AN1980

Capacitor Discharge Ignition Using the Angular Timer

Authors: *Ashutosh Tiwari,*
Shailendra Vengurlekar,
Namrata Dalvi,
Swathi Sridhar

Microchip Technology Inc.

INTRODUCTION

The Capacitor Discharge Ignition (CDI) system is an electronic ignition system used in internal combustion engines. An ignition system provides a high-voltage spark in the engine's cylinders to ignite the air-fuel mixture. The CDI system uses high-voltage capacitor discharge current output to fire the spark plug.

This application note briefly explains the implementation of CDI for single Profile Ignition Pickup (PIP) systems using standard peripherals on an 8-bit PIC® microcontroller and the challenges associated with such designs. It also covers the implementation of the CDI system using the following advanced Core Independent Peripherals (CIP) of PIC microcontrollers:

- Angular Timer (AT)
- Signal Measurement Timer (SMT)
- Math Accelerator
- Configurable Logic Cell (CLC)

The merits of using these peripherals, by demonstrating how the challenges of the conventional design are overcome, are also explained. Furthermore, this application note provides information on the improvements realized using CIPs in terms of performance of the CDI system and reduced usage of the CPU.

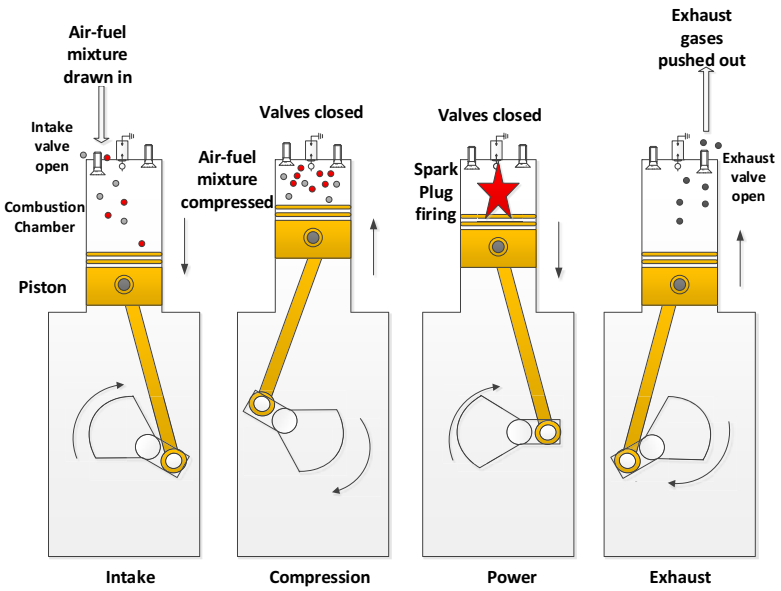
Lastly, this document offers information on multiple PIP systems, mathematical formats and certain useful tips and tricks for improving mathematical computations.

INTERNAL COMBUSTION ENGINE

The internal combustion engine is a basic building block of an automobile, which converts the chemical and heat energy into mechanical energy. The air and fuel chemical mixture is burnt and extreme heat is generated, expanding the exhaust gases which force the cylinder piston to move, causing the camshaft to rotate and create kinetic energy. This kinetic energy is coupled to the vehicle's wheels by gear trains in order to convert the angular motion into linear motion.

A four-stroke cycle engine that utilizes four distinct piston strokes, intake, compression, power and exhaust to complete one operating cycle is shown in Figure 1.

FIGURE 1: FOUR-STROKE ENGINE WITH ENGINE PARTS



To burn the air and fuel mixture in the cylinder, a high-voltage spark is generated by passing electrical current through a spark plug during the power phase. This ignites the air-fuel mixture, producing a pressure wave which forces the piston down. The momentum of the crank shaft caused by the power stroke of the pistons continually moves the engine through the four strokes.

Top dead center (TDC) is the highest position of the piston near the spark plug and bottom dead center (BDC) is the lowest position near the camshaft. After the spark plug fires into the power stroke, the air-fuel mixture needs some time to completely burn; this burning process is progressive in nature (i.e., the mixture at the top burns first and quickly moves towards the bottom). Therefore, to completely burn the air-fuel mixture and produce the maximum pressure wave, the spark plug should be precisely fired moments before the piston reaches TDC and at the proper angle, which is determined by the engine piston speed. There are other factors, such as temperature and throttle position (i.e., amount of air-fuel mixture) which also determine the spark firing angle. To fire the spark correctly and accurately, a separate module known as the ignition control mechanism, is used. There are two types of ignition systems:

- Inductive Discharge Ignition (IDI) or Transistor Controlled Ignition (TCI)
- Capacitor Discharge Ignition (CDI)

In a Transistor Controlled Ignition (TCI) system, the spark for igniting the air-fuel mixture is produced by building up the charge in the primary winding of an ignition coil and releasing it. The Capacitive Discharge Ignition (CDI) system produces the spark by discharging a capacitor into the ignition coil at the right time.

The CDI system is characterized by a short and accurate spark which makes it suitable for high RPMs. The TCI system provides a longer spark duration that ensures complete combustion, but takes time to charge the ignition coil, called dwell time, which makes it less efficient at higher RPMs. For more information on TCI, refer to application note AN2095 – *Transistor Coil Ignition with Integrated Remote Keyless Entry and Immobilizer Using PIC® Microcontrollers*."

This application note will focus on the CDI system.

THE CAPACITOR DISCHARGE IGNITION (CDI) SYSTEM

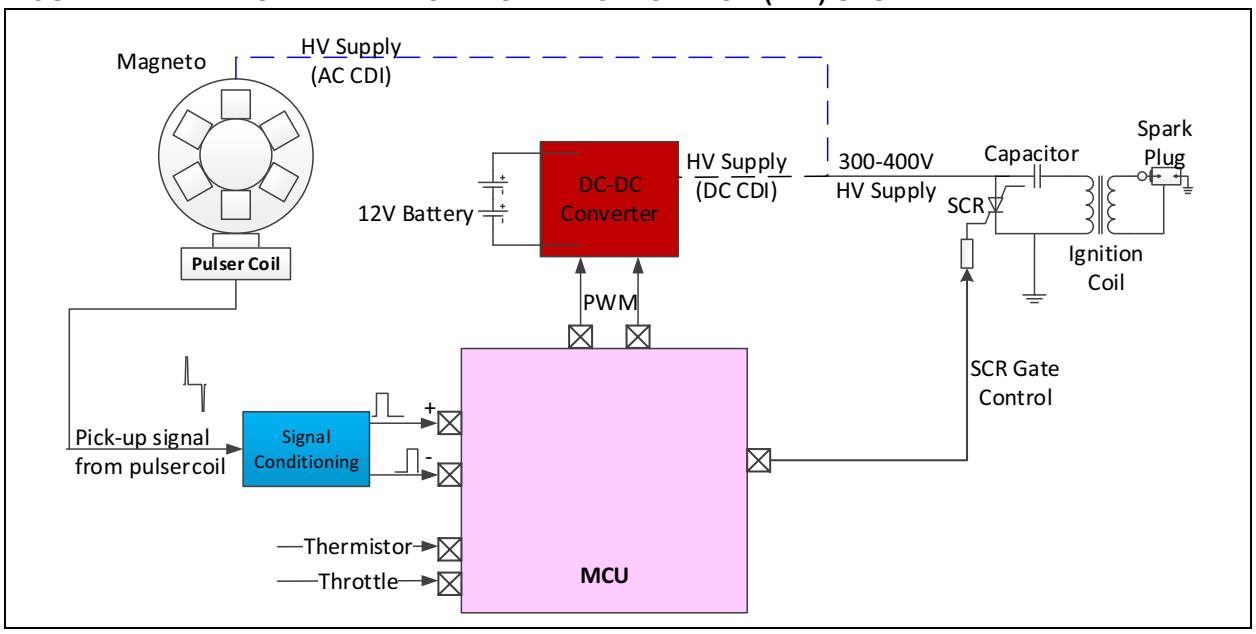
The CDI system uses the current produced by a capacitor discharge to fire the spark plugs. Figure 2 shows a basic CDI system.

The following section explains the basic blocks of a CDI system.

The parts that make up a CDI system are listed below:

- High-Voltage Supply
- Capacitor
- Ignition Coil and High-Power Switch
- Pulser Coil
- Signal Conditioning
- Microcontroller

FIGURE 2: BASIC CAPACITOR DISCHARGE IGNITION (CDI) SYSTEM



High-Voltage Supply

There are two types of CDI systems:

- Alternating Current Capacitor Discharge Ignition (AC-CDI)
- Direct Current Capacitor Discharge Ignition (DC-CDI)

In an AC-CDI system, an alternator or stator (magneto) generates enough power for all electronic systems including the CDI. The capacitor is charged through rectified output of magneto AC supply, which is 200V DC to 400V DC. When the engine is cold (not running), a kick-start is required to rotate both the engine and the magneto. This does not generate sufficient power from the magneto to completely charge the capacitor for a high-voltage spark. For very low RPM, the firing angle is always constant. Hence, analog firing is used to fire at negative PIP output from the magneto-flywheel pulser coil without calculating the RPM.

In a DC-CDI system, a constant 12V DC power is always available from the battery. It requires an additional DC/DC converter circuit to raise the 12V DC to 200-400V DC. This additional circuitry makes the CDI module slightly larger than an AC-CDI system. When the engine is not running, it can be started easily at a precisely calculated firing angle, as the input DC supply is always available.

Capacitor

To create the high-voltage spark in the spark plug, a high-voltage capacitor with a high-charge capacity is charged using either the output of the DC/DC converter (DC-CDI) or using the output of the magneto, an AC alternator (AC-CDI). The capacitor is charged to a high-voltage supply, usually 200V to 400V.

Ignition Coil and High-Power Switch

The capacitor is connected to an ignition coil or step-up pulse transformer which produces a very high voltage, in the range of 40 kV or more.

The switch is used to connect the capacitor to the primary of the ignition coil. The switch is fired when the microcontroller gives a pulse at the gate of the switch. The sudden rush of current in the primary of the coil produces a very high voltage in the secondary, which generates the spark to ignite the air-fuel mixture. Thus, the microcontroller controls the firing angle of the switch for generating the spark.

A Silicon-Controlled Rectifier (SCR) is most commonly used as a high-power switch in CDI. It is highly durable due to the higher operating voltages and current ranges with moderate frequency response. The disadvantage of the SCR is that it is a one-sided switch (i.e., the switch can only be switched ON at desired time). The SCR automatically goes to the OFF state when the anode current falls below the holding current of the SCR.

Insulated Gate Bipolar Transistors (IGBT) and Metal-Oxide Semiconductor Field Effect Transistors (MOSFET) can also be used as high-power switches in CDI systems instead of SCR.

Pulser Coil

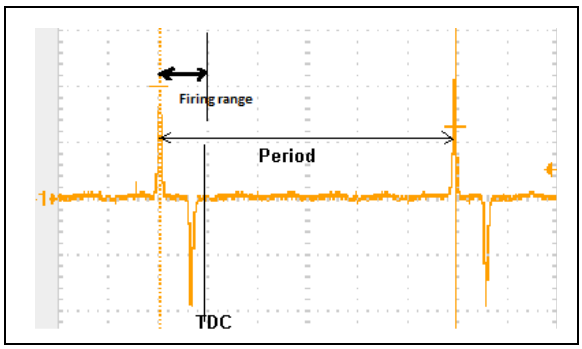
The pulser coil or pick-up/timing coil is responsible for providing the timing signal to the ignition control system. A magnet is mounted on a flywheel. The flywheel is mounted on the magneto shaft. When the flywheel rotates, the magnet passes near the pulse coil producing a timing pulse. There is one pulse per pole. Hence, for each magnet there are two outputs, one positive pulse followed by a negative pulse, generating one alternating pulse pair. For a single PIP system, there is only one pair. For multi-pulse systems there are multiple pulse pairs, based on the number of magnets on the flywheel. The alternating pulses are at a fixed angle with respect to the TDC piston position in the engine for each rotation. The period of the pulses triggers the rotation of the engine.

Based on the number of alternating pulses from the pick-up, per engine rotation, the pulser coil system can be divided into the following types:

- Single PIP System
- Multiple PIP System

Note: Hall effect sensors are also used for providing timing pulses to the CDI system. When a magnet moves in close proximity of the Hall effect sensor, the sensor's output voltage jumps to maximum voltage, indicating the presence of a magnetic field. When the magnetic field is removed, the voltage abruptly changes to zero. To produce an output signal, a Hall effect sensor must be supplied with a reference voltage, which is taken from the CDI system. This produces a square wave output that can be used for timing purposes

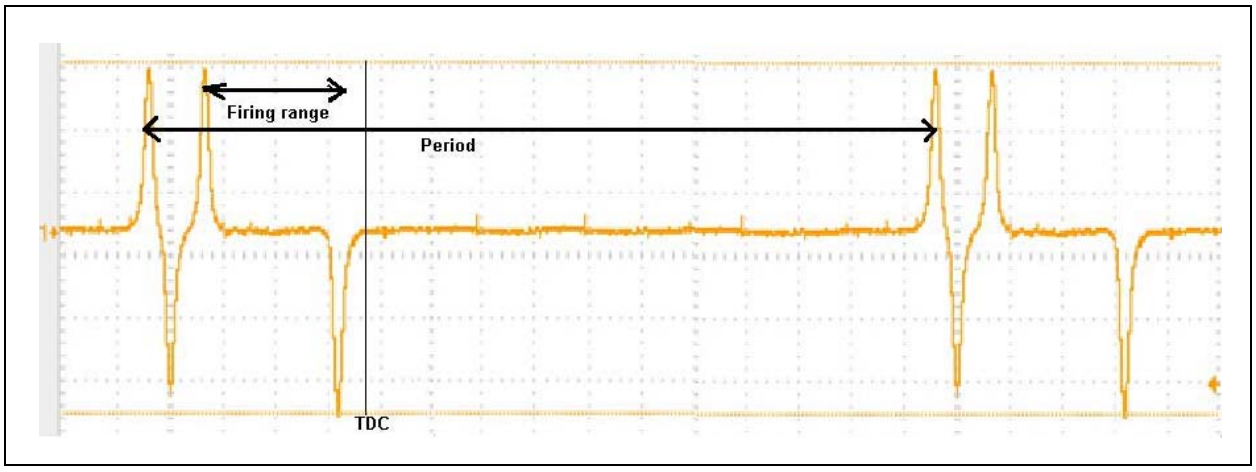
FIGURE 3: PICK-UP SIGNAL IN THE SINGLE PIP SYSTEM



SINGLE PIP SYSTEM

In a single PIP system, the pulser coil produces alternate positive and negative pulses, as shown in Figure 3. The angular distance between the positive reference pulse and the negative reference pulse is called the PIP length. This waveform refers to the Top Dead Centre (TDC) position of the piston. The angle between the pulses is fixed and this period is useful for determining the RPM. The distance between the negative reference pulse and the occurrence of the TDC is also fixed. The negative pulse is used as a reference for firing the pulses (i.e., the angular distance from this position is used for determining the firing angle). With this position and RPM information and also with knowledge of parameters like engine temperature and throttle position, the firing angle is determined.

FIGURE 4: PICK-UP SIGNAL IN THE DUAL PIP SYSTEM



MULTIPLE PIP SYSTEM

In a multiple PIP system the pulser coil provides more than one alternating pulse. The pick-up signal for the dual PIP system is shown in Figure 4, along with the TDC and the firing range. The second negative pulse is the reference point before TDC. This is the minimum firing angle at which the spark should be generated for engine speeds below idle speed.

The first positive and negative pulses can be used for calculating the RPM of the engine. The second positive pulse can be used as a reference point for deciding the firing angle for higher speeds.

Signal Conditioning

The pulser coil generates the timing signals that contain both positive and negative pulses. These pulses are in the range of ±3V to ±90V, depending on the magnetic field strength of the magnet mounted on the flywheel. A signal conditioning circuit is used to invert the negative pulse and limit the pulses to a range of 0V to 5V. It is also used to filter any spurious noise. The signal conditioning circuit will provide two positive outputs, one corresponding to positive pulses and another for negative pulses. Output of the signal conditioning circuit is connected to the microcontroller.

Microcontroller

In a digital CDI system the microcontroller has two major functions:

- Deciding the advance firing angle by reading input from the sensors, such as the pulser coil, thermistor, and throttle position, then producing the firing pulse.
- Setting the duty cycle of pulse-width modulation (PWM) for the DC to DC converter.

The advance angle required for achieving optimum performance of the engine is mainly dependent on the RPM. Hence, the system must be aware of parameters such as RPM, temperature and throttle position. Spark plug firing timing information, referred to as timing maps, is stored as look-up tables in the microcontroller. These tables provide the appropriate advance angles with respect to the RPM. Multiple timing maps are stored based on different combinations of throttle positions and temperature ranges. After the controller computes the RPM, it can then fetch the advance angle information from the look-up table.

The following sections of the application note will explain two different implementations of CDI on PIC® microcontrollers: CDI Implementation using Timer, ADC, capture and compare modules, which is the conventional method and CDI implementation using AT. Challenges that are observed in the CCP method are explained and how these challenges can be simplified using Angular Timer is explained in the Comparison Between CDI Using the AT, and CDI Using the Conventional Approach section.

CDI IMPLEMENTATION USING TIMER, CAPTURE AND COMPARE MODULES (CONVENTIONAL METHOD)

Peripherals on PIC MCUs, such as Capture Compare or PWM (CCP) and the ADC, together with the interrupt pin INT, are used to determine the firing angle control in CDI, as explained in the following section.

Capture/Compare/PWM (CCP) Capture Mode

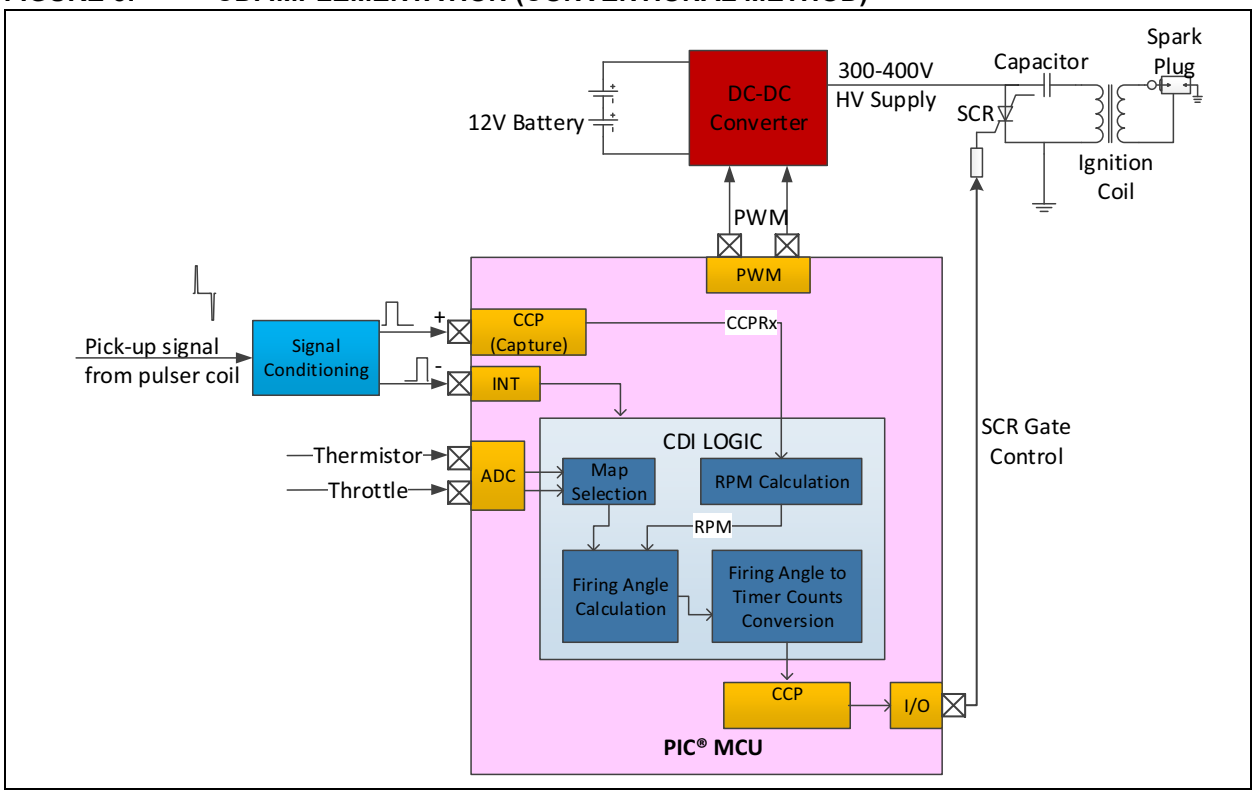
Output for the positive PIP signal of the signal conditioning circuit is given to the capture module. The capture module measures the time between two positive pulses (the period of the pulser coil output). The period of the pulses gives the RPM of the engine.

Analog-to-Digital Converter (ADC)

An ADC is used to determine the engine temperature and the throttle position (if it is analog). Throttle position input can be either analog or digital. In the case of a digital throttle, the position for the wide open throttle (WOT) is one state and the partially open throttle (POT) is another. There are distinct firing maps for different throttle positions and different temperature ranges.

Figure 5 shows the conventional method for the CDI implementation.

FIGURE 5: CDI IMPLEMENTATION (CONVENTIONAL METHOD)



CDI Logic

RPM CALCULATION

The RPM of the engine can be calculated from the external signal frequency f_{signal} as shown in Equation 1:

EQUATION 1: RPM CALCULATION USING TIMER1

$$f_{signal} = \frac{Timer1\ CLK}{Timer1_{Prescaler} \times CCPRx}$$

$$RPM_{Engine} = \frac{Timer1\ CLK}{Timer1_{Prescaler} \times CCPRx} \times 60$$

Where: Timer1 CLK = Clock input for Timer1 using T1CON register = Fosc/4 = 32/4 = 8 MHz
Timer 1Prescalar = Timer1 clock prescaler selected in T1CON register = 1:8 = 8
CCPRx is the captured value of Timer1 at falling edge of f_{signal}
f_{signal} is input signal frequency of the pick-up signal
60 scalar value is multiplied to convert Hz to RPM, RPMEngine = f_{signal} * 60

FIRING MAP SELECTION

Every engine is associated with a firing map which shows the relationship between input engine speeds in RPM and the firing angle of spark in degrees. These maps vary depending on the throttle position and the engine temperature. The current engine temperature and throttle position is measured through sensors and the firing map is selected by using these values.

FIRING ANGLE CALCULATION

If the current RPM is in between the values stored in the look-up table, then the firing angle for that RPM can be calculated using linear interpolation.

FIRING ANGLE TO TIMER COUNTS CONVERSION

The firing angle is converted to corresponding timer counts.

Capture/Compare/PWM (CCP) Compare Mode

The timer count corresponding to the firing angle is stored in the CCP compare register. When the timer count and compare values match, the interrupt flag is set. In the Interrupt Service Routine, a pulse is given on the digital output connected to the SCR gate. The output of the compare function can also drive the pin connected to the SCR directly by means of the Peripheral Pin Select.

Challenges in the Conventional CDI Approach

FIRING ANGLE TO TIMER COUNTS CONVERSION

After selecting the firing angle from the map and the interpolation, the firing angle has to be converted to corresponding timer counts. Even if the firing angle value is the same for the range of RPM, the corresponding timer counts will be different. Thus, the calculation of the firing angle to timer count is necessary for every RPM change. If computations are not complete before the expected firing event, a previous value of the firing angle is used.

RPM CALCULATION

Conventional method of RPM calculation uses a 16-bit Timer1 along with capture peripheral. In case of lower RPM values, less than 60 (corresponding to 1 Hz frequency), the 16-bit Timer1 will overflow if the timer clock frequency is 1 MHz. The timer overflow bit should be taken into account for RPM calculations.

RESOLUTION

The firing angle resolution is RPM dependent.

For example, for a Timer1 clock of 1 MHz and RPM of 250 the angular resolution is AR = 360/Timer1 counts for 250 RPM = 360/4000 = 0.09˚ and for 10.000 RPM the angular resolution is AR = 360/Timer1 counts for 10.000 RPM = 360/100 = 3.6˚.

CDI IMPLEMENTATION USING AT

The PIC16F161X family of 8-bit PIC microcontrollers has a CIP called the Angular Timer (AT), which can be used in internal combustion engines to fire the spark at the exact firing angle with very little CPU intervention. The features of the AT need to be explored before understanding how they reduce the need of CPU intervention with the help of few other CIPs.

AT

The AT divides the incoming periodic signal (one single cycle) into smaller equidistant angles. This allows the division of signals based on a phase angle, instead of time. Irrespective of the frequency of an input signal, the division is always constant and configurable.

As shown in Figure 6, the periodic pulse input to the AT can be selected either from the internal core independent peripherals or from the external pin. The AT has two counters: the period counter and the phase counter. The period counter counts the length of the external periodic pulse which is stored in the ATxPER register after every external pulse.

The period counter uses (ATxCLK/(ATxRES+1)) as a clock for counting. The phase counter divides the input periodic pulse or one cyclic rotation to equidistant angles. The phase counter uses (ATxCLK/(ATxPER+1)) as a clock for counting. The ATxRES register defines the number of angular divisions per cycle. This value is determined by the user. If the external pulse frequency is reduced, the period counter value crosses the ATxPER register that was stored during the previous period. When the period counter value is 1.5 times of ATxPER, a missing pulse trigger is given by the AT to indicate sudden decrease in external frequency. This is called missing pulse trigger in the Adaptive mode. The ATxMISS register is configured by the user to give a missing pulse output at a fixed value which can be used to detect errors in line frequency detection to control generator engine speed. This is called a Fixed Missing Pulse mode.

The AT has no output, but it generates an interrupt at every period pulse, phase pulse, and missing pulse.

There are two modes in which the Angular Timer can be configured:

- Single-Pulse mode
- Multi-Pulse mode

The following sections explains each of the modes in detail.

FIGURE 6: AT BLOCK DIAGRAM IN SINGLE-PULSE MODE

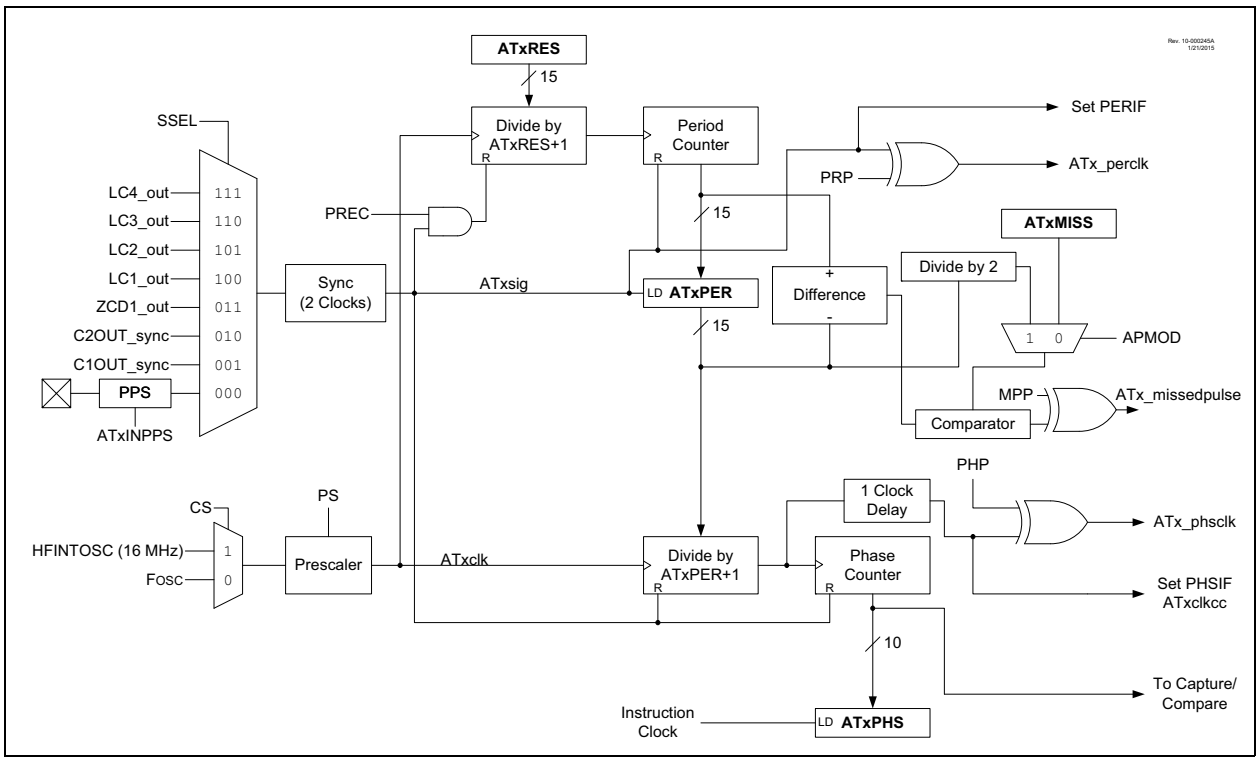
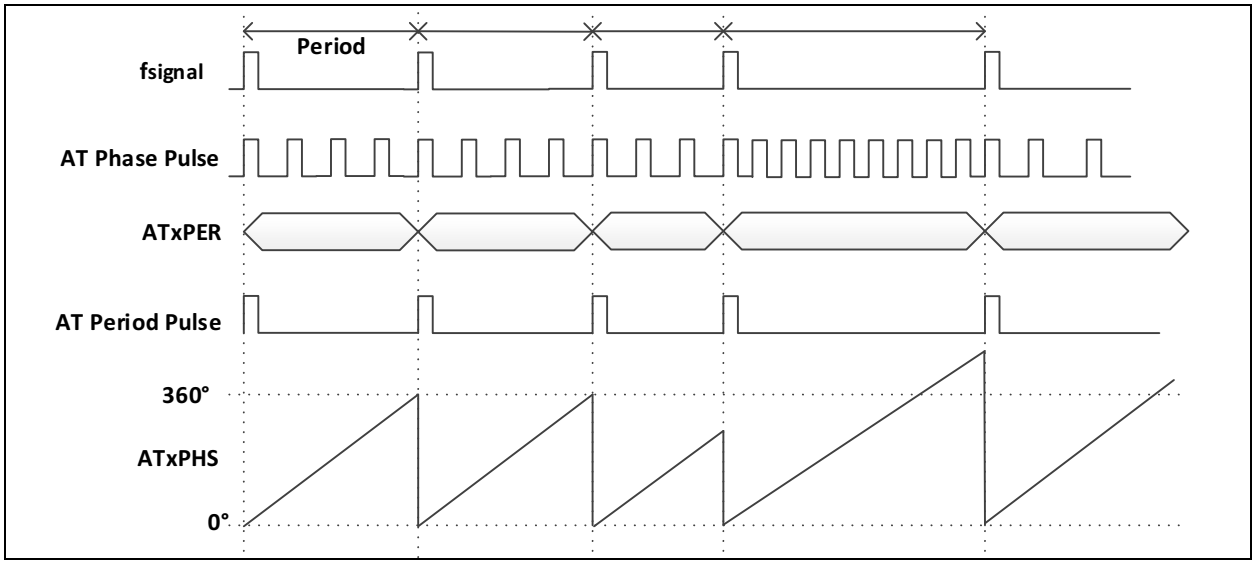


FIGURE 7: SINGLE-PULSE MODE TIMING DIAGRAM



SINGLE-PULSE MODE

In Single-Pulse mode, every external pulse is a period pulse which determines one complete cycle or rotation of 0 to 2π (360°), as shown in Figure 7. The phase pulse output depends on the previous period value, since the period register (ATxPER) only updates once every period.

The AT uses the clocks in Equation 2 for the period and phase counter.

EQUATION 2: AT CLOCKS

$$AT_{periodClock} = \frac{AT_{clock}}{ATxRES + 1}$$

$$AT_{phaseClock} = \frac{AT_{clock}}{ATxPER + 1}$$

Where ATclock is AT input clock given by

$$AT_{clock} = \frac{ATxCLK}{AT_{Prescalar}}$$

Where: ATxCLK is the clock selection using the ATxCLK register.

ATPrescalar is the AT clock prescaler value.

ATxRES is the AT resolution register set by the user, determines the number of angular divisions per cycle.

ATxPER is the period counter value for one complete period cycle of the external signal fsignal (input periodic pulse signal).

The instantaneous values of ATxPER and ATxPHS in Single-Pulse mode can be calculated as shown in Equation 3.

EQUATION 3: ATxPER AND ATxPHS IN SINGLE-PULSE MODE

$$ATxPER = \frac{AT_{periodClock}}{f_{signal}} = \frac{ATxCLK}{AT_{Prescalar} \times (ATxRES + 1) \times f_{signal}}$$

$$ATxPHS = \frac{AT_{phaseClock}}{f_{signal}} = \frac{ATxCLK}{AT_{Prescalar} \times (ATxPER + 1) \times f_{signal}}$$

Maximum phase value in the ATxPHS counter in the case of Single-Pulse mode will be equal to ATxRES (angular resolution), as indicated in Equation 4.

EQUATION 4: ATxPHS MAXIMUM PHASE VALUE

$$ATxPHS_{max} = ATxRES = AR - 1$$

Where: ATxPHSmax is maximum value of the AT phase register.
ATxRES is value configured in the AT angle resolution register.
AR is the desired angular resolution.

EXAMPLE 1: AT REGISTER VALUES IN SINGLE-PULSE MODE

Suppose, ATxCLK = 32 MHz, ATPrescalar = 1:1 = 1, AR = 360˚, f_{signal} = 250 Hz.

Hence, ATxRES = AR-1= 359, as shown below:

$$ATxPER = \frac{32000000}{1 \times (359 + 1) \times 250} = 355.55555555556 \approx 355 \text{ and}$$

$$ATxPHS = \frac{32000000}{1 \times (355 + 1) \times 250} = 359.55 \approx 360$$

MULTI-PULSE MODE

In AT Multi-Pulse mode, the input signal contains more than one pulse per cycle, as shown in Figure 9 as f_{signal}.

FIGURE 8: AT BLOCK DIAGRAM IN MULTI-PULSE MODE

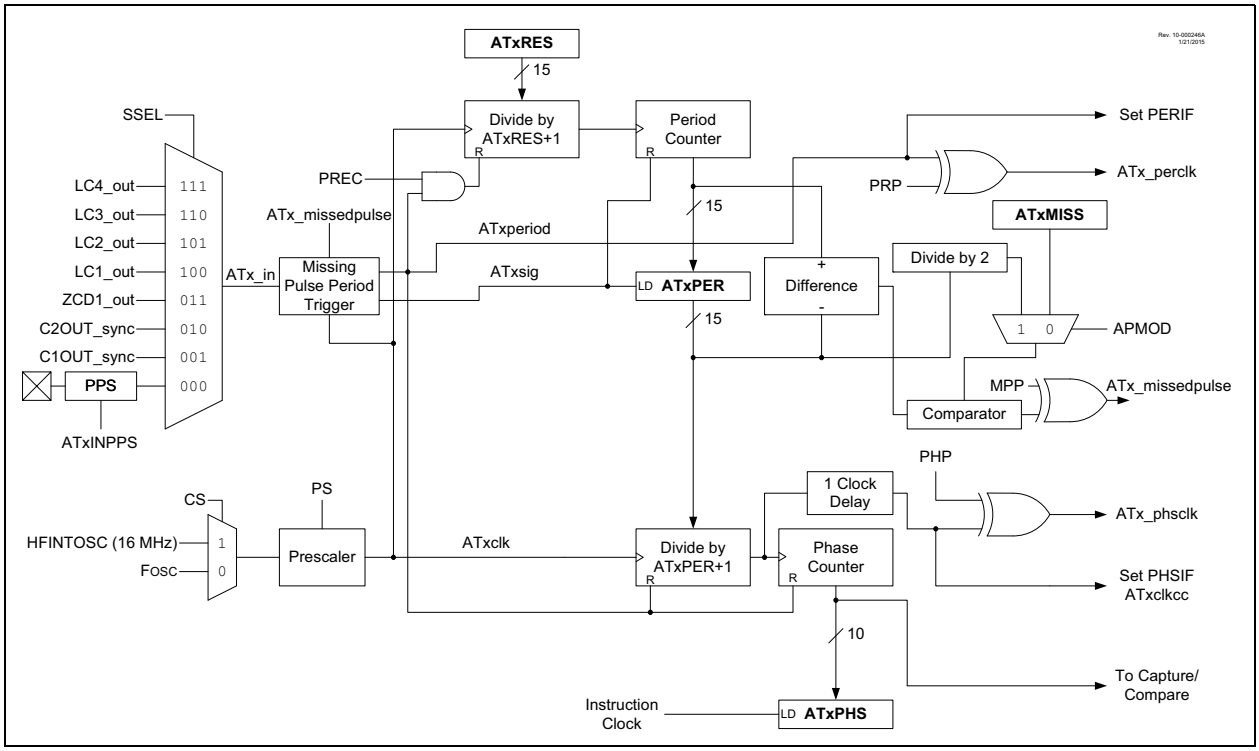
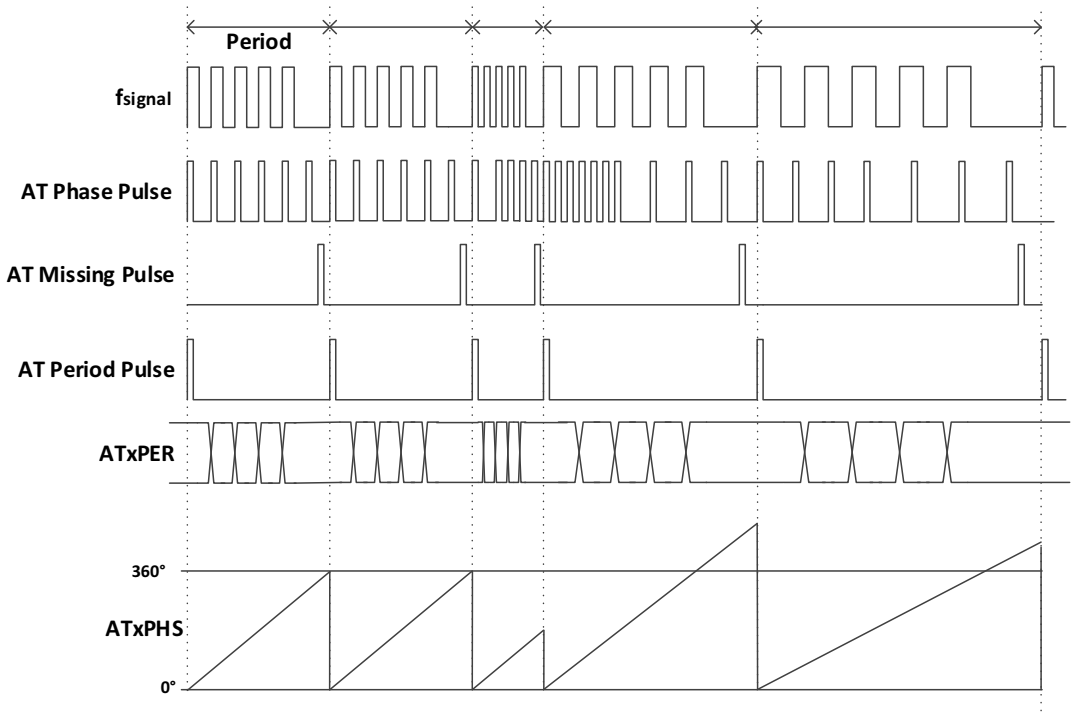


FIGURE 9: MULTI-PULSE MODE TIMING DIAGRAM



To determine the period of input signal, the signal must have at least one missing pulse. As indicated in Figure 9, there are five signal pulses with one missing pulse which makes it six pulses per period. A missing pulse is generated by the AT when no pulse is detected after the fifth one. After the missing pulse, the first rising edge on f_{signal} gives the AT period pulse output. Hence, the missing pulse and the period pulse have exactly the same length. The ATxPER register updates on every rising edge of the external pulse, except for the pulse immediately following the missing pulse. This helps ATxPHS to correct itself according to ATxPER after every rising edge in order to output phase pulses correctly within the same period.

Since the ATxPER value resets every external pulse except for the pulse after the missing pulse, the ATxRES should be chosen so that ATxPHS should not cross the maximum count of 1024 (10-bit register which is 210).

A Multi-Pulse mode example, Example 2, shows choosing the resolution register (ATxRES) value for the required angular resolution (AR).

The instantaneous values of period and phase registers in Multi-Pulse mode can be given as shown in Equation 5.

EQUATION 5: ATxPER AND ATxPHS IN MULTI-PULSE MODE

$$ATxPER = \frac{ATxCLK}{AT_{Prescalar} \times f_{pulse} \times (ATxRES + 1)}$$

$$ATxPHS = \frac{ATxCLK}{AT_{Prescalar} \times f_{MissingPulse} \times (ATxPER + 1)}$$

Where: f_{pulse} is the pulse frequency in a period signal.
$f_{missingPulse}$ is the missing pulse frequency of AT, which is same as f_{signal}.

So, the relationship between maximum counts of ATxPHS with respect to ATxRES can be determined, as demonstrated in Equation 6.

EQUATION 6: MAX ATxPHS VALUE FOR MULTI-PULSE MODE

$$ATxPHS_{max} = ATxRES \cdot \left(\frac{f_{pulse}}{f_{missingPulse}} \right)$$

As a result, the value of ATxRES will be different for the same angular division or angular resolution (AR) in the Multi-Pulse mode compared to the Single-Pulse mode. See Equation 7.

EQUATION 7: ATxRES CALCULATION FOR MULTI-PULSE MODE

$$ATxRES = \left[AR \cdot \left(\frac{f_{missingPulse}}{f_{pulse}} \right) \right] - 1 = \left[AR \cdot \left(\frac{T_{pulse}}{T_{missingPulse}} \right) \right] - 1$$

Where: $T_{pulse} = 1/f_{pulse}$ and $T_{missingPulse} = 1/f_{missingPulse}$.

Using the above value of ATxRES and Equation 5, the values of ATxPER and ATxPHS can be calculated as indicated in Example 2.

EXAMPLE 2: AT REGISTER VALUES IN MULTI-PULSE MODE

Suppose, ATxCLK = 32 MHz, $AT_{Prescalar}$ = 1:1 = 1, AR = 360°.

$f_{signal} = f_{missingPulse}$ = 250 Hz and f_{pulse} = 2500 Hz;

Hence, using Equation 7 the below results will be obtained:

$$ATxRES = \left[360 \times \frac{250}{2500} \right] - 1 = \left[\frac{360}{10} \right] - 1 = 36 - 1 = 35$$

$$ATxPER = \frac{32000000}{1 \times 2500 \times (35 + 1)} = 355.55555555556 \approx 355$$

$$ATxPHS = \frac{32000000}{1 \times 250 \times (355 + 1)} = 35.955 \approx 35$$

CAPTURE AND COMPARE FUNCTIONS

The AT has three capture/compare modules along with it.

Capture Mode

In Capture mode, on the rising or on the falling edge of the capture input signal, the value of the phase counter (ATxPHS) is captured into the capture register (ATxCCy).

The capture event also generates a pulse that can be used to trigger other peripherals such as an ADC or a CLC input, or to generate an interrupt.

Compare Mode

In the Compare mode, the AT module compares the current phase counter value (ATxPHS) to the value in the compare register (ATxCCy). The compare register (ATxCCy) can be loaded with an angle value which is less than or equal to the angular resolution AR (ATxRES) in the Single-Pulse mode. When the two values match, a compare event is generated. This can be used to trigger other peripherals such as an ADC or a CLC input, or to generate an interrupt.

CDI Implementation with AT

Figure 10 shows the AT with Core Independent Peripherals (CIPs) in PIC16F1615/9 for the implementation of the CDI system.

CIPs of PIC microcontrollers, such as CLC, AT, SMT, and Math Accelerator are used for the firing angle control in CDI.

CONFIGURABLE LOGIC CELL (CLC)

The outputs of the signal conditioning circuit are connected to the CLC. The outputs of the signal conditioning circuit will be in logic 1 state except for the positive and negative pulses from the pick-up coil. The CLC is configured as an XOR gate, so that the CLC output will give positive pulses for both inputs, as shown in Figure 11. The CLC output is connected to the AT input internally. The AT will provide a period pulse to SMT.

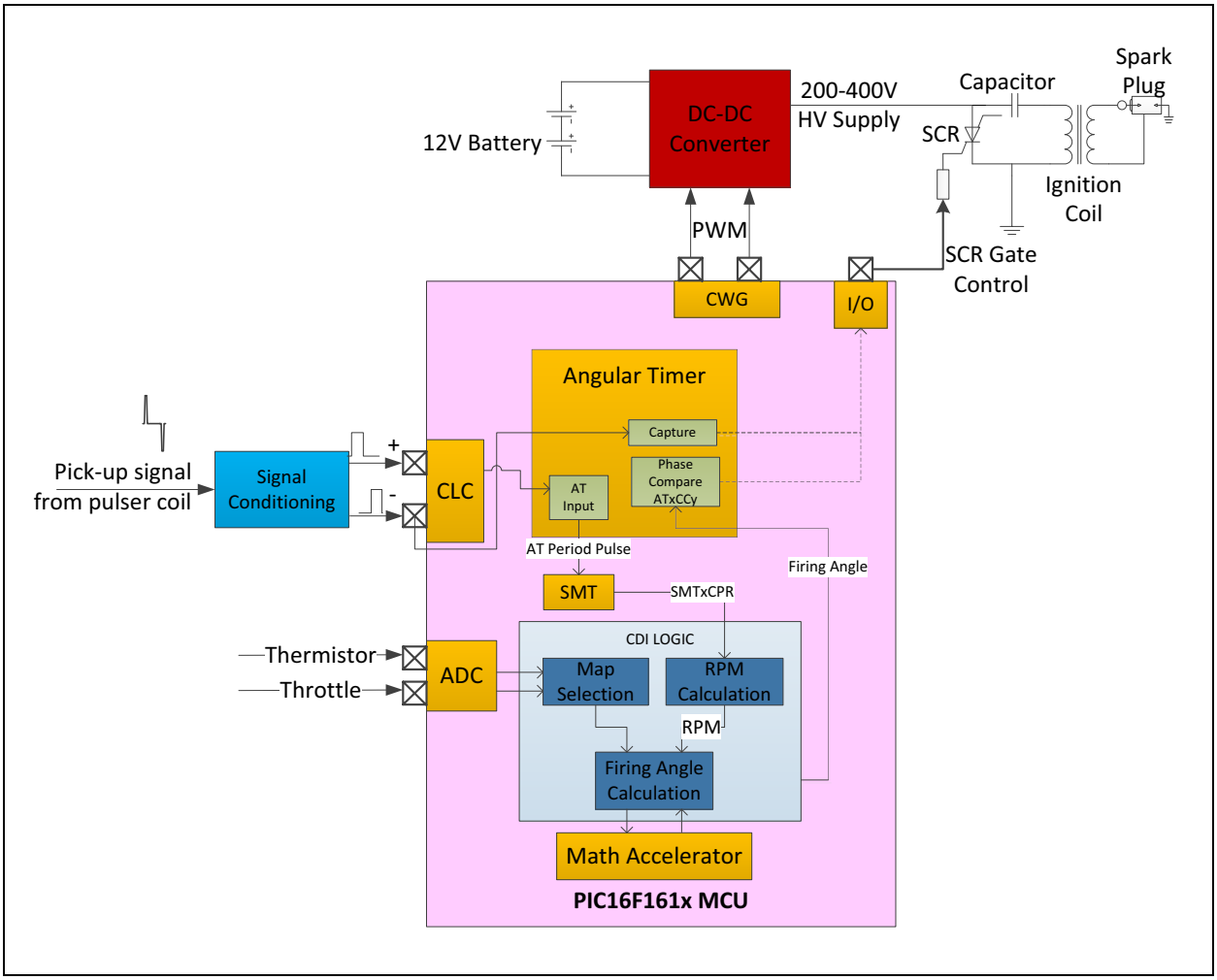
SIGNAL MEASUREMENT TIMER (SMT)

For the RPM calculation, a 24-bit SMT is used. SMT is configured in Windowed Measurement mode with the window input set to AT period pulse. Whenever AT gives period pulse, the SMT captures the timer value into SMTxCPR register, resets its timer count, and restarts counting. The timer capture into the SMTxCPR register generates a captured period interrupt.

ADC

An ADC is used to find the engine temperature and the throttle position (if it is analog). Throttle position input can be either analog or digital. In the case of a digital throttle position, the wide open throttle (WOT) is one state and the partially open throttle (POT) is another. There are different firing maps for different throttle positions and different temperature ranges.

FIGURE 10: CDI IMPLEMENTATION WITH PIC16F161X

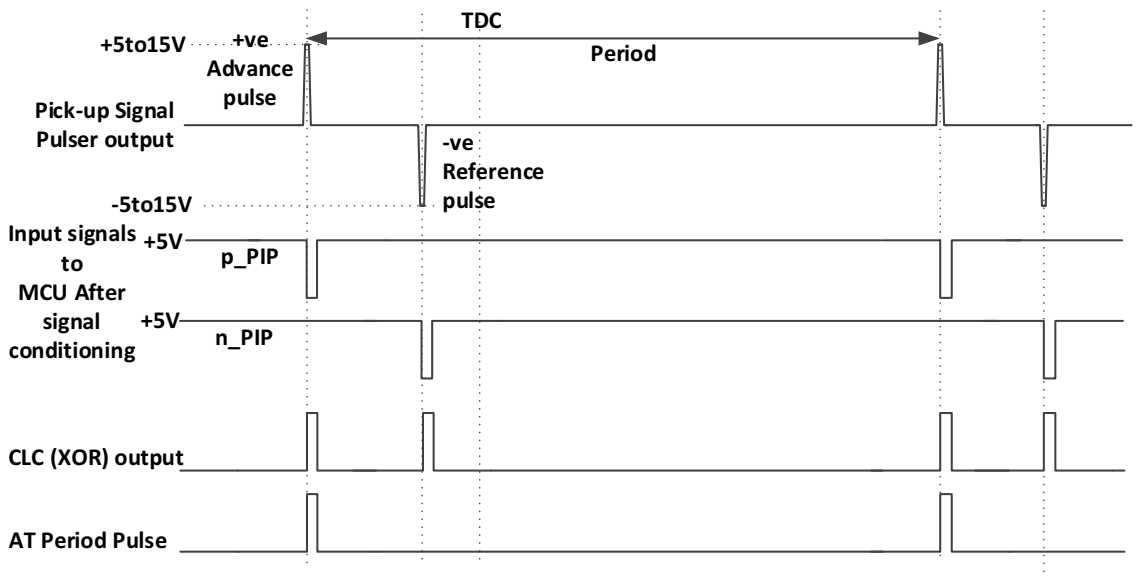


CDI LOGIC

RPM Calculation

The RPM of the engine can be calculated from the external signal frequency f_{signal} as shown in Equation 8.

FIGURE 11: SIGNAL CONDITIONING OUTPUT



EQUATION 8: RPM CALCULATION USING SMT

$$f_{signal} = \frac{SMTxCLK}{SMT_{Prescalar} \times SMTxCPR}$$

$$RPM_{Engine} = \frac{SMTxCLK}{SMT_{Prescalar} \times SMTxCPR} \times 60$$

Where: SMTxCLK = clock input for SMT using SMTxCLK register= Fosc = 32 MHz.
SMTPrescalar = SMT clock prescaler selected in SMTxCON0 = 1:8 = 8.
SMTxCPR is captured value of SMTxTMR at window input event.
fsignal is input signal frequency to AT and SMT.
60 scalar value is multiplied to convert Hz to RPM, RPMEngine = fsignal * 60.

Firing Map Selection

Every engine is associated with a firing map which shows the relation between the input engine speeds in RPM and the firing angle of the spark in degrees. These maps vary depending on throttle position and engine temperature. The current temperature of the engine and the throttle position can be used to select one of the firing maps. The map is nonlinear in nature and nondeterministic, which can change from engine to engine. Hence, these maps are divided into smaller linear maps (piecewise linear curves) and are expressed in terms of a table with RPM on LHS and firing angle on RHS.

FIGURE 12: EXAMPLE OF RPM TO FIRING ANGLE (THETA) MAP

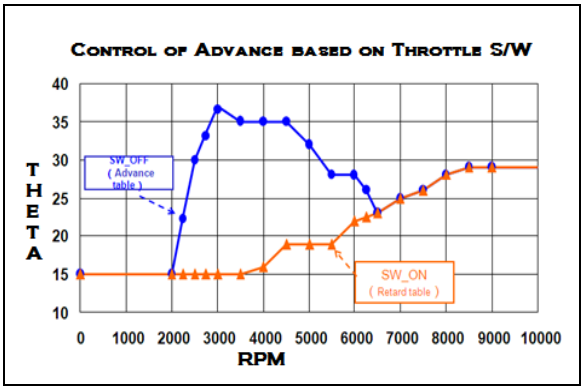


Figure 12 shows an example of a firing map. This firing map is a plot of a piecewise linear curve. Here, the red curve shows the RPM to firing angle relationship for throttle ON and the blue curve shows RPM to firing angle relationship for throttle OFF.

Firing Angle Calculation

To find the firing angle of an engine between two points in the map, linear interpolation is used as follows:

EQUATION 9: FIRING ANGLE CALCULATION USING LINEAR INTERPOLATION

$$y = y_1 + (x - x_1) \cdot \frac{y_2 - y_1}{x_2 - x_1}$$

Where: y is the instantaneous value of the firing angle to be calculated, which is in between y_1 and y_2.
x is the instantaneous value of engine RPM which is known and in between x_1 and x_2.
(x_1, y_1) and (x_2, y_2) are the successive points in the graph between which a line is drawn these are shown as row 1 and row 2 in the firing map table, respectively.
x_1 and x_2 represent values of Engine RPM of two endpoints. y_1 and y_2 represent the values of the firing angle of two endpoints of a straight line curve.

In Equation 9, the ratio ((y2-y1)/(x2-x1)) is always constant and is called slope *m* of a straight line. This can be precalculated for a curve to reduce the amount of calculation.

For example, Table 1 is the firing map given by the engine specifications. The values in the column "Slope" are calculated from the previous two column values, and are called line slopes.

TABLE 1: Firing Map for Example Engine

Engine Speed (RPM)	Firing Angle (θ°)	Slope (m)
250	10	0
500	10	0
1600	10	0
1800	10	0.075
2000	25	0.012
2500	31	0
4100	31	0
4200	31	0
5100	31	0
5200	31	0
5900	31	0
8500	31	-0.042
9000	10	0
10000	10	0

EXAMPLE 3: FIRING ANGLE CALCULATION USING INTERPOLATION

Suppose, RPM_{Engine} = 1915 rpm.

From Table 1, this RPM value lies between 1800 rpm (with a firing angle of 10˚) and 2000 rpm (with a corresponding firing angle of 25˚).

Thus, from Equation 9, the terms are as follows:

$y_1 = 10$, $y_2 = 25$, $x_1 = 1800$, $x_2 = 2000$, $x = 1915$

The result is shown below:

$$y = 10 + (1915 - 1800) \cdot \frac{25 - 10}{2000 - 1800} = 18.625 \approx 18°$$

Thus, the firing angle for the given RPM of 1915 is 18˚.

MATH ACCELERATOR

The math accelerator peripheral, also called a PID module, is used to calculate the firing angle for the current RPM using Equation 9. Using the math accelerator, the calculation can be done faster. It performs an addition and a multiplication as shown in Equation 10.

EQUATION 10: MULTIPLY/ADD OPERATION IN MATH ACCELERATOR

$$R = (A + B) \times C$$

Where:

A is the first input operand for addition (PIDxIN register).
B is the second input operand for addition (PIDxSET register).
C is the third input operand for multiplying with the result of addition (PIDxK1 register).
R is the results of one complete add and multiply operation (PIDxOUT register).

The math accelerator performs these operations in 16-bit integer, either in signed or unsigned format, configured by the user. The result of this operation is 32-bit in size, and is stored in the PIDxOUT register. The result of this operation can be configured to be accumulated or non-accumulated. For accumulation of the result, the PIDxACC register is used to accumulate previous results.

AT

Depending on the angular resolution settings in the resolution register (ATxRES) the AT will start counting. AT will divide the input signal into equidistant angles. The phase counter (ATxPHS) is incremented accordingly.

Selecting the ATxRES Value of the AT for CDI Implementation

The granularity of the phase counter is called angle resolution. The value in the ATxRES register is used to determine the angle resolution of the AT phase counter.

The angle resolution can be achieved through Equation 11 below.

EQUATION 11: ANGLE RESOLUTION

$$Angle\ Resolution = \frac{360°}{ATxRES + 1}$$

In the AT, the period is measured in increments of ATxRES. For smaller ATxRES values, the ATxPER values will be high. AT truncates the actual period value in order to load the value in the period register (ATxPER). For a smaller percentage of error, the higher ATxPER value is required.

The figure of merit (FOM) for the system is defined as the minimum expected ATxPER value. For error to be less than 0.2%, FOM should be as per Equation 12.

EQUATION 12: FIGURE OF MERIT

$$FOM = 100 \times \frac{1}{Error} = 100 \times \frac{1}{0.2} = 500$$

Therefore, it is recommended that the AT clock and resolution (ATxRES) be chosen so that the minimum expected ATxPER value (FOM) corresponding to the highest RPM will be greater than 500.

EXAMPLE 4: ATxRES CALCULATIONS

Consider the following values:

Maximum engine RPM = 10000

f_{signal} = RPM/60= 10000/60 = 166.66 Hz

ATxCLK = 32 MHz

ATPrescalar = 1:1

ATxPER is given by

$$ATxPER_{min} = \frac{ATxCLK}{AT_{Prescalar} \times (ATxRES_{max} + 1) \times f_{signal\_max}}$$

So, for a FOM of 500 (ATxPERmin), ATxRESmax will be as shown below:

$$ATxRES_{max} = \frac{ATxCLK}{AT_{Prescalar} \times ATxPER_{min} \times f_{signal\_max}} - 1 = \left(\frac{32000000}{1 \times 500 \times 166.66} - 1\right) = 383$$

It is preferable to select the value of ATxRES smaller than $ATxRES_{max}$.

If ATxRES is chosen as 359°, to get an angle resolution of 1° the minimum ATxPER value (FOM), corresponding to the highest RPM will be as shown below:

$$ATxPER = \frac{ATxCLK}{AT_{Prescalar} \times (ATxRES + 1) \times f_{signal\_max}} = \frac{32000000}{1 \times 360 \times 166.66} = 533.33$$

LOW RPM CORRECTION

From zero RPM to very low RPM, the AT period counter can overflow due to the limited size of the AT period counter, which is 15 bits in length. The value in this AT period counter will be high for lower RPM counts. Therefore, a sufficiently low clock needs to be chosen for the AT input, so that lower RPM can also be taken into consideration. As a result, the minimum value of RPM is 60, as calculated in Example 5. Decreasing the AT input clock frequency and increasing the AT clock prescaler would result in less accuracy of angular divisions, but this design trade-off is necessary here to take lower RPM into account.

EXAMPLE 5: LOW RPM CORRECTION

Suppose ATxCLK = 32 MHz, ATxRES = 60, ATxPER max = 32767 (15-bit counter), hence f_{min} is calculated as shown below:

$$f_{min} = \frac{ATxCLK}{AT_{Prescaler} \times (ATxRES + 1) \times ATxPER_{max}}$$

As a result, for the case above with ATPrescalar = 1, f_{min} is 16.27 Hz or 976 RPM.

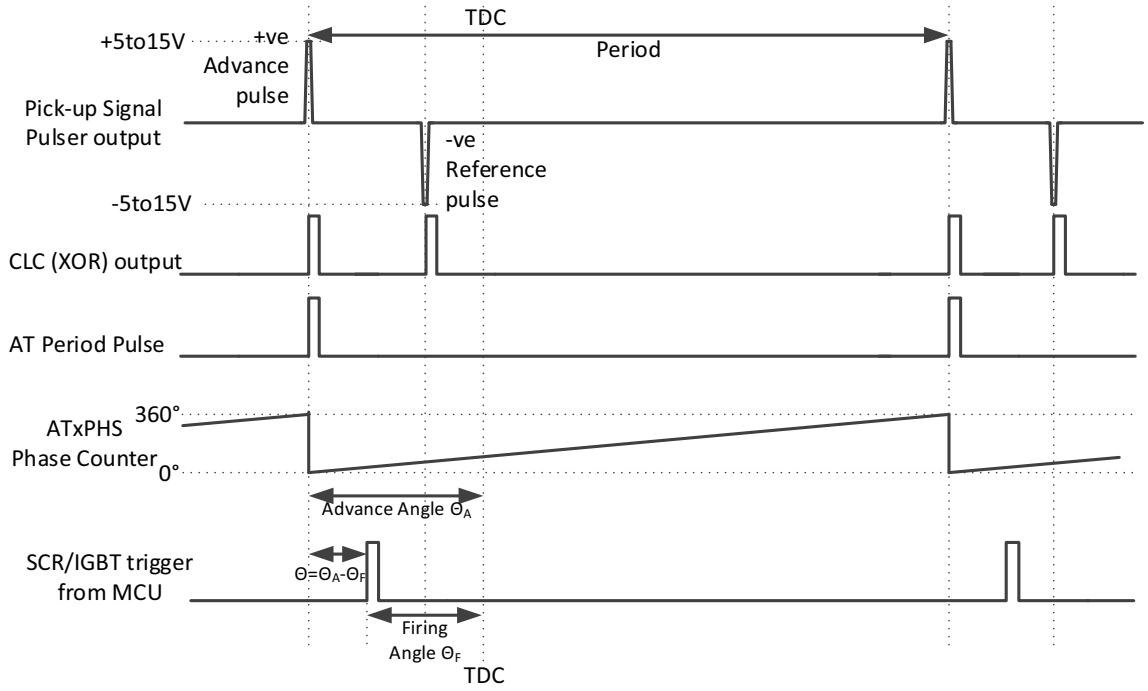
To include lower RPM, if ATxCLK = 16 MHz and $AT_{Prescalar}$ = 8 are selected, the value will be calculated as follows:

$$f_{min} = \frac{16000000}{8 \times 60 \times 32767} = 1.017\ Hz \equiv 60\ RPM$$

SCR/IGBT FIRING

The calculated firing angle is loaded into the AT compare register. When the value in the compare register matches the AT phase counter, a compare interrupt is generated and IGBT or SCR is fired, as shown in Figure 13. In case of IGBT, another compare register is used to switch off the firing.

FIGURE 13: TIMING DIAGRAM FOR SCR FIRING



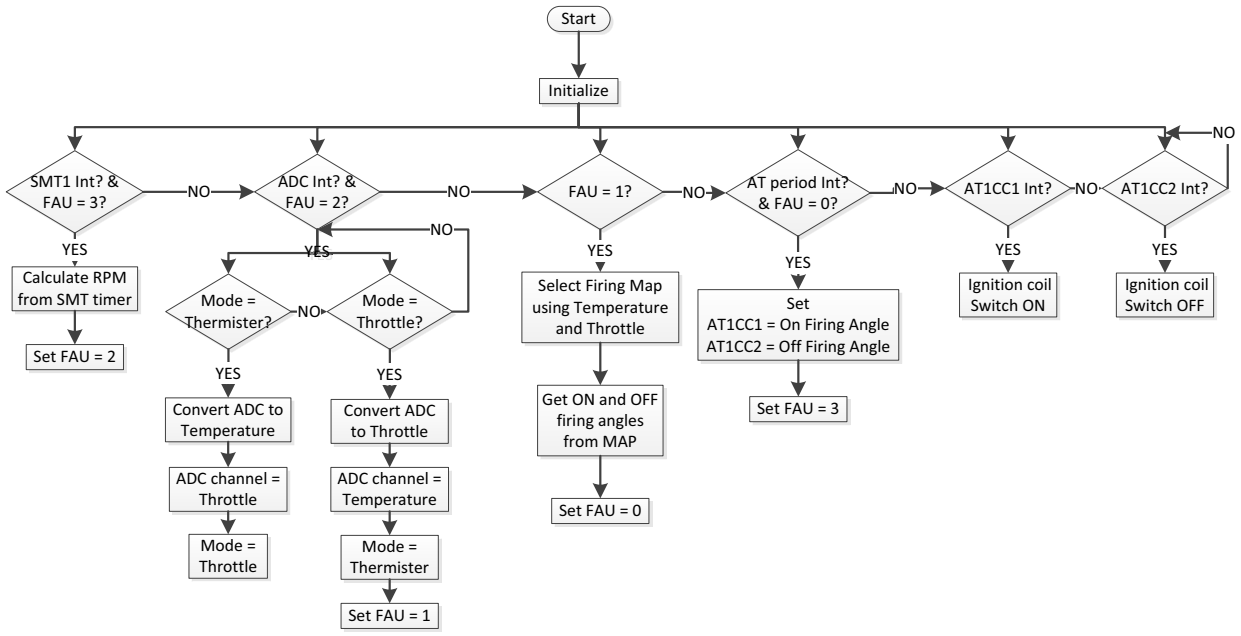
COMPLEMENTARY WAVEFORM GENERATOR (CWG)

The CWG is used to generate a single or complementary sine wave modulated PWM signal for the DC to AC inversion. This inversion is required to convert 12V DC from battery to 220V AC RMS which is rectified again to generate 300V DC. This 300V DC is used to charge the CDI capacitor. A complementary PWM can be used to make full-wave inversion while a single PWM only makes half-wave inversion. For very small engines, only half cycle is enough to charge the capacitor where RPM is very low (i.e., chainsaws, lawn mowers, etc.).

FIRMWARE

Firmware flowchart is shown in Figure 14.

FIGURE 14: FIRMWARE FLOWCHART



Note 1: FAU is a firing angle update variable which keeps track of the firing angle calculation step number. When it is done, update the ON and OFF firing angles.

2: The ADC will sample the engine temperature and the throttle position for each AT period pulse. These will be taken as current parameter values, and a firing map is selected based on these values.

Program Flow

The program execution is sequentially divided into a number of steps to compute the RPM and firing angle as explained below:

1. **Initialize:** The board is powered on or reset; all the peripherals (ADC, PID, CWG, CLC1, SMT1 and AT1) are initialized.
2. **Input Acquisition:** The CLC1 takes two inputs to generate one XORed output connected to AT input. AT starts counting and gives period pulses periodically. This AT period pulse is used by SMT as window input in “Windowed Measure mode”. The SMT at every AT period input captures the value of the SMT1TMR register to the SMT1CPR register and a corresponding interrupt is generated. The program is notified about a new SMT value available for calculation. This is the acquisition step.
3. **ADC:** The analog output of the sensor for engine temperature and throttle position is digitized by an internal 10-bit ADC.
4. **RPM calculation:** The RPM of the engine is calculated from the SMT1CPR register value.
5. **Map selection:** Using temperature and throttle position values, a particular map is selected. From this selected map, the firing angle corresponding to the present RPM is read from the look-up table. If the current RPM is between the values stored in the look-up table, the firing angle for that RPM is calculated using interpolation.
6. **Firing Angle Update:** The firing angle value is stored in the AT compare register (AT1CCy).
7. **AT compare interrupt:** In the AT compare ISR a pulse is given on the port pin connected to SCR/IGBT gate to turn ON the switch for firing the spark. If the IGBT is used, another ATxCCy register is used to switch OFF the IGBT.
8. Steps 2 through 7 are repeated every AT period cycle.

COMPARISON BETWEEN CDI USING THE AT, AND CDI USING THE CONVENTIONAL APPROACH

Firing Angle to Timer Counts Conversion

When the conventional method is used, the firing angle has to be converted to corresponding timer counts. Whereas the firing angle value obtained from the map can be written directly to AT compare register. When the angle value matches, it generates an interrupt and the SCR can be fired. There is no CPU intervention because the AT is a separate on-chip peripheral.

Interpolation Calculation for Firing Angle

For the conventional method, the interpolation calculation of the firing angle and its conversion to timer count calculation take a longer time.

When using the math accelerator, the firing angle interpolation calculation can be expedited.

RPM Calculation

When the conventional method is applied, there is the possibility of a timer overflow at lower RPM.

SMT is a 24-bit timer. If used for RPM calculations with a clock frequency of 1 MHz, it can count until 0.125 Hz (7.5 RPM) without overflowing.

Resolution

The angular resolution varies with RPM in the case of conventional methods using the timer, capture and compare peripherals. Whereas, the desired resolution of the firing angle can be directly loaded into the resolution register (ATx RES) of the AT. For a resolution of 1°, the resolution register is loaded with a value of 359, in the case of Single-Pulse mode. The angular resolution is independent of the RPM of the engine.

Overall Comparison

The maximum frequency of the input pick-up signal to the CDI system can be 500 Hz, corresponding to 30000 RPM (500 Hz or 2 ms period). Within this period of 2 ms, the CDI operation must finish all of the execution and calculations. Considering PIC16F devices for both the conventional method and that using the AT, which has eight MIPS of execution speed at 32 MHz, the comparison table below shows the CPU usage for both systems.

The MIPS can be calculated as indicated in Equation 13.

EQUATION 13: MIPS CALCULATION

$$MIPS_{Actual} = \frac{MIPS_{Total}}{T_{MIN}} \times T_{Execution}$$

Where: $MIPS_{Actual}$ is the calculated MIPS for current program execution

T_{MIN} is the minimum input period which is $1/f_{MAX}$ i.e., inverse of minimum input signal frequency

$MIPS_{Total}$ is the max MIPS of the device

$T_{Execution}$ is the actual execution time of program

In the case above, the T_{MIN} = 2 ms and $MIPS_{Total}$ = 8. And CPU usage, based on maximum MIPS, is given as shown in Equation 14, below.

EQUATION 14: % CPU USAGE

$$\%CPU_{USAGE} = \frac{MIPS_{Actual}}{MIPS_{Total}} \times 100$$

TABLE 2: TIMING COMPARISON

Program Operations	CDI Using the Angular Timer			CDI Using the Conventional Method		
	Execution Time (µs)	MIPS	% CPU Usage	Execution Time (µs)	MIPS	% CPU Usage
Long Division	150	0.6	7.5	200 (includes extra time for the T1 count over-flow value inclusion in the calculation)	0.8	10
Firing Angle Calculation	150 (using Math Accelerator)	0.6	7.5	350	1.4	15.5
T1 count to degree	0	0	0	200	0.8	10
ISR and other program commands	50	0.2	2.5	50	0.2	2.5
Total	**350**	**1.4**	**17.5**	**800**	**3.2**	**40**

Table 3 summarizes the resource comparisons of the two methods for CDI implementation.

TABLE 3: RESOURCE COMPARISON

Parameters	CDI Using the AT	CDI Using the Conventional Method
Flash Memory (Words)	1994	3347
RAM (Bytes)	65	233
Peripherals Used	ADC, CLC, AT, SMT, Math Accelerator, CWG	ADC, CCP, Timer 1, PWM

Table 4 summarizes the performance comparisons of the two methods for CDI implementation.

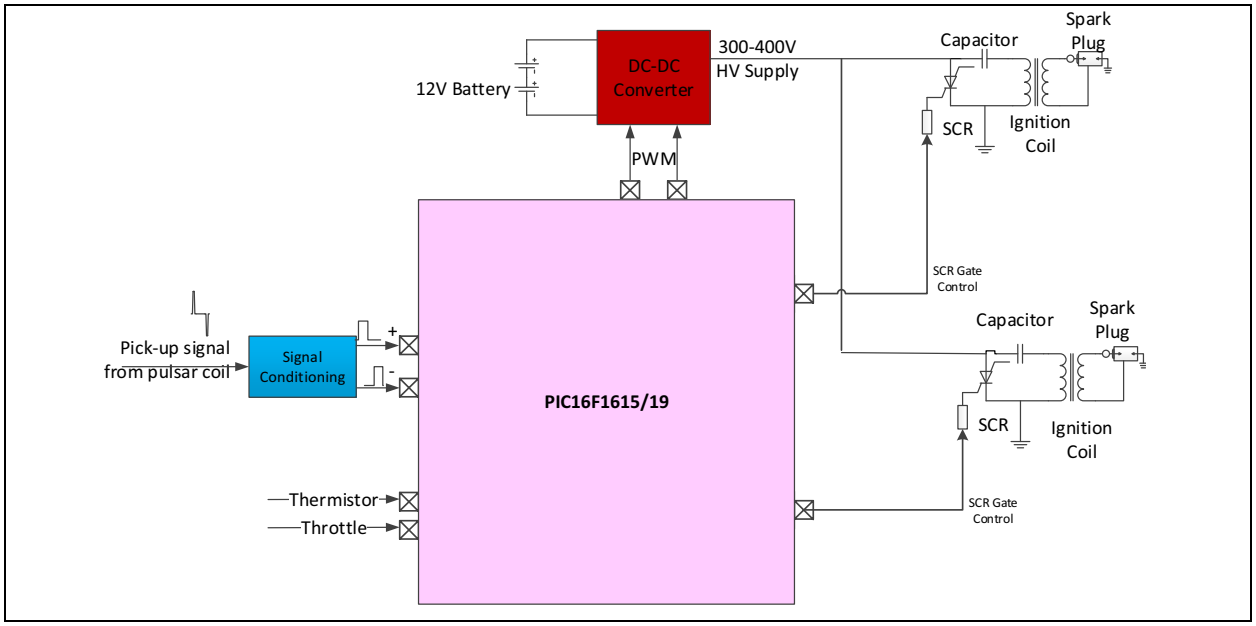
TABLE 4: PERFORMANCE COMPARISON

Parameters	CDI Using AT	CDI Using conventional Method
Accuracy	Depends on FOM and the ATxRES value	Depends on engine speed (i.e., RPM and granularity of timer count for firing angle calculation).
Resolution	Minimum 0.35˚ resolution (i.e., 10 bit) ATxRES value	Depends on Timer1 clock and RPM (i.e., for 1 MHz clock, 250 RPM, the angular resolution is 0.09˚, and for 10000 RPM the angular resolution is 3.6 ˚).

EXTENDING THE CDI DESIGN USING CIPs FOR DUAL SPARK TOPOLOGY

The CDI system can have more than one spark per ignition system for improving fuel consumption and exhaust emissions, thus improving performance. The CDI design using CIPs of PIC16F1615/9 can be modified to support dual spark ignition. The two sparks can be used either in two different cylinders of an engine or in the same cylinder in multi-spark topology per cylinder, in order to have a high performance (i.e., RPM to ignite air fuel mixture more quickly). Figure 15 illustrates the modification over the existing design to suit the dual spark topology.

FIGURE 15:



CONCLUSION

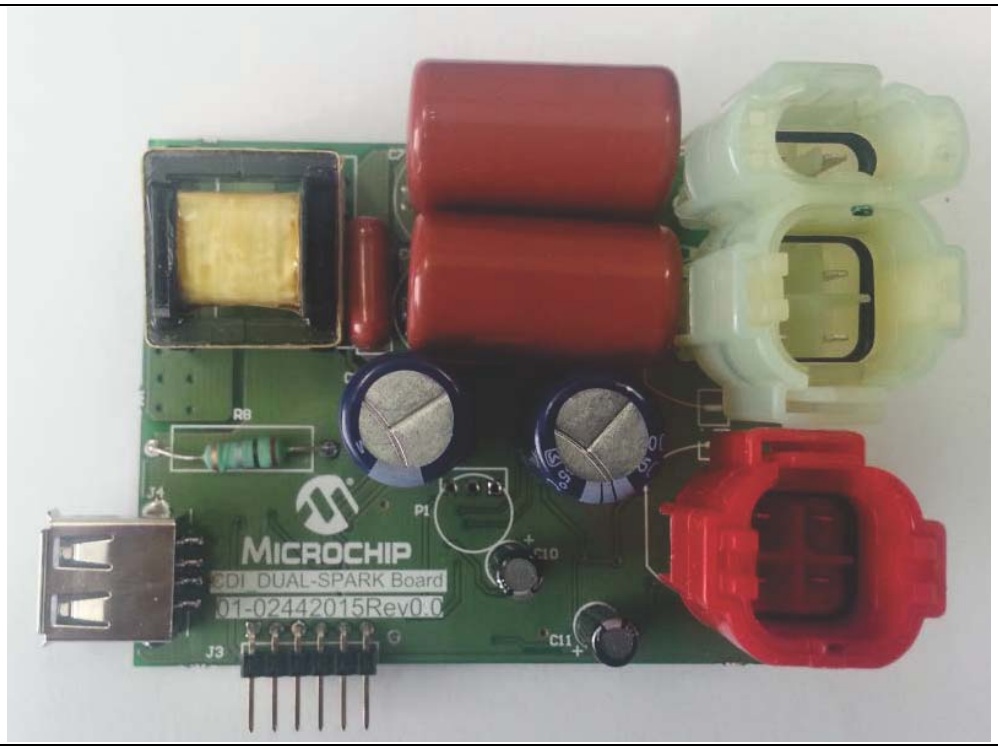
CDI systems can be implemented using PIC16F microcontrollers either through the conventional method or by using CIPs, such as AT, CLC, SMT, Math Accelerator, and CWG. However, using the CIPs greatly improves the overall performance and implementation of CDI.

The AT successfully divides the input signal to angular division without CPU intervention, which helps to boost performance by removing the need for firing angle conversion from degree to equivalent time. These angular divisions are also very accurate and constant throughout the input signal range. As shown in the comparison tables (Table 2 through Table 4) of this document, the performance of the CDI system can be greatly enhanced using the AT, with significant CPU bandwidth remaining for other calculations. In addition, with the use of the Math Accelerator, the calculations are now more accurate and faster. The SMT with its inherent high-bit resolution helps in tracking low engine RPM and taking necessary action, without the need for large computations. Albeit being outside the scope of this application note, a similar implementation technique can be extended to other systems, such as inductive discharge ignition (IDI), that use the same CIPs.

APPENDIX A: SCHEMATICS

For schematics and reference board, contact the local sales office.

FIGURE A-1: CAPACITIVE DISCHARGE IGNITION REFERENCE DESIGN



APPENDIX B: CDI IMPLEMENTATION WITH MULTI PIP SYSTEM

If the pick-up signal from the pulser coil consists of more than one pulse per period, then AT Multi-Pulse mode should be used. The example of multi-pulse output is shown in Figure 4.

The implementation of the CDI system for both Single and Multi-Pulse modes is the same, except for the configuration of the AT mode and the ATxRES register value, which is explained in the Multi-Pulse mode section of CDI Implementation Using AT (Equation 7). LSb in the AT1CON0 register should be set to 1 for the Multi-Pulse mode.

The RPM calculation, map selection, and firing angle calculations are all the same. The value to be loaded in the ATxCCy register is also the same, and it is noted as such in the Equation B-1 below.

EQUATION B-1: AT COMPARE MODULE VALUE CALCULATION

$$ATxCCy = \varnothing - \theta$$

Where: $\varnothing$ is the angle of positive pulse with respect to TDC in an anticlockwise rotation.

θ is the calculated firing angle in an anticlockwise rotation.

In Single or Multi-Pulse mode, the φ represents the angle where the pulser coil gives the first positive pulse at an angle before TDC and after BDC in a counterclockwise direction.

APPENDIX C: FIRING ANGLE CALCULATION WITH Q15 NUMBER FORMAT AND MATH ACCELERATOR

Equation 9 of the firing angle calculation requires one addition, three subtractions, one division, and one multiplication operation per external signal cycle for implementation. In 8-bit MCU, these arithmetic operations might require quite a long time to finalize. To minimize the number of computations, the slope value can be precalculated in firing angle calculation as shown in Table 1. Hence, Equation 9 can be represented as shown in Equation C-1:

EQUATION C-1: LINEAR INTERPOLATION

$$y = y_1 + (x - x_1) \times m$$

Where: m = $(y_2–y_1)/(x_2–x_1)$.

m is the slope of straight line between (x_1, y_1) and (x_2, y_2).

As seen from Table 1, the value of *m* can be smaller than 1 which requires Equation C-1 to be calculated using the floating point math. In 8-bit MCU, this math can take significant amount of time. To ease this calculation, the Q15 number format is used. In Q15 format, every number is divided with the max value of that parameter type and then multiplied with the maximum value of 16-bit signed integer (i.e., 32767). This brings the number to a 16-bit numeral form. This technique is also called normalization.

Now, the normalized Q15 number can be used in calculation and integer math can easily be used. When all calculations are finished, the result of the operation is converted back to normal units.

EXAMPLE C-1: FIRING ANGLE CALCULATION IN Q15 FORMAT

Suppose, RPMMAX = 30660, f_{MAX} = RPMMAX/60 = 511

θ_{MAX} = 63 (Maximum firing angle)

Applying Q15 on RPM = 10000 and FiringAngle = θ = 25, the following string of equations is obtained:

$$f = \frac{RPM}{60} = \frac{10000}{60} = 166.666667 \approx 167$$

$$f_{Q15} = \frac{f}{f_{MAX}} \times 32767 = \frac{167}{511} \times 32767 = 10687.21461 \approx 10687$$

$$\theta_{Q15} = \frac{\theta}{\theta_{MAX}} \times 32767 = \frac{25}{63} \times 32767 = 13002.77778 \approx 13002$$

Applying the above formulas, Table 1 can be written as Table C-1 below.

TABLE C-1: FIRING MAP REPRESENTED IN Q15 FORMAT

f_{signal} (Hz to Q15)	Firing Angle (θ° to Q15)	Slope (m °/Hz)
267	5201	0
534	5201	0
1710	5201	0
1924	5201	37
2137	13003	6
2672	16123	0
4382	16123	0
4489	16123	0
5450	16123	0
5557	16123	0
6305	16123	0
9084	16123	-20
9618	5201	0
10687	5201	0

The slope value in Table C-1 is calculated from the Q15 formatted firing map values and not from the actual RPM and firing angle values. To calculate the actual firing angle from Table C-1, Equation C-1 can be used, with some modifications:

EQUATION C-2: CALCULATION OF FIRING ANGLE

$$y = \frac{\left[y1_{Q15} + \left(x_{Q15} - x1_{Q15}\right) \times m\right] \times \theta_{MAX}}{Q15_{MAX}}$$

EXAMPLE C-2: FIRING ANGLE CALCULATION IN Q15 FORMAT WITH CONVERSION FROM Q15 TO DEGREES

By using the values given in Example 3 and performing Q15 calculations, the following results are obtained.

$$f = \frac{RPM}{60} = \frac{1915}{60} = 31.91666667 \approx 32$$

$$f_{Q15} = \frac{f}{f_{MAX}} \times 32767 = \frac{32}{511} \times 32767 = 2046.6 \approx 2046$$

$$\theta_{Q15} = \frac{\theta}{\theta_{MAX}} \times 32767 = \frac{10}{63} \times 32767 = 5201.111111 \approx 5201$$

Hence, from Table C-1 and the above calculated values, the results are shown here:

y_{1Q15} = 5201, x_{Q15} = 2046, x_{1Q15} = 1924, m = 37, $Q15_{MAX}$ = 32767. Using Equation C-1 y can be calculated as shown:

$$y = \frac{[5201 + (2046 - 1924) \times 37] \times 63}{32767} = 18.6787 \approx 18°$$

These results match the results in Example 3, Firing Angle Calculation Using Interpolation.

The integer math above can easily be done with the Math Accelerator. The result in the above example contains Equation 10 operation twice. The first multiply/add operation is [(2046-1924)x37] in numerator. The second multiply/add operation is adding the result of the first operation to 5201, which is multiplied by 63. The final output of the second operation can then be shifted logically to the right 15 times to make the division by 32767.

APPENDIX D: TIPS AND TRICKS TO REMOVE INTERPOLATION CALCULATION

The interpolation operation in Equation C-1 takes a long time to finalize due to the inclusion of one division and one multiplication operation, since these operations are implemented using an emulator in an 8-bit MCU. To increase the speed of execution of the CDI program, the interpolation operation can be replaced by a firing angle look-up table. For this, Table 1 can be implemented as a look-up table for 1˚ change in firing angle and the left hand side will contain the range of RPM values for that firing angle. Below is Table 1 expanded as a look-up table.

TABLE D-1: EXPANSION OF TABLE 1 FOR EVERY 1˚ CHANGE IN FIRING ANGLE

Engine Speed Range (RPM)	Firing Angle (θ˚)
250-1813	10
1814-1826	11
1827-1839	12
1840-1853	13
1854-1866	14
1867-1879	15
1880-1893	16
1894-1906	17
1907-1919	18
1920-1933	19
1934-1946	20
1947-1959	21
1960-1973	22
1974-1986	23
1987-1999	24
2000-2083	25
2084-2166	26
2167-2249	27
2250-2333	28
2334-2416	29
2417-2499	30
2500-8500	31
8501-8523	30
8524-8547	29
8548-8571	28
8572-8595	27

TABLE D-1: EXPANSION OF TABLE 1 FOR EVERY 1˚ CHANGE IN FIRING ANGLE (CONTINUED)

Engine Speed Range (RPM)	Firing Angle (θ˚)
8596-8619	26
8620-8642	25
8643-8666	24
8667-8690	23
8691-8714	22
8715-8738	21
8739-8761	20
8762-8785	19
8786-8809	18
8810-8833	17
8834-8857	16
8858-8880	15
8881-8904	14
8905-8928	13
8929-8952	12
8953-8976	11
8977-10000	10

This method increases the performance of the CDI program by simply finding the firing angle against the range of RPM at the cost of requiring more Flash memory. If enough memory is available for implementing all the maps as a look-up table, and performance is of prime importance, the above look-up table method can be used. For example, for a RPM value of 1915, the firing angle from Table D-1 is 18˚.

APPENDIX E: ENGINE AND VEHICLE SPEED CALCULATION

The engine speed is calculated from Equation 8 which is in Hz or RPS (rotations per second). This value is multiplied by 60 to convert it to engine speed (RPM). This rotation per second or speed in Hz can also be used to find the angular speed or velocity of the engine given as follows:

EQUATION E-1: ENGINE ANGULAR VELOCITY

$$\omega_{Engine} = 2\pi \times f_{signal}$$

Where: ω_{Engine} is the angular velocity of engine in radians/second.

f_{signal} is the engine speed in Hz.

2π is the length of circumference of a circle in radian angles or one complete rotation of circle in radian.

Equation E-1 can be used to find the linear speed of the engine as shown below:

EQUATION E-2: ENGINE LINEAR VELOCITY

$$V_{Engine} = r \times \omega_{Engine}$$

Where: V_{Engine} is the linear velocity or speed of the engine piston in meters/second

r is the radius which is equal to half of the distance between the TDC and BDC position of the piston in meters

The engine rotation is mechanically coupled to automobile wheels through gear trains. The mechanical coupling through gears has a ratio called gear turn ratio, which measures how fast the wheel will rotate for one complete rotation of the engine. The gear turn ratio value can be used to convert the engine linear velocity into automobile linear velocity or speed as follows:

EQUATION E-3: AUTOMOBILE LINEAR VELOCITY

$$V_{Automobile} = V_{Engine} \times G$$

Where: $V_{Automobile}$ is the automobile linear velocity or speed in meters/second.

G is the gears' turn ratio of engine rotation and the automobile wheel rotation.

APPENDIX F: MPLAB® CODE CONFIGURATOR (MCC)

The MPLAB® Code Configurator (MCC) plug-in for MPLAB X can be used to configure the peripherals in PIC MCU. The MCC provides graphical user interface (GUI) tools to easily understand the configuration and select the required settings. This reduces the time to development. The MCC also provides some built-in functions for working with specific peripherals.

Figure F-1 to Figure F-5 shows the configuration of the PIC16F1615 device peripherals such as AT, CLC, Math Accelerator and SMT, respectively, used in the CDI implementation.

FIGURE F-1: AT CONFIGURATION IN MCC FOR PIC16F1615

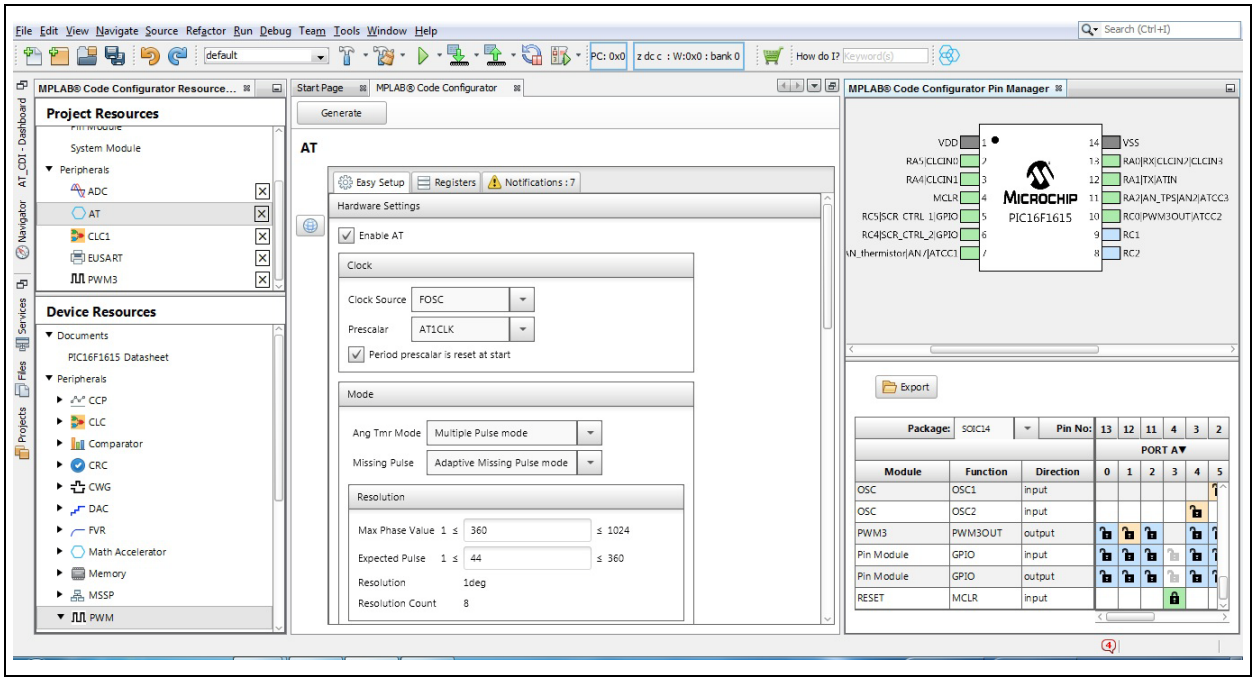


FIGURE F-2: CLC CONFIGURATION IN MCC FOR PIC16F1615

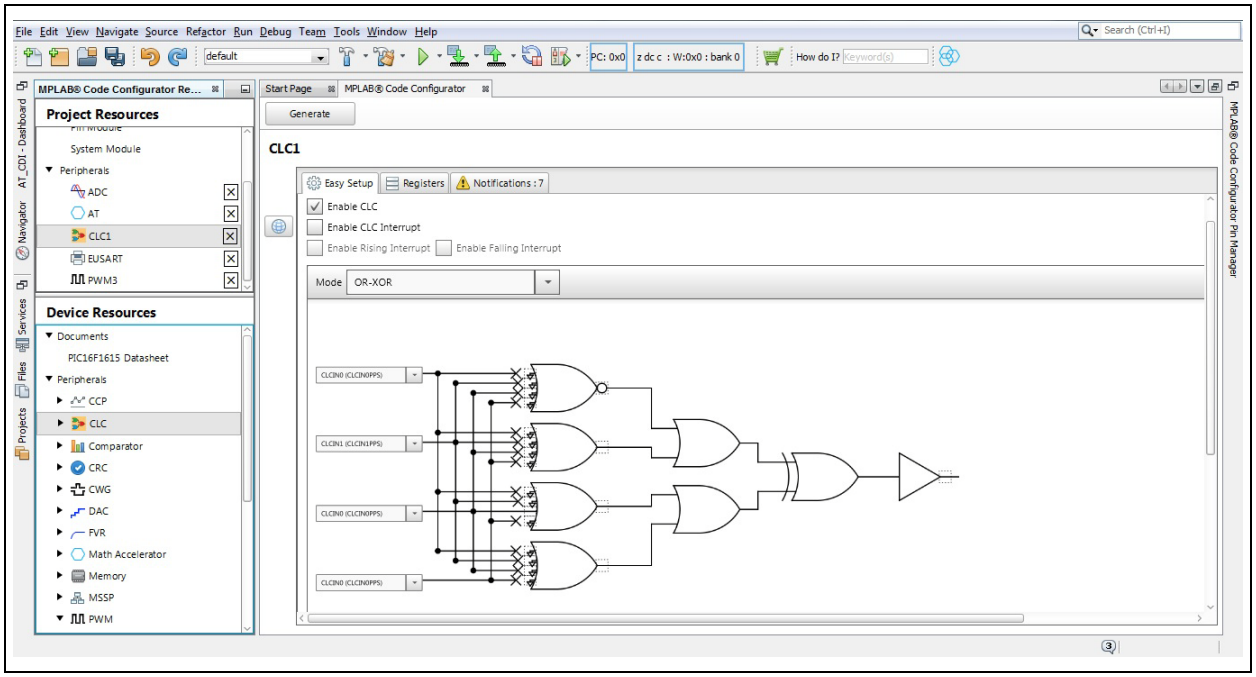


FIGURE F-3: MATH ACCELERATOR IN MCC FOR PIC16F1615

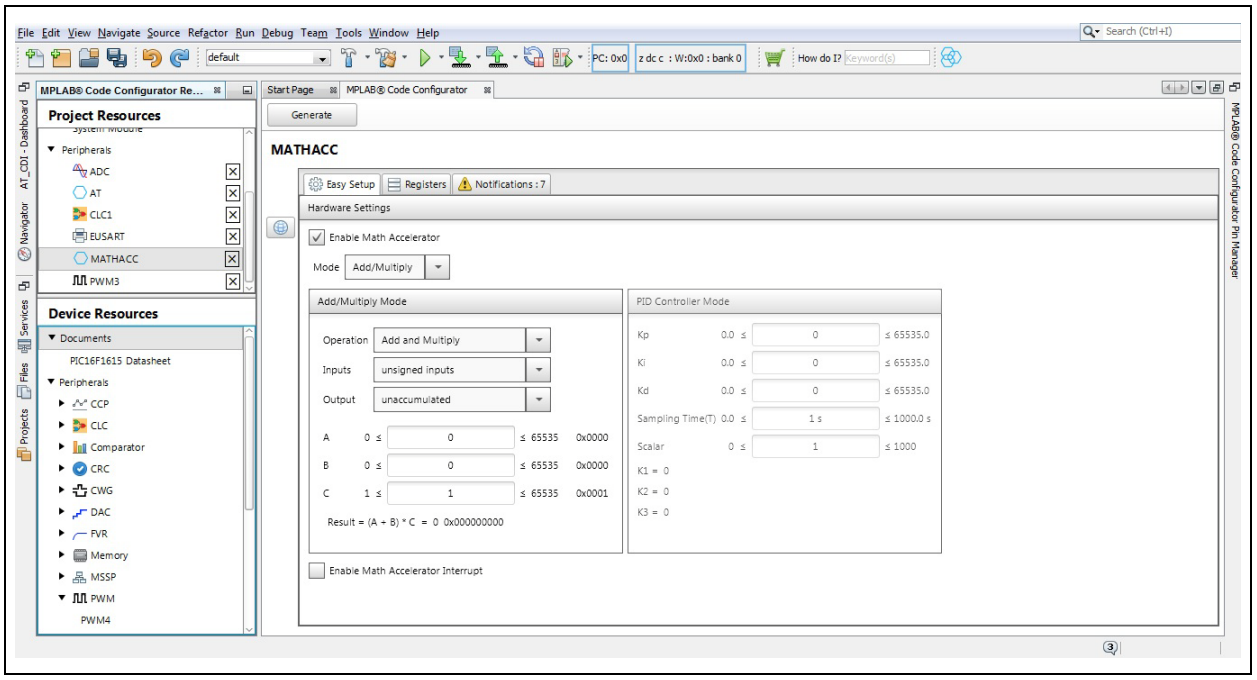
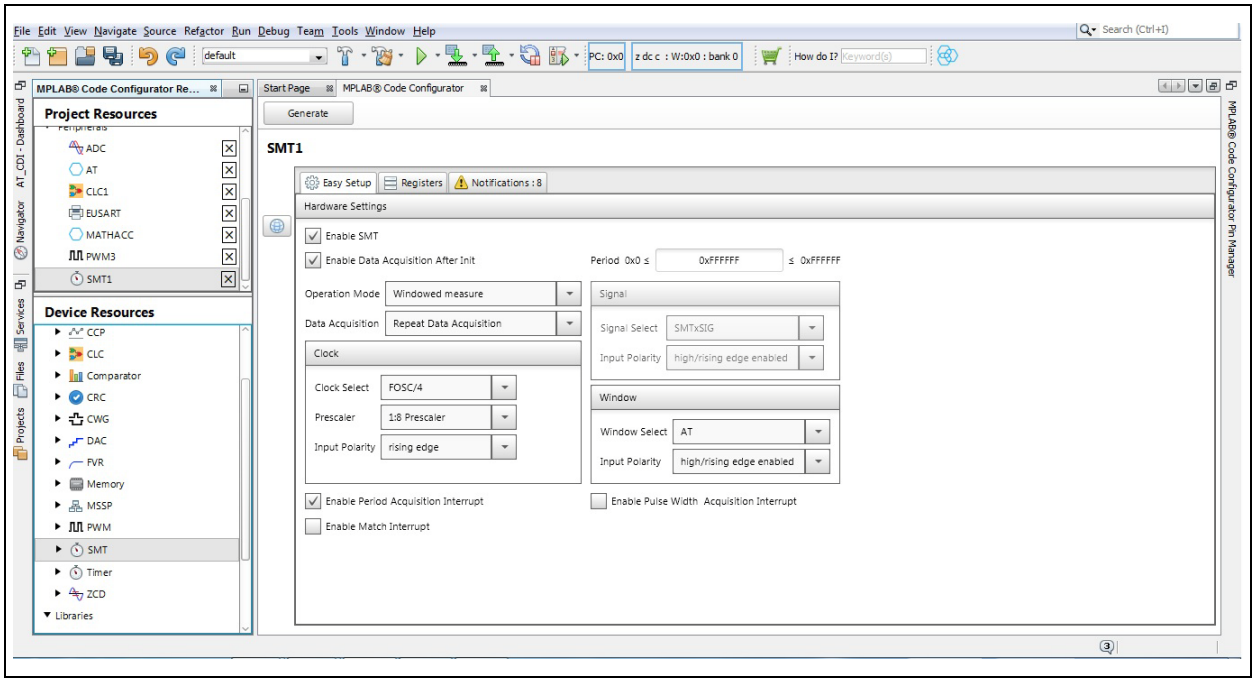
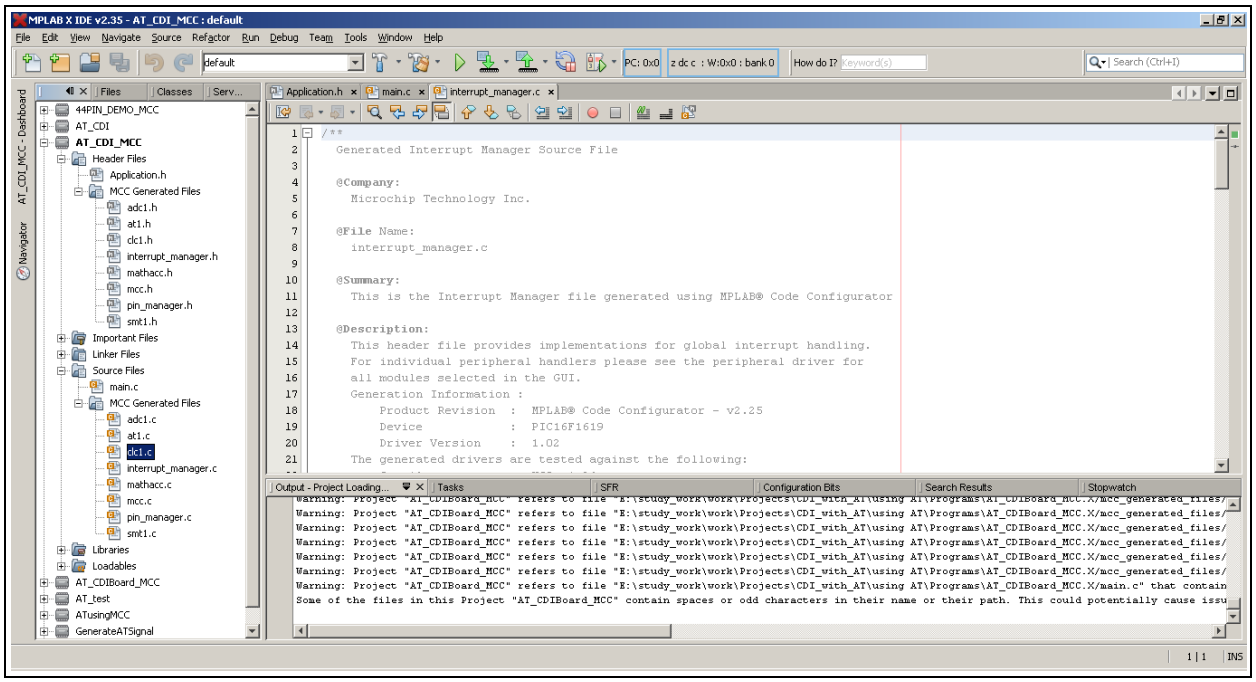


FIGURE F-4: SMT1 CONFIGURATION IN MCC FOR PIC16F1615



To install MCC, go to Tools → Plugins Available Plugins → MPLAB X Code Configurator → Install. To start and use the MCC after installation, go to Tools → Embedded → MPLAB Code Configurator.

FIGURE F-5: MCC GENERATED FILES AND FOLDER STRUCTURE



After all the settings are selected in MCC, click the *Generate Code* button to generate the peripheral code, and create a header and a C file for every peripheral separately. Figure F-2 shows the files created and arranged under the *MCC Generated Files* folder for CDI implementation using AT. The programmer should add the code wherever it is required in these files.

APPENDIX G: ENGINE TUNING USING PC APPLICATION

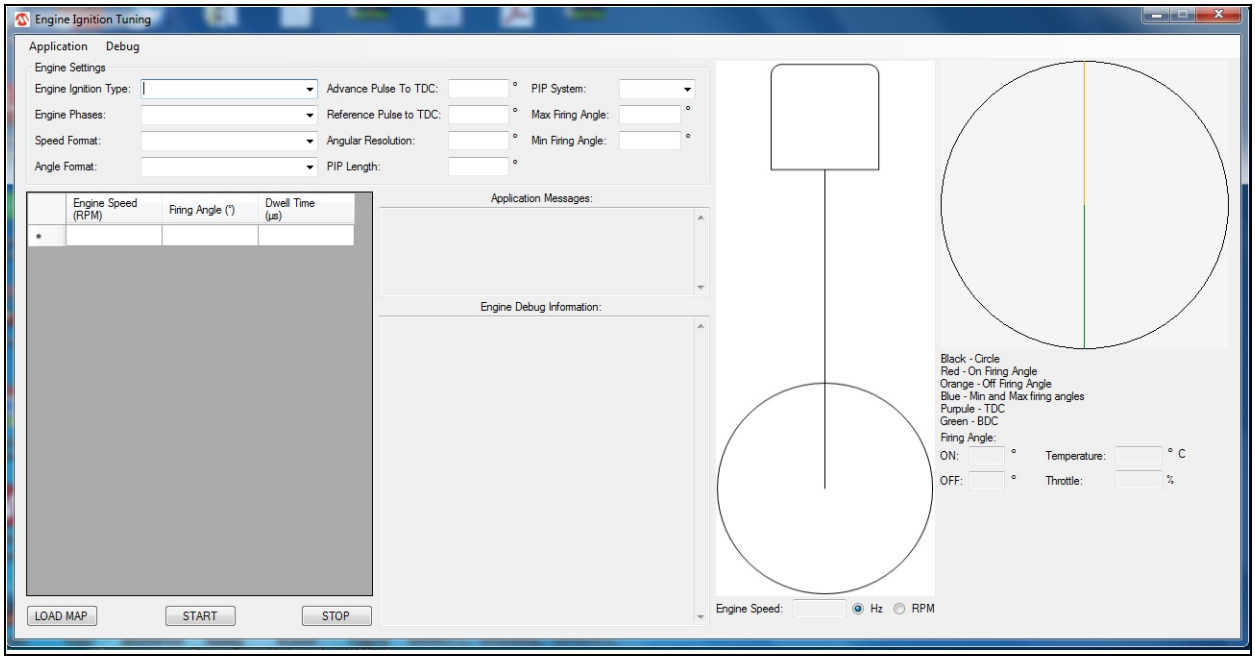
This section explains the PC-based GUI application which is developed for the debugging and the engine tuning of a CDI system.

As explained in the previous sections, microcontroller-based CDI applications have a timing map/look-up table to handle the advance of the ignition depending upon the engine speed, temperature and throttle position. These are based on the ignition timing advance characteristics specified by the engine manufacturers. Tuning a CDI system by varying firing angles at different RPMs helps us to check and improve the efficiency, thereby enhancing the performance of the engine.

Installation and Usage of the Desktop Application

1. Start the `IgnitionSystemTuning.exe` application. Figure G-1 shows the GUI of the tuning application. The GUI has an Engine Settings area. In this area, the user enters engine parameters and the Angular Timer parameters for timing map generation and tuning. These values can be modified by the user.

FIGURE G-1: DEBUGGING/TUNING APPLICATION AT START-UP



As shown in Figure G-1, the GUI has a table on the left consisting of three columns:

- Engine speed in RPM
- Firing Angle in Degrees
- Dwell Time in µs

There are two text boxes labeled “Application Messages” and “Application Debug Information”. “Application Messages” displays errors and any messages during the operation. “Application Debug Information” displays the user with real-time engine parameters like engine speed, firing angle, temperature and throttle position.

2. The application has a menu labeled “Application” on the left topmost corner. Upon clicking on it, it displays several menu options such as “Clear All” and “File”. The “Clear All” menu option clears all settings done by the user in the Engine Settings area and the table. It resets the GUI to it’s default state. The File menu has two options: Save and Load. The Save option saves the engine information entered by the user in a file at a user-specified location. The Load option loads the engine information file from a user-defined location.

3. The following list describes the functions of various buttons on the GUI:

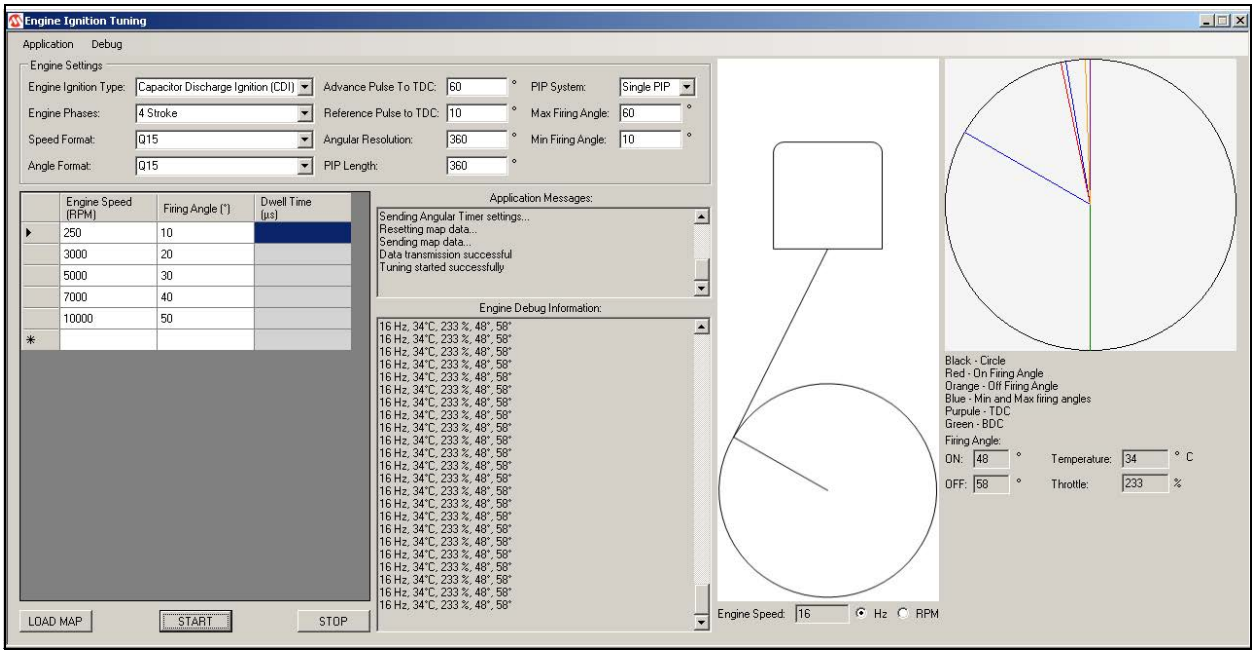
- LOAD MAP – When this button is clicked, the application finds any connected CDI/TCI board and starts sending the tuning map data and engine parameters set by the user. Figure G-2 illustrates the messages received while loading the map using the GUI. If the data transmission is successful, a “Data Transmission is successful” message is displayed in the application messages box. If unsuccessful, an error message indicating the nature of the error is displayed.

- The following errors can occur:

  “Error: Communication time-out Error” – This error message occurs when a communication time out happens due to causes such as baud rate mismatch or any other incorrect serial port settings. The application attempts to resend the data in case of an error.

  “Error: Checksum error” – This error message occurs when checksums do not match due to erroneous data transmission or reception. The GUI tries to retransmit up to three times to recover from this kind of error. If the error is resolved then a “Data Transmission is successful” message appears on the Application Message text box. If the error persists despite three attempts to retransmit, then data transmission is halted completely. The user has to manually check and resolve any data discrepancies and then press the **Load Map** button.

FIGURE G-2: OCCURRENCE OF COMMUNICATION ERROR MESSAGE AND ITS RESOLUTION

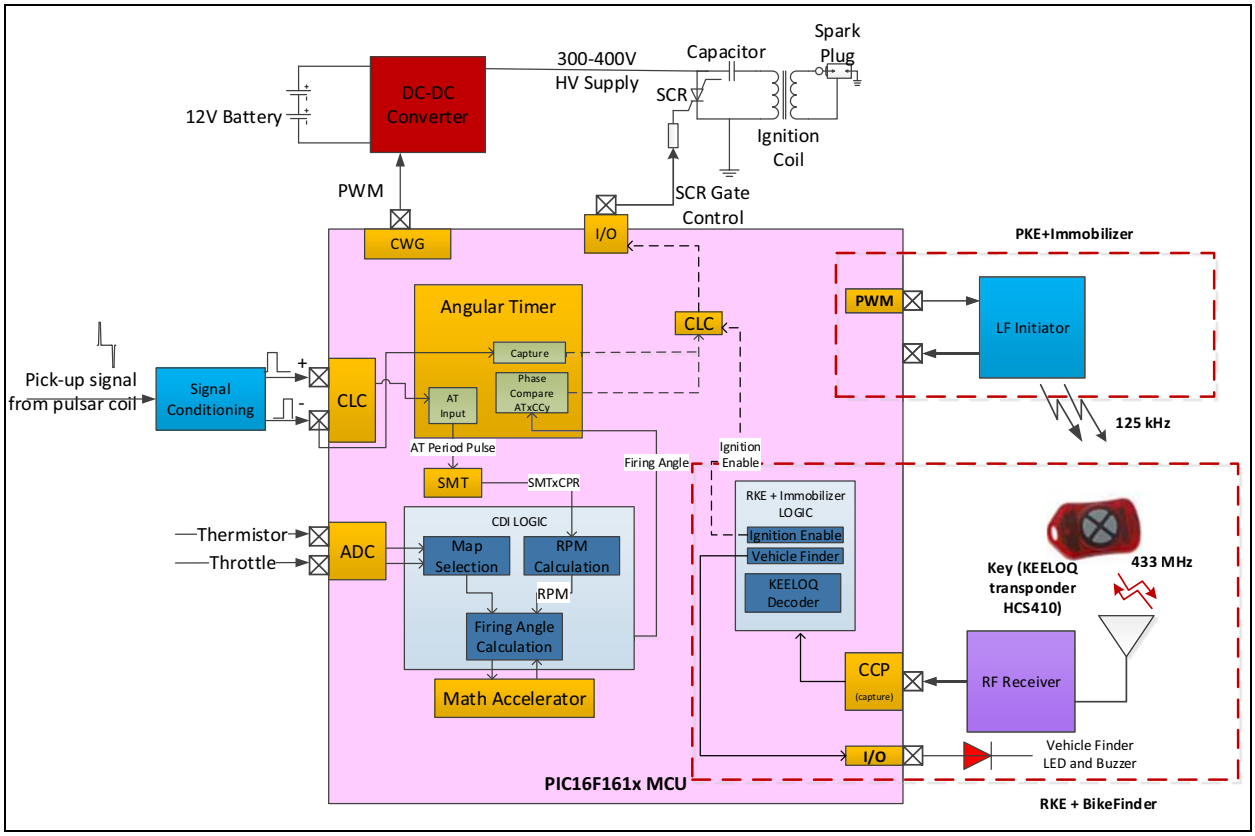


- START – Clicking this button will start the tuning operation with CDI/TCI board. The board sends real-time data and it is displayed on the GUI. When the tuning starts successfully, the “Tuning started successfully” message is displayed.
- STOP – Clicking this button will stop the tuning operation with CDI/TCI board and will reset the board for normal operation. If the user tries to start tuning while the operation is already ON, then the message “Tuning mode already on” appears. If the user tries to stop the tuning after it has been stopped before, then the message “Tuning mode already off” appears.

APPENDIX H: CDI WITH REMOTE KEYLESS ENTRY, BIKE FINDER AND IMMOBILIZER

One of the advantages of using the CIPs of the PIC16F1615/9 for the implementation of the CDI system is that it frees up the CPU time and also reduces the code size. This can be utilized for adding different enhancement features to the system, such as remote keyless entry (RKE) (i.e., an anti-theft remote lock, a bike finder and an immobilizer to name a few). This appendix explores the possibility of implementing the CDI system with integrated RKE, bike finder and immobilizer functionalities.

FIGURE H-1: BLOCK DIAGRAM OF CDI WITH RKE, BIKE FINDER AND PASSIVE KEYLESS ENTRY FUNCTIONALITIES



Remote Keyless Entry and Bike Finder System

The remote keyless entry (RKE) system is an electronic lock that controls the access to a vehicle without using a traditional key. The RKE system can be used for unlocking doors, as well as starting the engine. The RKE system consists of two blocks: a keyfob/ remote and an RF receiver on the vehicle (Figure H-1).

The keyfob has three different push buttons for lock, unlock and bike finder functionalities. For every different key press, the transponder in the key sends a stream of 64 or 128 bits encoded data to the key's RF transmitter. The transmitter modulates the carrier and the data is radiated through a simple printed-circuit loop antenna. The wireless carrier frequency by the RKE system is currently 315 MHz in the US/Japan and 433.92 MHz (ISM band) in Europe and Asia.

Keyfob uses an HCS410, which is both an LF transponder (battery less) plus KEELOQ® encoder in a single chip. The same keyfob can be used both for the RKE and the immobilizer functionalities.

For details on how to use an HCS410 for RKE, bike finder and keyfob applications, refer to the application note *AN744, "Modular Mid-range PICmicro® KEELOQ® Decoder in C"*, (DS00000744).

RF Receiver in Vehicle Side

The data sent by keyfob is captured by the RF receiver on the vehicle side. The microcontroller decodes the data captured. It then sends an appropriate message to lock or unlock the vehicle. If the vehicle is parked in a crowded parking lot, the bike finder key can be pressed to locate the vehicle. If the bike finder key is pressed, then the flash lights on the vehicle will blink along with the buzzer beep, so the user can easily locate the vehicle.

For details on how to implement the RF receiver section, refer to *MICRF229-433 Evaluation Board User's Guide*.

Passive Keyless Entry and Immobilizer

In a passive keyless entry system, the base unit inside the vehicle initiates communication by transmitting out a low-frequency signal between fixed time intervals to search for a transponder (key) in the field. When located, the transponder in the vehicle automatically responds to the base unit. If a valid authentication response is received, then the base unit performs the necessary action.

An immobilizer is an electronic security feature fitted to an automobile, that prevents the engine from running unless a correct key is transmitted. When the keyfob is in the proximity of the vehicle, it broadcasts a unique binary code. This code is read by the automobile's immobilizer unit. When the immobilizer unit determines that the coded key is both current and valid, it enables the ignition.

When the low-frequency (LF) initiator detects a trigger input, a coded 125 kHz message is transmitted. Any transponder within range of this signal receives this message and validates the coded data field. If the initiator is recognized, an RF KEELOQ encoded message is transmitted. A standard RKE receiver decodes the packet and, if recognized, appropriate action is taken.

A key or a transponder is a feature of the base unit. This allows the addition of new transmitters into the system without having to reprogram from the outside. During this process, the base station reads the transponder's serial number, calculates the transponder's key, and gets the transponder's synchronization counter. The next time the base unit receives a transmission or reads the serial number from a transponder, the base unit searches through its 'database' of serial numbers. If the base unit finds the newly acquired serial number, the code hopping portion of a transmission (or response from the transponder) is decrypted. The resulting value is then compared to the expected value. An LED can be lit to indicate if the data is valid.

For more information on the KEELOQ code hopping, refer to the technical brief *TB003, "An Introduction to KEELOQ® Code Hopping"*, (DS91002).

APPENDIX I: REFERENCES

1. *TB3129 – Signal Measurement Timer on PIC® Microcontrollers* (DS90003129)
2. *TB3133 – Configurable Logic Cell on PIC® Microcontrollers* (DS90003133)
3. *TB3134 – 16-Bit Multiplication and Addition by using the Math Accelerator Peripheral on PIC16(L)F161X* (DS90003134)
4. *TB3118 – Complementary Waveform Generator Technical Brief* (DS90003118)
5. *MPLAB® Code Configurator User's Guide* (DS40001725)
6. *AN744 – Modular Mid-range PICmicro® KEELOQ® Decoder in C* (DS00000744)
7. *MICRF229-433 Evaluation Board User's Guide*
8. *TB003 – An Introduction to KEELOQ® Code Hopping* (DS91002)
9. *AN2095 – Transistor Coil Ignition with Integrated Remote Keyless Entry and Immobilizer Using PIC® Microcontrollers* (DS00002095)

NOTES:

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
═ ISO/TS 16949 ═

Trademarks

ISBN: 978-1-5224-0452-1

Worldwide Sales and Service

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://www.microchip.com/support
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Austin, TX
Tel: 512-257-3370

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Cleveland
Independence, OH
Tel: 216-447-0464
Fax: 216-447-0643

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Novi, MI
Tel: 248-848-4000

Houston, TX
Tel: 281-894-5983

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

New York, NY
Tel: 631-435-6000

San Jose, CA
Tel: 408-735-9110

Canada - Toronto
Tel: 905-673-0699
Fax: 905-673-6509

ASIA/PACIFIC

Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon

Hong Kong
Tel: 852-2943-5100
Fax: 852-2401-3431

Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing
Tel: 86-10-8569-7000
Fax: 86-10-8528-2104

China - Chengdu
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

China - Chongqing
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

China - Dongguan
Tel: 86-769-8702-9880

China - Hangzhou
Tel: 86-571-8792-8115
Fax: 86-571-8792-8116

China - Hong Kong SAR
Tel: 852-2943-5100
Fax: 852-2401-3431

China - Nanjing
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

China - Qingdao
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

China - Shanghai
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

China - Shenyang
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

China - Shenzhen
Tel: 86-755-8864-2200
Fax: 86-755-8203-1760

China - Wuhan
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

China - Xian
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

ASIA/PACIFIC

China - Xiamen
Tel: 86-592-2388138
Fax: 86-592-2388130

China - Zhuhai
Tel: 86-756-3210040
Fax: 86-756-3210049

India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

India - Pune
Tel: 91-20-3019-1500

Japan - Osaka
Tel: 81-6-6152-7160
Fax: 81-6-6152-9310

Japan - Tokyo
Tel: 81-3-6880- 3770
Fax: 81-3-6880-3771

Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

Taiwan - Hsin Chu
Tel: 886-3-5778-366
Fax: 886-3-5770-955

Taiwan - Kaohsiung
Tel: 886-7-213-7828

Taiwan - Taipei
Tel: 886-2-2508-8600
Fax: 886-2-2508-0102

Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Dusseldorf
Tel: 49-2129-3766400

Germany - Karlsruhe
Tel: 49-721-625370

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Italy - Venice
Tel: 39-049-7625286

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Poland - Warsaw
Tel: 48-22-3325737

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

Sweden - Stockholm
Tel: 46-8-5090-4654

UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820

07/14/15