Update CRD25DA12N-FMC V1.X.X Firmware to Measure VDC on V2.0 Hardware

Revision 1, November 19, 2025

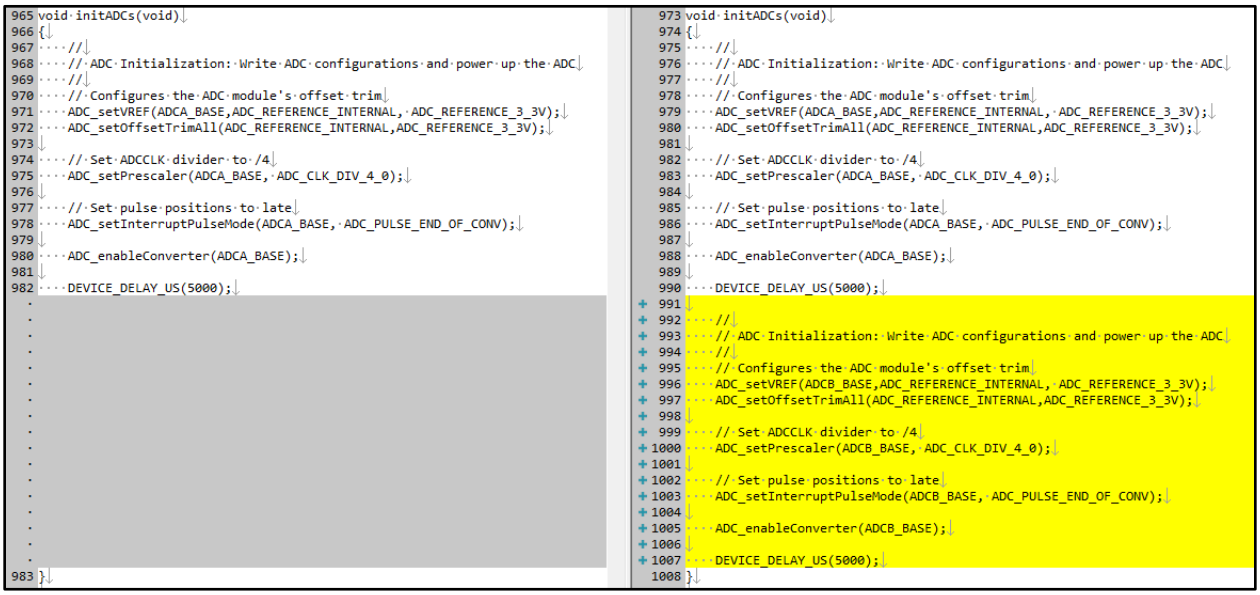
Several upgrades were made to the CRD25DA12N-FMC hardware when it transitioned to version 2.0 (V2.0). These changes were added based on continued internal hardware testing and customer feedback. Many of these changes are backwards compatible with previous versions of the firmware (firmware v1.X.X). However, the DC voltage measurement (V_{DC}) is not backwards compatible. The analog-to-digital conversion (ADC) pinout mapping was changed for V_{DC} to place all the voltage measurements on the same ADC channel and to place the voltage and current measurements on separate ADC channels. These changes allow for better utilization of the post-processing blocks (PPB) built into the C2000 controller and Code Composer Studio. A new version of the firmware (v2.X.X) is already available for the CRD25DA12N-FMC to support V2.0 hardware. For users that are already familiar with the v1.X.X firmware, this document details what changes need to be made for an accurate V_{DC} measurement on V2.0 hardware. Note that in all the *diff* comparisons presented in this document, the original firmware is shown on the left, and the updated changes are shown on the right.

CRD25DA12N-FMC_main.c

Configure ADC Channel B (ADCB)

In the **initADCs** function, add the following code to initialize ADC channel **B**.

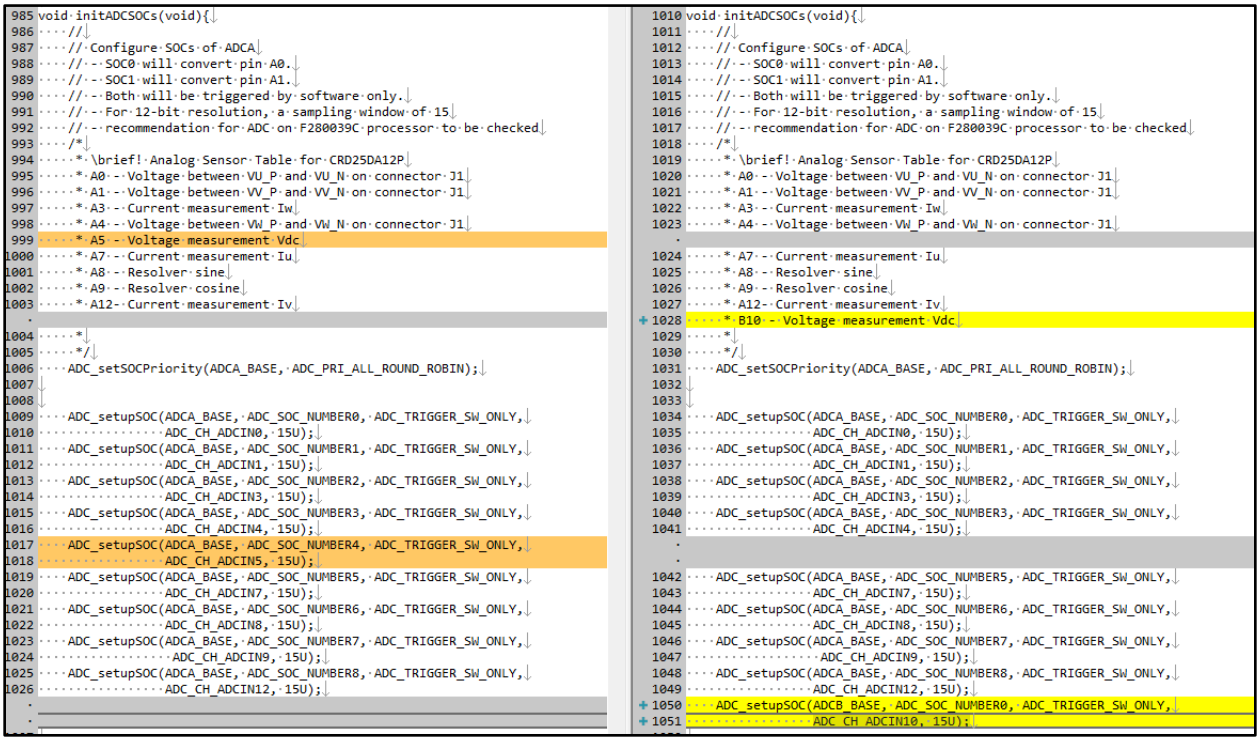
```
//
// ADC Initialization: Write ADC configurations and power up the ADC
//
// Configures the ADC module's offset trim
ADC_setVREF(ADCB_BASE,ADC_REFERENCE_INTERNAL, ADC_REFERENCE_3_3V);
ADC_setOffsetTrimAll(ADC_REFERENCE_INTERNAL,ADC_REFERENCE_3_3V);

// Set ADCCLK divider to /4
ADC_setPrescaler(ADCB_BASE, ADC_CLK_DIV_4_0);

// Set pulse positions to late
ADC_setInterruptPulseMode(ADCB_BASE, ADC_PULSE_END_OF_CONV);

ADC_enableConverter(ADCB_BASE);

DEVICE_DELAY_US(5000);
```



Configure ADC channel B start of conversion (SOC) 0

In the **initADCSOCs** function, add the following code to initialize ADC**B** SOC**0**. Remove the setup for ADC**A** SOC**4** which is the legacy V_{DC} configuration.

```
ADC_setupSOC(ADCB_BASE, ADC_SOC_NUMBER0, ADC_TRIGGER_SW_ONLY, ADC_CH_ADCIN10, 15U);
```



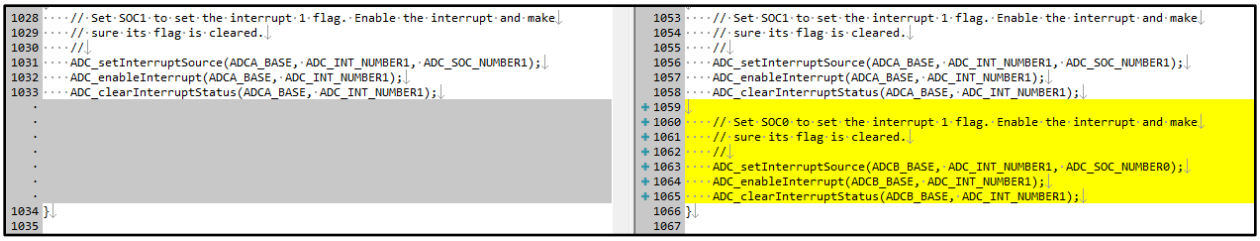
```
void initADCSOCs(void){
    //
    // Configure SOCs of ADCA
    //  -- SOC0 will convert pin A0.
    //  -- SOC1 will convert pin A1.
    //  -- Both will be triggered by software only.
    //  -- For 12-bit resolution, a sampling window of 15
    //  -- recommendation for ADC on F280039C processor to be checked
    /*
     * \brief Analog Sensor Table for CRD25DA12P
     * A0 -- Voltage between VU_P and VU_N on connector J1
     * A1 -- Voltage between VV_P and VV_N on connector J1
     * A3 -- Current measurement Iw
     * A4 -- Voltage between VW_P and VW_N on connector J1
     * A5 -- Voltage measurement Vdc
     * A7 -- Current measurement Iu
     * A8 -- Resolver sine
     * A9 -- Resolver cosine
     * A12-- Current measurement Iv
     *
     */
    ADC_setSOCPriority(ADCA_BASE, ADC_PRI_ALL_ROUND_ROBIN);

    ADC_setupSOC(ADCA_BASE, ADC_SOC_NUMBER0, ADC_TRIGGER_SW_ONLY,
                 ADC_CH_ADCIN0, 15U);
    ADC_setupSOC(ADCA_BASE, ADC_SOC_NUMBER1, ADC_TRIGGER_SW_ONLY,
                 ADC_CH_ADCIN1, 15U);
    ADC_setupSOC(ADCA_BASE, ADC_SOC_NUMBER2, ADC_TRIGGER_SW_ONLY,
                 ADC_CH_ADCIN3, 15U);
    ADC_setupSOC(ADCA_BASE, ADC_SOC_NUMBER3, ADC_TRIGGER_SW_ONLY,
                 ADC_CH_ADCIN4, 15U);
    ADC_setupSOC(ADCA_BASE, ADC_SOC_NUMBER4, ADC_TRIGGER_SW_ONLY,
                 ADC_CH_ADCIN5, 15U);
    ADC_setupSOC(ADCA_BASE, ADC_SOC_NUMBER5, ADC_TRIGGER_SW_ONLY,
                 ADC_CH_ADCIN7, 15U);
    ADC_setupSOC(ADCA_BASE, ADC_SOC_NUMBER6, ADC_TRIGGER_SW_ONLY,
                 ADC_CH_ADCIN8, 15U);
    ADC_setupSOC(ADCA_BASE, ADC_SOC_NUMBER7, ADC_TRIGGER_SW_ONLY,
                 ADC_CH_ADCIN9, 15U);
    ADC_setupSOC(ADCA_BASE, ADC_SOC_NUMBER8, ADC_TRIGGER_SW_ONLY,
                 ADC_CH_ADCIN12, 15U);
    .
    .
```

```
void initADCSOCs(void){
    //
    // Configure SOCs of ADCA
    //  -- SOC0 will convert pin A0.
    //  -- SOC1 will convert pin A1.
    //  -- Both will be triggered by software only.
    //  -- For 12-bit resolution, a sampling window of 15
    //  -- recommendation for ADC on F280039C processor to be checked
    /*
     * \brief Analog Sensor Table for CRD25DA12P
     * A0 -- Voltage between VU_P and VU_N on connector J1
     * A1 -- Voltage between VV_P and VV_N on connector J1
     * A3 -- Current measurement Iw
     * A4 -- Voltage between VW_P and VW_N on connector J1
     .
     * A7 -- Current measurement Iu
     * A8 -- Resolver sine
     * A9 -- Resolver cosine
     * A12-- Current measurement Iv
+    * B10 -- Voltage measurement Vdc
     *
     */
    ADC_setSOCPriority(ADCA_BASE, ADC_PRI_ALL_ROUND_ROBIN);

    ADC_setupSOC(ADCA_BASE, ADC_SOC_NUMBER0, ADC_TRIGGER_SW_ONLY,
                 ADC_CH_ADCIN0, 15U);
    ADC_setupSOC(ADCA_BASE, ADC_SOC_NUMBER1, ADC_TRIGGER_SW_ONLY,
                 ADC_CH_ADCIN1, 15U);
    ADC_setupSOC(ADCA_BASE, ADC_SOC_NUMBER2, ADC_TRIGGER_SW_ONLY,
                 ADC_CH_ADCIN3, 15U);
    ADC_setupSOC(ADCA_BASE, ADC_SOC_NUMBER3, ADC_TRIGGER_SW_ONLY,
                 ADC_CH_ADCIN4, 15U);
    .
    .
    ADC_setupSOC(ADCA_BASE, ADC_SOC_NUMBER5, ADC_TRIGGER_SW_ONLY,
                 ADC_CH_ADCIN7, 15U);
    ADC_setupSOC(ADCA_BASE, ADC_SOC_NUMBER6, ADC_TRIGGER_SW_ONLY,
                 ADC_CH_ADCIN8, 15U);
    ADC_setupSOC(ADCA_BASE, ADC_SOC_NUMBER7, ADC_TRIGGER_SW_ONLY,
                 ADC_CH_ADCIN9, 15U);
    ADC_setupSOC(ADCA_BASE, ADC_SOC_NUMBER8, ADC_TRIGGER_SW_ONLY,
                 ADC_CH_ADCIN12, 15U);
+   ADC_setupSOC(ADCB_BASE, ADC_SOC_NUMBER0, ADC_TRIGGER_SW_ONLY,
+                ADC_CH_ADCIN10, 15U);
```

Configure ADCB Interrupt 1

In the **initADCSOCs** function, add the following code to initialize ADC**B** interrupt 1.

```
// Set SOC0 to set the interrupt 1 flag. Enable the interrupt and make
// sure its flag is cleared.
//
ADC_setInterruptSource(ADCB_BASE, ADC_INT_NUMBER1, ADC_SOC_NUMBER0);
ADC_enableInterrupt(ADCB_BASE, ADC_INT_NUMBER1);
ADC_clearInterruptStatus(ADCB_BASE, ADC_INT_NUMBER1);
```



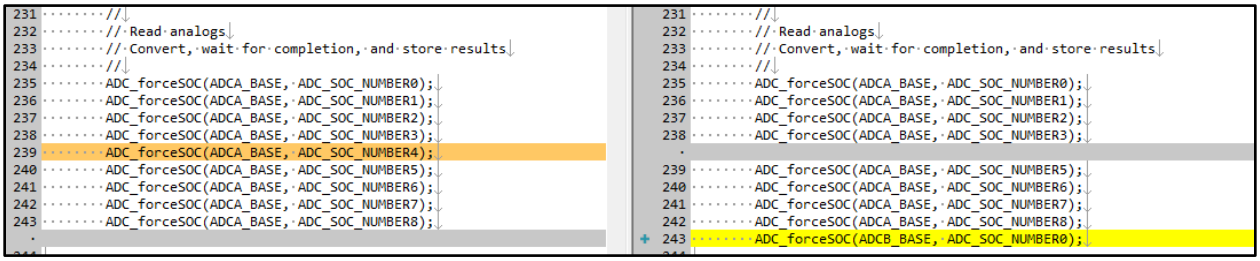
```
    // Set SOC1 to set the interrupt 1 flag. Enable the interrupt and make
    // sure its flag is cleared.
    //
    ADC_setInterruptSource(ADCA_BASE, ADC_INT_NUMBER1, ADC_SOC_NUMBER1);
    ADC_enableInterrupt(ADCA_BASE, ADC_INT_NUMBER1);
    ADC_clearInterruptStatus(ADCA_BASE, ADC_INT_NUMBER1);
    .
    .
    .
    .
    .
    .
}
```

```
    // Set SOC1 to set the interrupt 1 flag. Enable the interrupt and make
    // sure its flag is cleared.
    //
    ADC_setInterruptSource(ADCA_BASE, ADC_INT_NUMBER1, ADC_SOC_NUMBER1);
    ADC_enableInterrupt(ADCA_BASE, ADC_INT_NUMBER1);
    ADC_clearInterruptStatus(ADCA_BASE, ADC_INT_NUMBER1);
+
+   // Set SOC0 to set the interrupt 1 flag. Enable the interrupt and make
+   // sure its flag is cleared.
+   //
+   ADC_setInterruptSource(ADCB_BASE, ADC_INT_NUMBER1, ADC_SOC_NUMBER0);
+   ADC_enableInterrupt(ADCB_BASE, ADC_INT_NUMBER1);
+   ADC_clearInterruptStatus(ADCB_BASE, ADC_INT_NUMBER1);
}
```

Start ADCB SOC0 on Each Loop Iteration

In the **main** function, add the following code to start ADC**B** SOC**0** conversion on each loop iteration. Remove the start for ADC**A** SOC**4** which is the legacy V_{DC} configuration.

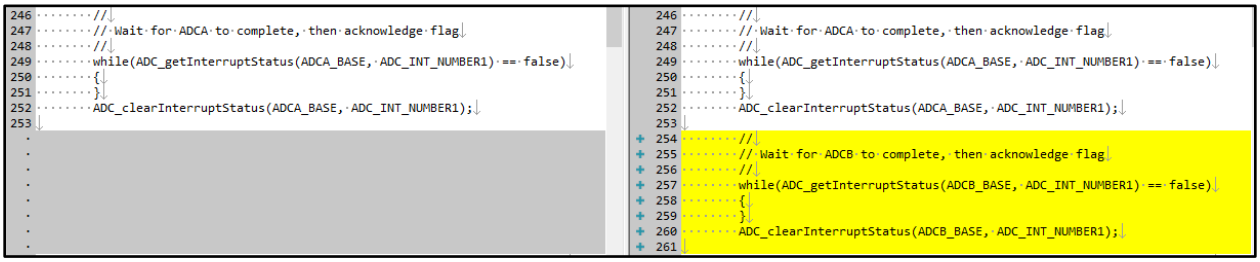
```
ADC_forceSOC(ADCB_BASE, ADC_SOC_NUMBER0);
```



Wait for ADCB Interrupt 1 on Each Loop Iteration

In the **main** function, add the following code to wait for the ADC**B** interrupt 1 flag on each loop iteration which indicates the ADC**B** conversions are complete.

```
//
// Wait for ADCB to complete, then acknowledge flag
//
while(ADC_getInterruptStatus(ADCB_BASE, ADC_INT_NUMBER1) == false)
{
}
ADC_clearInterruptStatus(ADCB_BASE, ADC_INT_NUMBER1);
```



Board_source/Voltage.c

Read ADCB SOC0 for VDC measurement:

Update the **getVoltageDC** function to match the following code. This ensures that the correct ADC register value is used for the calculation.

```
float32_t getVoltageDC()
{
    float val = 0;
    val = (float32_t)1320*((float32_t)ADC_readResult(ADCBRESULT_BASE,
ADC_SOC_NUMBER0)/(float32_t)4096);   //ADC voltage range 0-3.3V; Vref = 3.3V
    return val;
}
```