C2000™ MCU DesignDRIVE Solutions for Industrial Motor Drives
# Table of contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>3</td>
</tr>
<tr>
<td>DesignDRIVE Platform and Architecture</td>
<td>4</td>
</tr>
<tr>
<td>Easily Increase Functionality in Motor Drive Applications</td>
<td>4</td>
</tr>
<tr>
<td>Industrial Drive Control Architectures</td>
<td>5</td>
</tr>
<tr>
<td>Designing the Next Generation of Industrial Drive and Control Systems</td>
<td>9</td>
</tr>
<tr>
<td>Additional Resources</td>
<td>20</td>
</tr>
<tr>
<td>Fast Current Loop Control</td>
<td>21</td>
</tr>
<tr>
<td>Achieve Unprecedented Current-Loop Performance from an Off-The-Shelf MCU</td>
<td>21</td>
</tr>
<tr>
<td>A Faster Current Loops Pays Off in Servo Motor Control</td>
<td>22</td>
</tr>
<tr>
<td>Fast Current Loop Performance: Better Than We Thought, and Measurable in Your Own Lab</td>
<td>27</td>
</tr>
<tr>
<td>Performance Analysis of Last Current Loop (FCL) in Servo Drives Using SFRA on C2000™ Platform</td>
<td>29</td>
</tr>
<tr>
<td>Additional Resources</td>
<td>38</td>
</tr>
<tr>
<td>Position Manager</td>
<td>39</td>
</tr>
<tr>
<td>Simple Interfacing to Analog and Digital Position Sensors for Industrial Drive Control System</td>
<td>39</td>
</tr>
<tr>
<td>Additional Resources</td>
<td>43</td>
</tr>
<tr>
<td>Communications</td>
<td>44</td>
</tr>
<tr>
<td>EtherCAT and C2000 MCUs – Real-Time Communications Meets Real-Time Control</td>
<td>44</td>
</tr>
<tr>
<td>Additional Resources</td>
<td>48</td>
</tr>
<tr>
<td>Product Overviews</td>
<td>49</td>
</tr>
<tr>
<td>The TMS320F2837xD Architecture: Achieving a New Level of High Performance</td>
<td>49</td>
</tr>
<tr>
<td>Additional Resources</td>
<td>69</td>
</tr>
</tbody>
</table>
Introduction

C2000™ MCU DesignDRIVE Solutions for Industrial Motor Drives provides a cohesive compilation of resources regarding C2000 Real-Time Control Microcontrollers (MCUs) and their respective industrial inverter and servo drives solutions. This e-book is intended to be your one-stop shop for all related content, allowing you to easily access relevant information on TI solutions in one place. The content centers on various technologies a part of DesignDRIVE – evaluation platforms that let you explore numerous industrial drive subsystems, motor control topologies and control performance bandwidth, with the purpose of helping you get to market faster with a more valuable product.

You will find material on the following key DesignDRIVE technologies:

- Fast Current Loop (FCL) – software that significantly increases a drive’s system torque response and maximum speed – only available on C2000 MCUs
- Position Manager – technology that integrates on-chip interfaces to the leading analog and digital position sensors
- EtherCAT® – hardware and software solutions to help you optimize the EtherCAT stack for your custom C2000-based drive
- Current Sensing – invasive measurement method where the voltage across the sense element is measured to determine the current value
- Flexible Actuation – high-performance method of converting electrical energy to mechanical energy for pulse width modulator (PWM) modules
- System Protection – technology that minimizes latency in MCUs in order for the system to react quickly and resiliently to any abnormal events

You will also find an overview of products relative to the C2000 MCU product line, as well as tools and trainings you may find useful to further your understanding of C2000 DesignDRIVE capabilities.
DesignDRIVE Platform and Architecture

Easily increase functionality in motor drive applications

If you’re a developer of servo drive or motor drive applications, then you’re facing a growing number of technical questions regarding your future designs:

- Is it possible to design one controller capable of handling different motor types?
- How do you stay up with the latest position sensor standards and the revisions?
- Which current sensing technique is best for your application?
- What about all those industrial networking options?
- And now your customers are also demanding integrated functional safety?

Taken all together, these questions may seem overwhelming. At the same time, your company competencies are probably in motion control and motor control. Therefore, time spent exploring and developing new solutions to these questions is actually taking you away from building on those core competencies.

With the C2000™ microcontroller (MCU) DesignDRIVE software and kit, we want to help you spend more of your time differentiating your product in your core areas and less of it evaluating new technologies that will eventually become table stakes in the industry. Using DesignDRIVE software will ultimately help you get to market faster with a more valuable product.

The DesignDRIVE kit and example software offers an easy path to begin exploring current sensing technologies and interfacing various position sensors and motor types. The on board expansion slots make the kit ready for adding real-time EtherNet as well as functional safety circuits. The integration and flexibility of the kit will allow you to investigate combinations of solutions without the need to fabricate your own drive hardware every time.

Based on the real-time control architecture of TI C2000 MCUs, DesignDRIVE is a deal for the development of industrial inverter and servo drives used in robotics, computer numerical control machinery, elevators, materials conveyance, and other industrial manufacturing applications. The new DesignDRIVE kit hardware offers an integrated power module, delivering up to 8 amps to drive a three-phase 1 horsepower motor.

In addition, you can also explore options that allow the MCU to be placed on either side of the high-voltage isolation barrier. The kit uses a powerful Delfino™ MCU that integrates dualC28x real-time processing cores and dual CLA, real-time co-processors, thus providing up to 800 MIPS with integrative floating point trigonometry and FFT acceleration.

Along with flexible best in class PWM actuation, the sophisticated sensing peripherals integrated on the Delfino MCU include sigma delta filters, high performance 16-bit ADCs and window comparators, thus enabling the DesignDRIVE kit to support shunt, fluxgate, hall and delta-sigma current sensing simultaneously.

For position feedback, the kit again leverages Delfino MCU peripherals for simplified interfacing to resolvers and incremental and absolute encoders. To help developers jump start their work, DesignDRIVE software is now part of the C2000 MCU ControlSUITE™ library.
Many industrial inverter and servo drive manufacturers have traditionally relied upon field-programmable gate-array (FPGA) or ASIC technology to complete functions that are not supported by commercial, off-the-shelf (COTS) products, like 32-bit microcontrollers (MCUs). However, adding FPGAs and ASICs to software programmable controllers, in order to support position sensor feedback or sigma-delta filtering, as examples, will increase system cost and add development complexity.

In this series of blogs we will take a look at the history of how industrial drive and servo control architectures have evolved, some of the challenges of introducing an FPGA into these architectures and how new capabilities of industrial drive control SoCs (COTS MCUs) shift the cost-benefit model away from using FPGAs in today’s industrial drives.

So how did FPGAs became common practice in inverter drive and servo control architectures? When new system functions were not achievable using the off-the-shelf controllers available at the time, many developers had to implement their specific PWM/IGBT protection scheme, for example, outside of the MCU. Others may have felt that their current loop timing was too short to be handled by a programmable MCU and that it could only be accomplished in logic gates vs. software. Once an FPGA is used in the system, it then becomes a bit of a system design crutch, albeit a logical crutch, to integrate support for the new technologies introduced by an evolving drives and servo market. For example, the FPGA functions expanded to integrate clockwise/counter-clockwise (CW/CCW) and pulse train output (PTO) ports for communications with PLCs and motion controllers.

Next, FPGAs became the in-system destination for supporting emerging standard and proprietary position sensor interfaces like EnDat and BiSS. Further, the interfaces to the modulated outputs of isolated sigma-delta ADCs, typically based on sinc filters, were also integrated into the FPGA device. In addition, a few of the industrial Ethernet standards have made their MAC controllers available in FPGA gates as well. All of these different functions quickly add up to hundreds of thousands of gates and a very expensive FPGA.

While this expansion of drive functions was being absorbed by the FPGA, an interesting new market dynamic was also taking place. Catalog controllers started bringing these functions on-chip delivering off-the-shelf features for any drive developer to use. The difference here is substantial: The on-chip functions are available for the developer to use immediately – that is, to purchase an MCU available in a catalog without the need to construct these solutions using an FPGA. This correlates directly as a decrease in system cost.
and board space savings that are not possible with an FPGA solution. With the birth of new drive control SoCs, like C2000™ F28379 MCUs with DesignDRIVE Position Manager technology, developers can now avoid the many drawbacks cited above.

Now, let’s take a look at some of the challenges of using an FPGA in an industrial drive/servo architecture and how new capabilities of control systems-on-a-chip (SoCs) in the form of COTS MCUs shift the cost-benefit model of using FPGAs for industrial drives.

Many industrial inverter and servo drive manufacturers have traditionally relied upon field-programmable gate-array (FPGA) or ASIC technology to complete functions that are not supported by commercial, off-the-shelf (COTS) products, like 32-bit microcontrollers (MCUs). However, adding FPGAs and ASICs to software programmable controllers, in order to support position sensor feedback or sigma-delta filtering, as examples, will increase system cost and add development complexity. Shouldn’t we really be asking: Are the functions that are being placed in the FPGA bringing any real differentiation to the drive products? Hasn’t it become standard practice for every drive maker to include some of these functions? In short, are these premium-priced FPGA gates being used to fulfill features that have become the table stakes to play in the industrial drives industry?

While FPGAs are re-programmable and are perceived to potentially provide system adaptability and improved system performance, they also carry some drawbacks when compared to the opportunities with today’s MCUs for industrial drives applications. Developers should weigh the impact of the required specialized engineering skill set, the total project effort and the total system cost.

Many drives systems being developed maintain a C programmable microcontroller or microprocessor coupled with an FPGA. The processor’s C code generation and debug development environments are well known and required. Introducing an FPGA into the system now requires an additional development flow and tool set. Despite the claimed advances in ease-of-use of these tools, it is typically not the same engineering staff that develops the MCU C code as well as the FPGA VHDL code. The VHDL coding style and development flow are quite different from MCU software development and require special engineering resources. In addition, it is the FPGA development staff that also must become low-level and system-level experts for the hardware IP they are implementing. Not only do they need to know how to implement the VHDL for a BiSS master, for example, but they also need to know the BiSS protocol since they need to validate that their FPGA implementation meets the BiSS sensor requirements. This specialized engineering skill set may not be something every motion control or inverter manufacturer can afford to staff and it certainly is a diversion of effort away from their true differentiation of motion and motor control performance. Wouldn’t it be easier to just use a microcontroller that supported BiSS encoders natively?

From a development perspective, managers need to view FPGA creation as a custom development, effectively. Their development teams have an additional degree of ownership and responsibility for the product features taken to market in the FPGA. If the VHDL is not coded properly, they cannot turn to the FPGA vendor; they can only turn to themselves as the cause of the issue and to themselves as the source for the remedy as well. When you compare this to the model of using a COTS MCU, the custom responsibilities associated with FPGA development go well beyond the gates designed into the FPGA. The printed circuit board (PCB) impact, the MCU gate-level/register interface, the software abstraction and overall system integration efforts are all non-standard, i.e. not off-the-shelf solutions. See Figure 1 for details. Beyond just development, this model has the added engineering complications in customer support, product maintenance releases and long term conformance as new interfacing components are released or revised. Wouldn’t it be much easier to turn to a standard MCU with these features implemented and a supplier taking responsibility for whole product solutions (hardware, software, tools and designs)?
Next, and possibly the most obvious, is the impact of an additional component to the bill of materials. The cost of the FPGA goes beyond just the unit price impact. The FPGA device will require additional PCB area, as well as pins required for MCU interfacing and power supply. These costs are unavoidable when working with an FPGA, but are not needed when these functions already exist on the drives SoC MCU. In several cases, it is observed that the FPGAs require an additional and more complex power supply circuit than the drives SoC devices by themselves. Also, implementing the FPGA introduces gates that are otherwise unnecessary to the system, such as the register interface to the MCU and the interface to an external analog to digital converter (ADC) for phase current and voltage sensing. A drives SoC includes a high-performance ADC built for drives applications and does not require this extra logic. So, using a single COTS drives SoC includes many opportunities for overall system cost reduction vs. MCU plus FPGA architectures.

C2000™ MCUs with DesignDRIVE technology are COTS MCUs that deliver this higher-level of drive system integration with a whole-product philosophy that benefit drives developers by reducing the need for specialty engineering talents, saving development time and simplifying system costs.

Many industrial inverter and servo drive manufacturers have traditionally relied upon field-programmable gate-array (FPGA) or ASIC technology to complete functions that are not supported by commercial, off-the-shelf (COTS) products, like 32-bit microcontrollers (MCUs). However, adding FPGAs and ASICs to software programmable controllers, in order to support position sensor feedback or sigma-delta filtering, as examples, will increase system cost and add development complexity.
The Delfino MCU portfolio has the ability to achieve a two and one-half microsecond floating-point torque-loop calculation and contains highly flexible 150ps high-resolution PWMs. The CLA real-time co-processor is a great target for these algorithms to minimize the period between the sample and the next PWM command. Up to eight channels of integrated sigma-delta filters are also on-chip and include parallel under/over-range comparators on the same channel. The on-chip comparator subsystem and PWM trip-zone capabilities can invoke a safe PWM state (off) in 50 ns! Delfino MCUs are highly flexible when generating on-chip conditions for triggering a trip-zone event. Many different types of PWM protection concepts can be implemented using these on-chip resources and thus eliminating the need to place the circuits in external FPGAs. See Figure 2 for a drives system overview.

In addition, the new Delfino TMS320F28379S and TMS320F28379D MCUs and the DesignDRIVE Position Manager technology are the latest additions to the Delfino MCU family. Combining the DesignDRIVE Position Manager software with the new Delfino F28379 MCUs, the on-chip functions mentioned above are expanded to allow for easy and direct connections to EnDat2.2 and BiSS-C absolute position sensors – features that required FPGAs in the past. What’s more, by using some of the sophisticated analog circuits on-chip, these same devices are capable of decoding resolver signals as well as angles from SIN/COS transducers.

This solution is the first of its kind to offer the breadth of position sensor support, flexibility, scalability and robustness, which allows developers to decrease system cost by reducing the board area of the FPGA or ASIC.

The C2000 MCU DesignDRIVE technology offers on-chip solutions for industrial drives, integrated into COTS real-time MCUs. DesignDRIVE solutions simplify the system, saving board space and development effort, which frees developers from making unnecessary investments in features that are non-differentiating in the industry. Instead, developers can focus on product differentiation as well as core competencies, like motor control and motion control, and not on building FPGAs or writing the complex code needed to complete non-differentiating tasks.

Start investigating DesignDRIVE Position Manager technology and a range of industrial drive design...
topologies today with TI’s DesignDRIVE Kits (TMDXIDDK379D or TMDXIDDK379D-MTR-BNDL). The kits are available from the TI Store or any authorized TI distributor.

DesignDRIVE Position Manager technology will help you get a compelling industrial inverter or servo drive product to market quickly. For more information on Position Manager, DesignDRIVE, C2000 real-time MCUs or other TI solutions for industrial drives please visit www.ti.com/c2000 and www.ti.com/c2000Drives.

---

### Designing the next generation of industrial drive and control systems

#### Introduction

A typical industrial system requires control, application and connectivity capabilities. The control subsystem directly manages motor operation and feedback, the application directs the overall motion and the connectivity subsystem downloads application, control data and allows the system to be remotely managed. In general, the core technology that makes up the foundation of each of these subsystems is clearly understood. At the high-end, developers continue to innovate new ways to improve overall performance and accuracy. As these technologies mature and the cost to implement them decreases, these solutions, which were available for high-end applications, work their way down the value chain. The challenge that developers of next-generation systems face today is efficiently implementing incremental innovations to provide better performance with lower latency and greater precision for their target application. To expand market share, they need to be able to deliver better functionality such as new feedback algorithms or novel approaches that improve position accuracy and current sensing at lower costs. To achieve this, processors offering higher performance and greater integration are required; however, this approach increases development costs and adds system complexities, which in turn delays time-to-market and ultimately reduces the competitive advantage of delivering next-generation designs. Implementing new technology must be seamless, simple and value added both to end users and developers.

#### A new architecture for next-generation industrial designs

Many original equipment manufacturers (OEMs) have traditionally relied upon field-programmable gate-array (FPGA) technology to augment their microcontrollers (MCUs) and push leading-edge performance of critical functions like torque-loop management. FPGAs, however, increase system cost and are difficult to program. In addition, FPGAs offer a relatively fixed implementation that lacks scalability across multiple applications without requiring a redesign.

The dual-core C2000™ Delfino™ F2837xD MCU from TI makes it easy to implement various mathematical transforms and trig-heavy computations that enable efficient torque-loop management in a programmable processor platform. The Delfino MCU’s dual-core architecture is also designed to maximize hardware and software performance for industrial drive and control applications. For example, its fast torque-loop calculation can reach sub-2 microseconds, which is comparable to FPGA implementations.

The F2837xD MCUs are extending control-loop performance with their fast CPUs further boosted by tightly coupled accelerators. The dual-core MCU is based on TI’s proven C28x CPU. Each CPU core...
provides 32-bit floating-point-processing capabilities at 200 MHz, and dual real-time control accelerators (CLAs) also running at 200 MHz each. Each C28x CPU is augmented by its own Trigonometric Math Unit (TMU) accelerator that provides hardware based acceleration useful for control-based tasks. These four powerful engines are capable of pounding out the equivalent of 800 MIPS or 1600 MFLOPS of performance enabling consolidation of multi-processor architectures in control loop systems (Figure 5).

For example, in an industrial drive application one CPU + CLA + TMU can be used to implement control-side functionality; i.e., the torque loop. The other CPU + CLA + TMU can be used to implement the application side of the system; i.e., tracking speed and position, computing trajectories, comparing motion profiles and so on.

This division of the industrial drive system into control and application segments between CPUs provides developers with a clean partitioning to simplify design. Because only control code is running on one of the CPUs, it is isolated from application code, thus, developers do not need to spend valuable development hours mitigating the potential impact of application code on the responsiveness and latency of real-time tasks (Figure 6).

The Delfino F2837x MCUs enable developers to migrate high-end functionality down the value chain to mid- and low-range applications. It achieves this through a combination of innovative technologies, including:

- Greater processing capacity at a lower cost than the current solutions
- A streamlined, low-latency architecture that provides higher performance in a deterministic manner
- Advanced hardware-based engines that accelerate common but compute-intensive tasks
- Integration of essential functions into the processor’s architecture to reduce external component count and cost
- Simplified new design migration and reuse of an OEM’s existing code investment.
- Pin and software compatibility across devices offering multiple performance and Flash size options, including the Piccolo™ F2807x MCU family as well.

### Table: Processors Comparison

<table>
<thead>
<tr>
<th>Processing</th>
<th>Processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>C28x DSP core</td>
<td>C28x DSP core</td>
</tr>
<tr>
<td>200 MHz</td>
<td>200 MHz</td>
</tr>
<tr>
<td>FPU</td>
<td>FPU</td>
</tr>
<tr>
<td>TMU</td>
<td>TMU</td>
</tr>
<tr>
<td>VCU-II</td>
<td>VCU-II</td>
</tr>
<tr>
<td>CLA core</td>
<td>CLA core</td>
</tr>
<tr>
<td>200 MHz</td>
<td>200 MHz</td>
</tr>
<tr>
<td>Floating Point</td>
<td>Floating Point</td>
</tr>
</tbody>
</table>

*Figure 6: A closer look at the dual C28x cores and accompanying accelerators.*
Boosting system performance

The F2837xD MCU is a powerhouse enabling 800 MIPS of total system performance. This is provided through dual C28x CPUs and dual CLAs. The CPUs also integrate hardware accelerators which enable swift execution of trigonometric-based control functions ideal for very fast current-loop execution and complex math operations common in vibration analysis and encoded communication applications. These hardware accelerators include:

- **Trigonometric Math Unit (TMU):** TI developed the TMU hardware accelerator to assist the main C28x CPU execute trigonometric functions like SIN, COS, ARCTAN and 1/X that are commonly used in applications such as robotic motion where hinged joints require linear to angular translation. These complex functions are computationally intensive and typically require 30 to 90 cycles to complete even when using a CPU with floating-point capabilities. The TMU can be used as an high-octane accelerator to perform floating-point-unit calculations in parallel to the CPU. With an average execution time of five cycles per instruction, the speed of math transforms requiring trigonometric calculations, can achieve 10× improvement in performance versus the competition when the TMU is used (Figure 7).

- **Viterbi Complex Math Unit (VCU II):** This accelerator efficiently processes complex math functions (Figure 8). The VCU has been designed to be flexible in supporting various communication technologies. It can accelerate the performance of communications-based algorithms by as much as 10 times, thus enabling C2000 MCUs to operate at a lower MHz, reducing system cost and power consumption. The VCU II with acceleration is ideal for OFDM interleaving and de-interleaving, Viterbi decoding, CRC calculations and more.

For the typical MCU, key signal-processing operations can consume much of the processing power when performing complex Fast Fourier Transforms (FFT/iFFT) and complex filters. Besides communications, the VCU is very useful for general-purpose DSP applications such as filtering and spectral analysis. From an industrial motor drives perspective, spectral analysis can be used to process motor vibration noise to determine the impact of vibration on a system, estimate the motor operating life and calibrate the control loop to improve efficiency, thus increasing operational efficiency and reducing system downtime.

Using the hardware capabilities of the VCU, drives applications will significantly benefit from the increased performance over a software implementation (Figure 9).
Along with the TMU and VCU II hardware accelerators, the F2837xD MCUs include two real time CLAs, thus enabling processing optimizations through intelligent partitioning of critical control tasks in a drives system.

- **Real-time control accelerator (CLA):** The CLAs are standalone, floating-point processors which are tied to the main CPUs. CLAs are dedicated low-latency CPU-like architectures with direct access to control peripherals. They are pure mathematical engines that operate independently of the main CPU.

The CLAs can be used in a variety of ways to completely offload intensive signal-processing tasks from the CPU (*Figure 10*). For example, the CLA can serve as part of the analog-to-digital conversion by post processing the incoming signals to filter noise and then buffer data in its own random access memory (RAM). In this way, the CPU is involved only when there is an entire block of preprocessed data ready for it to work.

Another way to use the CLA is by performing Fast Fourier analysis on incoming current wave forms, the CLA can then profile a motor’s real-time performance. The profile can be continuously compared to a “golden signature” based on the type of motor. As the profile begins to deviate from the expected signature, indicating a potential fault, the industrial drive system can alert the operator to take preemptive action before failure occurs. Other tasks the CLA can perform are feedback preprocessing, feed-forward control and special signal analysis or packet processing. These are just a few examples of the many possible features that can be implemented using the CLA.

**Low-latency and deterministic architecture**

Although performance is required for industrial drives, system design also needs to be simplified. When enhancements increase design complexity, this in turn could increase the development time. For example, determinism is essential for control-based applications. Determinism in essence means how feedback signals are sampled, processed accurately and arrive in time for control-loop actuation using the PWM signals.

With the loosely coupled memory architecture, like those with caches and/or memory management units (MMUs), accounting for the unpredictable timing of cache misses or MMU lookups makes the determination of *worst-case responsiveness* a very difficult calculation. Typically designers must rely on profiling real-time execution of the system to confirm worst-case operation—the largest possible number of cycles possible—wasted performance. This means that

---

**Figure 10:** The CLA can increase frequency or number of control loops, offloading the C28x to perform more background control and system tasking.
Figure 11: A differentiated architecture; built for industrial drive control.

modifications to system code may require the system to be profiled multiple times to ensure the deterministic limits of the system have not been exceeded.

This C28x CPU core architecture determinism also carries over to the Delfino F2837x MCU’s peripherals. The latency of ADC conversions is small and consistent for every sample. CPU reads and writes to ADC and PWM registers, respectively, are zero wait-state every time. This greatly simplifies overall system design because there is no need to guess what the worst-case execution path is.

The Delfino F2837xD MCU architecture is built upon a deterministic foundation that allows developers to ensure reliability without increasing design complexity (Figure 11). With its tightly coupled memory architecture, there is no need for a cache, thus eliminating the arbitrary delays that arise from cache misses. All memory transactions to static random-access memory (SRAM), Flash and peripherals are designed to fit within finite and consistent busing cycle counts, thus providing highly deterministic throughput.

The dual six-channel direct memory access (DMA) peripheral augments an efficient memory management to ensure data is always available when the CPU or the accelerators need them.

To facilitate efficient communication between cores, the Delfino F2837x MCU uses shared memory where both cores have full read/write access to data. Developers also have access to two message RAMs. Each core has write privileges to one of the message RAMs and read-only privileges to the other. In this way, code on one core cannot accidentally corrupt critical data belonging to the other core. This greatly simplifies messaging, especially for developers new to dual-core design.

Of special note, there are several error-check functionalities that are spread across the subsystems in Delfino F2837x MCU architecture. Non-volatile memory and SRAM offer ECC and parity capabilities. Device-level diagnostics are collated to generate flags, interrupts and external error signal. This can be enabled during power up and run time of the applications.
System consolidation with integrated control and analog peripherals

The hallmark of the Delfino F2837x MCU is its control peripherals. These are the powerful industry proven PWM timers, 32-bit enhanced capture units (ECAP) and quadrature encoder peripherals. Each of the PWM modules is enhanced to support high resolution capabilities on both A and B channels. These high-resolution channels extend 150-ps PWM step resolution to enable high-frequency PWM modulation techniques and advanced control topologies.

Performance is directly impacted by the precision of the PWM control feedback loop. Integrated analog peripherals reduce latency and cost compared to the use of external components. In higher-end control applications such as servo drives, high-resolution feedback is required to provide precise phase-current measurements for low torque ripple and precision positioning; however, for some measurements, precise sample rates are more important than higher resolution, such as when making high-speed, low-side shunt current measurements.

To support different sensing accuracy requirements, the Delfino F2837xD MCU architecture offers flexible ADCs that can support two resolution modes: 16-bit resolution at 1.1 MSPS and 12-bit resolution at 3.5 MSPS. The F2837xD MCU features four independently integrated ADCs which provide simultaneous conversions, thereby enabling industrial systems to accurately monitor multiple signals in real-time (Figure 12). For example, in a servo drive, designers can monitor the phase voltages and currents of a three-phase motor while simultaneously sampling the DC link voltage.

Delfino F2837x MCU offers three 12-bit buffered digital-to-analog converters (DACs) to provide analog actuation signals, including resolver excitation, that are very useful for tracking engineering parameters at the system level. Further increasing integration is the availability of eight sigma-delta demodulators/filters. Because of the high voltages associated with industrial motor control, isolation is required when measuring feedback signals. Developers can use TI’s AMC130x delta-sigma converter, for example, to convert analog values into a digital bit stream that feeds directly into Delfino F2837xD MCU’s Sigma-Delta interface and is reconstructed by the filter. This enables hot-side/ high-side current sensing of motor phases, providing the required feedback fidelity, which is essential in high-performance industrial drives.

Figure 12: System integration is enabled with high-integrity analog and control peripherals incorporated on chip.
Eight windowed comparators are integrated into the Delfino F2837xD MCU architecture, providing over-voltage or under-voltage “trip points” and operate independently from the CPU so there is no additional CPU loading. The comparators are also fast acting and minimize latency with trip signals so the system can react quickly to any abnormal events or over/under limit conditions. The comparator trip events can be configured to help provide a full shut down action in the case of a catastrophic event in fifty nanoseconds, making the system more resilient in industrial drive and power systems.

**Position Manager**

Historically, interfacing a position sensor to an MCU could be a time-consuming task that often involves the integration of the communication protocol into an FPGA or the programming of an additional MCU with the decode protocols. In addition, this situation is exacerbated by the fact that there are multiple encoder protocols available, each suited to certain types of functionality and subsystems. System design teams might be forced to develop several protocol-specific FPGAs which would not scale effectively from one application to another. Of course, this type of FPGA implementation would add cost to the system by increasing the system’s electronic component bill of materials (BOM), impacting the necessary board space and requiring lengthy development cycles. Moreover, developers also have to complete extensive compliance testing to certify conformance with industry standards.

This situation begs for a solution that would simplify the interfacing of position sensors to control elements in industrial drive systems and thereby free designers to concentrate on features and functionality that would make their systems truly distinctive, as well as more competitive, in the marketplace.

**Integrating position feedback**

Starting with the processing capabilities required by sophisticated and precise control systems, the C2000 Delfino F28379D and F28379S MCUs are equipped with a full complement of on-chip resources, including DesignDRIVE Position Manager technology supporting today’s most popular off-the-shelf analog and digital position sensor interfaces (Figure 13). This relieves system designers from many of the more basic, repetitive tasks, saving design time.

---

**Figure 13**: A closer look at the C2000 single-core F28379S MCU with DesignDRIVE position manager.
TI has extensive expertise with interfacing position sensors to digital controllers. Beginning with standalone interface solutions for resolver-to-digital solutions, such as the TMDSRSLVR, TI has continued to add to its position feedback interface support. Expensive resolver-to-digital chipsets have been replaced by C2000 MCU on-chip capabilities, leveraging high-performance ADCs and DACs. Moreover, the powerful trigonometric math processing of C2000 MCUs is particularly well-suited to the additional processing needed to calculate the angle, and extract high-resolution speed information from a resolver’s amplitude modulated sinusoidal signals.

C2000 F28379 MCUs support up to three enhanced quadrature encoder pulse (eQEP) modules. The eQEP modules interface directly with linear or rotary incremental encoders that are counting pulses to obtain position (once an index is known), direction and speed information from rotating machines used in high performance motion and position control systems. In addition, the eQEPs can be employed to interface to pulse train output (PTO) signals generally output by a programmable logic controller (PLC) in industrial automation for motion control. Also, C2000 MCU eQEPs can interface to clockwise/counter clockwise (CW/CCW) signals. CW/CCW signals are typically used in conjunction with stepper or servo drives for controlling motors or other motion-based hardware.

Resolver and QEP capabilities provide fast, efficient and integrated solutions for effectively interfacing position sensors with C2000 Delfino MCUs. The next step has been to extend that support with complementary solutions that would allow the MCU to connect directly to more advanced digital and analog position sensors.

**DesignDRIVE Position Manager technology**

Available through TI’s DesignDRIVE platform, Position Manager technology takes advantage of the on-chip hardware resources of the C2000 Delfino F28379S and F28379D MCUs to interface to the most popular digital and analog position sensors. Already incorporating support for incremental encoders (eQEP), CW/CCW communications and standalone resolver solutions, Position Manager adds solutions for analog position sensing, integrating both resolver excitation and sensing, as well as a SinCos transducer interface/manager (Figure 14). Unique to C2000 MCUs, Position Manager combines the analog sensor support with the popular digital absolute encoders, EnDat 2.2 and BiSS-C, giving system designers a wide range of position sensor types to choose from.

This integrated Position Manager technology offers system designers a real opportunity to accelerate development cycles and reduce BOM costs by eliminating the need for an FPGA to interface a specific...
encoder to the MCU or by drastically reducing the size of the FPGA that may still be needed for other functions. Figure 15 demonstrates how Position Manager technology relieves system designers from the burden of developing the high- and low-level software drivers, as well as any custom hardware and logic that previously may have been implemented on an external FPGA. Example closed-loop, position-sensor-based control projects downloaded from DesignDRIVE can be modified for integration into customer projects. The lower system layers are provided on-chip or through reference designs and a ready-to-use library of application programming interface (API) modules.

In addition to reducing development time, Position Manager technology also decreases the compliance and interoperability testing that system manufacturers have undertaken in the past. The Position Manager technology is fully tested across a variety of sensors. Please see the user’s guides for details on the testing results. Moreover, future revisions and updates to the applicable standards will also be supported by Position Manager technology.

**New position sensor interfacing capabilities**

With its rich heritage of position feedback technologies as a starting point, TI has been able to expand its position-sensor interface solutions with enhanced capabilities and performance. Table 1 shows some of the solutions available through DesignDRIVE Position Manager technology.

By enabling a direct connection between a C2000 MCU and a position sensor, Position Manager technology frees developers from the more mundane tasks of device connectivity so they can focus on the features and capabilities that will make their system solutions truly distinctive in the marketplace with significant competitive advantages.

For more information on Position Manager, please refer to the Position Manager White Paper, the Position Manager solutions User’s Guides and the DesignDRIVE applications page.

**DesignDRIVE software examples optimize the evaluation of the latest C2000 MCU innovations**

With DesignDRIVE, the example software is built to illustrate how to use the new architectural, peripheral and CPU innovations (Figure 16). The project(s) show:

- How to improve your current-loop timing without impacting your control bandwidth by off-loading computations to the CLA
- The benefit of the Trigonometric Math Unit on current-loop timing
- How to interface with many different position sensors

![Figure 15: EnDat 2.2 solution example: Stackup vs. FPGA.](image)

<table>
<thead>
<tr>
<th>Sensor type</th>
<th>Speed</th>
<th>Tested length</th>
<th>Resolution</th>
<th>C2000 MCU supported devices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incremental</td>
<td>12,000 rpm</td>
<td>N/A</td>
<td>Encoder dependent</td>
<td>F2803x, F2806x, F2807x, F2837xS, F2837xD</td>
</tr>
<tr>
<td>EnDat 2.2 / 2.1</td>
<td>8 MHz</td>
<td>100 m</td>
<td>Up to 35 bits</td>
<td>F28379S, F28379D</td>
</tr>
<tr>
<td>BISS-C</td>
<td>5 MHz</td>
<td>100 m</td>
<td>Up to 26 bits</td>
<td>F28379S, F28379D</td>
</tr>
<tr>
<td>1-format (Tamagawa)</td>
<td>2.5 MHz</td>
<td>100 m</td>
<td>–</td>
<td>F28379S, F28379D</td>
</tr>
<tr>
<td>SIN.COS</td>
<td>12,000 rpm</td>
<td>70 m</td>
<td>± 4.5 arcsecs</td>
<td>F2807x, F2837xS, F2837xD</td>
</tr>
<tr>
<td>Resolver</td>
<td>&gt;3,000 rpm</td>
<td>1 m</td>
<td>&gt; 13.47 ENOB</td>
<td>F2802x, F2807x, F2837xS, F2837xD</td>
</tr>
</tbody>
</table>

Table 1: Position Manager solutions by sensor type.
- How to configure the on-chip over-current protection circuits
- How to communicate with isolated Delta-Sigma ADC modulators
- How to use shunt resistors and fluxgate sensors for current sensing
- How to use (and replace as needed) simple P-I control loops for current, speed and position

**Seamless migration and development**

Multiple processing units and accelerators can substantially improve system performance. However, many OEMs are building on existing designs and have made a significant investment in developing a code base with Delfino and other MCUs.

TI understands that developers need to be able to exploit architectural enhancements seamlessly with simplified partitioning in the firmware. Use of the TMU, for example, is managed by the C compiler. When a native TMU function is available for use, the compiler will automatically utilize the TMU instead of calling a function from the math library. Thus, existing C28x CPU-based designs can take immediate advantage of the TMU’s 5× performance boost without any code needing to be rewritten. The TMU can boost the performance of MathLAB®/ SIMULINK®-based application code, as well. This also improves portability of intellectual property (IP) since the same code can be used with TI MCUs with and without TMU capabilities. The memory subsystem offers a very flexible code protection mechanism to help vendors and developers to exchange valued-added IPs.

To accelerate development, TI and its partners offer a wide variety of software libraries, tools, development kits and technical support as part of its extensive development ecosystem. For example, math libraries are available for both the CPU and CLA to assist developers in getting the highest performance from the Delfino F2837x MCU. TI also provides a wide range of low-level and application-specific libraries to accelerate design of control applications, as well as development boards that provide developers with easy access to all of Delfino’s control-based functionality.
For applications that require communications, the Delfino F2837x MCU provides several options of serial ports: USB, UART, SPI, CAN and I²C. High-speed serial peripheral interface (SPI) is also available for inter-processor connectivity and network connectivity.

For applications that require Ethernet and real-time Ethernet connectivity and protocol, TI's Sitara™ AM335x processors are available as companion communications processors. The Sitara AM335x processors are built around ARM® Cortex®-A8 cores with differentiated peripherals and certified industrial communication stacks/protocols, such as PROFINET®, EtherCAT® and more. These devices are fully featured to support the ARM ecosystem for extended application processing, if required.

Delfino F2837x dual-core devices offer scalable derivatives to meet several value points. This enables applications from low-end industrial drives to energy conversion across the industry. Furthermore, to support a wider breadth of industrial drive applications, while maintaining code and pinout compatibility, single-core versions of the Delfino F2837x and Piccolo F2807xMCUs are also available.

**Conclusion**

Powerful and programmable MCUs like TI's C2000 Delfino F28379 MCUs represent the next step toward industrial drive control systems-on-chip (SoC). They empower more effective and efficient system architectures for ancillary processing or auxiliary drive peripherals by eliminating the need for an external FPGA or by reducing the size of the FPGA significantly.

With the Delfino F2837x MCU, TI has redefined how industrial drives are designed. With its focus on performance, integration, simplicity and transparency, the Delfino F2837xD MCU architecture enables developers to implement classical and proven control systems with next generation capabilities. Its advanced hardware based engines and high level of subsystem integration provide greater performance at a lower cost and smaller system footprint. Developers can also speed time-to-market through simplified design based on its low-latency and deterministic architecture, as well as using the DesignDRIVE IDDK and software examples seamlessly implement Delfino F2837x MCU's architectural enhancements in new and existing industrial drive applications.

Get started with DesignDRIVE—a single hardware and software platform that makes it easy to develop and evaluate solutions for many industrial drive, motor control and servo topologies. DesignDRIVE offers support for a wide variety of motor types, sensing technologies, encoder standards and communications networks, as well as easy expansion to develop with industrial communications and functional safety topologies, thus enabling more comprehensive, integrated drive system solutions. Based on the real-time control architecture of TI's C2000 MCUs, DesignDRIVE is ideal for the development of industrial inverter and servo drives used in robotics, computer numerical control machinery (CNC), elevators, materials conveyance and other industrial manufacturing applications.

**DesignDRIVE Industrial Drive Development Kit (IDDK)**

Jump start your industrial drives and servo control evaluation and development with the DesignDRIVE Development Kit, or IDDK, which includes:

- Examples of vector control of motors, incorporating torque, speed and position
- Multiple current-sense topologies – Supports reinforced, isolated Delta-Sigma modulator ADCs, Hall / Fluxgate sensors and shunt-resistor current sensing
- Analog and digital position sensor interfaces – Supports Position Manager Technology – EnDat2.2, BISS-C, SIN/COS, Resolver and incremental encoders
- Integrated power module and DC Link supply as well as DC Bias supplies for control circuits – operates from a single AC Main connection
- Flexible real-time connectivity – Expansion support for multiple real-time Ethernet protocols
- Configurable power plane location (hot-side or cold-side) for the control circuit
Delfino F28379D controlCARD

The Delfino F28379D controlCARD from Texas Instruments is Position Manager-ready and an ideal product for initial software development and short-run builds for system prototypes, test stands and many other projects that require easy access to high-performance controllers. All C2000 MCU controlCARDs are complete board-level modules that utilize a HSEC180 or DIMM100 form factor to provide a low-profile single-board controller solution. The host system needs to provide only a single 5-V power rail to the controlCARD for it to be fully functional.

C2000 MCU Experimenter Kits

C2000 MCU Experimenter Kits provide a robust hardware prototyping platform for real-time, closedloop control development with Texas Instruments C2000 32-bit microcontroller family. This platform is a great tool to customize and proveout solutions for many common power electronics applications, including industrial drives, motor control, digital power supplies, solar inverters, digital LED lighting and more.

The C2000 MCU Experimenter Kit board hardware includes isolated XDS100 USB JTAG emulation facilitating easy programming and debugging, header pins access to key microcontroller signals, breadboard area for customizable routing, HSEC controlCARD plug-in slot, included controlCARD based on the Delfino TMS320F28379D microcontroller, and more.

Additional resources

Here are additional resources regarding DesignDRIVE Platform and Architecture that you may find of interest:

- C2000 Real-Time Control MCUs – Industrial Drives
- C2000 DesignDRIVE Development Kit for Industrial Motor Drives
- C2000 DesignDRIVE Software for Industrial Drives and Motor Control
- DesignDRIVE On Line Training

Figure 17: IDDK (TMDXIDDK28379D) - U.S. $999.

Figure 18: Delfino F28379D control CARD (TMDSCNCD28379D) - U.S. $159.

Figure 19: Delfino F28379D experimenter’s kit (TMDSDOCK28379D) - U.S. $219.
Fast Current Loop Control

Achieve unprecedented current-loop performance from an off-the-shelf MCU

TI’s expertise in real-time control architectures and experience with industrial drive control systems over the last 20 years has led to many cycle scavenging enhancements to its C2000™ family of real-time microcontrollers (MCUs) and corresponding software solutions.

The latest advancement, Fast Current Loop (FCL), is featured in the DesignDRIVE developers kit and takes advantage of C2000 MCU’s real-time cycle-scavenging architecture, high-performance processing resources and fast data throughput to significantly increase the bandwidth of the current control loop, achieving subcycle updates of the pulse-width modulator (PWM) in less than 1 microsecond and without the assistance of external processing components like a field-programmable gate array (FPGA) or analog-to-digital controller (ADC).

In fact, the field-oriented-control (FOC) processing period, the central processing unit (CPU) time used after sampling and converting the current, is only 460ns on a 200 MHz clock. How is this possible with an off-the-shelf MCU? It’s because C2000 MCUs are not typical MCUs. Their design from the beginning was to minimize the time (in CPU cycles) that it takes to process samples and update the actuation, which subsequent C2000 MCU generations have only improved upon. The many cycle-scavenging data-path features built into the latest C2000 MCUs include an integrated high-performance successive-approximation-register (SAR) ADC, ADC post-processing hardware, single-cycle reads from the ADC, single-cycle writes to the PWM, a trigonometric math accelerator, a code law accelerator and ePWM immediate update mode. You can even use FCL to control two axes at the same time. It’s no problem. Just move to dual-core configurations like the TMS320F28379D and use FCL on each C28x core.

Why should you care about Fast Current Loop? Because the improvements it makes to your current loop enable significant improvements to your speed and position loop control bandwidth. And improvements in these specifications on your servo drive are what your customers are really looking at. More efficient control of their factory automation equipment means more productivity, resulting in more throughput and ultimately greater profitability.

- Less than 1 microsecond current loop
- More than 3 kHz bandwidth on 10 kHz carrier
- Greatly improved performance at high speed
- Innovative controller results in perfect pole-zero cancellation at all times, ensuring stability at higher speeds
From a hardware design perspective, with FCL you can achieve greater bandwidth without increasing your carrier frequency or adding additional processing components. A higher carrier frequency means higher switching losses and additional heat dissipation, necessitating more extensive and expensive thermal-management strategies. More components mean more bill-of-material (BOM) cost, more board space, more current, etc.

Beyond the super-fast FOC processing and the resulting bandwidth improvements, FCL also includes a new, efficient control algorithm option that compensates for the inherent transport delay of the motor drive system. The DesignDRIVE Complex Controller (CC) results in perfect pole-zero cancellation at all times, ensuring stability at higher speeds than those achieved by traditional digital control algorithms.

A C2000 drive control system-on-chip like the TMS320F28379 with FCL delivers similar performance compared to FPGA-based systems while simplifying servo drive development and reducing system costs, power complexities and board space. And compared to traditional MCU-based systems, the FCL can potentially triple the drive system’s torque response and double its maximum speed without increasing the carrier frequency.

A faster current loops pays off in servo motor control

Improving the performance or bandwidth of industrial servo-drive applications like robotic assembly systems, factory automation machinery and material handling systems has several direct effects on a manufacturer’s bottom line. Less waste in the manufacturing process, increased production throughput, and greater product consistency and quality will reduce manufacturing costs and in turn increase a manufacturer’s revenues and profitability.

At the heart of the matter of improving servo drive performance is the system’s current-loop performance or torque response. The current loop is the linchpin upon which the performance of the rest of the system depends. Even though several alternative architectures can provide a pathway to a greater current-loop bandwidth, some are more effective and cost-efficient than others. Only one solution in particular can optimize or eliminate the design trade-offs involving long, drawn-out and complex development programs, expensive bill of materials (BOM) costs, excessive power consumption, cumbersome thermal-management strategies and others. This solution, which is based on TI’s commercial-off-the-shelf (COTS) C2000™ real-time microcontrollers (MCUs) and the enhanced DesignDRIVE development platform, increases the bandwidth of a servo motor drive’s current control loop.

Loops within loops

The architecture of real-time industrial servo drive applications involves several control loops that monitor the motor’s critical parameters. These control loops provide feedback for the purpose of maintaining an established set point as operating conditions change or when responding to a new set point, thus controlling parameters such as current, torque, speed and position. The control system uses feedback loops to ensure that its actions achieve the intended results. If the feedback indicates that a parameter has strayed beyond its target for that cycle, the system must respond quickly and adjust its actions. Responding quickly to sudden disturbances reduces the risk of faulty or inconsistent servo motor operations that will affect the output of the industrial process.

Servo applications typically employ at least three nested control loops for current, speed and position. The current-loop bandwidth is the most critical because it dictates the maximum performance of the speed and position loops. The system’s control loops are interrelated and dependent upon each other, but the current loop is foundational. In fact, the bandwidth of the current loop will limit the maximum performance of the entire servo system. In most systems, the current-loop bandwidth will exceed the bandwidth of the speed and position loops by nearly 10 times or even higher. As a result, a faster current loop is essential to higher system performance.
Current loop close-up

The current loop controls the torque in a servo motor by manipulating the pulse-width modulator (PWM) outputs that drive an inverter. The motor currents are monitored and fed back to the current-loop controller and the controller updates the PWM outputs if necessary. The current-loop feedback path quantifies the analog output of the motor current sensor with a high-precision analog-to-digital converter (ADC), and then feeds the result to the current-loop controller. Several different modules of field-oriented control (FOC) algorithms process this sample before the controller makes any adjustments to the PWM’s outputs.

Many conventional proportional integral (PI) current-loop controllers limit the current-loop bandwidth to approximately 10 percent of the PWM carrier frequency, which is typically in the 10-kHz range. This would yield a current-loop bandwidth of 1 kHz. To increase the bandwidth, some drive systems increase the carrier frequency to 30 kHz or higher, which can increase the current loop’s bandwidth but will also increase the inverter switching losses and demand higher processing performance from the digital control system. A higher switching frequency requires increased gate-drive power, which then increases the size and complexity of the system’s power supply. Plus, the higher switching losses will mean additional heat dissipation, necessitating more extensive thermal-management strategies such as larger, heavier and more expensive heat sinks, fans or coolants.

Clearly, the historical solutions for increasing current-loop bandwidth come with strings attached. The cleanest, most effective architecture would simultaneously address the need for improved processing capabilities to accommodate shorter feedback cycles and retain a lower carrier frequency to avoid the undesirable power and thermal trade-offs. To augment the system’s processing resources, some designs have employed current-loop processing data paths in a field programmable gate array (FPGA) in addition to the MCU, but this only addresses the processing half of the equation and comes with its own set of strings.

FPGA architectures

Adding an FPGA does not eliminate the need for an MCU in a servo motor drive design because the MCU controls the speed and position feedback loops, and in some cases the system’s user interface and connectivity. Inserting an FPGA into the current control loop has a ripple effect throughout the rest of the design. See Figure 20 on the following page.

To accommodate the entire current control loop, the FPGA will require custom design work to incorporate PWM generation, inverter protection circuitry and logic to control the analog sampling. This design work will complicate system development, drive up the gate count and cost of the FPGA, and increase design and support risks.

In addition, most FPGAs do not have embedded ADCs capable of sufficient performance and resolution for servo motor-control applications, thus necessitating the design of discrete ADCs and analog comparator circuitry into the system. Also, the FPGA will likely require parallel input/output (I/O) interfaces to minimize data-transfer latencies from the ADC and enable fast communications with the MCU. Again, the FPGA’s pin count will go up, along with cost and board-space requirements. And lastly, an FPGA design will typically demand a more complex power subsystem, possibly requiring additional power rails and higher currents than the MCU and other components would typically need.

Of course, once an FPGA is present in a design, it invariably becomes the home for functionality that may have been previously implemented in discrete devices or not included in the system at all. In a servo motor-control application, this could mean integrating delta-sigma ADC filters, encoder interfaces, pulse-train output generation and other capabilities. Once this trend begins, the logical end is a very large, high-pin-count, expensive FPGA.
Fast current loop software

An effective alternative that avoids many of the challenges that arise with the addition of an expensive FPGA to a servo-control application would be to implement these functions in the MCU. Outside of TI’s C2000 family of real-time MCUs, few if any such capable MCUs exist. TI harnessed two decades of real-time industrial control system design experience when applying C2000 MCUs to the development of the new Fast Current Loop (FCL) software. The DesignDRIVE development platform with FCL software vastly simplifies the design process while overcoming the less desirable trade-offs.

FCL software, quite simply, breaks many of the long-held assumptions in current-loop design. For instance, designers of servo motor control applications have for years placed limitations on the current-loop bandwidth because they assumed that the controller could update the PWM only once per cycle of the control loop. Now, with an integrated high-performance successive approximation register (SAR) ADC, ADC post-processing hardware, trigonometric math accelerator and other cycle scavenging resources, C2000 MCUs can sample the motor currents, convert them to digital data, process the data and update the PWM generator in less than 1 µs. The field-oriented control (FOC) processing and PWM update takes less than 500 ns. Because of FCL software and C2000 MCU ePWM immediate mode capabilities, subcycle PWM updates occur as soon as possible, instead of waiting for an entire control loop cycle to transpire.

Designers also assumed that achieving a higher current-loop bandwidth would require higher carrier frequencies. So, for example, increasing the bandwidth by a factor of 3 from 1 kHz to 3 kHz would require increasing the carrier frequency from 10 kHz to 30 kHz.

But the C2000 MCU’s ability to perform subcycle PWM updates in less than a microsecond improves the current-loop bandwidth, making it possible to reduce or optimize the switching frequency for a given bandwidth requirement. Tests have shown that FCL can increase the current-loop bandwidth by a factor of 3 from 1 kHz to approximately 3.3 kHz while maintaining a carrier frequency of 10 kHz. Therefore, using FCL software avoids the trade-offs of increased power consumption, greater heat dissipation, and more complex and expensive thermal-management strategies.

Figure 20: System challenges associated with FPGA-based current-loop control.
The TMS320F28379 MCUs are highly integrated for industrial drive systems. On-chip features such as Sigma-Delta filters, analog comparators and PWM protection circuits save the system-level costs when compared to adding these external components. The Position Manager solutions for absolute encoder interfaces as well as Pulse Train Output generation reduce the need to add these logic gates as well. With the release of the Fast Current Loop software, the system-level functions that have traditionally been developed for FPGAs are now all available in the DesignDRIVE solutions library. See Figure 21. Therefore, we can refer to the TMS320F2837x family of MCUs as drive-control systems-on-chip.

Another challenge of high-performance current-loop design has been the stability of the current-loop controller at high speeds. Digital or transport delays progressively reduce the phase margin of the current control loop at higher speeds, eventually leading to a loss of control. Traditional controllers in the current loop do not model these digital delays well.

However, FCL software includes an efficient algorithm that assumes a more appropriate system model, compensating for the inherent transport delay of the system. This is embodied in the DesignDRIVE complex controller (CC) and results in perfect pole-zero cancellation at all times, ensuring stability at higher speeds.

The hardware and technology breakthroughs that contribute to FCL software have emerged from more than 20 years of experience in embedded processing and motor-control systems by the C2000 MCU DesignDRIVE engineers. Certainly, the MCUs themselves have been endowed with powerful, high-performance processing units, co-processors and specialized processing units, but the C2000 MCU’s architecture is just as essential because it enables a high degree of real-time parallel processing through very low latencies and high processing determinism. Ultimately, this makes a C2000 MCU in a servo-drive application a cycle scavenger, saving processor cycles over the entire loop and transforming what may have been longer sequential steps into shorter simultaneous processes.

For example, 200-MHz C28x processing cores are, of course, critical for breaking apart the workload to optimize parallel processing, but so is the realtime deterministic architecture that surrounds them. Because the architecture keeps the system's processing as jitter-free as possible, cycles are not wasted recovering from timing issues or running error-correction routines. There are no data-transfer latencies between the ADC and the C28x cores, or between the cores and the PWM. Transferring a sample from the ADC to the C28x cores or updating the PWM takes only a single cycle. See Figure 22.

![Figure 21: C2000 system integration for industrial drives.](image-url)
Several C2000 MCUs, such as the Delfino™ TMS320F28379D processor, include a powerful ADC post-processing block capable of performing certain sample-preparation routines in dedicated hardware that the C28x cores would normally have executed, saving cycles on the main processors and freeing them to perform other tasks in parallel. The cores themselves have been equipped with co-processors and specialized processing units that prove useful in servo-control applications. For instance, the main processing cores can offload encoder-feedback processing to the control law accelerator (CLA) co-processor, which excels at high-level mathematical processing. Moreover, both the FOC processing and the CC use a specialized processing unit, the trigonometric math unit (TMU). With the aid of the TMU, you can improve Park transform cycle performance by a factor of 8 over comparable MCUs in the industry. In addition, the C28x core’s 32-bit floating-point processing accelerates the execution of several different types of mathematically intense calculations used by the FCL algorithms, such as minimum/maximum, compare and square root.

**Conclusion**

During the development of a servo motor-drive application, designers face a diverse set of subtle trade-offs, many times placing high performance, cost effectiveness, power efficiency and other factors in direct opposition with one another. Simple yet effective solutions may be hard to come by, but fortunately, TI’s experience and expertise in real-time industrial drive control systems has led to many enhancements to the C2000 family of realtime MCUs.

The DesignDRIVE developers kit features the latest enhancement, FCL software. Taking advantage of a real-time cycle-scavenging architecture, high-performance processing resources and the fast data throughput of C2000 MCUs to significantly increase the current control-loop bandwidth, FCL software achieves sub-cycle updates of the PWM in less than 1 µs and without the assistance of external processing elements like an FPGA or ADC. Compared to FPGA-based systems, a C2000 drive control system-on-chip with FCL software delivers similar performance, while simplifying servo drive development and reducing system costs, power complexities and board space. Compared to traditional MCU-based systems, FCL software can potentially triple a drive system’s torque response and double its maximum speed without increasing the carrier frequency.
Fast current loop performance: better than we thought, and measurable in your own lab

In June 2017, I posted about unprecedented current-loop performance from an off-the-shelf microcontroller (MCU) achieved by the C2000™ family with the fast current loop (FCL) software solution. Closing the current loop in less than 1ms had previously been the domain of custom application-specific integrated circuit (ASIC) and field-programmable gate arrays (FPGAs) with parallel data-path architectures.

The DesignDRIVE C2000 FCL release at that time was targeted for the industrial drive development kit (IDDK) and, while you could measure the loop time with a scope, analyzing the control bandwidth of the current loop meant using an expensive dynamometer and control response analyzer tools in your own laboratory environment.

In early 2018, TI released its second version of the FCL solution, with these updates:

- Improved loop control bandwidth
- Free software frequency response analysis (SFRA) tools
- New low-cost evaluation tools – including the dynamometer
- Simultaneous FCL control of two motors (dual axis) example on a single C28x central processing unit (CPU)

Better bandwidth

The most important update to share is the fact that our own in-depth measurements exceeded our previous expectations of “over 3 kHz” of control bandwidth versus traditional approaches. Figure 23 summarizes the results of FCL tests versus traditional current-loop designs. In each case, we designed control loops for a target bandwidth and measured the phase margin. By plotting these results against each other, you can see the dramatic difference in bandwidth at a given phase margin between the two control approaches.

Every one of these tests uses only a 10 kHz pulse-width modulation (PWM) carrier frequency and in-line current sensing enabled by INA240 sensors on the low-voltage three-phase gallium nitride (GaN) inverter BoosterPack. Using in-line sensing enables the current to be sampled twice per PWM period during the polarity transition points. The results shown in both curves of Figure 23 leverage the double sampling technique. (For reference, an incremental or ABZ encoder provides the shaft angle feedback.)

Most in the industry measure their current-loop bandwidth at 45 degrees of phase margin in order to avoid instability situations. If you compare the results in Figure 23 at 45 degrees (the dashed red line), you will see that the FCL controller delivers about 5kHz of control bandwidth, while the traditional approach is less than 2 kHz.

Figure 23: Gain margin vs. phase margin results comparing FCL to traditional current-loop control.
For more details on how we obtained this plot, see the technical brief, “Performance Analysis of Fast Current Loop (FCL) in Servo Drives Using SFRA on C2000™ Platform.” Or better yet, see the Additional Resources section at the end of this post and validate these measurements for yourself.

**SFRA tools, now with motor-control support**

So how will you measure the frequency and phase margin of FCL or the specific controllers that you design in the future? One option is to leverage the free C2000 SFRA tool. Part of the powerSUITE tool chain, the SFRA tool enables the measurement of the open loop gain and plant frequency response of a closed-loop digital controller using only software. Taking advantage of software libraries linked into the C2000 project and executed in silicon as well as the digital debugging connection, you take the measurements and plot them in the SFRA PC application. This makes the in-system measurement of digital control loop bandwidth, gain margin and phase margin quick and easy.

Now available for motor current control analysis as well, these tools are included for free in the FCL download. **Figure 24** shows an example of the frequency response plots of an FCL project as presented by the SFRA tool.

**But I need a motor and a controlled load to measure the response**

That’s true, and a dynamometer machine and the hardware tools you need to control it can be really expensive. But not in this case! You can replicate our performance analysis with a bundled package that includes a C2000 Delfino™ controller, two GaN three-phase inverters and dual motor dynamometer in a single orderable product on the TI store. See **Figure 25**.

Available for less than $600, this is the same configuration that produced the test results shown in **Figure 23**. The inverter BoosterPack development boards included in this bundle also include the in-phase current sensing, and therefore enable double sampling of the current per PWM period.

![Figure 24: SFRA PC graphical user interface.](image)

![Figure 25: A low-cost GaN-based bundle for motor-control performance testing.](image)

![Figure 26: A low-cost, shunt-based sensing bundle for motor-control performance testing.](image)
If you are interested in a quick way to see FCL results on your own bench and establish your own performance motor-control development and testing environment, this bundle can get you moving fast. It also serves as a great tool for testing your motor-control ideas beyond FCL – for speed and position loop control, for example.

We have also tested FCL with TI’s DRV8305N three-phase motor drive BoosterPack evaluation module with integrated sense amplifiers. In this case, the current is sampled on the inverter shunts; therefore, a single sample is taken per PWM period. These tools have also been packaged with the two-motor dynamometer and are shown in Figure 26.

**Dual-axis support**

Another development with the FCL release is that the software project incorporates the control of two motors concurrently from a single C28x core. We decided to use a second FCL instance in order to incorporate the load-side control. The coordination of analog-to-digital converter (ADC) sampling, field-oriented control (FOC) processing and PWM updates are resolved in this release. We used the second controller to act as the load, but in your dual-axis servo drive architecture, you can use these axes independently for whatever your industrial application may require. When you investigate this for yourself, you will see that both loops are capable of delivering the same bandwidth results as in Figure 23.

---

**Performance analysis of last current Loop (FCL) in servo drives using SFRA on C2000™ platform**

**Abstract**

The latest C2000 Delfino™ family of microcontrollers supports fast current loop (FCL) implementation for high bandwidth control of motor drives over a wide speed range in high end multi axes industrial servo control applications. Due to the stringent computational demands of the control algorithm and the demands of interfacing to various position feedback sensors used in these applications, traditionally FPGAs and discrete ADCs have been widely used to implement the core control solution. However, recent C2000 MCUs can cost effectively replace FPGAs and external ADCs in these applications and exceed the functional requirements due to superior features. This technical brief analyses the functional behavior of the servo loops using fast current loop algorithms in terms of bandwidth and phase margin. The test bench consists of a motor-generator set (2MTR-DYNO), an F28379D launch pad (LAUNCHXL-F28379D) and TI’s low voltage GaN inverter modules (BOOSTXL-3PHGANINV). The results show a three times improvement in current loop bandwidth for a given phase margin.

**Introduction**

High performance motor drives in servo control applications are expected to provide high precision and high bandwidth control of current, speed and position loops for superior control of end applications such as robotic arm, CNC machines, and so forth. Since the current loop makes up the inner most control loop, it must have a high bandwidth to enable the outer speed or position loops to be faster. Hence, a high bandwidth FCL is needed in high performance industrial servo control applications. However, the delay due to ADC conversion and algorithm execution limit the current controller bandwidth to about a tenth of the sampling frequency.

The major challenge in digital motor control systems is the influence of the sample and hold (S/H) and transportation delay inside the loop that slows down the system. In a time critical algorithm such as the fast current loop, the latency between feedback sampling and PWM update should be as small as possible. A minimal current loop time not only helps to improve the control bandwidth, but it also enables a higher modulation index (M-I) for the inverter. A higher M-I translates into the higher phase voltage that the inverter can apply on the motor. Higher loop latency will reduce the maximum available voltage and can restrict the
rate of current change in the motor, thereby, adversely impacting the controller performance.

In addition to latency considerations, the flexibility to interface various position encoders to the control system can justify the need for FPGAs and external ADCs to implement the fast current loop. However, with the advent of new generation C2000 microcontrollers, it is now possible to replace the functionality of FPGAs and external ADCs. In addition, the MCU can also run speed and position loops with a minimal board space, thereby providing a cost effective solution. This paper outlines the implementation of fast current loop on a C2000 platform running two mechanically coupled motors, and verifies the frequency response of the control loops using TI’s Software Frequency Response Analyzer (SFRA) software library. Dynamic frequency response analysis in real-time on a motor drive system is unique among MCU suppliers and is currently capable only on C2000 MCUs.

**Control system**

The speed control block diagram of a field oriented control (FOC) based AC motor control system is shown in **Figure 27**. The current loop is highlighted because this is the inner most loop and has a higher influence on the bandwidth of the outer speed and position loops. For the outer loop to have a higher bandwidth, the inner loop must have a far higher bandwidth, typically more than 3 times.

**PMW update latency**

The major challenge in implementing the current loop lies in reducing the latencies between feedback sampling and PWM updates. In traditional control schemes, this latency is typically one sampling period thereby delaying the control action. In other words, it leads to one sampling period of inaction to any disturbances in the loop. For a fast current loop, this delay must be as small as possible to improve the loop performance over the wide operating speed range of the motor. Typically, a latency of one microsecond or less is considered acceptable in many applications, and is illustrated in **Figure 28**. This requires a controller with a fast compute engine, a fast ADC, low latency control peripherals and a superior control algorithm. The TMS320F2837x has the much needed architecture and hardware on-chip to deliver higher performance. The FCL library running on this C2000 MCU, provides the high performance algorithmic support.

![Figure 27: Speed control block diagram of AC motors.](image-url)
FCL library

TI provides the algorithm for fast current loop as a linkable library that utilizes the following features in the F2837x MCU:

- 4 high speed 12-/16-bit ADCs
- Trigonometric and Math Unit (TMU)
- Parallel processing core - Control Law Architecture (CLA)
- Enhanced PWM
- Enhanced QEP or Absolute encoder feedback

The block diagram of the FCL library with its inputs and outputs is shown in Figure 29. The earlier version of the FCL library, Fast Current Loop Library, partitions the algorithm across CPU, CLA and TMU to bring down the PWM update latency to less than one microsecond. Further optimization is possible if the algorithm is written in assembly. FCL is available as a published example as in C2000 DesignDRIVE application through controlSUITE software. However, in this context of the analysis, the coding process is simplified using only one CPU. The Fast Current Loop (FCL), frequency response analysis (SFRA) and multiple axis control are integrated in this example to facilitate performance evaluation. The entire control algorithm for both motors is run on CPU1 without using the CLA or the second CPU core on the F28379D. The modified FCL library, Fast Current Loop (C28x) Library, supports a complex PI controller.

SFRA library

Texas Instruments’ Software Frequency Response Analyzer (SFRA) library is designed to enable frequency response analysis on any digitally controlled closed loop system using software only. This enables performing frequency response analysis of the closed loop system with relative ease as no external connections or equipment is required. The optimized library can be used to identify the plant and the open loop characteristics of a closed loop system. In this study, it can be used to get stability information such as the gain margin, phase margin and bandwidth to evaluate the control loop performance.

Figure 28: PWM update latency.

Figure 29: Fast current loop (FCL) library block diagram.
Consider a digitally controlled closed loop power converter, as shown in Figure 30, where:

- $H$ is the transfer function of the plant that needs to be controlled
- $G$ is the digital compensator
- $GH$ is referred to as the open loop transfer function
- $r$ is the instantaneous set point or the reference of the converter
- Ref is the DC set point reference
- $y$ the ADC feedback
- $e$ the instantaneous error
- $d$ the sensor noise/disturbance
- $u$ the PWM duty cycle

It is clear from Equation 1 and Equation 2 that by knowing the open loop transfer function ($GH$), one can determine if the system meets the objectives. A Bode plot of the open loop transfer function $GH$ is frequently used for this purpose and quantities such as gain margin (GM), phase margin (PM) and bandwidth (BW) are often used to comment on the stability and robustness of a closed loop system.

SFRA library can enable measurement of the $GH$ and $H$ frequency response by software. This data can be used to:

- Verify the plant model ($H$) or extract the plant model ($H$)
- Design a compensator ($G$) for the closed loop plant
- Verify the close loop performance of the system by plotting the open loop ($GH$) Bode diagram

As the frequency response of $GH$ and $H$ carry information of the plant, the data can be used to comment on the health of the power stage or control loop by periodically measuring the frequency response.

This library is used to study the current loop performance in motor drive system.

**Evaluation**

The system evaluation consists of two parts:

- Implementing FCL for inner current loop control in two servo drives
- Performance analysis of FCL using SFRA and obtain the loop bandwidth

Implementing FCL consists of integrating the library inside the speed control loop. However, performance analysis and bandwidth determination needs some considerations. In order to study the current loop bandwidth, the back emf component of the loop needs proper decoupling or compensation otherwise it can influence and distort the analysis. At zero speed when there is no back emf, the loop performance can be analyzed using frequency response analysis methods. This can be used as a reference to verify the same at different speeds to see if there is any change in the controller behavior. This helps to ensure the robustness of controller implementation at various speeds.
To perform this effectively, a motor generator set, like the 2MTR-DYNO, that can hold zero speed is helpful. The software is built such that two different motors, controlled independently, are coupled together as motor-generator for performing frequency response analysis on the current loop of the generator while the other motor is controlled in constant speed mode. Since the speed is held constant by the drive motor, the generator current loop sees minimal speed jitter, if any. This helps to obtain a frequency analysis report free of speed related jitters. Once the user is comfortable with the test, the same can be repeated on drive side motor too for verification.

The evaluation setup is built using hardware that is readily available from TI and is given in the next section. It consists of an F28379D MCU based LaunchPad, inverter BoosterPack based off GaN+INA240, and a motor-dyno set for load testing the drive motor.

The inverter booster pack provides in-line current sense feedback using the high performance current sense amplifier INA240 that provides the instantaneous motor currents at all times. This allows the FCL algorithm to study several sampling schemes and their impact on loop bandwidth.

**Hardware**

The details of the evaluation hardware, all available from TI eStore, and reference to their user guides are given below:

- Controller - LAUNCHXL-F28379D - 1 unit – LAUNCHXL-F28379D Overview User’s Guide
- Inverter (INV) - BOOSTXL-3PHGANINV - 2 units – BOOSTXL-3PhGaNInv Evaluation Module User Guide
- Motor Dyno Set - 2MTR-DYNO - 1 unit (2 motors with mounting hardware)

![Figure 31: Layout of LAUNCHXL-F28379D.](image)

A lab power supply (variable) rated at 48V/5A will be sufficient to run these experiments. In this document, the inverter is sometimes referred as digital motor control (INV) kit for convenience.

**Launch pad**

For immediate reference, the layout of LAUNCHXL-F28379D is given in **Figure 31**. For further details, see the LAUNCHXL-F28379D Overview User’s Guide.

It can support controlling two motors with QEP position feedback. In addition, it has a couple of SPI ports available that can be used to drive SPI configurable inverters. While this board can support external AFE to bring in Absolute encoders signals, this document is currently limited to show the analysis using QEP as the position feedback mechanism.

**PWM DACs**

LaunchPad has four PWM-DACs, referenced in this document as PWMDAC1-4. They are available on jumper pins J4-31 and J4-32, and J8-71 and J8-72. PWM-DACs are basically low pass filtered signals that are originally given out as high frequency PWM carrier signals modulating the signal of interest. In the evaluation project, these PWM-DACs are used to display intermediate system variables for debug purposes.
DACs

LaunchPad also has a couple of DACs available on jumper pins J3-30 and J7-70. Depending on the booster pack in use, they may or may not be available as DAC. With the GaN BoosterPack, this functionality is not available as it is used up by this BoosterPack.

Inverter - BOOSTXL-3PHGANINV

For immediate reference, the functional block diagram of BOOSTXL-3PHGANINV is given in Figure 32. For more details, see the BOOSTXL-3PhGaInInv Evaluation Module User Guide.

Because of the dimensions of the GaN + INA240 booster pack (BP), it is not practical to fit two of them on the same side of a launch pad. One of the mounting methods is shown in Figure 33 that uses 2x10 headers for not only extending the signals out but also for providing some spatial clearance between various PCBs giving better clarity. Depending on user convenience, a passive extender board can be designed to bring out a pair of launch pad headers to mount the second booster pack or a short flat 20 pin ribbon cable can be used.
Two motor dyno

The two motor dyno setup helps to perform load test on the drive motor by mechanically coupling it to the other motor that acts as a generator. The kit comes with a coupler, mounting screws and key. The motor dyno set can be assembled as shown in Figure 34.

Software

The software is developed based on FCL and SFRA libraries released already in C2000 MCU software environment called controlSUITE. FCL is used to improve the current loop bandwidth and SFRA is used to do frequency response analysis of any control loop. For ease of evaluation, the FCL library is customized to execute out of CPU1 to control two motors. This solution can adapted using F28379D’s dual CPU cores and their CLAs to further speed up the computation, which will facilitate a higher DC bus utilization by the inverter and increase the motor speed range. The software is built such that two different motors can be controlled independently and then coupled together as motor-generator for performing frequency response analysis.

Incremental system build

The system is incrementally built up like any other projects in controlSUITE. In each build level, a certain operation of the system, be it hardware or software, is verified and integrated incrementally. In the final build level, all operations are integrated to show case the complete system. Software modules are written as either C macros or C callable functions. Table 2 summarizes the functional integration and library integrated in each incremental system build.

FCL implementation is gradually built up through build levels 1 through 4. Frequency response analysis is gradually built up through build levels 5 and 6. Build levels 1 through 4 can be done with the motor shafts coupled together or decoupled, but levels 5 and 6 require the motor shafts be coupled so as to perform load tests as well as frequency response analysis.

QEP calibration

Both motors in the test set up have QEPs, and they both need to be calibrated before using them for control. Calibration is nothing but knowing the position of QEP index pulse in the circular span of one rotation. Each motor will be spun for a maximum of one rotation until it catches its QEP index pulse. When the motor shafts are disengaged, it is trivial to do the alignment. Motor 1 shaft is brought to alignment followed by spinning it one direction until its QEP index pulse is received. Then the same is repeated on motor 2. However, when the two shafts are connected together, a coordinated sequence is followed. Firstly, the shaft will move to a certain position and hold for a while (alignment by motor 1) before spinning slowly. When QEP1 index pulse is received, motor 1 will stop spinning the shaft and leave it to the control of motor 2. Notice that the shaft

<table>
<thead>
<tr>
<th>Build Level</th>
<th>Functional Integration</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level 1</td>
<td>PWM generation</td>
<td></td>
</tr>
<tr>
<td>Level 2</td>
<td>Run motor open loop - feedback verification and calibration</td>
<td></td>
</tr>
<tr>
<td>Level 3</td>
<td>Run motor current loops using FCL library</td>
<td>FCL</td>
</tr>
<tr>
<td>Level 4</td>
<td>Run motor speed loop with the inner current loop using FCL library</td>
<td>FCL</td>
</tr>
<tr>
<td>Level 5</td>
<td>Run both motors - one as motor and another as generator</td>
<td>FCL</td>
</tr>
<tr>
<td>Level 6</td>
<td>Use SFRA GUI to run SFRA on target CPU - CPU using SFRA library</td>
<td>SFRA</td>
</tr>
</tbody>
</table>

Table 2: Testing modules in each increment system build.
will realign to another position for a while (alignment by motor 2), before spinning in opposite direction until QEP2 index pulse is received.

**FCL integration - build levels 1-5**

By working through build levels 1 through 4 as mentioned in Table 2, the control hardware can be calibrated for analog feedbacks, QEP can be verified and FCL library can be integrated into the controller to run two motors independently. The control block diagram of build level 4 is shown in Figure 35. Completion of this build level signifies that FCL integration is successful and that the motors could be independently controlled in speed loop. Once this is established, the motor shafts can be coupled together for verification of motor generator operation by working through build level 5.

**SFRA integration - build level 6**

Successful completion of build level 5 enables performance analysis of the current loop in build level 6. The control block diagram of the twin motor assembly is shown in Figure 36.

Build level 6 integrates the SFRA library where it injects a small noise into the control loop under test and monitors the system response and identifies the magnitude and phase response of the loop at various

---

**Figure 35: Level 4 incremental system build block diagram.**

**Figure 36: Level 6 incremental system build block diagram.**
noise frequencies. This information is all stored in the CPU memory as local data. Using a PC based GUI tool for SFRA, this SFRA data is transferred through USB/JTAG and the GUI tool plots the gain response and phase response of the system at the frequencies the loop was tested. A typical plot is shown in Figure 37 and Figure 38. From the gain and phase plots, it identifies the gain margin, phase margin and loop bandwidth and displays them at the bottom.

These are plots of the D-axis current loop FRA showing a bandwidth of 992Hz and 5420Hz as against the designed 1100 Hz and 5000Hz bandwidths, respectively. As mentioned earlier, the test can be performed at zero speed and obtain the bandwidth, phase margin and gain margin for reference. This test can then be repeated at different speeds and load conditions to verify if there is any change in bandwidth or phase margin from that obtained at zero speed. Any variation in the plot at different speed is indicative of the quality of decoupling in control loops. However, with this test setup, it can be seen that the bandwidth remains nearly the same regardless of running speed. This test can be repeated for Q loop as well, and at different bandwidth settings.

Comparison of current loops - classical vs FCL

The test setup enables the users to test the current loop by controlling it using conventional method as well for comparison purposes. Two sets of tests are performed, one with, and the other without, FCL. The bandwidth and phase margin results obtained at different bandwidth design settings are noted down for each set of tests, and the collected data are plotted as shown in Figure 39.

The group of plots at the bottom is obtained with conventional control (without using FCL) and the one at the top is obtained with FCL. It is self-revealing that, without FCL, the control bandwidth is too low and that with increasing control bandwidth, the phase margin is decreasing drastically. When FCL is used, the controller can provide a higher bandwidth at a higher phase margin and that the reduction in phase margin for increase in bandwidth is much less. The best performance is found to be when the bandwidth is about 1/6th of the sampling frequency where it almost behaves like deadbeat control.
For frequencies beyond that, overshoots may be noticed. In this particular evaluation, the PWM carrier is 10 kHz, the sampling frequency is 20 KHz and the best performance is obtained when the control bandwidth is 3.3 KHz.

**Summary**

This evaluation platform helps to control two different motors, either from one CPU core or two different CPU cores, and with or without FCL technology. When FCL is used, it shows an increase in control bandwidth. The SFRA tool showed the impact of FCL on control bandwidth.

Dynamic frequency response analysis in real-time on a motor drive system is unique among MCU suppliers and is currently capable only on C2000 MCUs. The presence of fast ADC, control law architecture (CLA) and trigonometric math unit (TMU) helps to reduce the latency between feedback sampling and PWM update resulting in higher control bandwidth and increase in maximum modulation index. Higher modulation index helps to improve DC bus utilization by the drive and to increase the control speed range of the motor.

Depending on the control speed range of motors in target applications, the MCU is possible to control multiple motors in multi-axes configurations using FCL based off the dual core F28379D MCU platform. This makes it suitable for high end servo control applications.

**Additional resources**

Here are additional resources regarding Fast Current Loop Control that you may find of interest:

- [C2000 DesignDRIVE Software for Industrial Drives and Motor Control](#)
- [DesignDRIVE On Line Training](#)
Position Manager

Simple interfacing to analog and digital position sensors for industrial drive control system

In many respects, system designers of industrial drive control systems, such as robotics and other applications involving servo and brushless motors, have to expend considerable time and effort developing, integrating and testing many of the control and connectivity building blocks—those “glue” elements—that go into their systems. This can cause many challenges such as lengthier development cycles, a larger board area or a higher bill of materials (BOM) cost. Due to this, these developers are unable to concentrate on differentiating features like enhanced performance, greater precision and improved control loops.

A particular example of this is the task of interfacing microcontrollers (MCUs) to position sensors. These sensors can be linear, angular or multi-axis and typically are used to sense the relative or absolute position of a mechanical system in motion, propelled by a motor. The sensed position is then converted to an analog or digital electrical signal for transmission to the controlling circuit.

Historically, interfacing a position sensor to an MCU could be a time-consuming task that often involved the integration of the communication protocol into a field programmable gate array (FPGA) or the programming of an additional MCU with the decode protocols. In addition, this situation is exacerbated by the fact that there are multiple encoder protocols available, each suited to certain types of functionality and subsystems. The system design team might be forced to develop several protocol specific FPGAs which would not scale effectively from one application to another. Of course, this type of FPGA implementation would add cost to the system by increasing the system’s electronic BOM, impacting the necessary board space and requiring lengthy development cycles. Moreover, developers also have to complete extensive compliance testing to certify conformance with industry standards.

This situation begs for a solution that would simplify the interfacing of position sensors to control elements in industrial drive systems and thereby free designers to concentrate on features and functionality that would make their systems truly distinctive, as well as more competitive, in the marketplace.

Integrating position feedback

Building on the C2000™ Delfino™ MCU portfolio, Texas Instruments provides a comprehensive platform for industrial drive and control systems. Starting with the processing capabilities required by sophisticated and precise control systems, the C2000 Delfino F28379D and F28379S MCUs are equipped with a full complement of on-chip resources, including DesignDRIVE Position Manager technology supporting today’s most popular off-the-shelf analog and digital position sensor interfaces. This relieves system designers from many of the more basic, repetitive tasks, saving design time.

TI has extensive expertise with interfacing position sensors to digital controllers. Beginning with standalone

![Figure 40: A closer look at the C2000™ dual-core F28379D MCU with DesignDRIVE position manager technology.](image-url)
interface solutions for resolver-to-digital solutions, such as the TMDSRSLVR, TI has continued to add to its position feedback interface support. Expensive resolver-to-digital chipsets have been replaced by C2000 MCU on-chip capabilities, leveraging high-performance analog-to-digital converters (ADCs) and digital-to-analog converters (DACs). Moreover, the powerful trigonometric math processing of C2000 MCUs is particularly well-suited to the additional processing needed to calculate the angle, and extract high-resolution speed information from a resolver’s amplitude modulated sinusoidal signals.

Many C2000 MCUs support enhanced quadrature encoder pulse (eQEP) modules that are capable of interfacing with linear or rotary incremental encoders. These encoders count pulses to obtain position (once an index is known), direction and speed information from rotating machines used in high-performance motion and position control systems. In addition, the eQEPs can be employed to interface to pulse train output (PTO) signals generally output by a programmable logic controller (PLC) in industrial automation for motion control. Also, eQEPs can interface to clockwise/counter clockwise (CW/CCW) signals. CW/CCW signals are typically used in conjunction with stepper or servo drives for controlling motors or other motion-based hardware. The C2000 F28379 MCUs support up to three eQEP modules.

Resolver and QEP capabilities provide fast, efficient and integrated solutions for effectively interfacing position sensors with C2000 Delfino MCUs. The next step has been to extend that support with complementary solutions that would allow the MCU to connect directly to more advanced digital and analog position sensors.

**DesignDRIVE Position Manager technology**

Available through TI’s DesignDRIVE platform, Position Manager technology takes advantage of the on-chip hardware resources of the C2000 Delfino F28379S and F28379D MCUs to interface to the most popular digital and analog position sensors. Already incorporating support for incremental encoders (eQEP), CW/CCW communications and standalone resolver solutions, Position Manager adds solutions for analog position sensing, integrating both resolver excitation and sensing, as well as a SinCos transducer manager. Unique to C2000 MCUs, Position Manager combines the analog sensor support with the popular digital absolute encoders, EnDat 2.2 and BiSS-C, giving system designers a wide range of position sensor types to choose from.

This integrated Position Manager technology offers system designers a real opportunity to accelerate development cycles and reduce BOM costs by eliminating the need for an FPGA to interface a specific encoder to the MCU or by drastically reducing the size of the FPGA that may still be needed for other functions. The illustration below demonstrates how Position Manager technology relieves system designers from the burden of developing the high- and low-level software drivers, as well as any custom hardware and logic that previously may have been implemented on an external FPGA. In addition, example closed-loop, position-sensor-based control projects downloaded from DesignDRIVE can be modified for integration into customer projects. The lower system layers are provided on-chip or through reference designs and a ready-to-use library of application programming interface (API) modules.

In addition to reducing development time, Position Manager technology also decreases the compliance and interoperability testing that system manufacturers have undertaken in the past. The Position Manager technology is fully tested across a variety of sensors. Please see the user’s guides for details on the testing results. Moreover, future revisions and updates to the applicable standards will also be supported by Position Manager technology.

Figure 41: DesignDRIVE position manager technology supports the leading analog and digital position sensor.
New position sensor interfacing capabilities

With its rich heritage of position feedback technologies as a starting point, TI has been able to expand its position sensor interface solutions with enhanced capabilities and performance. The following are several recent additions to TI's capabilities through DesignDRIVE Position Manager technology.

SinCos

SinCos is a feedback methodology which is incorporated into encoder interfaces like Hiperface® as well as other proprietary interfaces. These so called sinusoidal absolute encoders typically offer much higher position and speed resolutions than do resolver or incremental encoders. In conventional quadrature encoders, angle information is obtained by counting the edges of a pair of quadrature pulses. Angular resolution is fixed by the number of pulses per mechanical revolution. However, in SinCos transducers, precision of the angular measurement is increased by computing the angle between edges using the relationship between a pair of sine and cosine outputs from the sensor. Effectively, an interpolation between edges is made to obtain a “fine” angle. The fine angle is computed using an arctangent of the two sinusoidal inputs. For this computation to be valid, both inputs must be sampled simultaneously. Typically, several thousand electrical revolutions of the sinusoidal signals occur during each mechanical revolution of the encoder shaft.

The internal analog sub-system of the F28379 Delfino MCUs is ideal for interfacing to SinCos transducers. The presence of multiple ADCs, which can be triggered from the same source, allows simultaneous measurements of both input channels. In addition, the F28379 MCUs include a native ARCTAN instruction as part of the Trigonometry Math Unit (TMU) which means the angle calculation can be done in as little as 70 nanoseconds!

Another consideration is the high motor shaft speed state. In this case, there is no longer a need for precise angle information and the measurement algorithm only needs to count the number of complete sinusoidal revolutions to determine a “coarse” angle measurement.

Typically, this is done using a pair of analog comparators which compare the incoming sinusoids with a threshold representing the zero crossing point. The comparator outputs correspond to the sign of each sinusoid and the resulting digital signals are similar to those produced by a quadrature encoder. On the F28379 MCUs, there are up to eight pairs of analog comparators, each with its own programmable threshold voltage. These allow the quadrature pulses to be generated which are then fed internally to one of the on-chip quadrature encoder peripheral (QEP) modules for coarse angle and speed measurements.
EnDat

EnDat is a digital bi-directional four-wire interface developed by the German company, HEIDENHAIN. A sensor with an EnDat encoder can communicate position values, transmit and update information stored in the encoder, or save the information. Data is sent along with clock signals. The C2000 MCU can select the type of data the encoder will transmit, including position values, parameters, diagnostics and others.

Position Manager technology interfaces the C2000 F28379 MCU directly to the EnDat encoder (Figure 44). The only components external to the MCU are two RS-485 transceivers and the encoder power supply circuit. The EnDat Master is implemented using the C2000 MCU's configurable logic block, where the communication protocol is handled. Position Manager technology has been tested against a range of rotary, linear and multi-turn encoders from HEIDENHAIN and across resolutions from 13 bits to 35 bits at distances of 70 meters or more.

BiSS-C

The open source BiSS (bi-directional/serial/synchronous) digital interface is based on a real-time communications protocol. The original specification was developed by iC-Haus GmbH of Germany. BiSS-continuous mode (BiSS-C) is employed in industrial applications. The specification has its roots in the Synchronous Serial Interface (SSI). The BiSS-C interface consists of two uni-directional or bi-directional lines for the clock and data.

As with all interfaces supported by Position Manager technology, a BiSS-C master running on a C2000 F28379 MCU can connect directly to a BiSS-C encoder slave on a position sensor (Figure 45). The interface transmits position values and additional information directly from the encoder to the MCU. The MCU is able to read and write directly to the encoder's internal memory. TI's Position Manager technology includes a feature-rich BiSS-C library of capabilities, which system developers can readily draw on for their development projects. For example, clock frequencies of 8 MHz are supported on cables up to 100 meters long. In addition, the C2000 MCU BiSS interface can be adjusted to feature improved control of modular functions and timing by transmitting position information from encoders every control cycle.

Industrial drive control systems-on-chip

Powerful and programmable MCUs like TI's C2000 Defino F28379 MCUs represent the next step toward industrial drive control systems-on-chip (SoC). They empower more effective and efficient system architectures by eliminating the need for an external FPGA for ancillary processing requirements or by reducing the size of the FPGA significantly.
Now, TI has taken the next step to help industrial drives system developers deliver highly differentiated products including lower latencies, higher resolution and more powerful processing resources. That step involves simplifying the interfacing of MCUs to position sensors with Position Manager technology. By enabling a direct connection between a C2000 MCU and a position sensor, Position Manager technology frees developers from the more mundane tasks of device connectivity so they can focus on the features and capabilities that will make their system solutions truly distinctive in the marketplace with significant competitive advantages.

Additional resources

Here are additional resources regarding Position Manager that you may find of interest:

- C2000 Delfino MCUs F28379D LaunchPad Development Kit
- C2000 DesignDRIVE Position Manager BoosterPack
- C2000 Position Manager EnDat22 Library Module User’s Guide
- Using Position Manager EnDat22 Library on IDDK Hardware User’s Guide
- C2000 Position Manager BiSS-C Library User’s Guide
- Using Position Manager BiSS-C Library on IDDK Hardware User’s Guide
- C2000 Position Manager T-Format Library Module User’s Guide
- Using Position Manager T-Format Library on IDDK Hardware and Software User’s Guide
- C2000 Position Manager SinCos Library User’s Guide
- Using Position Manager SinCos Library on IDDK User’s Guide
- Reduce System Costs with Resolver-to-Digital Conversion Implementation on C2000 Microcontrollers
- CW/CCW Support on the C2000 eQEP Module
- C2000 Position Manager PTO Library Application Report
- A Cost-Effective Option to Get Started with Integrated Position Sensing
- EnDat 2.2 Absolute Encoder Master Interface Reference Design for C2000 MCUs
- Tamagawa T-Format Absolute-Encoder Master Interface Reference Design for C2000 MCUs
- BiSS-C Absolute Encoder, Master-Interface Reference Design for C2000 MCUs
- Designing an EMC-Compliant Interface to Motor Position Encoders – Part 1 (Overview)
- Designing an EMC-Compliant Interface to Motor Position Encoders – Part 2 (BiSS)
- Designing an EMC-Compliant Interface to Motor Position Encoders – Part 3 (EnDat)
- Designing an EMC-Compliant Interface to Motor Position Encoders – Part 4 (HIPERFACE DSL)
- Designing an EMC-Compliant Interface to Motor Position Encoders – Part 5 (Sin/cos)
- Designing an EMC-Compliant Interface to Motor Position Encoders – Part 6 (Summary)
- DesignDRIVE On Line Training
Communications

EtherCAT and C2000 MCUs – real-time communications meets real-time control

With the release of the EtherCAT Slave and C2000 Delfino™ MCU controlCARD kit and the EtherCAT solution reference software in controlSUITE, you can jump-start the development of an EtherCAT slave stack directly on C2000 MCUs.

In the following blog series, I will discuss the market opportunity for EtherCAT slave stack support on C2000 MCUs; what’s different about the TI implementation; and then a deeper dive into the supported development phases: evaluate, validate and create.

The redeeming qualities of EtherCAT for industrial applications are well-known. The worldwide adoption of EtherCAT, especially in multi-axis servo-based machines, has been quite remarkable, at least from TI’s C2000 market perspective. With membership in the EtherCAT Technology Group (ETG) reaching over 4,300 and an ever-growing and balanced mix of worldwide members not dominated by Europe, it is pretty clear that the overall EtherCAT solution, technology, availability, licensing, support, etc., resonates with industrial customers.

We have seen many customers take on the task of adapting C2000 MCUs to run the EtherCAT stack natively, especially in motion-control applications. The integration of real-time communications directly into the environment of real-time control development simplifies the development process. It should also be obvious that the integration onto a single central processing unit (CPU) would ease the synchronization between the network interrupts and the real-time control-loop timing.

The C2000 MCU is perfectly capable of executing a software stack and performing demanding real-time motor control simultaneously, especially since EtherCAT hardware handles the most challenging real-time elements of EtherCAT communications. Compared to adding a secondary CPU to run the stack, a single CPU has obvious benefits in cost and board space reduction. System performance also benefits from the reduced transport latency that would otherwise be introduced if you used an additional CPU to handle the stack. In motion-control applications, the time or latency of the system to respond to new target positions is critical. Why spend all that effort minimizing the position control loop time if the position command input timing cannot keep up with the control loop capability? That interim CPU running your EtherCAT stack could be impacting your overall motion performance.

Given the complementary aspects of EtherCAT and C2000 MCU motion control and because of customer support requests, TI applied its embedded software development experience and C2000 MCU architecture knowledge to create a set of software solutions designed to aid in the development of EtherCAT slave nodes on TI real-time control MCUs. With very little development investment, the software package will help you:

- Evaluate the stack for a slave node
- Validate the correct operation/hookup of your custom hardware
- Create a complete slave-node application using your hardware and a stack distributed by the ETG

Figure 46 shows three phases of the development of your own slave node application. It also outlines the hardware assumptions and describes software modules available to support each phase. We will spend more time breaking down the support in each phase in subsequent posts. The next post in this series, however, will describe some of the specific features enabled and actions taken to make slave stack development on C2000 MCUs more compelling for real time control applications than you may expect.

Beyond this three-faceted development flow, what has TI done to make our solution more compelling than a typical stack porting exercise? First, we applied our knowledge
of the C28 CPU architecture and tuned the software to leverage the CPU services accordingly. For example, we optimized the interrupt handling, direct memory access (DMA) and control (pulse-width modulation [PWM]) synchronization routines to make the best use of the on-chip hardware. In addition, we worked with Beckhoff to update/enhance the released stacks to support data handling so that it's more tailored for the C28 CPU.

Speaking of working with Beckhoff, the C28 slave stack and hardware abstraction layer (HAL) for both Serial Peripheral Interface (SPI) and parallel (external memory interface [EMIF]) communications are supported as part of their Slave Stack Configuration tool (SSC). Based on previous C2000 MCU experience, you will find the reference software solution release “Plugfest-ready.”

Supporting both the SPI and EMIF hardware interfaces provides implementation options, enabling system trade-offs on printed circuit board (PCB) routing complexity vs. the lowest latency communications. In addition, while we tested the release directly with the Beckhoff ET1100 device, you will be able to adapt the SPI and EMIF HAL drivers to support other EtherCAT slave silicon solutions, including TI's AMIC110 SoC (SPI – please contact the Sitara™ forum) or field-programmable gate array (FPGA) instances (EMIF or SPI) of the slave hardware such as the Beckhoff ET1816.

Among SPI and EMIF processor data interfaces (PDIs), there is no difference between the EtherCAT slave stack code and application code. Only the device name and product code differ, so both SPI and EMIF slave nodes can be different even when they are both in the same network. EMIF slave nodes can be different when they are both in the same network.

A C2000 MCU-specific feature of TI's EtherCAT software package is the Echoback application example. While the application demonstration is not complex, it simply loops slave-node output data structures back to inputs, viewable using any EtherCAT master or the TwinCAT master as detailed in the “EtherCAT Solution Reference Guide.”

By including the Echoback application, TI shows an example of the basic use of the stack software and provides a placeholder for you to create your own applications. Following the Echoback example will make it easier to take advantage of the many optimizations TI has made in the stack and HAL software modules and gives a good example of the EtherCAT slave information (ESI) file as well. Echoback is also included as part of the quick evaluation binary project.

Finally, there is the compelling EtherCAT controlCARD hardware platform itself, Figure 47, that leverages the common 180-pin interface of other C2000 MCU control cards. The EtherCAT controlCARD is physically
compatible with any 180-pin C2000 application evaluation module (EVM) or dock, thus making it possible to add EtherCAT connectivity to many existing real-time control application examples. For example, you can use the EtherCAT controlCARD in conjunction with the industrial drives development kit (IDDK) to add real-time connectivity to a servo using fast current loop and/or position manager technology.

Another nice feature is that the EtherCAT controlCARD can operate stand-alone. Powering the hardware with a Micro USB cable eliminates the need for a dock or EVM to supply power. See the “TMDSECATCNCD379D EtherCAT Solution Reference Guide” for details. This can help reduce hardware complexity in multi-node network testing and configuration.

As you can see, the C2000 DesignDRIVE EtherCAT support has taken several additional steps to ease your development of your own slave node and help you get great performance running the stack on our microcontrollers.

I’ll break down the three phases of the development of your own slave-node application and how the TI EtherCAT package in C2000 controlSUITE™ software supports each phase.

Given the previously outlined complementary aspects of EtherCAT technology and C2000 MCUs for motion control applications – and because of customer requests – TI applied its embedded software development experience and C2000 MCU architecture knowledge to create a set of software solutions designed to aid in the development of EtherCAT slave nodes on C2000 real-time control MCUs. The software package will help you:

- Evaluate the stack for a slave node
- Validate the correct operation/hookup of your custom hardware
- Create a complete slave-node application using your hardware and a stack distributed by the EtherCAT Technology Group

Figure 47: TMDSECATCNCD379 EtherCAT board image.

Step 1: Evaluate the C2000 EtherCAT node (stack and hardware)

The first software solution included in the C2000 EtherCAT support is meant to make it easy to quickly get familiar with the basic solution. Leveraging the TwinCAT PC software for the EtherCAT master node, the simple “echoback” example application and the EtherCAT Slave and C2000 Delfino MCU controlCARD kit, you can build a multinode EtherCAT network on your bench in minutes.

The EtherCAT “slave demo” project is a binary image release for the EtherCAT controlCARD, optimized for the C28x CPU, that includes the slave stack binary version. So all you need to do is program the C2000 MCU and connect a CAT5 cable to the EtherCAT master – a TwinCAT PC/PLC in this case.

This evaluation project will help you answer questions like: What is the transfer latency from the external EtherCAT controller (ET1100) to the C2000 MCU? How much central processing unit (CPU) bandwidth does this solution take? What is the memory footprint of the hardware abstraction layer (HAL) and the slave stack?

After downloading the latest controlSUITE software, you can install the demonstration and evaluation example by executing the setup file, “EtherCAT_Slave_Demo_Code_v01_00_00_00_setup,” in the controlSUITE development kit project.
Step 2: Validate your EtherCAT hardware

The second software solution shifts from being based on the TI controlCARD tool to helping you develop your own hardware. The EtherCAT community refers to the interface to the programmable controller as the processor data interface (PDI). Recall that you can find reference hardware designs for both the parallel PDI, “ASYNC16” (C2000 MCU external memory interface [EMIF]) and serial PDI (C2000 MCU serial peripheral interface [SPI]) in the EtherCAT Interface for High Performance MCU Reference Design. This software project will help you verify that hardware interfaces you have built “from the MCU to the wire” are correct.

Since the hardware abstraction layer (HAL) test application (Figure 48) can communicate with a TwinCAT master node, the software project helps you validate that the physical PDI interfaces (EMIF and SPI port) to the EtherCAT controller (such as the ET1100) – as well as the physical layer (PHY) connections and Ethernet magnetics – are all connected properly.

You do not need an EtherCAT slave stack to validate the hardware connection in this case. Unlike step 1, the HAL test project is available as source code to aid in validation or to adapt C2000 MCUs to other physical interfaces for EtherCAT controllers.

Step 3: Create your own EtherCAT slave-node solution

Finally, the EtherCAT application reference project provides a step-by-step example when it’s time to integrate your validated hardware with the EtherCAT stack. This project contains the source code necessary to build an example EtherCAT slave node (initialization, HAL, echoback application) except for the EtherCAT slave stack itself. You must obtain the slave stack through the EtherCAT Technology Group; it is distributed as part of the slave stack configuration (SSC) tool.
In the event that your version of the configuration tool does not include support for C2000 MCUs, the controlSUITE project includes a patch that will make C2000 MCUs (and the ASYNC16 and SPI PDI options) available to you in the drop-down processor selection dialog box. See the “EtherCAT Solution Reference Guide” for more information on how to use the SSC patch code.

Two project configurations are available:

- A random access memory (RAM)-based project, which is usually helpful during the initial stages of a project when code changes are frequent (less time spent programming flash).

- A flash-based project, when the code is more mature and ready for testing through power-cycling of the target C2000 MCU.

With the release of the EtherCAT Slave and C2000 Delfino™ MCU controlCARD kit and EtherCAT solution reference in controlSUITE, software you can jump-start the development of an EtherCAT slave stack running directly on C2000 MCUs.

**Additional Resources**

Here are additional resources regarding Communications that you may find of interest:

- [EtherCAT Slave and C2000 Delfino MCU controlCARD Kit for Industrial Factory Automation Applications](#)
- [DesignDRIVE On Line Training](#)
Product Overviews

The TMS320F2837xD architecture: achieving a new level of high performance

Abstract
With the utilization of advanced high-performance microcontroller architectures, sophisticated real-time control systems can be realized. By combining both analog and digital control peripherals into a single device, along with a dual-core design, these systems can be cost-effectively implemented. This document provides an introduction and general overview to the TMS320F2837xD device architectural features. Even though the topics presented in this document are based on the TMS320F2837xD dual-core device family, most all of the topics are fully applicable to the TMS320F2837xS and TMS320F2807x single-core device families.

Introduction
Many control systems utilize powerful high performance microcontrollers in order to meet real-time design requirements. In addition to the microcontroller, these systems require various analog components to sense signals for the feedback control loops, as well as for hardware protection. Some advanced control systems may even require two microcontrollers: for example one can be used to track speed and position, while the other can be used to control torque and current loops. A system implementation such as this might be intelligently partitioned; however, communications between the two microcontrollers can dramatically increase the complexity of the system design. Ideally, combining the two microcontrollers with a common communication link between them, along with the necessary analog components into a single device would increase system performance and reliability, and at the same time simplify the board layout and reduce the overall system cost. To address these types of high performance applications, the C2000™ family of microcontrollers from Texas Instruments developed a special class of dual-core devices.

Figure 51: F2837xD functional block diagram.
The TMS320F2837xD microcontroller (MCU) family, referred to as the F2837xD in this document, is a dual-core MCU design based on the TI 32-bit C28x CPU architecture. Each core is identical with access to its own local RAM and Flash memory, as well as globally shared RAM memory. Sharing information between the two CPU cores is accomplished with an Inter-Processor Communications (IPC) module. Additionally, each core shares access to a common set of highly integrated analog and control peripherals, providing a complete solution for demanding real-time high-performance signal processing applications, such as digital power, industrial drives, inverters, and motor control.

In addition to the high-performance CPUs, each core has a Control Law Accelerator (CLA), which is an independent 32-bit floating-point processor designed to execute math intensive calculations. The CLA runs concurrently and at the same speed of the main CPU, thereby effectively doubling the computational performance of each core. With each CPU running at 200 MHz, the CLAs can then effectively boost the total performance of the device to 800 MIPS.

In-depth details about this device can be found in the technical documentation. Although this document is based on the F2837xD dual-core device family, most all topics are fully applicable to the F2837xS singlecore device family and the F2807x device family which has been designed for cost-sensitive applications. For reference, a device matrix is included in Section 15 to provide a general comparison between the three device families.

The C28x Core Processor

Each of the C28x CPU cores is designed around a 32-bit fixed-point accumulator-based architecture. It incorporates the best features of digital signal processors and microcontroller architectures, providing excellent 32-bit processing capabilities. From a pure architectural point of view, it utilizes a modified Harvard architecture for flexibility and speed, which enables instructions and data reads to be performed in parallel, along with simultaneous data writes. This is accomplished using six separate address/data bus structures which enable full speed processor execution, and with the 8-stage pipeline most operations can be performed in a single cycle.

The addition of the Floating-Point Unit (FPU) to the C28x fixed-point CPU core enables support for hardware IEEE-754 single-precision floating-point format operations. The FPU adds an extended set of floating-point registers and instructions to the standard C28x architecture, providing seamless integration of floating-point hardware into the CPU. In the pipeline decode stage, the instruction is decoded to determine if it is a standard C28x instruction or a FPU instruction, and is routed accordingly. Since the FPU instructions are extensions of the standard C28x instruction set, most instructions operate in one or two pipeline cycles and some can be done in parallel.

The Trigonometric Math Unit (TMU) is an extension of the FPU and the C28x instruction set, and it efficiently executes trigonometric and arithmetic operations commonly found in control system applications. Similar to the FPU, the TMU provides hardware support for IEEE-754 single-precision floating-point operations that are specifically focused on trigonometric math functions. Seamless code integration is accomplished by built-in compiler support that automatically generates TMU instructions where applicable. This dramatically increases the performance of trigonometric functions, which would otherwise be very cycle intensive. It uses the same pipeline, memory bus architecture, and FPU registers as the FPU, thereby removing any special requirements for interrupt context save or restore.

The Viterbi, Complex Math, and CRC Unit (VCU) adds an extended set of registers and instructions to the standard C28x architecture for supporting various communications-based algorithms, such as power line communications (PLC) standards PRIME and G3. These algorithms typically require Viterbi decoding, complex Fast Fourier Transform (FFT), complex filters, and cyclical redundancy check (CRC). By utilizing the VCU a significant performance benefit is realized over a software implementation. It performs fixed point operations using the existing instruction set format,
pipeline, and memory bus architecture. Additionally, the VCU is very useful for general-purpose signal processing applications such as filtering and spectral analysis.

**Memory**

The F2837xD MCU utilizes a memory map where the unified memory blocks can be accessed in either program space, data space, or both spaces. This type of memory map lends itself well for supporting high level programming languages. Each of the CPU subsystems has a memory structure consisting of dedicated RAM blocks, shared local RAM blocks, shared global RAM blocks, message RAM blocks, Flash, and one-time programmable (OTP) memory.

Each CPU subsystem has four dedicated RAM blocks named M0, M1, D0, and D1. The M0 and M1 blocks have a size of 1Kx16 words each, and the D0 and D1 blocks have a size of 2Kx16 words each. These memory blocks are tightly coupled with the CPU, and only the CPU has access to them. All four of these blocks have error-correcting code (ECC) protection and memory blocks D0 and D1 can be secured.

![Simplified memory map](image-url)

*Figure 52: Simplified memory map.*
Each CPU subsystem has six local shared RAM blocks named LS0 through LS5, which are only accessible by its CPU and CLA. The LSx RAM blocks have a size of 2Kx16 words each, have parity protection, and can be secured. By default, the LSx RAM blocks are dedicated to the CPU only; however, each block can be shared between the CPU and CLA by configuring the appropriate bit in the LSx Memory Selection register. When shared between the CPU and CLA, each LSx RAM block can then be configured as either CLA program memory or CLA data memory.

The global shared RAM blocks are accessible from both CPU subsystems and their respective direct memory access (DMA) modules. There are sixteen global shared RAM blocks named GS0 through GS15, and each block has a size of 4Kx16 words each. Each GSx RAM block can be owned by either CPU subsystem, which is determined by the appropriate bit setting in the GSx Master Selection register. When CPU1 subsystem owns a GSx RAM block it has full fetch/read/write access to that block and CPU2 subsystem only has read access. Likewise, when CPU2 subsystem owns a GSx RAM block it has full fetch/read/write access to that block and CPU1 subsystem only has read access.

There are two types of message RAM blocks: CPU message RAM blocks and CLA message RAM blocks. The CPU message RAM blocks are used to share data between CPU1 subsystem and CPU2 subsystem. There is a dedicated CPU message RAM block for “CPU1 to CPU2” and another dedicated CPU message RAM block for “CPU2 to CPU1”. The CPU message RAM blocks have a size of 1Kx16 words each. The CPU message RAM blocks have CPU and DMA read/write access from its own CPU subsystem, and CPU and DMA read-only access from the other CPU subsystem. Since these CPU message RAM blocks are used for inter-processor communications they are also known as IPC RAM blocks.

The CLA message RAM blocks are used to share data between the CPU and CLA. There is a dedicated CLA message RAM block for “CLA to CPU” and another dedicated CLA message RAM block for “CPU to CLA”. The CLA message RAM blocks have a size of 1Kx16 words each, have parity protection, and can be secured. The CLA has write access to the “CLA to CPU” message RAM block, and the CPU has write access to the “CPU to CLA” message RAM block. The CPU and CLA have read access to both CLA message RAM blocks.

Each CPU subsystem has its own flash bank of memory, which is primarily used to store program code, but can also be used to store static data. Each flash bank size can be up to 256Kx16 words, for a maximum total of 512Kx16 words of flash on a device. The flash sectors have ECC protection providing single-error correction and double-error detection (SECDED), and each sector can be secured. Additionally, a code pre-fetch mechanism and data cache is used to achieve optimum system performance.

Each CPU subsystem has two 1Kx16 words of OTP memory blocks: TI-OTP and USER-OTP. The TI-OTP contains device-specific calibration data for the ADC, internal oscillators, and buffered DACs, in addition to settingsused by the flash state machine for erase and program operations. The USER-OTP contains locations for programming security settings, such as passwords for selectively securing memory blocks, configuring the standalone boot process, as well as selecting the boot-mode pins in case the factory default pins cannot be used. This information is programmed into the dual code security module (DCSM).

The DCSM offers protection for two zones (zone-1 and zone-2), and is used to block access and visibility to the various on-chip memory resources with the purpose of preventing duplication and reverse engineering of proprietary code. Note that each CPU subsystem has its own DCSM, and each DCSM has code protection for two zones. The security options for both zones are identical, and each memory resource can be assigned to either zone or not secured. Either zone can secure each sector of flash individually, each LSx and Dx memory block individually, and the CLA message RAM blocks. The term “secure” implies that all data read/
write accesses to the resource are blocked by any means including an emulator. Data reads and writes from secured memory are only allowed for code running from secured memory. If a resource is “unsecure”, then access is allowed by any means.

Each zone is secured by its own 128-bit (four 32-bit words) user defined CSM password, which is stored in its dedicated OTP location based on a zone-specific link pointer. The user accessible CSM Key Register (CSMKEY) is used to secure and unsecure the device, and a new or un-programmed device has all zone password bit fields set to 1’s by default, thereby unlocking the device. Each zone’s dedicated OTP block contains its security configuration settings, such as the CSM passwords, and the allocations for securing the RAM blocks and flash sectors. Since the OTP cannot be erased, flexibility is provided by using a link pointer to select the location of the active zone region within the OTP block, allowing the user to make multiple modifications to the configuration up to thirty times. This is accomplished by exploiting the fact that each bit in the OTP can be programmed one bit at a time, and a “1” can be programmed to a “0”, but not erased back to a “1”. The most significant bit position in the link pointer that is programmed to a “0” defines the valid offset base address for the active zone region within the OTP block.

Reset and Clocks

Upon reset and by default, the CPU1 subsystem is the master and it owns the device configuration and control. Then via software running on CPU1, the peripherals and input/output pins can be configured to be accessible by the CPU2 subsystem. Once configured, a series of “lock” registers can be used to protect several system configuration settings from spurious CPU writes. After the lock registers bits are set, the respective locked registers can no longer be modified by software.

Each CPU subsystem has its own NMI (non-maskable interrupt) module to handle different exceptions during run time, as well as its own watchdog timer module for software use. The device has various reset sources, but in general resets on CPU1 will reset the entire device and resets on CPU2 will reset only the CPU2 subsystem. A Reset Cause Register (RESC) is available for each CPU subsystem which can be read to determine the cause of the reset. The external reset pin is the main chip-level reset for the device, and it resets both CPU subsystems to their default state. The power-on reset (POR) circuit is used to create a clean reset throughout the device during power-up, while suppressing glitches on the input/output pins.

By default, the CPU1 subsystem owns the PLL clock configuration, however a clock control semaphore is available for CPU2 to access the clock configuration registers. All of the clock signals in the device are derived from one of four clock sources: Internal Oscillator 1 (INTOSC1), Internal Oscillator 2 (INTOSC2), External Oscillator (XTAL), and Auxiliary Clock Input (AUXCLKin). At power-up, the device is clocked from the on-chip 10 MHz oscillator INTOSC2. INTSOC2 is the primary internal clock source, and is the default system clock at reset. The device also includes a redundant on-chip 10 MHz oscillator INTOSC1. INTOSC1 is a backup clock source, which normally only clocks the watchdog timers and missing clock detection circuit. Additionally, the device includes dedicated X1 and X2 pins for supporting an external clock source such as an external oscillator, crystal, or resonator. The AUXCLKin is used as the bit clock source for the USB and CAN to generate the precise frequency requirements. These four clock sources can be multiplied using the PLL and divided down to produce the desired clock frequencies for a specific application.

Boot Modes

During the F2837xD dual-core MCU booting, CPU1 controls the boot process and starts execution from the CPU1 boot ROM while CPU2 is held in reset. CPU2 goes through its own boot process under the control of CPU1, except when CPU2 is set to boot-to-flash. The IPC registers are used to communicate between CPU1 and CPU2 during the boot process. Additionally, the boot ROM contains the necessary boot loading routines to support peripheral boot loading.
When the device is reset, the peripheral interrupt expansion block, also known as the PIE block, and the master interrupt switch INTM are disabled. This prevents any interrupts during the boot process. The program counter is set to 0x3FFFC0, where the reset vector is fetched. In the boot code the JTAG Test Reset line, or TRST line, is checked to determine if the emulator is connected. If the emulator is connected, then the boot process follows the Emulation Boot mode flow. In Emulation Boot mode, the boot is determined by the EMU_BOOTCTRL register located in the PIE RAM.

Specific details about the boot flow are then determined by the EMU_KEY and EMU_BMODE bit fields in the EMU_BOOTCTRL register. If the emulator is not connected, the boot process follows the Stand-alone Boot mode flow. In Stand-alone Boot mode, the boot is determined by two GPIO pins and the Z1-BOOTCTRL and Z2-BOOTCTRL registers located in the OTP. Specific details about the boot flow are then determined by the OTP_KEY and OTP_BMODE bit fields in the Z1-BOOTCTRL and Z2-BOOTCTRL registers.

![Figure 53: Emulation boot mode.](image-url)
In Emulation Boot mode, first the EMU_KEY bit fields are checked for a value of $0x5A$. If either EMU_KEY or EMU_BMODE bit fields are invalid, the “Wait” boot mode is entered. These bit field values can then be modified using the debugger and then a reset is issued to restart the boot process. This is the typical sequence followed during device power-up with the emulator connected, allowing the user to control the boot process using the debugger.

Once the EMU_KEY bit fields are set to $0x5A$, then the EMU_BMODE bit field values determine the boot mode. The various Emulation Boot modes supported are Parallel I/O, SCI, SPI, I²C, CAN, M0 RAM, FLASH, USB, and Wait. The GetMode and when EMU_BMODE bit fields have a value of $0xFE$ or $0xFF$ are used to emulate the Stand-alone Boot mode.

In Stand-alone boot mode, first GPIO pins 72 and 84 are checked to determine if the boot mode is Parallel I/O, SCI, Wait, or GetMode. These pin can be remapped to any GPIO pins, if needed, and the default “unconnected” pins set the boot mode to GetMode. In GetMode the OTP_KEY bit fields in the Z1BOOTCTRL and Z2-BOOTCTRL registers are checked for a value of $0x5A$. An un-programmed device will have these locations set as 1’s, and the flash boot mode is entered, as expected for the default mode. If the OTP_KEY bit fields in either Z1-BOOTCTRL or Z2-BOOTCTRL registers has a value of $0x5A$, then the OTP_BMODE bit field values in the registers determines the boot mode. The various Stand-alone Boot modes supported are Parallel I/O, SCI, SPI, I²C, CAN, M0 RAM, FLASH, USB, and Wait.

**Interrupt structure**

The C28x CPU core has a total of fourteen interrupt lines, of which two interrupt lines are directly connected to CPU Timers 1 and 2 (on INT13 and INT14, respectively) and the remaining twelve interrupt lines (INT1 through INT12) are used to service the peripheral interrupts. A Peripheral Interrupt Expansion (PIE) module multiplexes up to sixteen peripheral interrupts into each of the twelve CPU interrupt lines, further expanding...
support for up to 192 peripheral interrupt signals. The PIE module also expands the interrupt vector table, allowing each unique interrupt signal to have its own interrupt service routine (ISR), permitting the CPU to support a large number of peripherals.

By using a series of flag and enable registers, the CPU can be configured to service one interrupt while others remain pending, or perhaps disabled when servicing certain critical tasks. The PIE module has an individual flag and enable bit for each peripheral interrupt signal. Each of the sixteen peripheral interrupt signals that are multiplexed into a single CPU interrupt line is referred to as a “group”, so the PIE module consists of 12 groups. Each PIE group has a 16-bit flag register (PIEIFRx), a 16-bit enable register (PIEIERx), and a bit field in the PIE acknowledge register (PIEACK) which acts as a common interrupt mask for the entire group. Likewise, internal to the CPU there is a bit field in a Interrupt Flag Register (IFR) and a bit field in an Interrupt Enable Register (IER) for each of the interrupt lines. Also, internal to the CPU is a global interrupt mask (INTM) bit located in the status register. For a peripheral interrupt to propagate to the CPU, the appropriate PIEIFR must be set, the PIEIER enabled, the CPU IFR set, the IER enabled, and the INTM enabled. Note that some peripherals can have multiple events trigger the same interrupt signal, and the cause of the interrupt can be determined by reading the peripheral’s status register.

During processor initialization, the interrupt vector table is copied to the PIE RAM and then the PIE module is enabled. When the CPU receives an interrupt, the vector address of the ISR is fetched from the PIE RAM, and the interrupt with the highest priority that is both flagged and enabled is executed. Priority is determined by the location within the interrupt vector table. The lowest numbered interrupt has the highest priority when multiple interrupts are pending.

Each C28x CPU core in the F2837xD device has its own PIE module, and each PIE module is configured independently. Some interrupt signals are sourced from shared peripherals that can be owned by either CPU, and these interrupt signals are sent to both CPU PIE modules regardless of which CPU owns the peripheral. Therefore, if enabled a peripheral owned by one CPU can cause an interrupt on the other CPU.

**General-Purpose Input/Output (GPIO) structure**

The F2837xD device incorporates a multiplexing scheme to enable each I/O pin to be configured as a GPIO pin or one of several peripheral I/O signals. Sharing a pin across multiple functions maximizes application flexibility while minimizing package size and cost. A GPIO Group multiplexer and four GPIO Index multiplexers provide a double layer of multiplexing to allow up to twelve independent peripheral signals and a digital I/O function to share a single pin. Each output pin can be controlled...
by either a peripheral or CPU1, CPU1 CLA, CPU2, or CPU2 CLA. However, the peripheral multiplexing and pin assignment can only be configured by CPU1. By default, all of the pins are configured as GPIO, and when configured as a signal input pin, a qualification sampling period can be specified to remove unwanted noise. Optionally, each pin has an internal pullup resistor that can be enabled in order to keep the input pin in a known state when no external signal is driving the pin. The I/O pins are grouped into six ports, and each port has 32 pins except for the sixth port which has nine pins (that is, the remaining I/O pins). For a GPIO, each port has a series of registers that are used to control the value on the pins, and within these registers each bit corresponds to one GPIO pin.

If the pin is configured as GPIO, a direction register (DIR) is used to specify the pin as either an input or output. By default, all GPIO pins are inputs. The current state of a GPIO pin corresponds to a bit value in a data register (DAT), regardless if the pin is configured as GPIO or a peripheral function. Writing to the DAT register bit field clears or sets the corresponding output latch, and if the pin is configured as an output the pin will be driven either low or high. The state of various GPIO output pins on the same port can be easily modified using the SET, CLEAR, and TOGGLE registers. The advantage of using these registers is a single instruction can be used to modify only the pins specified without disturbing the other pins. This also eliminates any timing issues that may occur when writing directly to the data registers.

**Crossbars (X-BAR)**

The X-BARs provide a flexible means for interconnecting multiple inputs, outputs, and internal resources in various configurations. The F2837xD contains three X-BARs: the Input X-BAR, the Output X-BAR, and the ePWM X-BAR.

The Input X-BAR is used to route external GPIO signals into the device. It has access to every GPIO pin, where each signal can be routed to any or multiple destinations which include the ADCs, eCAPs, ePWMs, Output X-BAR, and external interrupts. This provides additional flexibility above the multiplexing scheme used by the

---

*Figure 56: General-Purpose Input/Output (GPIO) pin block diagram.*
Figure 57: Input X-BAR.

Figure 58: Output X-BAR.
GPIO structure. Since the GPIO does not affect the Input X-BAR, it is possible to route the output of one peripheral to another, such as measuring the output of an ePWM with an eCAP for frequency testing.

The Output X-BAR is used to route various internal signals out of the device. It contains eight outputs that are routed to the GPIO structure, where each output has one or multiple assigned pin positions, which are labeled as OUTPUTXBARx. Additionally, the Output X-BAR can select a single signal or logically OR up to 32 signals.

The ePWM X-BAR is used to route signals to the ePWM Digital Compare submodules of each ePWM module for actions such as trip zones and synchronizing. It contains eight outputs that are routed as TRIPx signals to each ePWM module. Likewise, the ePWM X-Bar can select a single signal or logically OR up to 32 signals.

Analog subsystem

Analog components are a critical element in many control systems. The F2837xD analog peripherals include four analog-to-digital converters, eight comparator subsystems, and three buffered digital-to-analog converters.

Analog-to-Digital Converter (ADC)

The F2837xD includes four independent high-performance ADC modules which can be accessed by both CPU subsystems, allowing the device to efficiently manage multiple analog signals for enhanced overall system throughput. Each ADC module has a single sample-and-hold (S/H) circuit and using multiple ADC modules enables simultaneous sampling or independent operation. The ADC module is implemented using a successive approximation (SAR) type ADC with a configurable resolution of either 16-bits or 12-bits. For 16-bit resolution, the ADC performs differential signal conversions with a performance of 1.1 MSPS, yielding 4.4 MSPS for the device. In differential signal mode, a pair of pins (positive input ADCINxP and negative input ADCINxN) is sampled and the input applied to the converter is the difference between the two pins (ADCINxP – ADCINxN). A benefit of differential signaling mode is the ability to cancel noise that may be introduced common to both inputs. For 12-bit resolution, the ADC performs single-ended signal conversions with a performance of 3.5 MSPS, yielding 14 MSPS for the device. In single-ended mode, a single pin (ADCINx) is sampled and applied to the input of the converter.

The ADC triggering and conversion sequencing is managed by a series of Start-of-Conversion (SOCx) configuration registers. Each SOCx register configures a single channel conversion, where the SOCx register specifies the trigger source that starts the conversion, the channel to convert, and the acquisition sample window duration. Multiple SOCx registers can be configured for the same trigger, channel, and/or acquisition window. Configuring multiple SOCx registers to use the same trigger will cause that trigger to perform a sequence of conversions, and configuring multiple SOCx registers for the same trigger and channel can be used to oversample the signal.

The various trigger sources that can be used to start an ADC conversion include the General-Purpose Timers from each CPU subsystem, the ePWM modules, an external pin, and by software. Also, the flag setting of either ADCINT1 or ADCINT2 can be configured as a trigger source which can be used for continuous conversion operation.
When multiple triggers are received at the same time, the ADC conversion priority determines the order in which they are converted. Three different priority modes are supported. The default priority mode is round robin, where no SOCx has an inherently higher priority over another, and the priority depends upon a round robin pointer. The round robin pointer operates in a circular fashion, constantly wrapping around to the beginning. In high priority mode, one or more than one SOCx is assigned as high priority. The high priority SOCx can then interrupt the round robin wheel, and after it has been converted the wheel will continue where it was interrupted. High priority mode is assigned first to SOC0 and then in increasing numerical order. If two high priority SOCx triggers occur at the same time, the lower number will take precedence. Burst mode allows a single trigger to convert one or more than one SOCx sequentially at a time. This mode uses a separate Burst Control register to select the burst size and trigger source.

After each SOCx channel conversion is completed, an end-of-conversion (EOC) signal can be used to trigger an ADC interrupt. Each ADC module has four configurable ADC interrupts (ADCINT1-4) which can be triggered by any of the EOC signals. The ADC can be configured to generate the EOC signal either at the beginning of the conversion or one cycle prior to the conversion being written into the result register. Generating the EOC signal at the beginning of the conversion provides “just-in-time” reading of the ADC results, which reduces the sample to output delay and enables faster system response.

To further enhance the capabilities of the ADC, each ADC module incorporates four post-processing blocks (PPB), and each PPB can be linked to any of the ADC result registers. The PPBs can be used for offset correction, calculating an error from a set-point, detecting a limit and zero-crossing, and capturing a trigger-to-sample delay. Offset correction can simultaneously remove an offset associated with an ADCIN channel that was possibly caused by external sensors or signal sources with zero-overhead, thereby saving processor cycles. Error calculation can automatically subtract out a computed error from a set-point or expected result register value, reducing the sample to output latency and software overhead. Limit
and zero-crossing detection automatically performs a check against a high/low limit or zero-crossing and can generate a trip to the ePWM and/or generate an interrupt. This lowers the sample to ePWM latency and reduces software overhead. Also, it can trip the ePWM based on an out-of-range ADC conversion without any CPU intervention which is useful for safety conscious applications. Sample delay capture records the delay between when the SOCx is triggered and when it begins to be sampled. This can enable software techniques to be used for reducing the delay error.

**Comparator Subsystem (CMPSS)**

The F2837xD includes eight independent Comparator Subsystem (CMPSS) modules that are useful for supporting applications such as peak current mode control, switched-mode power, power factor correction, and voltage trip monitoring. Each CMPSS module is designed around a pair of analog comparators which generates a digital output indicating if the voltage on the positive input is greater than the voltage on the negative input. The positive input to the comparator is always driven from an external pin. The negative input can be driven by either an external pin or an internal programmable 12-bit digital-to-analog (DAC) as a reference voltage. Values written to the DAC can take effect immediately or be synchronized with ePWM events. A falling-ramp generator is optionally available to the control the internal DAC reference value for one comparator in the module. Each comparator output is feed through a programmable digital filter that can remove spurious trip signals. The output of the CMPSS generates trip signals to the ePWM event trigger submodule and GPIO structure.

**Buffered Digital-to-Analog Converter (DAC)**

The F2837xD includes three buffered 12-bit DAC modules that can provide a programmable reference output voltage capable of driving an external load. Values written to the DAC can take effect immediately or be synchronized with ePWM events.

**Control peripherals**

The high-performance control peripherals are an integral component for all digital control systems, and within the F2837xD these peripherals are common between the two CPU subsystems. After reset they are connected to the CPU1 subsystem, and a series of CPU Select registers are used to configure each peripheral individually to be either controlled CPU1 subsystem or CPU2 subsystem. The control peripherals include 12 pulse width modulators, six capture modules, three quadrature encoder pulse modules, and two sigma-delta filter modules.

---

*Figure 60: Comparator Subsystem (CMPSS) block diagram.*
Enhanced Pulse Width Modulator (ePWM) module

Power switching devices can be difficult to control when operating in the proportional region, but are easy to control in the saturation and cutoff regions. Since PWM is a digital signal by nature and easy for an MCU to generate, it is ideal for use with power switching devices. Essentially, PWM performs a DAC function, where the duty cycle is equivalent to the DAC analog amplitude value. The F2837xD ePWM modules are highly programmable, extremely flexible, and easy to use, while being capable of generating complex pulse width waveforms with minimal CPU overhead or intervention. Each ePWM module is identical with two PWM outputs, EPWMxA and EPWMxB, and multiple modules can synchronized to operate together as required by the system application design. The ePWM module consists of eight submodules: time-base, counter-compare, action-qualifier, dead-band generator, PWM chopper, trip-zone, digital-compare, and event-trigger.

The time-base submodule consists of a dedicated 16-bit counter, along with built-in synchronization logic to allow multiple ePWM modules to work together as a single system. A clock pre-scaler divides the EPWM clock to the counter and a period register is used to control the frequency and period of the generated waveform. The period register has a shadow register, which acts like a buffer to allow the register updates to be synchronized with the counter, thus avoiding corruption or spurious operation from the register being modified asynchronously by the software. The time-base counter operates in three modes: up-count, down-count, and up-down-count. In up-count mode the time-base counter starts counting from zero and increments until it reaches the period register value, then the time-base counter resets to zero and the count sequence starts again. Likewise, in down-count mode the time-base counter starts counting from the period register value and decrements until it reaches zero, then the time-base counter is loaded with the period value and the count sequence starts again. In up-down-count mode the time-base counter starts counting from zero and increments until it reaches the period register value, then the time-base counter decrements until it reaches zero and the count sequence starts again. In up-down-count mode the time-base counter starts counting from zero and increments until it reaches the period register value, then the time-base counter decrements until it reaches zero and the count sequence repeats. The up-count and down-count modes are used to generate asymmetrical waveforms, and the up-down-count mode is used to generate symmetrical waveforms.

The counter-compare submodule continuously compares the time-base count value to four Counter Compare

Figure 61: Enhanced pulse width modulator (ePWM) module block diagram.
Registers (CMPA, CMPB, CMPC, and CMPD) and 
generates four independent compare events (that is, time-
base counter equals a compare register value) which are 
fed to the action-qualifier and event-trigger submodules. 
The counter compare registers are shadowed to prevent 
corruption or glitches during the active PWM cycle. 
Typically CMPA and CMPB are used to control the 
duty cycle of the generated PWM waveform, and all 
four compare registers can be used to start an ADC 
conversion or generate an ePWM interrupt. For the up-
count and down-count modes, a counter match occurs 
only once per cycle, however for the up-down-count 
mode a counter match occurs twice per cycle since there 
is a match on the up count and down count.

The action-qualifier submodule is the key element in the 
ePWM module which is responsible for constructing 
and generating the switched PWM waveforms. It utilizes 
match events from the time-base and counter-compare 
submodules for performing actions on the EPWMxA 
and EPWMxB output pins. These actions are setting 
the pin high, clearing the pin low, toggling the pin, or do 
nothing to the pin, based independently on count-up 
and count-down time-base match event. The match 
events are when the time-base counter equals the 
period register value, the time-base counter is zero, the 
time-base counter equals CMPA, the time-base counter 
equals CMPB, or a Trigger event (T1 and T2) based on 
comparator, trip, or sync signal. Note that zero and 
period actions are fixed in time, whereas CMPA and 
CMPB actions are movable in time by programming 
their respective registers. Actions are configured 
independently for each output using shadowed registers, 
and any or all events can be configured to generate 
actions on either output.

The dead-band submodule provides a classical 
approach for delaying the switching action of a power 
device. Since power switching devices turn on faster 
than they turn off, a delay is needed to prevent having 
a momentary short circuit path from the supply rail 
to ground. This submodule supports independently 
programmable rising-edge and falling-edge delays with 
various options for generating the appropriate signal 
outputs on EPWMxA and EPWMxB.

The PWM chopper submodule is used with pulse 
transformer-based gate drives to control the power 
switching devices. This submodule modulates a high-
frequency carrier signal with the PWM waveform that 
is generated by the action-qualifier and dead-band 
submodules. Programmable options are available to 
support the magnetic properties and characteristics of 
the transformer and associated circuitry.

The trip-zone submodule utilizes a fast clock 
independent logic mechanism to quickly handle fault 
conditions by forcing the EPWMxA and EPWMxB 
outputs to a safe state, such as high, low, or high 
impedance, thus avoiding any interrupt latency that 
may not protect the hardware when responding to over 
current conditions or short circuits through ISR software. 
It supports one-shot trips for major short circuits or 
over current conditions, and cycle-by-cycle trips for 
current limiting operation. The trip-zone signals can be 
generated externally from any GPIO pin which is mapped 
through the Input X-Bar (TZ1 – TZ3), internally from an 
inverted eQEP error signal (TZ4), system clock failure 
(TZ5), or from an emulation stop output from the CPU 
(TZ6). Additionally, numerous trip-zone source signals 
can be generated from the digital-compare subsystem.

The digital-compare subsystem compares signals 
external to the ePWM module, such as a signal from 
the CMPSS analog comparators, to directly generate 
PWM events or actions which are then used by the trip-
zone, time-base, and event-trigger submodules. These 
‘compare’ events can trip the ePWM module, generate 
a trip interrupt, sync the ePWM module, or generate an 
ADC start of conversion. A compare event is generated 
when one or more of its selected inputs are either high 
or low. The signals can originate from any external GPIO 
pin which is mapped through the Input X-Bar and from 
various internal peripherals which are mapped through 
the ePWM X-Bar. Additionally, an optional ‘blanking’ 
function can be used to temporarily disable the compare 
action in alignment with PWM switching to eliminate 
noise effects.

The event-trigger submodule manages the events 
generated by the time-base, counter-compare, and
digital-compare submodules for generating an interrupt to the CPU and/or a start of conversion pulse to the ADC when a selected event occurs. These event triggers can occur when the time-base counter equals zero, period, zero or period, the up or down count match of a compare register (that is, CMPA, CMPB, CMPC, or CMPD). Recall that digital-compare subsystem can also generate an ADC start of conversion based on one or more compare events. The event-trigger submodule incorporates pre-scaling logic to issue an interrupt request or ADC start of conversion at every event or up to every fifteenth event.

The ePWM module is capable of significantly increase its time resolution capabilities over the standard conventionally derived digital PWM. This is accomplished by adding 8-bit extensions to the High Resolution Compare Register (CMPxHR), Time Base Period High Resolution Register (TBPRDHR), and High-Resolution Phase Register (TBPHSHR), providing a finer time granularity for edge positioning control. This is known as high-resolution PWM (HRPWM) and it is based on micro edge positioner (MEP) technology.

The MEP logic is capable of positioning an edge very finely by sub-dividing one coarse system clock of the conventional PWM generator with time step accuracy on the order of 150 ps. A self-checking software diagnostics mode is used to determine if the MEP logic is running optimally, under all operating conditions such as for variations caused by temperature, voltage, and process. HRPWM is typically used when the PWM resolution falls below approximately 9 or 10 bits which occurs at frequencies greater than approximately 200 kHz with an EPWMCLK of 100 MHz.

Enhanced Capture (eCAP) module

The eCAP module is used to accurately time external events by time stamping transitions on the capture input pin. It can be used to measure the speed of a rotating machine, determine the elapsed time between pulses, calculate the period and duty cycle of a pulse train signal, and decode current/voltage measurements derived from duty cycle encoded current/voltage sensors. The eCAP module captures signal transitions on a dedicated input pin and sequentially loads a 32-bit time-base counter value in up to four 32-bit time-stamp Capture Registers (CAP1 – CAP4). Independent edge polarity can be configured as rising or falling edge, and the module can be run in either one-shot mode for up to four time-stamp events or continuous mode to capture up to four time-stamp events operating as a circular buffer. The capture input pin is routed through the Input X-Bar, allowing any GPIO pin on the device to be used as the input. Also, the input capture signal can be pre-scaled and interrupts can be generated on any of the four capture events. The time-base counter can be run optimally, under all operating conditions such as for variations caused by temperature, voltage, and process.

Figure 62: Enhanced Capture (eCAP) module block diagram.
in either absolute or difference (delta) time-stamp mode. If the module is not used in capture mode, the eCAP module can be configured to operate as a single channel asymmetrical PWM module (that is, the time-base counter operates in count-up mode).

**Enhanced Quadrature Encoder Pulse (eQEP) module**

The eQEP module interfaces with a linear or rotary incremental encoder for determining position, direction, and speed information from a rotating machine that is typically found in high-performance motion and position-control systems. The inputs include two pins (QEPA and QEPB) for quadrature-clock mode or direction-count mode, an index pin (QEPI), and a strobe pin (QEPS). These pins are configured using the GPIO multiplexer and need to be enabled for synchronous input. In quadrature-clock mode, two square wave signals from a position encoder are inputs to QEPA and QEPB which are 90 electrical degrees out of phase. This phase relationship is used to determine the direction of rotation. If the position encoder provides direction and clock outputs, instead of quadrature outputs, then direction-count mode can be used. QEPA input will provide the clock signal and QEPB input will have the direction information. The QEPI index signal occurs once per revolution and can be used to indicate an absolute start position from which position information is incrementally encoded using quadrature pulses. The QEPS strobe signal can be connected to a sensor or limit switch to indicate that a defined position has been reached.

**Sigma-Delta Filter Module (SDFM)**

The SDFM is a four-channel digital filter designed specifically for current measurement and resolver position decoding in motor control applications. Each channel can receive an independent delta-sigma modulator bit stream which is processed by four individually programmable digital decimation filters. The filters include a fast comparator for immediate digital threshold comparisons for over-current and undercurrent monitoring. Also, a filter-bypass mode is available to enable data logging, analysis, and customized filtering. The SDFM pins are configured using the GPIO multiplexer. A key benefit of the SDFM is it enables a simple, cost-effective, and safe high-voltage isolation boundary.

**Control Law Accelerator (CLA)**

The CLA is an independent 32-bit floating-point math hardware accelerator which executes real-time control algorithms in parallel with the main C28x CPU, effectively doubling the computational performance. Each CPU subsystem has its own CLA that responds directly to peripheral triggers, which can free up the C28x CPU for other tasks, such as communications and diagnostics. With direct access to the various control...
and communication peripherals, the CLA minimizes latency, enables a fast trigger response, and avoids CPU overhead. Also, with direct access to the ADC results registers, the CLA is able to read the result on the same cycle that the ADC sample conversion is completed, providing “just-in-time” reading, which reduces the sample to output delay.

The CLA has access to the LSx RAM blocks and each memory block can be configured to be either dedicated to the CPU or shared between the CPU and CLA. After reset the memory block is mapped to the CPU, where it can be initialized by the CPU before being shared with the CLA. Once it is shared between the CPU and CLA it then can be configured to be either program memory or data memory. When configured as program memory it contains the CLA program code, and when configured as data memory it contains the variable and coefficients that are used by the CLA program code. Additionally, dedicated message RAMs are used to pass data between the CPU and CLA, and CLA and CPU.

Programming the CLA consists of initialization code, which is performed by the CPU, and tasks. A task is similar to an interrupt service routine, and once started it runs to completion. Tasks can be written in C or assembly code, where typically the user will use assembly code for high performance time-critical tasks, and C for non-critical tasks. Each task is capable of being triggered by a variety of peripherals without CPU intervention, which makes the CLA very efficient since it does not use interrupts for hardware synchronization, nor must the CLA do any context switching. Unlike the traditional interrupt-based scheme, the CLA approach becomes deterministic. The CLA supports eight independent tasks and each is mapped back to an event trigger. Since the CLA is a software programmable accelerator, it is very flexible and can be modified for different applications.

**Direct Memory Access (DMA)**

The DMA module provides a hardware method of transferring data between peripherals and/or memory without intervention from the CPU, effectively freeing up the CPU for other functions. Each CPU subsystem has its own DMA and using the DMA is ideal when an application requires a significant amount of time spent moving large amounts of data from off-chip memory to on-chip memory, or from a peripheral such as the ADC result register to a memory RAM block, or between two peripherals. Additionally, the DMA is capable of rearranging the data for optimal CPU processing such as binning and “ping-pong” buffering.

A DMA transfer is started by a peripheral or software trigger. There are six independent DMA channels, where
Figure 65: Direct Memory Access (DMA) - triggers, sources, and destinations.

Each channel can be configured individually and each DMA channel has its own unique PIE interrupt for CPU servicing. All six DMA channels operate the same way, except channel 1 can be configured at a higher priority over the other five channels. At its most basic level the DMA is a state machine consisting of two nested loops and tightly coupled address control logic which gives the DMA the capability to rearrange the blocks of data during the transfer for post processing.

Inter-Processor Communications (IPC)

The IPC module facilitates communications between the two CPU subsystems, and all IPC features are independent of each other. As discussed in the Memory section, there are two dedicated 1Kx16 blocks of Message RAM that are used to transfer messages or data between CPU1 and CPU2. One block configuration is fixed for “CPU1 to CPU2”, and the other block configuration is fixed for “CPU2 to CPU1”.

Messaging can be accomplished using IPC flags and interrupts. There are 32 IPC event signals from CPU1 to CPU2, and vice-versa. These signals can be used for flag-based event polling and four of them (IPC0 – IPC3) can be configured to generate IPC interrupts on the remote CPU. IPC Command registers provide a simple and flexible means for CPU1 and CPU2 to exchange more complex messages. Each CPU has eight dedicated registers, four for sending messages and four for receiving messages. On the local CPU, three are writable registers and one is a read-only register. These same registers are accessible on the remote CPU as three read-only registers and one writable register. The given register names were chosen to support a simple command/response protocol, but they can be used for any purpose to suit the applications software.

A variety of options exist for supporting IPC. The basic option does not require any software drivers and uses only the IPC registers for simple message passing. An IPC-Lite software API driver uses only the IPC registers (that is, no memory used), but is limited to one IPC interrupt or one IPC command/message at a time. The full IPC software API driver uses circular buffers for message RAMs, and can queue up to four messages prior to processing. It can also be used with multiple IPC ISRs at a time, but it requires additional setup in the application code prior to use. Each option has trade offs between complexity, processing overhead, and messaging capabilities.
Communications Peripherals

The F2837xD dual-core MCU includes numerous communications peripherals that extend the connectivity of the device. There are up to three Serial Peripheral Interface (SPI) modules, four Serial Communication Interface (SCI) modules, two Multi-channel Buffered Serial Port (McBSP) modules, two Inter-Integrated Circuit (I2C) modules, two Controller Area Network (CAN) modules, one Universal Serial Bus (USB) module, and one Universal Parallel Port (uPP) module. These peripherals can be assigned to either the CPU1 subsystem or the CPU2 subsystem, except for the USB and uPP which is dedicated to only the CPU1 subsystem.

The SPI is a high-speed synchronous serial port that shifts a programmable length serial bit stream into and out of the device at a programmable bit-transfer rate. It is typically used for communications between processors and external peripherals, and it has a 16-level deep receive and transmit FIFO for reducing servicing overhead.

The SCI is a two-wire asynchronous serial port (also known as a UART) that supports communications between the processor and other asynchronous peripherals that use the standard non-return-to-zero (NRZ) format. A receiver and transmitter 16-level deep FIFO is used to reduce servicing overhead.
The McBSP provides a high-speed direct interface to codecs, analog interface chips (AICs), and other serially connected A/D and D/A devices. It has double-buffered transmission and triple-buffered reception for supporting continuous data streams. There are 128 channels for transmission and reception, and data size selections of 8, 12, 16, 20, 24, and 32 bits, along with μ-law and A-law companding.

The I²C provides an interface between devices that are compliant I²C-bus specification version 2.1 and connect using an I²C-bus. External components attached to the 2-wire serial bus can transmit or receive 1 to 8-bit data to or from the device through the I²C module.

The CAN module is a serial communications protocol that efficiently supports distributed real-time control with a high level of security. It supports bit-rates up to 1 M-bit/s and is compliant with the CAN 2.0B protocol specification.

The USB operates as a full-speed function controller during point-to-point communications with a USB host. It complies with the USB 2.0 standard, and a dynamically sizable FIFO supports queuing of multiple packets.

The uPP is a high-speed parallel interface with dedicated data lines and minimal control signals. It interfaces with high-speed ADCs or DACs with 8-bit data widths, as well as field-programmable gate arrays (FPGAs) or other uPP devices to achieve high-speed data transfer. An internal DMA controller is used to maximize throughput and minimize processing overhead during high-speed data transmission.

Summary

The F2837xD MCU device family is designed to solve the most demanding control system requirements found in many high performance real-time control applications. Based on an extremely fast C28x dual-core architecture, advanced control peripherals, and integrated analog functions, the F2837xD can reduce overall system cost while increasing system reliability. With each CPU running up to 200 MHz, combined with its own CLA running concurrently, the device has the capability for delivering the equivalent performance of 800 MHz. The F2837xD MCU device family is ideal for applications requiring advanced signal processing, such as industrial drives, digital power, motor control, renewable energy, and smart sensing. Table 1 provides a general feature comparison between the F2837xD, F2837xS, and F2807x families. For more details, see the device-specific technical reference documentation.

Additional Resources

Here are additional resources regarding Product Overviews that you may find of interest:

- TI Training

Important Notice: The products and services of Texas Instruments Incorporated and its subsidiaries described herein are sold subject to TI's standard terms and conditions of sale. Customers are advised to obtain the most current and complete information about TI products and services before placing orders. TI assumes no liability for applications assistance, customer’s applications or product designs, software performance, or infringement of patents. The publication of information regarding any other company’s products or services does not constitute TI's approval, warranty or endorsement thereof.

The platform bar is a trademark of Texas Instruments. All other trademarks are the property of their respective owners.
IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), 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, 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 (www.ti.com/legal/termsofsale.html) 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.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2020, Texas Instruments Incorporated