

RTEC – A Microcontroller Peripheral Optimized for Real-Time Control of Internal Combustion Engines

Frank Emmett

Automotive Integrated Electronics Corporation

Copyright © 2001 Society of Automotive Engineers, Inc.

ABSTRACT

The Real Time Engine Controller (RTEC) peripheral differs from general-purpose peripherals in that RTEC is designed specifically to control internal combustion engines. Dedicated hardware designed specifically to minimize processor intervention actually increases effective processor throughput, allowing more complex algorithms to be executed by a given processor, without requiring a dedicated peripheral control processor. This dedicated hardware provides precision real-time tracking of the engine and precision fuel metering and ignition for the optimization of engine torque, fuel economy, and exhaust emissions.

This paper compares the RTEC approach to existing general-purpose peripherals based on the criteria of requirements for processor intervention, accuracy, and ease of programming. Typical engine applications such as position tracking, ignition, injection, and variable valve control will be used as a basis for comparison.

INTRODUCTION

Innovations now approaching manufacturability, such as ion sensing and in-cylinder pressure measurement, allow engine control algorithms to operate on a per-cylinder, cycle-by-cycle basis, as opposed to the existing quasi-steady-state control methods currently used. This dramatically increases the throughput requirements on the microprocessor, making it even more critical than ever that smart peripherals be used to perform input processing and control tasks, preserving microprocessor bandwidth for these complex high-level control functions.

Some current approaches to input/output processing for the control of internal combustion engines often take the form of complex, programmable general-purpose timing peripherals. These peripherals are complex to program and require microprocessor intervention.

Other approaches involve the introduction of a second microprocessor into the system that is dedicated to I/O processing, along with comparatively simple timer channels. This approach requires the silicon overhead of additional code and data memory for the second

processor, as well as the development of custom driver software to allow the I/O processor to control the timer channels associated with it. This driver software is often supplied by the silicon vendor, or is developed by application programmers.

The advent of semiconductor processes which enable density suitable for the creation of system on chip (SOC) designs makes it possible to create a dedicated peripheral which is optimized for the control of internal combustion engines and tight integration with a microprocessor. The Real Time Engine Controller (RTEC) is such a peripheral. Its major advantages over other approaches are accuracy, ease of programming, and extremely low microprocessor overhead.

RTEC OVERVIEW

The block diagram of the RTEC is shown in **Figure 5** below. The major components of RTEC are the microprocessor interface, which is responsible for converting the particular microprocessor bus being interfaced to into the internal signals to load and read RTEC registers; the cam/crank processor, which is responsible for tracking the engine position and velocity in real time; the queued data acquisition unit, which is responsible for converting analog signals from the outside world into digital representations for processing; the ignition channels, which are responsible for generating control pulses for the ignition drivers; knock signals generator, which is responsible for generating clocking and integration control signals for knock sensing; the injector channels, which are responsible for generating control pulses for the fuel injector drivers, and the low resolution processor, which is responsible for performing a variety of timing, interrupt, and pulse generation functions.

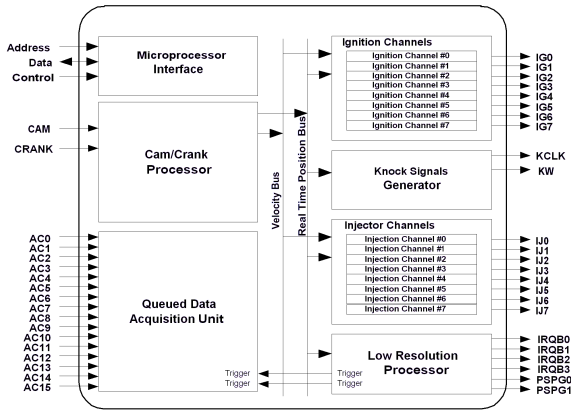


Figure 1 -- RTEC Block Diagram

POSITION TRACKING

In the RTEC peripheral, tracking of engine position is performed by a cam/crank processor (CCP). Conceptually, the CCP is similar in principle to a digital PLL circuit. However, due to the pattern-matching algorithm employed, the CCP is capable of “locking on” to the cam and crank waveforms much more quickly than a digital PLL circuit. With a single-gap crank waveform and a “50/50” cam waveform, the CCP will achieve lock within approximately 360° in the worst case. In the average case, the distance to lock is much shorter, approaching 200°. With more complex cam and crank waveforms, the distance to lock can be significantly shortened beyond these figures.

A block diagram of the cam/crank processor is shown in Figure 2 below.

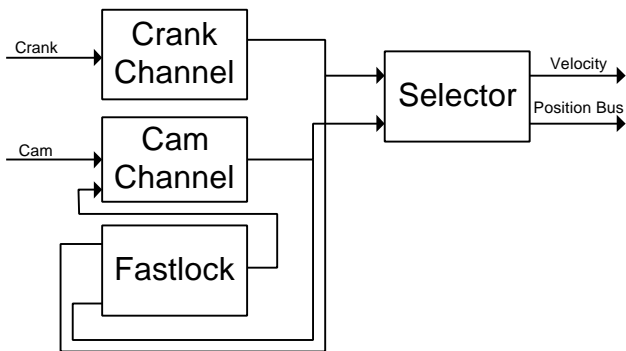


Figure 2 - Cam/Crank Processor Block Diagram

The major components of the CCP are independent tracking channels for the cam and the crank inputs, a “fastlock” circuit which allows the cam channel to take advantage of what is known about crank position in order to more quickly acquire lock on the cam channel, and a selector circuit which allows the engine to be controlled by either the crank channel engine position or the cam channel engine position.

The CCP contains independent synchronization channels for the cam and the crank inputs. Having two independent channels allows the CCP to support

operation if either the crank or cam sensor becomes inoperative, allowing limp-home modes to be programmed. This requires the use of more complex cam waveforms than the standard “50/50” waveform commonly used in current engines. The presence of a full independent channel for the cam enables usage of these more complex waveforms. Each channel is programmed at system initialization with a digitized representation of the ideal waveform, along with parameters to allow the CCP to compensate for distortion effects introduced by changes in engine speed. To account for acceleration and deceleration, the CCP resynchronizes on every tooth. Between teeth, the CCP uses a linear interpolation scheme to estimate current engine position to a resolution of 0.1° at all times. Under constant engine speed, this interpolation is accurate to within one system clock. A block diagram of one of the CCP channels is shown below in Figure 3.

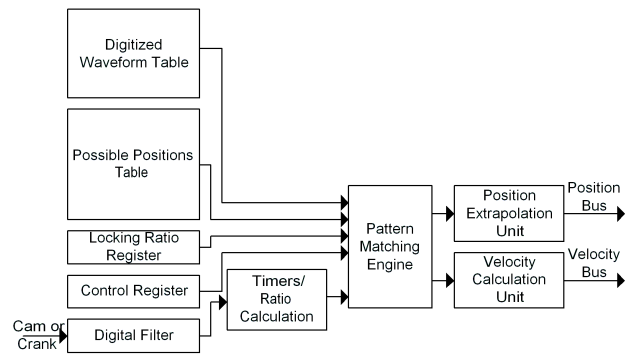


Figure 3 - CCP Channel Block Diagram

Each channel achieves synchronization by first assuming that the camshaft or crankshaft may be at any position. As new edges arrive, the intervals between these edges are accumulated and the ratios between successive time intervals are calculated. These time ratios are compared with the distance ratios stored in the waveform table, and positions at which the distance ratios do not match the time ratios are eliminated, until only one position remains as a possibility. At this point, the position bus is set to this position, the position extrapolation begins counting its way up to the next tooth, the engine velocity is calculated, and the channel declares itself “locked on” to the engine position.

Once the CCP has synchronized to the engine position, it examines each edge on the incoming sensor waveform. If the time ratios between successive edges deviate from the expected by a user-specified amount, as may happen if noise appears on the cam or crank sensor wire, then the CCP begins the synchronization process anew. If so configured, it will also generate an interrupt to the controlling microprocessor, which may choose to take additional action. Also, the positions on the cam and crank channels are compared once per revolution. If the cam and crank positions differ by more than a programmable amount, which would happen if there was noise on one of the input signals or if a timing belt slipped, then an interrupt will be generated to inform the controlling microprocessor of the event, if so configured.

The CCP continuously outputs the current engine on a real-time position bus, which is used as a position reference by all of the other engine-specific RTEC peripherals. The CCP also outputs the engine velocity based on the time between the two most recently received edges on a real-time velocity bus. This velocity indication is used to calculate starting angles for all pulses with a specified duration and end angle. Under normal operation, the CCP will increment the position bus and the velocity bus without further microprocessor interaction required.

Although the channels are capable of operating independently of one another, it is often possible to use coordination between the channels to take advantage of the higher information rate provided by the smaller tooth spacing on the crank waveform as opposed to the cam waveform. A feature known as “early gap detection” uses crank rotation information to eliminate possible cam positions from consideration. As a rough example, if we know that the crank has rotated thirty degrees without an intervening edge detected on the cam input, then we can eliminate all twenty-degree gaps in the cam waveform from consideration as possible positions. This allows the cam channel to “lock on” to the camshaft position more quickly than otherwise possible and also allows the cam channel to use looser locking ratios. This is beneficial because the distortion introduced by acceleration on the wider tooth spacings of the cam waveform is much larger, for a given velocity and acceleration, than on the comparatively smaller tooth spacings of the crank waveform.

Another feature that makes use of coordinated information between the cam and crank channels to achieve faster locking time is the “fastlock” circuit. Most crank waveforms are structured with a large number of equally spaced high data rate teeth and one or two gaps, which are much larger. Therefore, as soon as a gap is encountered, the engine position can be determined to be either at the position of the gap or at that position offset by 360°. When this occurs, the fastlock circuit, if enabled, compares the two possible crank positions to the current possible cam positions. If one and only one of the possible cam positions is within a programmable distance of one and only one of the crank positions, then the fastlock circuit can conclude that this cam position must be the correct one, and can cause the cam channel to immediately “lock on” to the cam waveform at this position. This feature also enables the cam/crank processor to achieve “full synchronization” (both cam and crank channels “locked on” to their respective waveforms) much more quickly than is otherwise possible.

INITIALIZATION – The cam/crank processor initialization routine is as follows:

- Program the cam and crank waveforms, which have been digitized at one degree resolution, into the cam and crank waveform tables

- Program mode bits to enable the appropriate value into the digital filters for cam and crank input signals to eliminate noise pulses
- Program the edge desired into the control registers for the cam and crank channels. The search algorithm can be triggered off the rising edge of the input signal, the falling edge, or both edges.
- Program the locking ratio desired for the cam and crank channels into the appropriate channel's locking ratio register. The locking ratio is a number that controls the tolerance to acceleration and deceleration.
- If desired, enable early gap detection and fast locking.
- Enable the cam and crank channels

OPERATION – Once the engine is running, the only routines that are required are interrupt service routines to handle a situation where either the cam or crank channels lose lock on the incoming waveforms, in case special action needs to be taken. If no action is taken, the cam/crank processor will automatically relock on the waveforms at the earliest opportunity.

IGNITION SUBSYSTEM

The ignition channels in the RTEC generate output pulses for the ignition coil drivers. A block diagram of the ignition subsystem is shown below in Figure 4 below.

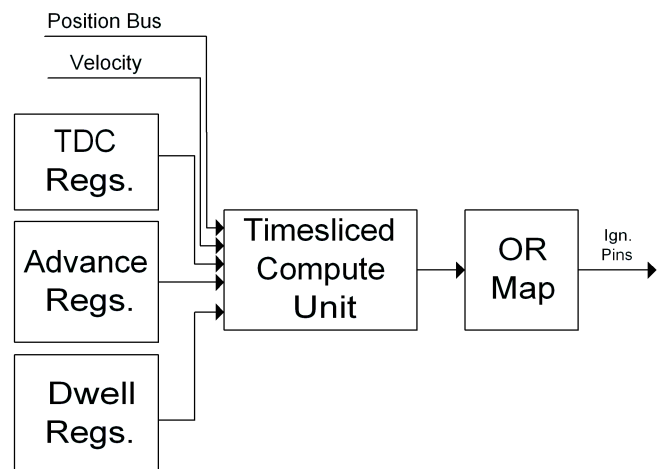


Figure 4 - Ignition Block Diagram

The user needs only program a desired advance angle and a desired dwell time for a given channel. The channel will then produce ignition pulses of the programmed width and duration without microprocessor intervention until the application software changes advance or dwell parameters. Current incarnations of

RTEC have up to ten ignition channels, which allow support of up to ten cylinder engines. A flexible wired-OR channel-to-pin assignment mechanism allows support of virtually any ignition strategy, from single-coil to coil-on-plug.



Figure 5 - Ignition Waveform

INITIALIZATION – The initialization routine for the ignition subsystem is as follows. This example assumes a one-to-one mapping between ignition channels and engine cylinders

- Program the ignition OR map, assigning each ignition channel to the cylinder with the same number
- Program the top dead center (TDC) registers with the absolute angle of the TDC for each cylinder
- Program the enable bits for each channel

OPERATION – While the engine is running, the update routine is as follows. Note that this can be executed at any time, and even multiple times during an engine revolution.

- If it needs to be changed, write the desired advance angle to the advance angle register.
- If it needs to be changed, write the desired dwell time to the dwell time register.

For conservation of silicon area, the ignition subsystem is generally implemented as a timesliced operation, with one channel being evaluated every fifty system clocks (or every 2µs, assuming a system clock frequency of 25 MHz).

KNOCK SENSOR SIGNAL GENERATION

The RTEC contains a knock sensor signal generation circuit to generate two output signals for the control of an external knock sensor. A block diagram of this circuit is shown below in Figure 6.

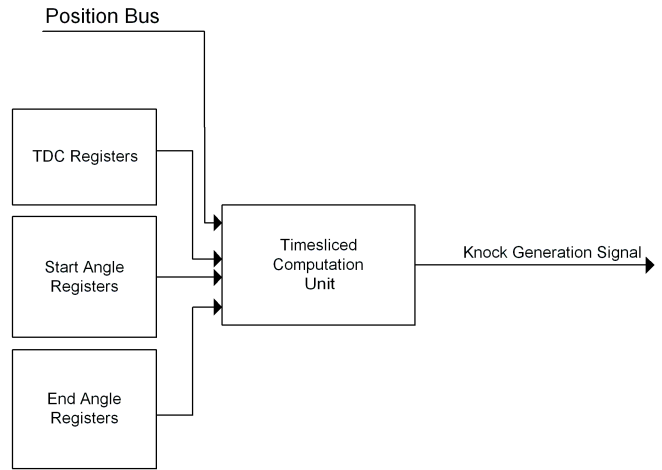


Figure 6 - Knock Signal Generation Block Diagram

The first signal, KCLK, is a clock signal of programmable frequency for use by external knock sensors that use switched capacitor technology. The second signal, knock window (KW), indicates knock analysis window for use by the external knock sensor. The KW signal is enabled by programming a start and stop angle. The values of these angles are determined relative to the top dead center (TDC) angle of each cylinder. Figure 7 below show an example of a knock analysis window signal. Once these angles are programmed, knock window generation continues indefinitely without further microprocessor interaction required.

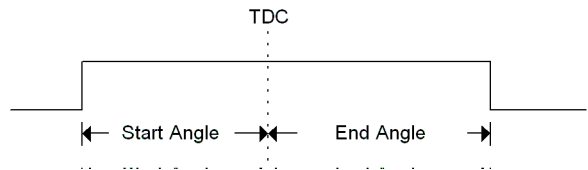


Figure 7 - Knock Signal Waveform

INJECTION SUBSYSTEM

The injection channels in the RTEC generate output pulses for the fuel injector drivers. A block diagram of the injection system is shown below in Figure 8.

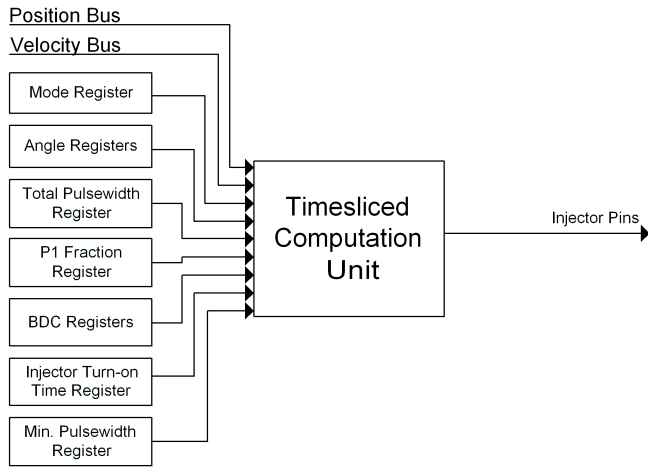


Figure 8 -- Injection Block Diagram

Injection channels in the RTEC operate similarly to the ignition channels. The injection system in RTEC currently supports the generation of two pulses per engine cycle, although more pulses are possible. Each pulse is configurable to begin at a specified angle and end after a specified time has elapsed or to end at a specified angle, after being active for a specified time. The injection subsystem is designed so that the angles and/or times may be programmed at any time, allowing the determination and updating of these angles in the RTEC to be done completely asynchronous to engine operation. The basic injection subsystem supports the generation of two fuel pulses, either of which may be programmed in angle/time (start at a specified angle and run for a specified time) or time/angle (run for a specified time and then end at a specified angle) mode. The four possible modes of injector pulse generation are shown in the figures 9 through 12.

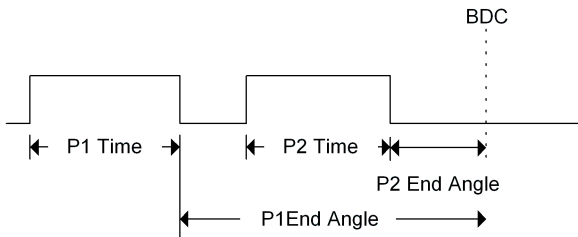


Figure 9 - Injection Waveform (P1 Time-Angle, P2 Time-Angle)

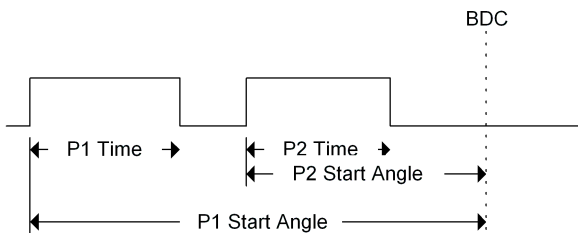


Figure 10 - Injection Waveform (P1 Angle-Time, P2 Angle-Time)

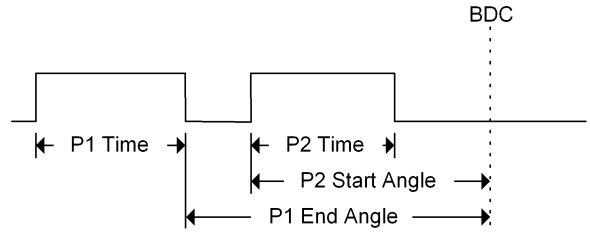


Figure 11 - Injection Waveform (P1 Time-Angle, P2 Angle-Time)

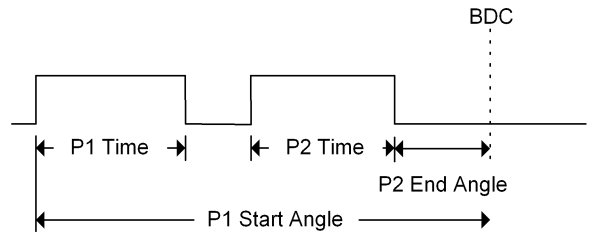


Figure 12 - Injection Waveform (P1 Angle-Time, P2 Time-Angle)

INITIALIZATION – the injection subsystem initialization routine is as follows:

- Program the absolute bottom dead center (BDC) angle for each cylinder into the corresponding BDC register
- Set a bit for each of the two pulses indicating whether each pulse should run in time/angle or angle/time mode
- Set the starting and ending angles for the two pulses
- Program the injector turn-on time register. This time will be automatically added to each fuel pulse (only once if the pulses merge together), to account for the time from the turn-on of the injector to the time when it begins delivering fuel
- Program the minimum pulse width register. To account for the fact that injectors do not function properly when extremely short enable pulses are delivered, pulses that are smaller than the value in this register will not be delivered
- Set the fraction of commanded fuel to be delivered in the first pulse (the remainder will be delivered in the second pulse)
- Set enable bits for the channels

OPERATION – While the engine is running, the update routine is as follows. Note that this can be executed at any time, and even multiple times during an engine revolution.

- If it needs to be changed, write the total fuel to be delivered into the total pulse width register.
- If it needs to be changed, write a new value to the injector turn-on time register.

For conservation of silicon area, the ignition subsystem is generally implemented as a timesliced operation, with one channel being evaluated every 128 system clocks (or every 5.12 μ s, assuming a system clock frequency of 25 MHz).

LOW RESOLUTION PROCESSOR

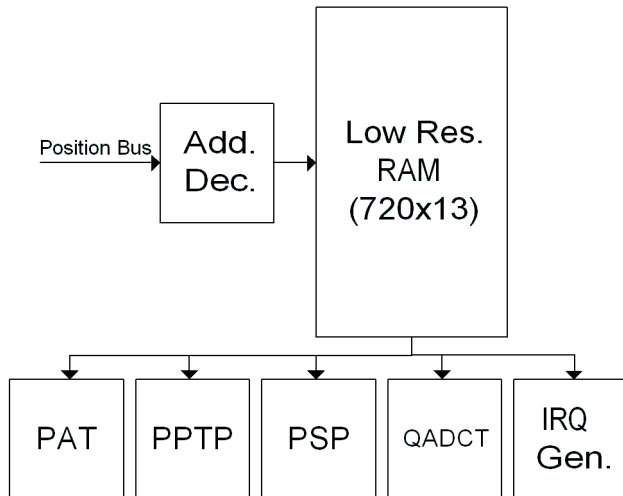


Figure 13 - Low Resolution Processor

The low resolution processor (LRP) is so named because of its ability to perform certain operations at 1° intervals as opposed to the relatively high-resolution 0.1° intervals of other RTEC functions. The LRP can be thought of as a 720 deep RAM by 13 bits wide. Each RAM location corresponds to an angular position of the engine (i.e., RAM location 0 corresponds to engine angular position 0°, RAM location 1 corresponds to engine angular position 1°, and so on). The real time position bus acts as an address pointer into this RAM. The LRP contains the following control functions:

PRECISION ACCUMULATED TIME FUNCTION – The precision accumulated time (PAT) function measures the time during multiple programmable intervals of the engine rotation to an accuracy as small as one system clock (40 ns with a 25 MHz clock). The time over a given interval can be accumulated into one of three accumulators, which can then all be transferred to the microprocessor at a given engine position for evaluation. This function is useful for detecting misfire conditions or any other application that involves detecting changes in engine speed across angular intervals during an engine revolution.

PRECISION POINT-TO-POINT TIMING FUNCTION – The precision point-to-point (PPTP) timing function is a single timer which measures the time between specified

positions of the engine rotation to an accuracy as small as one system clock. This function is useful for measuring engine speed (RPM), time between cylinder firings, or can be used for other applications that require precision time measurements

POSITION SYNCHRONIZED PULSE GENERATION – The position synchronized pulse (PSP) generation function generates pulses that start at a given engine position and last for a programmable number of degrees. These pulses have programmable polarity and may be used for communicating engine speed to transmission modules or for triggering external hardware modules based on engine position. The PSP function is programmed by setting bits in the low resolution table corresponding to the angular positions at which output pulses are desired. By default, these pulses are one degree wide. To generate longer pulses, merely set several bits in a row; e.g., to generate a pulse that goes high when the engine reaches the 10 degree position and returns low when the engine reaches the 15 degree position, set the PSP bit in locations 10 through 14 of the low resolution table. RTEC has two independent PSP channels. Possible uses for the PSP circuit include the generation of engine speed information for a transmission module, or generating position-based trigger signals for other external circuitry.

QUEUED DATA ACQUISITION TRIGGER – The queued data acquisition trigger (QDAT) function commences execution of either of the two position-based queues in the queued data acquisition unit (please see the section entitled “QUEUED DATA ACQUISITION” for details). There are two separate QDAT bits, corresponding to each of the two position-based queues in the QDA module. Setting a bit at a particular position triggers execution of the commands in the appropriate position-based queue starting at the location of the current queue pointer. In this manner, acquisitions can be triggered at user-specified positions, for example, once per revolution or once per cylinder.

INTERRUPT REQUEST GENERATION – The interrupt request generation (IRQ Gen.) function generates interrupt requests to the controlling microprocessor at programmable engine angular positions. Multiple bits allow the encoding of up to fifteen different interrupt types, allowing low-latency determination of gross engine position by the interrupt service routine, or the generation of four separate interrupt signals, which allows the microprocessor to directly vector to the appropriate interrupt service routine for a given engine position for the minimum latency.

QUEUED DATA ACQUISITION

The Queued Data Acquisition unit (QDA) contains a 10-bit successive approximation analog to digital converter with 16 analog channels. The QDA provides several mechanisms for automatically converting one or more analog signals into its digital result and storing the data within 1 of the 64 result registers. Important features of

the QDA are the multiple triggering sources, multiple queues/sub-queues, and the programmable channel control and opcode registers. Figure 4-1 shows the architectural organization of the QDA block with the main signal flow indicated.

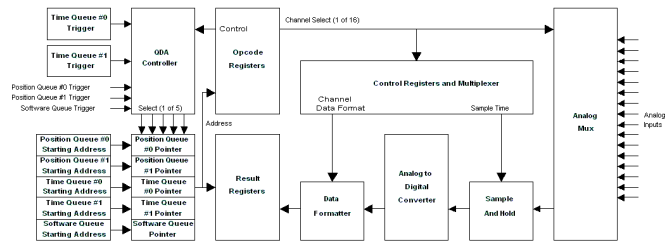


Figure 14 - Queued Data Acquisition Block Diagram

Analog to digital conversions can be triggered in three different ways. A conversion can be triggered at any given time by use of the software queue. This is useful for analog signals that need to be monitored infrequently. A conversion can be triggered at specific engine positions (as indicated by the position bus) by use of one of the two position queues. It is convenient to use one of the position queues for conversions that need to be performed per revolution and the second position queue for conversions that need to be performed per cylinder, for example, in-cylinder pressure sensing. A conversion can be triggered at specific time intervals by use of one of the two time-based queues. Since the two time-based queues can be programmed with separate time bases, it is convenient to use one queue for conversions that need to be performed once in a longer time interval and the second queue for conversions that need to be performed more often.

Any conversion may trigger an interrupt when completed, or the software can read the most recent conversion results at any time from the result registers. Each channel can be programmed with a different data format and conversion time. In case multiple queues request a conversion at the same time, an arbiter circuit grants access to the conversion hardware first to the queue with the highest priority. The priorities for all queues are user-programmable.

The flexibility of the time and position-based QDA block allows for measurements of analog signals to be taken as close as possible to the time they are used in calculations of engine control parameters, removing delay in the control loop for quasi-steady-state control algorithms, reducing one source of error, and enabling the creation of per-combustion event control strategies.

COMPARISON AND BENEFITS OVER THE CURRENT STATE OF THE ART

RESOLUTION – Resolution of the various signal generation components of RTEC (ignition, injection, and knock sensor signal generation) are on the order of several microseconds (these are programmable in some cases). These resolutions are deterministic; i.e., they are unaffected by how many functions are running at a given

time, as all hardware functions are independent of one another, as compared to microcoded timer units in which a single sequencer engine services many channels, with considerable latencies on the order of ten to fifteen system clocks between servicing of different channels. The dedicated hardware approach also has advantages over a general purpose timer unit which requires microprocessor interrupts to be serviced in order to change modes or update parameters, which can result in inaccuracies if the microprocessor is unable to service interrupts in the required time. The resolution of all angle-based functions is 0.1° . This ensures that control signals, such as ignition and injection pulses, are delivered with maximum accuracy regardless of the microprocessor load.

PROCESSOR INTERVENTION – The RTEC is capable of running an engine in steady-state open loop conditions even if the controlling microprocessor is halted. The only time that the microprocessor needs to access the RTEC peripheral is to change the control parameters of engine operation; this can be done at any time, independent of engine position, removing any constraints on minimum run time of the control algorithm. The result is that microprocessor bandwidth can be devoted to the tasks of running high-level control algorithms which focus on performance and emissions issues, rather than low-level control issues.

EASE OF PROGRAMMING -- Since algorithms are part of the hardware implementation, drivers consist mainly of scaling control variables and writing them to the appropriate RTEC registers. RTEC requires no specialized tools or programming environment other than a standard C or C++ compiler and/or an assembler. Compared to the use of general-purpose or microcoded timer units, the time and cost required for the development of low-level drivers is substantially decreased.

CONCLUSION

The RTEC peripheral combines a high degree of accuracy and the ultimate in real-time responsiveness with ease of programming. The RTEC peripheral, being designed specifically to control the operation of internal combustion engines, is free from obligations faced by more general-purpose timer units of having to be capable of controlling stepper motors, enabling this unique combination of high capability and low complexity.

The precision of generated control signals, along with the low microprocessor overhead required by the circuit, allow a microprocessor of a given throughput to execute more complex algorithms. The ease of programming allows software development effort and cost to be concentrated on the high-level control algorithms that provide high value to the end product. Both of these benefits contribute to achieving the end goals of higher performance, better fuel economy, and lower emissions. The exact quantity of improvement in these areas is highly dependent on the high-level control software itself.

The Real Time Engine Controller has been implemented with Motorola 68K, Motorola PowerPC, ARM7 and ARM9 from ARM Ltd., and STMicroelectronics ST10 / Infineon SAB16x microprocessors.

REFERENCES

1. RTEC Reference Manual, Automotive Integrated Electronics Corp., 2000.

CONTACT

Frank Emnett Automotive Integrated Electronics Corp.
Design Engineer
9034 N. 23rd Ave., Suite 13, Phoenix, AZ 85331
Tel: 602-943-7499 Fax: 602-861-1777
Email: frank.emnett@aiec.com
Web: <http://www.aiec.com>

DEFINITIONS, ACRONYMS, ABBREVIATIONS

BDC: Bottom Dead Center
CCP: Cam/Crank Processor
IRQ: Interrupt Request
KCLK: Knock Clock signal
KW: Knock Window signal
LRP: Low Resolution Processor
PAT: Precision Accumulated Time
PLL: Phase Locked Loop
PPTP: Precision Point-To-Point
PSP: Position Synchronized Pulse
QDA: Queued Data Acquisition
RPM: Revolutions Per Minute
RTEC: Real Time Engine Controller
SOC: System On Chip
TDC: Top Dead Center