

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);
```

```
965 void initADCs(void)  
966 {  
967     // //  
968     // // ADC Initialization: Write ADC configurations and power up the ADC  
969     // //  
970     // // Configures the ADC module's offset trim  
971     ADC_setVREF(ADCA_BASE,ADC_REFERENCE_INTERNAL, ADC_REFERENCE_3_3V);  
972     ADC_setOffsetTrimAll(ADC_REFERENCE_INTERNAL,ADC_REFERENCE_3_3V);  
973     // //  
974     // // Set ADCCLK divider to /4  
975     ADC_setPrescaler(ADCA_BASE, ADC_CLK_DIV_4_0);  
976     // //  
977     // // Set pulse positions to late  
978     ADC_setInterruptPulseMode(ADCA_BASE, ADC_PULSE_END_OF_CONV);  
979     // //  
980     ADC_enableConverter(ADCA_BASE);  
981     // //  
982     DEVICE_DELAY_US(5000);  
983 }  
  
+ 991 //  
+ 992 //  
+ 993 // // ADC Initialization: Write ADC configurations and power up the ADC  
+ 994 // //  
+ 995 // // Configures the ADC module's offset trim  
+ 996 ADC_setVREF(ADCB_BASE,ADC_REFERENCE_INTERNAL, ADC_REFERENCE_3_3V);  
+ 997 ADC_setOffsetTrimAll(ADC_REFERENCE_INTERNAL,ADC_REFERENCE_3_3V);  
+ 998 // //  
+ 999 // Set ADCCLK divider to /4  
+ 1000 ADC_setPrescaler(ADCB_BASE, ADC_CLK_DIV_4_0);  
+ 1001 // //  
+ 1002 // Set pulse positions to late  
+ 1003 ADC_setInterruptPulseMode(ADCB_BASE, ADC_PULSE_END_OF_CONV);  
+ 1004 // //  
+ 1005 ADC_enableConverter(ADCB_BASE);  
+ 1006 // //  
+ 1007 DEVICE_DELAY_US(5000);  
+ 1008 }
```

Configure ADC channel B start of conversion (SOC) 0

In the **initADCsOCs** function, add the following code to initialize ADCB SOC0. Remove the setup for ADCA SOC4 which is the legacy V_{DC} configuration.

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

<pre> 985 void initADCSOCs(void){ 986 ...// 987 ...// Configure SOC0 of ADCA 988 ...// SOC0 will convert pin A0. 989 ...// SOC1 will convert pin A1. 990 ...// Both will be triggered by software only. 991 ...// For 12-bit resolution, a sampling window of 15. 992 ...// recommendation for ADC on F280039C processor to be checked. 993 ...// 994 ...// \brief! Analog Sensor Table for CRD25DA12P 995 ...// * A0 -- Voltage between VU_P and VU_N on connector J1 996 ...// * A1 -- Voltage between VW_P and VW_N on connector J1 997 ...// * A3 -- Current measurement Iw 998 ...// * A4 -- Voltage between VW_P and VW_N on connector J1 999 ...// * A5 -- Voltage measurement Vdc 1000 ...// * A7 -- Current measurement Iu 1001 ...// * A8 -- Resolver sine 1002 ...// * A9 -- Resolver cosine 1003 ...// * A12-- Current measurement Iv 1004 ...// 1005 ...// 1006 ADC_setSOPriority(ADCA_BASE, ADC_PRI_ALL_ROUND_ROBIN); 1007 1008 1009 ADC_setupSOC(ADCA_BASE, ADC_SOC_NUMBER0, ADC_TRIGGER_SW_ONLY, 1010 ADC_CH_ADCIN0, 15U); 1011 ADC_setupSOC(ADCA_BASE, ADC_SOC_NUMBER1, ADC_TRIGGER_SW_ONLY, 1012 ADC_CH_ADCIN1, 15U); 1013 ADC_setupSOC(ADCA_BASE, ADC_SOC_NUMBER2, ADC_TRIGGER_SW_ONLY, 1014 ADC_CH_ADCIN3, 15U); 1015 ADC_setupSOC(ADCA_BASE, ADC_SOC_NUMBER3, ADC_TRIGGER_SW_ONLY, 1016 ADC_CH_ADCIN4, 15U); 1017 ADC_setupSOC(ADCA_BASE, ADC_SOC_NUMBER4, ADC_TRIGGER_SW_ONLY, 1018 ADC_CH_ADCIN5, 15U); 1019 ADC_setupSOC(ADCA_BASE, ADC_SOC_NUMBER5, ADC_TRIGGER_SW_ONLY, 1020 ADC_CH_ADCIN7, 15U); 1021 ADC_setupSOC(ADCA_BASE, ADC_SOC_NUMBER6, ADC_TRIGGER_SW_ONLY, 1022 ADC_CH_ADCIN8, 15U); 1023 ADC_setupSOC(ADCA_BASE, ADC_SOC_NUMBER7, ADC_TRIGGER_SW_ONLY, 1024 ADC_CH_ADCIN9, 15U); 1025 ADC_setupSOC(ADCA_BASE, ADC_SOC_NUMBER8, ADC_TRIGGER_SW_ONLY, 1026 ADC_CH_ADCIN12, 15U); 1027 1028 </pre>	<pre> 1010 void initADCSOCs(void){ 1011 ...// 1012 ...// Configure SOC0 of ADCA 1013 ...// SOC0 will convert pin A0. 1014 ...// SOC1 will convert pin A1. 1015 ...// Both will be triggered by software only. 1016 ...// For 12-bit resolution, a sampling window of 15. 1017 ...// recommendation for ADC on F280039C processor to be checked. 1018 ...// 1019 ...// \brief! Analog Sensor Table for CRD25DA12P 1020 ...// * A0 -- Voltage between VU_P and VU_N on connector J1 1021 ...// * A1 -- Voltage between VW_P and VW_N on connector J1 1022 ...// * A3 -- Current measurement Iw 1023 ...// * A4 -- Voltage between VW_P and VW_N on connector J1 1024 1025 ...// * A7 -- Current measurement Iu 1026 ...// * A8 -- Resolver sine 1027 ...// * A9 -- Resolver cosine 1028 ...// * A12-- Current measurement Iv 1029 ...// 1030 ...// 1031 ADC_setSOPriority(ADCA_BASE, ADC_PRI_ALL_ROUND_ROBIN); 1032 1033 1034 ADC_setupSOC(ADCA_BASE, ADC_SOC_NUMBER0, ADC_TRIGGER_SW_ONLY, 1035 ADC_CH_ADCIN0, 15U); 1036 ADC_setupSOC(ADCA_BASE, ADC_SOC_NUMBER1, ADC_TRIGGER_SW_ONLY, 1037 ADC_CH_ADCIN1, 15U); 1038 ADC_setupSOC(ADCA_BASE, ADC_SOC_NUMBER2, ADC_TRIGGER_SW_ONLY, 1039 ADC_CH_ADCIN3, 15U); 1040 ADC_setupSOC(ADCA_BASE, ADC_SOC_NUMBER3, ADC_TRIGGER_SW_ONLY, 1041 ADC_CH_ADCIN4, 15U); 1042 1043 1044 ADC_setupSOC(ADCA_BASE, ADC_SOC_NUMBER5, ADC_TRIGGER_SW_ONLY, 1045 ADC_CH_ADCIN7, 15U); 1046 ADC_setupSOC(ADCA_BASE, ADC_SOC_NUMBER6, ADC_TRIGGER_SW_ONLY, 1047 ADC_CH_ADCIN8, 15U); 1048 ADC_setupSOC(ADCA_BASE, ADC_SOC_NUMBER7, ADC_TRIGGER_SW_ONLY, 1049 ADC_CH_ADCIN9, 15U); 1050 ADC_setupSOC(ADCA_BASE, ADC_SOC_NUMBER8, ADC_TRIGGER_SW_ONLY, 1051 ADC_CH_ADCIN12, 15U); 1052 1053 1054 ADC_setupSOC(ADCB_BASE, ADC_SOC_NUMBER0, ADC_TRIGGER_SW_ONLY, 1055 ADC_CH_ADCIN10, 15U); </pre>
---	--

Configure ADCB Interrupt 1

In the `initADCSOCs` function, add the following code to initialize ADCB 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);

```

<pre> 1028 ...// Set SOC1 to set the interrupt 1 flag. Enable the interrupt and make 1029 ...// sure its flag is cleared. 1030 ...// 1031 ADC_setInterruptSource(ADCA_BASE, ADC_INT_NUMBER1, ADC_SOC_NUMBER1); 1032 ADC_enableInterrupt(ADCA_BASE, ADC_INT_NUMBER1); 1033 ADC_clearInterruptStatus(ADCA_BASE, ADC_INT_NUMBER1); 1034 } 1035 </pre>	<pre> 1053 ...// Set SOC1 to set the interrupt 1 flag. Enable the interrupt and make 1054 ...// sure its flag is cleared. 1055 ...// 1056 ADC_setInterruptSource(ADCA_BASE, ADC_INT_NUMBER1, ADC_SOC_NUMBER1); 1057 ADC_enableInterrupt(ADCA_BASE, ADC_INT_NUMBER1); 1058 ADC_clearInterruptStatus(ADCA_BASE, ADC_INT_NUMBER1); 1059 1060 // Set SOC0 to set the interrupt 1 flag. Enable the interrupt and make 1061 // sure its flag is cleared. 1062 // 1063 ADC_setInterruptSource(ADCB_BASE, ADC_INT_NUMBER1, ADC_SOC_NUMBER0); 1064 ADC_enableInterrupt(ADCB_BASE, ADC_INT_NUMBER1); 1065 ADC_clearInterruptStatus(ADCB_BASE, ADC_INT_NUMBER1); 1066 } 1067 </pre>
--	--

Start ADCB SOC0 on Each Loop Iteration

In the `main` function, add the following code to start ADCB SOC0 conversion on each loop iteration. Remove the start for ADCA SOC4 which is the legacy V_{DC} configuration.

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

```
231 .....//|
232 .....// Read analogs|
233 .....// Convert, wait for completion, and store results|
234 .....//|
235 .....ADC_forceSOC(ADCA_BASE, ADC_SOC_NUMBER0);|
236 .....ADC_forceSOC(ADCA_BASE, ADC_SOC_NUMBER1);|
237 .....ADC_forceSOC(ADCA_BASE, ADC_SOC_NUMBER2);|
238 .....ADC_forceSOC(ADCA_BASE, ADC_SOC_NUMBER3);|
239 .....ADC_forceSOC(ADCA_BASE, ADC_SOC_NUMBER4);|
240 .....ADC_forceSOC(ADCA_BASE, ADC_SOC_NUMBER5);|
241 .....ADC_forceSOC(ADCA_BASE, ADC_SOC_NUMBER6);|
242 .....ADC_forceSOC(ADCA_BASE, ADC_SOC_NUMBER7);|
243 .....ADC_forceSOC(ADCA_BASE, ADC_SOC_NUMBER8);|
244 .....|

231 .....//|
232 .....// Read analogs|
233 .....// Convert, wait for completion, and store results|
234 .....//|
235 .....ADC_forceSOC(ADCA_BASE, ADC_SOC_NUMBER0);|
236 .....ADC_forceSOC(ADCA_BASE, ADC_SOC_NUMBER1);|
237 .....ADC_forceSOC(ADCA_BASE, ADC_SOC_NUMBER2);|
238 .....ADC_forceSOC(ADCA_BASE, ADC_SOC_NUMBER3);|
239 .....|
240 .....ADC_forceSOC(ADCA_BASE, ADC_SOC_NUMBER5);|
241 .....ADC_forceSOC(ADCA_BASE, ADC_SOC_NUMBER6);|
242 .....ADC_forceSOC(ADCA_BASE, ADC_SOC_NUMBER7);|
243 .....ADC_forceSOC(ADCA_BASE, ADC_SOC_NUMBER8);|
+ 244 .....ADC_forceSOC(ADCB_BASE, ADC_SOC_NUMBER0);|
245 .....|
```

Wait for ADCB Interrupt 1 on Each Loop Iteration

In the **main** function, add the following code to wait for the ADCB interrupt 1 flag on each loop iteration which indicates the ADCB 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);
```

```
246 .....//|
247 .....// Wait for ADCA to complete, then acknowledge flag|
248 .....//|
249 .....while(ADC_getInterruptStatus(ADCA_BASE, ADC_INT_NUMBER1) == false)|
250 .....{|
251 .....|
252 .....ADC_clearInterruptStatus(ADCA_BASE, ADC_INT_NUMBER1);|
253 .....|

+ 254 .....//|
+ 255 .....// Wait for ADCB to complete, then acknowledge flag|
+ 256 .....//|
+ 257 .....while(ADC_getInterruptStatus(ADCB_BASE, ADC_INT_NUMBER1) == false)|
+ 258 .....{|
+ 259 .....|
+ 260 .....ADC_clearInterruptStatus(ADCB_BASE, ADC_INT_NUMBER1);|
+ 261 .....|
```

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;
}
```

<pre>34 float32_t getVoltageDC() 35 { 36 float val = 0; 37 val = (float32_t)1320*((float32_t)ADC_readResult(ADCBRESULT_BASE, ADC_SOC_NUMBER4)/(float32_t)4096); //ADC voltage range 0-3.3V; Vref = 3.3V 38 return val; 39 }</pre>	<pre>34 float32_t getVoltageDC() 35 { 36 float val = 0; + 37 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 38 return val; 39 }</pre>
---	---