Application Note

The Essential Guide for Developing With C2000™ Real-Time Microcontrollers

Matthew Pate

ABSTRACT

Performance, efficiency, flexibility and protection – these are the attributes paramount to power electronics technologies, such as motor control, digital power, renewable energies, lighting, and electrical vehicles. Backed by over 20 years of working with customers developing real-time control applications, the C2000™ real-time Microcontroller (MCU) platform enables developers to cost-efficiently meet all of the above criteria while also differentiating their designs. This application report is intended to provide a deeper look into the components providing differentiation to Real-Time Control Systems and give the next steps for evaluation.

Table of Contents

List of Figures

List of Tables

Trademarks

C2000™ and Code Composer Studio™ are trademarks of Texas Instruments.
Bosch™ is a trademark of Robert Bosch GmbH, Germany.
Arm® is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.
All trademarks are the property of their respective owners.

1 C2000 and Real-Time Control

Starting in 1997, Texas Instruments integrated flash memory, an Analog-to-Digital Converter (ADC), a Digital Signal Processor (DSP) , and Pulse Width Modulation (PWM) units on a single device. The first C2000 real-time control MCU was born.

Over more than two decades, this device family grew and now millions of C2000's can be found in numerous applications in industrial and automotive applications like Motor Control, Solar Inverters, Digital Power, Electrical Vehicles and more (see Figure 1-1). There is one tie that binds all the above applications; their real-time nature and the need for a real-time controller.

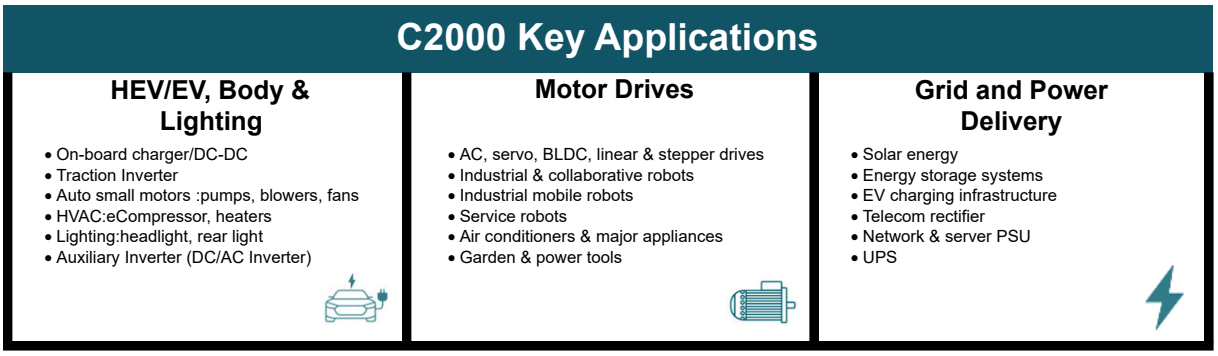


Figure 1-1. Common C2000 Real-Time Control Applications

A real-time control system is typically composed of four main elements: see Figure 1-2:

- **Sensing:** or feedback acquisition. The application needs to measure several key parameters (voltage, current, motor speed, temperature) in an accurate manner and at a very precise moment in time.
- **Processing:** Use the sensing information to apply control algorithms to the incoming data and calculate the next output command.
- **Control:** The command is applied to the system, typically via a PWM unit driving the power electronics system, for example, the motor turns faster, the current to the solar installed system is reduced, the car is accelerating.
- **Interface:** The ability of the device to communicate to other external components. While not necessarily involved in the control of the system, communications to other system components also has to co-exist with the main control loop.

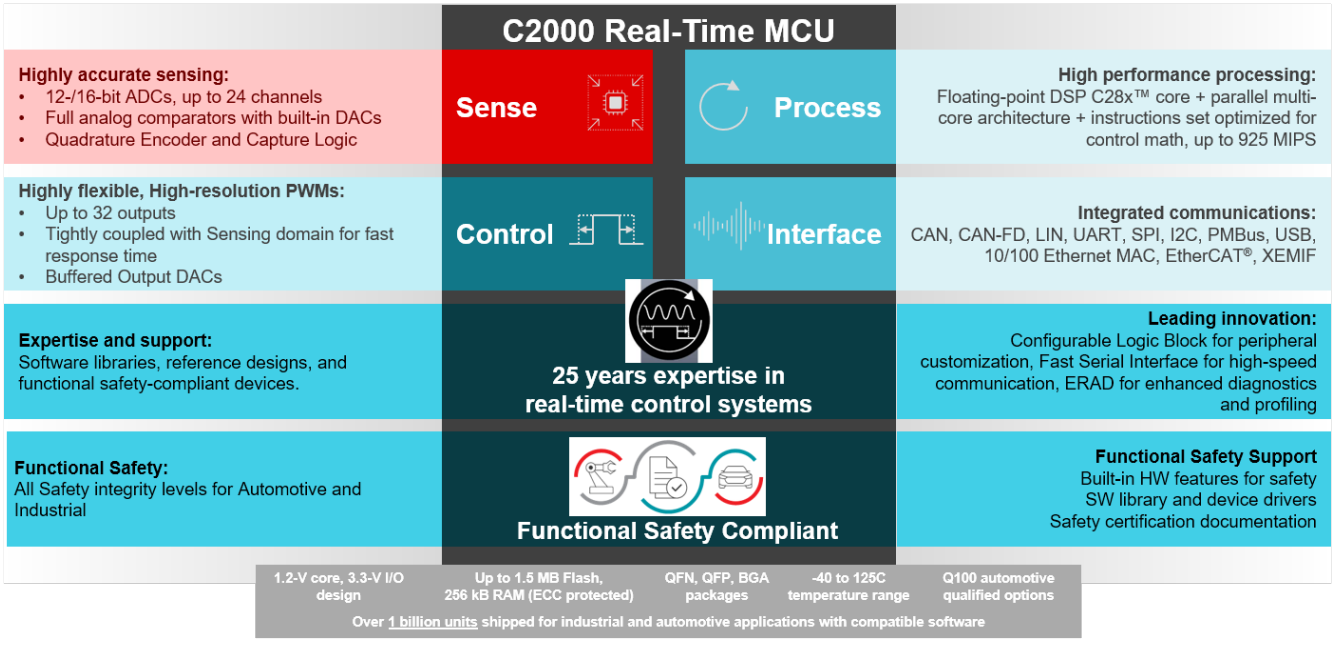


Figure 1-2. C2000 Real-Time MCU Components

The key to real-time control is to minimize the time between Sensing, Processing and Control: this time is defined as the Real-Time Signal Chain. Figure 1-3 illustrates how the entire process is critical to understand the overall system performance of a real-time controller, vs simply looking at the time it takes the main processing unit to complete the control algorithm (step 4).

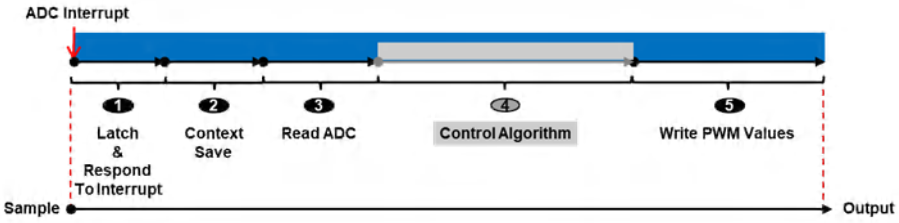


Figure 1-3. Real-Time Signal Chain Components

Many benchmarks, only focus on the time is takes to complete the step 4, typically expressed in Millions Instructions Per Second (MIPS) while the real-time challenge for designers is to minimize the time between Sample to Output: the real-time MCU architecture choice is critical.

The C2000 real-time MCU is a scalable, ultra-low latency, real-time controller platform designed for efficiency in power electronics, such as high power density, high switching frequencies, GaN and SiC technologies and was designed with the best Real-Time Signal Chain performance in mind and can deliver 2-times more real-time signal chain performance than an Arm®-based architecture.

The following sections showcase these advantages in terms of CPU cycle counts for easy comparison:

- Section 3.1
- Section 3.7

For more detailed information on the advantages C2000 brings to the real-time signal chain, including SW benchmarks, see *Signal Chain Benchmarking - A Demonstration of Optimized Real-time Performance of C2000™ MCU*.

The next sections will zoom in each key elements that enables C2000 MCUs to deliver the best real-time signal performance in the market. Backed up with 25 years of expertise, which has translated in the largest reference design offering for power conversion in the industry, coming with robust production ready software and open source hardware documentation, designers can now innovate to and build energy system of the future. The C2000 real-time MCU continues to expand with a platform of software compatible device from the low-end to the high-end. Check the home page on TI.com and register for the TI newsletter to stay up to date on new innovations from C2000 real-time MCUs.

1.1 Getting Started Resources

Hardware

- **Select Part**
  - Device Selection Guide
  - Peripheral Reference Guide
- Pick a C2000 EVM
  - Launchpads: Low cost evaluation board
  - controlCARDs: Full-featured development board
  - Buy an EVM to get started
- Reference Designs

Software/Tools

- SW Overview
- **Tool-Chain**
  - Code Composer Studio
    - Standalone Download
    - Cloud Based
  - SysConfig and PinMux Support
- C2000ware SDK - Low-level drivers and highly optimized libraries
  - Digital Power SDK
  - Motor Control SDKw/ InstaSpin
- **Model-Based Evaluation**
  - MathWorks Embedded Coder

Training

- **Getting Started**
- C2000 Academy
- Motor Control Workshop Series
- Digital Power Training Series
- EV Training Modules
- Safety Overview

1.2 Processing

As seen in Figure 1-4, the C2000 real-time MCU uses the C28x DSP(Digital Signal Processor) core as the main processing unit. Capable of both 32-bit float or fixed point operations with dedicated instructions tailored to real-time control applications. Additional components of this sub-system are outlined below:

- Section 3.4: A state machine based 32-bit floating point co-processor capable of independent code execution from the main C28x DSP core
- C28x Extended Instructions:
  - Floating Point Unit (FPU): Supporting 32-bit floating point operations and on select devices supports Section 3.3
  - Section 3.1: Provides intrinsic instructions to support common trigonometric math functions commonly found in transfroms and torque loop calculations.
  - Viterbi and CRC Unit(VCU): Reducing cycle count for both Viterbi and Cyclical Redundancy Check(CRC) operations found in complex math equations.

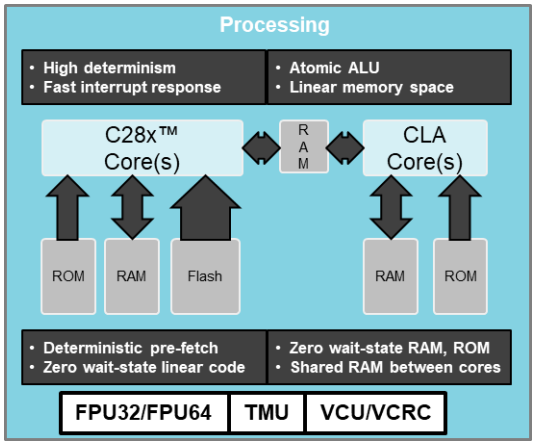


Figure 1-4. C2000 Processing

1.3 Control

The control sub-system includes modules that will stimulate the system under control. Typically, this is done with Pulse Width Modulation (PWM) outputs. This could also be the output of the on-chip Digital-to-Analog Converter (DAC) or just a General-Purpose Input/Output (GPIO) pin.

- PWM – Principle actuation module on the C2000 real-time MCU. Responsible for driving the external Field Effect Transistors (FETs) that exist in most power electronics systems. Supports both standard and high resolution modes for duty-cycle, period, and dead-band control of the waveform
  - All PWMs on a given C2000 have global load capability, ensuring synchronous updates across multiple switch controls
  - High resolution (~150ps) edge placement helps reduce limit cycling in control systems. Available for duty cycle, phase, and deadband placement.
  - Tightly coupled with the analog comparators to provide over voltage/current protectionCPU independent duty control for systems like Peak Current Mode Control as well as CPU independent duty control for systems like Peak Current Mode Control.
- Buffered DAC – 12-bit DAC capable of driving a defined external load. Typically used to create a bias voltage in the analog domain.
- Configurable Logic Block (CLB) – Group of look up tables and state machine logic that operates on internal signal nodes in the hardware domain. Can be an endpoint or intermediary step to realize increased system performance.

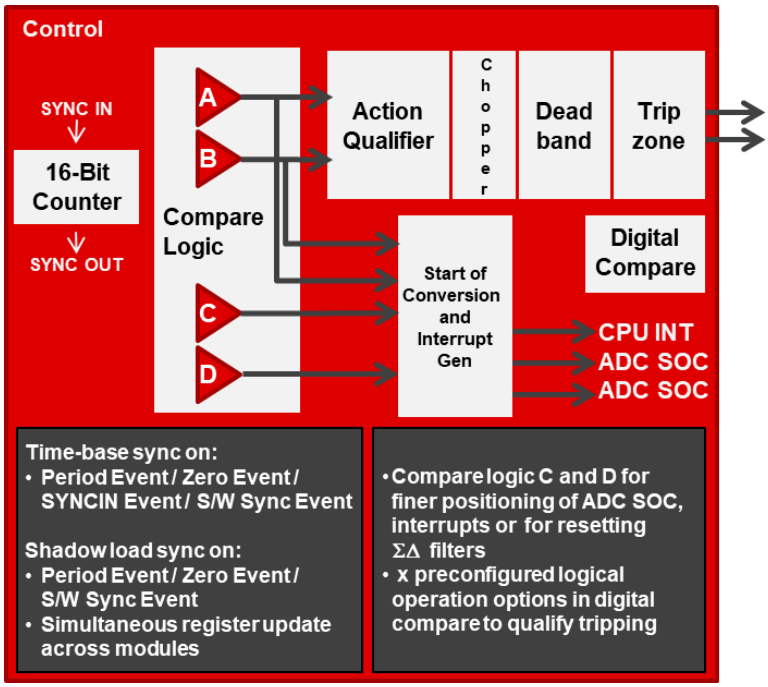


Figure 1-5. PWM Block Diagram

1.4 Sensing

The sensing sub-system includes modules that translate the state of the external system under control (analog domain) into data usable by the C2000 real-time controller (digital domain). Often this is the work of the Analog-to-Digital Converters (ADCs) on the MCU, but could also be handled by comparators or demodulators for external ADCs. Other unit converters are included in this domain, such as quadrature encoders and time pulse measurement devices.

- ADC – Multiple 12 or 16-bit ADCs that are used primarily to convert the voltage or current (through a shunt) of the controlled system into the digital domain. Both an internal reference or external references are supported, with sample rates of 3.5MSPS (12-bit) and 1MSPS (16-bit) to quickly translate the system conditions to information that the control system can act upon.
- Comparator (COMP) – Multiple on-chip comparators provide system protection as well as cycle by cycle PWM control by comparing a system voltage to an internal reference point(generated by the internal 12-bit DACs). Direct connection to the ePWM modules exist to change the output state as quickly as possible without need for CPU intervention.
- Sigma Delta Demodulator – On-chip logic used to decode the serial bit stream output from external sigma delta ADCs. Often used to cross the isolation boundary of high power systems. Includes hardware threshold detection.
- Quadrature Encoder Pulse Measurement (eQEP) – Counts pulses from a variety of encoders to determine motor shaft position. Can be coupled with the CLB module for customer encoder solutions in HW.
- Time Capture (eCAP) – Measures the time duration between external pulse events, useful for evaluating Hall Sensors. Both standard and high resolution modes are available.

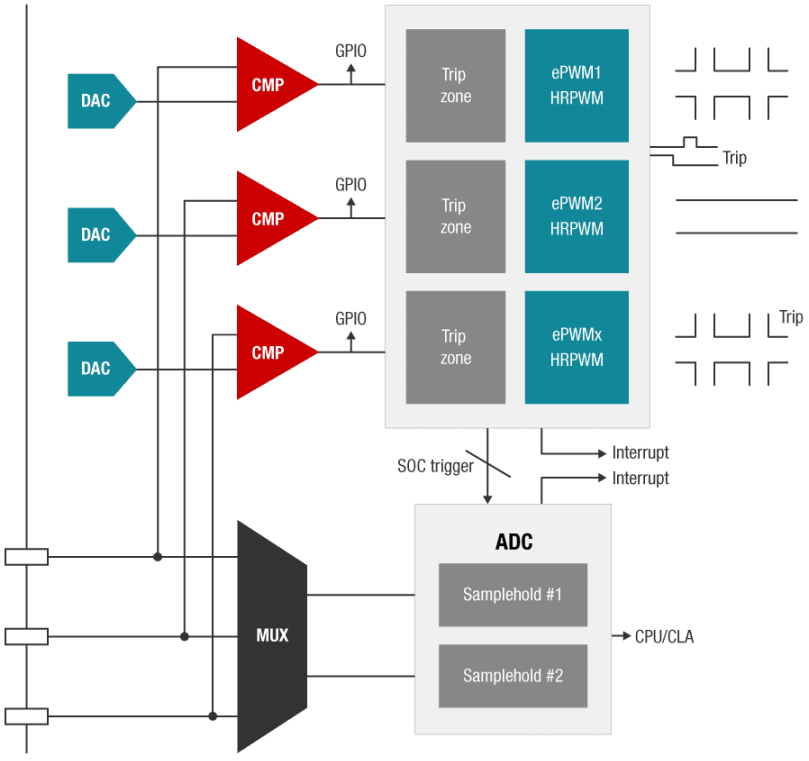


Figure 1-6. C2000 Analog Integration

1.5 Interface

An additional block that, while not typically associated with the real-time control loop of the system, is almost always needed from a system integration point of view. Good integration of the interface block with the CPU is essential to avoid high overhead that could impact the control loop. From serial data streams to multi-channel inputs, as well as industry standard options to proprietary formats, the interface sub-system supports a wide array of communications options.

- Controller Area Network (CAN) – The CAN module supports the Bosch™ CAN protocol standard.
- External Memory InterFace (EMIF) – Parallel data bus typically used to support connections to SDRAM as well as wide bus peripherals.
- EtherCAT Slave Controller (EtherCAT) – This module allows for the C2000 MCU to act as a slave node in an EtherCAT network.
- Ethernet – 10/100 Mbps Ethernet controller and physical interface for external communications across this bus.
- Fast Serial Interface (FSI) – 2 or 3 line simplex serial data transmit or receive. Designed to meet both the high speed (100Mbps) as well as the variable latency introduced when crossing an isolation boundary.
- Host Interface Controller (HIC) - Grants ability for other devices to control/interact with the C2000 peripherals
- Inter-Integrated Circuit (I2C) – Interface/controller for an I2C bus
- Serial Peripheral Interface (SPI) – Interface/controller for a standard Serial Peripheral Interface bus.
- Universal Asynchronous Receiver/Transmitter (UART) – Interface/controller for Universal Asynchronous Receiver Transmitter bus
- Universal Serial Bus (USB) – USB 2.0 MAC and PHY used to interface to standard USB network.

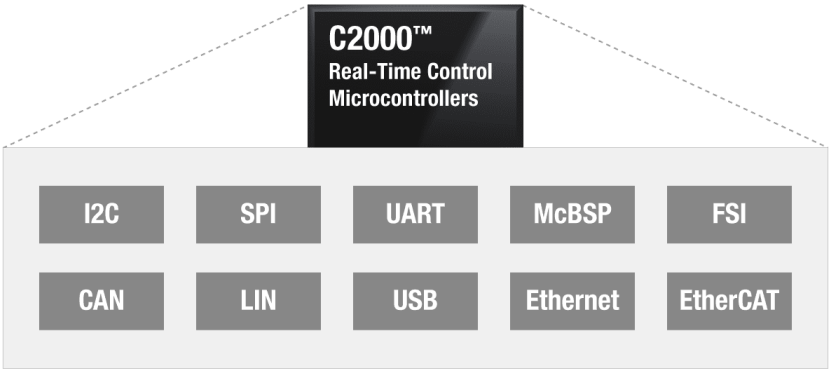


Figure 1-7. C2000 MCU Supported Interfaces

Note

Peripheral counts, as well as features, may vary from device to device. For a complete listing of the number of a peripherals on a device, see the data sheets referenced at the end of each section. For the feature sets supported on a given device, see the *C2000 Real-Time Control Peripherals Reference Guide*.

1.6 Functional Safety

According to the International Electrotechnical Commission (IEC), safety is defined as freedom from unacceptable risk of physical injury or damage to thehealth of people, either directly or indirectly, as a resultof damage to property or to the environment. The IEC defines functional safety as the part of overall safetythat depends on a system or equipment operating correctly in response to its inputs.

With over 300 safety mechanisms defined and independently assessed by TUV SUD for its effectiveness, C2000 MCUs provide the required diagnostic coverage to meet a random hardware capability of SIL 2/ASIL B at a component level. Functional safety manuals provide detailed information on the safety mechanisms, techniques for achieving non-interference between elements and avoiding dependent failures, to aid customers in the development of compliant systems up to SIL 3/ASIL D.

Examples of functional safety enablers across the C2000 device can be seen in Functional Safety Enablers.

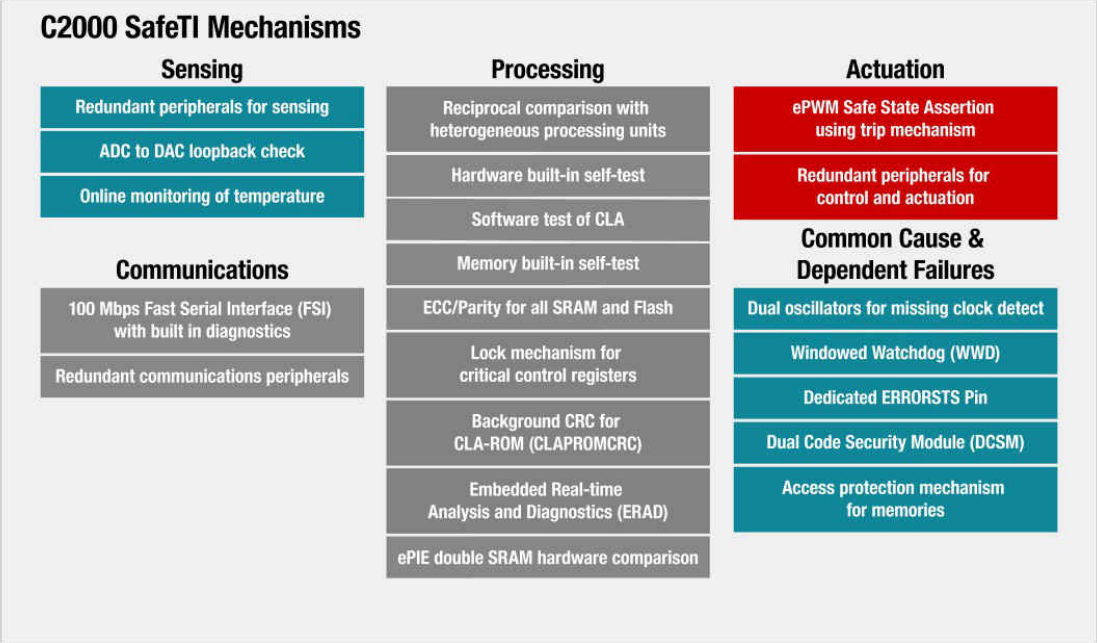


Figure 1-8. Functional Safety Enablers

For more detailed information please see the following technical documents:

- Industrial Functional Safety forC2000™ Real-Time Microcontrollers
- Automotive Functional Safety forC2000™ Real-Time Microcontrollers

2 Sensing Key Technologies

2.1 Accurate Digital Domain Representation of Analog Signals

2.1.1 Value Proposition

Many MCUs have integrated ADCs as part of their sensing subsystem. The ability of the ADC to accurately convert the analog domain to the digital space is one of the most crucial aspects of the MCU in order to realize a proper control system. The data sheet specifications for a C2000 MCU ADC are such that performance in the system can be properly evaluated prior to system implementation.

2.1.2 In Depth

The first step when selecting an MCU for a real-time control system is relatively straightforward process; comparing the components of the MCU to the system needs. There are questions of memory size, CPU speed, communications standards used, analog content, number of I/Os, and so forth. When looking at the fit for an analog module like the ADC, it can appear straightforward to base the decision on sampling rate, number of inputs, and bit level. In practice, however, there is much more to this decision.

Too often ADC selection is based solely on the top level specifications, only to realize during development there are limitations to the system performance due to the ADC itself:

- Will the system be using the analog inputs for frequency analysis? Then, AC specifications like SNR and THD become important to consider when picking an MCU with an on-chip ADC.
- Is overall accuracy a key care about? Looking at the DC specifications like INL, Gain, and Offset are key parameters to consider.
  A quick summary of ADC specifications and their relevance to the system:
  - AC Specifications: Parameters related to how accurately the converter can resolve the fundamental frequency tone of a signal from other noise sources. Includes SNR, SINAD, THD, and SFDR all expressed in dB. Also includes ENOB, which is the SINAD translated into number of bits. Typically SINAD and ENOB based on SINAD are considered when choosing an ADC, the importance will vary depending on the end application.
  - DC Specifications: Parameters related to the accuracy of the converter as it applies to representing an analog input in the digital domain. Includes Gain, Offset, DNL, and INL. The weighted summation of the Gain, Offset, and INL are often referred to as "Total Unadjusted Error" (Equation 1). This equation is typically used to determine the real-world impact of these parameters on the accuracy of a conversion.

$$\sum \sqrt{(Err_{gain})^2 + (Err_{offset})^2 + (Err_{INL})^2} \quad (1)$$

where

- Err_{gain} is the maximum gain error of the ADC in LSBs
- Err_{offset} is the maximum offset error ADC in LSBs
- Err_{INL} is the maximum INL error of the ADC in LSBs

An example of how the C2000 ADC is specified and the parameters can be seen in Table 2-1, a dynamic link to this same table in the data sheet is located here.

One final aspect of all the parameters that C2000 devices list in the data sheet is what is implied by the inclusion of the parameter itself. For parameters that have a MIN/MAX, these are assured specs over the full operational range and lifetime of the device. The typical (TYP) column is also significant for all parameters, as it represents the mean performance of a parameter across its operational range.

Table 2-1. TMS320F28379D 16-Bit ADC Specifications

Parameter	Test Conditions	Min	Typ	Max	Unit
ADC conversion cycles		29.6		31	ADCCLKs
Power-up time (after setting ADCPWDNZ to first conversion)				500	µs
Gain error		–64	±9	64	LSBs
Offset error		–16	±9	16	LSBs
Channel-to-channel gain error			±6		LSBs
Channel-to-channel offset error			±3		LSBs
ADC-to-ADC gain error	Identical V_{REFHI} and V_{REFLO} for all ADCs		±6		LSBs
ADC-to-ADC offset error	Identical V_{REFHI} and V_{REFLO} for all ADCs		±3		LSBs
DNL		> –1	±0.5	1	LSBs
INL		–3	±1.5	3	LSBs
SNR	V_{REFHI} = 2.5 V, f_{in} = 10 kHz		87.6		dB
THD	V_{REFHI} = 2.5 V, f_{in} = 10 kHz		–93.5		dB
SFDR	V_{REFHI} = 2.5 V, f_{in} = 10 kHz		95.4		dB
SINAD	V_{REFHI} = 2.5 V, f_{in} = 10 kHz		86.6		dB
ENOB	V_{REFHI} = 2.5 V, f_{in} = 10 kHz, single ADC		14.1		bits
	V_{REFHI} = 2.5 V, f_{in} = 10 kHz, synchronous ADCs		14.1		
	V_{REFHI} = 2.5 V, f_{in} = 10 kHz, asynchronous ADCs		Not supported		
PSRR	V_{DDA} = 3.3-V DC + 200 mV DC up to Sine at 1 kHz		77		dB
PSRR	V_{DDA} = 3.3-V DC + 200 mV Sine at 800 kHz		74		dB
CMRR	DC to 1 MHz		60		dB
V_{REFHI} input current			190		µA
ADC-to-ADC isolation	V_{REFHI} = 2.5 V, synchronous ADCs	–2		2	LSBs
	V_{REFHI} = 2.5 V, asynchronous ADCs		Not supported		

2.1.3 Device List

- TMS320F2838xD/S
- TMS320F2837xD/S
- TMS320F2807x
- TMS320F28004x
- TMS320F28003x
- TMS320F28002x
- TMS320F280013x
- TMS320F280015x

2.1.4 Hardware Platforms and Software Examples

All controlCARDs for their specific C2000 MCU have been verified to reproduce the DS specifications for the on-chip ADC

- TMDSCNCD28388D
- TMDSCNCD28379D
- TMDSCNCD280049C
- TMDSCNCD280025
- TMDSCNCD2800137
- TMDSCNCD2800157

2.1.5 Documentation

- Texas Instruments: *TMS320F2838x Real-Time Microcontrollers With Connectivity Manager Data Sheet* (see the *C28x Analog Peripherals* section)
- Texas Instruments: *TMS320F2837xD Dual-Core Microcontrollers Data Sheet* (see the *Analog Peripherals* section)
- Texas Instruments: *TMS320F28004x Microcontrollers Data Sheet* (see the *Analog Peripherals* section)
- Texas Instruments: *TMS320F28003x Microcontrollers Data Sheet* (see the *Analog Peripherals* section)
- Texas Instruments: *TMS320F28002x Microcontrollers Data Sheet* (see the *Analog Peripherals* section)
- Texas Instruments: *TMS320F280013x Microcontrollers Data Sheet*(see the *Analog Peripherals* section)
- Texas Instruments: *TMS320F280015x Microcontrollers Data Sheet* (see the *Analog Peripherals* section)

2.2 Optimizing Acquisition Time vs Circuit Complexity for Analog Inputs

2.2.1 Value Proposition

Control systems have the need to interface with a variety of feedback and monitoring sources. Signal sources in these systems differ in their ability to drive a capacitive input circuit like those typically found in the sample-and-hold (S+H) input circuit of an analog-to-digital converter (ADC). The ADCs on C2000 devices allow the acquisition time of the S+H to be individually configured for each input channel over a wide range. This allows the system to simultaneously interface with a mix of high-performance and low-cost signal sources.

2.2.2 In Depth

The inputs of an ADC are typically modeled as a switched capacitor circuit where the hold capacitor inside the ADC, C_h, needs to be charged from an unknown voltage to a value close to the input voltage during the acquisition time. An example, taken from the TMS320F2837xD device, is shown in Figure 2-1.

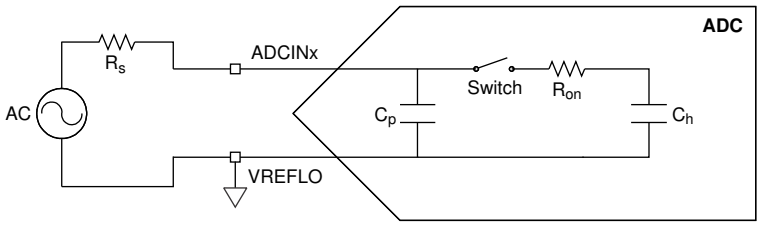


Figure 2-1. Single-Ended Input Model

The required acquisition time for charging C_h is determined by the external impedance of passive components, bandwidth of any buffers or sensors, the internal ADC input parasitics, and the resolution of the ADC.

The system designer can make a variety of trade-offs with respect to external circuit cost and complexity vs settling speed, for example:

- **Adding/upgrading the op-amp buffer driving the ADC inputs:** Lowering acquisition time through better charge transfer to the sample and hold capacitor inside the ADC
- **Increasing the amount of resistance and/or capacitance seen by the ADC input:** Helps reduce noise by adding additional low-pass filtering at the expense of a longer acquisition time
- **Tolerating less accuracy:** Alternatively, using a smaller acquisition window to decrease the sampling time, at the expense of accuracy/resolution.

With all the above possible trade-offs, it is difficult to select a single acquisition time that is appropriate for all analog inputs in the system. C2000 ADCs allow a separate acquisition window to be selected for each channel, giving the system designer a great deal of flexibility to make whatever speed vs signal conditioning circuit cost vs accuracy trade-offs they would like.

The acquisition window (controlled by the ACQPS field of the ADC SOC configuration register) can also be configured over a wide range of values and with a small step size as shown in Table 2-2.

Table 2-2. Range of Acquisition Time Configuration (per Channel)

C2000 MCU	Device SYSCLK	Minimum S+H Time	Maximum S+H Time	S+H Time Configuration Resolution
TMS320F28004x and TMS320F28002x	100 MHz	80 ns	5.1 µs	10.00 ns
TMS320F2807x,TMS320F28003x ,TMS320F280013x, TMS320F280015x	120 MHz	75 ns	4.3 µs	8.33 ns
TMS320F2837xD and TMS320F2837xS	200 MHz	75 ns	2.6 µs	5.00 ns

There are a variety of ways to model the ADC input, Texas Instruments offers free tools to help design the ADC input driver circuit as well as instructional videos on proper front end component selection.

2.2.3 Device List

- TMS320F2838xD/S
- TMS320F2837xD/S
- TMS320F2807x
- TMS320F28004x
- TMS320F28003x
- TMS320F28002x
- TMS320F280013x
- TMS320F280015x

2.2.4 Hardware Platforms and Software Examples

- TMDSCNCDF28388D
- TMDSCNCDF28379D
- TMDSCNCDF280049C
- TMDSCNCD280039C
- TMDSCNCD280025
- TMDSCNCD2800137
- TMDSCNCD2800157
- F2838xD ADC SW Example
- F2837xD ADC SW Example
- F28004x ADC SW Example
- F28002x ADC SW Example
- F280013x ADC SW Example
- F280015x ADC SW Example

2.2.5 Documentation

- *Charge-Sharing Driving Circuits for C2000 ADCs*
- *ADC Input Circuit Evaluation for C2000 MCUs*
- SAR ADC Input Driver Design
- TI Precision Labs - ADCs: Introduction to SAR ADC Front-End Component Selection
- *TMS320F2838xD Real-Time Microcontrollers Technical Reference Manual* (for more information, see the *Choosing an Acquisition Window Duration* section)
- *TMS320F2837xD Real-Time Microcontrollers Technical Reference Manual* (for more information, see the *Choosing an Acquisition Window Duration* section)
- *TMS320F28004x Real-Time Microcontrollers Technical Reference Manual* (for more information, see the *Choosing an Acquisition Window Duration* section)
- *TMS320F28003x Real-Time Microcontrollers Technical Reference Manual* (for more information, see the *Choosing an Acquisition Window Duration* section)

- *TMS320F28002x Real-Time Microcontrollers Technical Reference Manual* (for more information, see the *Choosing an Acquisition Window Duration* section)
- *TMS320F280013x Real-Time Microcontrollers Technical Reference Manual* (for more information, see the *Choosing an Acquisition Window Duration* section)
- *TMS320F280015x Real-Time Microcontrollers Technical Reference Manual* (for more information, see the *Choosing an Acquisition Window Duration* section)

2.3 Hardware Based Monitoring of Dual-Thresholds Using a Single Pin Reference

2.3.1 Value Proposition

C2000 MCUs help to mitigate the cost and complexity of using multiple comparators to monitor a feedback signal by providing single pin access to two embedded voltage comparators per each Comparator Subsystem (CMPSS) module.

2.3.2 In Depth

Control systems commonly use voltage comparators to monitor feedback signals for threshold crossing events. These crossing events can represent a variety of states that range from nominal to critical conditions. A single feedback signal may sometimes be monitored by multiple comparators in order to trigger a custom response for each state.

Consider a simple hysteresis controller (Figure 2-2) that uses two threshold levels to define the actuation on-off behavior:

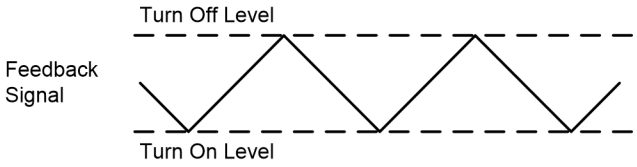


Figure 2-2. Threshold Levels in a Hysteresis Controller

1. A low-level "floor" threshold defines when the actuation should turn on, and
2. A high-level "ceiling" threshold defines when the actuation should turn off

A comparator-based monitoring and triggering scheme for this hysteresis controller can be implemented using a single CMPSS pin as shown in Figure 2-3. Similarly, a second CMPSS pin can be used to detect both over-voltage and under-voltage fault conditions for system protection.

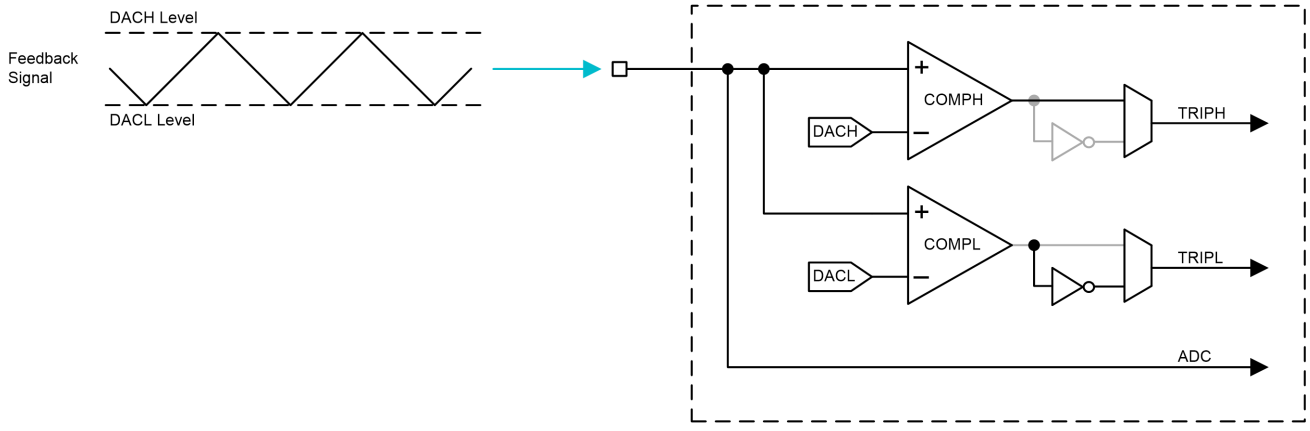


Figure 2-3. CMPSS Block Diagram

Each CMPSS comparator is provided with its own voltage reference DAC, output conditioning logic, and unique trip signals for independent operation. Additionally, each CMPSS pin is also assigned to an ADC channel that can sample the pin voltage in parallel with comparator monitoring. These ADC samples can be used to influence sophisticated system behaviors, and to serve as a redundant form of voltage monitoring.

The multi-function capability of the CMPSS pin demonstrates a significant advantage in resource optimization over other embedded solutions with dedicated pin functionality; the optimization advantage is even greater when compared to discrete solutions that require localized resources like power supplies and reference voltages. System cost and complexity can be reduced by utilizing the full resources available from each C2000 MCU pin.

2.3.3 Device List

- TMS320F2838xD/S
- TMS320F2837xD/S
- TMS320F2807x
- TMS320F28004x
- TMS320F28003x
- TMS320F28002x
- TMS320F280013x
- TMS320F280015x

2.3.4 Hardware Platforms and Software Examples

- TIDM-DC-DC-BUCK
- TIDM-DC-DC-BUCK Example SW
- TIDM-02002
- TIDM-02002 Example SW
- TIDM-1022
- TIDM-1022 Software Example
- TMDXIDDK379D
- TMDXIDDK379D Software Examples

2.3.5 Documentation

- *TMS320F2838x Real-Time Microcontrollers With Connectivity Manager Data Sheet* (for more information, see the *Comparator Subsystem (CMPSS)* chapter)
- *TMS320F2837xD Dual-Core Microcontrollers Data Sheet* (for more information, see the *Comparator Subsystem (CMPSS)* section)
- *TMS320F28004x Microcontrollers Data Sheet* (for more information, see the *Comparator Subsystem (CMPSS)* section)
- *TMS320F28003x Microcontrollers Data Sheet* (for more information, see the *Comparator Subsystem (CMPSS)* section)
- *TMS320F28002x Real-Time Microcontrollers Data Sheet* (for more information, see the *Comparator Subsystem (CMPSS)* section)
- *TMS320F280013x Real-Time Microcontrollers Data Sheet* (for more information, see the *Comparator Subsystem (CMPSS)* section)
- *TMS320F280015x Real-Time Microcontrollers Data Sheet*(for more information, see the *Comparator Subsystem (CMPSS)* section)
- *TMS320F2838xD Real-Time Microcontrollers Technical Reference Manual* (for more information, see the *Comparator Subsystem (CMPSS)* chapter)
- *TMS320F2837xD Real-Time Microcontrollers Technical Reference Manual* (for more information, see the *Comparator Subsystem (CMPSS)* chapter)
- *TMS320F28004x Real-Time Microcontrollers Technical Reference Manual* (for more information, see the *Comparator Subsystem (CMPSS)* chapter)
- *TMS320F28003x Real-Time Microcontrollers Technical Reference Manual* (for more information, see the *Comparator Subsystem (CMPSS)* chapter)
- *TMS320F28002x Real-Time Microcontrollers Technical Reference Manual* (see the *Comparator Subsystem (CMPSS)* chapter)
- *TMS320F280013x Real-Time Microcontrollers Technical Reference Manual* (see the *Comparator Subsystem (CMPSS)* chapter)
- *TMS320F280015x Real-Time Microcontrollers Technical Reference Manual*(see the *Comparator Subsystem (CMPSS)* chapter)

2.4 Resolving Tolerance and Aging Effects During ADC Sampling

2.4.1 Value Proposition

The ADC result is often manipulated mathematically before its used in the control law of a given system. This is typically done with some additional operations by the CPU, adding increased latency to the system as well as loading the CPU for such operations. The C2000 MCU as the ability to correct for this in hardware with no CPU overhead and no impact to ADC sample rate.

2.4.2 In Depth

Prior to using the ADC result in control calculations, it is often necessary to remove any known offset introduced by external factors such as component tolerances or layout differences (Table 2-3). While aspects of the above issues can be partially addressed through PCB layout or choosing higher tolerance/stable resistors there are always deviations from the ideal.

Table 2-3. Typical Resistor Tolerance Over Time and System Impact

Life Cycle Stage	Total Tolerance	Associated 12-Bit Error
Purchase	±0.05%	±2 LSBs
Post Assembly	±0.5%	±20 LSBs
Post Storage/Moisture	±0.75%	±30 LSBs
Temp Coeff and EOL	±1.00%	± 40 LSBs

C2000 MCUs implement an integrated hardware block to correct up to a 10-bit signed value co-incident to the ADC conversion process, saving valuable cycles in the system. The cycle value to the system is effectively doubled, as the ADC sample rate is maintained and no CPU cycles are used to perform the correction. Saturation is built in as well.

For the implementation of the offset correction in addition to the other modules included in the ADC Post Processing Block, see Figure 2-4.

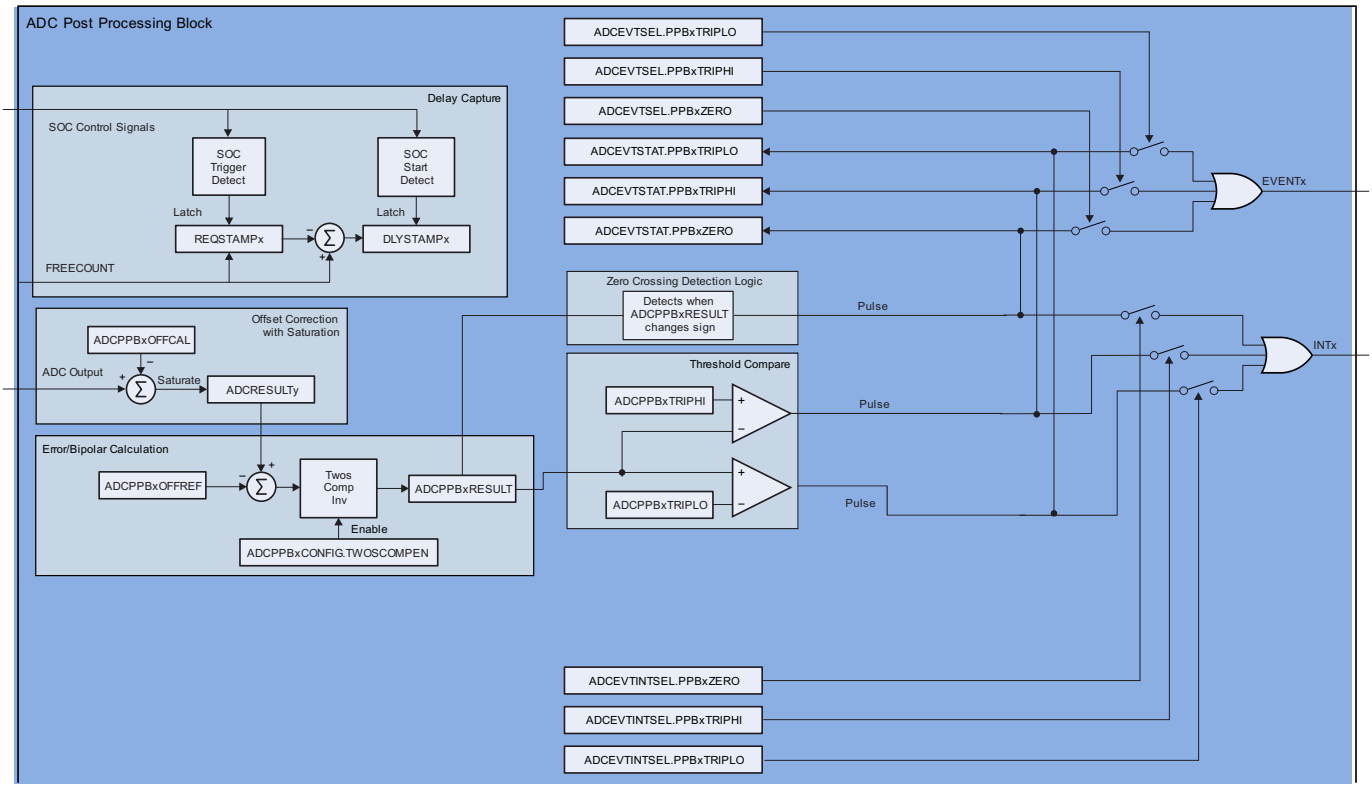


Figure 2-4. ADC Post Processing Block on TMS320F2837xD

2.4.3 Device List

- TMS320F2838xD/S
- TMS320F2837xD/S
- TMS320F2807x
- TMS320F28004x
- TMS320F28003x
- TMS320F28002x
- TMS320F280013x
- TMS320F280015x

2.4.4 Hardware Platforms and Software Examples

- TMDSCNCDF28388D
- TMDSCNCDF28379D
- TMDSCNCDF280049C
- TMDSCNCD280039C
- TMDSCNCD280025
- TMDSCNCD2800137
- TMDSCNCD2800157
- F2838xD ADC PPB SW Example
- F2837xD ADC PPB SW Example
- F28004x ADC PPB SW Example
- F28003x ADC PPB SW Example
- F28002x ADC PPB SW Example
- F280013x ADC PPB SW Example
- F280015x ADC PPB SW Example

2.4.5 Documentation

- *TMS320F2838xD Real-Time Microcontrollers Technical Reference Manual* - see the *Post-Processing Blocks* section in the *Analog-to-Digital Converter (ADC)* chapter
- *TMS320F2837xD Real-Time Microcontrollers Technical Reference Manual* - see the *Post-Processing Blocks* section in the *Analog-to-Digital Converter (ADC)* chapter
- *TMS320F28004x Real-Time Microcontrollers Technical Reference Manual* - see the *Post-Processing Blocks* section in the *Analog-to-Digital Converter (ADC)* chapter
- *TMS320F28003x Real-Time Microcontrollers Technical Reference Manual* - see the *Post-Processing Blocks* section in the *Analog-to-Digital Converter (ADC)* chapter
- *TMS320F28002x Real-Time Microcontrollers Technical Reference Manual* - see the *Post-Processing Blocks* section in the *Analog-to-Digital Converter (ADC)* chapter
- *TMS320F280013x Real-Time Microcontrollers Technical Reference Manual* - see the *Post-Processing Blocks* section in the *Analog-to-Digital Converter (ADC)* chapter
- *TMS320F280015x Real-Time Microcontrollers Technical Reference Manual* -see the *Post-Processing Blocks* section in the *Analog-to-Digital Converter (ADC)* chapter

2.5 Realizing Rotary Sensing Solutions Using C2000 Configurable Logic Block

2.5.1 Value Proposition

As mentioned in the article: On-Chip Hardware Customization (Section 4.3), the C2000 Configurable Logic Block (CLB) technology enables systems designers to implement new logic within the C2000 device and eliminate external circuitry. A specific example for a real time control system is an integrated interface to digital rotary encoders; using the CLB, developers have the freedom to integrate an industry-standard encoder communications protocol or an application-specific customized protocol inside the C2000 MCU without additional external circuitry.

2.5.2 In Depth

A digital rotary encoder is a device that converts the position of a shaft to a digital signal. There are two main types of encoders:

- Absolute Encoders (Figure 2-5): The output of an absolute encoder indicates the current angular position in a message sent back to the master as defined by a particular protocol.
- Incremental Encoders: The output of an incremental encoder provides information in a train of modulated pulses that are typically further processed by the system master into information such as speed, distance and position.

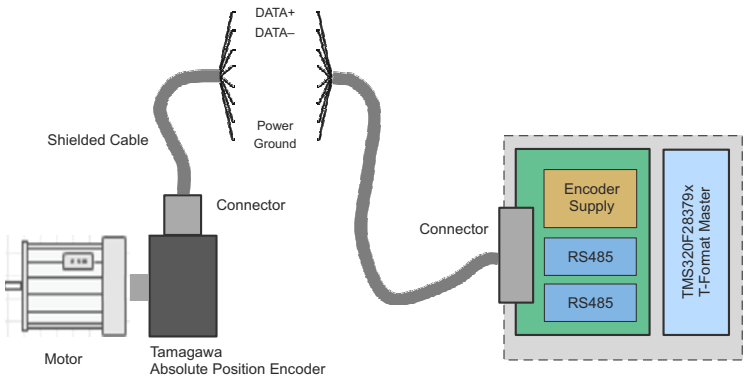


Figure 2-5. Industrial Servo Drive With T-Format Absolute Position Encoder Interface

C2000 CLB technology enables an integrated solution to interface to the most popular digital rotary position encoders, eliminating the necessity for external field-programmable gate arrays (FPGAs) or application-specific integrated circuits (ASICs). The PositionManager BoosterPackplug-in module is a flexible low voltage platform intended for evaluating various encoder interfaces and designed to work with multiple C2000 LaunchPad development kits such as the LAUNCHXL-F28379D, LAUNCHXL-F280049C, or the LAUNCHXL-F280025C.

Below are examples of encoders that are available for evaluation today as part of the Motor Control SDK software package. Future updates are planned to add examples for both the BiSS-C and EnDAT22 protocols.

- T-Format Absolute Encoder Interface:

  The Tamagawa T-Format protocol is a popular digital, bidirectional interface for absolute encoders. The easy-to-use library and example software delivered with TIDM-1011 demonstrates Tamagawa's T-Format standard. In this example, the T-Format absolute encoder interface is integrated into the C2000 using on-chip resources such as the CLB, SPI, and GPIO as shown in Figure 2-5. This TI Design includes the following features:
  - A T-Format encoder interface library and CRC library that implements protocol commands
  - A demonstration project (with full source code access) to exercise the T-Format commands
  - Fully supported by the C2000 CLB Tool integrated inside Code Composer Studio
- T-Format Evaluation in High-Bandwidth Current Loop Applications:

  With the recent C2000 MCUs such as TMS320F2837x and TMS320F28004x, it is possible to implement Fast Current Loop (FCL) algorithms that provide a high current loop bandwidth with the same external hardware as used in classical Field Oriented Control (FOC) methods. TI has developed the FCL algorithm on these MCUs and implemented it on the DesignDRIVE IDDK platform. The T-Format encoder interface has been integrated into an evaluation implementation of FCL algorithms on C2000 devices.

  Quick Response Control of PMSM Using Fast Current Loop studies the frequency response analysis of current loops in real time and also verifies the interface logic for the T-format encoder interface implemented in TIDM-1011. The position loop in this example can be closed using a QEP encoder or a T-format encoder and FCL can be implemented in both cases.

- Pulse Train Output (PTO) QepDiv and PulseGen:

  Incremental rotary encoders output a pulse train to indicate that the shaft being monitored has moved. The system master typically processes this pulse train to determine information such as speed, distance and position. In the QepDiv implementation, position information is sent from the encoder to the Enhanced Quadrature Encoder Pulse (eQEP) module on a C2000 MCU. In the PulseGen case a custom pulse stream is generated to meet the system needs. The PTO-QepDiv example demonstrates how the CLB can be used to generate a divided pulse stream from these eQEP inputs as shown in Figure 2-6. The divided pulse stream can then be sent to another device in the system. Both the QepDiv and PulseGen examples are documented in the *C2000 Position Manager PTO API Reference Guide*.

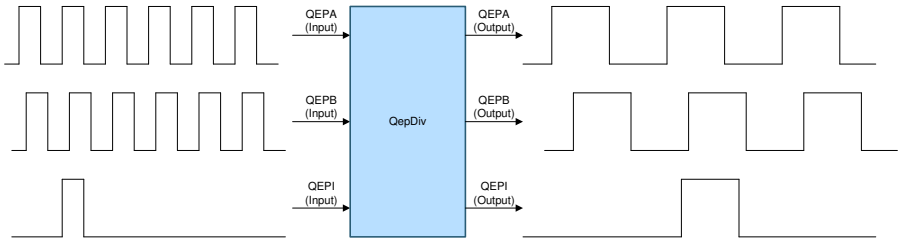


Figure 2-6. QepDIv Input and Output Diagram

2.5.3 Device List

- TMS320F2838xD/S
- TMS320F2837xD/S
- TMS320F2807x
- TMS320F28004x
- TMS320F28003x
- TMS320F28002x

2.5.4 Hardware Platforms and Software Examples

- *Tamagawa T-Format Absolute-Encoder Master Interface Reference Design for C2000™ MCUs*
- *C2000 Position Manager PTO API Reference Guide*
- MotorControl software development kit (SDK) for C2000 MCUs
- Position Manager BoosterPack (BOOSTXL-POSMGR)
- C2000™ DesignDRIVE Development Kit for Industrial Motor Control (TMDXIDDK379D)

2.5.5 Documentation

- Training: How the C2000 Configurable Logic Block (CLB) tool integrates custom logic in my design
- *CLB Tool User's Guide*
- *Designing With the C2000 Configurable Logic Block (CLB)*
- *How to Migrate Custom Logic From an FPGA/CPLD to C2000 Microcontrollers*
- *Fast Current Loop Library*

2.6 Smart Sensing Across An Isolation Boundary

2.6.1 Value Proposition

High voltage control systems often make use of a sigma-delta ADC in order to easily pass the analog information of the high voltage domain to the lower voltage domain where the MCU exists. Before the data stream can be used, it must be processed by a filter and demodulator. This logic exists on the C2000 MCU and is called the Sigma Delta Filter Module (SDFM) module. The high and low comparators inside the SDFM on a C2000 MCU can actuate the PWMs without CPU intervention saving valuable time to better control the system.

2.6.2 In Depth

A sigma-delta type ADC is by design an over-sampling architecture. The ADC itself is a single bit design that over-samples the signal of interest to produce a higher order output. The modulation of this type of converter results in a binary output and, hence, a serial data stream. This serial data stream is then sent to the filter/demodulator on the C2000 MCU for re-construction into a higher bit order digital representation of the sampled signal (Figure 2-7) .

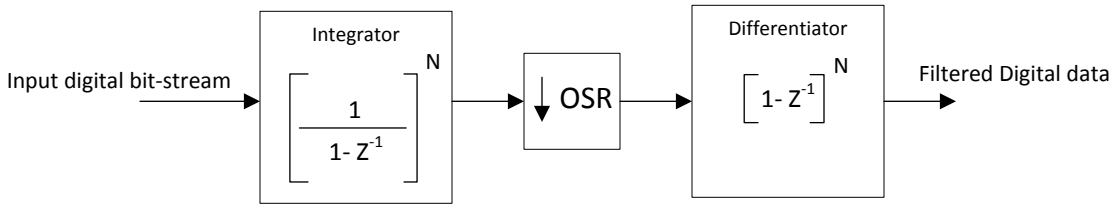


Figure 2-7. Filter and Demodulator Inside the C2000 SDFM

At this point an interrupt is generated to the MCU, informing the other domains there is new data to process and act upon.

The SDFM module on the C2000 MCU is unique in that not only can the converted data be read post filtering, but the PWMs can be switched based on this data automatically. Each SDFM module contains four channels. Inside each channel exists two filters: a primary filter that produces the SDFM data and a secondary filter containing both high and low limit comparators (Figure 2-8). This allows the system to control the PWM signals without waiting for CPU intervention, resulting in both lower latency as well as lower overall CPU utilization.

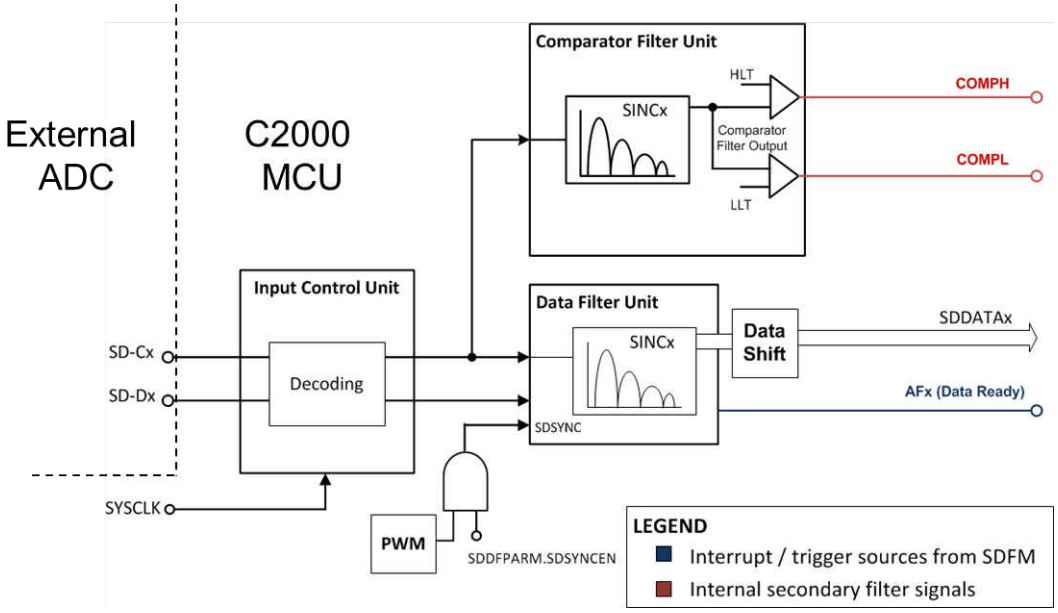


Figure 2-8. SDFM Module With Both Primary and Secondary Filter Blocks on the TMS320F2837xD MCU

2.6.3 Device List

- TMS320F2838xD/S
- TMS320F2837xD/S
- TMS320F2807x
- TMS320F28004x
- TMS320F28003x

2.6.4 Hardware Platforms and Software Examples

- TMDSCNCD28388D controlCARD evaluation module
- LAUNCHXL-F28379D LaunchPad
- LAUNCHXL-F280049C LaunchPad
- TMDSCNCD280039C controlCARD evaluation module
- SDFM_filter_sync_cpuread SW example for F2838xD/S
- SDFM_pwm_sync_cpuread SW example for the F2838xD/S

2.6.5 Documentation

- *TMS320F2838x Real-Time Microcontrollers Technical Reference Manual* (for more information, see the *SDFM* chapter)
- *TMS320F2837xD Real-Time Microcontrollers Technical Reference Manual* (for more information, see the *SDFM* chapter)
- *TMS320F28004x Real-Time Microcontrollers Technical Reference Manual* (for more information, see the *SDFM* chapter)
- *TMS320F28003x Real-Time Microcontrollers Technical Reference Manual* (for more information, see the *SDFM* chapter)
- Real-Time controllers get new connectivity capabilities

2.7 Enabling Intra-Period Updates in High Bandwidth Control Topologies

2.7.1 Value Proposition

One of the most important considerations in designing a real-time control system is what the minimum sample to output delay the system can achieve (Figure 2-9). From the basics of providing stable control of the real-time system to realizing increased system efficiency the on-chip ADCs inside every C2000 real-time microcontroller are designed to minimize their effective time delay to the system in a variety of ways. Looking beyond the basics of a discrete ADCs and comparing to the C2000 on-chip ADCs there are differentiated features as a result of decades of involvement in real-time control.

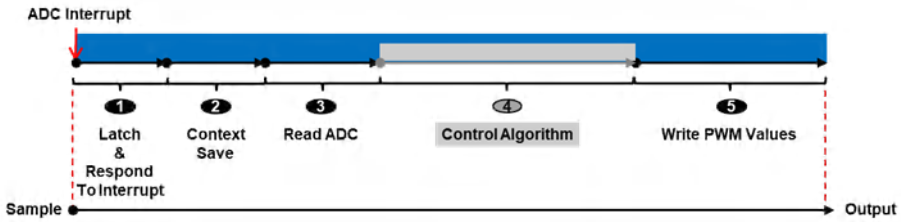


Figure 2-9. Real-Time Signal Chain

2.7.2 In Depth

While choosing a discrete ADC for a system many factors are considered: raw converter speed, DC error terms like gain and offset, and AC error terms like SNR, THD(Shown in:Section 2.1). Beyond those parameters are the aspects of system integration, conversion speed with respect to the control loop, and sampling capability that can make or break a real-time control design.

System integration: One advantage of an integrated ADC is the lack of an external communications bus that adds complexity to the system in both latency of the comms channel but also noise considerations for bus routing. Once the ADC conversion is complete, minimizing the delay into the control scheme of the system is vital. The ADC results are accessible by all CPU cores, CLA cores, as well as DMAs.

Another common delay is introduced due to the latency introduced when the CPU is notified that the ADC conversion is complete. The C2000 ADCs implement a selectable early interrupt that takes advantage of the time the ADC incurs between the start of a sample and the completion of the analog to digital conversion. Since the ADC is integrated the C2000 MCU has the ability to sychronize to either the end of the analog sampling phase or the full conversion complete. This allows the C28x CPU or CLA to continue to do other tasks or calculations in parallel to the ADC conversion and use the ADC conversion as soon as it is available. Not only are wasted cylces minimized, but results in a more stable system control as the converted value used by the CPU is closer in time to the real world signal state.

Conversion Speed: One aspect that is common in the evaluation of any ADC, discrete or integrated, is its conversion speed. This is how long it takes the ADC to convert the sampled analog signal into a digital representation. This is especially important in high bandwidth systems as it is beneficial to update the system intracycle vs waiting for the next full cycle of the control loop to happen (Figure 2-10). This is shown the (Fast Current Loop examples). The ADC on the C2000 MCU is capable of generating a conversion in as little as 260ns after the sample completes. This allows enough time for the C28x or CLA core to calculate the new PWM frequencies well ahead of the next control loop period allowing for the intra-cycle update to happen. The ePWM module plays a role here as well, allowing the PWM to be updated by writing to the active registers vs the shadow resigster, which would pend the update until the next full period.

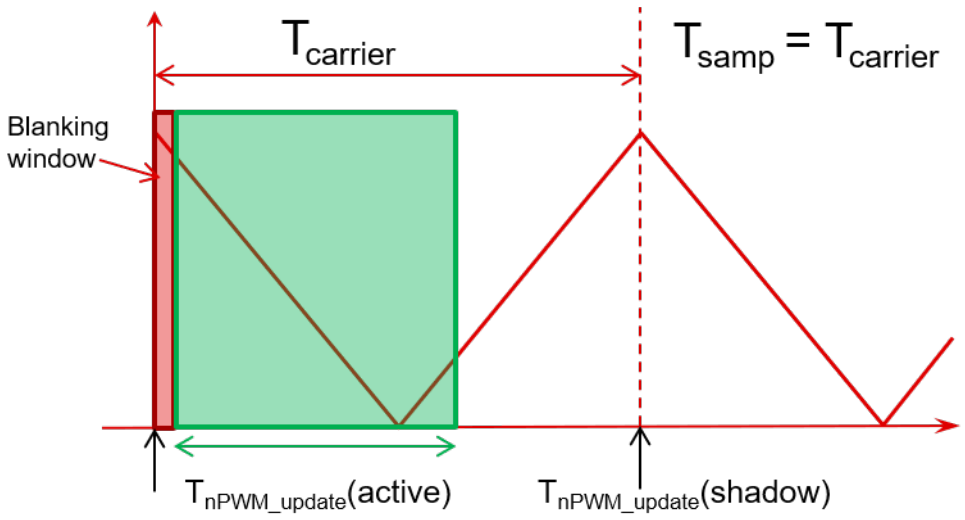


Figure 2-10. Intra-Period Update of PWM

Simultaneous sampling: Each C2000 device referenced in this guide has at least 2 if not more ADCs. Looking at common real-time control topologies this allows simultaneous sampling of the motor currents. For the case of a system that uses a resolver for posistion sensing this would allow both the sine and cosine feedbacks to be sampled at the same point in time. Table 2-4 shows the potential effect on the system if such inputs are not sampled at the same point in time

Table 2-4. Potential Phase Error Caused by Sample Delay

Control Loop Frequency	Phase Error With 500ns Sample Lag	Phase Error With 1µs Sample Lag
10 kHz	1.8 degrees	3.6 degrees
20 kHz	3.6 degrees	7.2 degrees
50 kHz	8.9 degrees	17.8 degrees

2.7.3 Device List

- TMS320F2838xD/S
- TMS320F2837xD/S
- TMS320F2807x
- TMS320F28004x
- TMS320F28003x
- TMS320F28002x
- TMS320F280013x
- TMS320F280015x

2.7.4 Hardware Platforms and Software Examples

- TIDM-02007 Dual-axis motor drive using fast current loop (FCL) and SFRA on a single MCU reference design
- BOOSTXL-3PHGANINV 48-V Three-Phase Inverter With Shunt-Based In-Line Motor Phase Current Sensing Evaluation Module

- BOOSTXL-3PHGANINV examples in C2000Ware Motor Control SDK
- TMDXIDDK379D C2000 DesignDRIVE Development Kit for Industrial Motor Control

2.7.5 Documentation

- *Performance Analysis of Fast Current Loop (FCL) in Servo*
- *Quick Response Control of PMSM Using Fast Current Loop*

2.8 Accurate Monitoring of Real-Time Control System Events Without the Need for Signal Conditioning

2.8.1 Value Proposition

As mentioned in Section 4.4, quick detection of over-shoots and under-shoots is critical to the operation of a real-time control systems. However in reality, the transients of concern often have coupled noise, which can cause false trips and shut downs resulting in reliability issues in system functionality.

One approach typically used to remedy this is external passive RC filters to remove the high frequency noise from the signal of interest. This has drawbacks of BOM cost, PCB space, and slowing down the signal of interest. The CMPSS module on the C2000 MCU has configurable filters integrated in the device to filter out noise events without the need for an external RC filter.

2.8.2 In Depth

The CMPSS filter works on a principle of majority vote. It captures a window of high/low trip events from the comparator and sets the output to high or low depending on the threshold setting. The number of trip events captured by the window and the threshold setting used to determine the output are both configurable. Due to the configurability, the application can trade-off between how much noise to reject and how much the final trip is delayed. Figure 2-11 is a high level illustration of the filter in work. For further details, please refer to the CMPSS chapter in the TRM.

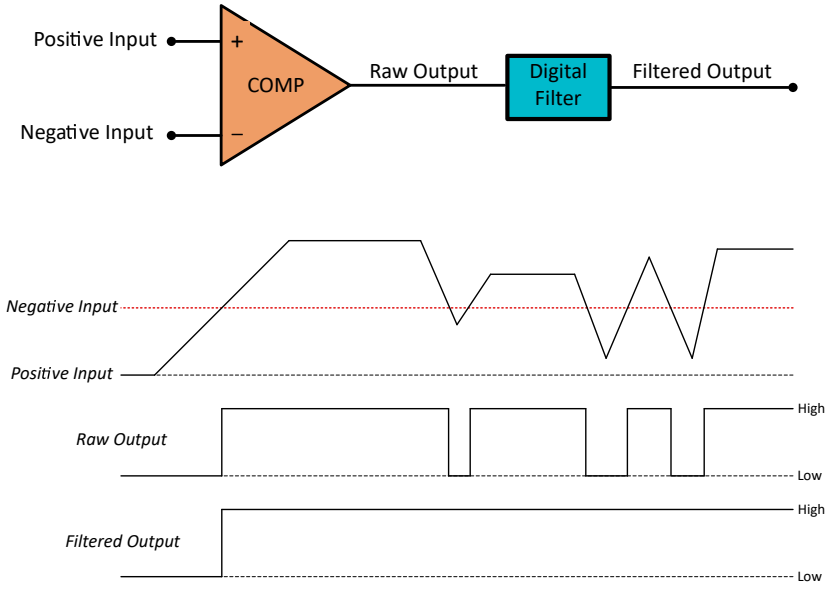


Figure 2-11. CMPSS Raw vs Filtered Output

2.8.3 Device List

- TMS320F2838xD/S
- TMS320F2837xD/S
- TMS320F2807x
- TMS320F28004x
- TMS320F28003x
- TMS320F28002x
- TMS320F280013x

- TMS320F280015x

2.8.4 Hardware Platforms and Software Examples

- TMDSCNCDF28388D
- TMDSCNCDF28379D
- TMDSCNCDF280049C
- TMDSCNCD280039C
- TMDSCNCD280025
- TMDSCNCD2800137
- TMDSCNCD2800157
- F2838xD CMPSS Digital Filter SW Example
- F2837xD CMPSS Digital FilterSW Example
- F28004x CMPSS Digital Filter SW Example
- F28003x CMPSS Digital Filter SW Example
- F28002x CMPSS Digital Filter SW Example
- F280013x CMPSS Digital Filter SW Example
- F280015x CMPSS Digital Filter SW Example

2.8.5 Documentation

- *TMS320F2838x Real-Time Microcontrollers Technical Reference Manual* (for more information, see the *CMPSS* chapter)
- *TMS320F2837xD Real-Time Microcontrollers Technical Reference Manual* (for more information, see the *CMPSS* chapter)
- *TMS320F28004x Real-Time Microcontrollers Technical Reference Manual* (for more information, see the *CMPSS* chapter)
- *TMS320F28003x Real-Time Microcontrollers Technical Reference Manual* (for more information, see the *CMPSS* chapter)
- *TMS320F28002x Real-Time Microcontrollers Technical Reference Manual* (for more information, see the *CMPSS* chapter)
- *TMS320F280013x Real-Time Microcontrollers Technical Reference Manual* (for more information, see the *CMPSS* chapter)
- *TMS320F280015x Real-Time Microcontrollers Technical Reference Manual* (for more information, see the *CMPSS* chapter)

3 Processing Key Technologies

3.1 Accelerated Trigonometric Math Functions

3.1.1 Value Proposition

Trigonometric functions are used heavily in real-time control systems, both in power applications as well as motor control. Park Transforms (Figure 3-1), Space Vector Generation, and resolver angle are a few of these examples that rely on trigonometric math. The Trigonometric Math Unit (TMU) on C2000 MCUs enables an extended instruction set targeted at 32-bit floating-point trigonometry based calculations. This results both faster performance and smaller code size for Trigonometric functions.

$$\text{Park}\begin{bmatrix} i_d \\ i_q \\ i_o \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} X \begin{bmatrix} i_\alpha \\ i_\beta \\ i_o \end{bmatrix}$$

Figure 3-1. Park Transform

3.1.2 In Depth

Many common mathematical techniques in real-time control rely on the use of trigonometric functions: sine, cosine, and arc tangent are all examples. The TMU adds dedicated instructions to the C28x core for these functions as well as their inverse, that supersede the standard C library calls. As shown in Figure 3-2, significant cycle count reductions are possible using the TMU based instructions vs their Fast RTS counterparts.

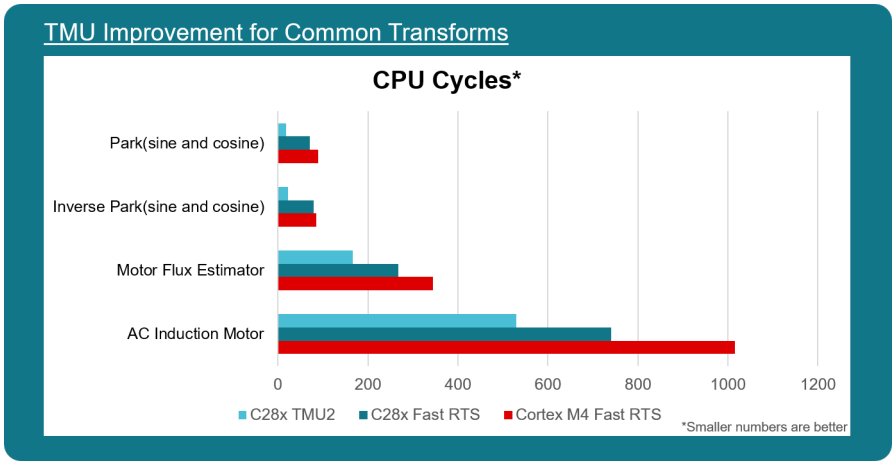


Figure 3-2. TMU Improvement for Common Transforms

There is also single instruction support for both square root as well as floating point division. These are often used in conjunction with the trigonometric functions previously listed. A full list of supported instructions and their cycle counts can be seen in Table 3-1. These instructions are inserted automatically by the C compiler on devices that have a TMU.

Table 3-1. TMU Supported Instructions Summary

Operation	C Equivalent Operation	C28x Pipeline Cycles
Multiply by 2*pi	a = b * 2pi	2 cycles + Sine/Cosine function
Divide by 2*pi	a = b / 2pi	2 cycles + Sine/Cosine function
Divide	a = b / c	5 cycles
Square Root	a = sqrt(b)	5 cycles
Sin Per Unit	a = sin(b*2pi)	4 cycles
Cos Per Unit	a = cos(b*2pi)	4 cycles
Arc Tangent Per Unit	a = atan(b)/2pi	4 cycles
Arc Tangent 2 and Quadrant Operation	Operation to assist in calculating ATANPU2	5 cycles

Note

While the C2000 MCUs listed below all have a TMU module, the C compiler used to generate the target code must have the correct options selected to utilize this hardware. This is controlled in the Processor Options of the C2000 Compiler via the TMU support drop down, as well as using the "relaxed" setting for the floating point mode under C2000 Compiler → Optimizations. Specific TMU based functions can be called explicitly as inline functions in the C source if that is preferable to a global setting.

3.1.3 Device List

- TMS320F2838xD/S
- TMS320F2837xD/S
- TMS320F2807x
- TMS320F28004x
- TMS320F28003x
- TMS320F28002x
- TMS320F280013x
- TMS320F280015x

3.1.4 Hardware Platforms and Software Examples

- TIDM-1007 Interleaved CCM Totem Pole Bridgeless Power Factor Correction (PFC) Reference Design
- TIDM-HV-1PH-DCAC Single-Phase Inverter Reference Design With Voltage Source and Grid Connected Modes
- TMDXIDDK379D C2000 DesignDRIVE Development Kit for Industrial Motor Control
- TMDSHVMTRINSPIN High Voltage Motor Control Kit with InstaSPIN-FOC and InstaSPIN-MOTION enabled for F280049C device lab7 and lab8

3.1.5 Documentation

- *Enhancing the Performance Capabilities of the C2000™ MCU Family*
- *Signal Chain Benchmarking - A Demonstration of Optimized Real-time Performance of C2000™ MCU*
- *TMS320C28x Extended Instruction Sets Technical Reference Manual*

3.2 Fast Onboard Integer Division

3.2.1 Value Proposition

Compared to other standard arithmetic operations, accurate integer division are typically more complex and have much higher cycles counts in a MCU. Additionally in control algorithms, there is need for more linear division operations, such as Euclidean and Modulus that often consume additional CPU overhead. The FID module addresses this by supporting these special division definitions through dedicated HW, which enables control algorithms without the typical additional overhead associated with these methods. Finally, the FID supports various size and signed operands with optimum cycle counts.

3.2.2 In Depth

There are multiple definitions for division and modulo operations according to the programming language and computer science literature, each of these definitions provide different mathematical properties that can be beneficially employed in the application context. Truncated division is the standard division definition widely used in many programming languages like C. In this definition, the remainder will always have the sign of numerator. The transfer function of truncated division is shown in Figure 3-3 , as can be observed that the function is non-periodic since there is a “platform” around zero point.

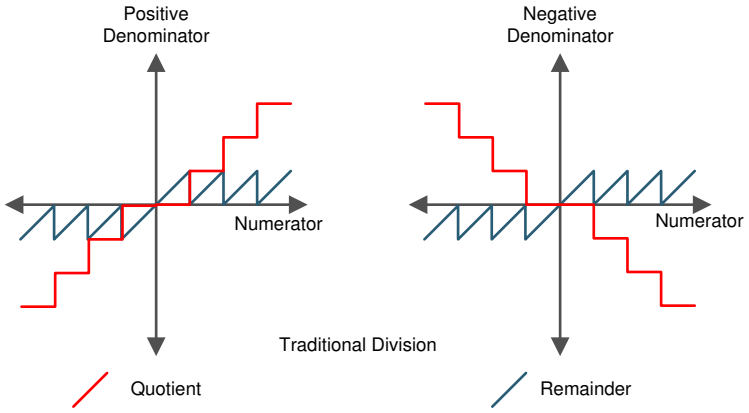


Figure 3-3. Truncated Division Function

Due to the non-linearity around zero point, the function is not preferred in control applications. Thus non-conventional definitions of division and modulo operations are sometimes preferred for better linearity and periodicity. The transfer function of Floored/Modulo (Figure 3-4) and Euclidean division (Figure 3-5) functions are shown below. In the modulo division function the remainder will always have the sign of denominator, thus the function is linear around the zero point. In the Euclidean division, the remainder is always positive so the division function is linear around zero point and also the modulo function is periodic.

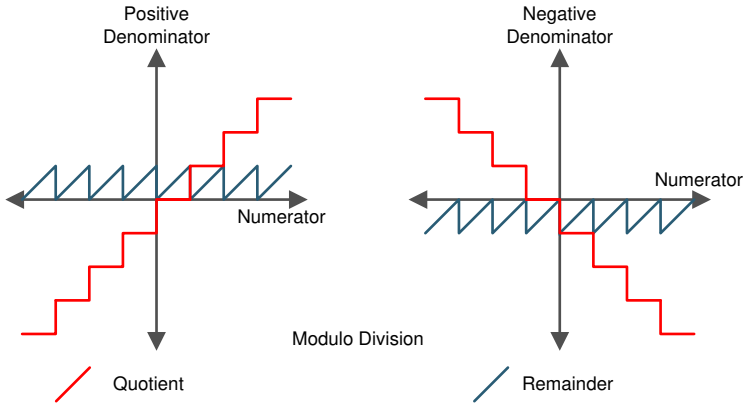


Figure 3-4. Floored Division Function

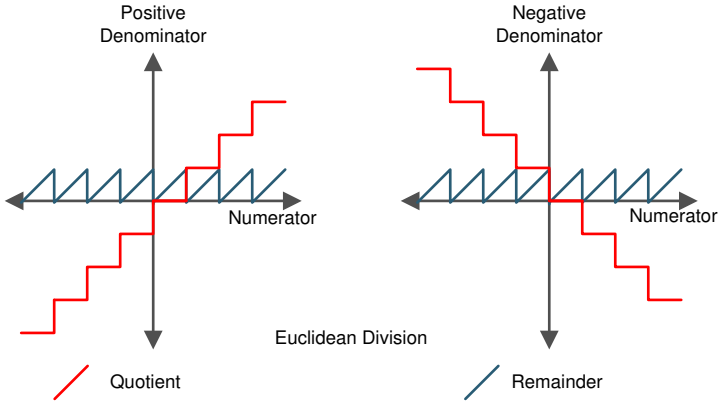


Figure 3-5. Euclidean Division Function

The C28x CPU has added specialized instructions to enable applications to implement the above division and modulo definitions efficiently in hardware. These new instructions used to enable integer division are interruptible, have very low latency and support different types of operations (ui32/ui32, i32/ui32, i64/i32, ui64/ui32, ui64/ui64, i64/i64, and so forth). The cycles count for the different types of division operations and sizes of the operands achieved using Fast Integer Division unit are listed in and are also compared with the cycles without FID module in Table 3-2. As evident from the table, the FID unit provides several folds improvement in performance for different types of division operations which helps in minimizing the latency of control loop calculations.

Table 3-2. Integer Division With and Without the FID Module

Division Operation	Using C Operator '/' Without FASTINTDIV Hardware on C28x	Using Intrinsics With FASTINTDIV Hardware + C28x	Improvement Factor
i16/i16 traditional	52	16	3.3
i16/i16 Euclidean	56	14	4.0
i16/i16 Modulo	56	14	4.0
u16/u16	56	14	4.0
i32/i32 traditional	59	13	4.5
i32/i32 Euclidean	63	14	4.5
i32/i32 Modulo	63	14	4.5
i32/u32 traditional	37	14	2.6
i32/u32 Modulo	41	14	2.9
u32/u32	37	12	3.1
i32/i16 traditional	60	18	3.3
i32/i16 Euclidean	64	16	4.0
i32/i16 Modulo	64	16	4.0

Table 3-2. Integer Division With and Without the FID Module (continued)

Division Operation	Using C Operator '/' Without FASTINTDIV Hardware on C28x	Using Intrinsics With FASTINTDIV Hardware + C28x	Improvement Factor
u32/u16	38	13	2.9
i64/i64 traditional (1)	78-2631	42	1.9-62.6
i64/i64 Euclidean (1)	82-2635	42	2.0-62.7
i64/i64 Modulo (1)	82-2635	42	2.0-62.7
i64/u64 traditional (1)	54-2605	42	1.3-62.0
i64/u64 Euclidean (1)	58-2609	42	1.4-62.1
i64/u6 Modulo(1)	58-2609	42	1.4-62.1
u64/u64/ (1)	53-2548	42	1.3-60.7

(1) The FASTINTDIV hardware implements 64-bit integer division with optimal fixed number of cycles for fast deterministic behavior. MCUs without such hardware acceleration, implement 64-bit integer division using generic CPU instructions that are not optimized for division or use algorithm techniques that optimize execution based on the value of the numerator and denominator. For instance, if the value of the numerator and denominator is less than 32-bits the software will execute a 32-bit division. Hence, the number of cycles can vary significantly and for large numerator and denominator values, overall cycles are much higher than achievable by the FASTINTDIV accelerator

3.2.3 Device List

- TMS320F2838xD/S
- TMS320F28003x
- TMS320F28002x

3.2.4 Hardware Platforms and Software Platforms

- F2838xD Fast Integer Division Example in C2000Ware
- F28002x Fast Integer Division Example in C2000Ware
- TMDSCNCD28388D Control CARD
- TMDSCNCD280039C Control CARD
- LAUNCHXL-F280025C LaunchPad

3.2.5 Documentation

- *Fast Integer Division - A Differentiated Offering From C2000 Product Family*
- *TMS320C28x Optimizing C/C++ Compiler User's Guide*
- *TMS320C28x Assembly Language Tools User's Guide*

3.3 Hardware Support for Double-Precision Floating-Point Operations

3.3.1 Value Proposition

The FPU64 module extends the computing capabilities of the C28x CPU by providing native hardware support for IEEE-754 single-precision and double-precision floating point operations. This enables customers who need double-precision floating point in their applications to implement them without seeing a significant increase in their CPU cycle load.

3.3.2 In Depth

The FPU64 module is extremely useful when 32-bit precision, such that, single-precision floating point is not sufficient for applications. In many real-time control applications, this is the case, so 64-bit precision (double-precision floating point) is needed. This normally comes at a price since native hardware on the device does not support double-precision floating point operations. Users see a significant increase in CPU cycles when running double-precision floating-point algorithms. By incorporating hardware support, this increase is avoided.

Table 3-3 compares the performance of double-precision operations on FPU64 vs FPU (single-precision floating point hardware).

Table 3-3. Cycle Comparison Between FPU64 and FPU32

Floating-Point Operations	FPU64 (cycles) -- fp_mode=relaxed	FPU (cycles) -- fp_mode=strict	FPU (cycles) -- fp_mode=relaxed FPU64 disabled	FPU (cycles) -- fp_mode=strict FPU64 disabled
32-bit Division	8	234	8	234
64-bit Division	27	27	2222	2222

The results in the above table (profiled with optimization off, code running from RAM) illustrate that the performance of double-precision floating-point division on FPU64 is slightly more expensive than single-precision floating-point division on FPU32. In the single-precision case, when the floating point mode (fp_mode) is set to relaxed, the compiler generates hardware instructions to perform single-precision division, at the slight expense of accuracy. When the floating point mode is set to strict, the compiler does not generate hardware instructions and calls into the RTS library to maintain accuracy, but at the cost of cycles.

In the double-precision case, the floating point mode does not matter, because the FPU64 implementation is accurate. As long as the FPU64 is enabled, the compiler generates hardware instructions supported by the FPU64 to perform double-precision division. If FPU64 is disabled, only then the compiler does not generate FPU64 hardware instructions and calls into the RTS library. 64-bit floating-point division cannot take advantage of 32-bit floating-point hardware, even in relaxed mode, which is why the cycles remain 2222.

Devices with the FPU64 utilize the same registers as the FPU except for the addition of eight floating-point results extension registers for double-precision floating-point operations. The FPU64 enhancements support all existing FPU single-precision floating-point instructions in addition to the 64-bit double-precision floating-point instructions. FPU64 64-bit instructions operate in one to three pipeline cycles, with some instructions also supporting a parallel move operation.

Using the FPU64 is straightforward. Users simply write C code with double-precision floating-point variables and operations, compile it with TI C28x C/C++ Compiler v18.9.0.STS (or later), and with compiler switches --float_support = fpu64. This will generate C28x native double-precision floating-point instructions. It may be helpful to refer to Table 3-4 for those unfamiliar with the size of the standard C data types on a C28x CPU. Users who want to write hand-optimized assembly for FPU64 may do so easily as well. There is roughly a one-to-one correspondence between single-precision and double-precision floating point assembly instructions. For the complete instruction set and details, see the document referenced in the documentation section. Software examples are listed in a section below that point to DSP and Math double-precision floating-point hand-optimized assembly routines that users can use for application development.

Table 3-4. Data Type Size C28x vs Arm

Data Type	C28x Bit Length (EABI)	Arm Bit Length
char	16	8
short	16	16
int	16	32
long	32	32
long long	64	64
float	32	32
double	64(1)	64
long double	64	64
pointer	32	32

(1) C28x COFF treats double as 32-bits

With the advent of auto code generation tools like MATLAB's Embedded Coder, many customers are migrating to them for auto code generation. Since MATLAB uses double-precision floating-point by default, it becomes easier to port code from a simulation environment to an embedded environment without having to revalidate operating performance of the system due to a reduction in precision. This is another key benefit of using the FPU64.

3.3.3 Device List

- TMS320F2838xD/S

3.3.4 Hardware Platforms and Software Examples

- TMDSCNCD28388D Control CARD Evaluation Module
- 64-bit floating point examples in the FPUFastRTS library in C2000Ware
- 64-bit floating point examples in the DSP FPU library in C2000Ware

3.3.5 Documentation

- *TMS320C28x Extended Instructions Sets Technical Reference Manual*

3.4 Increasing Control Loop Bandwidth With An Independent Processing Unit

3.4.1 Value Proposition

A primary concern in any control application is the time that elapses between sampling the system (sensing), applying the control function(processing), and applying the stimulus to the external system (actuation). The CLA was created specifically to address the need to minimize this time while increasing overall system throughput.

3.4.2 In Depth

The CLA found on the devices listed below is a fully parallel processor to the main C28x core. While the C28x core is a more traditional processor, executing instructions and servicing interrupts, the CLA is a task driven state machine. The CLA is a 32-bit floating point architecture.

Due to the nature of control systems, there are specific times when the sensing subsystem has new data to be processed. Advanced planning for these events with the other functions of an MCU can be difficult to time slice without introducing delay into both the system under control, but also any other functions the CPU has to perform.

As a task driven state machine, the CLA is constantly waiting in an idle state for an event, such as an ADC conversion, to process that data and actuate the system. Additionally, the CLA has full access to key control peripherals so it can fully realize the control system independent of the C28x CPU (Figure 3-6).

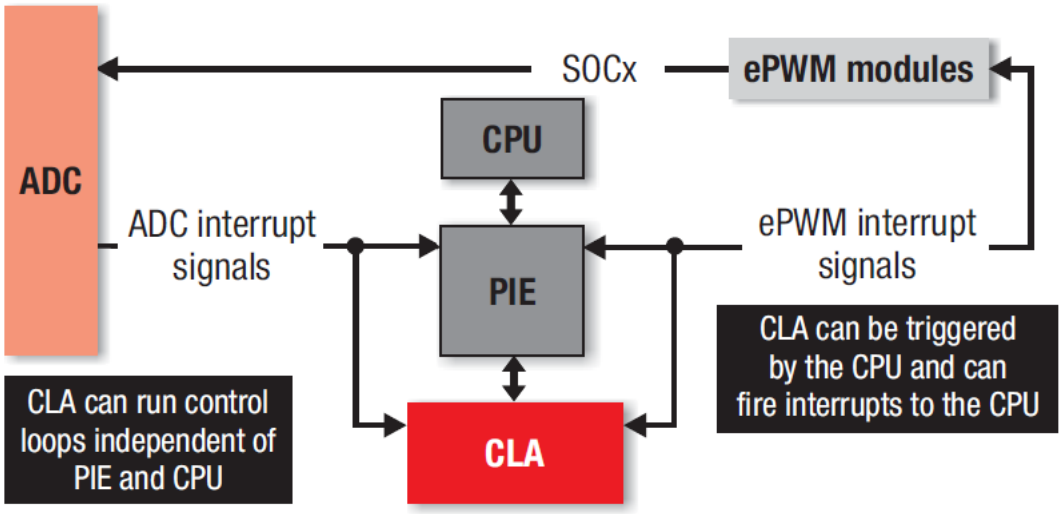


Figure 3-6. C28x and CLA Interfacing With the ADC and ePWM Modules

The above is beneficial for many reasons:

- There is little to no delay in processing the data, typically caused by the context switching of the main C28x core, and applying the new external stimulus to the system.
- There is no interruption or impact to the current C28x program execution.
- Potential to have parallel control systems running independently on the same MCU device.

The CLA has its own dedicated memory region for its code and shared memory for passing information between it and a C28x CPU in the system. The CLA is supported in Code Composer Studio™ by its own C compiler.

3.4.3 Device List

- TMS320F2838xD/S
- TMS320F2837xD/S
- TMS320F2807x
- TMS320F28004x
- TMS320F28003x

3.4.4 Hardware Platforms and Software Examples

- Valley switching boost power factor correction (PFC) reference design
- C2000 DesignDRIVE Development Kit for Industrial Motor Control

3.4.5 Documentation

- *Software Examples to Showcase Unique Capabilities of TI's C2000™ CLA*
- CLA Hands On Workshop
- CLA Usage in Valley Switching Boost Power Factor Correction (PFC) Reference Design
- CLA FAQ on E2E
- *Enhancing the Performance Capabilities of the C2000™ MCU Family*
- *Software Examples to Showcase Unique Capabilities of TI's C2000™ CLA*

3.5 Flexible System Interconnect: C2000 X-Bar

3.5.1 Value Proposition

Three on-chip signal crossbars (X-bars): Input, Output, and ePWM provide the necessary mechanism in hardware to efficiently connect multiple subsystems across different control system implementations. Lower system latency, simpler PCB routing, and consistent timing are all key benefits.

3.5.2 In Depth

The inter-dependence of the Sensing and Actuation subsystems in a real-time control MCU is an obvious one. Whether it is incoming signals into the device, signals generated by on-chip logic (comparator, SDFM, ADC, and so forth), or outgoing signals; routing these signals in the system can be challenging at best. The on-chip X-bars provide a flexible mechanism to do that in hardware inside the MCU. There are three key benefits that these modules provide to the system:

- Simple routing of external signals into the chip: Any GPIO can go to multiple modules on-chip (Figure 3-7). For example, this allows the eCAP module to choose from any input pin as its source, or any input pin to go to the CPU/CLA as an external interrupt. This also gives flexibility in signal routing and layout of the Printed Circuit Board (PCB) since pins are not hard configured for a set group of functions.

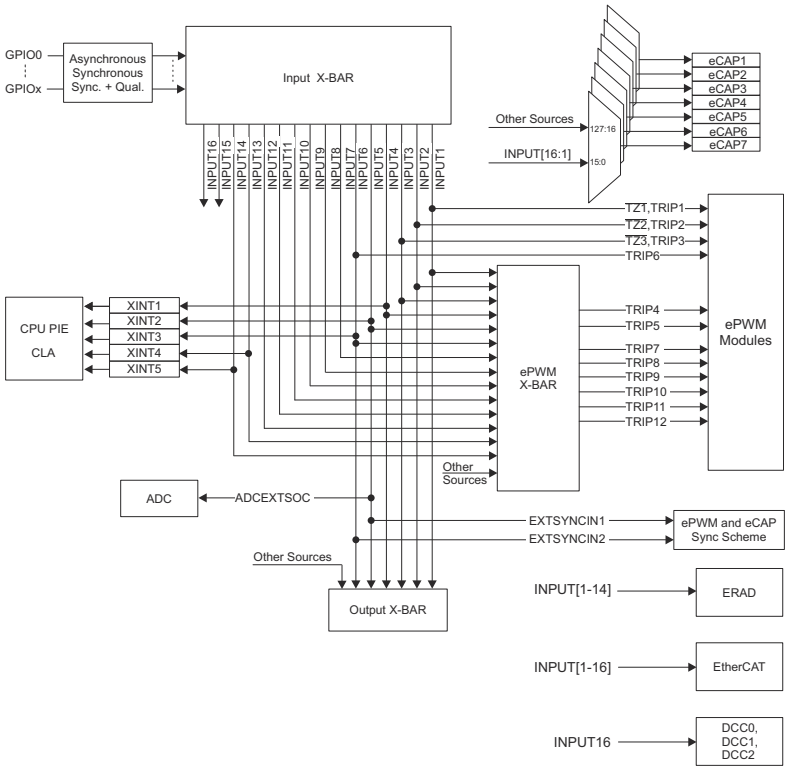


Figure 3-7. Input X-Bar on the TMS320F2837xD MCU

- Lower system latency: For simple point-to-point transactions, there is no need for the main C28x CPU or the CLA to spend cycles routing signals from one domain to another when using the X-bars. Inside of each X-bar, there is also a simple logical OR (Figure 3-8) that allows combination of any of the inputs to the X-bar in hardware. This not only saves cycles for the processors, but allows flexibility of the system since the mux selections are all controlled in software and can be changed dynamically as needed.

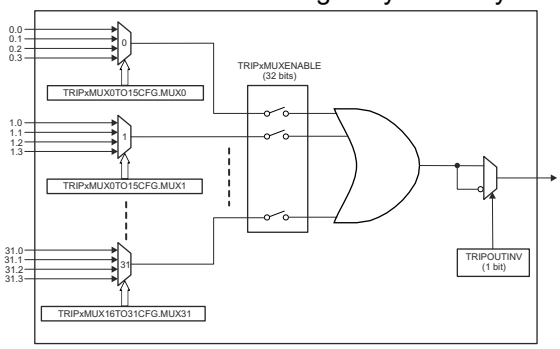


Figure 3-8. Local Mux and Logical OR on the TMS320F2837xD MCU

- Consistent timing: Similar to the above, since there is no CPU involvement, the signal propagates at the system clock in real time as it happens and is not contingent on another block to allow the signal to pass (Figure 3-9). This results in more predictable and repeatable system behavior. Given that the nature of many of the signals is to pass into the actuation sub-system timing is especially critical.

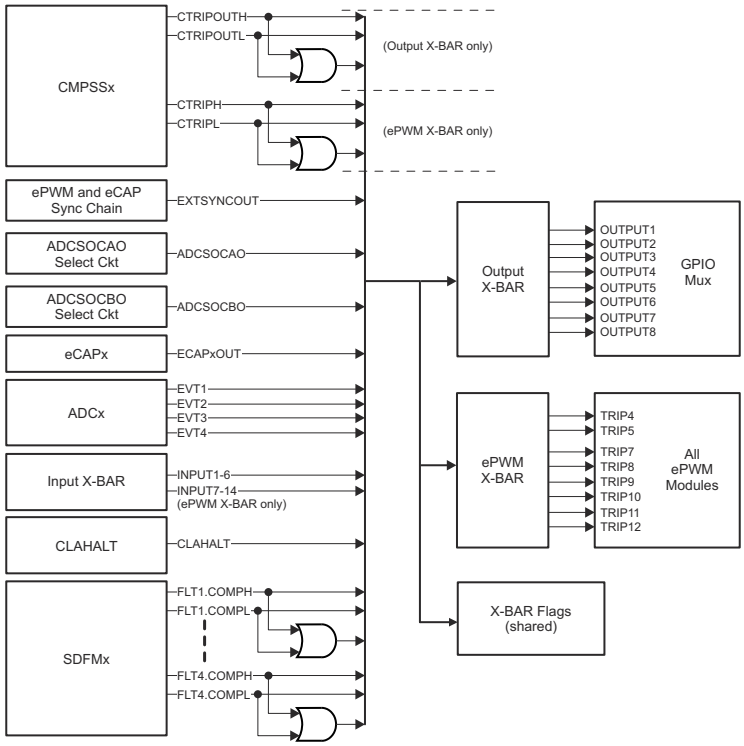


Figure 3-9. X-Bar Sources and Destinations on the TMS320F2837xD MCU

3.5.3 Device List

- TMS320F2838xD/S
- TMS320F2837xD/S
- TMS320F2807x
- TMS320F28004x
- TMS320F28003x
- TMS320F28002x
- TMS320F280013x
- TMS320F280015x

3.5.4 Hardware Platforms and Software Examples

- C2000 DesignDRIVE Development Kit for Industrial Motor Control
- Use TI SysConfig Tool to Simplify X-bar Setup

3.5.5 Documentation

- *TMS320F2838xD Real-Time Microcontrollers Technical Reference Manual*
- *TMS320F2837xD Real-Time Microcontrollers Technical Reference Manual*
- *TMS320F28004x Real-Time Microcontrollers Technical Reference Manual*
- *TMS320F28003x Real-Time Microcontrollers Technical Reference Manual*
- *TMS320F28002x Real-Time Microcontrollers Technical Reference Manual*
- *TMS320F280013x Real-Time Microcontrollers Technical Reference Manual*
- *TMS320F280015x Real-Time Microcontrollers Technical Reference Manual*

3.6 Improving Control Performance With Nonlinear PID Control

3.6.1 Value Proposition

The nonlinear PID (NLPID) provides the ability to improve control loop performance beyond that normally available with linear controllers. The controller is available in the Digital Control Library (DCL), making it easy to integrate with existing C2000 code.

3.6.2 In Depth

Conventional linear controllers such as the Proportional, Integral, Derivative (PID) are widely used with digital power applications, including motor control and motion control. The nonlinear PID (DCL Training Video - Non-linear Control) provided with the DCL extends the performance of its linear counterpart by shaping the loop error using a nonlinear law. A nonlinear shaping block is introduced in series with each of the three controller paths as shown below.

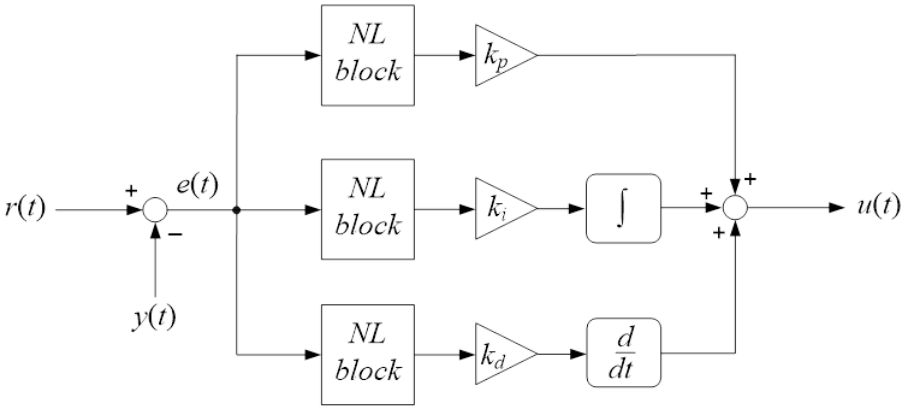


Figure 3-10. Non-Linear PID Block Diagram

The shape and aggressiveness of nonlinear action are configurable via six additional controller parameters (two in each nonlinear block), which are typically tuned in an iterative fashion to optimize a transient response. Like other controllers in the DCL, the NLPID parameters can be updated safely using a shadow parameter set and an update function.

Figure 3-11 shows an example of the potential improvement in step response available from the use of nonlinear control action.

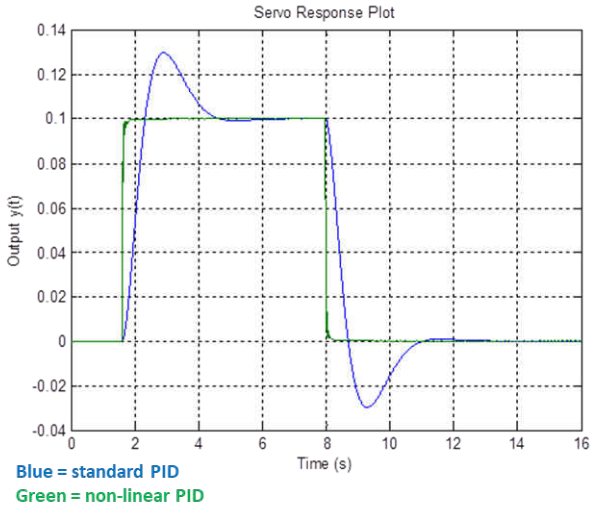


Figure 3-11. Comparison of Response Time Between Linear and Non-linear PID

The NLPID executes with highest efficiency on devices equipped with the type 1 TMU (see Section 3.1), such as the F280025 device. These devices have CPU instructions that allow the nonlinear controller to be executed in 117 cycles compared with around 3,300 cycles without such instructions. This cycle efficiency allows the nonlinear PID to be used in high frequency applications such as switching power supplies and current control loops.

The DCL is packaged in C2000Ware, which is available for free download by C2000 users. The library includes a PID controller tuning guide to help users get the most from the NLPID controller.

3.6.3 Device List

- TMS320F28003x
- TMS320F28002x

3.6.4 Hardware Platforms and Software Examples

- Digital Control Library in C2000Ware
- F280039C controlCARD
- F280025 controlCARD

3.6.5 Documentation

- DCL Training Video - Non-linear Control

3.7 Understanding Flash Memory Performance In Real-Time Control Applications

3.7.1 Value Proposition

In spite of the wait state penalty for flash memory accesses, the 128-bit-wide prefetch logic in association with the pipeline buffer makes sequential code execution performance equal to that of 0 wait state (WS) RAM. Allowing for common code discontinuities, most applications will run with an efficiency of approximately 80% relative to the code executing from RAM. In addition, there is a 128-bit cache on the data bus as well to improve data read performance. All of this performance enhancement comes with zero cycle-cost Error Code Correction (ECC) evaluation.

3.7.2 In Depth

Flash memory is a non-volatile memory that provides the advantage of retaining its content even after a power cycle. However, due to its physical construction Flash memory is typically not as fast as volatile memories (SRAM, DRAM, and so forth). As a result, wait states are used in order to scale the MCU clock rate while accessing Flash memory, which can impact CPU performance . In order to greatly reduce this impact on performance, C2000 FMC's (Flash Module Controller) read interface provides a prefetch mode (Figure 3-12). This mode significantly improves the performance of linear code, which typically makes up the majority of application code in real-time control systems.

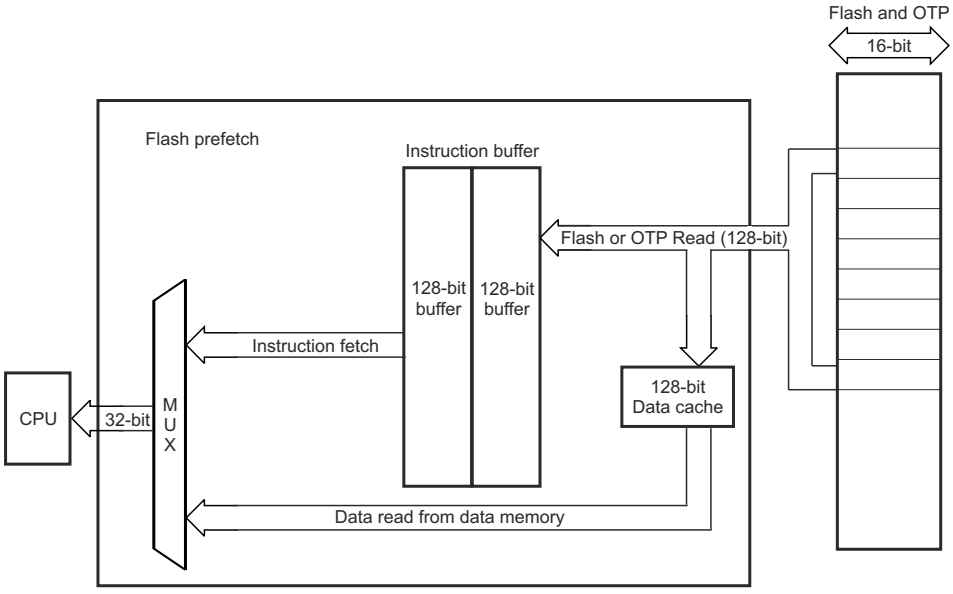


Figure 3-12. C2000 Flash Prefetch Module

When enabled on C2000 MCUs, the module does a look-ahead prefetch (128-bit aligned) on linear address increments starting from the last instruction fetch address and stores it in a 128-bit wide by 2-level deep instruction prefetch buffer. This buffer can hold up to sixteen 16-bit instructions and will be continuously filled in the background by the prefetch mechanism as the CPU continues to use the already fetched instructions in the buffer. As a result, no wait states are incurred for each opcode fetch, which is a significant performance boost when compared to a one time wait-stated fetch. Wait states are incurred only when there is a program counter (PC) discontinuity such as a branch, function call, and so forth.

Table 3-5 provides some real-world examples of the code efficiency that can be expected on two different classes of C2000 flash devices.

Table 3-5. Effective Flash Access Times With Prefetch Enable

Part Number	Device Properties	32-Bit Float Math	16-Bit If-Then-Else	ACI Motor Signal Chain
TMS320F2838xD/S TMS320F2837xD/S	• 200 MHz CPU Clock • 50 MHz Flash Speed • 3 Wait States	• **Flash Access Efficiency:** 93% • **Effective Performance:** 188 MHz	• **Flash Access Efficiency:**87% • **Effective Performance:**174 MHz	• **Flash Access Efficieny:** 92% • **Effecitve Perormance:** 184 MHz
TMS320F28004x TMS320F28002x	• 100 MHz CPU Clock • 20 MHz Flash Speed • 4 Wait States	• **Flash Access Efficiency:**84% • **Effective Performance:**84 MHz	• **Flash Access Efficiency:**84% • **Effective Performance:**84 MHz	• **Flash Access Efficiency:** 82% • **Effective Performance:**82 MHz
TMS320F28003x	• 120 MHz CPU Clock • 20 MHz Flash Speed • 5 Wait States	• **Flash Access Efficiency:**70% • **Effective Performance:**84 MHz	• **Flash Access Efficiency:**78% • **Effective Performance:**93 MHz	• **Flash Access Efficiency:** 73% • **Effective Performance:**87 MHz
TMS320F280013x TMS320F280015x	• 120 MHz CPU Clock • 40 MHz Flash Speed • 2 Wait States	• **Flash Access Efficiency:**96% • **Effective Performance:**115 MHz	• **Flash Access Efficiency:**91% • **Effective Performance:**109 MHz	• **Flash Access Efficiency:** 95% • **Effective Performance:**114 MHz

Note

The number of flash wait states is always 1 less than the access time.

- FLASH_wait_states = FLASH_access_cycles - 1
- FLASH_access_cycles = round_up(CPU MHz/FLASH MHz) For Example: = round_up(165/50) = round_up(3.3) = 4

While the efficiency of program code executing from flash has been considered thus far, there also exists a 128-bit data cache. Users can enable this to increase the flash data read performance. When the CPU requests data from a flash address, the flash wrapper will store entire 128-bits of Flash data (aligned) in this cache instead of simply providing the requested address's data to CPU. The CPU can access the remaining data in this cache without incurring any wait states. This data cache gets flushed and refilled when there is a cache miss.

Finally, there is also an Error Correction Code (ECC) value for each 64-bits of flash memory. The ECC is such that it provides for single bit error correction and dual bit error detection per 64-bits. The ECC is evaluated for correctness before the data is placed into the prefetch buffer with no impact to the access times/latencies mentioned previously. If an uncorrectable error is detected, a non-maskable interrupt is generated to halt normal code execution in parallel to the normal code execution of the CPU. You have the ability to set a threshold for correctable errors to trigger an interrupt to the C28x core as well.

3.7.3 Device List

- TMS320F2838xD/S
- TMS320F2837xD/S
- TMS320F2807x
- TMS320F28004x
- TMS320F28003x
- TMS320F28002x
- TMS320F280013x
- TMS320F280015x

3.7.4 Hardware Platforms and Software Examples

All code examples for the supported devices enable the prefetch buffer and ECC logic as part of their initialization routines inside the SysCtrl.c file.

3.7.5 Documentation

- *Signal Chain Benchmarking - A Demonstration of Optimized Real-time Performance of C2000™ MCU*
- *TMS320F2838x Real-Time Microcontrollers With Connectivity Manager Data Sheet* (for more information, see the *Flash Parameters* section)
- *TMS320F2837xD Dual-Core Microcontrollers Data Sheet* (for more information, see the *Flash Parameters* section)
- *TMS320F28004x Microcontrollers Data Sheet* (for more information, see the *Flash Parameters* section)
- *TMS320F28003x Real-Time Microcontrollers Data Sheet* (for more information, see the *Flash Parameters* section)
- *TMS320F28002x Real-Time Microcontrollers Data Sheet* (for more information, see the *Flash Parameters* section)
- *TMS320F280013x Real-Time Microcontrollers Data Sheet* (for more information, see the *Flash Parameters* section)
- *TMS320F280015x Real-Time Microcontrollers Data Sheet* (for more information, see the *Flash Parameters* section)

3.8 Deterministic Program Execution With the C28x DSP Core

3.8.1 Value Proposition

One important aspect of any real-time control system is the consistency of program execution over time. Whether it is performance across the system update period or from multiple power ups over time, optimization of real-time systems relies on critical system events taking place at known points in time. The well defined 8 stage CPU pipeline of the C28x MCU, as well as the complementary behavior of its interrupt handling logic, provides this level of determinism.

3.8.2 In Depth

A processor that is not consistent in its behavior over time can cause perturbations to a real-time control system, either by not actuating the system in a timely manner or by sampling the state of the system at the incorrect point in time. Furthermore, real-time control systems are largely interrupt driven in their program flow. For this reason, a cache memory is undesirable since it would need to be discarded often whenever a program discontinuity occurs (in this case quite often).

Instead, large amounts of fast memory are desirable for program execution as well as a processor with a instruction pipeline that is deep enough to parallelize instructions, but also shallow enough to not incur large time penalties when discontinuities occur. The C28x CPU employees an 8 stage pipeline as shown in Figure 3-13. Once an instruction has entered the D2 phase of the pipeline, it cannot be stopped from full execution by an interrupt. Conversely, any instruction that is in a pre-D2 phase of the pipeline will be flushed when the incoming interrupt is received by the C28x core. Upon returning from the interrupt program, execution begins again with the F1 fetch stage. It is beyond the scope of this article to go deeper into the nuances of the pipeline, but those details are covered in the *TMS320C28x DSP CPU and Instruction Set Reference Guide*.

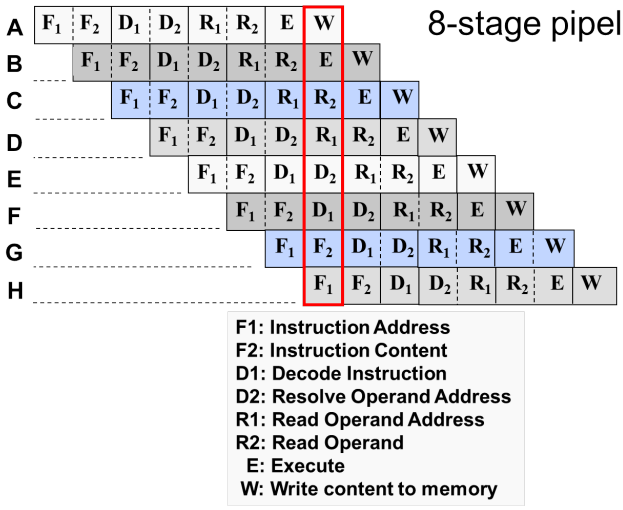


Figure 3-13. C28x Pipeline Visualization

While you can see that the behavior of the program execution is repeatable over time, what about the behavior of the incoming interrupt? Figure 3-14 shows that this, too, is deterministic. Note that unless manually altered all other interrupt requests will pend until the current interrupt has been fully serviced. This is important in order to keep servicing of any interrupt consistent in the time domain once it begins. Both of these components of the C28x core help maintain deterministic code execution for the system.

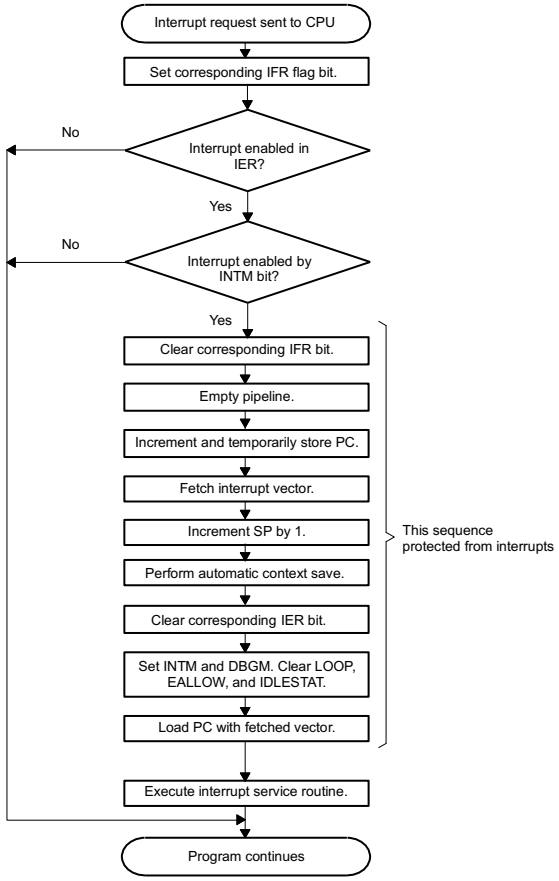


Figure 3-14. Standard Operation for a C28x CPU Maskable Interrupt

3.8.3 Device List

- TMS320F2838xD/S
- TMS320F2837xD/S
- TMS320F2807x
- TMS320F28004x
- TMS320F28003x
- TMS320F28002x
- TMS320F280013x
- TMS320F280015x

3.8.4 Hardware Platforms and Software Examples

- F28388D controlCARD evaluation module
- C2000 MCU F28379D LaunchPad™ development kit
- C2000 MCU F280049C LaunchPad™ development kit
- F280039C controlCARD evaluation module
- F280025 controlCARD evaluation module
- F2800137 controlCARD evaluation module
- F2800157 controlCARD evaluation module

3.8.5 Documentation

- *TMS320C28x CPU and Instruction Set Reference Guide*
- C2000™ Academy Overview

3.9 Efficient Live Firmware Updates (LFU) and Firmware Over-The-Air (FOTA) updates

3.9.1 Value Proposition

Industrial end equipment like Server Power Supply Units (PSU) require LFU support, so that a firmware update does not take the system offline. Automotive markets require FOTA support, where the firmware of any supported ECU (Electronic Control Unit) within the vehicle can be remotely updated. C2000 real-time MCUs have hardware features like multi flash bank support, interrupt vector table swapping, RAM block swapping and software examples, to efficiently perform both LFU and FOTA.

3.9.2 In Depth

Terminology:

It is important to understand the terminology since there is no single standard that defines this, at least for LFU. The following definitions are used:

- Installation – transfer of the firmware application or “image” from a host controller to the target (on which the firmware will execute).

  This also includes programming the new firmware on the Flash memory of the target device.
- Activation – as the name suggests, activating the new firmware application. In some of the LFU documentation, the term "Switchover" is also used
- LFU – installation and activation of new firmware occurs without a device reset
- FOTA – installation of new firmware occurs while the old firmware is running, but activation of new firmware requires a device reset
- Swapping – ability to map more than one physical hardware blocks to the same address space. Hardware could refer to flash banks, RAM blocks, interrupt vector tables, and so forth.

Building blocks:

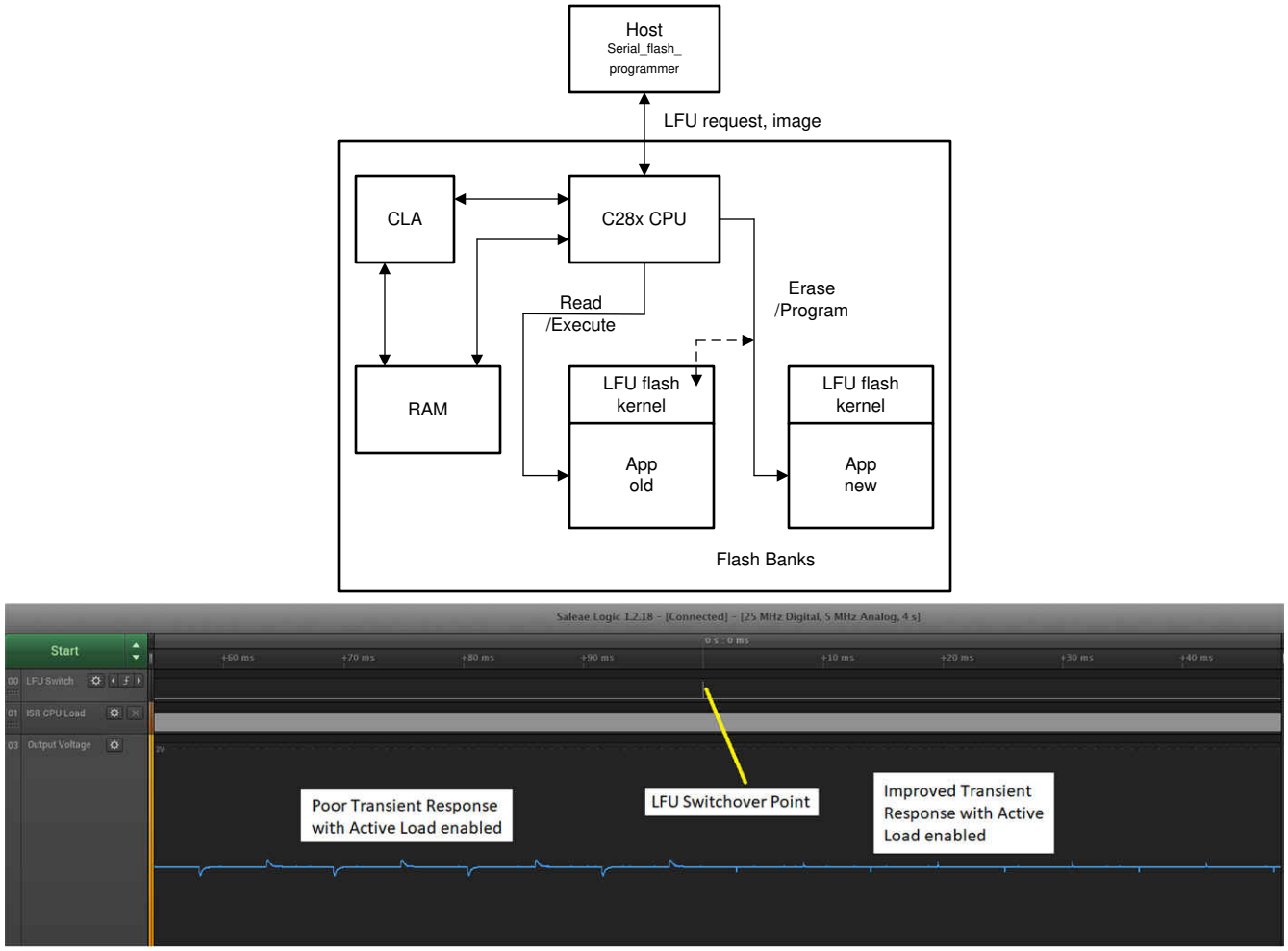
A number of Software and Hardware pieces are available to enable system level features like LFU and FOTA:

- Multiple flash banks – for example, one flash bank containing the active firmware, and the others containing inactive firmware.
- Hardware features to enable LFU – for example, interrupt vector table swapping and RAM block swapping, which enable fast activation for LFU.

- Compiler – an LFU aware compiler, which makes it easier for the user to integrate LFU support into their solution. For LFU, the compiler uses the existing firmware executable as a reference to generate the new firmware executable, which allows activation to be fast and efficient by optimizing the LFU initialization routine.
- Flash kernels – these refer to bootloaders that are provided as examples, reside in Flash, and support LFU and FOTA functionality. They can interact with a host to install firmware, and contain bank selection logic to determine, after a device reset, whether a flash bank contains valid firmware, and which firmware needs to be executed. Bank selection logic is also now built into the Boot ROM of a C2000 MCU and available as a boot mode.
- LFU software examples – simple examples as well as a reference design and a detailed user guide that walks through all the building blocks and LFU flow, enabling them to quickly integrate LFU into their system and achieve optimal performance. This is illustrated with examples of LFU on both the C28x CPU as well as the CLA. With a few changes, these examples can be repurposed for FOTA as well.

There are two flows defined for a LFU:

- At production:
  - Program Flash kernel, firmware in one or more flash banks
- In the field:
  - Option 1 - with this approach, you need to know which flash bank the firmware update is targeted for:
    - Developer creates new image (using previous image as reference image and Compiler with LFU support) knowing which flash bank it is targeted to
    - Host initiates LFU – downloads image corresponding to inactive bank
    - Flash kernel installs image to inactive bank
    - Old App (with LFU software/hardware support) activates new App without device reset (if previous step was successful)
    - Presence of old App allows fallback option in case of failure
  - Option 2 – with this approach, you **do not** need to know which Flash bank the firmware update is targeted for:
    - Developer creates two new images (using previous image as reference image and Compiler with LFU support) without knowing which flash bank it is targeted to. For example, if 2 Flash banks are used, and the new version is vN, the user would create vN built for Bank 1 using as reference vN-1 built for Bank 0. The user would also create vN built for Bank 0 using as reference vN-1 built for Bank 1.
    - Host initiates LFU – transfers **both images** to the target device
    - Flash kernel installs only the image targeted to the inactive bank, ignores the image targeted to the active bank
    - Old App (with LFU software/hardware support) activates new App without device reset (if previous step was successful)
    - Presence of old App allows fallback option in case of failure



There are two flows applicable to FOTA as well:

- At production:
  - Program Flash kernel, firmware in one or more flash banks
- In the field:
  - Without hardware support for Flash bank swapping, the factory would need to know which Flash Bank (Bank1 or Bank0) to build the firmware for. This approach is not acceptable because with automotive FOTA, you may freely update from any version to any other version, that is, skip versions, so it is impossible to pose a constraint like this
  - So a viable approach would be to always build the firmware to be Loaded to Bank1, and Run from Bank0. Since swapping is not available, the activation process involves copying the image from Bank1 to Bank0. The problem here is once the firmware is copied from Bank1 to Bank0, both Banks have the same new firmware now, without a valid backup present in case a rollback is needed. Since a rollback is an essential requirement with FOTA, a 3rd partition, or Flash bank, is needed
  - Developer creates new firmware to be loaded to Bank 1 and run from Bank 0
  - Host (FOTA Master ECU) initiates FOTA – transfers image to target device
  - Flash kernel installs image to Bank 1
  - Host initiates device reset. The old image in Bank 0 is copied by the Flash kernel to Bank 2 to be available in case a rollback is needed. Then the new image in Bank 1 is copied by the Flash kernel to Bank 0. Then the regular image activation steps occur

- Constraints:
  - FOTA without Flash bank swapping requires three Flash banks or partitions
  - In general, with LFU, firmware updates cannot be skipped since the reference image is needed for activation to be fast and efficient.
  - With FOTA, firmware updates can be skipped since there is no constraint on activation time, and activation occurs after a device reset, so a reference image is not needed

3.9.3 Device List

- TMS320F2837xS
- TMS320F28004x
- TMS320F28003x

3.9.4 Hardware Platforms and Software Examples

- Live firmware update reference design with C2000 real-time MCUs
- Flash kernel example illustrating LFU with device reset on F28002x (single flash bank)
- C2000Ware Flash kernel examples with LFU build configurations: F28003x, F28004x

3.9.5 Documentation

- Live Firmware Update With Device Reset on C2000™ MCUs
- Live Firmware Update Without Device Reset on C2000™ MCUs
- Serial Flash Programming of C2000 Microcontrollers
- TMS320F28003x Real-Time Microcontrollers

4 Control Key Technologies

4.1 Reducing Limit Cycling in Control Systems With C2000 HRPWMs

4.1.1 Value Proposition

Limit cycling in a PWM controlled system refers to the in-ability for the PWM output to physically converge on the mathematical solution to the control law. This causes the PWM output to cycle about the true solution, resulting in instability in the control system. The High Resolution PWM (HRPWM) module on the C2000 MCU has the ability to modulate the PWM edge in 150 ps increments. This represents a 60 fold improvement over traditional PWM creation techniques based off the system clock rate (Figure 4-1) and can be used to realize a higher order of accuracy in PWM edge placement. A waveform's period phase relationship to its complement, as well as deadband insertion time can all realize this high resolution benefit.

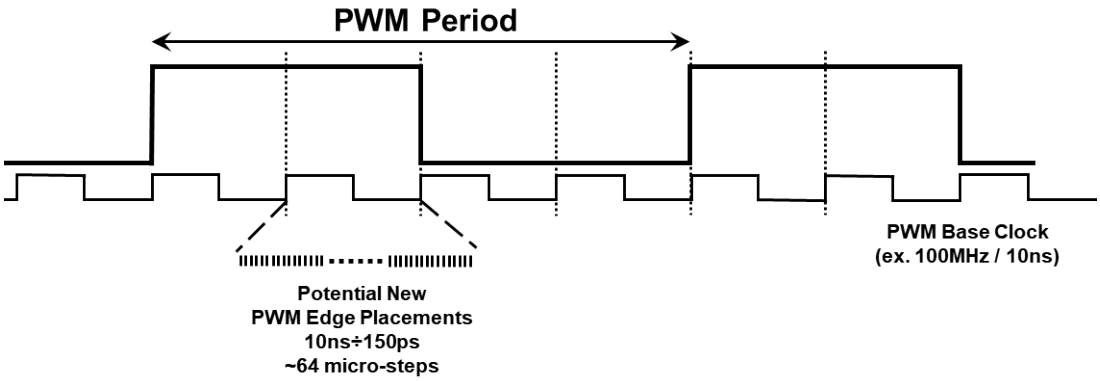


Figure 4-1. HRPWM Capability vs Traditional PWM Generation Methods

4.1.2 In Depth

All PWM-controlled power topologies are inherently bandwidth limited by the ability of the controller to place the PWM edge as close as possible to the mathematical solution of the control law. Whatever error exists from “rounding” the solution created in the form of an output PWM signal dictates the maximum efficiency that can be realized in the system.

In this sense, it may be helpful to think of the PWM as a type of DAC with a fixed resolution. Any error that results in the selection of the next available PWM edge placement would then be equivalent to quantization error term that is inherent to any DAC. Therefore, the minimum time step that can be achieved by a PWM module can be translated into “bits” of resolution of this equivalent DAC.

As shown in Table 4-1, the increase of resolution of the C2000 MCU HRPWM vs a traditional PWM is very apparent, increasing the effective resolution by approximately 6 bits.

Table 4-1. Resolution for PWM vs HRPWM

PWM Freq (kHz)	Regular Resolution (PWM) 100 MHz EPWMCLK		High Resolution (HRPWM)	
	Bits	% Error	Bits	% Error
20	12.3	0.02	18.1	0.000
50	11	0.05	16.8	0.001
100	10	0.1	15.8	0.002
150	9.4	0.15	15.2	0.003
200	9	0.2	14.8	0.004
250	8.6	0.25	14.4	0.005
500	7.6	0.5	13.4	0.009
1000	6.6	1	12.4	0.018
1500	6.1	1.5	11.9	0.027
2000	5.6	2	11.4	0.036

4.1.3 Device List

- TMS320F2838xD/S
- TMS320F2837xD/S
- TMS320F2807x
- TMS320F28004x
- TMS320F28003x
- TMS320F28002x
- TMS320F280013x
- https://www.ti.com/product/TMS320F2800157

4.1.4 Hardware Platforms and Software Examples

- TIDM-02002 Bidirectional CLLLC resonant dual active bridge for HEV/EV onboard charger
- TIDA-00961 Highly Efficient 1.6kW High Density GaN Based 1MHz CrM Totem-pole PFC Converter
- TIDA-010054 Bi-directional, dual active bridge reference design for level 3 electric vehicle charging stations
- C2000Ware HRPWM Example for TMS320F28388D
- C2000Ware HRPWM Example for TMS320F28379D
- C2000Ware HRPWM Example for TMS320F280049C
- C2000Ware HRPWM Example for TMS320F280039C
- C2000Ware HRPWM Example for TMS320F280025
- C2000Ware HRPWM Example for TMS320F2800137
- C2000Ware HRPWM Example for TMS320F2800157

4.1.5 Documentation

- *TMS320F2838xD Dual-Core Real-Time Microcontrollers Technical Reference Manual*
- *TMS320F2837xD Dual-Core Real-Time Microcontrollers Technical Reference Manual*
- *TMS320F28004x Real-Time Microcontrollers Technical Reference Manual*
- *TMS320F28003x Real-Time Microcontrollers Technical Reference Manual*
- *TMS320F28002x Real-Time Microcontrollers Technical Reference Manual*
- *TMS320F280013x Real-Time Microcontrollers Technical Reference Manual*
- *TMS320F280015x Real-Time Microcontrollers Technical Reference Manual*
- C2000 F2837xD Microcontroller 1-Day Workshop Section 1.6: Control Peripherals skip to 14:38 for HRPWM

4.2 Shoot Through Prevention for Current Control Topologies With Configurable Deadband

4.2.1 Value Proposition

C2000 MCUs have the ability to implement current control techniques, such as Peak Current Mode Control (PCMC), in hardware using the on-chip comparators to control the PWM duty cycle. Variable deadband insertion has been added to the Type 4 PWM, enabling the ability to tune out potential shoot through, without the need for any CPU overhead.

4.2.2 In Depth

For better power efficiency, many DC-DC systems implement a synchronous boost controller, where the secondary switch replaces the feed forward diode that exists in a regular boost controller (Figure 4-2). Peak Current Mode Control is one of the more common methods used to control this topology and the C2000 MCU has some unique features that allow it to implement this type of control very efficiently.

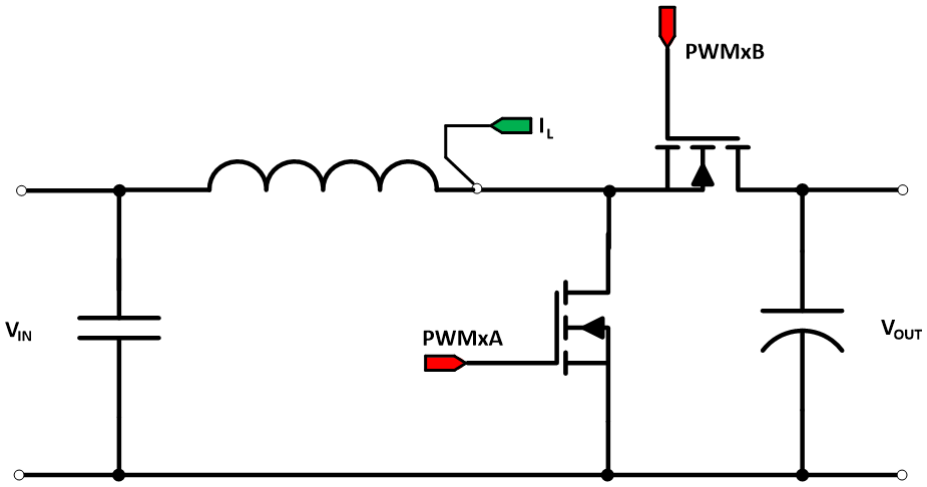


Figure 4-2. Synchronous Boost Controller

The addition of the second FET demands precise control of the ON/OFF time of the primary and secondary switch relative to one another. If the switches are in the “ON” state at the same time, there is a direct path to ground for the active current to flow, which is not only in-efficient but also potentially harmful to the lifetime of the FET switches.

An accurate way to implement this type of system is to have the comparator monitor the inductor current and actuate the FETs when the current exceeds a predefined threshold. Ideally, when one FET is switched ON, the other FET can be switched OFF at the same time. However, due to switch mismatch and board propagation delays, simultaneous switching from the controller likely will not result in simultaneous switching at the FETs, which creates the shoot through mentioned earlier. While techniques in software can help hold off the switching of the secondary switch to avoid this condition, these can be challenging to implement with the various hardware interdependencies coupled with the time constraints of the control loop.

The C2000 MCU has implemented programmable deadband control, derived from the comparator output itself, to prevent this condition while keeping the C28x CPU unloaded (Figure 4-3). This allows a complete PCMC solution to be realized outside of the CPU domain once initialized. This logic exists on all PWM modules on a given device, allowing multiple stages to have different deadbands, such as a Phase Shifted Full Bridge, where there are multiple switching pairs.

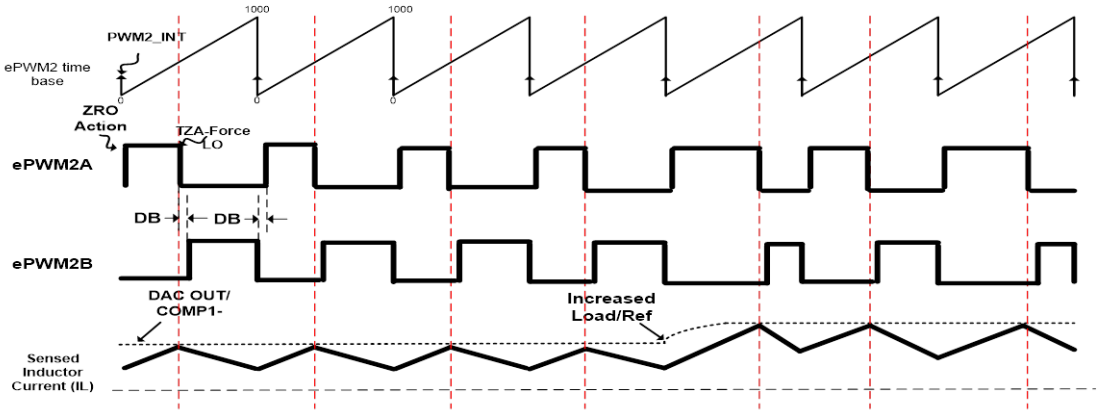


Figure 4-3. Cycle by Cycle Trip Action of the COMP Module With Configurable Deadband

4.2.3 Device List

- TMS320F2838xD/S
- TMS320F2837xD/S
- TMS320F2807x
- TMS320F28004x
- TMS320F28003x
- TMS320F28002x
- TMS320F280013x
- TMS320F280015x

4.2.4 Documentation

- *TMS320F2838xD Real-Time Microcontrollers Technical Reference Manual* (for more information, see the *Action-Qualifier (AQ) Submodule* and *Dead-Band Generator (DB) Submodule* sections of the *Enhanced Pulse Width Modulator (ePWM)* chapter)
- *TMS320F2837xD Real-Time Microcontrollers Technical Reference Manual* (for more information, see the *Action-Qualifier (AQ) Submodule* and *Dead-Band Generator (DB) Submodule* sections of the *Enhanced Pulse Width Modulator (ePWM)* chapter)
- *TMS320F28004x Real-Time Microcontrollers Technical Reference Manual* (for more information, see the *Action-Qualifier (AQ) Submodule* and *Dead-Band Generator (DB) Submodule* sections of the *Enhanced Pulse Width Modulator (ePWM)* chapter)
- *TMS320F28003x Real-Time Microcontrollers Technical Reference Manual* (for more information, see the *Action-Qualifier (AQ) Submodule* and *Dead-Band Generator (DB) Submodule* sections of the *Enhanced Pulse Width Modulator (ePWM)* chapter)
- *TMS320F28002x Real-Time Microcontrollers Technical Reference Manual* (for more information, see the *Action-Qualifier (AQ) Submodule* and *Dead-Band Generator (DB) Submodule* sections of the *Enhanced Pulse Width Modulator (ePWM)* chapter)
- *TMS320F280013x Real-Time Microcontrollers Technical Reference Manual* (for more information, see the *Action-Qualifier (AQ) Submodule* and *Dead-Band Generator (DB) Submodule* sections of the *Enhanced Pulse Width Modulator (ePWM)* chapter)
- *TMS320F280015x Real-Time Microcontrollers Technical Reference Manual*(for more information, see the *Action-Qualifier (AQ) Submodule* and *Dead-Band Generator (DB) Submodule* sections of the *Enhanced Pulse Width Modulator (ePWM)* chapter)
- C2000 Academy ePWM Module

4.3 On-Chip Hardware Customization Using the C2000 Configurable Logic Block

4.3.1 Value Proposition

The Configurable Logic Blocks (CLB) provide a mechanism for creating custom logic inside the chip. Enhancing existing subsystems, creating new subsystem and replacing off-chip logic circuits are all possible.

4.3.2 In Depth

The on-chip CLB provide a flexible mechanism to add personalized logic customization in hardware inside the MCU. Whether it is modifying an existing subsystem to satisfy an application specific need (Realizing Rotary Sensing Solutions Section 2.5), creating a completely new subsystem, or replacing external logic circuits, the CLB can accomplish all these tasks.

There are three key benefits that the CLB modules provide to the system:

1. Enhancing existing on-chip subsystems: The custom logic implemented inside the CLB modules can be inserted inside other on-chip subsystem, such as ePWM subsystem, to enhance the capability of the peripheral. Examples of this include creating a T-format absolute encoder (List item.referenceTitle) interface or Pulse Train Output (List item.referenceTitle).

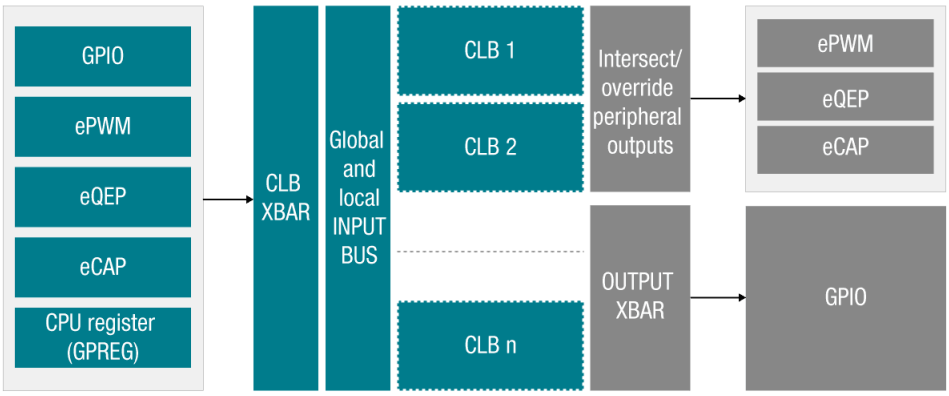


Figure 4-4. CLB Integration in C2000 MCU Architecture

2. Creating new subsystems: The CLB modules can be combined to form new subsystems, which could be a completely new peripheral that does not exist in C2000 MCUs, or replicate a C2000 MCU peripheral and create an extra subsystem, for example:
   a. Creating an auxilary PWM module SW example
   b. *Designing With the C2000 Configurable Logic Block* as shown in CLB State Machine SW example
3. Replacing external logic: In some cases the CLB modules can be used to absorb external devices such as FPGA or CPLD, which implements application specific customized logic. The CLB modules can also be used to replace external logic circuits. Examples of this are:
   a. Migrating from FPGA/CPLD to CLB
   b. Replacing external PWM protection circuit with custom logic in the CLB SW Example

While the physical implementation of the custom logic is controlled through uploaded memory values into the C2000 MCU, TI provides several GUI tools such as the CLB Configuration Tool (Figure 4-5) to both realize the logical implementation as well as verify the operation of the logic in simulation before it is used in the system.

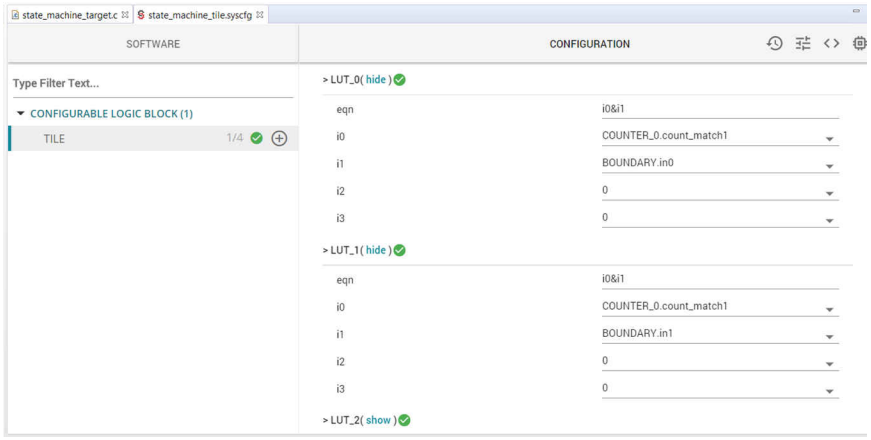


Figure 4-5. CLB Configuration Tool in SysConfig

4.3.3 Device List

- TMS320F2838xD/S
- TMS320F2837xD/S
- TMS320F2807x
- TMS320F28004x
- TMS320F28003x
- TMS320F28002x

4.3.4 Hardware Platforms and Software Examples

- F28388D controlCARD evaluation module
- LAUNCHXL-F28379D
- LAUNCHXL-F280049C
- F280039C controlCARD evaluation module
- LAUNCHXL-F280025C
- Creating an auxilary PWM module SW example
- CLB State Machine SW example
- Replacing external PWM protection circuit with custom logic in the CLB SW Example

4.3.5 Documentation

- CLB Programming Tool Training(Video)
- *Designing with the C2000™ Configurable Logic Block*
- *How to Migrate Custom Logic From an FPGA/CPLD to C2000™ Microcontrollers*
- *CLB Tool User's Guide*

4.4 Fast Detection of Over and Under Currents and Voltages

4.4.1 Value Proposition

Every control system can experience random events that can cause damage to the system. Fast detection and reaction to these events is critical to keeping the system safe and in good working condition. The on-chip comparators can detect and react to these events in a fraction of the time that it would take for an ADC and processor.

4.4.2 In Depth

Fault detection and reaction is important in most systems, not only for avoiding an undefined output, but for preventing damage to components both on and off the main Printed Circuit Board (PCB). The speed at which the fault detection takes place, as well as the final FET output state change, is critical to the system. A dedicated subsystem that ties together the analog and digital domains has been implemented on the C2000 MCU for handling this requirement: the Comparator Subsystem or CMPSS (Figure 4-6).

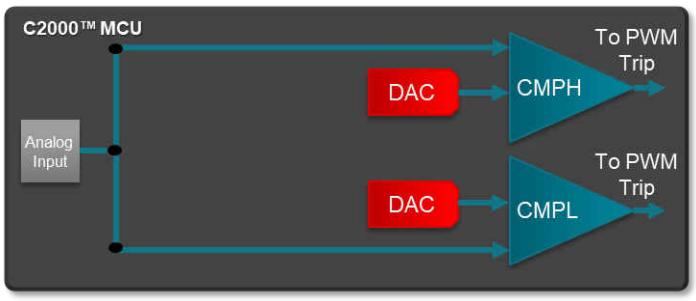


Figure 4-6. CMPSS Visualization

There are up to eight CMPSS modules on each C2000 MCU, with internal DACs that give the inverting/comparison detection level for the line that is being monitored. As shown in Figure 4-6, each CMPSS module has two comparators for simultaneous high and low detection. Using the CMPSS has several advantages over using the ADC for fault detection:

- System Overhead: Using the CMPSS to monitor a pin is an essentially zero overhead operation after the initial setup. The pin is always monitored against the comparison value until disabled. Other techniques would require periodic ADC conversions and threshold checking.
- Latency: While the ADC sampling rate can be simply factored into the period of the control loop, there is not a deterministic constant to a fault condition. As such there will be an inherent delay to detect the fail, both from a point of sample to the conversion time of the ADC itself. The comparator has no such trigger requirement or sample time, it is continuously monitoring the analog signal.
- Dedicated PWM Trip Zone input: The output of every CMPSS module can be tied directly into the Trip Zone of any PWM, and the action when the signal is received is configurable in software. This means there is no software overhead as there would be in processing an ADC ISR to then create the action to the PWM in software.
- No clock dependence: Since by definition a comparator is a purely analog domain circuit there are not clock dependencies to the changing state of its output based on the input. The C2000 MCU has carried this forward to give a asynchronous path from the comparator to the PWM. This allows for the fastest possible time from fault detection to pin state change, in addition to removing any clock dependence (Table 4-2).

Table 4-2. Comparison of Fault Detection and Trip Methods

Sampling Method	Sample Time (min)	Result Ready (min)	Latch and Change PWM Pin (@200 MHz SysClk)	Total Time From Fault To Trip
12-bit ADC	75 ns	260 ns	approximately 100 ns(inc ISR)	435 ns
12-bit ADC w/PPB	75 ns	260 ns	10 ns	355 ns
CMPSS	NA	NA	NA	60 ns

- Simultaneous high and low detection: Each input to a CMPSS module routes the signal to two physically independent comparators that give the ability to detect both overshoot and undershoot at the same time.

4.4.3 Device List

- TMS320F2838xD/S
- TMS320F2837xD/S
- TMS320F2807x
- TMS320F28004x
- TMS320F28003x
- TMS320F28002x
- TMS320F280013x
- TMS320F280015x

4.4.4 Hardware Platforms and Software Examples

The following kits implement the CMPSS to detect out of range current/voltage events

- TIDM-02002 Bi-Directional CLLLC Resonant Dual Active Bridge Reference Design for HEV/EV Onboard Charger
- TIDM-1022 Valley Switching Boost Power Factor Corrector
- TMDXIDDK379D C2000 Design DRIVE Development Kit for Industrial Motor Control

4.4.5 Documentation

- *TMS320F2838x Real-Time Microcontrollers With Connectivity Manager Data Sheet* (for more information, see the *Comparator Subsystem (CMPSS)* chapter)
- *TMS320F2837xD Dual-Core Microcontrollers Data Sheet* (for more information, see the *Comparator Subsystem (CMPSS)* section)
- *TMS320F28004x Microcontrollers Data Sheet* (for more information, see the *Comparator Subsystem (CMPSS)* section)
- *TMS320F28003x Microcontrollers Data Sheet* (for more information, see the *Comparator Subsystem (CMPSS)* section)
- *TMS320F28002x Real-Time Microcontrollers Data Sheet* (for more information, see the *Comparator Subsystem (CMPSS)* section)
- *TMS320F280013x Real-Time Microcontrollers Data Sheet* (for more information, see the *Comparator Subsystem (CMPSS)* section)
- *TMS320F280015x Real-Time Microcontrollers Data Sheet*(for more information, see the *Comparator Subsystem (CMPSS)* section)
- *TMS320F2838xD Real-Time Microcontrollers Technical Reference Manual* (for more information, see the *Comparator Subsystem (CMPSS)* chapter)
- *TMS320F2837xD Real-Time Microcontrollers Technical Reference Manual* (for more information, see the *Comparator Subsystem (CMPSS)* chapter)
- *TMS320F28004x Real-Time Microcontrollers Technical Reference Manual* (for more information, see the *Comparator Subsystem (CMPSS)* chapter)
- *TMS320F28003x Real-Time Microcontrollers Technical Reference Manual* (for more information, see the *Comparator Subsystem (CMPSS)* chapter)
- *TMS320F28002x Real-Time Microcontrollers Technical Reference Manual* (see the *Comparator Subsystem (CMPSS)* chapter)
- *TMS320F280013x Real-Time Microcontrollers Technical Reference Manual* (see the *Comparator Subsystem (CMPSS)* chapter)
- *TMS320F280015x Real-Time Microcontrollers Technical Reference Manual*(see the *Comparator Subsystem (CMPSS)* chapter)

4.5 Improving System Power Density With High Resolution Phase Control

4.5.1 Value Proposition

In typical charging systems used for EV such as On-Board Chargers or Off Board DC Chargers only 1% of ripple is allowed at the output voltage and current. Dual Active Bridge (Figure 4-7) is a popular topology to implement the DC-DC converter in these charging systems as it is able to achieve a wide operating range of voltages. The ability of the C2000 HRPWM to control phase shift within 150 ps allows for reducing component size while still meeting output ripple requirements.

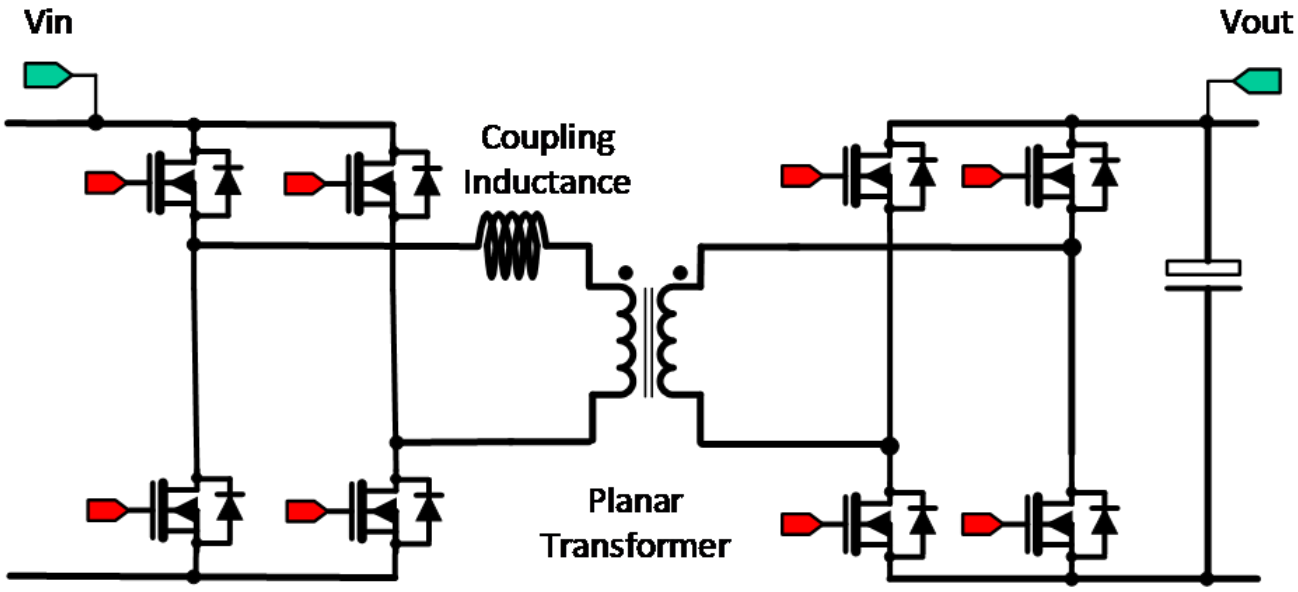


Figure 4-7. Dual Active Bridge Block Diagram

4.5.2 In Depth

Power transfer in single phase shift DAB is related to the coupling inductance (L) value and the switching frequency (Fs) and the phase (θ).

$$P_{output} = \frac{nV_{in}V_{out}\theta(\pi - \theta)}{2\pi^2 F_s L}$$

where

- n is number of turns of the transformer
- Vin is voltage input to the DAB
- Vout is voltage output from the DAB
- θ is the phase
- Fs is the switching frequency
- L is the inductance

For improved power density (smaller overall physical dimensions of the design), both lower inductance and higher switching frequency are necessary. A reference example is TIDA-010054, which is a 10kW DC-DC DAB converter: 100 kHz PWM switching frequency is used along with a 35 µH for the leakage inductor. A three phase PFC with 800 V bus such as Vienna Rectifier (TIDM-1000), or T-Type PFC (TIDA-010039) is used as the front end. Finally, for an output voltage of 400V at 10kW we can then determine the resultant output steps in the voltage the phase shift causes and check if it allows it to be within the ± 1% of our ripple requirement.

As shown in Table 4-3, the voltage cannot be regulated to 1% accuracy about 400 V by using typical PWM generation logic that is clocked at MCU clock rate. Assuming a 100 MHz CPU clock, would result in 10 ns step changes that is too large for our output ripple tolerance. Using the HRPWM module it is possible to modulate the phase shift to within 150 ps, regardless of the main MCU clock. More so, exceeding the phase step requirement represents the ability to modulate at a 13x step size within those bounds providing capability to find tune the output to more closely match the ideal.

Table 4-3. Phase Shift Requirements to Meet 1% Output Tolerance in a DAB Topology

Vin=800V, Pout=10kW, Fs=100kHz	Vout (V)	Required Phase Shift (ns)
θ = 19.032 deg	400	528.66
θ = 18.9249 deg	402	525.69
θ = 18.7662 deg	405	521.11
θ = 18.5077 deg	408	517.10

4.5.3 Device List

- TMS320F2838xD/S
- TMS320F2837xD/S
- TMS320F2807x
- TMS320F28004x
- TMS320F28003x
- TMS320F28002x
- TMS320F280013x
- TMS320F280015x

4.5.4 Hardware Platforms and Software Examples

- TIDA-010054 Bi-directional, dual active bridge reference design for level 3 electric vehicle charging stations
- TIDM-1000 Vienna Rectifier-Based Three Phase Power Factor Correction Reference Design
- TIDM-1000 Reference Software Project in Digital Power SDK
- TIDA-010039 Three-level, three-phase SiC AC-to-DC converter reference design

4.5.5 Documentation

- *TMS320F2838xD Real-Time Microcontrollers Technical Reference Manual* (for more information, see the *HRPWM* chapter)
- *TMS320F2837xD Real-Time Microcontrollers Technical Reference Manual* (for more information, see the *HRPWM* chapter)
- *TMS320F28004x Real-Time Microcontrollers Technical Reference Manual* (for more information, see the *HRPWM* chapter)
- *TMS320F28003x Real-Time Microcontrollers Technical Reference Manual* (for more information, see the *HRPWM* chapter)
- *TMS320F28002x Real-Time Microcontrollers Technical Reference Manual* (for more information, see the *HRPWM* chapter)
- *TMS320F280013x Real-Time Microcontrollers Technical Reference Manual* (for more information, see the *HRPWM* chapter)
- *TMS320F280015x Real-Time Microcontrollers Technical Reference Manual* (for more information, see the *HRPWM* chapter)

4.6 Safe and Optimized PWM Updates in High-Frequency, Multi-Phase and Variable Frequency Topologies

4.6.1 Value Proposition

The global load and global link features simplify applications that require active management of multiple PWM parameters, such as variable frequency, and multi-phase power topologies. More specifically, the global load feature helps eliminate PWM glitches by providing a single strobe signal to load PWM parameters from the shadow to active registers either within a single PWM module, or across multiple modules. The global link feature reduces CPU overhead by allowing simultaneous updates to linked registers across multiple PWM modules.

4.6.2 In Depth

Shadow registers are used as a temporary location for active registers until a load event occurs that enables the transfer of its content into the active registers at a strategic point within the PWM cycle. This shadow to active load feature prevents corruption and spurious operation due to registers being asynchronously modified by software. Traditionally, the load events that cause a transfer of content from shadow to active registers are configured for each register individually. However, when global load mode is enabled (GLDCTL[GLD] =1), the transfer of contents from shadow register to active register occurs at the same event (GLDCTL[GLDMODE]) for all registers with this mode enabled. Critical PWM registers for submodules such as the Time-Base, Action Qualifier, and Deadband can be enabled to use the global load feature through the GLDCFG register. Additionally, the number of load strobe events that need to occur before the active registers are updated GLDCTL[GLDPRD] can also be configured to meet application needs.

If the control ISR is asynchronous to the PWM switching frequency, the one-shot load mode feature is required to ensure all the registers within a PWM module are updated only once all of the required PWM parameters have been updated. The one-shot load mode is enabled through the GLDCTL[OSHTMODE] register. When GLDCTL2[OSHTLD] is set to one, the transfer of contents from shadow to active registers, for registers that are configured to use the global load mechanism, will occur on the next event selected by GLDCTL[GLDMODE]. After the shadow to active load event happens anymore global load events will be blocked until GLDCTL2[OSHTLD] is set to one again. Figure 4-8 is an example of the one-shot load mode action.

For variable frequency applications, there is a need for simultaneous writes of period and compare registers between ePWM modules. The EPWMXLINK register provides the capability to update the period and compare registers across multiple PWM modules by creating links between them. To further expand upon this, the global load mode register GLDCTL2 which allows for a reload or force of the one-shot load mode can also be linked between modules. This creates a scheme in which the period and compare registers can all be updated at the same time with the same value across multiple PWM modules.

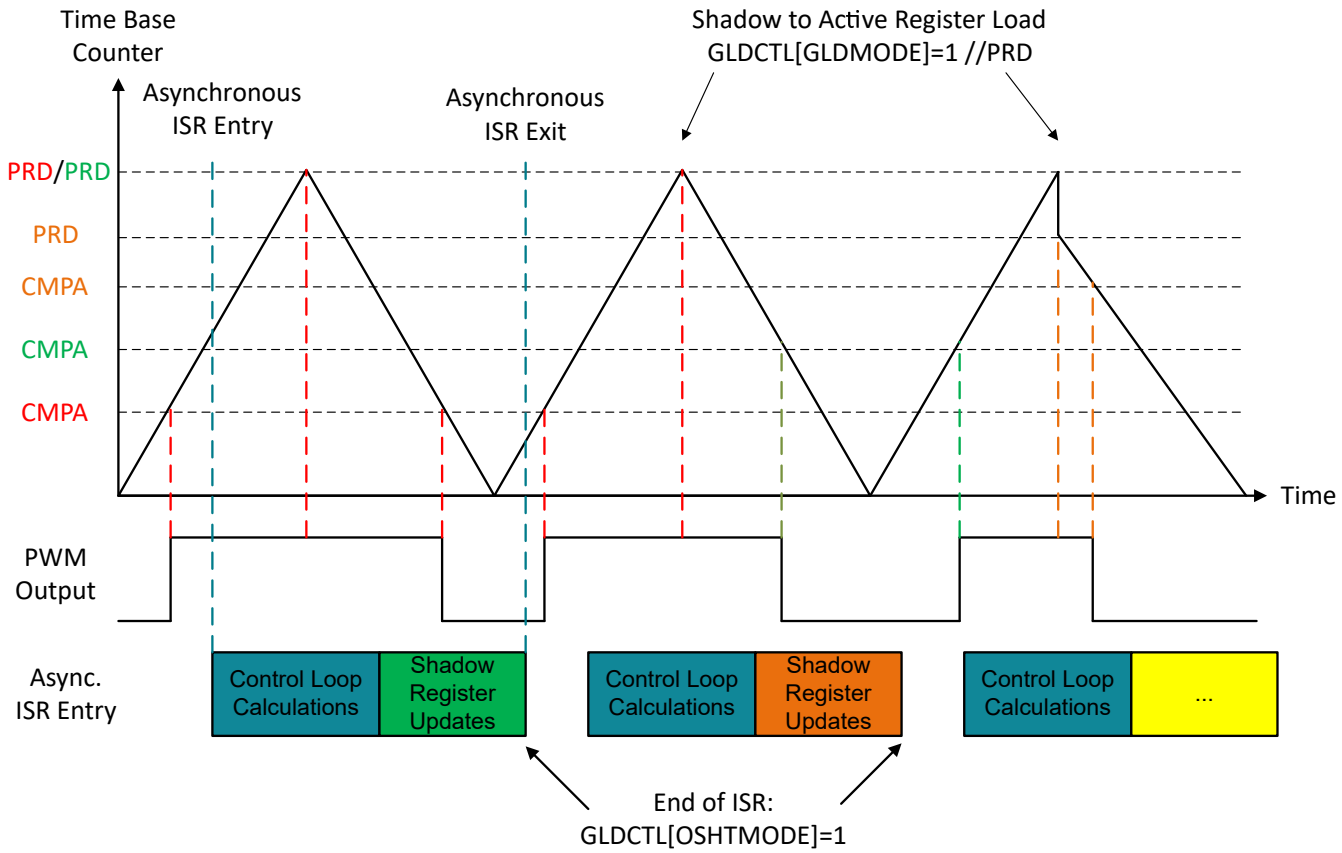


Figure 4-8. Shadow to Active Load Action

4.6.3 Device List

- TMS320F2838xD/S
- TMS320F2837xD/S
- TMS320F2807x
- TMS320F28004x
- TMS320F28003x
- TMS320F28002x
- TMS320F280013x
- TMS320F280015x

4.6.4 Hardware Platforms and Software Examples

- TIDM-1001 *Two Phase Interleaved LLC Resonant Converter Reference Design Using C2000 MCUs* (Global Load)
- TIDM-02002 *Bidirectional CLLLC Resonant Dual Active Bridge Ref Design HEV/EV Onboard* (Global Link)

4.6.5 Documentation

- *Leverage New Type ePWM Features for Multiple Phase Control*
- 'Global Load' section of ePWM 'Time-Base (TB) Submodule section of TRM

4.7 Solving Event Synchronization Across Multiple Controllers in Decentralized Control Systems

4.7.1 Value Proposition

Decentralized architectures used in industrial and automotive real-time control applications need a way to synchronize certain events with a minimal amount of trigger latency and jitter between devices. FSI is a high-throughput, low latency, wired communications peripheral that can achieve sub ~100ns time synchronized event triggering in multi-device network topologies. Using an already existing communication port, like FSI, for system event synchronization eliminates the need for additional Sync signals between devices saving device resources and additional design/BOM cost.

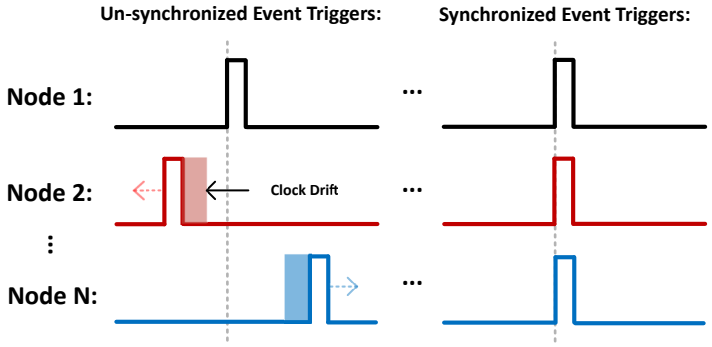


Figure 4-9. Un-synchronized vs Synchronized Event Triggers

4.7.2 In Depth

The communication interfaces between devices in decentralized architectures can vary in network topology, number of node devices, physical distance between the nodes, and more. Devices can also have slight deviations in their local clock during operation due to manufacturing uncertainties, thermal effects, aging, and so forth. Regardless of these variations it is often important for specific events to occur at the same time across all devices in a system, such as ADC start of conversions or rising/falling edges of PWM signals. Synchronous events between C2000 devices can be implemented using the FSI module and custom CLB logic.

With the goal of synchronizing equivalent PWM signals across devices as an example, the implementation can be explored further and is depicted in Figure 4-10. The lead device in a network will periodically send PWM sync requests, as FSI PING frames, to all node devices. In a daisy-chain topology each node device will forward the sync request frame to the next device in the chain. When a node device receives the sync request frame the CLB module will internally route the Ping Packet Received (PING_PKT_RCVD) signal from the FSI RX module through a configurable delay before connecting to the ePWM's EPWMSYNCIN signal. The configurable delay would be calibrated such that all node device's EPWMSYNCIN signals get triggered at the same time and match the lead device's TBCTR equals zero or PRD event. The implementation is completely hardware based and does not require software intervention after initialization. In a daisy-chain topology, nodes at the beginning of the chain will need a longer configurable delay compared to those at the end, since it takes time for the sync request frame to reach all devices. This will ensure the lead and all node devices PWM signals remain in-sync during operation.

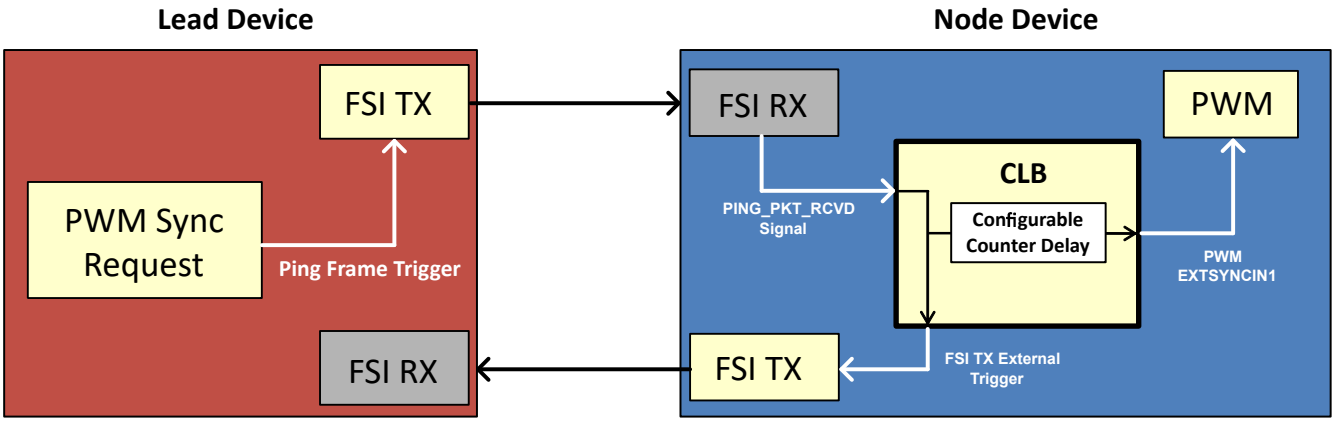


Figure 4-10. Lead and Node Device Implementation

Figure 4-11 demonstrates the PWM sync sequence and timing for an N node daisy-chain network. Some minimal jitter will be experienced on the PWM edge of each node due to device and physical layer uncertainties, but it will increase with the number of node devices in a daisy-chain network. Other network topologies, such as star or bus, that don't require frame forwarding would experience even less jitter.

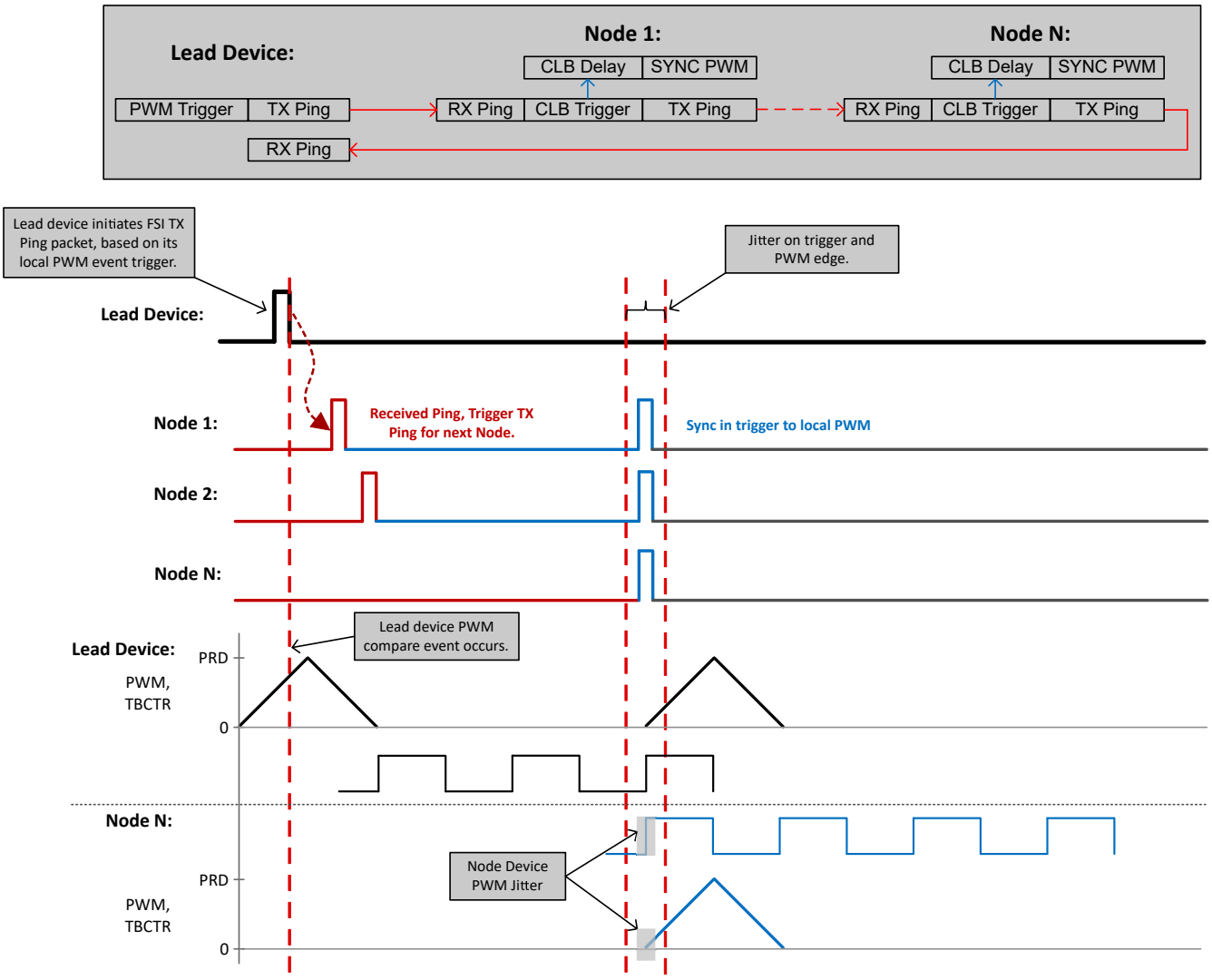


Figure 4-11. Daisy Chain Network Synchronization and Timing

For a more detailed explanation of the event synchronization over FSI along with test results, see the **Event Synchronization Over FSI** section of the *Using the Fast Serial Interface (FSI) With Multiple Devices in an Application* .

4.7.3 Device List

- TMS320F2838xD/S
- TMS320F28004x
- TMS320F28003x
- TMS320F28002x

4.7.4 Hardware Platforms and Software Examples

- TMDSFSIADAPEVM
- FSI Daisychain ePWM Sync Lead
- FSI Daisychain ePWM Sync Node

4.7.5 Documentation

- *Using the Fast Serial Interface (FSI) With Multiple Devices in an Application*

5 Interface Key Technologies

5.1 Direct Host Control of C2000 Peripherals

5.1.1 Value Proposition

Many industrial control systems implement a host controller that oversees multiple aspects of the system, including interfacing with the C2000 real-tim MCU as it implements the direct control loop. In other cases there may be a desire to add the differentiated IP of the C2000 to an existing system. The Host Interface Controller (HIC) logic inside the C2000 real-time MCU allows control of various peripherals using a common asynchronous interface.

Through the HIC, the host can use the C2000 device as a peripheral expander via the ASRAM port, by taking advantage of the C2000's differentiated features such as Fast Serial Interface (FSI), Configurable Logic Block (CLB), and others.

5.1.2 In Depth

The HIC supports a variety of configurations with which the host can be connected. Each configuration has certain advantages and implications with respect to latency, Input/Output (IO) pin overheads and so on.

The HIC allows access to many common C2000 MCU peripherals such as but not limited to: Analog to Digital Converters, Comparators, PWMs, CAN controller, Fast Serial Interface (FSI) communications, and the Configurable Logic Block (CLB).

For a full overview of the HIC features, see the *Design Guide for Enabling Peripheral Expansion Applications Using the HIC*.

The following section details few practical applications of HIC using the FSI for isolated communications and CLB to implement an absolute encoder.

5.1.2.1 HIC Bridge for FSI Applications

The Fast Serial Interface (FSI) is capable of supporting low-latency and robust high-speed communication across isolation in a system. Figure 5-1 shows how the host with an Asynchronous RAM interface can use the C2000 device as a peripheral bridge for high-speed communication across isolation using the FSI.

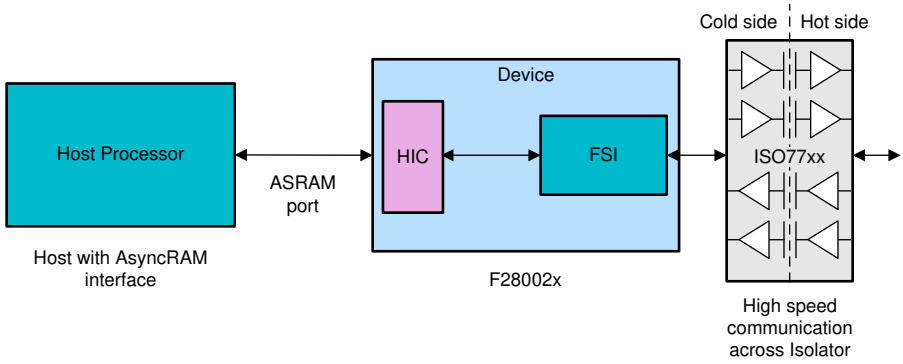


Figure 5-1. HIC Bridge for FSI Applications

5.1.2.2 HIC Bridge for Position Encoder Applications Using CLB

The CLB enables custom logic implementation and augments the existing C2000 peripheral set, thereby eliminating or reducing the need for FPGA, CPLD, or external logic components to achieve the same results. There are numerous solutions that can be implemented with the CLB, one of which is the absolute encoder protocol implementation to interface with the position sensors in an industrial drive control system (Figure 5-2). For the encoder protocols that are supported by the CLB, see the Position Manager Technology Section on the C2000 Website. The HIC can be used as a bridge for the host to interface with the position sensors as shown in Figure 5-2.

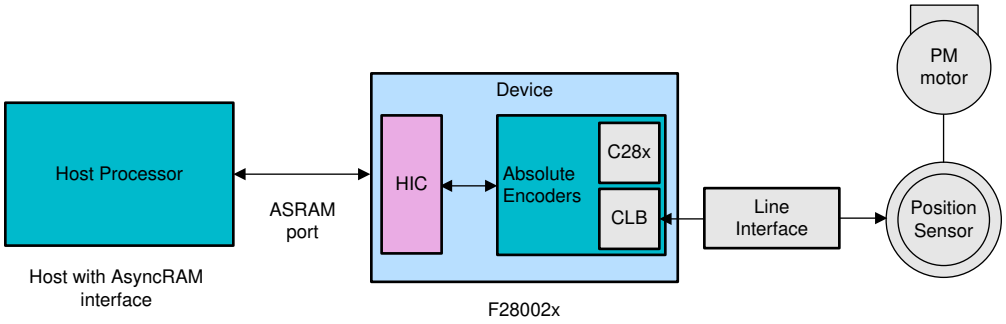


Figure 5-2. HIC Bridge for Position Encoder Applications

5.1.3 Device List

- TMS320F28003x
- TMS320F28002x

5.1.4 Hardware Platforms and Software Examples

- 16 Bit FSI Bridge Example in C2000Ware
- 16 Bit EMIF Example to access HIC
- 8 Bit Example in C2000Ware
- 8 Bit EMIF Example to access ADC using HIC

5.1.5 Documentation

- *Design Guide for Enabling Peripheral Expansion Applications Using the HIC*
- *TMS320F28003x Microcontrollers Technical Reference Manual*
- TMS320F28003x Microcontrollers Data Sheet
- *TMS320F28002x Microcontrollers Technical Reference Manual*
- TMS320F28002x Microcontrollers Data Sheet

5.2 Securing External Communications and Firmware Updates With an AES Engine

5.2.1 Value Proposition

Integrity and confidentiality of incoming commands, such as firmware updates, are essential to securing your system. Performing cryptographic algorithms in software is slow and requires a certain level of expertise. TI C2000 products have implemented a dedicated Advanced Encryption Standard (AES) engine in hardware to reduce the overhead associated with the encryption, decryption, tagging, and authentication of messages and data.

5.2.2 In Depth

The Advanced Encryption Standard (AES) hardware accelerator found on TI C2000 products is capable of operating in the most common modes for encryption/decryption, authentication, and authenticated encryption with associated data (AEAD). It supports the key sizes of 128 bits, 192 bits, and 256 bits, with a throughput of up to 4 bits/cycle, depending on the operation and key size. Utilizing a DMA, the engine requires very little CPU intervention, allowing you to manage the security of your communications while performing other vital tasks with the CPU.

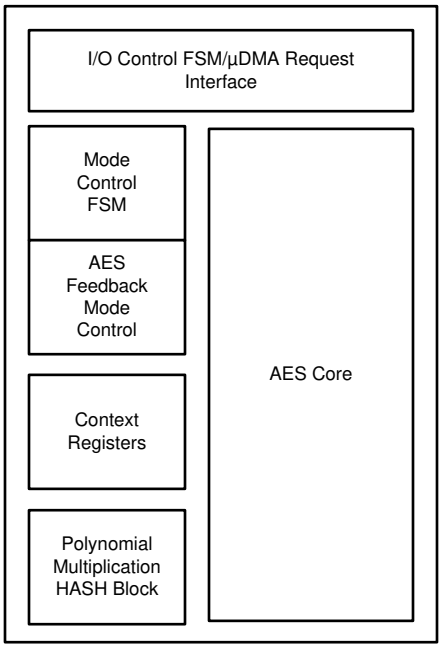


Figure 5-3. AES Block Diagram

For message confidentiality alone, the following encryption/decryption modes are supported:

- Electronic Code Book (ECB)
- Cipher Block Chaining (CBC)
- Counter (CTR) or Integer Counter (ICM)
- Cipher Feedback (CFB)
- Cipher Text Stealing (XTS)
- F8

For message integrity alone, the following authentication modes are supported:

- Cipher Block Chaining Message Authentication Code (CBC_MAC)
- F9

If both confidentiality and integrity of messages are required, the following authenticated encryption with associated data (AEAD) modes are supported:

- Galois Counter Mode (GCM)
- Counter Mode with CBC-MAC (CCM)

5.2.3 Device List

- TMS320F2838xD/S
- TMS320F28003x

5.2.4 Hardware Platforms and Software Examples

- TMDSCNCD28388D Control CARD
- AES ECB Encrypt
- AES ECB Decrypt
- AES GCM Encrypt
- AES GCM Decrypt

5.2.5 Documentation

- *TMS320F2838x Microcontrollers Technical Reference Manual* (for more information, see the *AES* chapter)
- *TMS320F28003x Microcontrollers Technical Reference Manual* (for more information, see the *AES* chapter)

5.3 Distributed Real-Time Control Across an Isolation Boundary

5.3.1 Value Proposition

Many real-time control systems rely on external communications for vital system information either as inputs into the control loop or as monitors to the system. Due to the time critical nature of any operation in these systems both the absolute speed as well as the integrity of the data is important. C2000 MCU's Fast Serial Interface (FSI) fulfills this need offering higher throughput compared to other serial communication peripherals, up to 200 Mbps, along with additional features in HW that add data integrity using only a few wires.

5.3.2 In Depth

There are a number of serial communication peripherals to choose from when designing a multi-device real-time control system. With processors needing to pass critical data between each other within very short periods of time, latency is a primary concern for the system designer.

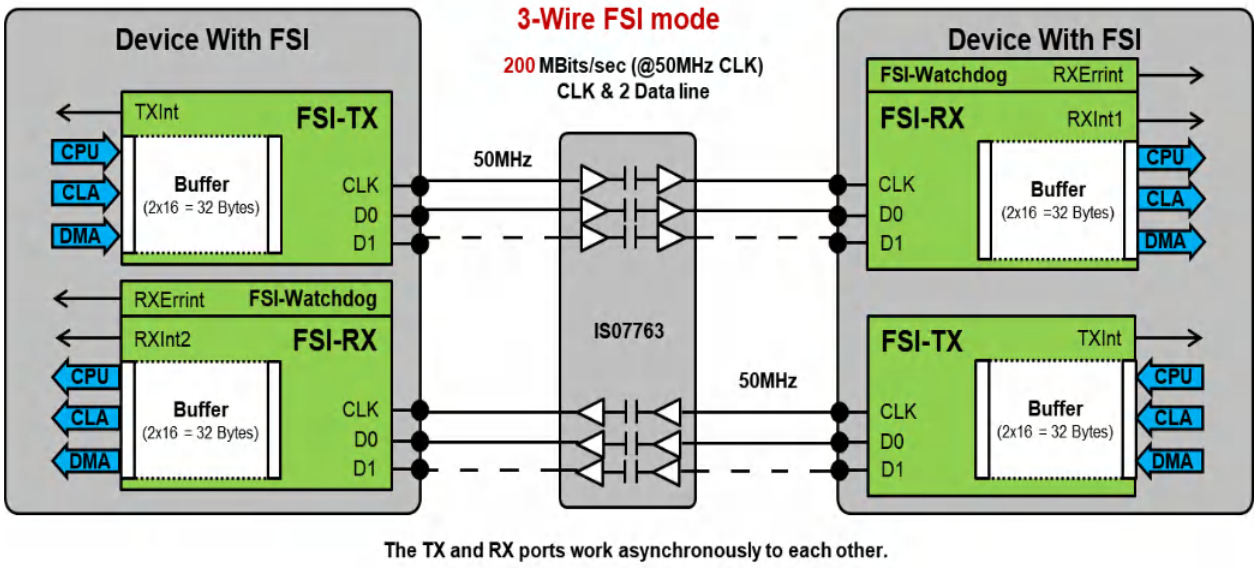


Figure 5-4. Full Duplex 3-Wire FSI Implementation

Note

Single data (D0) and clock (CLK) is minimum set of signals for FSI communications at 100Mbps with lowest signal count and isolation cost. Figure 5-4 shows that signal D1 is an optional data line used to achieve the full 200Mbps data rate of the FSI.

The FSI physical interface consists of three wires, a clock and two data signals, where one of the data signals is optional (see Figure 5-4). Data is transferred on both the rising and falling edge which permits a 50 MHz maximum FSI clock frequency to transfer data at 100 Mbps with two wires (CLK and D0) and 200 Mbps with three wires (CLK, D0, and D1). The high through-put along with defined data packets (frames) that contain limited header and footer allows data to be transferred between devices with very little latency. The FSI module consists of independent transmitter and receiver cores which allow for simultaneous full speed communications in both directions with no concept of a master or slave. A real-time system using FSI for distributed control is showcased in *Distributed Multi-axis Servo Drive Over Fast Serial Interface (FSI) Reference Design*.

Features that FSI offers over other commonly used communications peripherals include:

- Hardware implemented CRC at both the transmitter and receiver side eliminating the CPU overhead of a SW implementation
- Delay line control at the receiver module to compensate for channel-to-channel skew
- Line break detection using ping and data frame watchdogs
- FSI protocol has no concept of masters and slaves enabling devices to send feedback at any time without the master device having to make a request
- High bit rate with low signal count reduces the amount of isolators needed in the system

There are a number of system topologies which have components operating on both the “hot” (high voltage) and “cold” (low voltage) sides of the system that must communicate with each other. In this case digital isolators are used to bring data across an isolation barrier and the potential skew between signals that cross the isolation boundary can prove difficult to predict across many units of production. Even in systems without isolation, skew could be introduced by unequal signal trace lengths.

The delay line control feature at the receiver makes FSI well suited for this application as it can actively compensate for this skew (Figure 5-5) by adding delay to the individual FSI signals. See the *Fast Serial Interface (FSI) Skew Compensation* for more information on this differentiated feature. Also see the TMDSFSIADAPEVM for hardware evaluation of FSI with digital isolators.

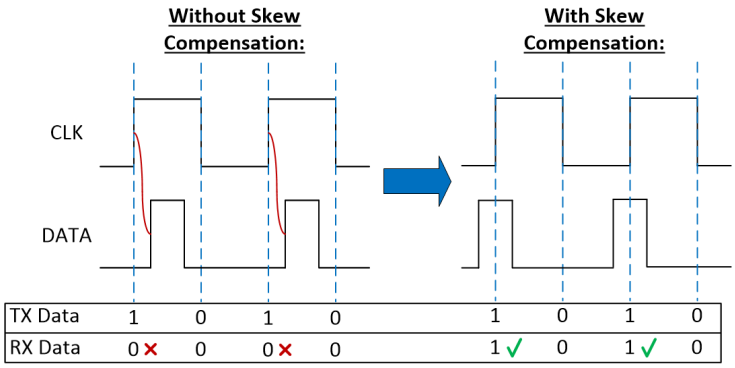


Figure 5-5. FSI Skew Compensation

Note

Data is always transmitted and received on both the rising and falling edges of the FSI clock.

While only one data line is show in Figure 5-5 there is option of second data line in all FSI implementations

5.3.3 Device List

- TMS320F2838xD/S
- TMS320F28004x
- TMS320F28003x
- TMS320F28002x

5.3.4 Hardware Platforms and Software Examples

- TMDSFSIADAPEVM
- LAUNCHXL-F280049C LaunchPad
- LAUNCHXL-F280039C LaunchPad
- LAUNCHXL-F280025C LaunchPad
- TIDM-02006

5.3.5 Documentation

- *Fast Serial Interface (FSI) Skew Compensation*
- *Using the Fast Serial Interface (FSI) With Multiple Devices in an Application*

5.4 Custom Tests and Data Pattern Generation Using the Embedded Pattern Generator (EPG)

5.4.1 Value Proposition

Unique pattern generation can be a useful addition to a real-time system, either for self testing purposes or as keep alive signals to other ICs in the system. This is typically accomplished with SW routines that toggle GPIOs manually, if possible. The EPG module removes the CPU overhead involved in these applications in addition to providing greater flexibility in patterns that can be generated.

5.4.2 In Depth

The Embedded Pattern Generator (EPG) module is a customizable pattern and clock generator that could servemany test and application scenarios that require a simple pattern generator or a periodic clock generator. TheEPG module can also be used to capture an incoming serial stream of data. The EPG module allows users todesign new clock generators,pulse width modulators (PWMs), serial communication modules, and so forth.

The EPG module can be used as an independent peripheral or it can be used along side other peripherals suchas CAN. When used alongside other peripherals, such as CAN, EPG can provide signal patterns for testing andrunning diagnostic inside the C2000™ device.

When used for clock generation, the EPG can generate:

- Independent clock generation and clock division
- Synchronous clock generation with programmable offsets(Figure 5-6)

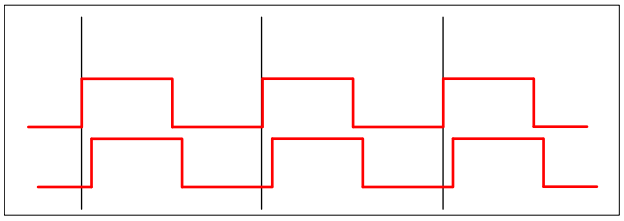


Figure 5-6. Clocks with Offset

When used for pattern generation, the EPG can generate:

- Independent serial data stream generation
- Serial data stream and the associated clock generation(Figure 5-7)
- Skew clock with respect to serial data
- Synchronous data stream with programmable offset with respect to one another

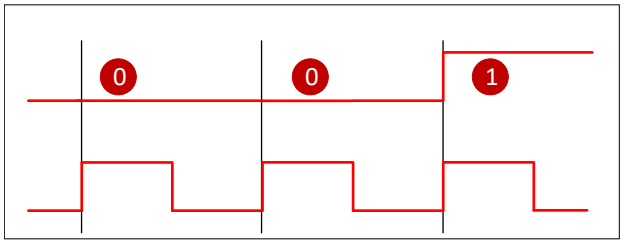


Figure 5-7. Serial Stream and Clock

5.4.3 Device List

- TMS320F28003x
- TMS320F280013x
- TMS320F280015x

5.4.4 Hardware Platforms and Software Examples

- C2000Ware EPG example for clock generation
- C2000Ware EPG example for serial data
- TMS320F280039C LaunchPad
- TMS320F2800137 LaunchPad
- TMDSCNCDF2800157

5.4.5 Documentation

- Designing With the C2000™ Embedded Pattern Generator(EPG)
- TMS320F28003x Real-Time Microcontrollers TRM
- TMS320F280013x Real-Time Microcontroller TRM
- TMS320F280015x Real-Time Microcontroller TRM

6 Safety Key Technologies

6.1 Non-Intrusive Run Time Monitoring and Diagnostics as Part of the Control Loop

6.1.1 Value Proposition

CPU resource management is often a key concern when evaluating the ability to control the system. Understanding the bandwidth required by the different functions is important not only during the development of the system but also while the system is in its final use. The ERAD (Embedded Real Time and Diagnostics) module provides this capability on the C2000 MCU, decreasing time spent during developement, but also ensuring stable and safe performance in the end application.

6.1.2 In Depth

ERAD is a hardware module with enhanced bus comparators and system event counters that sits within the MCU bus architecture as shown in Figure 6-1. ERAD on its own can generate system level interrupts and flags and can also feed into other peripherals such as the CLB for further enhancing the capabilities. For complete understanding of the relationship between these two modules, see the CLB documentation.

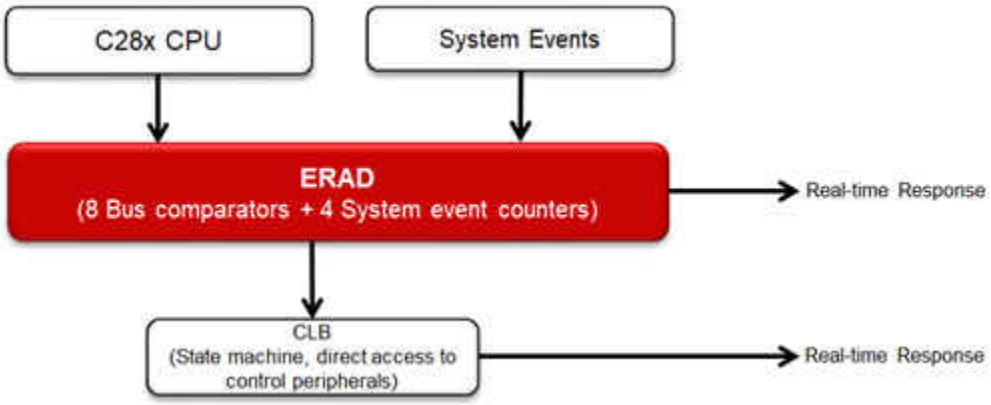


Figure 6-1. ERAD Block Diagram

Enhanced bus comparator monitors some of the critical CPU internal buses and signals that convey information about the CPU code execution and pipeline. Most of the critical CPU interfaces are monitored by EBC units, which can be configured to match any address/data/instruction patterns that appear on the CPU interface and generate events based on this. A simple example would be to generate an event when an address of interest is accessed for a read or write. Additionally these events can be exported to CLB to define state machines to keep track of sequences of these events.

On the other hand, system event counters monitor various events in the system like interrupts and peripheral activity which is another important requirement for any real-time system to monitor and profile. All the key system events like interrupts, DMA triggers and important peripheral events (via interrupts, CLA events, and so forth) are available for profiling inside of the ERAD. Hence, these can be time-stamped, counted and measured with relation to the CPU activity. Enhanced counter block allows the system events as well as CPU activity to be measured and analyzed with reference to other events too.

Example: Stack Overflow Detection

A frequently encountered issue with embedded code development is detecting an overflowing stack. ERAD can simply use a single enhanced bus-comparator unit and then map the stack end address with a small margin to generate an interrupt when there is a write that is attempted. With the below example (Figure 6-2), just use a comparator with address 0x99FF0 and mask the last 4-bits for comparison, so any write access between addresses 0x99FF0 to 0x99FFF would generate an event/interrupt, which can then be handled appropriately based on kind of the application.

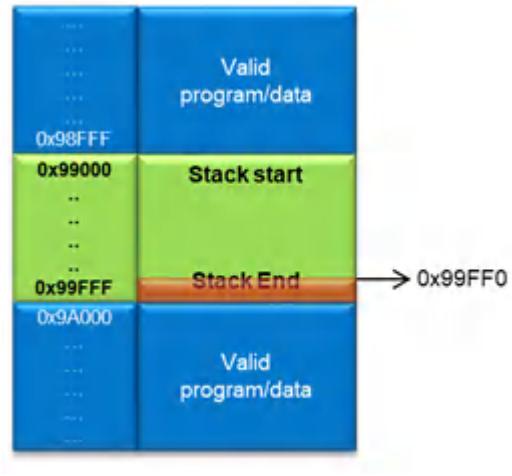


Figure 6-2. Stack Overflow Protection

6.1.3 Device List

- TMS320F2838xD/S
- TMS320F28004x
- TMS320F28003x
- TMS320F28002x

6.1.4 Hardware Platforms and Software Examples

- F28388D controlCARD evaluation module
- C2000 MCU F280049C LaunchPad™ development kit
- F280039C controlCARD evaluation module
- F280025 controlCARD evaluation module

6.1.5 Documentation

- ERAD Introduction video
- *Embedded Real-Time Analysis and Response for Control Applications*
- *TMS320F2838x Real-Time Microcontrollers TRM*
- *TMS320F28004x Real-Time Microcontrollers TRM*
- *TMS320F28003x Real-Time Microcontrollers TRM*
- *TMS320F28002x Real-Time Microcontrollers TRM*

6.2 Hardware Built-In Self-Test of the C28x CPU

6.2.1 Value Proposition

Safety applications require both start up and run time diagnostics of critical device components. The most critical is the main CPU; in the case of a C2000 device this is the C28x real-time core. Furthermore, without deep understanding of the CPU this is not feasible for a user to create such diagnostics in software. On supported C2000 MCUs, a block of logic is provided to automate and perform these tests to high level of diagnostic coverage to satisfy safety standards like ISO 26262 or IEC 61508.

6.2.2 In Depth

The Hardware Built-In Self-Test (HWBIST) provides a very high diagnostic coverage on the CPU, including the FPU, TMU, and VCU, at a transistor level during start-up and application time. This logic utilizes Design for Test (DfT) structures inserted into the device for rapid execution of high quality manufacturing tests, but with an internal test engine rather than external automated test equipment (ATE). This technique has proven to be effective in providing high coverage in less time.

HWBIST tests are triggered by the software. The user may select to run all tests or only a subset of the tests based on the execution time allocated for the diagnostic. This time sliced test feature enables the HWBIST to be used effectively as a runtime diagnostic with execution of test in parallel with the application. HWBIST execution failure such as detection of a logic failure or time out without the self-test completing triggers an NMI. The maximum level of diagnostic coverage provided by HWBIST depends on the device as shown in HWBIST Supported Diagnostic Coverage by Device.

Table 6-1. HWBIST Supported Diagnostic Coverage by Device

F2838xD/S	F2837xD/S	F2807x	F28002x
> 99% DC	> 99% DC	> 99% DC	> 99% DC

To configure and run HWBIST it is required to use the HWBIST APIs provided by the C2000 Software Diagnostic library. The functions provided enable running a single micro-run or running the full HWBIST test as well as configuring a self-test mode within HWBIST to allow a few different types of faults to be injected to check that it's functioning correctly.

6.2.3 Device List

- TMS320F2838xD/S
- TMS320F2837xD/S
- TMS320F2807x
- TMS320F28003x
- TMS320F28002x

6.2.4 Hardware Platforms and Software Examples

- C2000 Software Diagnostic Library in C2000Ware (F28002x, F2838x)
- C2000 SafeTI™ Diagnostic Software Library (F2837xD/S, F2807x)

6.2.5 Documentation

- Safety Manual for TMS320F28002x
- Safety Manual for TMS320F2837xD, TMS320F2837xS, and TMS320F2807x
- C2000™ Hardware Built-In Self-Test

6.3 Zero CPU Overhead Cyclic Redundancy Check for Embedded On-Chip Memories

6.3.1 Value Proposition

During development of any control system great care is taken to understand the interactions and timings of the code that will be executed by the device. However, no matter the previous effort, if that code is unintentionally modified or corrupted the behavior of the system is no longer guaranteed. Many systems, especially those concerned with safety, will periodically check the goodness of the memories, but this comes at the expense of CPU and system bandwidth. The Background CRC(BGCRC) module on C2000 MCUs solves this problem, implementing a CRC engine that can check memory integrity with no CPU overhead or system performance impact.

6.3.2 In Depth

The memory blocks present in C2000 MCUs are parity or ECC protected. This helps in detecting memory corruptions if any. But parity/ECC feature can detect errors only if application does a read access to the memory. Hence it is very important to do a periodic readback of the memories to detect such issues earlier. Adding a CRC check for the memories confirms that the memory contents are unaltered.

The BGCRC module in C2000 MCUs can be used to perform readback of the memories in the background, hence ensuring uninterrupted execution of application without consuming any CPU cycles. Once triggered, it reads the memory contents in the background, computes CRC and compare it with the programmed golden value and flag error if there is mismatch. It also supports Scrub mode, where it does not perform a CRC comparison, but check for Parity/ECC errors. The reads happen during the idle time (when none of the other memory owners such as CPU, CLA, DMA are accessing the memory block) and hence the functional accesses are not impacted.

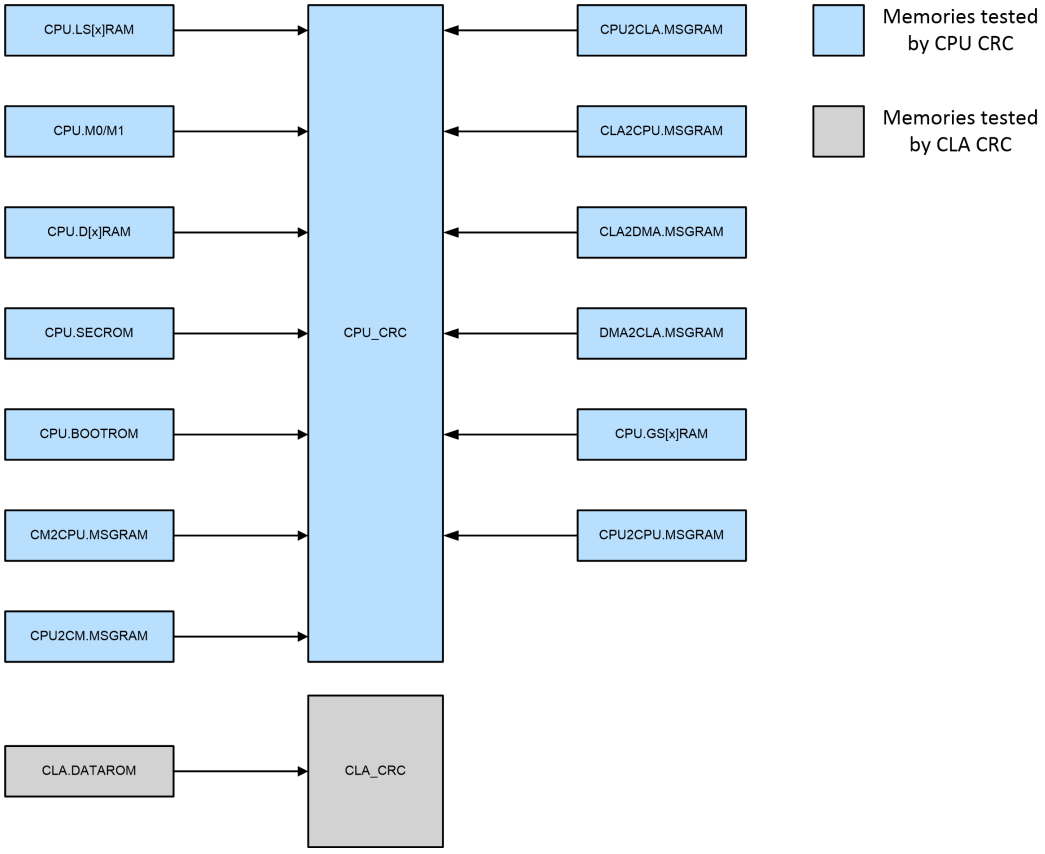


Figure 6-3. BGCRC Implementation on the TMS320F2838xD Device

6.3.3 Device List

- TMS320F2838xD/S

- TMS320F28003x
- TMS320F28002x

6.3.4 Hardware Platforms and Software Examples

- F2838x BGCRC Example
- F28003x BGCRC Example
- F28002x BGCRC Example

6.3.5 Documentation

- *CRC Engines in C2000™ Devices*

6.4 Boot Code Authentication Prior To Code Execution

6.4.1 Value Proposition

A real-time control system typically has multiple FirmWare(FW) updates over its lifetime. While these can be done in person, it is much more commonplace and convient to update the FW remotely using a bootloader on the device. With a remote update there is increased concern to ensure the contents of the new FW are not only correct but have not been manipulated by an outside source. The secure boot feature on C2000 MCUs provides a method to verify the contents of the new FW prior to code execution.

6.4.2 In Depth

One of the C2000 MCU features related to the application flash boot is the ability to authenticate the user application code in flash before execution. This ascertains the integrity of the application code by ensuring that it has not been tampered with, after getting programmed into the Flash memory.

The secure flash boot is realized using the 128-bit AES-CMAC Authentication algorithm that is run on the application code contents returning a pass/fail status and proceeds to execute the application code only if the authentication succeeds. Cipher-based Message Authentication Code (CMAC) is an AES-based authentication algorithm that constructs an authentication tag from a block of input data. The input data block is fed 128 bits at a time, into the cryptoengine/software (based on the CPU subsystem), along with a 128-bit CMAC key (Figure 6-4).

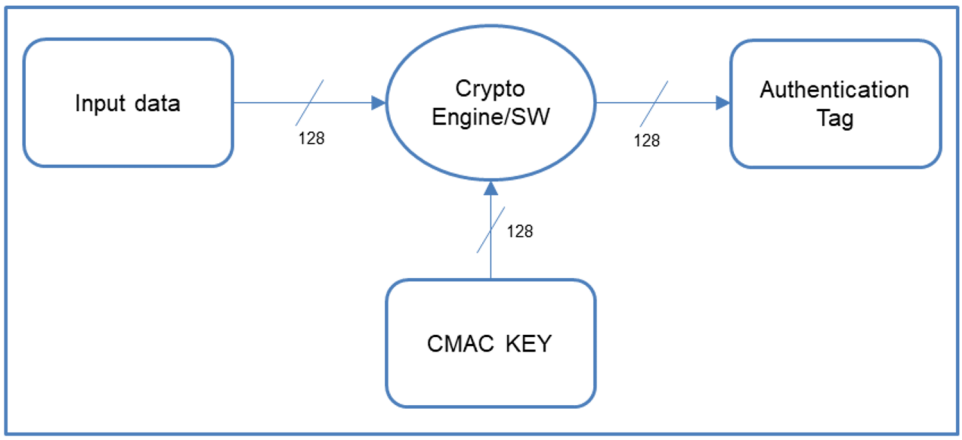


Figure 6-4. CMAC Operation

6.4.3 Device List

- TMS320F2838xD/S
- TMS320F28003x
- TMS320F280013x
- TMS320F280015x

6.4.4 Hardware Platforms and Software Examples

- C2000Ware Secure Boot Example for F2838xD
- F28388D controlCARD evaluation module
- TMS320F280039C LaunchPad
- TMS320F2800137 LaunchPad
- TMDSCNCD2800157 controlCARD

6.4.4.1 Documentation

- *Secure BOOT on C2000 Device*
- *TMS320F2838x Real-Time Microcontrollers With Connectivity Manager TRM*
- *TMS320F28003x Real-Time Microcontrollers TRM*
- *TMS320F280013x Real-Time Microcontrollers TRM*
- TMS320F280015x Real-Time Microcontrollers TRM

7 References

7.1 Device List

- TMS320F2838xD/S
- TMS320F2837xD/S
- TMS320F2807x
- TMS320F28004x
- TMS320F28003x
- TMS320F28002x
- TMS320F280013x
- TMS320F280015x

7.2 Hardware/Software Resources

- TIDM-1007 Interleaved CCM Totem Pole Bridgeless Power Factor Correction (PFC) Reference Design
- TIDM-HV-1PH-DCAC Single-Phase Inverter Reference Design With Voltage Source and Grid Connected Modes
- TMDXIDDK379D C2000 DesignDRIVE Development Kit for Industrial Motor Control
- TMDSHVMTRINSPIN High Voltage Motor Control Kit with InstaSPIN-FOC and InstaSPIN-MOTION enabled for F280049C device lab7 and lab8
- Vienna Rectifier-Based Three Phase Power Factor Correction Reference Design Using C2000 MCU
- TIDM-02002 Bidirectional CLLLC resonant dual active bridge for HEV/EV onboard charger
- TIDA-00961 Highly Efficient 1.6kW High Density GaN Based 1MHz CrM Totem-pole PFC Converter
- F28388D controlCARD evaluation module
- C2000 MCU F28379D LaunchPad™ development kit
- C2000 MCU F280049C LaunchPad™ development kit
- C2000 MCU F280039C LaunchPad™ development kit
- C2000 MCU F280025 LaunchPad™ development kit
- C2000 MCU F2800137 LaunchPad™ development kit
- C2000 DesignDRIVE Development Kit for Industrial Motor Control
- Valley switching boost power factor correction (PFC) reference design
- TMDSIDDK379D
- TMDSCNCD28379D
- TMDSCNCD280049C
- TMDSCNCD280039C
- TMDSCNCD2800137
- TMDSCNCD2800157
- TIDM-1022 Valley Switching Boost Power Factor Corrector

7.3 Documentation

- Texas Instruments: *C2000 Real-Time Control Peripherals Reference Guide*
- Texas Instruments: *TMS320F2837xD Dual-Core Real-Time Microcontrollers Technical Reference Manual*
- Texas Instruments: *TMS320F28004x Real-Time Microcontrollers Technical Reference Manual*

- Texas Instruments: *Enhancing the Performance Capabilities of the C2000™ MCU Family*
- Texas Instruments: *TMS320C28x Extended Instruction Sets Technical Reference Manual*
- Texas Instruments: *TMS320C28x CPU and Instruction Set Reference Guide*
- Texas Instruments: *TMS320F28002x Real-Time Microcontrollers Data Sheet*
- Texas Instruments: *TMS320F2838x Real-Time Microcontrollers Technical Reference Manual*
- Texas Instruments: *TMS320F28002x Real-Time Microcontrollers Technical Reference Manual*
- Texas Instruments: *Designing with the C2000™ Configurable Logic Block*
- Texas Instruments: *How to Migrate Custom Logic From an FPGA/CPLD to C2000™ Microcontrollers*
- Texas Instruments: *CLB Tool User's Guide*
- Texas Instruments: *Quick Response Control of PMSM Using Fast Current Loop*
- Texas Instruments: *Fast Current Loop Library*
- Texas Instruments: *C2000 Position Manager PTO API Reference Guide*
- Texas Instruments: *Distributed Multi-axis Servo Drive Over Fast Serial Interface (FSI) Reference Design*
- Texas Instruments: *Fast Serial Interface (FSI) Skew Compensation*
- Texas Instruments: *Using the Fast Serial Interface (FSI) With Multiple Devices in an Application*
- Training: How the C2000 Configurable Logic Block (CLB) tool integrates custom logic in my design
- ADC Specifications for TMS320F2838xD/S
- ADC Specifications for TMS320F2837xD/S
- ADC Specifcations for TMS320F2807x
- ADC Specifications for TMS320F28004x
- C2000™ F2837xD Microcontroller Workshop
- CLA Hands On Workshop
- CLA Usage in Valley Switching Boost Power Factor Correction (PFC) Reference Design
- CLA FAQ on E2E
- C2000 F2837xD Microcontroller 1-Day Workshop Section 1.6: Control Peripherals

8 Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from Revision E (June 2022) to Revision F (March 2023)	Page
• Updated the numbering format for tables, figures and cross-references throughout the document	4
• Added F280013x and F280015x device to applicable devices as well as any support hardware tools	4
• Updated Section 3.7.2	36
• Updated Section 3.9	40
• Updated Section 3.9.2	40

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