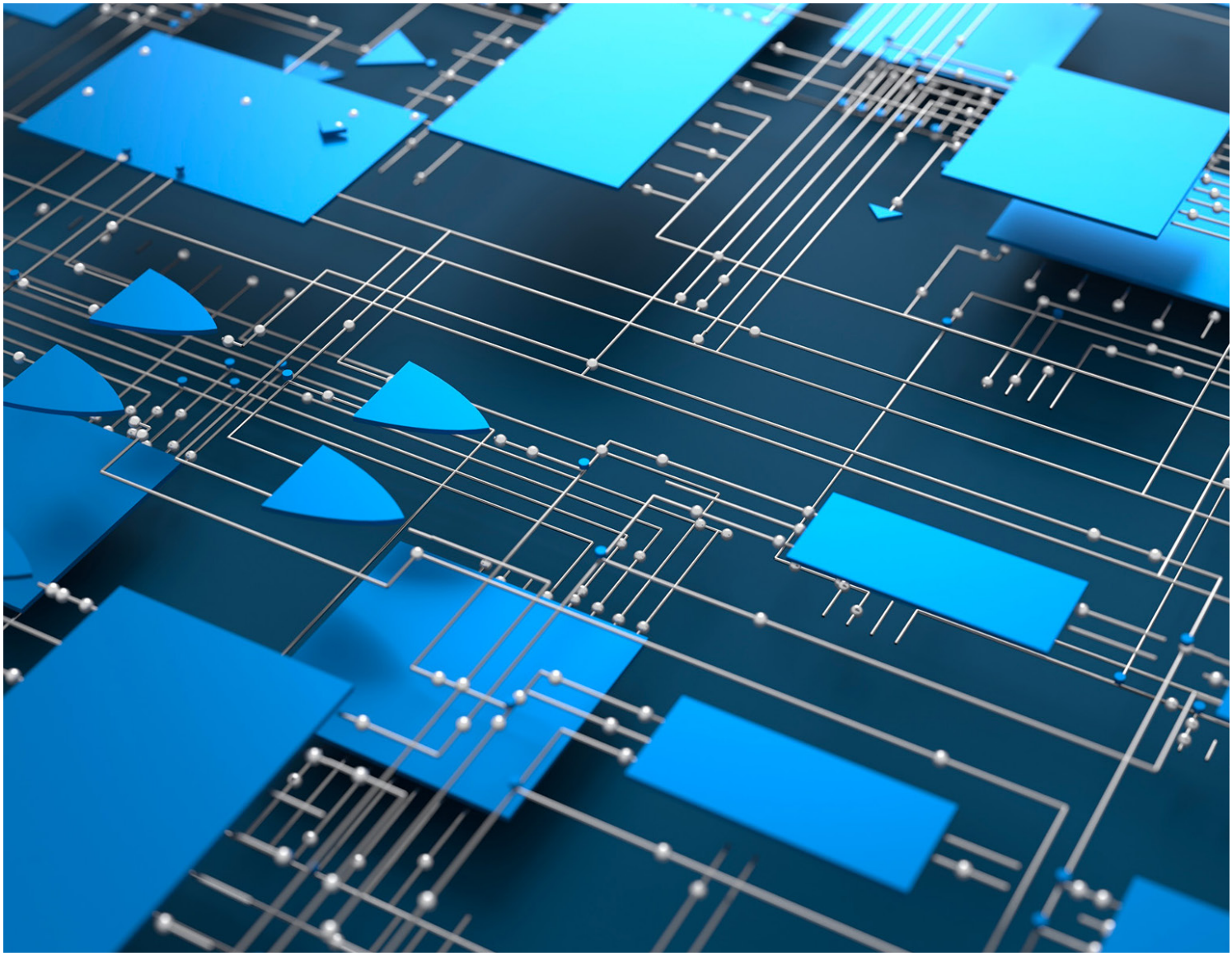
Implementing Quadrature Encoders with Configurable Logic

Optimizing BOM and Power While Maintaining Flexibility

By Brandon Lewis for Mouser Electronics



Source: Ricardo/stock.adobe.com

MOUSER ELECTRONICS | MICROCHIP

Quadrature encoders are often used in applications that need bidirectional rotation tracking. They are commonly integrated into user interface elements—like dials and knobs to track user input—and into motion control systems to provide the position feedback necessary for precise movement control.

These encoders serve several key industries:

- Industrial automation and robotics use quadrature encoders for precise positioning of motor-driven systems.
- Manufacturers of e-bikes and other mobility products rely on them for motor control and user input.
- Consumer electronics implement them in dials, scroll wheels, and other controls.

While they are popular, challenges arise when implementing quadrature encoders. Decoding the signals from the encoder requires either dedicated hardware or significant processing resources from the microcontroller. Both design paths involve difficult trade-offs in cost, design complexity, and power consumption.

This paper explains how the configurable logic block (CLB) in the Microchip Technology PIC16F13145 family of microcontrollers provides a solution for effectively implementing a quadrature encoder. In doing so, this paper shows how to decode quadrature signals efficiently, reducing power usage, component count, and design effort.

How Quadrature Encoded Signals Work

Quadrature encoders generate two digital signals that provide information about the position and direction of rotation. The physical construction of quadrature encoders can take several forms, each suited to different applications and environmental conditions:

- **Mechanical encoders** employ contact pairs that open and close as a shaft rotates, making them robust but subject to contact bounce.
- **Optical encoders** use LED and photosensor pairs, with a slotted disk interrupting the light path to generate pulses. These encoders offer high precision and reliability in clean environments.
- **Magnetic encoders** use Hall effect sensors to detect the position of a magnetized wheel, providing good reliability in harsh conditions without mechanical wear.

Regardless of the physical implementation, all quadrature encoders produce two output channels, commonly labeled A and B. These channels are intentionally offset by 90 degrees (one-quarter of a cycle, hence “quadrature”). This phase relationship is key to determining the direction of rotation.

Direction is determined by examining the phase relationship between channels A and B. When rotation is clockwise (CW), signal A will lead signal B; that is, there will be either a rising or falling edge on channel A followed by the same transition on channel B. Conversely, when rotation is counterclockwise (CCW), signal B will lead signal A (Figure 1).

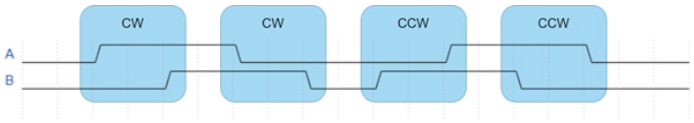


Figure 1: *The direction of rotation in a quadrature encoder is determined by the relationship between the A and B channel signals. (Source: Microchip Technology)*

In other words, the quadrature signals form a repeating four-state sequence (**Table 1**). During clockwise rotation, the states progress in order, 0→1→2→3→0. During counterclockwise rotation, they progress in reverse order, 0→3→2→1→0.

Table 1: *Quadrature encoders progress through four states*

State	Channel A	Channel B
0	0	0
1	1	0
2	1	1
3	0	1

Position can be tracked by maintaining a counter that increments or decrements based on the detected direction. These counters can also be used to determine velocity by recording the time between position changes or by counting the number of changes within a fixed time window. This capability is particularly valuable in motor control applications that require speed regulation.

While quadrature encoders' high-resolution feedback makes them critical in applications requiring accurate position and speed measurements, decoding these signals efficiently can be challenging, especially in resource-constrained systems.

Quadrature Encoded Design Challenges

The first challenge with quadrature encoders is signal integrity. All encoders can produce noisy signals: Mechanical contacts can bounce, optical sensors can get dusty, and magnetic sensors may pick up interference. Thus, signal conditioning (often called "debouncing," regardless of the noise source) is needed before decoding to ensure reliability.

The need for both debouncing and decoding requires a careful balance of hardware and software resources. Engineers typically choose between hardware-based or software-based implementation approaches, each with its own trade-offs.

Traditional Hardware Solutions

The traditional approach uses discrete components to condition and decode the quadrature signals. These components typically include:

- Resistor-capacitor (RC) networks for debouncing mechanical contacts,
- Schmitt-trigger ICs to ensure clean signal edges, and
- Dedicated decoder ICs to convert the quadrature signals into count and direction outputs.

This hardware-centric approach offers excellent power efficiency since it requires minimal central processing unit (CPU) intervention. However, it impacts the bill of materials (BOM) and demands additional printed circuit board (PCB) area.

In addition, hardware solutions can be inflexible. Updating the design to accommodate different sensors or detection algorithms usually requires circuit modifications.

Alternative Software Solutions

The alternative approach is to handle quadrature decoding in software, reading the encoder signals directly into GPIO pins. This method simplifies the hardware but introduces other challenges.

First, the CPU must constantly monitor the input pins to detect transitions. This prevents the microcontroller from entering sleep mode, which results in higher power consumption—a critical consideration for battery-powered devices.

The other challenge is timing. The software approach may struggle with timing constraints on resource-limited microcontrollers. The CPU must be fast enough to:

- Sample the inputs frequently enough to catch all transitions,
- Process the samples through debouncing algorithms,
- Decode the quadrature sequence, and
- Handle any other system tasks.

These timing requirements establish both a maximum detectable rotation speed and minimum pulse width constraints. If the encoder rotates too quickly or produces pulses that are too narrow, the software may miss transitions or misinterpret the sequence.

Implementing an Efficient Quadrature Encoder

The PIC16F13145 family of microcontrollers (MCUs) from **Microchip Technology** provides an efficient alternative for quadrature signal decoding. These 8-bit processors incorporate a configurable logic block that brings custom hardware capabilities to the microcontroller domain.

The CLB consists of 32 basic logic elements (BLEs), each featuring a four-input lookup table, combinatorial logic, and a flip-flop (**Figure 2**). An interconnect fabric allows signals to be routed flexibly between BLEs, creating a wide variety of logic functions.

This flexible architecture allows the CLB to implement custom logic functions while operating independently of the CPU. Once configured, the CLB can process quadrature signals without requiring constant CPU intervention, enabling the processor to remain in sleep mode and conserve power.

Additional power savings and improved system responsiveness can be achieved during active power states as well. This is because the CPU can handle tasks in parallel while the CLB manages the quadrature encoder functionality, meaning less time is spent in active power mode compared to a software approach, where the CPU would need to rely on interrupts and other software latencies to manage the quadrature encoder and other tasks. The responsiveness benefit is due to the CPU's ability to handle separate tasks simultaneously, while the CLB, with its very low propagation delay, reduces delays in determining direction or speed.

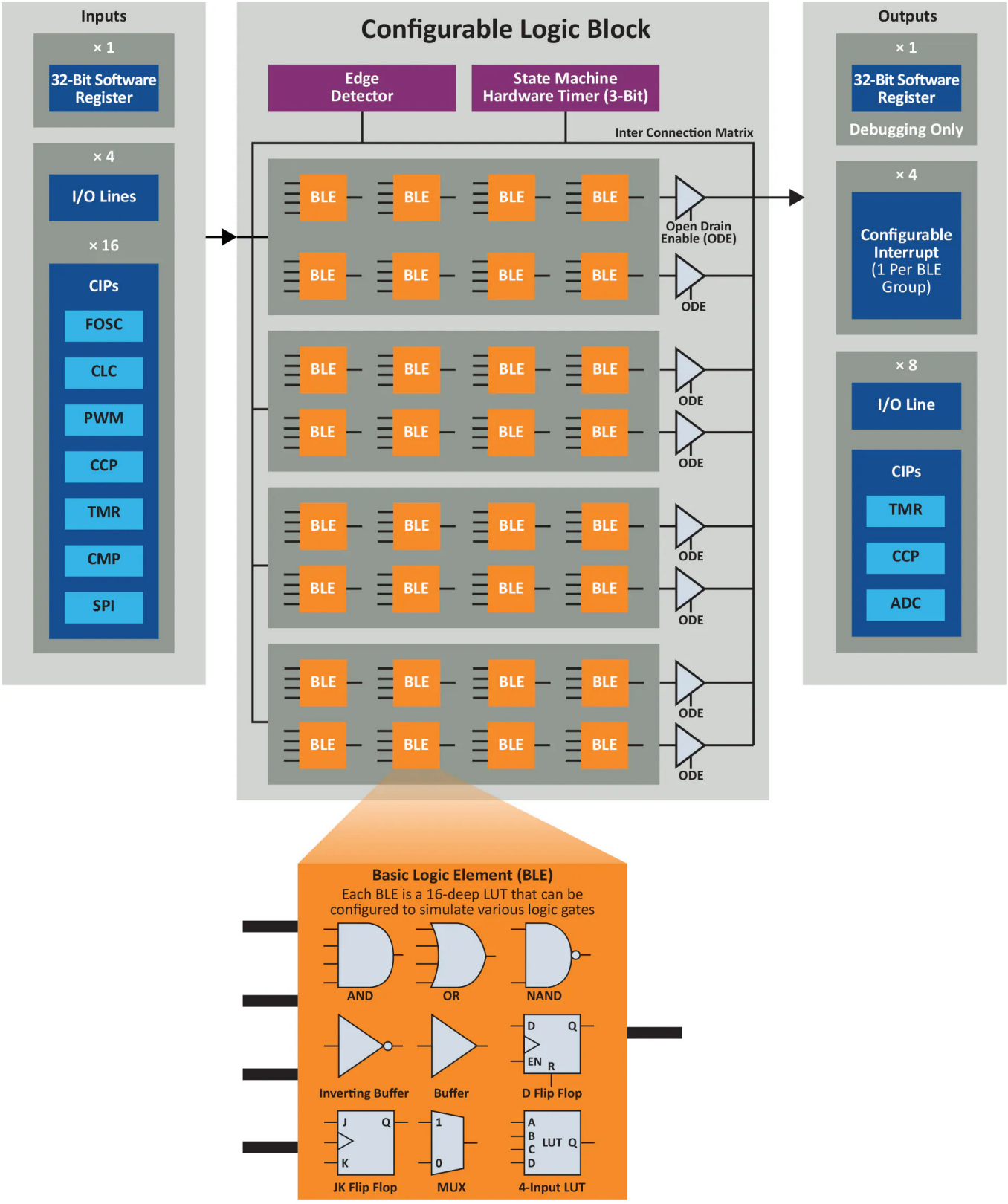


Figure 2: *The PIC16F13145 MCU's CLB contains 32 BLEs, each of which can be used to perform the function of various logic gates. (Source: Microchip Technology)*

Beyond the CLB, the PIC16F13145 family of MCUs includes several peripherals particularly valuable for quadrature encoder applications:

- Multiple 8-bit and 16-bit timers that can be used for position counting
- A configurable interrupt system for real-time responses to position changes
- Communication interfaces like serial peripheral interface (SPI) for sharing encoder data with host systems

Available in 8- to 20-pin packages with a bevy of power-saving features, the PIC16F13145 family of MCUs is well-suited to applications with tight cost, power, and space constraints.

Inside the Quadrature Decoder Architecture

The CLB can be configured to implement both the debouncing and decoding functions needed for quadrature signal processing. **Figure 3** shows a block diagram of the complete quadrature decoder implementation, illustrating how the CLB interfaces with other peripherals to create an end-to-end solution.

The quadrature decoder implementation uses the CLB's BLEs to perform two critical functions: debouncing the input signals and decoding the quadrature sequence. When valid rotation is detected, the decoder outputs a single-clock pulse on one of two dedicated lines—one for clockwise rotation and one for counterclockwise rotation.

Two timer modules count these pulses: TMR0 for clockwise rotations and TMR1 for counterclockwise rotations. The encoder's current position can then be determined by calculating the difference between these timer values. **Figure 4** shows the specific CLB configuration for this implementation.

This timer-based approach provides significant advantages for battery-powered applications. Rather than requiring constant CPU monitoring of the decoder signals, the CPU can remain in sleep mode, waking only periodically to determine the current position. That position information can then be transmitted to a host system via the SPI interface, if needed.

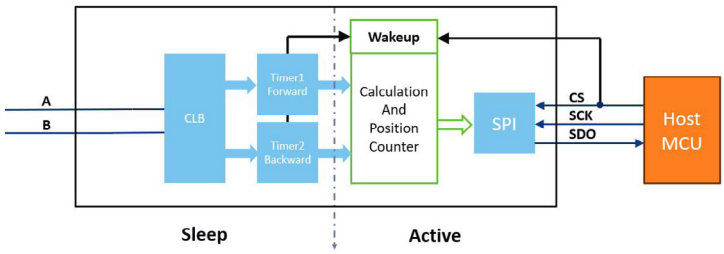


Figure 3: Block diagram showing the quadrature decoder implementation using the CLB and timer peripherals. (Source: Microchip Technology)

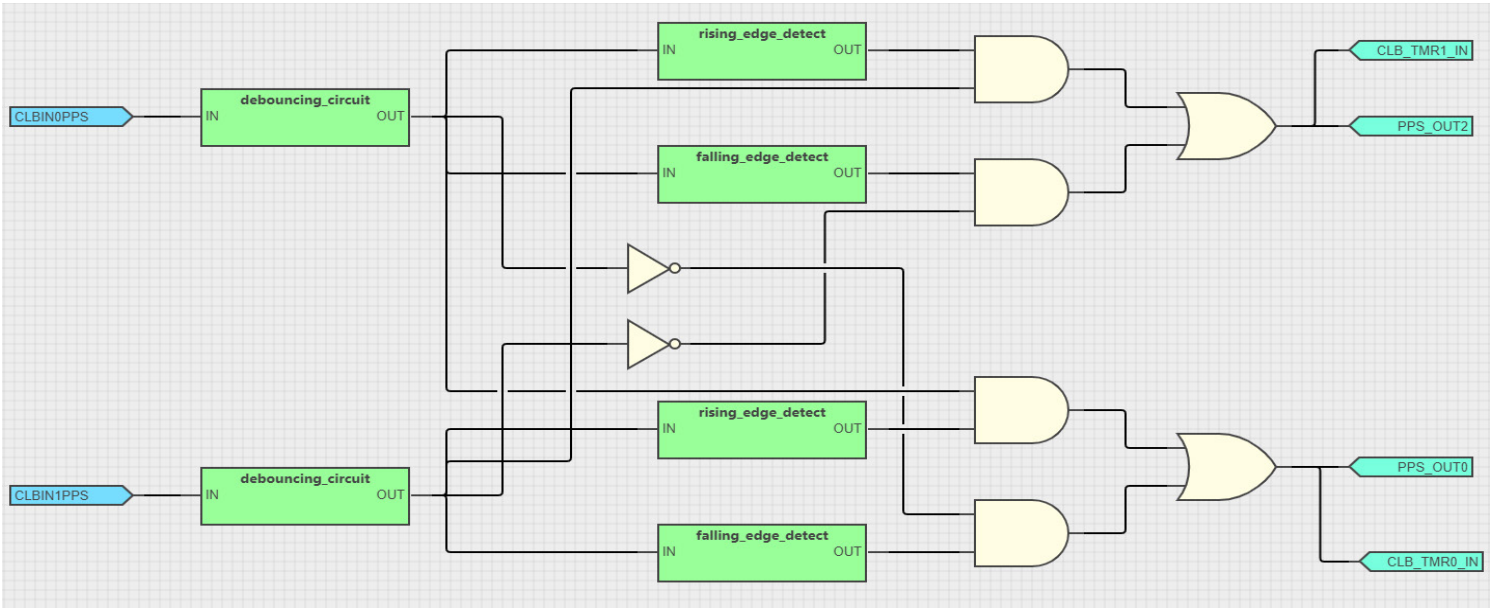


Figure 4: CLB configuration showing the implementation of the quadrature decoder circuit. (Source: Microchip Technology)

This implementation is suitable for user interface applications where periodic position polling is sufficient. Here, polling an input once every 10 milliseconds is usually enough to avoid perceptions of latency.

In motion control applications, real-time accuracy matters more. To handle this, the CLB can be set up to trigger an interrupt whenever the encoder's rotation speed surpasses the predefined limit. This allows the system to respond immediately, which makes it useful in precision motor control, where tracking position changes in real time is important for maintaining smooth and accurate motion.

Regardless of the desired approach, engineers can easily create their designs using the graphical **CLB Synthesizer** tool. This tool is included in Microchip's MPLAB® Code Configurator and is also available as a stand-alone **online utility** (**Figure 5**). The CLB Synthesizer provides a user-friendly interface for configuring the CLB, with example projects and documentation available to help developers get started. And once they get started, the **PIC16F13145 Curiosity Nano Evaluation Kit** offers an integrated debugger to allow for streamlined and hassle-free testing and development.

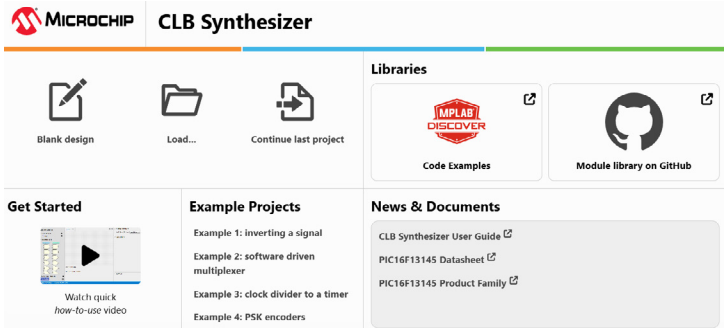


Figure 5: *Microchip Technology's CLB Synthesizer offers users an accessible interface with the right tools to start development. (Source: Microchip Technology)*

Conclusion

The PIC16F13145 family of MCUs from Microchip Technology offers a convenient solution to the challenges of implementing quadrature encoders. Moving decoder logic into configurable hardware eliminates the need for external components while also offering considerably better power efficiency than all-software implementations.

The CLB's flexibility supports everything from simple user interface applications to high-precision motor control systems, allowing developers to use the technology for various use cases.

Lastly, by using MPLAB's graphical design tools, engineers can more efficiently configure and implement this solution, which cuts down development time and system complexity. These advantages make this MCU a practical and flexible choice for a variety of quadrature encoder applications.