Application Report
SPRACO5–September 2019

Leveraging High Resolution Capture (HRCAP) for Single Wire Data Transfer

Himanshu Chaudhary

ABSTRACT

In various industrial and automotive applications, it is often required to transfer data between two electrically isolated devices. The standard protocols require more than one communication line to transfer data, which leads to the use of multiple isolation buffers that ultimately increase the system cost. Also, in case of standard serial protocols, unit pulse/clock is used to encode only a single bit. This application report discusses how to leverage the high resolution capture unit in C2000™ to improve the transmission latency and reduce the cost by facilitating data transfer in just a single pulse through a single wire. The experimental setup details and results are also being discussed in this application report. You can quickly verify and utilize this system in various applications using the provided algorithm and source code that can be downloaded from the C2000Ware.

Contents

List of Figures

List of Tables

Trademarks

C2000 is a trademark of Texas Instruments.
All other trademarks are the property of their respective owners.

1 Introduction

Pulse Width Modulation (PWM) is a powerful way of encoding analog signal levels and is commonly used for controlling DC power to an electrical device. PWM can also be used for data transmission by varying the output pulse width proportionally with the data to be transferred, but a powerful decoding unit is required at the receiver. Normal capture peripheral eCAP cannot support high transmission frequencies for high resolution data communication, which results in a high latency solution. However, HRCAP of C2000 operates over and above standard eCAP and is capable of measuring width of external pulses to a higher degree of accuracy compared to standard capture unit eCAP. Thus, using HRCAP as a decoding unit at the receiver adds differentiation to the system by significantly reducing the latency for high resolution data transfer.

The application report describes the high resolution peripherals of C2000 in brief and discusses the system implementation details. The document also discusses the software flow of both the transmitter and receiver examples and the achieved effective number of bits (ENOB) and latency at various transmission frequencies. Use Table 1 to choose various parameters while designing your own system. The showcasing example configures F2838x as transmitter and F28004x as receiver, but the software can easily be ported to any C2000 device that supports HRCAP Type 1. Also in order to quickly evaluate the system, the single device loopback example is also provided where the same F28004x device is used as both transmitter and receiver.

The showcasing example code discussed in this document can be found in C2000Ware v2.00.00.03 or latest, located within the following local directory after installation:

C:\ti\c2000\C2000Ware_<version_number>\demo_examples\hrcap_hrpwm_data_transfer

The available example projects are:

- cross_device_data_tansfer\Transmitter
- cross_device_data_tansfer\Receiver
- device_loopback

2 System Showcase

The showcasing data communication system utilizes the high resolution PWM (HRPWM) module of C2000 as an encoding unit and high resolution capture (HRCAP) of C2000 as decoding unit. Figure 1 illustrates the transmitter and the receiver blocks. Based on the analog input sampled by transmitter, CPU configures the high resolution PWM signal that is then captured using high resolution capture at receiver and the decoded data is communicated to CPU.

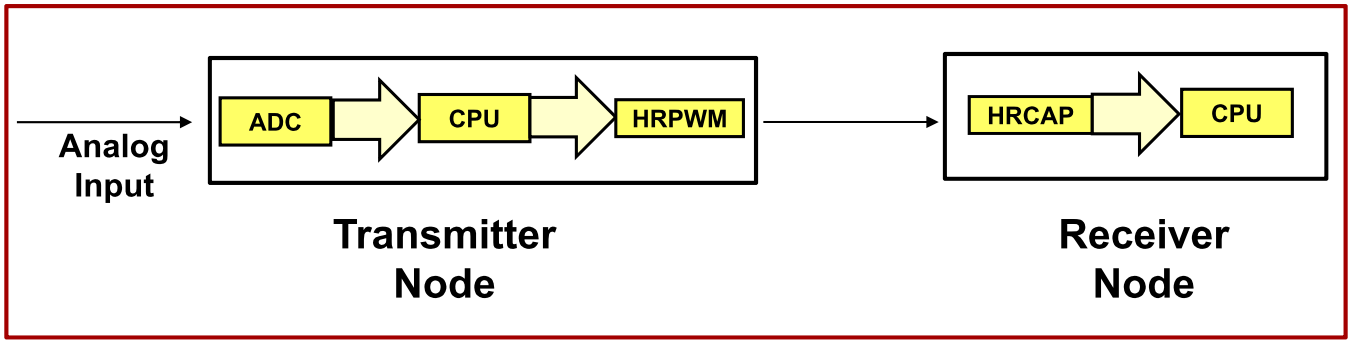

Figure 1. Transmitter and Receiver Blocks

2.1 HRPWM: Encoding Unit

HRPWM extends the time resolution capabilities of the conventionally derived digital PWM by using micro-edge positioner (MEP) technology. MEP logic is capable of positioning an edge very finely by subdividing one system clock of conventional PWM generator and provides accuracy on order of picoseconds. The conventional ePWM module does not offer sufficient resolution at high frequencies and that's where HRPWM can really differentiate if used as an encoding unit as the following system aims to establish low latency and high resolution data transfer. For more details on HRPWM, see the *TMS320F2838x Microcontrollers With Connectivity Manager Data Sheet* and the *TMS320F2838x Microcontrollers Technical Reference Manual*.

The function of the encoding unit is to vary the PWM pulse width proportionally to the analog input. This is realized by configuring ePWM to provide conventional PWM of desired frequency and then programming HRPWM MEP to provide precision control on the falling edge (FE), while keeping the rising edge (RE) constant. Also, the MEP is programmed to use the control mode as duty control instead of phase control as the duty parameter of the signal is used for decoding at the receiver. In order to speed-up the encoding routine, auto-conversion mode is enabled in conjunction with scale factor optimization (SFO) library which will automatically calculate the appropriate number of MEP steps represented by the fractional duty cycle and move the falling edge of signal accordingly. The HRPWM programming sequence for the discussed configuration using driverlib APIs is also shown in the code below. For more details, see the *Driverlib API User's Guide* inside C2000ware. Shadowing (double buffering) is enabled for the registers to prevent any signal distortion by making sure any register update does not affect the current cycle.

```
//
// Configure MEP edge & control mode for channel A & B, MEP Edge control is
// on falling edge. Control mode is duty control.
//
HRPWM_setMEPEdgeSelect (base, HRPWM_CHANNEL_A, HRPWM_MEP_CTRL_FALLING_EDGE);
HRPWM_setMEPControlMode (base, HRPWM_CHANNEL_A, HRPWM_MEP_DUTY_PERIOD_CTRL);

HRPWM_setMEPEdgeSelect (base, HRPWM_CHANNEL_B, HRPWM_MEP_CTRL_FALLING_EDGE);
HRPWM_setMEPControlMode (base, HRPWM_CHANNEL_B, HRPWM_MEP_DUTY_PERIOD_CTRL);

//
// Enable Automatic Conversion mode
//
HRPWM_enableAutoConversion (base);
```

2.2 HRCAP: Decoding Unit

HRCAP enhances the capture resolution capabilities of the conventional capture unit eCAP by utilizing the high-speed asynchronous clock source called HRCLK to measure the ticks between any two edges with a resolution in order of nanoseconds as discussed in the *TMS320F28004x Piccolo™ Microcontrollers Data Sheet*. In order to convert the captured tick counts to time-converted measurements, it is required to make continuous period calibrations as HRCLK is sensitive to changes in both temperature and voltage. Also it is important to note that HRCAP raw measured output is constant or fixed offset-shifted and has some variations that result in a probability distribution, as indicated in the *TMS320F28004x Piccolo™ Microcontrollers Data Sheet* as well.

The function of the decoding unit is to measure signal pulse width and map it to the corresponding encoded data. In order to minimize the variations in HRCAP decoded output, duty based decoding approach is used in this system. In duty based decoding, HRCAP is configured in a continuous mode to latch absolute counter values at 3 edges: rising edge, falling edge followed by another rising edge. The programming sequence for this configuration using driverlib APIs (see the *Driverlib API User's Guide* inside C2000ware) is shown in the code below. These absolute values are then converted to relative measurements in order to find the ON time and period and then duty is calculated by computing the ratio of ON time count and period count. This duty value is finally used to map to the encoded data rather than the ON time. This approach smartly suppresses the variations in the absolute raw output and ultimately leads to better system performance by achieving higher ENOB value. Another major advantage of using duty based decoding is that there is no need of converting ON time and period raw counts into time-converted measurements as ultimately duty is just a ratio of these two. This makes the decoded output immune to any temperature/pressure variations and also eliminates the need of continuous periodic HRCAP calibration ultimately freeing the CPU bandwidth.

```
//
// Continous mode, stop at 3 events
//
ECAP_setCaptureMode (ECAP6_BASE,
                     ECAP_CONTINOUS_CAPTURE_MODE,
                     ECAP_EVENT_3);
//
// Event 1, Event 3 rising edge. Event 2 falling edge
//
ECAP_setEventPolarity (ECAP6_BASE,
                       ECAP_EVENT_1,
                       ECAP_EVENT_RISING_EDGE);
ECAP_setEventPolarity (ECAP6_BASE,
                       ECAP_EVENT_2,
                       ECAP_EVENT_RISING_EDGE);
ECAP_setEventPolarity (ECAP6_BASE,
                       ECAP_EVENT_3,
                       ECAP_EVENT_RISING_EDGE);
```

3 Software Flow

The proposed data transfer technique is demonstrated using F2838x as transmitter and F28004x as receiver, both the transmitter and receiver projects are provided in the C2000Ware. The transmitter example makes use of the internal analog-to-digital converter (ADC) to sample the external real signal at a configurable sampling frequency and converts it into a digital value, which is then sent across to the receiver using proposed communication technique. For better visual inspection of the system functionality, the example also allows you to manually write to the "dig_value_input" using the CCS expressions window at transmitter window and observe the receiver's expression window so as to verify that the sent data is received correctly. In order to make use of this manual inspection mode, set USE_ADC_INPUT = 0, then you need to set "USE_ADC_NPUT" = 0, which will use the user-written value for data transfer instead of the sampled ADC data. The block diagram for the cross device communication example is shown in Figure 2. The same data transfer is also demonstrated as loopback example as well using the same device F28004x as both transmitter and receiver, in case you want to validate the functionality by just having one device. The block diagram for the loopback example is shown in Figure 3.

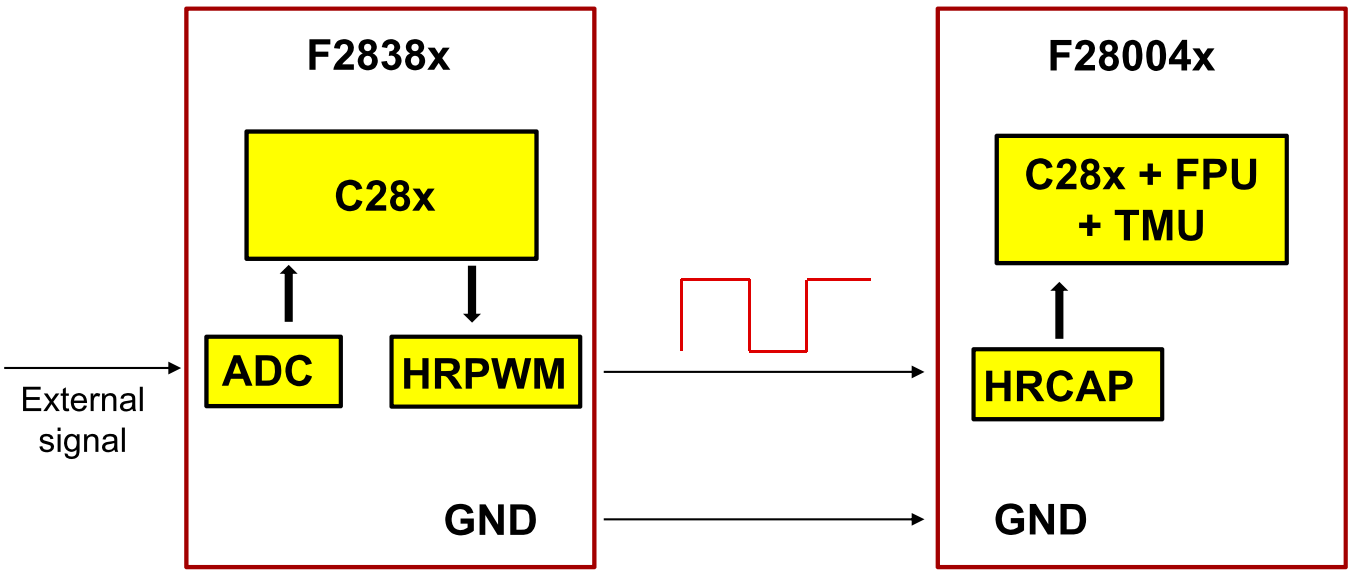

Figure 2. Cross Device Data Transfer

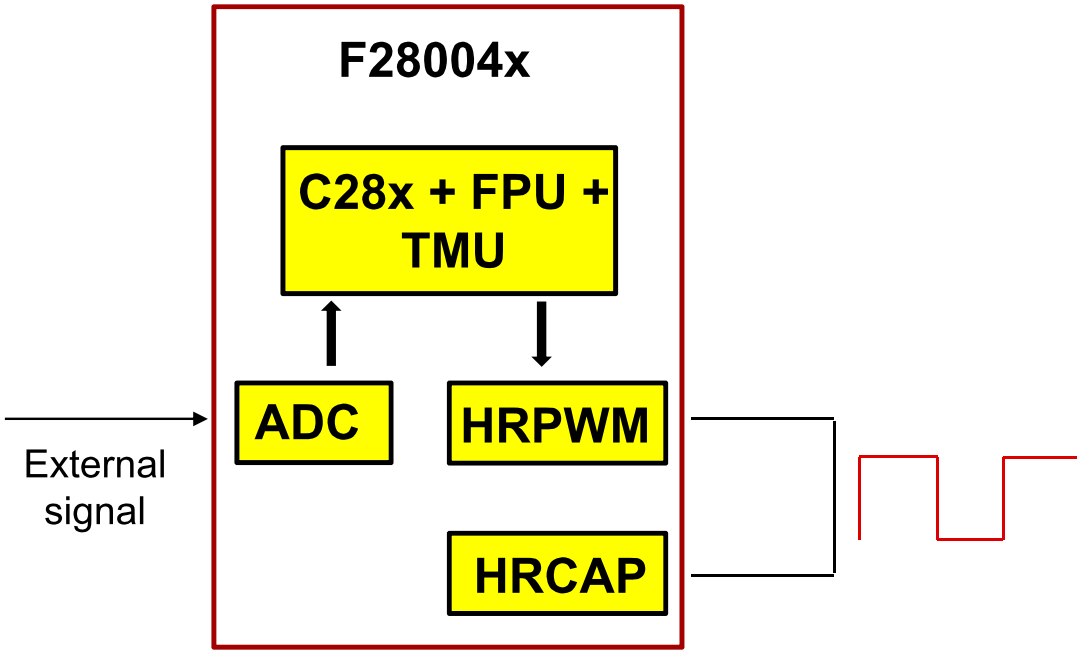

Figure 3. Device Loopback Data Transfer

3.1 User Configurable Parameters

The example supports the following user configurable parameters to let customers quickly adopt this technique based on their design requirements:

- **BIT_RESOLUTION**: The resolution at which you want to encode/decode data. To choose the appropriate resolution based on the ENOB and latency requirements, see Table 1. For a particular PWM transmission frequency, the ENOB value specified in Table 1 suggests that particular number of bits of data can be transferred accurately and reliably using this technique. If a higher resolution than ENOB is chosen, then the least significant (BIT_RESOLUTION - ENOB) bits could fluctuate. By default, resolution is set to 11-bit in the given examples.
- **ADC_NORM**: Based on selected BIT_RESOLUTION, you need to compute the scale factor as (1 / 2^(BIT_RESOLUTION)) and program this value. This scale factor is actually going to be used for encoding, thus, do not forget to update this value along with BIT_RESOLUTION.
- **PWMCLK**: The EPWM module clock at which you want to operate, choose either 100 MHz/200 Mhz depending upon maximum supported by the particular transmitter device. For example, F2838x can support the 200 MHz module clock while F28004x can support the maximum up to 100 MHz. By default, it is set to 100 MHz in transmitter code.
- **PWM_FREQ**: The PWM signal frequency used for transmission. To choose the appropriate PWM transmission frequency based on ENOB and latency requirements, see Table 1. By default, it is set to 200 KHz in the example.
- **ADC_SAMPLE_FREQ**: The ADC trigger frequency in KHz, by default its set to 25 KHz. The maximum programmable value of ADC_SAMPLE_FREQ can be equal to PWM_FREQ.
- **USE_ADC_INPUT**: Control switch used to choose between manual user-written data or ADC acquired data. By default, it is set to 0 (manual mode).

3.2 SFO Background Loop

The scale factor optimization (SFO) library is used to compute the appropriate MEP scale factor in a background loop. As discussed in Section 2.1, HRPWM utilizes MEP technology for accurately positioning the falling edge, but, the MEP step size varies based on worst case process parameters, operating temperature and voltage. In order to maintain a constant a mapping function between encoded and decoded value, the correct MEP scaling factor needs to be known to the transmitter software. That is why the SFO calibration is always active in the background loop to compensate for any variations in temperature, pressure or any external factors. The background SFO calibration loop in the application code is shown below:

```
while(1)
{
     while(status == SFO_INCOMPLETE)
       {
         status = SFO();
         if(status == SFO_ERROR)
         {
             error(); // SFO function returns 2 if an error occurs & #
                      // of MEP steps/coarse step exceeds maximum of 255
         }
         else if (status == SFO_COMPLETE)
             status = SFO_INCOMPLETE;
     }
} // end infinite for loop
```

3.3 Interrupt Based Encoding and Decoding

With SFO calibration is being implemented in the background, the encoding routine is implemented in the foreground at the transmitter using an interrupt-based mechanism. The showcasing example utilizes internal ADC for sampling the external signal at a specified sampling frequency, the completion of ADC conversion triggers an ISR, which then calls the encoding function. The encoding function calculates the normalized digital data based on the user-configured bit resolution. This normalized value is then mapped to the duty of PWM signal. For example values, see Figure 5. It is important to note that the HRCAP module has constraints in terms of minimum pulse width of captured signal. For more details, see the *TMS320F28004x Piccolo™ Microcontrollers Data Sheet*. In order to satisfy this constraint, the encoded duty values are mapped linearly in range of 10% - 90% (the smallest data value will be mapped to 10% duty and the maximum to 90%). Also, the auto-conversion feature, discussed in the *TMS320F2838x Microcontrollers Technical Reference Manual*, supported by the HRPWM module allows to update the fractional duty value of signal in just single register write. The MEP calibration module uses the values in the HRMSTEP and CMPAHR registers to automatically calculate the appropriate number of MEP steps represented by the fractional duty cycle and move the high-resolution PWM signal edge accordingly. For more information, see the *TMS320F2838x Microcontrollers Technical Reference Manual*. This differentiated feature saves critical CPU cycles and improves the encoding latency. The encoding routine snippet is shown below:

```
//
// Function for updating HRPWM duty
//
void configDuty (void) {
    //
    // Calculating normalized digital value and then calculating duty in
    // floating point format based on the normalized value. Then writing the
    // fixed point converted version of duty value to the CMPA register as
    // automatic conversion mode of HRPWM is used
    //
    dig_value_input_norm = dig_value_input * ADC_NORM;
    duty = 0.1f + dig_value_input_norm * 0.8f;
    HWREG(EPWM1_BASE + HRPWM_O_CMPA) = (uint32_t) ((duty* (float32_t)time_period)
                                    *(float_t) ((uint_32) 1 << 16) + 0.5f);
}
```

The decoding routine at the receiver is also implemented using an interrupt-based mechanism. As per the discussed capture sequence in Section 2.2, the second rising edge of transmission signal triggers an ISR where the duty ratio of the relative count values is computed. The constant offset is subtracted from the duty value and then the normalized data value is decoded based on the linear mapping equation. This normalized value is then scaled as per the user-configured bit resolution (see Figure 6). TMU/FPU based intrinsics are used for division/modulo operations in order to accelerate the decoding routine and minimize the latency. The decoding sequence is shown below:

```
//
// Calculating duty value_comp
//
duty_output = __divf32 ((float32_t)absCountOn1, (float32_t) absCountPeriod1);

//
// Removing constant offset
//
duty_output_minus_offset = duty_output - duty_offset;

//
// Decoding normalized output (i.e. between 0 and 1) using duty value_comp
// For duty value, 'x', the decoded normalized output will begin
// (x-0.1)/0.8 i.e. (x-1)*1.25
//
duty_output_norm = (duty_output_minus_offset - 0.1f) * 1.25f;

//
// Scaling the normalized output with the desired bit resolution
//
duty_value_output = (uint16_t) (dig_value_output_norm * (uint16_t)(1 < < BIT_RESOLUTION) + 0.5f);
```

3.4 Offset Calibration

As discussed in Section 2.2, the receiver decoded raw value has some fixed offset, thus one-time system calibration is required. Although not much variance values have been seen in the offset values from device-to-device, that is why a pre-calibrated offset value is already provided in the receiver code. But, in case you find some discrepancy in the performance with the pre-configured offset value, it is recommended to use the optional calibration code provided along with the device loopback example. Follow the calibration steps specified in the calibration_loopback.c to compute the offset value.

4 Experimental Setup and Results

The proposed data transfer scheme based on HRCAP can be utilized for board-to-board communications across an isolation barrier and even for direct communication (without isolation). The experimental setup (shown in Figure 4) is used for validating the proposed technique, which consists of a direct single line connection between F2838x and F28004x controlCARD evaluation modules without isolation. The proposed scheme is tested for different frequencies and the performance metrics achieved with this experimental setup are measured in terms of ENOB and total transmission latency. Table 1 shows the performance metrics at different frequencies. You can refer to this table while choosing appropriate transmission PWM frequency based on your ENOB and latency requirements. The transmission latency consists of time taken by transmission PWM signal to travel to the receiver (time period) added with the time taken by decoding ISR at receiver to decode the data. Follow the directions included in the header section of the examples to add necessary watch variables to the expression window for evaluation. The snapshot of the expression windows of both transmitter and receiver CCS projects for manual data inspection mode are shown in Figure 5 and Figure 6. Also, the plot of decoded data at the receiver for ADC data transfer mode is shown in Figure 7, where the external sinusoid signal of 1 Khz is sampled at 25 Khz by transmitter and the transmission PWM frequency is 200 Khz. As the configured sampling frequency is less than the transmission frequency, the same data element is received multiple times at the receiver, which accounts for the zero slope lines in the plot.

Table 1. Performance Metrics at Different PWM Transmission Frequencies

PWM Transmission Frequency	Effective Number of Bits (ENOB)	Transmission Latency
100 KHz	11.3	11.2 µs
200 KHz	11	6.2 µs
500 KHz	10.4	3.2 µs
800 KHz	9.83	2.45 µs

NOTE: Table 1 shows the example results achieved with the discussed experimental setup. Introducing isolators or CMOS buffers/transceivers in the setup can provide dissimilar rise and fall times, which can need to comprehended and compensated for in the final system implementation .

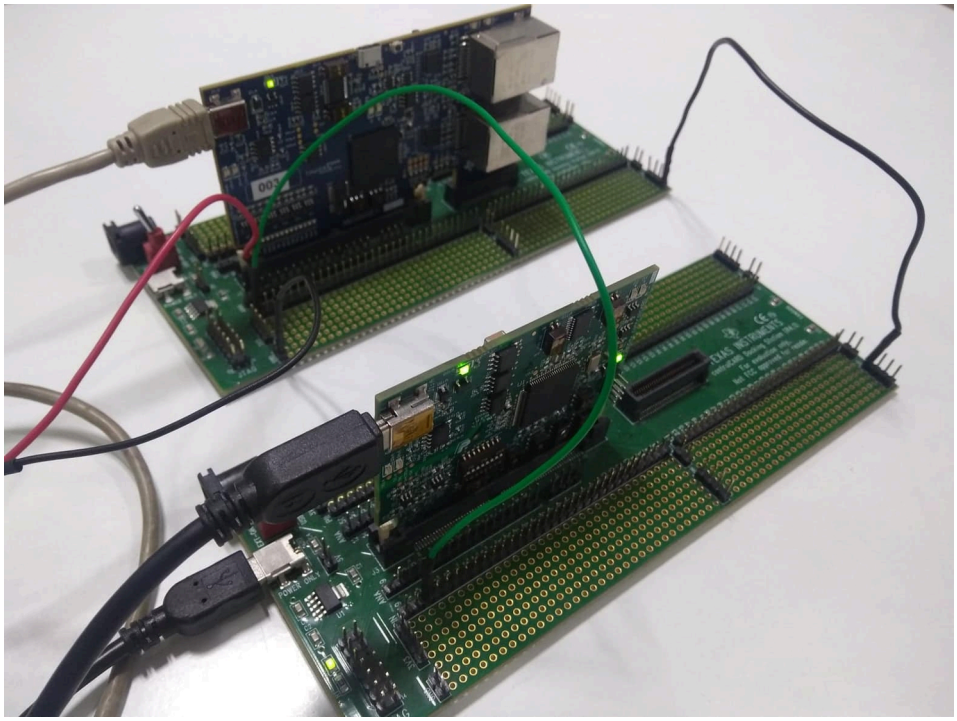

Figure 4. Test Setup for Cross Device Data Transfer

Variables | Expressions | Registers | Breakpoints

Expression	Type	Value
dig_value_output	unsigned int	680
duty_output	float	0.365929544
duty_output_minus_offset	float	0.365609556
offset	float	0.000159999996
dig_output_buff	unsigned int[500]	[680,680,680,680,680...]

Figure 5. Transmitter (F2838x) CCS Expression Window

Variables | Expressions | Registers

Expression	Type	Value
dig_value_input	unsigned int	680
dig_value_input_norm	float	0.33203125
duty	float	0.365624994
MEP_ScaleFactor	int	66
time_period	unsigned int	500
Add new expression		

Figure 6. Receiver (F28004x) CCS Expression Window

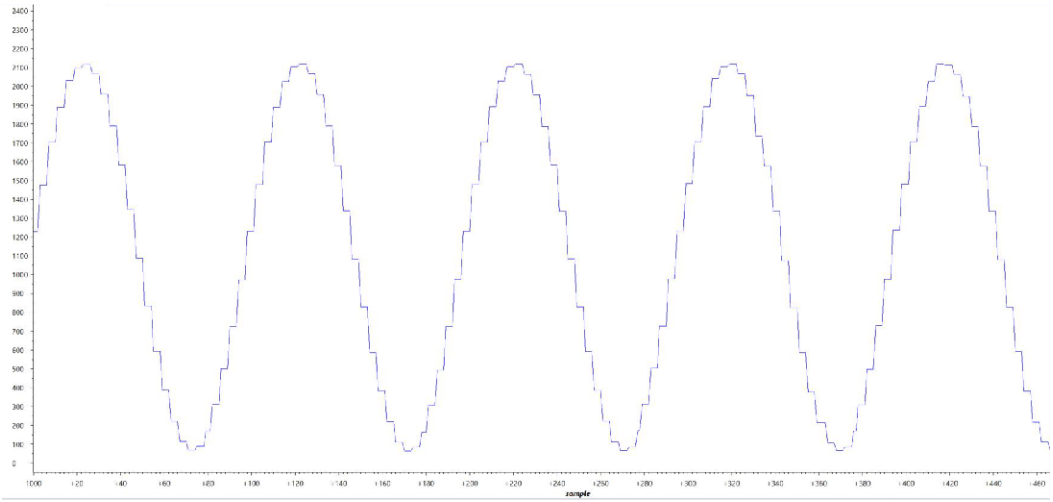

Figure 7. CCS Graph Plot of Decoded Data at Receiver (F28004x)

5 Summary

The proposed data transfer technique leverages the High Resolution Capture (HRCAP) module of C2000 for communicating reasonably high resolution data with low latency just using a single wire. Compared with the normal resolution capture unit (eCAP), HRCAP allows the system to operate at significantly high transmission frequencies, which is reflected in the transmission latencies shown in Table 1. Also looking at Table 1, clearly there is a trade-off between ENOB and transmission latency because at very high transmission frequencies, the transmission latency is less but the ENOB achieved is low. While at lower transmission frequencies, ENOB achieved is relatively high but the transmission latency is large. The transmission latency is less but the ENOB achieved is also low, while at lower transmission frequencies, ENOB achieved is relatively high but the transmission latency is large. Thus, based on the application, you need to choose the appropriate transmission frequency. With all of the other communication peripherals already existing in the C2000 devices, the proposed single wire technique offers an additional data transfer mechanism, which would be very handy for communication across isolation offering a significant cost advantage. Comparing this technique with the similar clock frequency (hundreds of KHz) communication mechanisms like inter-integrated circuit (I2C) and universal asynchronous receiver/transmitter (UART), the transmission latency is quite low as it encodes the entire data packet (10 to 12 bits) in a single pulse, unlike those standard serial protocols where unit pulse is used for encoding only a single bit. The demo examples showcasing this technique are available in C2000Ware software and support user-configurable parameters that will allow customers to quickly evaluate the example for any desired configuration.

6 References

- Texas Instruments: *TMS320F2838x Microcontrollers With Connectivity Manager Data Sheet*
- Texas Instruments: *TMS320F2838x Microcontrollers Technical Reference Manual*
- Texas Instruments: *TMS320F28004x Piccolo™ Microcontrollers Data Sheet*
- Texas Instruments: *TMS320F28004x Microcontrollers Technical Reference Manual*
- C2000Ware for C2000 MCUs: http://www.ti.com/tool/C2000WARE

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265