set to an output.
405
406   @Returns
407     None.
408
409   @Param
410     None.
411
412   @Example
413     <code>
414     // Toggle RB0
415     Stat3_Toggle();
416     </code>
417
418 */
419 #define Stat3_Toggle()            _LATB0 ^= 1
420 /**
421   @Summary
422     Reads the value of the GPIO pin, RB0.
423
424   @Description
425     Reads the value of the GPIO pin, RB0.
426
427   @Preconditions
428     None.
429
430   @Returns
431     None.
432
433   @Param
434     None.
435
436   @Example
437     <code>
438     uint16_t portValue;
439
440     // Read RB0
441     postValue = Stat3_GetValue();
442     </code>
443
444 */
```

```
445 #define Stat3_GetValue()          _RB0
446 /**
447   @Summary
448     Configures the GPIO pin, RB0, as an input.
449
450   @Description
451     Configures the GPIO pin, RB0, as an input.
452
453   @Preconditions
454     None.
455
456   @Returns
457     None.
458
459   @Param
460     None.
461
462   @Example
463     <code>
464     // Sets the RB0 as an input
465     Stat3_SetDigitalInput();
466     </code>
467
468 */
469 #define Stat3_SetDigitalInput()   _TRISB0 = 1
470 /**
471   @Summary
472     Configures the GPIO pin, RB0, as an output.
473
474   @Description
475     Configures the GPIO pin, RB0, as an output.
476
477   @Preconditions
478     None.
479
480   @Returns
481     None.
482
483   @Param
484     None.
485
486   @Example
487     <code>
488     // Sets the RB0 as an output
489     Stat3_SetDigitalOutput();
490     </code>
491
492 */
493 #define Stat3_SetDigitalOutput()  _TRISB0 = 0
494 /**
495   @Summary
496     Sets the GPIO pin, RB7, high using LATB7.
497
498   @Description
499     Sets the GPIO pin, RB7, high using LATB7.
500
501   @Preconditions
```

```
502     The RB7 must be set to an output.
503
504   @Returns
505     None.
506
507   @Param
508     None.
509
510   @Example
511     <code>
512     // Set RB7 high (1)
513     Hall1_SetHigh();
514     </code>
515
516 */
517 #define Hall1_SetHigh()           _LATB7 = 1
518 /**
519   @Summary
520     Sets the GPIO pin, RB7, low using LATB7.
521
522   @Description
523     Sets the GPIO pin, RB7, low using LATB7.
524
525   @Preconditions
526     The RB7 must be set to an output.
527
528   @Returns
529     None.
530
531   @Param
532     None.
533
534   @Example
535     <code>
536     // Set RB7 low (0)
537     Hall1_SetLow();
538     </code>
539
540 */
541 #define Hall1_SetLow()            _LATB7 = 0
542 /**
543   @Summary
544     Toggles the GPIO pin, RB7, using LATB7.
545
546   @Description
547     Toggles the GPIO pin, RB7, using LATB7.
548
549   @Preconditions
550     The RB7 must be set to an output.
551
552   @Returns
553     None.
554
555   @Param
556     None.
557
558   @Example
```

```
559     <code>
560     // Toggle RB7
561     Hall1_Toggle();
562     </code>
563 
564 */
565 #define Hall1_Toggle()           _LATB7 ^= 1
566 /**
567   @Summary
568     Reads the value of the GPIO pin, RB7.
569 
570   @Description
571     Reads the value of the GPIO pin, RB7.
572 
573   @Preconditions
574     None.
575 
576   @Returns
577     None.
578 
579   @Param
580     None.
581 
582   @Example
583     <code>
584     uint16_t portValue;
585 
586     // Read RB7
587     postValue = Hall1_GetValue();
588     </code>
589 
590 */
591 #define Hall1_GetValue()         _RB7
592 /**
593   @Summary
594     Configures the GPIO pin, RB7, as an input.
595 
596   @Description
597     Configures the GPIO pin, RB7, as an input.
598 
599   @Preconditions
600     None.
601 
602   @Returns
603     None.
604 
605   @Param
606     None.
607 
608   @Example
609     <code>
610     // Sets the RB7 as an input
611     Hall1_SetDigitalInput();
612     </code>
613 
614 */
615 #define Hall1_SetDigitalInput()  _TRISB7 = 1
```

```
616 /**
617   @Summary
618     Configures the GPIO pin, RB7, as an output.
619
620   @Description
621     Configures the GPIO pin, RB7, as an output.
622
623   @Preconditions
624     None.
625
626   @Returns
627     None.
628
629   @Param
630     None.
631
632   @Example
633     <code>
634     // Sets the RB7 as an output
635     Hall1_SetDigitalOutput();
636     </code>
637
638 */
639 #define Hall1_SetDigitalOutput() _TRISB7 = 0
640 /**
641   @Summary
642     Sets the GPIO pin, RB8, high using LATB8.
643
644   @Description
645     Sets the GPIO pin, RB8, high using LATB8.
646
647   @Preconditions
648     The RB8 must be set to an output.
649
650   @Returns
651     None.
652
653   @Param
654     None.
655
656   @Example
657     <code>
658     // Set RB8 high (1)
659     Hall2_SetHigh();
660     </code>
661
662 */
663 #define Hall2_SetHigh()          _LATB8 = 1
664 /**
665   @Summary
666     Sets the GPIO pin, RB8, low using LATB8.
667
668   @Description
669     Sets the GPIO pin, RB8, low using LATB8.
670
671   @Preconditions
672     The RB8 must be set to an output.
```

```
673
674   @Returns
675     None.
676
677   @Param
678     None.
679
680   @Example
681     <code>
682     // Set RB8 low (0)
683     Hall2_SetLow();
684     </code>
685
686 */
687 #define Hall2_SetLow()           _LATB8 = 0
688 /**
689   @Summary
690     Toggles the GPIO pin, RB8, using LATB8.
691
692   @Description
693     Toggles the GPIO pin, RB8, using LATB8.
694
695   @Preconditions
696     The RB8 must be set to an output.
697
698   @Returns
699     None.
700
701   @Param
702     None.
703
704   @Example
705     <code>
706     // Toggle RB8
707     Hall2_Toggle();
708     </code>
709
710 */
711 #define Hall2_Toggle()           _LATB8 ^= 1
712 /**
713   @Summary
714     Reads the value of the GPIO pin, RB8.
715
716   @Description
717     Reads the value of the GPIO pin, RB8.
718
719   @Preconditions
720     None.
721
722   @Returns
723     None.
724
725   @Param
726     None.
727
728   @Example
729     <code>
```

```
730     uint16_t portValue;
731
732     // Read RB8
733     postValue = Hall2_GetValue();
734     </code>
735
736 */
737 #define Hall2_GetValue()          _RB8
738 /**
739   @Summary
740     Configures the GPIO pin, RB8, as an input.
741
742   @Description
743     Configures the GPIO pin, RB8, as an input.
744
745   @Preconditions
746     None.
747
748   @Returns
749     None.
750
751   @Param
752     None.
753
754   @Example
755     <code>
756     // Sets the RB8 as an input
757     Hall2_SetDigitalInput();
758     </code>
759
760 */
761 #define Hall2_SetDigitalInput()   _TRISB8 = 1
762 /**
763   @Summary
764     Configures the GPIO pin, RB8, as an output.
765
766   @Description
767     Configures the GPIO pin, RB8, as an output.
768
769   @Preconditions
770     None.
771
772   @Returns
773     None.
774
775   @Param
776     None.
777
778   @Example
779     <code>
780     // Sets the RB8 as an output
781     Hall2_SetDigitalOutput();
782     </code>
783
784 */
785 #define Hall2_SetDigitalOutput() _TRISB8 = 0
786 /**
```

```
787   @Summary
788     Sets the GPIO pin, RB9, high using LATB9.
789
790   @Description
791     Sets the GPIO pin, RB9, high using LATB9.
792
793   @Preconditions
794     The RB9 must be set to an output.
795
796   @Returns
797     None.
798
799   @Param
800     None.
801
802   @Example
803     <code>
804     // Set RB9 high (1)
805     Hall3_SetHigh();
806     </code>
807
808 */
809 #define Hall3_SetHigh()           _LATB9 = 1
810 /**
811   @Summary
812     Sets the GPIO pin, RB9, low using LATB9.
813
814   @Description
815     Sets the GPIO pin, RB9, low using LATB9.
816
817   @Preconditions
818     The RB9 must be set to an output.
819
820   @Returns
821     None.
822
823   @Param
824     None.
825
826   @Example
827     <code>
828     // Set RB9 low (0)
829     Hall3_SetLow();
830     </code>
831
832 */
833 #define Hall3_SetLow()            _LATB9 = 0
834 /**
835   @Summary
836     Toggles the GPIO pin, RB9, using LATB9.
837
838   @Description
839     Toggles the GPIO pin, RB9, using LATB9.
840
841   @Preconditions
842     The RB9 must be set to an output.
843
```

```
844   @Returns
845     None.
846
847   @Param
848     None.
849
850   @Example
851     <code>
852     // Toggle RB9
853     Hall3_Toggle();
854     </code>
855
856 */
857 #define Hall3_Toggle()           _LATB9 ^= 1
858 /**
859   @Summary
860     Reads the value of the GPIO pin, RB9.
861
862   @Description
863     Reads the value of the GPIO pin, RB9.
864
865   @Preconditions
866     None.
867
868   @Returns
869     None.
870
871   @Param
872     None.
873
874   @Example
875     <code>
876     uint16_t portValue;
877
878     // Read RB9
879     postValue = Hall3_GetValue();
880     </code>
881
882 */
883 #define Hall3_GetValue()         _RB9
884 /**
885   @Summary
886     Configures the GPIO pin, RB9, as an input.
887
888   @Description
889     Configures the GPIO pin, RB9, as an input.
890
891   @Preconditions
892     None.
893
894   @Returns
895     None.
896
897   @Param
898     None.
899
900   @Example
```

```
901     <code>
902     // Sets the RB9 as an input
903     Hall3_SetDigitalInput();
904     </code>
905
906 */
907 #define Hall3_SetDigitalInput()   _TRISB9 = 1
908 /**
909   @Summary
910     Configures the GPIO pin, RB9, as an output.
911
912   @Description
913     Configures the GPIO pin, RB9, as an output.
914
915   @Preconditions
916     None.
917
918   @Returns
919     None.
920
921   @Param
922     None.
923
924   @Example
925     <code>
926     // Sets the RB9 as an output
927     Hall3_SetDigitalOutput();
928     </code>
929
930 */
931 #define Hall3_SetDigitalOutput()  _TRISB9 = 0
932
933 /**
934     Section: Function Prototypes
935 */
936 /**
937   @Summary
938     Configures the pin settings of the dsPIC33EV256GM102
939     The peripheral pin select, PPS, configuration is also handled by this
      manager.
940
941   @Description
942     This is the generated manager file for the MPLAB(c) Code Configurator
      device.  This manager
943     configures the pins direction, initial state, analog setting.
944     The peripheral pin select, PPS, configuration is also handled by this
      manager.
945
946   @Preconditions
947     None.
948
949   @Returns
950     None.
951
952   @Param
953     None.
954
```

```
955   @Example
956     <code>
957     void SYSTEM_Initialize(void)
958     {
959         // Other initializers are called from this function
960         PIN_MANAGER_Initialize();
961     }
962     </code>
963
964 */
965 void PIN_MANAGER_Initialize(void);
966
967 #endif
```

pin_manager.c file

```
/**
  System Interrupts Generated Driver File

  @Company:
    Microchip Technology Inc.

  @File Name:
    pin_manager.c

  @Summary:
    This is the generated manager file for the MPLAB(c) Code Configurator device.  This manager
    configures the pins direction, initial state, analog setting.
    The peripheral pin select, PPS, configuration is also handled by this manager.

  @Description:
    This source file provides implementations for MPLAB(c) Code Configurator interrupts.
    Generation Information :
        Product Revision  :  MPLAB(c) Code Configurator - pic24-dspic-pic32mm : 1.75.1
        Device            :  dsPIC33EV256GM102
    The generated drivers are tested against the following:
        Compiler          :  XC16 v1.35
        MPLAB             :  MPLAB X v5.05

    Copyright (c) 2013 - 2015 released Microchip Technology Inc.  All rights reserved.

    Microchip licenses to you the right to use, modify, copy and distribute
    Software only when embedded on a Microchip microcontroller or digital signal
    controller that is integrated into your product or third party product
    (pursuant to the sublicense terms in the accompanying license agreement).

    You should refer to the license agreement accompanying this Software for additional information regarding your rights and obligations.

    SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND,
    EITHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF
    MERCHANTABILITY, TITLE, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE.
    IN NO EVENT SHALL MICROCHIP OR ITS LICENSORS BE LIABLE OR OBLIGATED UNDER
    CONTRACT, NEGLIGENCE, STRICT LIABILITY, CONTRIBUTION, BREACH OF WARRANTY, OR
    OTHER LEGAL EQUITABLE THEORY ANY DIRECT OR INDIRECT DAMAGES OR EXPENSES
    INCLUDING BUT NOT LIMITED TO ANY INCIDENTAL, SPECIAL, INDIRECT, PUNITIVE OR
    CONSEQUENTIAL DAMAGES, LOST PROFITS OR LOST DATA, COST OF PROCUREMENT OF
    SUBSTITUTE GOODS, TECHNOLOGY, SERVICES, OR ANY CLAIMS BY THIRD PARTIES
    (INCLUDING BUT NOT LIMITED TO ANY DEFENSE THEREOF), OR OTHER SIMILAR COSTS.

*/
```

```
46
47
48  /**
49      Section: Includes
50  */
51  #include <xc.h>
52  #include "pin_manager.h"
53
54  /**
55      void PIN_MANAGER_Initialize(void)
56  */
57  void PIN_MANAGER_Initialize(void)
58  {
59      /*
        ****************************************************************************
60       * Setting the Output Latch SFR(s)
61
        ****************************************************************************
        */
62      LATA = 0x0000;
63      LATB = 0x0000;
64
65      /*
        ****************************************************************************
66       * Setting the GPIO Direction SFR(s)
67
        ****************************************************************************
        */
68      TRISA = 0x000C;
69      TRISB = 0x03FE;
70
71      /*
        ****************************************************************************
72       * Setting the Weak Pull Up and Weak Pull Down SFR(s)
73
        ****************************************************************************
        */
74      CNPDA = 0x0000;
75      CNPDB = 0x0000;
76      CNPUA = 0x0000;
77      CNPUB = 0x0000;
78
79      /*
        ****************************************************************************
80       * Setting the Open Drain SFR(s)
81
        ****************************************************************************
        */
82      ODCA = 0x0000;
83      ODCB = 0x0000;
84
85      /*
        ****************************************************************************
```

```
86      * Setting the Analog/Digital Configuration SFR(s)
87
        *********************************************************************
        */
88      ANSELA = 0x0014;
89      ANSELB = 0x0002;
90
91
92      /*
        *********************************************************************
93       * Set the PPS
94
        *********************************************************************
        */
95      __builtin_write_OSCCONL(OSCCON & 0xbf); // unlock PPS
96
97      RPOR0bits.RP20R = 0x000E;     //RA4->ECAN1:C1TX;
98      RPINR26bits.C1RXR = 0x0024;     //RB4->ECAN1:C1RX;
99
100     __builtin_write_OSCCONL(OSCCON | 0x40); // lock   PPS
101
102 }
```

pwm.h file

```
/**
  PWM Generated Driver API Header File

  @Company
    Microchip Technology Inc.

  @File Name
    pwm.h

  @Summary
    This is the generated header file for the PWM driver using PIC24 / dsPIC33
     / PIC32MM MCUs

  @Description
    This header file provides APIs for driver for PWM.
    Generation Information :
        Product Revision  :  PIC24 / dsPIC33 / PIC32MM MCUs - pic24-dspic-
    pic32mm : 1.75.1
        Device            :  dsPIC33EV256GM102
    The generated drivers are tested against the following:
        Compiler          :  XC16 v1.35
        MPLAB             :  MPLAB X v5.05
*/

/*
    (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this
    software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
    EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY
    IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
    PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS,
    COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
    INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
    BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS
    IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF
    ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF
    THESE
    TERMS.
*/

#ifndef _PWM_H
#define _PWM_H

/**
  Section: Included Files
```

```
50 */
51
52 #include <xc.h>
53 #include <stdint.h>
54 #include <stdbool.h>
55 #include <stdlib.h>
56
57 #ifdef __cplusplus  // Provide C++ Compatibility
58
59     extern "C" {
60
61 #endif
62
63 /**
64   Section: Data Types
65 */
66
67 /** PWM Generator Definition
68
69  @Summary
70    Defines the PWM generators available for PWM
71
72  @Description
73    This routine defines the PWM generators that are available for the module
       to use.
74
75  Remarks:
76    None
77  */
78 typedef enum
79 {
80     PWM_GENERATOR_1 =  1,
81     PWM_GENERATOR_2 =  2,
82     PWM_GENERATOR_3 =  3,
83 } PWM_GENERATOR;
84
85 /**
86   Section: Interface Routines
87 */
88
89 /**
90   @Summary
91     Initializes hardware and data for the given instance of the PWM module
92
93   @Description
94     This routine initializes hardware for the instance of the PWM module,
95     using the hardware initialization given data.  It also initializes all
96     necessary internal data.
97
98   @Param
99     None.
100
101  @Returns
102    None
103
104  @Example
105    <code>
```

```
106     bool status;
107     uint16_t masterPeriod,masterDutyCycle;
108     uint16_t postscaler,compareValue;
109 
110     postscaler = 0x1;
111     compareValue = 0x1;
112 
113     masterPeriod = 0xFFFF;
114     masterDutyCycle = 0xFF;
115 
116 
117     PWM_Initialize();
118 
119     PWM_ModuleDisable();
120 
121     PWM_MasterDutyCycleSet(masterDutyCycle);
122     PWM_PrimaryMasterPeriodSet(period);
123 
124     PWM_SpecialEventPrimaryInterruptDisable();
125     PWM_SpecialEventTriggerInterruptFlagClear();
126     PWM_PrimarySyncOutputDisable();
127     PWM_SpecialEventPrimaryPostscalerSet(postscaler);
128     PWM_SpecialEventPrimaryCompareValueSet(compareValue);
129     PWM_PrimarySyncOutputEnable();
130     PWM_SpecialEventPrimaryInterruptEnable();
131 
132     PWM_ModuleEnable();
133     </code>
134 */
135 void PWM_Initialize (void);
136 
137 /**
138   @Summary
139     Enables the PWM module.
140 
141   @Description
142     This routine is used to enable the PWM module.
143 
144   @Param
145     None.
146 
147   @Returns
148     None
149 
150   @Example
151     Refer to the example of PWM_Initialize();
152 */
153 inline static void PWM_ModuleEnable(void)
154 {
155   PTCONbits.PTEN = true;
156 }
157 
158 /**
159   @Summary
160     Disables the PWM module.
161 
162   @Description
```

```
163     This routine is used to disable the PWM module.
164
165   @Param
166     None.
167
168   @Returns
169     None
170
171   @Example
172     Refer to the example of PWM_Initialize();
173 */
174 inline static void PWM_ModuleDisable(void)
175 {
176   PTCONbits.PTEN = false;
177 }
178
179 /**
180   @Summary
181     Used to set the PWM master duty cycle register.
182
183   @Description
184     This routine is used to set the PWM master duty cycle register.
185
186   @Param
187     masterDutyCycle - Master Duty Cyle value.
188
189   @Returns
190     None
191
192   @Example
193     Refer to the example of PWM_Initialize();
194 */
195 inline static void PWM_MasterDutyCycleSet(uint16_t masterDutyCycle)
196 {
197     MDC = masterDutyCycle;
198 }
199
200 /**
201   @Summary
202     Sets the period value for the Primary Master Time Base generator.
203
204   @Description
205     This routine is used to set the period value for the Primary Master Time
        Base generator.
206
207   @Param
208     period - Period value.
209
210   @Returns
211     None
212
213   @Example
214     Refer to the example of PWM_Initialize();
215 */
216 inline static void PWM_PrimaryMasterPeriodSet(uint16_t period)
217 {
218     PTPER = period;
```

```
219 }
220
221 /**
222   @Summary
223     Enables synchronization output from the Primary PWM timebase generator.
224
225   @Description
226     This routine is used to enable synchronization output from the Primary PWM
        timebase generator.
227
228   @Param
229     None.
230
231   @Returns
232     None
233
234   @Example
235     Refer to the example of PWM_Initialize();
236 */
237 inline static void PWM_PrimarySyncOutputEnable(void)
238 {
239   PTCONbits.SYNCOEN = true;
240 }
241
242 /**
243   @Summary
244     Disables synchronization output from the Primary PWM timebase generator.
245
246   @Description
247     This routine is used to disbale synchronization output from the Primary
      PWM timebase generator.
248
249   @Param
250     None.
251
252   @Returns
253     None
254
255   @Example
256     Refer to the example of PWM_Initialize();
257 */
258 inline static void PWM_PrimarySyncOutputDisable(void)
259 {
260   PTCONbits.SYNCOEN = false;
261 }
262
263 /**
264   @Summary
265     Sets the postscaler value for the PWM special event trigger from the
        Primary time base generator.
266
267   @Description
268     This routine is used to set the postscaler value for the PWM special event
        trigger from the Primary time base generator.
269
270   @Param
271     postscaler - Special event postscaler value.
```

```
272
273   @Returns
274     None
275
276   @Example
277     Refer to the example of PWM_Initialize();
278 */
279 inline static void PWM_SpecialEventPrimaryPostscalerSet(uint16_t postscaler)
280 {
281     PTCONbits.SEVTPS = postscaler;
282 }
283
284 /**
285   @Summary
286     Sets the compare value for the Special Event Trigger from the Primary time
       base generator.
287
288   @Description
289     This routine is used to set the compare value for the Special Event
       Trigger from the Primary time base generator.
290
291   @Param
292     compareValue - Compare Value.
293
294   @Returns
295     None
296
297   @Example
298     Refer to the example of PWM_Initialize();
299 */
300 inline static void PWM_SpecialEventPrimaryCompareValueSet(uint16_t
      compareValue)
301 {
302     SEVTCMP = compareValue;
303 }
304
305 /**
306   @Summary
307     Clears PWM Special Event Trigger interrupt request flag
308
309   @Description
310     This routine is used to clear PWM Special Event Trigger interrupt request
       flag
311
312   @Param
313     None.
314
315   @Returns
316     None
317
318   @Example
319     Refer to the example of PWM_Initialize();
320 */
321 inline static void PWM_SpecialEventTriggerInterruptFlagClear(void)
322 {
323     IFS3bits.PSEMIF = false;
324 }
```

```
325
326 /**
327   @Summary
328     Enables interrupt request for Special Event Trigger from the Primary time
        base generator.
329
330   @Description
331     This routine is used to enable interrupt request for Special Event Trigger
         from the Primary time base generator.
332
333   @Param
334     None.
335
336   @Returns
337     None
338
339   @Example
340     Refer to the example of PWM_Initialize();
341 */
342 inline static void PWM_SpecialEventPrimaryInterruptEnable(void)
343 {
344   PTCONbits.SEIEN = true;
345 }
346
347 /**
348   @Summary
349     Disables interrupt request for Special Event Trigger from the Primary time
         base generator.
350
351   @Description
352     This routine is used to disable interrupt request for Special Event
        Trigger from the Primary time base generator.
353
354   @Param
355     None.
356
357   @Returns
358     None
359
360   @Example
361     Refer to the example of PWM_Initialize();
362 */
363 inline static void PWM_SpecialEventPrimaryInterruptDisable(void)
364 {
365   PTCONbits.SEIEN = false;
366 }
367
368 /**
369   @Summary
370     Enables PWM latched fault mode for specific instance.
371
372   @Description
373     This routine is used to enable PWM latched fault mode for specific
        instance.
374
375   @Param
376     genNum - PWM generator instance number.
```

```
377
378   @Returns
379     None
380
381   @Example
382     <code>
383     PWM_GENERATOR genNum;
384
385     genNum = PWM_GENERATOR_1;
386     PWM_FaultModeLatchEnable(genNum);
387     </code>
388 */
389 inline static void PWM_FaultModeLatchEnable(PWM_GENERATOR genNum)
390 {
391     switch(genNum) {
392         case PWM_GENERATOR_1:
393                 __builtin_write_PWMSFR(&FCLCON1, (FCLCON1 & 0xFFFC), &PWMKEY);
394                 break;
395         case PWM_GENERATOR_2:
396                 __builtin_write_PWMSFR(&FCLCON2, (FCLCON2 & 0xFFFC), &PWMKEY);
397                 break;
398         case PWM_GENERATOR_3:
399                 __builtin_write_PWMSFR(&FCLCON3, (FCLCON3 & 0xFFFC), &PWMKEY);
400                 break;
401         default:break;
402     }
403 }
404
405 /**
406   @Summary
407     Disables PWM latched fault mode for specific instance.
408
409   @Description
410     This routine is used to disable PWM latched fault mode for specific
      instance.
411
412   @Param
413     genNum - PWM generator instance number.
414
415   @Returns
416     None
417
418   @Example
419     <code>
420     PWM_GENERATOR genNum;
421
422     genNum = PWM_GENERATOR_1;
423     PWM_FaultModeLatchDisable(genNum);
424     </code>
425 */
426 inline static void PWM_FaultModeLatchDisable(PWM_GENERATOR genNum)
427 {
428     switch(genNum) {
429         case PWM_GENERATOR_1:
430                 __builtin_write_PWMSFR(&FCLCON1, (FCLCON1 | 0x0003), &PWMKEY);
431                 break;
432         case PWM_GENERATOR_2:
```

```
433                 __builtin_write_PWMSFR(&FCLCON2, (FCLCON2 | 0x0003), &PWMKEY);
434                 break;
435         case PWM_GENERATOR_3:
436                 __builtin_write_PWMSFR(&FCLCON3, (FCLCON3 | 0x0003), &PWMKEY);
437                 break;
438         default:break;
439     }
440 }
441
442 /**
443   @Summary
444     Clears PWM interupt request flag for specific instance.
445
446   @Description
447     This routine is used to clear PWM interupt request flag for specific
        instance.
448
449   @Param
450     genNum - PWM generator instance number.
451
452   @Returns
453     None
454
455   @Example
456     <code>
457     PWM_GENERATOR genNum;
458
459     genNum = PWM_GENERATOR_1;
460     PWM_InterruptFlagClear(genNum);
461     </code>
462 */
463 inline static void PWM_InterruptFlagClear(PWM_GENERATOR genNum)
464 {
465     switch(genNum) {
466         case PWM_GENERATOR_1:
467                 IFS5bits.PWM1IF = false;
468                 break;
469         case PWM_GENERATOR_2:
470                 IFS5bits.PWM2IF = false;
471                 break;
472         case PWM_GENERATOR_3:
473                 IFS6bits.PWM3IF = false;
474                 break;
475         default:break;
476     }
477 }
478
479 /**
480   @Summary
481     Clears PWM fault status for specific instance.
482
483   @Description
484     This routine is used to clear PWM fault status for specific instance.
485
486   @Param
487     genNum - PWM generator instance number.
488
```

```
489   @Returns
490     None
491
492   @Example
493     <code>
494     PWM_GENERATOR genNum;
495
496     genNum = PWM_GENERATOR_1;
497     PWM_FaultInterruptStatusClear(genNum);
498     </code>
499 */
500 inline static void PWM_FaultInterruptStatusClear(PWM_GENERATOR genNum)
501 {
502     switch(genNum) {
503         case PWM_GENERATOR_1:
504                 PWMCON1bits.FLTIEN = false;
505                 PWMCON1bits.FLTIEN = true;
506                 break;
507         case PWM_GENERATOR_2:
508                 PWMCON2bits.FLTIEN = false;
509                 PWMCON2bits.FLTIEN = true;
510                 break;
511         case PWM_GENERATOR_3:
512                 PWMCON3bits.FLTIEN = false;
513                 PWMCON3bits.FLTIEN = true;
514                 break;
515         default:break;
516     }
517 }
518
519 /**
520   @Summary
521     Gets PWM fault status for specific instance.
522
523   @Description
524     This routine is used to get PWM fault status for specific instance.
525
526   @Param
527     genNum - PWM generator instance number.
528
529   @Returns
530     Fault interrupt status value.
531
532   @Example
533     <code>
534     bool status;
535     PWM_GENERATOR genNum;
536
537     genNum = PWM_GENERATOR_1;
538     status = PWM_FaultInterruptStatusGet(genNum);
539     </code>
540 */
541 inline static bool PWM_FaultInterruptStatusGet(PWM_GENERATOR genNum)
542 {
543     switch(genNum) {
544         case PWM_GENERATOR_1:
545                 return PWMCON1bits.FLTSTAT;
```

```
546                     break;
547             case PWM_GENERATOR_2:
548                     return PWMCON2bits.FLTSTAT;
549                     break;
550             case PWM_GENERATOR_3:
551                     return PWMCON3bits.FLTSTAT;
552                     break;
553             default:break;
554         }
555 }
556
557 /**
558   @Summary
559     Clears PWM current limit status for specific instance.
560
561   @Description
562     This routine is used to clear PWM current limit status for specific
        instance.
563
564   @Param
565     genNum - PWM generator instance number.
566
567   @Returns
568     None
569
570   @Example
571     <code>
572     PWM_GENERATOR genNum;
573
574     genNum = PWM_GENERATOR_1;
575     PWM_CurrentLimitInterruptStatusClear(genNum);
576     </code>
577 */
578 inline static void PWM_CurrentLimitInterruptStatusClear(PWM_GENERATOR genNum)
579 {
580     switch(genNum) {
581         case PWM_GENERATOR_1:
582                 PWMCON1bits.CLIEN = false;
583                 PWMCON1bits.CLIEN = true;
584                 break;
585         case PWM_GENERATOR_2:
586                 PWMCON2bits.CLIEN = false;
587                 PWMCON2bits.CLIEN = true;
588                 break;
589         case PWM_GENERATOR_3:
590                 PWMCON3bits.CLIEN = false;
591                 PWMCON3bits.CLIEN = true;
592                 break;
593         default:break;
594     }
595 }
596
597 /**
598   @Summary
599     Gets PWM current limit status for specific instance.
600
601   @Description
```

```
602     This routine is used to get PWM current limit status for specific instance
        .
603
604   @Param
605     genNum - PWM generator instance number.
606
607   @Returns
608     Current Limit interrupt status value.
609
610   @Example
611     <code>
612     bool status;
613     PWM_GENERATOR genNum;
614
615     genNum = PWM_GENERATOR_1;
616     status = PWM_CurrentLimitInterruptStatusGet(genNum);
617     </code>
618 */
619 inline static bool PWM_CurrentLimitInterruptStatusGet(PWM_GENERATOR genNum)
620 {
621     switch(genNum) {
622         case PWM_GENERATOR_1:
623                 return PWMCON1bits.CLSTAT;
624                 break;
625         case PWM_GENERATOR_2:
626                 return PWMCON2bits.CLSTAT;
627                 break;
628         case PWM_GENERATOR_3:
629                 return PWMCON3bits.CLSTAT;
630                 break;
631         default:break;
632     }
633 }
634
635 /**
636   @Summary
637     Clears PWM trigger status for specific instance.
638
639   @Description
640     This routine is used to clear PWM trigger status for specific instance.
641
642   @Param
643     genNum - PWM generator instance number.
644
645   @Returns
646     None
647
648   @Example
649     <code>
650     PWM_GENERATOR genNum;
651
652     genNum = PWM_GENERATOR_1;
653     PWM_TriggerInterruptStatusClear(genNum);
654     </code>
655 */
656 inline static void PWM_TriggerInterruptStatusClear(PWM_GENERATOR genNum)
657 {
```

```
658     switch(genNum) {
659         case PWM_GENERATOR_1:
660                 PWMCON1bits.TRGIEN = false;
661                 PWMCON1bits.TRGIEN = true;
662                 break;
663         case PWM_GENERATOR_2:
664                 PWMCON2bits.TRGIEN = false;
665                 PWMCON2bits.TRGIEN = true;
666                 break;
667         case PWM_GENERATOR_3:
668                 PWMCON3bits.TRGIEN = false;
669                 PWMCON3bits.TRGIEN = true;
670                 break;
671         default:break;
672     }
673 }
674
675 /**
676   @Summary
677     Gets PWM trigger status for specific instance.
678
679   @Description
680     This routine is used to get PWM trigger status for specific instance.
681
682   @Param
683     genNum - PWM generator instance number.
684
685   @Returns
686     Trigger interrupt status value.
687
688   @Example
689     <code>
690     bool status;
691     PWM_GENERATOR genNum;
692
693     genNum = PWM_GENERATOR_1;
694     status = PWM_TriggerInterruptStatusGet(genNum);
695     </code>
696 */
697 inline static bool PWM_TriggerInterruptStatusGet(PWM_GENERATOR genNum)
698 {
699     switch(genNum) {
700         case PWM_GENERATOR_1:
701                 return PWMCON1bits.TRGSTAT;
702                 break;
703         case PWM_GENERATOR_2:
704                 return PWMCON2bits.TRGSTAT;
705                 break;
706         case PWM_GENERATOR_3:
707                 return PWMCON3bits.TRGSTAT;
708                 break;
709         default:break;
710     }
711 }
712
713 /**
714   @Summary
```

```
715     Enables PWM override on PWML output for specific instance.
716
717   @Description
718     This routine is used to enable PWM override on PWML output for specific
        instance.
719
720   @Param
721     genNum - PWM generator instance number.
722
723   @Returns
724     None
725
726   @Example
727     <code>
728     PWM_GENERATOR genNum;
729
730     genNum = PWM_GENERATOR_1;
731     PWM_OverrideLowEnable(genNum);
732     </code>
733 */
734 inline static void PWM_OverrideLowEnable(PWM_GENERATOR genNum)
735 {
736     switch(genNum) {
737         case PWM_GENERATOR_1:
738                 __builtin_write_PWMSFR(&IOCON1, (IOCON1 | 0x0100), &PWMKEY);
739                 break;
740         case PWM_GENERATOR_2:
741                 __builtin_write_PWMSFR(&IOCON2, (IOCON2 | 0x0100), &PWMKEY);
742                 break;
743         case PWM_GENERATOR_3:
744                 __builtin_write_PWMSFR(&IOCON3, (IOCON3 | 0x0100), &PWMKEY);
745                 break;
746         default:break;
747     }
748 }
749
750 /**
751   @Summary
752     Disables PWM override on PWML output for specific instance.
753
754   @Description
755     This routine is used to disable PWM override on PWML output for specific
        instance.
756
757   @Param
758     genNum - PWM generator instance number.
759
760   @Returns
761     None
762
763   @Example
764     <code>
765     PWM_GENERATOR genNum;
766
767     genNum = PWM_GENERATOR_1;
768     PWM_OverrideLowDisable(genNum);
769     </code>
```

```
770 */
771 inline static void PWM_OverrideLowDisable(PWM_GENERATOR genNum)
772 {
773     switch(genNum) {
774         case PWM_GENERATOR_1:
775                 __builtin_write_PWMSFR(&IOCON1, (IOCON1 & 0xFEFF), &PWMKEY);
776                 break;
777         case PWM_GENERATOR_2:
778                 __builtin_write_PWMSFR(&IOCON2, (IOCON2 & 0xFEFF), &PWMKEY);
779                 break;
780         case PWM_GENERATOR_3:
781                 __builtin_write_PWMSFR(&IOCON3, (IOCON3 & 0xFEFF), &PWMKEY);
782                 break;
783         default:break;
784     }
785 }
786
787 /**
788   @Summary
789     Enables PWM override on PWMH output for specific instance.
790
791   @Description
792     This routine is used to enable PWM override on PWMH output for specific
      instance.
793
794   @Param
795     genNum - PWM generator instance number.
796
797   @Returns
798     None
799
800   @Example
801     <code>
802     PWM_GENERATOR genNum;
803
804     genNum = PWM_GENERATOR_1;
805     PWM_OverrideHighEnable(genNum);
806     </code>
807 */
808 inline static void PWM_OverrideHighEnable(PWM_GENERATOR genNum)
809 {
810     switch(genNum) {
811         case PWM_GENERATOR_1:
812                 __builtin_write_PWMSFR(&IOCON1, (IOCON1 | 0x0200), &PWMKEY);
813                 break;
814         case PWM_GENERATOR_2:
815                 __builtin_write_PWMSFR(&IOCON2, (IOCON2 | 0x0200), &PWMKEY);
816                 break;
817         case PWM_GENERATOR_3:
818                 __builtin_write_PWMSFR(&IOCON3, (IOCON3 | 0x0200), &PWMKEY);
819                 break;
820         default:break;
821     }
822 }
823
824 /**
825   @Summary
```

```
826     Disables PWM override on PWMH output for specific instance.
827
828   @Description
829     This routine is used to disable PWM override on PWMH output for specific
        instance.
830
831   @Param
832     genNum - PWM generator instance number.
833
834   @Returns
835     None
836
837   @Example
838     <code>
839     PWM_GENERATOR genNum;
840
841     genNum = PWM_GENERATOR_1;
842     PWM_OverrideHighDisable(genNum);
843     </code>
844 */
845 inline static void PWM_OverrideHighDisable(PWM_GENERATOR genNum)
846 {
847     switch(genNum) {
848         case PWM_GENERATOR_1:
849                 __builtin_write_PWMSFR(&IOCON1, (IOCON1 & 0xFDFF), &PWMKEY);
850                 break;
851         case PWM_GENERATOR_2:
852                 __builtin_write_PWMSFR(&IOCON2, (IOCON2 & 0xFDFF), &PWMKEY);
853                 break;
854         case PWM_GENERATOR_3:
855                 __builtin_write_PWMSFR(&IOCON3, (IOCON3 & 0xFDFF), &PWMKEY);
856                 break;
857         default:break;
858     }
859 }
860
861 /**
862   @Summary
863     Updates PWM override data bits with the requested value for a specific
        instance.
864
865   @Description
866     This routine is used to updates PWM override data bits with the requested
        value for a specific instance.
867
868   @Param
869     genNum          - PWM generator instance number.
870     overrideData    - Override data
871
872   @Returns
873     None
874
875   @Example
876     <code>
877     PWM_GENERATOR genNum;
878     uint16_t overrideData;
879
```

```
880     overrideData = 0x01;
881
882     genNum = PWM_GENERATOR_1;
883     PWM_OverrideDataSet(genNum, overrideData);
884     </code>
885 */
886 inline static void PWM_OverrideDataSet(PWM_GENERATOR genNum, uint16_t
        overrideData)
887 {
888     switch(genNum) {
889         case PWM_GENERATOR_1:
890                 overrideData = ((overrideData & 0xFFFC)<<6);
891                 __builtin_write_PWMSFR(&IOCON1, (IOCON1 | overrideData), &
        PWMKEY);
892                 break;
893         case PWM_GENERATOR_2:
894                 overrideData = ((overrideData & 0xFFFC)<<6);
895                 __builtin_write_PWMSFR(&IOCON2, (IOCON2 | overrideData), &
        PWMKEY);
896                 break;
897         case PWM_GENERATOR_3:
898                 overrideData = ((overrideData & 0xFFFC)<<6);
899                 __builtin_write_PWMSFR(&IOCON3, (IOCON3 | overrideData), &
        PWMKEY);
900                 break;
901         default:break;
902     }
903 }
904
905 /**
906   @Summary
907     Sets the PWM duty cycle for specific instance.
908
909   @Description
910     This routine is used to set the PWM duty cycle for specific instance.
911
912   @Param
913     genNum      - PWM generator instance number.
914     dutyCycle   - Duty cycle value
915
916   @Returns
917     None
918
919   @Example
920     <code>
921     PWM_GENERATOR genNum;
922     uint16_t dutyCycle;
923
924     dutyCycle = 0x01;
925
926     genNum = PWM_GENERATOR_1;
927     PWM_DutyCycleSet(genNum, dutyCycle);
928     </code>
929 */
930 inline static void PWM_DutyCycleSet(PWM_GENERATOR genNum, uint16_t dutyCycle)
931 {
932     switch(genNum) {
```

```
933             case PWM_GENERATOR_1:
934                     PDC1 = dutyCycle;
935                     break;
936             case PWM_GENERATOR_2:
937                     PDC2 = dutyCycle;
938                     break;
939             case PWM_GENERATOR_3:
940                     PDC3 = dutyCycle;
941                     break;
942             default:break;
943         }
944 }
945
946 /**
947   @Summary
948     Sets the PWM period value while in Center-Aligned PWM mode for specific
        instance.
949
950   @Description
951     This routine is used to disable interrupt request for Special Event
        Trigger from the Primary time base generator.
952
953   @Param
954     genNum  - PWM generator instance number.
955     period  - Period value
956
957   @Returns
958     None
959
960   @Example
961     <code>
962     PWM_GENERATOR genNum;
963     uint16_t period;
964
965     period = 0x01;
966
967     genNum = PWM_GENERATOR_1;
968     PWM_PeriodCenterAlignedModeSet(genNum, period);
969     </code>
970 */
971 inline static void PWM_PeriodCenterAlignedModeSet(PWM_GENERATOR genNum,
        uint16_t period)
972 {
973     switch(genNum) {
974         case PWM_GENERATOR_1:
975                 PHASE1 = period;
976                 break;
977         case PWM_GENERATOR_2:
978                 PHASE2 = period;
979                 break;
980         case PWM_GENERATOR_3:
981                 PHASE3 = period;
982                 break;
983         default:break;
984     }
985 }
986
```

```
/**
  @Summary
    Sets the PWM deadtime value while in Center-Aligned PWM mode for specific
    instance.

  @Description
    This routine is used to set the PWM deadtime value while in Center-Aligned
     PWM mode for specific instance.

  @Param
    genNum      - PWM generator instance number.
    deadtime    - Dead time value.

  @Returns
    None

  @Example
    <code>
    PWM_GENERATOR genNum;
    uint16_t deadtime;

    deadtime = 0x01;

    genNum = PWM_GENERATOR_1;
    PWM_DeadTimeCenterAlignedModeSet(genNum, deadtime);
    </code>
*/
inline static void PWM_DeadTimeCenterAlignedModeSet(PWM_GENERATOR genNum,
    uint16_t deadtime)
{
    switch(genNum) {
        case PWM_GENERATOR_1:
                ALTDTR1 = deadtime;
                break;
        case PWM_GENERATOR_2:
                ALTDTR2 = deadtime;
                break;
        case PWM_GENERATOR_3:
                ALTDTR3 = deadtime;
                break;
        default:break;
    }
}

/**
  @Summary
    Sets the PWM trigger output divider to the desired value for specific
    instance.

  @Description
    This routine is used to set the PWM trigger output divider to the desired
    value for specific instance.

  @Param
    genNum          - PWM generator instance number.
    trigDivValue    - Trigger value.

```

```
1039   @Returns
1040     None
1041
1042   @Example
1043     <code>
1044     PWM_GENERATOR genNum;
1045     uint16_t trigDivValue;
1046
1047     trigDivValue = 0x01;
1048
1049     genNum = PWM_GENERATOR_1;
1050     PWM_TriggerDividerSet(genNum, trigDivValue);
1051     </code>
1052 */
1053 inline static void PWM_TriggerDividerSet(PWM_GENERATOR genNum, uint16_t
       trigDivValue)
1054 {
1055     switch(genNum) {
1056         case PWM_GENERATOR_1:
1057                 TRGCON1bits.TRGDIV = trigDivValue;
1058                 break;
1059         case PWM_GENERATOR_2:
1060                 TRGCON2bits.TRGDIV = trigDivValue;
1061                 break;
1062         case PWM_GENERATOR_3:
1063                 TRGCON3bits.TRGDIV = trigDivValue;
1064                 break;
1065         default:break;
1066     }
1067 }
1068
1069 /**
1070   @Summary
1071     Sets the PWM trigger start delay to the desired value for specific
       instance.
1072
1073   @Description
1074     This routine is used to set the PWM trigger start delay to the desired
       value for specific instance.
1075
1076   @Param
1077     genNum      - PWM generator instance number.
1078     delayValue  - Trigger start delay value.
1079
1080   @Returns
1081     None
1082
1083   @Example
1084     <code>
1085     PWM_GENERATOR genNum;
1086     uint16_t delayValue;
1087
1088     delayValue = 0x01;
1089
1090     genNum = PWM_GENERATOR_1;
1091     PWM_TriggerStartDelaySet(genNum, delayValue);
1092     </code>
```

```
1093 */
1094 inline static void PWM_TriggerStartDelaySet(PWM_GENERATOR genNum, uint16_t
         delayValue)
1095 {
1096     switch(genNum) {
1097         case PWM_GENERATOR_1:
1098                 TRGCON1bits.TRGSTRT = delayValue;
1099                 break;
1100         case PWM_GENERATOR_2:
1101                 TRGCON2bits.TRGSTRT = delayValue;
1102                 break;
1103         case PWM_GENERATOR_3:
1104                 TRGCON3bits.TRGSTRT = delayValue;
1105                 break;
1106         default:break;
1107     }
1108 }
1109
1110 /**
1111   @Summary
1112     Sets the PWM trigger compare to the desired value for specific instance.
1113
1114   @Description
1115     This routine is used to set the PWM trigger compare to the desired value
          for specific instance.
1116
1117   @Param
1118     genNum          - PWM generator instance number.
1119     trigCompValue   - Trigger compare value.
1120
1121   @Returns
1122     None
1123
1124   @Example
1125     <code>
1126     PWM_GENERATOR genNum;
1127     uint16_t trigCompValue;
1128
1129     trigCompValue = 0x01;
1130
1131     genNum = PWM_GENERATOR_1;
1132     PWM_TriggerCompareValueSet(genNum, trigCompValue);
1133     </code>
1134 */
1135 inline static void PWM_TriggerCompareValueSet(PWM_GENERATOR genNum, uint16_t
         trigCompValue)
1136 {
1137     switch(genNum) {
1138         case PWM_GENERATOR_1:
1139                 TRIG1 = trigCompValue;
1140                 break;
1141         case PWM_GENERATOR_2:
1142                 TRIG2 = trigCompValue;
1143                 break;
1144         case PWM_GENERATOR_3:
1145                 TRIG3 = trigCompValue;
1146                 break;
```

```
1147            default:break;
1148        }
1149    }
1150
1151    /**
1152      @Summary
1153        Enables trigger event interrupt requests from specific PWM instance.
1154
1155      @Description
1156        This routine is used to enable trigger event interrupt requests from
          specific PWM instance.
1157
1158      @Param
1159        genNum - PWM generator instance number.
1160
1161      @Returns
1162        None
1163
1164      @Example
1165        <code>
1166        PWM_GENERATOR genNum;
1167
1168        genNum = PWM_GENERATOR_1;
1169        PWM_TriggerInterruptEnable(genNum);
1170        </code>
1171    */
1172    inline static void PWM_TriggerInterruptEnable(PWM_GENERATOR genNum)
1173    {
1174         switch(genNum) {
1175            case PWM_GENERATOR_1:
1176                    PWMCON1bits.TRGIEN = true;
1177                    break;
1178            case PWM_GENERATOR_2:
1179                    PWMCON2bits.TRGIEN = true;
1180                    break;
1181            case PWM_GENERATOR_3:
1182                    PWMCON3bits.TRGIEN = true;
1183                    break;
1184            default:break;
1185        }
1186    }
1187
1188    /**
1189      @Summary
1190        Disables trigger event interrupt requests from specific PWM instance.
1191
1192      @Description
1193        This routine is used to disable trigger event interrupt requests from
          specific PWM instance.
1194
1195      @Param
1196        genNum - PWM generator instance number.
1197
1198      @Returns
1199        None
1200
1201      @Example
```

```
1202      <code>
1203      PWM_GENERATOR genNum;
1204
1205      genNum = PWM_GENERATOR_1;
1206      PWM_TriggerInterruptDisable(genNum);
1207      </code>
1208  */
1209  inline static void PWM_TriggerInterruptDisable(PWM_GENERATOR genNum)
1210  {
1211       switch(genNum) {
1212          case PWM_GENERATOR_1:
1213                   PWMCON1bits.TRGIEN = false;
1214                   break;
1215          case PWM_GENERATOR_2:
1216                   PWMCON2bits.TRGIEN = false;
1217                   break;
1218          case PWM_GENERATOR_3:
1219                   PWMCON3bits.TRGIEN = false;
1220                   break;
1221          default:break;
1222      }
1223  }
1224
1225  /**
1226    @Summary
1227      Enables current limit interrupt requests from specific PWM instance.
1228
1229    @Description
1230      This routine is used to enable current limit interrupt requests from
         specific PWM instance.
1231
1232    @Param
1233      genNum - PWM generator instance number.
1234
1235    @Returns
1236      None
1237
1238    @Example
1239      <code>
1240      PWM_GENERATOR genNum;
1241
1242      genNum = PWM_GENERATOR_1;
1243      PWM_CurrentLimitInterruptEnable(genNum);
1244      </code>
1245  */
1246  inline static void PWM_CurrentLimitInterruptEnable(PWM_GENERATOR genNum)
1247  {
1248      switch(genNum) {
1249          case PWM_GENERATOR_1:
1250                   PWMCON1bits.CLIEN = true;
1251                   break;
1252          case PWM_GENERATOR_2:
1253                   PWMCON2bits.CLIEN = true;
1254                   break;
1255          case PWM_GENERATOR_3:
1256                   PWMCON3bits.CLIEN = true;
1257                   break;
```

```
1258            default:break;
1259        }
1260    }
1261
1262    /**
1263      @Summary
1264        Disables current limit interrupt requests from specific PWM instance.
1265
1266      @Description
1267        This routine is used to disable current limit interrupt requests from
          specific PWM instance.
1268
1269      @Param
1270        genNum - PWM generator instance number.
1271
1272      @Returns
1273        None
1274
1275      @Example
1276        <code>
1277        PWM_GENERATOR genNum;
1278
1279        genNum = PWM_GENERATOR_1;
1280        PWM_CurrentLimitInterruptDisable(genNum);
1281        </code>
1282    */
1283    inline static void PWM_CurrentLimitInterruptDisable(PWM_GENERATOR genNum)
1284    {
1285         switch(genNum) {
1286            case PWM_GENERATOR_1:
1287                    PWMCON1bits.CLIEN = false;
1288                    break;
1289            case PWM_GENERATOR_2:
1290                    PWMCON2bits.CLIEN = false;
1291                    break;
1292            case PWM_GENERATOR_3:
1293                    PWMCON3bits.CLIEN = false;
1294                    break;
1295            default:break;
1296        }
1297    }
1298
1299    /**
1300      @Summary
1301        Enables fault event interrupt requests from specific PWM instance.
1302
1303      @Description
1304        This routine is used to enable fault event interrupt requests from
          specific PWM instance.
1305
1306      @Param
1307        genNum - PWM generator instance number.
1308
1309      @Returns
1310        None
1311
1312      @Example
```

```
1313      <code>
1314      PWM_GENERATOR genNum;
1315
1316      genNum = PWM_GENERATOR_1;
1317      PWM_FaultInterruptEnable(genNum);
1318      </code>
1319  */
1320  inline static void PWM_FaultInterruptEnable(PWM_GENERATOR genNum)
1321  {
1322      switch(genNum) {
1323          case PWM_GENERATOR_1:
1324                  PWMCON1bits.FLTIEN = true;
1325                  break;
1326          case PWM_GENERATOR_2:
1327                  PWMCON2bits.FLTIEN = true;
1328                  break;
1329          case PWM_GENERATOR_3:
1330                  PWMCON3bits.FLTIEN = true;
1331                  break;
1332          default:break;
1333      }
1334  }
1335
1336  /**
1337    @Summary
1338      Disables fault event interrupt requests from specific PWM instance.
1339
1340    @Description
1341      This routine is used to disable fault event interrupt requests from
         specific PWM instance.
1342
1343    @Param
1344      genNum - PWM generator instance number.
1345
1346    @Returns
1347      None
1348
1349    @Example
1350      <code>
1351      PWM_GENERATOR genNum;
1352
1353      genNum = PWM_GENERATOR_1;
1354      PWM_FaultInterruptDisable(genNum);
1355      </code>
1356  */
1357  inline static void PWM_FaultInterruptDisable(PWM_GENERATOR genNum)
1358  {
1359      switch(genNum) {
1360          case PWM_GENERATOR_1:
1361                  PWMCON1bits.FLTIEN = false;
1362                  break;
1363          case PWM_GENERATOR_2:
1364                  PWMCON2bits.FLTIEN = false;
1365                  break;
1366          case PWM_GENERATOR_3:
1367                  PWMCON3bits.FLTIEN = false;
1368                  break;
```

```
1369            default:break;
1370        }
1371    }
1372
1373    #ifdef __cplusplus  // Provide C++ Compatibility
1374
1375        }
1376
1377    #endif
1378
1379    #endif // _PWM_H
1380
1381    /**
1382     End of File
1383    */
```

pwm.c file

```
/**
  PWM Generated Driver API Source File

  @Company
    Microchip Technology Inc.

  @File Name
    pwm.c

  @Summary
    This is the generated source file for the PWM driver using PIC24 / dsPIC33
     / PIC32MM MCUs

  @Description
    This source file provides APIs for driver for PWM.
    Generation Information :
        Product Revision  :  PIC24 / dsPIC33 / PIC32MM MCUs - pic24-dspic-
    pic32mm : 1.75.1
        Device            :  dsPIC33EV256GM102
    The generated drivers are tested against the following:
        Compiler          :  XC16 v1.35
        MPLAB             :  MPLAB X v5.05
*/

/*
    (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this
    software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
    EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY
    IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
    PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS,
    COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
    INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
    BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS
    IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF
    ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF
    THESE
    TERMS.
*/

/**
  Section: Included Files
*/

#include <xc.h>
```

```
50 #include "pwm.h"
51
52 /**
53   Section: Driver Interface
54 */
55
56
57 void PWM_Initialize (void)
58 {
59     // PCLKDIV 1;
60     PTCON2 = 0x0;
61     // PTPER 1023;
62     PTPER = 0x3FF;
63     // SEVTCMP 0;
64     SEVTCMP = 0x0;
65     // MDC 0;
66     MDC = 0x0;
67     // CHOPCLK 0; CHPCLKEN disabled;
68     CHOP = 0x0;
69     // PWMKEY 0;
70     PWMKEY = 0x0;
71     // MDCS Master; FLTIEN disabled; CAM Edge Aligned; DTC Positive dead time
       for all Output modes; TRGIEN disabled; XPRES disabled; ITB Master; IUE
       disabled; CLIEN disabled; DTCP disabled;
72     PWMCON1 = 0x100;
73     // MDCS Master; FLTIEN disabled; CAM Edge Aligned; DTC Positive dead time
       for all Output modes; TRGIEN disabled; XPRES disabled; ITB Master; IUE
       disabled; CLIEN disabled; DTCP disabled;
74     PWMCON2 = 0x100;
75     // MDCS Master; FLTIEN disabled; CAM Edge Aligned; DTC Positive dead time
       for all Output modes; TRGIEN disabled; XPRES disabled; ITB Master; IUE
       disabled; CLIEN disabled; DTCP disabled;
76     PWMCON3 = 0x100;
77     //FLTDAT PWM1L Low, PWM1H Low; SWAP disabled; OVRENH enabled; PENL enabled
       ; PMOD Redundant Output Mode; OVRENL enabled; OSYNC enabled; POLL
       disabled; PENH enabled; CLDAT PWM1L Low, PWM1H Low; OVRDAT PWM1L Low,
       PWM1H Low; POLH enabled;
78     __builtin_write_PWMSFR(&IOCON1, 0xE701, &PWMKEY);
79     //FLTDAT PWM2L Low, PWM2H Low; SWAP disabled; OVRENH enabled; PENL enabled
       ; PMOD Redundant Output Mode; OVRENL enabled; OSYNC enabled; POLL
       disabled; PENH enabled; CLDAT PWM2L Low, PWM2H Low; OVRDAT PWM2L Low,
       PWM2H Low; POLH enabled;
80     __builtin_write_PWMSFR(&IOCON2, 0xE701, &PWMKEY);
81     //FLTDAT PWM3L Low, PWM3H Low; SWAP disabled; OVRENH enabled; PENL enabled
       ; PMOD Redundant Output Mode; OVRENL enabled; OSYNC enabled; POLL
       disabled; PENH enabled; CLDAT PWM3L Low, PWM3H Low; OVRDAT PWM3L Low,
       PWM3H Low; POLH enabled;
82     __builtin_write_PWMSFR(&IOCON3, 0xE701, &PWMKEY);
83     //FLTPOL disabled; CLPOL disabled; CLSRC FLT1; CLMOD disabled; FLTMOD
       Fault input is disabled; IFLTMOD disabled; FLTSRC FLT32;
84     __builtin_write_PWMSFR(&FCLCON1, 0xFB, &PWMKEY);
85     //FLTPOL disabled; CLPOL disabled; CLSRC FLT1; CLMOD disabled; FLTMOD
       Fault input is disabled; IFLTMOD disabled; FLTSRC FLT32;
86     __builtin_write_PWMSFR(&FCLCON2, 0xFB, &PWMKEY);
87     //FLTPOL disabled; CLPOL disabled; CLSRC FLT1; CLMOD disabled; FLTMOD
       Fault input is disabled; IFLTMOD disabled; FLTSRC FLT32;
88     __builtin_write_PWMSFR(&FCLCON3, 0xFB, &PWMKEY);
```

```
89      // PDC1 0;
90      PDC1 = 0x0;
91      // PDC2 0;
92      PDC2 = 0x0;
93      // PDC3 0;
94      PDC3 = 0x0;
95      // PHASE1 0;
96      PHASE1 = 0x0;
97      // PHASE2 0;
98      PHASE2 = 0x0;
99      // PHASE3 0;
100     PHASE3 = 0x0;
101     // DTR1 0;
102     DTR1 = 0x0;
103     // DTR2 0;
104     DTR2 = 0x0;
105     // DTR3 0;
106     DTR3 = 0x0;
107     // ALTDTR1 0;
108     ALTDTR1 = 0x0;
109     // ALTDTR2 0;
110     ALTDTR2 = 0x0;
111     // ALTDTR3 0;
112     ALTDTR3 = 0x0;
113     // TRGCMP 0;
114     TRIG1 = 0x0;
115     // TRGCMP 0;
116     TRIG2 = 0x0;
117     // TRGCMP 0;
118     TRIG3 = 0x0;
119     // TRGDIV 1; TRGSTRT 0;
120     TRGCON1 = 0x0;
121     // TRGDIV 1; TRGSTRT 0;
122     TRGCON2 = 0x0;
123     // TRGDIV 1; TRGSTRT 0;
124     TRGCON3 = 0x0;
125     // PWMCAP 0;
126     PWMCAP1 = 0x0;
127     // PWMCAP 0;
128     PWMCAP2 = 0x0;
129     // PWMCAP 0;
130     PWMCAP3 = 0x0;
131     // BPLL disabled; BPHH disabled; BPLH disabled; BCH disabled; FLTLEBEN
        disabled; PLR disabled; CLLEBEN disabled; BCL disabled; PLF disabled; PHR
         disabled; BPHL disabled; PHF disabled;
132     LEBCON1 = 0x0;
133     // BPLL disabled; BPHH disabled; BPLH disabled; BCH disabled; FLTLEBEN
        disabled; PLR disabled; CLLEBEN disabled; BCL disabled; PLF disabled; PHR
         disabled; BPHL disabled; PHF disabled;
134     LEBCON2 = 0x0;
135     // BPLL disabled; BPHH disabled; BPLH disabled; BCH disabled; FLTLEBEN
        disabled; PLR disabled; CLLEBEN disabled; BCL disabled; PLF disabled; PHR
         disabled; BPHL disabled; PHF disabled;
136     LEBCON3 = 0x0;
137     // LEB 0;
138     LEBDLY1 = 0x0;
139     // LEB 0;
```

```
140     LEBDLY2 = 0x0;
141     // LEB 0;
142     LEBDLY3 = 0x0;
143     // CHOPLEN disabled; CHOPHEN disabled; BLANKSEL No state blanking; CHOPSEL
         No state blanking;
144     AUXCON1 = 0x0;
145     // CHOPLEN disabled; CHOPHEN disabled; BLANKSEL No state blanking; CHOPSEL
         No state blanking;
146     AUXCON2 = 0x0;
147     // CHOPLEN disabled; CHOPHEN disabled; BLANKSEL No state blanking; CHOPSEL
         No state blanking;
148     AUXCON3 = 0x0;
149
150
151     // SYNCOEN disabled; SEIEN disabled; SESTAT disabled; SEVTPS 1; SYNCSRC
        SYNCI1; SYNCEN disabled; PTSIDL disabled; PTEN enabled; EIPU disabled;
        SYNCPOL disabled;
152     PTCON = 0x8000;
153 }
154
155
156
157 /**
158  End of File
159 */
```

reset.h file

```
/**
  RESET Generated Driver File

  @Company
    Microchip Technology Inc.

  @File Name
    reset.c

  @Summary
    This is the generated driver implementation file for the RESET driver
    using PIC24 / dsPIC33 / PIC32MM MCUs

  @Description
    This header file provides implementations for driver APIs for RESET.
    Generation Information :
        Product Revision  :  PIC24 / dsPIC33 / PIC32MM MCUs - 1.75.1
        Device            :  dsPIC33EV256GM102
    The generated drivers are tested against the following:
        Compiler          :  XC16 v1.35
        MPLAB             :  MPLAB X v5.05
*/

/*
    (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this
    software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
    EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY
    IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
    PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS,
    COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
    INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
    BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS
    IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF
    ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF
    THESE
    TERMS.
*/

#ifndef RESET_H
#define RESET_H

#include <stdint.h>
#include "reset_types.h"

```

```
51 /**
52 * Checks reset cause, flashes UI with an error code as a result.
53 *
54 * Note: this function should be called before any use of CLRWDT
55 * since it has a side-effect of clearing the appropriate bits in the
56 * register showing reset cause (see DS70602B page 8-10)
57 */
58 uint16_t RESET_GetCause(void);
59
60 /**
61  * It handles the reset cause by clearing the cause register values.
62  * Its a weak function user can override this function.
63  * @return None
64  * @example
65  * <code>
66  * RESET_CauseHandler();
67  * </code>
68  */
69 void __attribute__ ((weak)) RESET_CauseHandler(void);
70
71 /**
72  * This function resets the reset cause register.
73  * @return None
74  * @example
75  * <code>
76  * RESET_CauseClearAll();
77  * </code>
78  */
79 void RESET_CauseClearAll();
80
81 #endif  /* RESET_H */
82 /**
83  End of File
84 */
```

reset.c file

```
/**
  RESET Generated Driver File

  @Company
    Microchip Technology Inc.

  @File Name
    reset.c

  @Summary
    This is the generated driver implementation file for the RESET driver
    using PIC24 / dsPIC33 / PIC32MM MCUs

  @Description
    This header file provides implementations for driver APIs for RESET.
    Generation Information :
        Product Revision  :  PIC24 / dsPIC33 / PIC32MM MCUs - 1.75.1
        Device            :  dsPIC33EV256GM102
    The generated drivers are tested against the following:
        Compiler          :  XC16 v1.35
        MPLAB             :  MPLAB X v5.05
*/

/*
    (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this
    software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
    EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY
    IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
    PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS,
    COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
    INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
    BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS
    IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF
    ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF
    THESE
    TERMS.
*/

#include <stdbool.h>
#include <stdint.h>
#include "xc.h"
#include "reset.h"

/**
```

```
51  Section: Local Variables
52 */
53
54 /**
55  Section: Function prototypes
56 */
57 static bool RESET_CauseFromSoftware(uint16_t resetCause);
58 static bool RESET_CauseFromWatchdogTimer(uint16_t resetCause);
59 static bool RESET_CauseFromConfigurationMismatch(uint16_t resetCause);
60 static bool RESET_CauseFromIllegalOpcode(uint16_t resetCause);
61 static bool RESET_CauseFromExternal(uint16_t resetCause);
62 static bool RESET_CauseFromTrap(uint16_t resetCause);
63 static void RESET_CauseClear(RESET_MASKS resetFlagMask);
64
65 uint16_t RESET_GetCause(void)
66 {
67     return RCON;
68 }
69
70 void __attribute__ ((weak)) RESET_CauseHandler(void)
71 {
72     uint16_t resetCause = RESET_GetCause();
73     if(RESET_CauseFromTrap(resetCause))
74     {
75       RESET_CauseClear(RESET_MASK_TRAPR);
76       //Do something
77     }
78     if(RESET_CauseFromIllegalOpcode(resetCause))
79     {
80       RESET_CauseClear(RESET_MASK_IOPUWR);
81       //Do something
82     }
83     if(RESET_CauseFromConfigurationMismatch(resetCause))
84     {
85       RESET_CauseClear(RESET_MASK_CM);
86       //Do something
87     }
88     if(RESET_CauseFromExternal(resetCause))
89     {
90       RESET_CauseClear(RESET_MASK_EXTR);
91       //Do something
92     }
93     if(RESET_CauseFromSoftware(resetCause))
94     {
95       RESET_CauseClear(RESET_MASK_SWR);
96       //Do something
97     }
98     if(RESET_CauseFromWatchdogTimer(resetCause))
99     {
100      RESET_CauseClear(RESET_MASK_WDTO);
101      //Do something
102    }
103 }
104
105 static bool RESET_CauseFromTrap(uint16_t resetCause)
106 {
107     bool resetStatus = false;
```

```
108     if(resetCause & RESET_MASK_TRAPR)
109     {
110       resetStatus = true;
111     }
112     return resetStatus;
113 }
114 
115 static bool RESET_CauseFromIllegalOpcode(uint16_t resetCause)
116 {
117     bool resetStatus = false;
118     if(resetCause & RESET_MASK_IOPUWR)
119     {
120       resetStatus = true;
121     }
122     return resetStatus;
123 }
124 
125 static bool RESET_CauseFromConfigurationMismatch(uint16_t resetCause)
126 {
127     bool resetStatus = false;
128     if(resetCause & RESET_MASK_CM)
129     {
130       resetStatus = true;
131     }
132     return resetStatus;
133 }
134 
135 static bool RESET_CauseFromExternal(uint16_t resetCause)
136 {
137     bool resetStatus = false;
138     if(resetCause & RESET_MASK_EXTR)
139     {
140       resetStatus = true;
141     }
142     return resetStatus;
143 }
144 
145 static bool RESET_CauseFromSoftware(uint16_t resetCause)
146 {
147     bool resetStatus = false;
148     if(resetCause & RESET_MASK_SWR)
149     {
150       resetStatus = true;
151     }
152     return resetStatus;
153 }
154 
155 static bool RESET_CauseFromWatchdogTimer(uint16_t resetCause)
156 {
157     bool resetStatus = false;
158     if(resetCause & RESET_MASK_WDTO)
159     {
160       resetStatus = true;
161     }
162     return resetStatus;
163 }
164 
```

```
165 static void RESET_CauseClear(RESET_MASKS resetFlagMask)
166 {
167     RCON = RCON & (~resetFlagMask);
168 }
169
170 void RESET_CauseClearAll()
171 {
172     RCON = 0x00;
173 }
174 /**
175  End of File
176 */
```

reset_types.h file

```
/**
  RESET Generated Driver File

  @Company
    Microchip Technology Inc.

  @File Name
    reset_types.h

  @Summary
    This is the generated driver implementation file for the RESET driver
    using PIC24 / dsPIC33 / PIC32MM MCUs

  @Description
    This header file provides implementations for driver APIs for RESET.
    Generation Information :
        Product Revision  :  PIC24 / dsPIC33 / PIC32MM MCUs - 1.75.1
        Device            :  dsPIC33EV256GM102
    The generated drivers are tested against the following:
        Compiler          :  XC16 v1.35
        MPLAB             :  MPLAB X v5.05
*/

/*
    (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this
    software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
    EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY
    IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
    PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS,
    COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
    INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
    BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS
    IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF
    ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF
    THESE
    TERMS.
*/

#ifndef RESET_TYPES_H
#define RESET_TYPES_H

/**
  Section: Type defines
 */
```

```
51
52 /**
53  * RCON error type enumerator. Supported types:
54  * ERR_RCON_TRAPR
55  * ERR_RCON_IOPUWR
56  * ERR_RCON_CM
57  * ERR_RCON_WDTO_ISR
58  */
59 typedef enum tagERROR_TYPE
60 {
61     ERR_RCON_TRAPR      = 1, /** A Trap Conflict Reset has occurred **/
62     ERR_RCON_IOPUWR     = 2, /** An illegal opcode detection, an illegal
    address mode or Uninitialized W register used as an
63                              *   Address Pointer caused a Reset **/
64     ERR_RCON_CM         = 3, /** A Configuration Mismatch Reset has occurred *
    */
65     ERR_RCON_WDTO_ISR   = 4  /** WDT time-out has occurred **/
66 }RESET_TYPES;
67
68 /**
69  * RESET CAUSE Masks. Supported masks:
70  * RESET_MASK_WDTO
71  * RESET_MASK_SWR
72  * REST_MASK_EXTR
73  * RESET_MASK_CM
74  * RESET_MASK_IOPUWR
75  * RESET_MASK_TRAPR
76  */
77 typedef enum tagRESET_MASKS
78 {
79   RESET_MASK_WDTO = 0x0010,
80   RESET_MASK_SWR = 0x0040,
81   RESET_MASK_EXTR = 0x0080,
82   RESET_MASK_CM = 0x0200,
83   RESET_MASK_IOPUWR = 0x4000,
84   RESET_MASK_TRAPR = 0x8000,
85 } RESET_MASKS;
86
87 #endif  /* RESET_TYPES_H */
88 /**
89  End of File
90 */
```

systems.h file

```
/**
  @Generated PIC24 / dsPIC33 / PIC32MM MCUs Source File

  @Company:
    Microchip Technology Inc.

  @File Name:
    system.h

  @Summary:
    This is the system.h file generated using PIC24 / dsPIC33 / PIC32MM MCUs

  @Description:
    This header file provides implementations for driver APIs for all modules selected in the GUI.
    Generation Information :
        Product Revision  :  PIC24 / dsPIC33 / PIC32MM MCUs - pic24-dspic-pic32mm : 1.75.1
        Device            :  dsPIC33EV256GM102
    The generated drivers are tested against the following:
        Compiler          :  XC16 v1.35
        MPLAB             :  MPLAB X v5.05
*/

/*
    (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this
    software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
    EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
    PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS, COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
    INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
    BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF THESE
    TERMS.
*/

#include "xc.h"
#include "stdint.h"
#include "system_types.h"

#ifndef SYSTEM_H
```

```
50 #define SYSTEM_H
51 /**
52  * Initializes the CPU core control register.
53  * @example
54  * <code>
55  * SYSTEM_CORCONInitialize();
56  * </code>
57  */
58 inline static void SYSTEM_CORCONInitialize()
59 {
60     CORCON = (CORCON & 0x00F2) | CORCON_MODE_PORVALUES;     // POR value
61 }
62
63 /**
64  * Sets the CPU core control register operating mode to a value that is
       decided by the
65  * SYSTEM_CORCON_MODES argument.
66  * @param modeValue SYSTEM_CORCON_MODES initialization mode specifier
67  * @example
68  * <code>
69  * SYSTEM_CORCONModeOperatingSet(CORCON_MODE_ENABLEALLSATNORMAL_ROUNDUNBIASED)
       ;
70  * </code>
71  */
72 inline static void SYSTEM_CORCONModeOperatingSet(SYSTEM_CORCON_MODES modeValue
       )
73 {
74     CORCON = (CORCON & 0x00F2) | modeValue;
75 }
76
77 /**
78  * Sets the value of CPU core control register.
79  * @param value value that needs to be written to the CPU core control
       register
80  * @example
81  * <code>
82  *  SYSTEM_CORCONRegisterValueSet(0x00E2);
83  * </code>
84  */
85 inline static void SYSTEM_CORCONRegisterValueSet(uint16_t value)
86 {
87     CORCON = value;
88 }
89
90 /**
91  * Gets the value of CPU core control register.
92  * @return value of the CPU core control register
93  * @example
94  * <code>
95  * corconSave = SYSTEM_CORCONRegisterValueGet();
96  * </code>
97  */
98 inline static uint16_t SYSTEM_CORCONRegisterValueGet(void)
99 {
100     return CORCON;
101 }
102
```

```
103
104 /**
105  * Gets the base address of the DEVID register for the currently selected
        device
106  * @return base address of the DEVID register
107  * @example
108  * <code>
109  * uint32_t devIdAddress;
110  * devIdAddress = SYSTEM_DeviceIdRegisterAddressGet();
111  * </code>
112  */
113 inline static uint32_t SYSTEM_DeviceIdRegisterAddressGet(void)
114 {
115     return __DEVID_BASE;
116 }
117
118 /**
119  * @Param
120     none
121  * @Returns
122     none
123  * @Description
124     Initializes the device to the default states configured in the
125  *                  MCC GUI
126  * @Example
127     SYSTEM_Initialize(void);
128  */
129 void SYSTEM_Initialize(void);
130 #endif  /* SYSTEM_H */
131 /**
132  End of File
133 */
```

system.c file

```
/**
  @Generated PIC24 / dsPIC33 / PIC32MM MCUs Source File

  @Company:
    Microchip Technology Inc.

  @File Name:
    system.h

  @Summary:
    This is the sysetm.h file generated using PIC24 / dsPIC33 / PIC32MM MCUs

  @Description:
    This header file provides implementations for driver APIs for all modules selected in the GUI.
    Generation Information :
        Product Revision  :  PIC24 / dsPIC33 / PIC32MM MCUs - pic24-dspic-pic32mm : 1.75.1
        Device            :  dsPIC33EV256GM102
    The generated drivers are tested against the following:
        Compiler          :  XC16 v1.35
        MPLAB             :  MPLAB X v5.05
*/

/*
    (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS, COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF THESE
    TERMS.
*/

#include "pin_manager.h"
#include "clock.h"
#include "system.h"
#include "stdint.h"
#include "system_types.h"
```

```
50 #include "dma.h"
51 #include "pwm.h"
52 #include "ecan1.h"
53 #include "adc1.h"
54 #include "tmr2.h"
55 #include "tmr4.h"
56 #include "interrupt_manager.h"
57 #include "traps.h"
58
59 void SYSTEM_Initialize(void)
60 {
61     PIN_MANAGER_Initialize();
62     CLOCK_Initialize();
63     INTERRUPT_Initialize();
64     PWM_Initialize();
65     ADC1_Initialize();
66     DMA_Initialize();
67     TMR4_Initialize();
68     TMR2_Initialize();
69     ECAN1_Initialize();
70     INTERRUPT_GlobalEnable();
71     SYSTEM_CORCONModeOperatingSet(CORCON_MODE_PORVALUES);
72 }
73
74 /**
75  End of File
76 */
```

system_types.h file

```
/**
  @Generated PIC24 / dsPIC33 / PIC32MM MCUs Source File

  @Company:
    Microchip Technology Inc.

  @File Name:
    system_types.h

  @Summary:
    This is the system_types.h file generated using PIC24 / dsPIC33 / PIC32MM
    MCUs

  @Description:
    This header file provides implementations for driver APIs for all modules
    selected in the GUI.
    Generation Information :
        Product Revision  :  PIC24 / dsPIC33 / PIC32MM MCUs - pic24-dspic-
    pic32mm : 1.75.1
        Device            :  dsPIC33EV256GM102
    The generated drivers are tested against the following:
        Compiler          :  XC16 v1.35
        MPLAB             :  MPLAB X v5.05
*/

/*
    (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this
    software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
    EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY
    IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
    PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS,
    COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
    INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
    BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS
    IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF
    ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF
    THESE
    TERMS.
*/

#ifndef SYSTEM_TYPES_H
#define SYSTEM_TYPES_H

/**
```

```
49   Section: Type defines
50  */
51
52 /**
53  * CORCON initialization type enumerator. Supported types:
54  * CORCON_MODE_PORVALUES
55  * CORCON_MODE_ENABLEALLSATNORMAL_ROUNDBIASED
56  * CORCON_MODE_ENABLEALLSATNORMAL_ROUNDUNBIASED
57  * CORCON_MODE_DISABLEALLSAT_ROUNDBIASED
58  * CORCON_MODE_DISABLEALLSAT_ROUNDUNBIASED
59  * CORCON_MODE_ENABLEALLSATSUPER_ROUNDBIASED
60  * CORCON_MODE_ENABLEALLSATSUPER_ROUNDUNBIASED
61  */
62 typedef enum tagCORCON_MODE_TYPE
63 {
64     CORCON_MODE_PORVALUES    = 0x0020,                    /** Use POR values
      of CORCON */
65     CORCON_MODE_ENABLEALLSATNORMAL_ROUNDBIASED = 0x00E2,  /** Enable
     saturation for ACCA, ACCB
66                                                           *  and Dataspace
     write, enable normal
67                                                           *  ACCA/ACCB
     saturation mode and set
68                                                           *  rounding to
     Biased (conventional)
69                                                           *  mode. Rest of
     CORCON settings are
70                                                           *  set to the
     default POR values.
71                                                           *  */
72     CORCON_MODE_ENABLEALLSATNORMAL_ROUNDUNBIASED = 0x00E0, /** Enable
     saturation for ACCA, ACCB
73                                                           *  and Dataspace
     write, enable normal
74                                                           *  ACCA/ACCB
     saturation mode and set
75                                                           *  rounding to
     Unbiased (convergent)
76                                                           *  mode. Rest of
     CORCON settings are
77                                                           *  set to the
     default POR values.
78                                                           *  */
79     CORCON_MODE_DISABLEALLSAT_ROUNDBIASED = 0x0022,       /** Disable
     saturation for ACCA, ACCB
80                                                           *  and Dataspace
     write and set
81                                                           *  rounding to
     Biased (conventional)
82                                                           *  mode. Rest of
     CORCON settings are
83                                                           *  set to the
     default POR values.
84                                                           *  */
85     CORCON_MODE_DISABLEALLSAT_ROUNDUNBIASED = 0x0020,     /** Disable
     saturation for ACCA, ACCB
86                                                           *  and Dataspace
```

```
87      write and set                                                   *  rounding to
        Unbiased (convergent)
88                                                                      *  mode. Rest of
        CORCON settings are
89                                                                      *  set to the
        default POR values.
90                                                                      *  */
91      CORCON_MODE_ENABLEALLSATSUPER_ROUNDBIASED = 0x00F2,     /** Enable
        saturation for ACCA, ACCB
92                                                                      *  and Dataspace
        write, enable super
93                                                                      *  ACCA/ACCB
        saturation mode and set
94                                                                      *  rounding to
        Biased (conventional)
95                                                                      *  mode. Rest of
        CORCON settings are
96                                                                      *  set to the
        default POR values.
97                                                                      *  */
98      CORCON_MODE_ENABLEALLSATSUPER_ROUNDUNBIASED = 0x00F0,   /** Enable
        saturation for ACCA, ACCB
99                                                                      *  and Dataspace
        write, enable super
100                                                                     *  ACCA/ACCB
        saturation mode and set
101                                                                     *  rounding to
        Unbiased (convergent)
102                                                                     *  mode. Rest of
        CORCON settings are
103                                                                     *  set to the
        default POR values.
104                                                                     *  */
105 } SYSTEM_CORCON_MODES;
106
107 #endif  /* SYSTEM_TYPES_H */
108 /**
109  End of File
110 */
```

tmr2.h file

```
/**
  TMR2 Generated Driver API Header File

  @Company
    Microchip Technology Inc.

  @File Name
    tmr2.h

  @Summary
    This is the generated header file for the TMR2 driver using PIC24 /
    dsPIC33 / PIC32MM MCUs

  @Description
    This header file provides APIs for driver for TMR2.
    Generation Information :
        Product Revision  :  PIC24 / dsPIC33 / PIC32MM MCUs - 1.75.1
        Device            :  dsPIC33EV256GM102
    The generated drivers are tested against the following:
        Compiler          :  XC16 v1.35
        MPLAB             :  MPLAB X v5.05
*/

/*
    (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this
    software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
    EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY
    IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
    PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS,
    COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
    INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
    BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS
    IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF
    ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF
    THESE
    TERMS.
*/

#ifndef _TMR2_H
#define _TMR2_H

/**
  Section: Included Files
*/
```

```
51
52 #include <xc.h>
53 #include <stdint.h>
54 #include <stdbool.h>
55
56 #ifdef __cplusplus  // Provide C++ Compatibility
57
58     extern "C" {
59
60 #endif
61
62 #define TMR2_INTERRUPT_TICKER_FACTOR     1
63
64 /**
65   Section: Interface Routines
66 */
67
68 /**
69   @Summary
70     Initializes hardware and data for the given instance of the TMR module
71
72   @Description
73     This routine initializes hardware for the instance of the TMR module,
74     using the hardware initialization given data.  It also initializes all
75     necessary internal data.
76
77   @Param
78     None.
79
80   @Returns
81     None
82
83   @Example
84     <code>
85     bool statusTimer1;
86     uint16_t period;
87     uint16_t value;
88
89     period = 0x20;
90
91     TMR2_Initialize();
92
93     TMR2_Period16BitSet(period);
94
95     if((value = TMR2_Period16BitGet())== period)
96     {
97         TMR2_Start();
98     }
99
100    while(1)
101    {
102        TMR2_Tasks();
103        if( (statusTimer1 = TMR2_GetElapsedThenClear()) == true)
104        {
105            TMR2_Stop();
106        }
107    }
```

```
108     </code>
109 */
110 void TMR2_Initialize (void);
111 
112 /**
113   @Summary
114     Updates 32-bit timer value
115 
116   @Description
117     This routine updates 32-bit timer value
118 
119   @Param
120     value       - 32-bit period value
121 
122   @Returns
123     None
124 
125   @Example
126     <code>
127     bool statusTimer1;
128     uint32_t period;
129     uint32_t value;
130 
131     period = 0x20202020;
132 
133     TMR2_Initialize();
134 
135     TMR2_Period32BitSet(period);
136 
137     if((value = TMR2_Period32BitGet())== period)
138     {
139         TMR2_Start();
140     }
141 
142     while(1)
143     {
144         TMR2_Tasks();
145         if( (statusTimer1 = TMR2_IsElapsed()) == true)
146         {
147             TMR2_Stop();
148         }
149     }
150     </code>
151 */
152 
153 void TMR2_Period32BitSet( uint32_t value );
154 
155 /**
156   @Summary
157     Provides the timer 32-bit period value
158 
159   @Description
160     This routine provides the timer 32-bit period value
161 
162   @Param
163     None
164 
```

```
  @Returns
    Timer 32-bit period value

  @Example
    Refer to the example of TMR2_Period32BitSet();
*/

uint32_t TMR2_Period32BitGet( void );

/**
  @Summary
    Updates the timer's 32-bit value

  @Description
    This routine updates the timer's 32-bit value

  @Param
    value       - 32-bit Counter value

  @Returns
    None

  @Example
    <code>
    uint32_t value=0xF0F0F0;

    TMR2_Counter32BitSet(value));

    while(1)
    {
        TMR2_Tasks();
        if( (value == TMR2_Counter32BitGet()))
        {
            TMR2_Stop();
        }
    }
    </code>
*/

void TMR2_Counter32BitSet( uint32_t value );

/**
  @Summary
    Provides 32-bit  current counter value

  @Description
    This routine provides 32-bit current counter value

  @Param
    None

  @Returns
    32-bit current counter value

  @Example
    Refer to the example of TMR2_Counter32BitSet();
*/
```

```
222
223 uint32_t TMR2_Counter32BitGet( void );
224
225
226 /**
227   @Summary
228     Callback for timer interrupt.
229
230   @Description
231     This routine is callback for timer interrupt
232
233   @Param
234     None.
235
236   @Returns
237     None
238
239   @Example
240     Refer to the example of TMR2_Initialize();
241 */
242 void TMR2_CallBack(void);
243
244 /**
245   @Summary
246     Starts the TMR
247
248   @Description
249     This routine starts the TMR
250
251   @Param
252     None.
253
254   @Returns
255     None
256
257   @Example
258     Refer to the example of TMR2_Initialize();
259 */
260
261 void TMR2_Start( void );
262
263 /**
264   @Summary
265     Stops the TMR
266
267   @Description
268     This routine stops the TMR
269
270   @Param
271     None.
272
273   @Returns
274     None
275
276   @Example
277     Refer to the example of TMR2_Initialize();
278 */
```

```
279
280 void TMR2_Stop( void );
281
282 /**
283   @Summary
284     Returns the elapsed status of the timer and clears if flag is set.
285
286   @Description
287     This routine returns the elapsed status of the timer and clears
288     flag if its set.
289
290   @Param
291     None.
292
293   @Returns
294     True - Timer has elapsed.
295     False - Timer has not elapsed.
296
297   @Example
298     Refer to the example of TMR2_Initialize();
299 */
300
301 bool TMR2_GetElapsedThenClear(void);
302
303 /**
304   @Summary
305     Returns the software counter value.
306
307   @Description
308     This routine returns the software counter value.
309
310   @Param
311     None.
312
313   @Returns
314     Software counter value.
315
316   @Example
317     Refer to the example of TMR2_Initialize();
318 */
319
320 int TMR2_SoftwareCounterGet(void);
321
322 /**
323   @Summary
324     Clears the software counter value.
325
326   @Description
327     This routine clears the software counter value.
328
329   @Param
330     None.
331
332   @Returns
333     None
334
335   @Example
```

```
336     Refer to the example of TMR2_Initialize();
337 */
338
339 void TMR2_SoftwareCounterClear(void);
340
341 #ifdef __cplusplus  // Provide C++ Compatibility
342
343     }
344
345 #endif
346
347 #endif //_TMR2_H
348
349 /**
350  End of File
351 */
```

tmr2.c file

```
1
2  /**
3    TMR2 Generated Driver API Source File
4
5    @Company
6      Microchip Technology Inc.
7
8    @File Name
9      tmr2.c
10
11   @Summary
12     This is the generated source file for the TMR2 driver using PIC24 /
       dsPIC33 / PIC32MM MCUs
13
14   @Description
15     This source file provides APIs for driver for TMR2.
16     Generation Information :
17         Product Revision  :  PIC24 / dsPIC33 / PIC32MM MCUs - 1.75.1
18         Device            :  dsPIC33EV256GM102
19     The generated drivers are tested against the following:
20         Compiler          :  XC16 v1.35
21         MPLAB             :  MPLAB X v5.05
22 */
23
24 /*
25     (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this
26     software and any derivatives exclusively with Microchip products.
27
28     THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
29     EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY
       IMPLIED
30     WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
31     PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS,
       COMBINATION
32     WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.
33
34     IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
35     INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
36     WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
37     BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
38     FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS
       IN
39     ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF
       ANY,
40     THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.
41
42     MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF
       THESE
43     TERMS.
44 */
45
46 /**
47   Section: Included Files
48 */
49
50 #include <xc.h>
```

```
51 #include "tmr2.h"
52 #include "pwm.h"
53
54 /**
55   Section: Data Type Definitions
56 */
57
58 /** TMR Driver Hardware Instance Object
59
60   @Summary
61     Defines the object required for the maintainence of the hardware instance.
62
63   @Description
64     This defines the object required for the maintainence of the hardware
65     instance. This object exists once per hardware instance of the peripheral.
66
67   Remarks:
68     None.
69 */
70
71 typedef struct _TMR_OBJ_STRUCT
72 {
73     /* Timer Elapsed */
74     bool                                                timerElapsed;
75     /*Software Counter value*/
76     uint32_t                                             count;
77
78 } TMR_OBJ;
79
80 static TMR_OBJ tmr2_obj;
81
82 /**
83   Section: Driver Interface
84 */
85
86 void TMR2_Initialize (void)
87 {
88     //TMR3 0;
89     TMR3 = 0x00;
90     //PR3 0;
91     PR3 = 0x00;
92     //TMR2 0;
93     TMR2 = 0x00;
94     //Period = 0.0000999946 s; Frequency = 69093750 Hz; PR2 6910;
95     PR2 = 0x1AFE;
96     //TCKPS 1:1; T32 32 Bit; TON enabled; TSIDL disabled; TCS FOSC/2; TGATE
       disabled;
97     T2CON = 0x8008;
98
99
100    IFS0bits.T3IF = false;
101    IEC0bits.T3IE = true;
102
103    tmr2_obj.timerElapsed = false;
104
105 }
106
```

```
107
108 void __attribute__ ( ( interrupt, no_auto_psv ) ) _T3Interrupt (  )
109 {
110     /* Check if the Timer Interrupt/Status is set */
111
112     //***User Area Begin
113
114     // ticker function call;
115     // ticker is 1 -> Callback function gets called everytime this ISR
        executes
116         TMR2_CallBack();
117
118     //***User Area End
119
120     tmr2_obj.count++;
121     tmr2_obj.timerElapsed = true;
122     IFS0bits.T3IF = false;
123 }
124
125
126
127
128 void TMR2_Period32BitSet( uint32_t value )
129 {
130     /* Update the counter values */
131     PR2 = (value & 0x0000FFFF);
132     PR3 = ((value & 0xFFFF0000)>>16);
133 }
134
135 uint32_t TMR2_Period32BitGet( void )
136 {
137     uint32_t periodVal = 0xFFFFFFFF;
138
139     /* get the timer period value and return it */
140     periodVal = (((uint32_t)PR3 <<16) | PR2);
141
142     return( periodVal );
143
144 }
145
146 void TMR2_Counter32BitSet( uint32_t value )
147 {
148     /* Update the counter values */
149    TMR3HLD = ((value & 0xFFFF0000)>>16);
150    TMR2 = (value & 0x0000FFFF);
151
152 }
153
154 uint32_t TMR2_Counter32BitGet( void )
155 {
156     uint32_t countVal = 0xFFFFFFFF;
157     uint16_t countValUpper;
158     uint16_t countValLower;
159
160     countValLower = TMR2;
161     countValUpper = TMR3HLD;
162
```

```
163     /* get the current counter value and return it */
164     countVal = (((uint32_t)countValUpper<<16)| countValLower );
165 
166     return( countVal );
167 
168 }
169 
170 
171 void __attribute__ ((weak)) TMR2_CallBack(void)
172 {
173 //     switch (TMR2_SoftwareCounterGet()) {
174 //         case 0:
175 //             PWM_OverrideHighEnable(PWM_GENERATOR_1);
176 //             PWM_OverrideLowEnable(PWM_GENERATOR_1);
177 //             PWM_OverrideHighEnable(PWM_GENERATOR_2);
178 //             PWM_OverrideLowDisable(PWM_GENERATOR_2);
179 //             PWM_OverrideHighDisable(PWM_GENERATOR_3);
180 //             PWM_OverrideLowEnable(PWM_GENERATOR_3);
181 //             tmr2_obj.count++;
182 //             break;
183 //         case 1:
184 //             PWM_OverrideHighEnable(PWM_GENERATOR_1);
185 //             PWM_OverrideLowDisable(PWM_GENERATOR_1);
186 //             PWM_OverrideHighEnable(PWM_GENERATOR_2);
187 //             PWM_OverrideLowEnable(PWM_GENERATOR_2);
188 //             PWM_OverrideHighDisable(PWM_GENERATOR_3);
189 //             PWM_OverrideLowEnable(PWM_GENERATOR_3);
190 //             tmr2_obj.count++;
191 //             break;
192 //         case 2:
193 //             PWM_OverrideHighEnable(PWM_GENERATOR_1);
194 //             PWM_OverrideLowDisable(PWM_GENERATOR_1);
195 //             PWM_OverrideHighDisable(PWM_GENERATOR_2);
196 //             PWM_OverrideLowEnable(PWM_GENERATOR_2);
197 //             PWM_OverrideHighEnable(PWM_GENERATOR_3);
198 //             PWM_OverrideLowEnable(PWM_GENERATOR_3);
199 //             tmr2_obj.count++;
200 //             break;
201 //         case 3:
202 //             PWM_OverrideHighEnable(PWM_GENERATOR_1);
203 //             PWM_OverrideLowEnable(PWM_GENERATOR_1);
204 //             PWM_OverrideHighDisable(PWM_GENERATOR_2);
205 //             PWM_OverrideLowEnable(PWM_GENERATOR_2);
206 //             PWM_OverrideHighEnable(PWM_GENERATOR_3);
207 //             PWM_OverrideLowDisable(PWM_GENERATOR_3);
208 //             tmr2_obj.count++;
209 //             break;
210 //         case 4:
211 //             PWM_OverrideHighDisable(PWM_GENERATOR_1);
212 //             PWM_OverrideLowEnable(PWM_GENERATOR_1);
213 //             PWM_OverrideHighEnable(PWM_GENERATOR_2);
214 //             PWM_OverrideLowEnable(PWM_GENERATOR_2);
215 //             PWM_OverrideHighEnable(PWM_GENERATOR_3);
216 //             PWM_OverrideLowDisable(PWM_GENERATOR_3);
217 //             tmr2_obj.count++;
218 //             break;
219 //         case 5:
```

```
220 //              PWM_OverrideHighDisable(PWM_GENERATOR_1);
221 //              PWM_OverrideLowEnable(PWM_GENERATOR_1);
222 //              PWM_OverrideHighEnable(PWM_GENERATOR_2);
223 //              PWM_OverrideLowDisable(PWM_GENERATOR_2);
224 //              PWM_OverrideHighEnable(PWM_GENERATOR_3);
225 //              PWM_OverrideLowEnable(PWM_GENERATOR_3);
226 //              TMR2_SoftwareCounterClear();
227 //              break;
228 //          default:
229 //              PWM_OverrideHighEnable(PWM_GENERATOR_1);
230 //              PWM_OverrideLowEnable(PWM_GENERATOR_1);
231 //              PWM_OverrideHighEnable(PWM_GENERATOR_2);
232 //              PWM_OverrideLowEnable(PWM_GENERATOR_2);
233 //              PWM_OverrideHighEnable(PWM_GENERATOR_3);
234 //              PWM_OverrideLowEnable(PWM_GENERATOR_3);
235 //              TMR2_SoftwareCounterClear();
236 //      }
237 }
238
239 void TMR2_Start( void )
240 {
241     /* Reset the status information */
242     tmr2_obj.timerElapsed = false;
243
244     /*Enable the interrupt*/
245     IEC0bits.T3IE = true;
246
247     /* Start the Timer */
248     T2CONbits.TON = 1;
249 }
250
251 void TMR2_Stop( void )
252 {
253     /* Stop the Timer */
254     T2CONbits.TON = false;
255
256     /*Disable the interrupt*/
257     IEC0bits.T3IE = false;
258 }
259
260 bool TMR2_GetElapsedThenClear(void)
261 {
262     bool status;
263
264     status = tmr2_obj.timerElapsed;
265
266     if(status == true)
267     {
268         tmr2_obj.timerElapsed = false;
269     }
270     return status;
271 }
272
273 int TMR2_SoftwareCounterGet(void)
274 {
275     return tmr2_obj.count;
276 }
```

```
277
278 void TMR2_SoftwareCounterClear(void)
279 {
280     tmr2_obj.count = 0;
281 }
282
283 /**
284  End of File
285 */
```

tmr4.h file

```
/**
  TMR4 Generated Driver API Header File

  @Company
    Microchip Technology Inc.

  @File Name
    tmr4.h

  @Summary
    This is the generated header file for the TMR4 driver using PIC24 /
    dsPIC33 / PIC32MM MCUs

  @Description
    This header file provides APIs for driver for TMR4.
    Generation Information :
        Product Revision  :  PIC24 / dsPIC33 / PIC32MM MCUs - 1.75.1
        Device            :  dsPIC33EV256GM102
    The generated drivers are tested against the following:
        Compiler          :  XC16 v1.35
        MPLAB             :  MPLAB X v5.05
*/

/*
    (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this
    software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
    EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY
    IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
    PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS,
    COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
    INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
    BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS
    IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF
    ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF
    THESE
    TERMS.
*/

#ifndef _TMR4_H
#define _TMR4_H

/**
  Section: Included Files
*/
```

```
51
52 #include <xc.h>
53 #include <stdint.h>
54 #include <stdbool.h>
55
56 #ifdef __cplusplus  // Provide C++ Compatibility
57
58     extern "C" {
59
60 #endif
61
62
63 /**
64   Section: Interface Routines
65 */
66
67 /**
68   @Summary
69     Initializes hardware and data for the given instance of the TMR module
70
71   @Description
72     This routine initializes hardware for the instance of the TMR module,
73     using the hardware initialization given data.  It also initializes all
74     necessary internal data.
75
76   @Param
77     None.
78
79   @Returns
80     None
81
82   @Example
83     <code>
84     bool statusTimer1;
85     uint16_t period;
86     uint16_t value;
87
88     period = 0x20;
89
90     TMR4_Initialize();
91
92     TMR4_Period16BitSet(period);
93
94     if((value = TMR4_Period16BitGet())== period)
95     {
96         TMR4_Start();
97     }
98
99     while(1)
100    {
101        TMR4_Tasks();
102        if( (statusTimer1 = TMR4_GetElapsedThenClear()) == true)
103        {
104            TMR4_Stop();
105        }
106    }
107    </code>
```

```
108 */
109 void TMR4_Initialize (void);
110
111 /**
112   @Summary
113     Used to maintain the driver's state machine and implement its ISR
114
115   @Description
116     This routine is used to maintain the driver's internal state machine and
117     implement its ISR for interrupt-driven implementations.
118
119   @Param
120     None.
121
122   @Returns
123     None
124
125   @Example
126     Refer to the example of TMR4_Initialize();
127 */
128
129 void TMR4_Tasks_32BitOperation( void );
130
131 /**
132   @Summary
133     Updates 32-bit timer value
134
135   @Description
136     This routine updates 32-bit timer value
137
138   @Param
139     value       - 32-bit period value
140
141   @Returns
142     None
143
144   @Example
145     <code>
146     bool statusTimer1;
147     uint32_t period;
148     uint32_t value;
149
150     period = 0x20202020;
151
152     TMR4_Initialize();
153
154     TMR4_Period32BitSet(period);
155
156     if((value = TMR4_Period32BitGet())== period)
157     {
158         TMR4_Start();
159     }
160
161     while(1)
162     {
163         TMR4_Tasks();
164         if( (statusTimer1 = TMR4_IsElapsed()) == true)
```

```
165         {
166             TMR4_Stop();
167         }
168     }
169     </code>
170 */
171
172 void TMR4_Period32BitSet( uint32_t value );
173
174 /**
175   @Summary
176     Provides the timer 32-bit period value
177
178   @Description
179     This routine provides the timer 32-bit period value
180
181   @Param
182     None
183
184   @Returns
185     Timer 32-bit period value
186
187   @Example
188     Refer to the example of TMR4_Period32BitSet();
189 */
190
191 uint32_t TMR4_Period32BitGet( void );
192
193 /**
194   @Summary
195     Updates the timer's 32-bit value
196
197   @Description
198     This routine updates the timer's 32-bit value
199
200   @Param
201     value       - 32-bit Counter value
202
203   @Returns
204     None
205
206   @Example
207     <code>
208     uint32_t value=0xF0F0F0;
209
210     TMR4_Counter32BitSet(value));
211
212     while(1)
213     {
214         TMR4_Tasks();
215         if( (value == TMR4_Counter32BitGet()))
216         {
217             TMR4_Stop();
218         }
219     }
220     </code>
221 */
```

```
222
223 void TMR4_Counter32BitSet( uint32_t value );
224
225 /**
226   @Summary
227     Provides 32-bit  current counter value
228
229   @Description
230     This routine provides 32-bit current counter value
231
232   @Param
233     None
234
235   @Returns
236     32-bit current counter value
237
238   @Example
239     Refer to the example of TMR4_Counter32BitSet();
240 */
241
242 uint32_t TMR4_Counter32BitGet( void );
243
244
245
246 /**
247   @Summary
248     Starts the TMR
249
250   @Description
251     This routine starts the TMR
252
253   @Param
254     None.
255
256   @Returns
257     None
258
259   @Example
260     Refer to the example of TMR4_Initialize();
261 */
262
263 void TMR4_Start( void );
264
265 /**
266   @Summary
267     Stops the TMR
268
269   @Description
270     This routine stops the TMR
271
272   @Param
273     None.
274
275   @Returns
276     None
277
278   @Example
```

```
279     Refer to the example of TMR4_Initialize();
280 */
281
282 void TMR4_Stop( void );
283
284 /**
285   @Summary
286     Returns the elapsed status of the timer and clears if flag is set.
287
288   @Description
289     This routine returns the elapsed status of the timer and clears
290     flag if its set.
291
292   @Param
293     None.
294
295   @Returns
296     True - Timer has elapsed.
297     False - Timer has not elapsed.
298
299   @Example
300     Refer to the example of TMR4_Initialize();
301 */
302
303 bool TMR4_GetElapsedThenClear(void);
304
305 /**
306   @Summary
307     Returns the software counter value.
308
309   @Description
310     This routine returns the software counter value.
311
312   @Param
313     None.
314
315   @Returns
316     Software counter value.
317
318   @Example
319     Refer to the example of TMR4_Initialize();
320 */
321
322 int TMR4_SoftwareCounterGet(void);
323
324 /**
325   @Summary
326     Clears the software counter value.
327
328   @Description
329     This routine clears the software counter value.
330
331   @Param
332     None.
333
334   @Returns
335     None
```

```
336
337   @Example
338     Refer to the example of TMR4_Initialize();
339 */
340
341 void TMR4_SoftwareCounterClear(void);
342
343 #ifdef __cplusplus  // Provide C++ Compatibility
344
345     }
346
347 #endif
348
349 #endif //_TMR4_H
350
351 /**
352  End of File
353 */
```

tmr4.c file

```
1
2  /**
3    TMR4 Generated Driver API Source File
4
5    @Company
6      Microchip Technology Inc.
7
8    @File Name
9      tmr4.c
10
11   @Summary
12     This is the generated source file for the TMR4 driver using PIC24 /
       dsPIC33 / PIC32MM MCUs
13
14   @Description
15     This source file provides APIs for driver for TMR4.
16     Generation Information :
17         Product Revision  :  PIC24 / dsPIC33 / PIC32MM MCUs - 1.75.1
18         Device            :  dsPIC33EV256GM102
19     The generated drivers are tested against the following:
20         Compiler          :  XC16 v1.35
21         MPLAB             :  MPLAB X v5.05
22 */
23
24 /*
25     (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this
26     software and any derivatives exclusively with Microchip products.
27
28     THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
29     EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY
       IMPLIED
30     WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
31     PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS,
       COMBINATION
32     WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.
33
34     IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
35     INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
36     WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
37     BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
38     FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS
       IN
39     ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF
       ANY,
40     THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.
41
42     MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF
       THESE
43     TERMS.
44 */
45
46 /**
47   Section: Included Files
48 */
49
50 #include <xc.h>
```

```
51  #include "tmr4.h"
52
53  /**
54    Section: Data Type Definitions
55  */
56
57  /** TMR Driver Hardware Instance Object
58
59    @Summary
60      Defines the object required for the maintainence of the hardware instance.
61
62    @Description
63      This defines the object required for the maintainence of the hardware
64      instance. This object exists once per hardware instance of the peripheral.
65
66    Remarks:
67      None.
68  */
69
70  typedef struct _TMR_OBJ_STRUCT
71  {
72      /* Timer Elapsed */
73      bool                                                timerElapsed;
74      /*Software Counter value*/
75      uint8_t                                             count;
76
77  } TMR_OBJ;
78
79  static TMR_OBJ tmr4_obj;
80
81  /**
82    Section: Driver Interface
83  */
84
85  void TMR4_Initialize (void)
86  {
87      //TMR5 0;
88      TMR5 = 0x00;
89      //PR5 0;
90      PR5 = 0x00;
91      //TMR4 0;
92      TMR4 = 0x00;
93      //Period = 0 s; Frequency = 69093750 Hz; PR4 0;
94      PR4 = 0x00;
95      //TCKPS 1:1; T32 32 Bit; TON disabled; TSIDL disabled; TCS FOSC/2; TGATE
        disabled;
96      T4CON = 0x08;
97
98
99
100     tmr4_obj.timerElapsed = false;
101
102 }
103
104
105 void TMR4_Tasks_32BitOperation( void )
106 {
```

```
107     /* Check if the Timer Interrupt/Status is set */
108     if(IFS1bits.T5IF)
109     {
110         tmr4_obj.count++;
111         tmr4_obj.timerElapsed = true;
112         IFS1bits.T5IF = false;
113     }
114 }
115
116
117
118
119 void TMR4_Period32BitSet( uint32_t value )
120 {
121     /* Update the counter values */
122     PR4 = (value & 0x0000FFFF);
123     PR5 = ((value & 0xFFFF0000)>>16);
124 }
125
126 uint32_t TMR4_Period32BitGet( void )
127 {
128     uint32_t periodVal = 0xFFFFFFFF;
129
130     /* get the timer period value and return it */
131     periodVal = (((uint32_t)PR5 <<16) | PR4);
132
133     return( periodVal );
134
135 }
136
137 void TMR4_Counter32BitSet( uint32_t value )
138 {
139     /* Update the counter values */
140    TMR5HLD = ((value & 0xFFFF0000)>>16);
141    TMR4 = (value & 0x0000FFFF);
142
143 }
144
145 uint32_t TMR4_Counter32BitGet( void )
146 {
147     uint32_t countVal = 0xFFFFFFFF;
148     uint16_t countValUpper;
149     uint16_t countValLower;
150
151     countValLower = TMR4;
152     countValUpper = TMR5HLD;
153
154     /* get the current counter value and return it */
155     countVal = (((uint32_t)countValUpper<<16)| countValLower );
156
157     return( countVal );
158
159 }
160
161
162
163 void TMR4_Start( void )
```

```
164 {
165     /* Reset the status information */
166     tmr4_obj.timerElapsed = false;
167 
168 
169     /* Start the Timer */
170     T4CONbits.TON = 1;
171 }
172 
173 void TMR4_Stop( void )
174 {
175     /* Stop the Timer */
176     T4CONbits.TON = false;
177 
178 }
179 
180 bool TMR4_GetElapsedThenClear(void)
181 {
182     bool status;
183 
184     status = tmr4_obj.timerElapsed;
185 
186     if(status == true)
187     {
188         tmr4_obj.timerElapsed = false;
189     }
190     return status;
191 }
192 
193 int TMR4_SoftwareCounterGet(void)
194 {
195     return tmr4_obj.count;
196 }
197 
198 void TMR4_SoftwareCounterClear(void)
199 {
200     tmr4_obj.count = 0;
201 }
202 
203 /**
204  End of File
205 */
```

traps.h file

```
/**
  System Traps Generated Driver File

  @Company:
    Microchip Technology Inc.

  @File Name:
    traps.h

  @Summary:
    This is the generated driver implementation file for handling traps
    using PIC24 / dsPIC33 / PIC32MM MCUs

  @Description:
    This source file provides implementations for PIC24 / dsPIC33 / PIC32MM
    MCUs traps.
    Generation Information :
        Product Revision  :  PIC24 / dsPIC33 / PIC32MM MCUs - pic24-dspic-
    pic32mm : 1.75.1
        Device            :  dsPIC33EV256GM102
    The generated drivers are tested against the following:
        Compiler          :  XC16 v1.35
        MPLAB             :  MPLAB X v5.05
*/
/*
    (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this
    software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
    EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY
    IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
    PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS,
    COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
    INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
    BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS
    IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF
    ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF
    THESE
    TERMS.
*/

#ifndef _TRAPS_H
#define _TRAPS_H

#include <stdint.h>

```

```
50 /**
51  * Error codes
52  */
53 typedef enum
54 {
55     /* ------ Traps ------ */
56     TRAPS_OSC_FAIL = 0, /** Oscillator Fail Trap vector */
57     TRAPS_STACK_ERR = 1, /** Stack Error Trap Vector */
58     TRAPS_ADDRESS_ERR = 2, /** Address error Trap vector */
59     TRAPS_MATH_ERR = 3, /** Math Error Trap vector */
60     TRAPS_DMAC_ERR = 4, /** DMAC Error Trap vector */
61     TRAPS_HARD_ERR = 7, /** Generic Hard Trap vector */
62     TRAPS_DOOVR_ERR = 10, /** Generic Soft Trap vector */
63 } TRAPS_ERROR_CODE;
64 /**
65   @Summary
66     Default handler for the traps
67
68   @Description
69     This routine will be called whenever a trap happens. It stores the trap
70     error code and waits forever.
71     This routine has a weak attribute and can be over written.
72
73   @Preconditions
74     None.
75
76   @Returns
77     None.
78
79   @Param
80     None.
81
82   @Example
83     None.
84
85 */
86 void __attribute__((naked, noreturn, weak)) TRAPS_halt_on_error(uint16_t code)
       ;
87
88 #endif
```

traps.c file

```
/**
  System Traps Generated Driver File

  @Company:
    Microchip Technology Inc.

  @File Name:
    traps.h

  @Summary:
    This is the generated driver implementation file for handling traps
    using PIC24 / dsPIC33 / PIC32MM MCUs

  @Description:
    This source file provides implementations for PIC24 / dsPIC33 / PIC32MM
    MCUs traps.
    Generation Information :
        Product Revision  :  PIC24 / dsPIC33 / PIC32MM MCUs - pic24-dspic-
    pic32mm : 1.75.1
        Device            :  dsPIC33EV256GM102
    The generated drivers are tested against the following:
        Compiler          :  XC16 v1.35
        MPLAB             :  MPLAB X v5.05
*/
/*
    (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this
    software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
    EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY
    IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
    PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS,
    COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
    INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
    BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS
    IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF
    ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF
    THESE
    TERMS.
*/

/**
    Section: Includes
*/
#include <xc.h>
#include "traps.h"
```

```
50
51 #define ERROR_HANDLER __attribute__((interrupt, no_auto_psv, keep, section("
       error_handler")))
52 #define ERROR_HANDLER_NORETURN ERROR_HANDLER __attribute__((noreturn))
53 #define FAILSAFE_STACK_GUARDSIZE 8
54
55 /**
56  * a private place to store the error code if we run into a severe error
57  */
58 static uint16_t TRAPS_error_code = -1;
59
60 /**
61  * Halts
62  *
63  * @param code error code
64  */
65 void __attribute__((naked, noreturn, weak)) TRAPS_halt_on_error(uint16_t code)
66 {
67     TRAPS_error_code = code;
68 #ifdef __DEBUG
69     __builtin_software_breakpoint();
70     /* If we are in debug mode, cause a software breakpoint in the debugger */
71 #endif
72     while(1);
73
74 }
75
76 /**
77  * Sets the stack pointer to a backup area of memory, in case we run into
78  * a stack error (in which case we can't really trust the stack pointer)
79  */
80 inline static void use_failsafe_stack(void)
81 {
82     static uint8_t failsafe_stack[32];
83     asm volatile (
84         "   mov    %[pstack], W15\n"
85         :
86         : [pstack]"r"(failsafe_stack)
87     );
88 /* Controls where the stack pointer limit is, relative to the end of the
89  * failsafe stack
90  */
91     SPLIM = (uint16_t)(((uint8_t *)failsafe_stack) + sizeof(failsafe_stack)
92             - FAILSAFE_STACK_GUARDSIZE);
93 }
94
95
96 /** Oscillator Fail Trap vector**/
97 void ERROR_HANDLER_NORETURN _OscillatorFail(void)
98 {
99     INTCON1bits.OSCFAIL = 0;  //Clear the trap flag
100     TRAPS_halt_on_error(TRAPS_OSC_FAIL);
101 }
102 /** Stack Error Trap Vector**/
103 void ERROR_HANDLER_NORETURN _StackError(void)
104 {
105     /* We use a failsafe stack: the presence of a stack-pointer error
```

```
106      * means that we cannot trust the stack to operate correctly unless
107      * we set the stack pointer to a safe place.
108      */
109     use_failsafe_stack();
110     INTCON1bits.STKERR = 0;  //Clear the trap flag
111     TRAPS_halt_on_error(TRAPS_STACK_ERR);
112 }
113 /** Address error Trap vector**/
114 void ERROR_HANDLER_NORETURN _AddressError(void)
115 {
116     INTCON1bits.ADDRERR = 0;  //Clear the trap flag
117     TRAPS_halt_on_error(TRAPS_ADDRESS_ERR);
118 }
119 /** Math Error Trap vector**/
120 void ERROR_HANDLER_NORETURN _MathError(void)
121 {
122     INTCON1bits.MATHERR = 0;  //Clear the trap flag
123     TRAPS_halt_on_error(TRAPS_MATH_ERR);
124 }
125 /** DMAC Error Trap vector**/
126 void ERROR_HANDLER_NORETURN _DMACError(void)
127 {
128     INTCON1bits.DMACERR = 0;  //Clear the trap flag
129     TRAPS_halt_on_error(TRAPS_DMAC_ERR);
130 }
131 /** Generic Hard Trap vector**/
132 void ERROR_HANDLER_NORETURN _HardTrapError(void)
133 {
134     INTCON4bits.SGHT = 0;  //Clear the trap flag
135     TRAPS_halt_on_error(TRAPS_HARD_ERR);
136 }
137 /** Generic Soft Trap vector**/
138 void ERROR_HANDLER_NORETURN _SoftTrapError(void)
139 {
140     INTCON3bits.DOOVR = 0;  //Clear the trap flag
141     TRAPS_halt_on_error(TRAPS_DOOVR_ERR);
142 }
```

watchdog.h file

```
/**
  WATCHDOG Generated Driver File

  @Company
    Microchip Technology Inc.

  @File Name
    watchdog.h

  @Summary
    This is the generated driver implementation file for the WATCHDOG driver
    using PIC24 / dsPIC33 / PIC32MM MCUs

  @Description
    This header file provides implementations for driver APIs for WATCHDOG.
    Generation Information :
        Product Revision  :  PIC24 / dsPIC33 / PIC32MM MCUs - 1.75.1
        Device            :  dsPIC33EV256GM102
    The generated drivers are tested against the following:
        Compiler          :  XC16 v1.35
        MPLAB             :  MPLAB X v5.05
*/

/*
    (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this
    software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
    EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY
    IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
    PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS,
    COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
    INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
    BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS
    IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF
    ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF
    THESE
    TERMS.
*/

#ifndef WATCHDOG_H
#define WATCHDOG_H

/**
  Section: Type defines
 */
```

```
51
52 /**
53  * Enables Watch Dog Timer (WDT) using the software bit.
54  * @example
55  * <code>
56  * WATCHDOG_TimerSoftwareEnable();
57  * </code>
58  */
59 inline static void WATCHDOG_TimerSoftwareEnable(void)
60 {
61     RCONbits.SWDTEN = 1;
62 }
63
64 /**
65  * Disables Watch Dog Timer (WDT) using the software bit.
66  * @example
67  * <code>
68  * WATCHDOG_TimerSoftwareDisable();
69  * </code>
70  */
71 inline static void WATCHDOG_TimerSoftwareDisable(void)
72 {
73     RCONbits.SWDTEN = 0;
74 }
75
76 /**
77  * Clears the Watch Dog Timer (WDT).
78  * @example
79  * <code>
80  * WATCHDOG_TimerClear();
81  * </code>
82  */
83 inline static void WATCHDOG_TimerClear(void)
84 {
85     ClrWdt();
86 }
87
88 #endif  /* WATCHDOG_H */
89 /**
90  End of File
91 */
```