Technical Brief
SPRT720–February 2016

The TMS320F2837xD Architecture: Achieving a New Level of High Performance

Kenneth W. Schachter *C2000 Technical Staff*

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.

Contents

List of Figures

List of Tables

1 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.

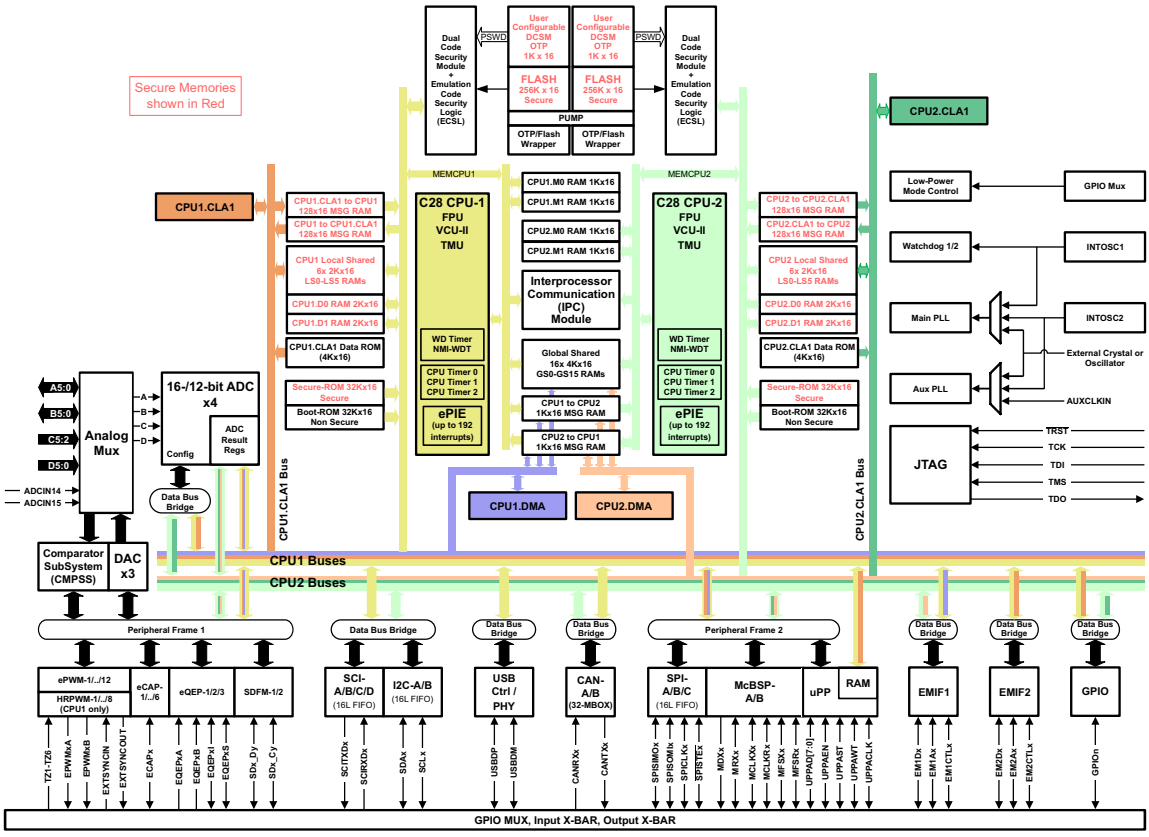

Figure 1. F2837xD Functional Block Diagram

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 single-core 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.

2 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.

3 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.

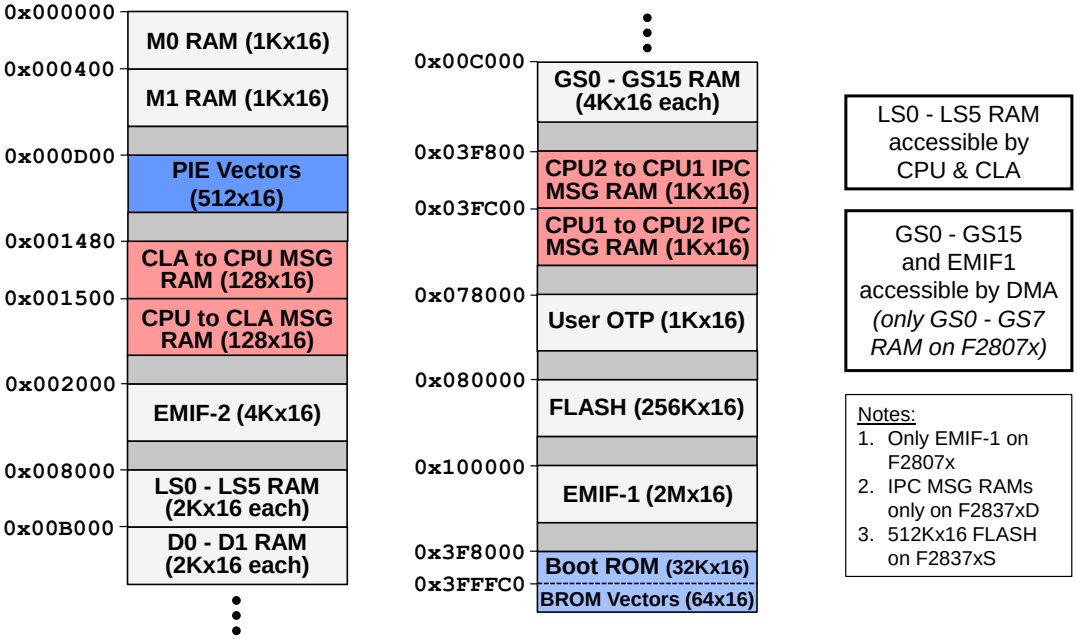

Figure 2. Simplified Memory Map

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 DO and D1 can be secured.

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 settings used 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.

4 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.

5 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.

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.

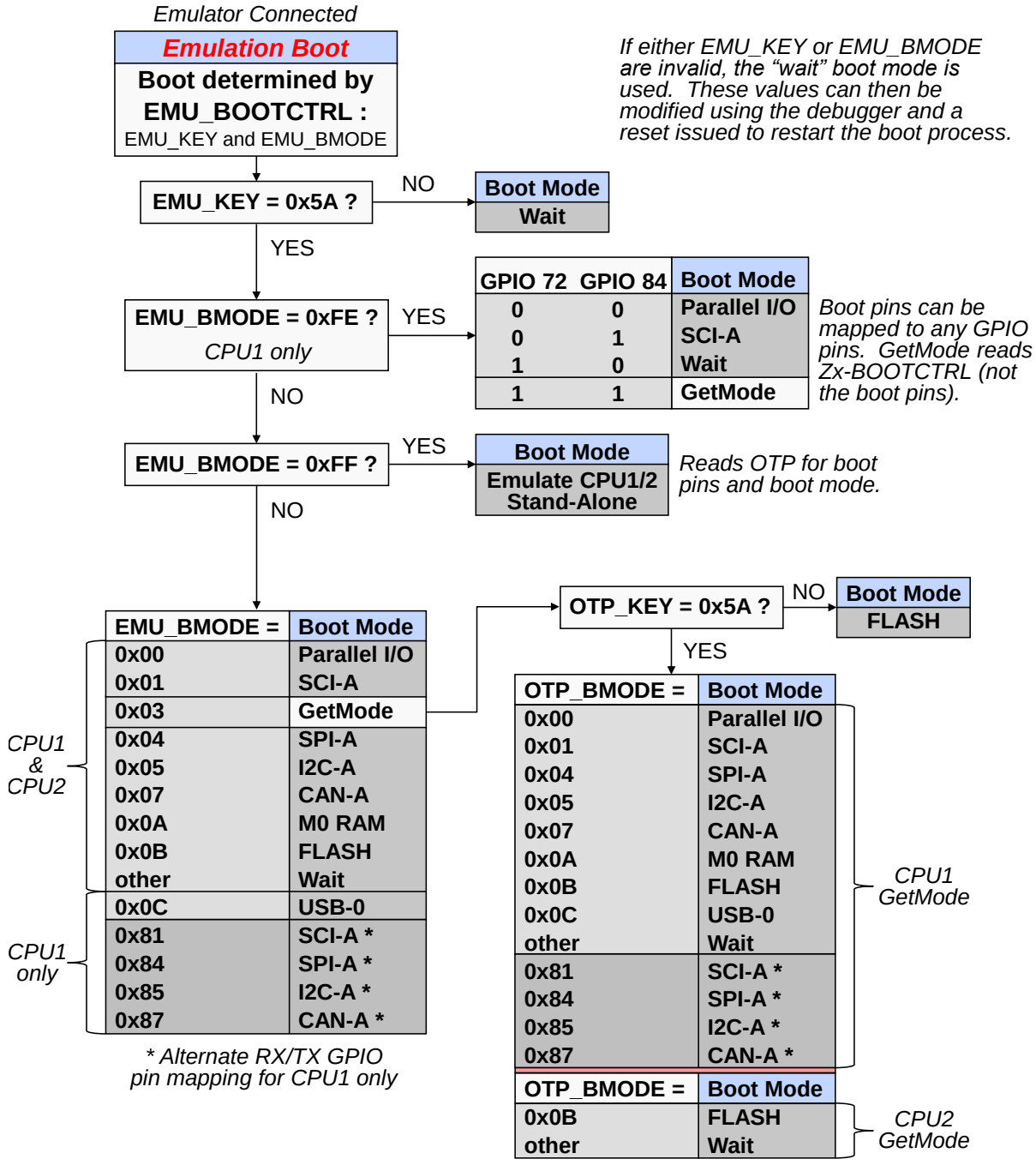

Figure 3. Emulation Boot Mode

Once the EMU_KEY bit fields are set to 0x5A, then the EMU_BMODE bit field values determines 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 Z1-BOOTCTRL 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.

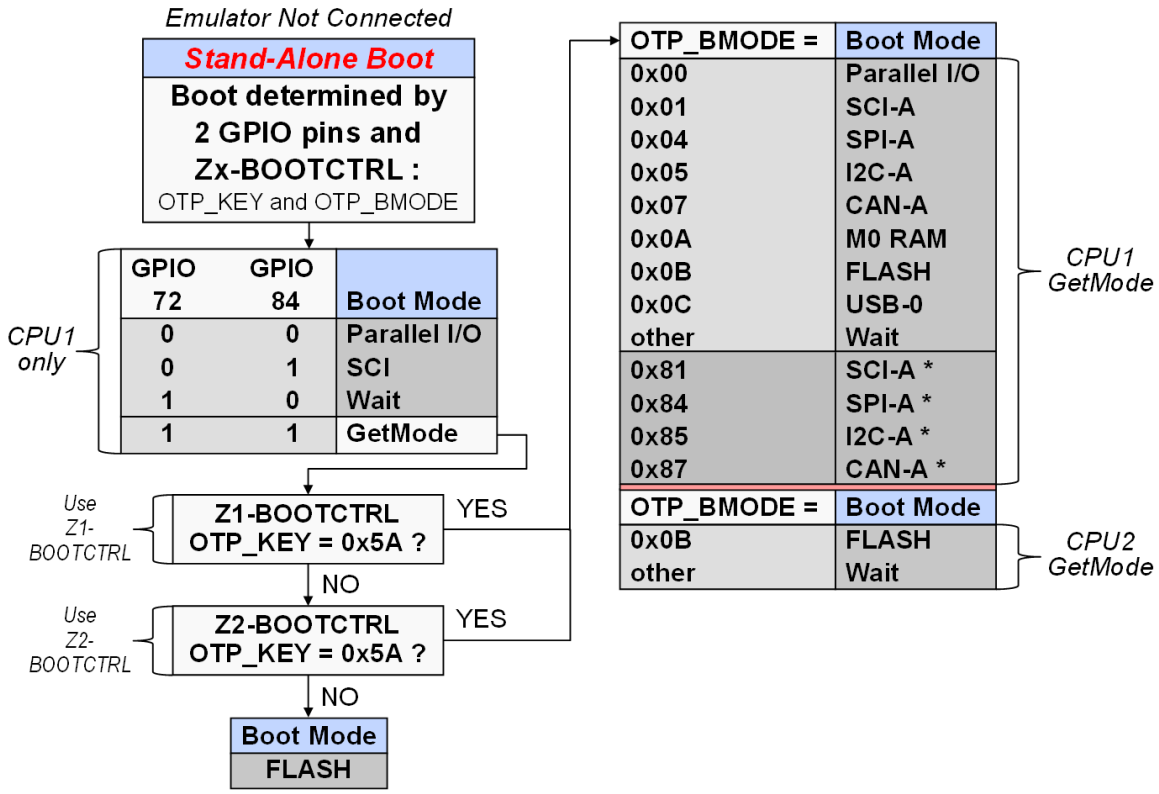

Figure 4. Stand-Alone Boot Mode

6 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.

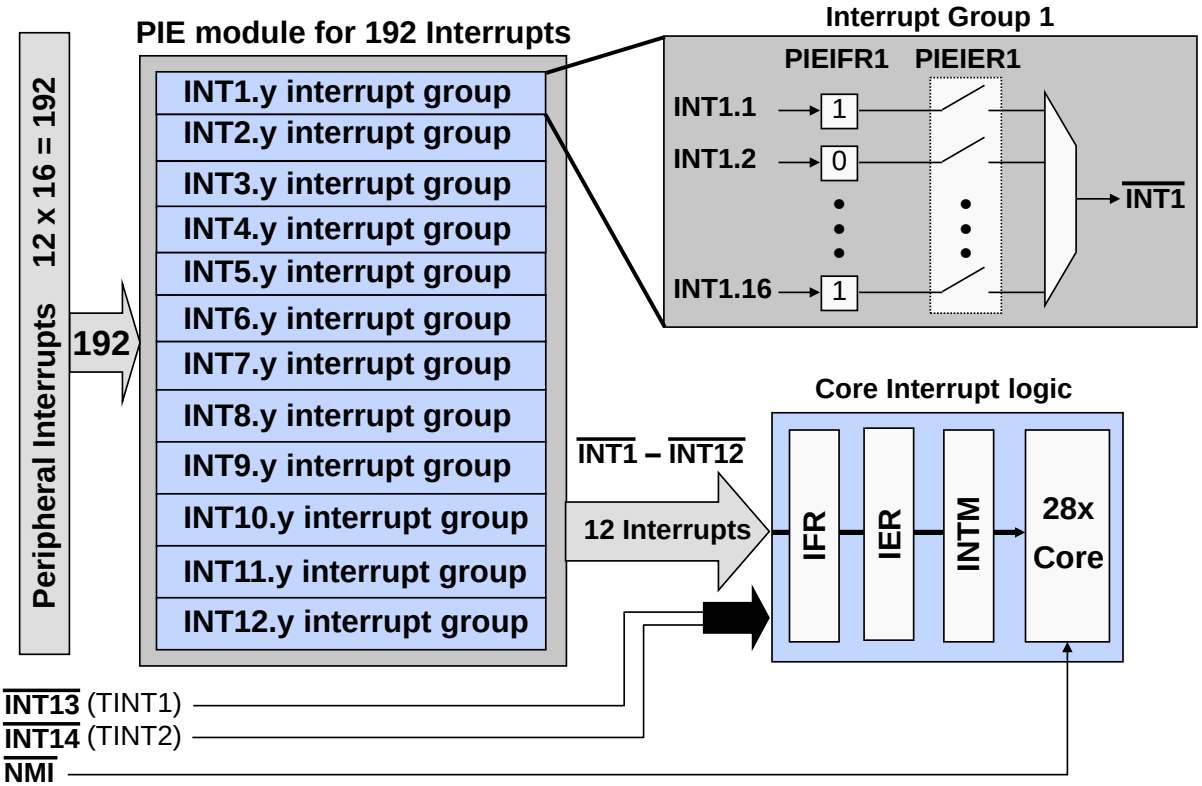

Figure 5. Peripheral Interrupt Expansion (PIE) Module

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.

7 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.

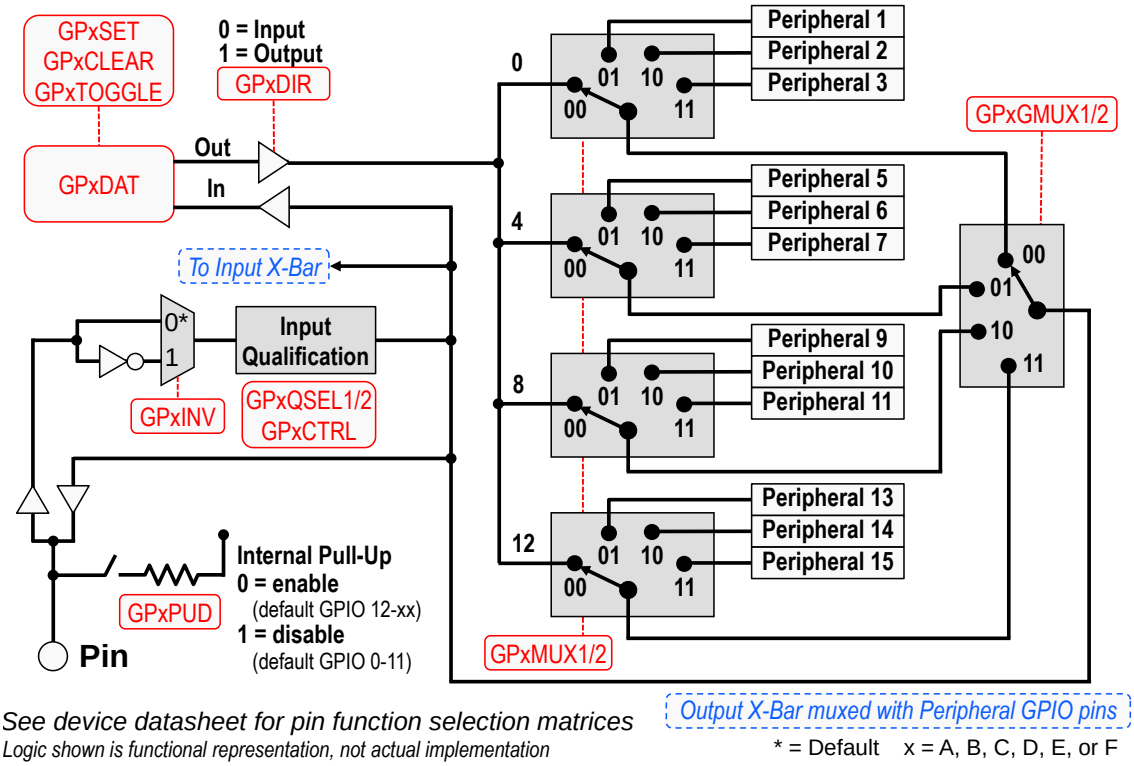

Figure 6. General-Purpose Input/Output (GPIO) Pin Block Diagram

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.

8 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 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.

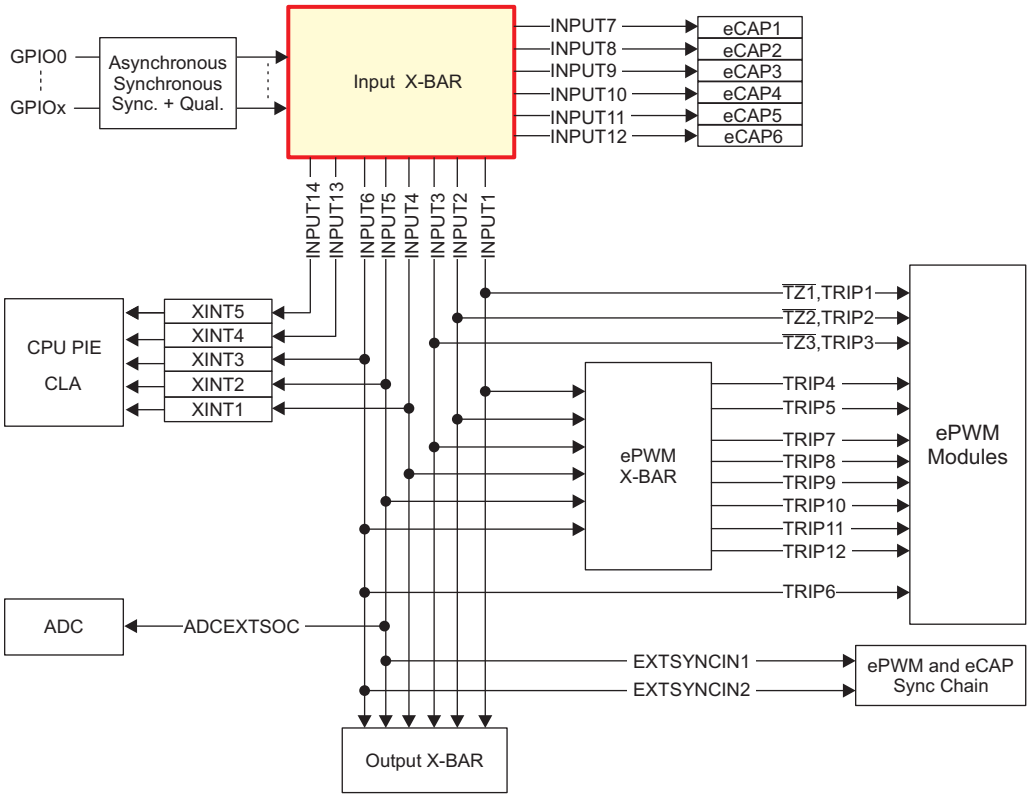

Figure 7. Input X-BAR

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.

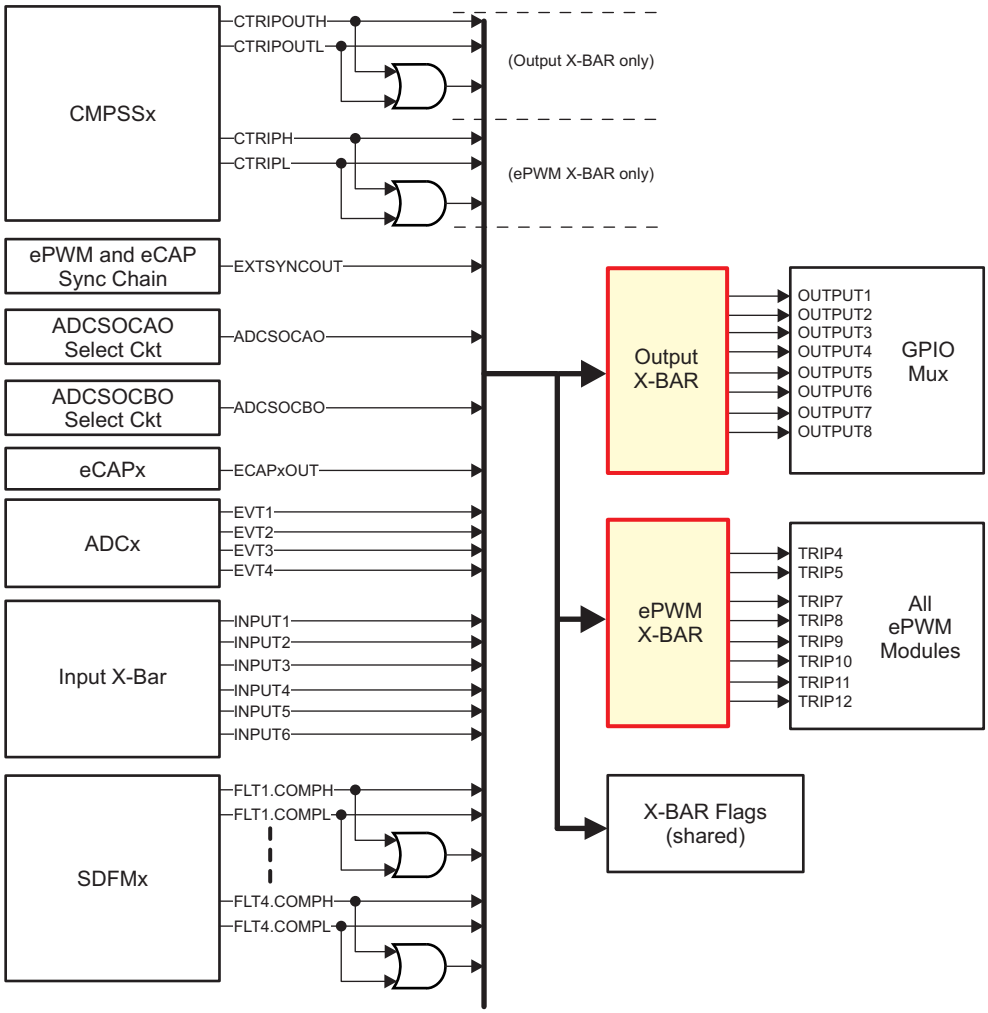

Figure 8. Output and ePWM X-BARs

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.

9 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.

9.1 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.

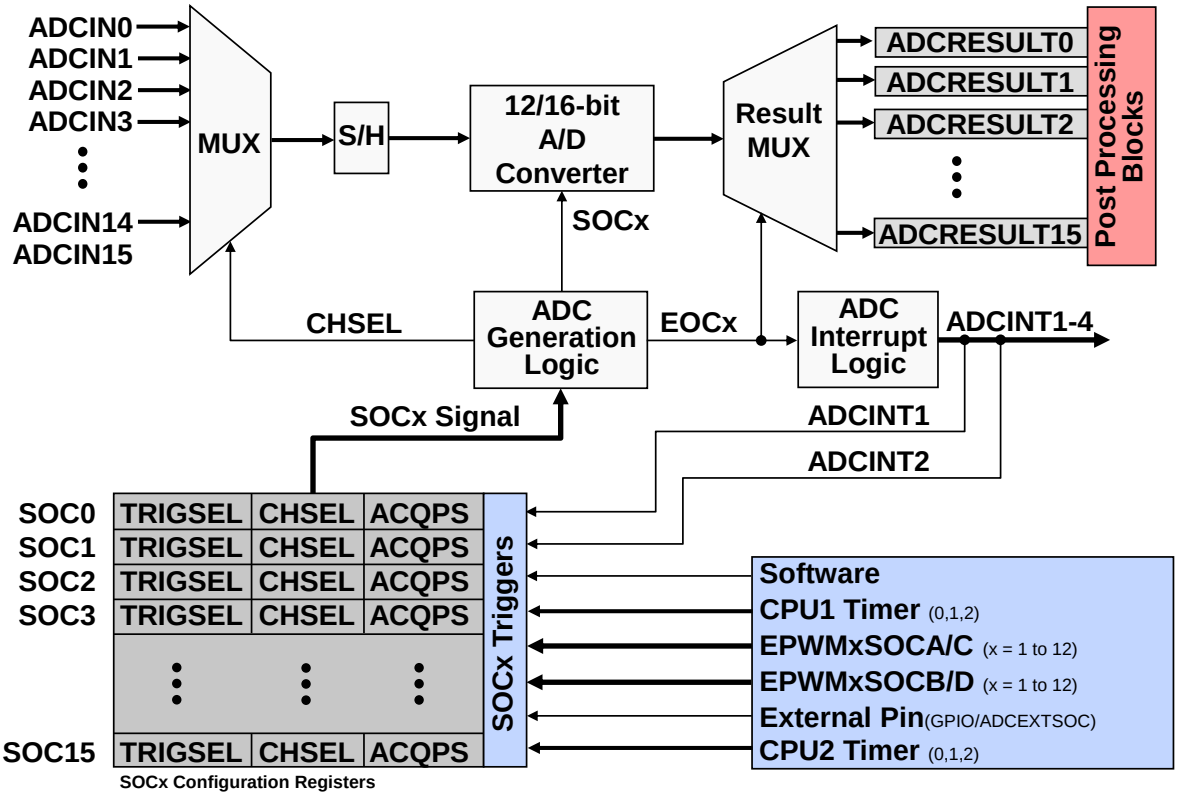

Figure 9. Analog-to-Digital Converter (ADC) Block Diagram

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.

9.2 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.

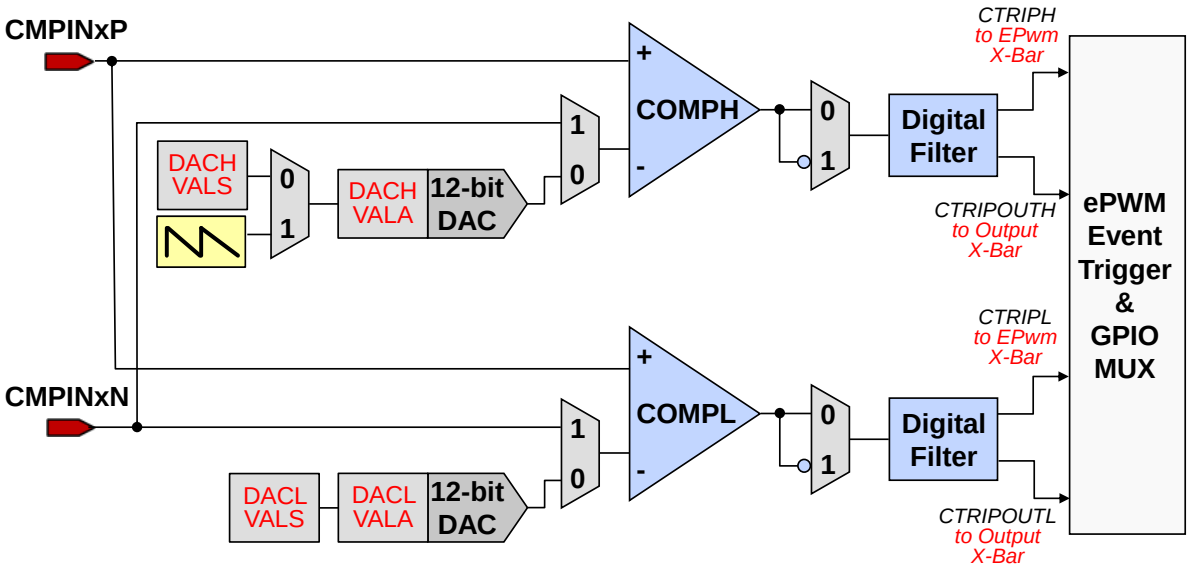

Figure 10. Comparator Subsystem (CMPSS) Block Diagram

9.3 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.

10 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.

10.1 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.

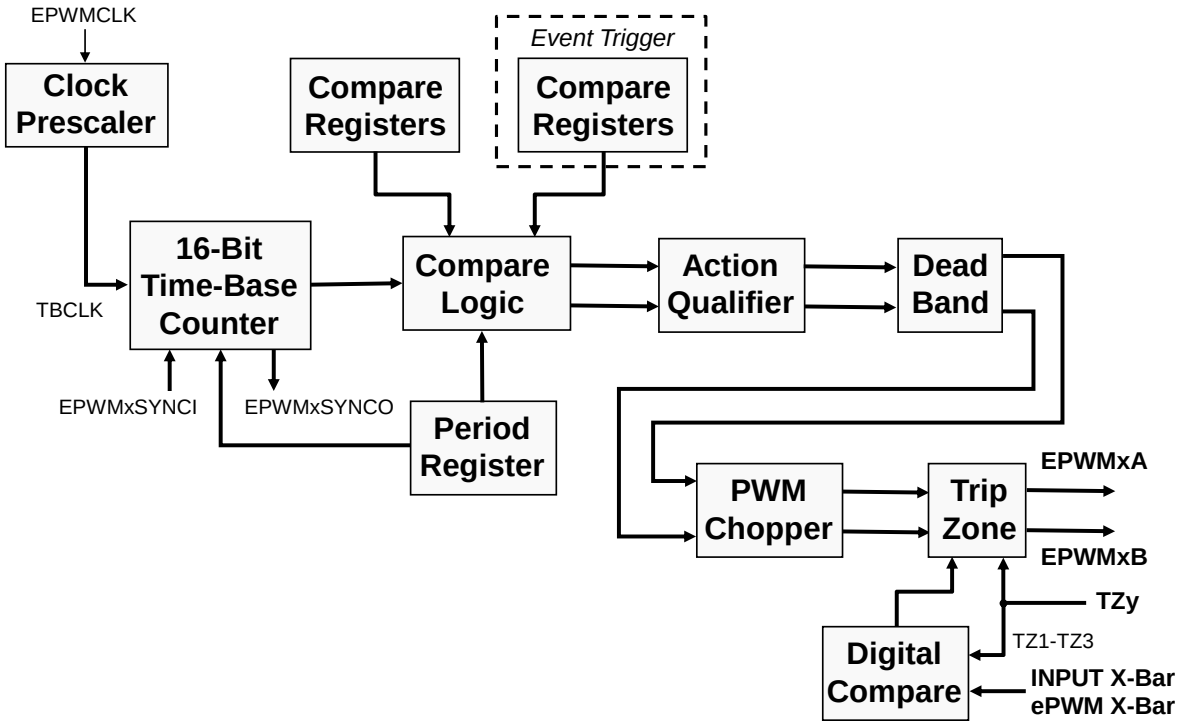

Figure 11. Enhanced Pulse Width Modulator (ePWM) Module Block Diagram

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 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 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 a comparator, trip, or sync signal. Note that zero and period actions are fixed in time, whereas CMPA and CMPB actions are moveable 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.

10.2 Enhanced Capture (eCAP) Module

The eCAP module is used to accurately time external events by timestamping 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 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).

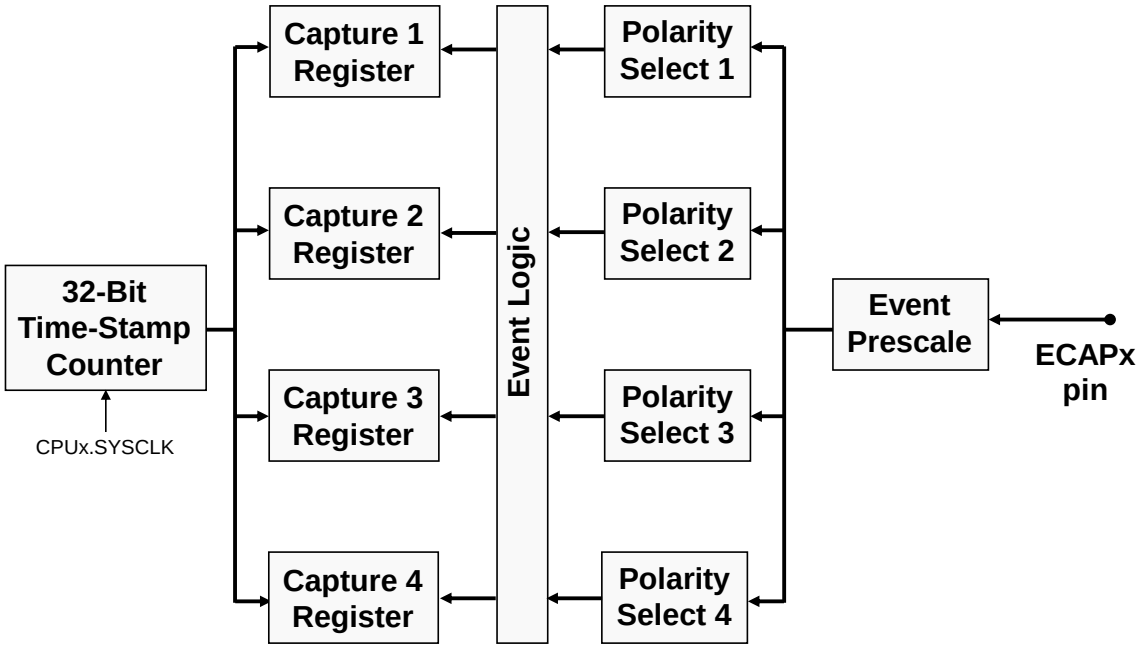

Figure 12. Enhanced Capture (eCAP) Module Block Diagram

10.3 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.

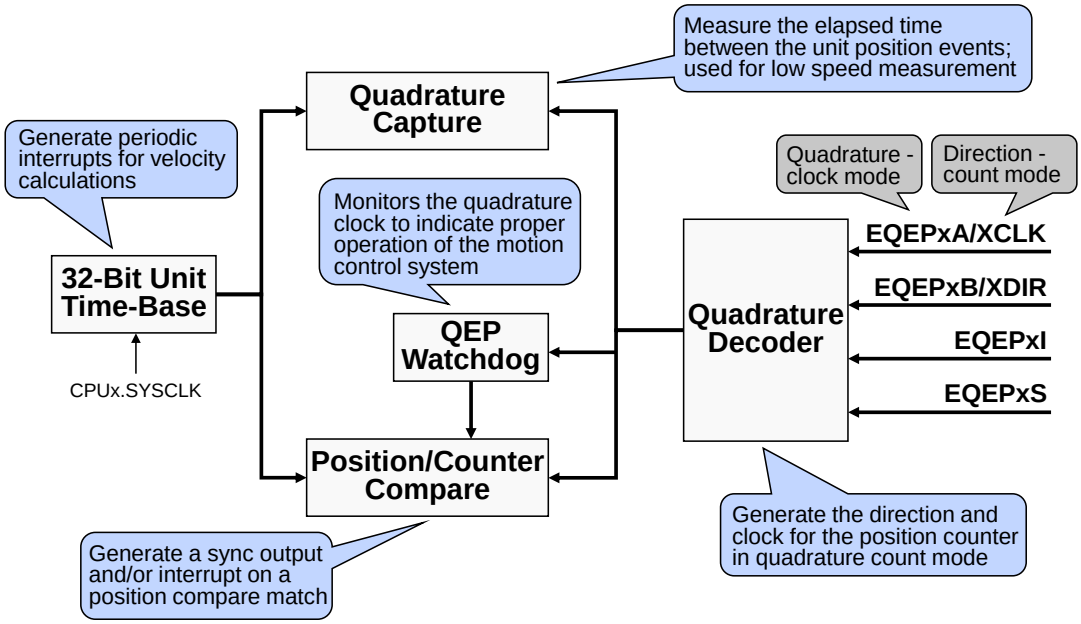

Figure 13. Enhanced Quadrature Encoder Pulse (eQEP) Module Block Diagram

10.4 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 under-current 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.

11 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.

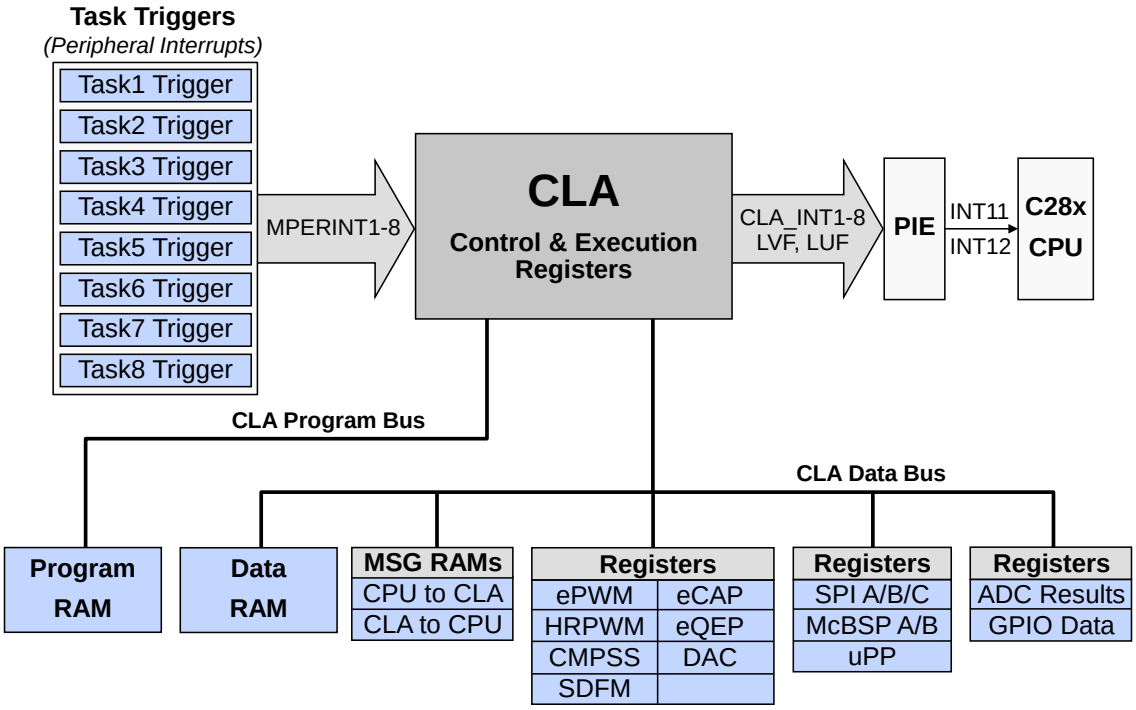

Figure 14. Control Law Accelerator (CLA) Block Diagram

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.

12 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.

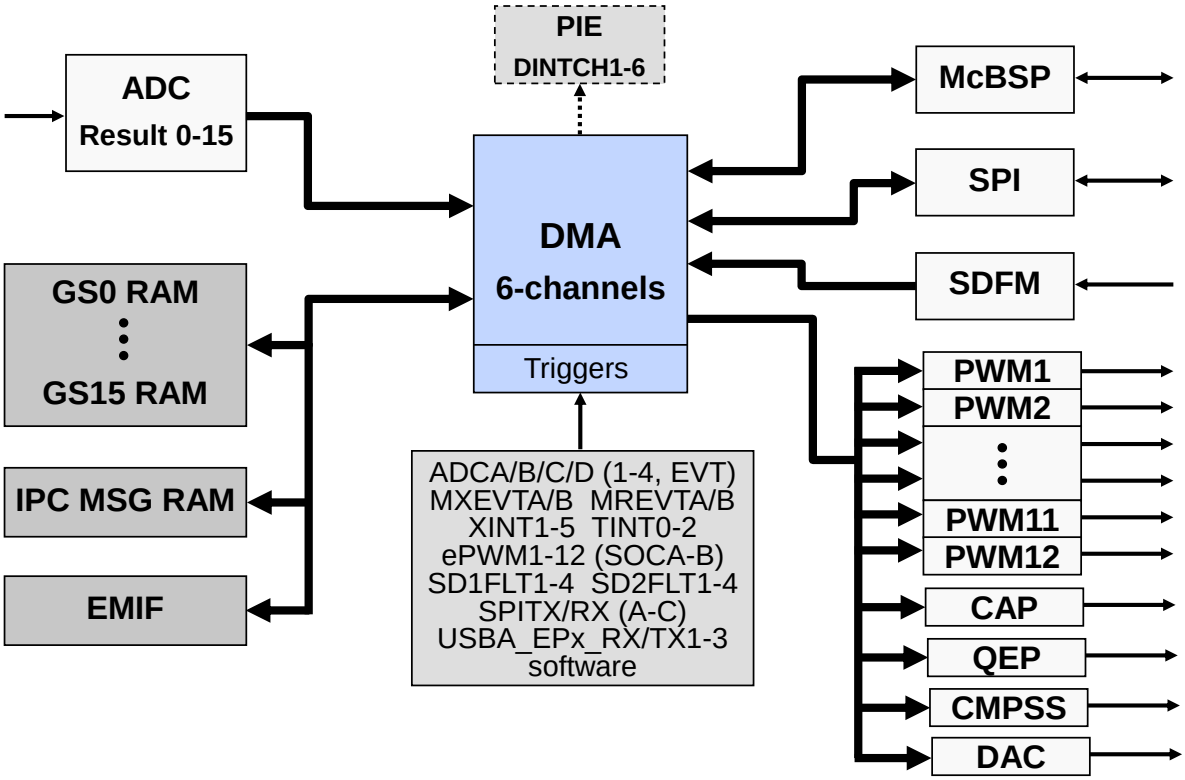

Figure 15. Direct Memory Access (DMA) – Triggers, Sources, and Destinations

A DMA transfer is started by a peripheral or software trigger. There are six independent DMA channels, where 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.

13 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 writeable registers and one is a read-only register. These same registers are accessible on the remote CPU as three read-only registers and one writeable 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 tradeoffs between complexity, processing overhead, and messaging capabilities.

14 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 Mbit/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 sizeable 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.

15 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.

Table 1. Device Matrix

	F2807x	F2837xS	F2837xD
C28x CPUs	1	1	2
Clock	120 MHz	200 MHz	200 MHz
Flash / RAM / OTP	256Kw / 50Kw / 2Kw	512Kw / 82Kw / 2Kw	512Kw / 102Kw / 2Kw
On-Chip Oscillators	√	√	√
FPU	√	√	√ (each CPU)
VCU	–	√	√ (each CPU)
TMU	√	√	√ (each CPU)
CLA	√	√	√ (each CPU)
6-Channel DMA	√	√	√ (each CPU)
32-Bit CPU Timers	√	√	√ (each CPU)
Watchdog Timer	√	√	√(each CPU)
ADC	Three 12-bit	Four 12/16-bit	Four 12/16-bit
CMPSS w/DAC	√	√	√
Buffered DAC	√	√	√
ePWM / HRPWM	√ / √	√ / √	√ / √
eCAP / HRCAP	√ / –	√ / –	√ / –
eQEP	√	√	√
SDFM	√	√	√
SCI	√	√	√
SPI	√	√	√
I2C	√	√	√
CAN	√	√	√
McBSP	√	√	√
USB	√	√	√
uPP	–	√	√
EMIF	1	2	2

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers’ products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers’ products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

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