Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

**Continuity of document content**
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

**Continuity of ordering part numbers**
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.
Introduction

Analog Devices’ TMP05 and TMP06 sensors are monolithic temperature sensors that generate a pulse-width modulated (PWM) serial digital output. This output varies in direct proportion to the ambient temperature of the devices. The high period (TH) of the PWM remains static over all temperatures, while the low period (TL) varies. It offers a high temperature accuracy of ±1°C from 0°C to 70°C with excellent transducer linearity. The digital output of the TMP05 is CMOS/TTL-compatible and, therefore, can be interfaced directly to PSoC. The digital output of the TMP06 is open-drain and requires a pull-up resistor for proper operation. Throughout the rest of this document, any references to TMP05 also apply to the TMP06 sensor.

The TMP05 Digital Temperature Sensing solution can be used in thermal management solutions for base-stations, telecommunications, server and storage applications. Typical applications may include, but are not limited to, areas where remote temperature sensing, environmental control systems, computer thermal monitoring, thermal protection, industrial process control, and power-system monitoring and management are required.
The TMP05 Sensor can be easily interfaced to PSoC 1 using a 1 or 2-wire serial interface as shown in Figure 1 and Figure 2. The project attached with the application note uses a 16 bit timer, 1 input pin, 1 output pin (depending on the mode in which TMP05 is configured) and the GPIO interrupt service routine for interfacing with the TMP05 sensor. This enables the designer to implement many other system management functions in the same device.
**TMP05 – Modes of Operation**

The TMP05 sensor has three modes of operation listed below:

1. **Continuous conversion mode** - The sensor outputs a temperature dependent PWM signal continuously.
2. **Daisy-chain mode** - Multiple TMP05 sensors are chained together.
3. **One shot mode** - The sensor outputs a temperature dependent PWM signal on request (as shown in Figure 3).

A three-state FUNC input pin, sampled at power-up, determines the mode in which the device operates. Setting the FUNC pin to a high state allows multiple TMP05s to be connected together in daisy-chain mode. In that mode, multiple TMP05 temperature sensors are connected together enabling PSoC to read all the sensors through the same 2 pin interface. In this mode, PSoC generates a “Start” pulse that begins the temperature-to-PWM conversion cycle. The output of each TMP05 sensor begins with the PWM outputs from all previous TMP05 sensors in the daisy-chain followed by the PWM output from the current sensor; the signal chain is terminated by a start pulse. The temperature to PWM conversion then stops until the controller generates another start pulse. The system level architecture is shown in Figure 4. For further details on the modes refer to TMP05 device datasheet.

![Figure 3. TMP05 PWM_OUT Line in One Shot Mode](image)

![Figure 4. Multiple TMP05 Sensors Daisy-Chained](image)

This application note focuses on how to use the TMP05 Digital Temperature Sensor with CY8C28xxx device by working through some example projects on the CY8CKIT-001 PSoC Development Kit (DVK). The examples show how to use the attached project to monitor the temperature of multiple TMP05 devices reliably. Supporting other management protocols and higher level functionality is beyond the scope of this application note.
Design Considerations When Using TMP05 Sensors in Daisy-Chain Mode

There are some timing constraints to be observed when operating TMP05 sensors in daisy-chain mode. PSoC needs to generate the conversion start pulse on the CONV/IN pin of the sensor. The start pulse lets the first TMP05 part know that it should now start a conversion and output its own temperature. The pulse width of the start pulse should be less than 25 μs but greater than 20 ns. These constraints are handled automatically in the project and are described in this application note for reference purposes only. Once the part has output its own temperature, it adds a start pulse for propagation to the next part in the daisy-chain.

Figure 5 and Figure 6 show the input and output waveforms for the first sensor in the daisy-chain (taken from the TMP05 device datasheet):

Figure 5. TMP05 Start Pulse Waveform

![Image of TMP05 Start Pulse Waveform]

Figure 6. TMP05 PWM Output Waveform

![Image of TMP05 PWM Output Waveform]

When sensors are daisy-chained, downstream sensors use the rising edge of the PWM signal from the previous sensor as the start pulse. Once detected, it initiates a conversion and inserts its own result at the end of the incoming PWM signal and then adds a start pulse for the next sensor in the daisy-chain. PSoC receives the sensor terminal input, the PWM representing sensor 1 first, followed immediately by the PWM representing sensor 2. A start pulse of 17 μs terminates the process.

System Architecture in PSoC

The resources used to interface the TMP05 sensor in CY8C28xxx are shown in Figure 7. As shown in the figure, only two digital blocks are used to implement a 16 bit timer for the interface (remaining resources used in the project are optional and are used to display/communicate the temperature data to the user). The rest of the design is implemented in firmware and GPIO ISR.

Figure 7. Resources Utilized for the Interface

![Image of Resources Utilized for the Interface]
**Continuous Conversion Mode**

To configure the TMP05 sensor in continuous conversion mode refer to the sensor datasheet or Figure 8. In this mode, the sensor outputs square wave continuously whose low time depends on the ambient temperature. To measure the pulse width in CY8C28xxx, the 16 bit timer clocked by ILO (32 kHz) along with PSoC GPIO interrupt is used. When PSoC is triggered to start the measurement, it starts the timer and enables ChangeFromRead interrupt on the pin which is reading the TMP device’s PWM output. When there is a fall edge or a rise edge, the timer value is captured in the ISR. The high and low pulse time are calculated in the ISR and are in turn used to calculate the temperature. Flow charts in Figure 9, Figure 10, Figure 11, and Figure 12 describe the firmware.

Figure 8. TMP05 in Continuous Mode

![Figure 8. TMP05 in Continuous Mode](image)

**Figure 9. TMP05_Start Function in Continuous Mode**

![Figure 9. TMP05_Start Function in Continuous Mode](image)

**Figure 10. Get Temperature Function in Continuous Mode**

![Figure 10. Get Temperature Function in Continuous Mode](image)
Figure 11. Timer Overflow ISR

1. Set Timer Overflow flag
2. Timer Overflow ISR()
   - Timer Overflow
   - ISR()
   - RETI

Figure 12. GPIO ISR for Continuous Mode

1. GPIO_ISR()
   - TempVar = Read_Timer()
   - Is this the 1st Interrupt?
     - Yes: RiseEdgeTimer = TempVar
     - No: RiseEdgeTimer = TempVar
     - Timer Overflow?
       - Yes: TempData |= 1
       - No: TempData |= 2
       - TempData &= ~1
       - Set Data Ready flag for temperature measurement
   - Timer_Start and Clear First Time enter flag
   - Timer Overflow?
     - Yes: PulseHighWidth = PERIOD + RiseEdgeTimer
     - No: PulseHighWidth = PERIOD + RiseEdgeTimer
     - PulseLowWidth = PERIOD + FallEdgeTimer
     - FallEdgeTimer = TempVar
     - Timer Overflow?
       - Yes: PulseHighWidth = PERIOD + RiseEdgeTimer
       - No: PulseHighWidth = PERIOD + RiseEdgeTimer
     - TempData |= 2
     - TempData &= ~1
     - Set Data Ready flag for temperature measurement
   - RETI
   - RETI
One Shot Mode

To configure the sensor in one shot mode see Figure 13. In one shot mode, the TMP05 outputs one square wave representing temperature when requested. The firmware implementation for that is shown in Figure 14 to Figure 17.

Figure 13. TMP05 in One Shot Mode

Figure 14. TMP05 Start Function in One Shot Mode

Figure 15. TMP05 Trigger Function in One Shot Mode
Figure 16. GPIO ISR for One Shot Mode

GPIO_ISR()

TempVar = ReadTimer();

Is PWM_OUT High?

Yes

Disable Timer and INT on PWM_OUT line

RiseEdgeVal = TempVar

FallEdgeVal = TempVar

lowWidth = FallEdgeVal - RiseEdgeVal

Set DataReady flag

RET

No

highWidth = RiseEdgeVal - FallEdgeVal

Figure 17. Get Temperature Function in One Shot Mode

Get Temperature()

Is DataReady?

Yes Temperatur (in C) = 421 – 751 * TH/TL

No

Temperature = DATA_NOT_READY

RET
**Daisy Chain Mode**

To configure the sensor in daisy chain mode see Figure 18. In this mode, multiple TMP05s are connected together and allow one input line to be the sole receiver of all temperature measurements. In this mode, the CONV/IN pin of TMP operates as the input of the daisy chain. In addition, conversions take place at the nominal conversion rate of TH/TL = 40 ms/76 ms at 25°C. The PSoC 1 implementation of the daisy chain mode is shown in Figure 19 to Figure 23.

![Figure 18. Two TMP05s in Daisy Chain Mode](image)

**Figure 18. Two TMP05s in Daisy Chain Mode**

**Figure 19. Start Function in Daisy Chain Mode**

```
TMP_Start()

Set drive mode based on Sensor Type (TMP05 or 06)

Set drive mode of CONV_IN to Strong

Configure the 16 bit timer with ILO as input clock

Enable Global interrupts

RET
```

**Figure 20. Trigger Function in Daisy Chain Mode**

```
TMP_Trigger()

Drive the CONV_IN line high

Give 10-15 µs delay

Enable Edge triggered INT on PWM_OUT

RiseEdge = PERIOD

Start Timer

Drive the CONV_IN line low

RET
```
Figure 21. GPIO ISR for Daisy Chain Mode

Figure 22. Get Temperature Function in Daisy Chain Mode

Figure 23. Get All Temperatures Function in Daisy Chain Mode
Hardware

The hardware used to verify the interface consists of the below kits:
1. CY8CKIT – 001 - PSoC DVK
2. CY8CKIT – 036 - Thermal management EBK
3. CY8CKIT – 020 - PSoC CY8C28 Family Processor module kit (comes with CY8CKIT – 001)
4. CY8CKIT – 002 - PSoC Miniprog3 Program and Debug kit (comes with CY8CKIT – 001) OR CY3217 – MiniProg1.
5. 12 V, 1 A DC power supply adapter.(comes with CY8CKIT – 001)

The respective examples cover how to configure the hardware used in various modes.

Associated Project Overview

Distributed with this application note is a PSoC Designer project ZIP file that contains the project implementing the TMP05 sensor interface in CY8C28 family. The same project can be cloned to other PSoC 1 families by following the instructions provided in the section Porting the project to other PSoC 1 devices.

Resources Utilized

PulseWidthTimer (Timer16 UM) - Required
This timer is used to measure the pulse high and low widths. It uses a Timer16 user module to implement the function. The parameters are configured as in Figure 24. The Input clock selected is ‘CPU_32_KHz’ to make it independent of the ‘sysclk’ dividers and to measure the slow PWM signal from the sensor.

Figure 24. PulseWidthTimer Parameters

LCD_Char (LCD) - Optional
This uses a software character LCD UM to display temperature in Celsius. The parameters for the UM is shown in Figure 25.

Figure 25. LCD_Char Parameters

Comm (EzI2Cs UM) - Optional
The I²C Slave interface for communicating the temperature to an external host or system. The parameters are configured as shown in Figure 26.

Figure 26. Comm Parameters

SleepTimer (SleepTimer UM) – Optional
The timer used to periodically sample the temperature of the sensor(s). The parameters are configured as shown in Figure 27.

Figure 27. SleepTimer Parameters

Firmware Overview

Most of the files, functions, variables, macros defined in the project are self explanatory and well documented in the comments. This document does not cover all those files, functions, variables and macros defined in the project. Some of the important macros and functions used in the project are covered below.
Changing the TMP Type:

Selecting which TMP sensor is being used in the design.

1. Browse to 'TMP_05_06.h' file as show in Figure 28.

2. Search for 'TEMPERATURE_SENSOR' macro in the file as in Figure 29.

3. Set its value to 'TMP05' or 'TMP06' based on the sensor used. The sensors in CY8CKIT – 036 are TMP05 sensors.

Changing TMP Mode:

1. Browse to TMP_05_06.h file as mentioned previously.

2. Search for 'TMP_MODE' macro in the file (Figure 30).

3. Set the macro according to the mode in which the TMP is configured. It is important to note that this configuration be done externally as shown in Figure 8, Figure 13, and Figure 18. The sensors by default are configured to be in 'daisy_chain' mode in CY8CKIT – 036.

Changing the TMP Conversion Rate:

1. Browse to the TMP_05_06.h file as previously described.

2. Search for the 'CONVERSION_RATE' macro in the file (Figure 31).

3. Set the macro as per the conversion rate configured. It is important to note that this is done externally as described in the code comment/sensor datasheet. In CY8CKIT – 036; they are configured in nominal rate. This parameter is valid only for 'ONE_SHOT' and 'CONTINUOUS' mode and ignored in the 'DAISY_CHAIN' mode.
Changing the Number of Sensors in Daisy Chain:
1. Browse to the TMP_05_06.h file as previously described.
2. Search for the ‘NO_OF_SENSORS’ macro in the file (as shown in Figure 32).

Figure 32. Number of Sensors in Daisy Chain

3. Set the macro as per the number of TMP sensors daisy chained in the system. In CY8CKIT – 036, a max of two sensors can be daisy chained.

Changing PWM Input Pin in the Device:
1. Browse to the TMP_05_06.h file as previously described.
2. Search for the below set of macros in the file (as shown in Figure 33).

Figure 33. PWM Input from TMP Sensor

3. Change the port registers, port number, and pin mask based on the port pin selected. In CY8CKIT – 036 at PORT A, the PWM input is connected to Port 4 Pin 0.

GPIO ISR:
The GPIO Interrupt Service Routine is utilized for capturing the timer at the rising and falling edge of the PWM pin. The GPIO_ISR function defined in ISR.c file is the default GPIO ISR for the entire device. The ISR is hard coded in the ‘boot.tpl’ (available in project directory) file as shown in Figure 35.

Figure 35. GPIO ISR in boot.tpl

3. Change the port registers, port number, and pin mask based on the port pin selected. In CY8CKIT – 036 at PORT A, the CONV/IN pin is connected to Port 4 Pin 0.

Changing CONV/IN Pin for Daisy Chain Mode:
1. Browse to the TMP_05_06.h file as previously described.
2. Search for the below set of macros in the file (Figure 34).

Figure 34. CONV/IN Pin from the TMP Sensor

3. Change the port registers, port number, and pin mask based on the port pin selected. In CY8CKIT – 036 at PORT A, the CONV/IN pin is connected to Port 4 Pin 1.

The ISR file is written in such a manner that allows for placement of the ISR code for other GPIO interrupts in the system. It is recommended to review the comments provided in the function header for a better understanding of how to use the ISR for other purposes as well.
Table 1 summarizes the important macros in the project.

Table 1. List of important Macros in Firmware

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEMPERATURE_SENSOR</td>
<td>Selects the type of TMP sensor used</td>
<td>#TMP05, #TMP06</td>
</tr>
<tr>
<td>TMP_MODE</td>
<td>Selects the mode in which TMP sensor is configured</td>
<td>#DAISY_CHAIN, #CONTINUOUS, #ONE_SHOT</td>
</tr>
<tr>
<td>CONVERSION_RATE</td>
<td>Selects the conversion rate in which TMP sensor is configured</td>
<td>#CONV_NOMINAL, #CONV_HIGH, #CONV_LOW</td>
</tr>
<tr>
<td>NO_OF_SENSORS</td>
<td>Sets the number of sensors chained in the daisy chain mode</td>
<td>1 to 255</td>
</tr>
<tr>
<td>CONV_OUT pin settings</td>
<td>These are set of macros to change pin drive mode, data registers of the CONV/IN pin in PSoC 1</td>
<td>Refer this section</td>
</tr>
<tr>
<td>PWM_IN pin settings</td>
<td>These are set of macros to change associated registers of the PWM input pin in PSoC 1</td>
<td>Refer this section</td>
</tr>
</tbody>
</table>

Table 2. List of important Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void TMP_Start(void)</td>
<td>This function initializes various settings of TMP05 sensor interface. Initialization includes setting drive modes, initial values of timers, state of pin, enabling interrupts.</td>
</tr>
<tr>
<td>void TMP_Trigger(void)</td>
<td>This function generates the trigger/start pulse for enabling the TMP to generate the corresponding PWM output in ONE_SHOT and DAISY_CHAIN mode.</td>
</tr>
<tr>
<td>INT GetTemperature(void)</td>
<td>This function calculates and returns the temperature value in 16 bit format – MSB 8 bit constitutes the integer part and LSB 8 bit is the decimal part.</td>
</tr>
<tr>
<td>INT GetTemperature(BYTE snsNo)</td>
<td>This function calculates and returns the temperature value of the sensor requested (passed parameter) in daisy chain mode</td>
</tr>
<tr>
<td>void GetAllTemperatures(void)</td>
<td>This function calculates the temperature values of all the sensors in daisy chain mode. The values are stored in a 16 bit array.</td>
</tr>
<tr>
<td>GPIO_ISR</td>
<td>This is the ISR function where the timer value is captured and the PWM high and low widths are measured.</td>
</tr>
<tr>
<td>TimerOverflow_ISR</td>
<td>The timer ISR used to set the timerOverflow flag of the 16 bit timer used to measure the PWM pulse width.</td>
</tr>
</tbody>
</table>
Porting the Project to Other PSoC 1 Devices:

While cloning the project (as explained here) to other PSoC 1 devices, follow these instructions:

1. Make sure that the PSoC 1 device supports the Timer16 UM
2. Replace the default GPIO ISR in boot.tpl (found inside the project folder) with the GPIO_ISR function as shown in Figure 35.
3. The TimerOverflow_ISR defined in ISR.c should be placed in PulseWidthTimer (Timer16) ISR.
4. The resources should be configured as described in the section Resources Utilized, if the device is found to be not compatible while cloning.

Example 1: Interfacing One TMP05 Sensor

In this example, only one of the TMP05 sensors in the CY8CKIT – 036 is interfaced to the CY8C28 device.

**Project Settings:**

1. Set the TEMPERATURE_SENSOR macro to TMP05
2. Set the TMP_MODE macro to ‘DAISY_CHAIN’ – the sensors in CY8CKIT – 036 are fixed in daisy chain mode.
3. Set the NO_OF_SENSORS macro to ‘1’
4. Check that the PWM_IN pin is at Port 4 Pin 1 and the CONV_IN pin is at Port 4 Pin 0.
5. Generate and build the project in PSoC Designer.
6. Download the ‘hex’ file into the CY8CKIT – 020 CY8C28 processor module using MiniProg3 or MiniPorg1.

**Hardware Settings:**

1. Set jumper settings for the CY8CKIT – 036 as shown in Table 2 and Figure 37.

Table 3. CY8CKIT - 036 Jumper Settings for Example 1

<table>
<thead>
<tr>
<th>Jumper</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>J2</td>
<td>2-3 (PMW_TMP to Single)</td>
</tr>
<tr>
<td>J3</td>
<td>2-3 (VDDIO to 3.3 V)</td>
</tr>
<tr>
<td>J9</td>
<td>2-3 (12 V to 12 V_DVK)</td>
</tr>
</tbody>
</table>

2. Disconnect the fan connectors FAN-1, FAN-2, FAN-3 and FAN-4.
3. Set jumper settings for CY8CKIT – 001 as shown in Table 3.

Table 4. CY8CKIT – 001 Jumper Settings

<table>
<thead>
<tr>
<th>Jumper</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>J6</td>
<td>VDD_ANALOG to VDD</td>
</tr>
<tr>
<td>J7</td>
<td>VDD_DIG to VDD</td>
</tr>
<tr>
<td>J8</td>
<td>VDD to VREG</td>
</tr>
<tr>
<td>J12</td>
<td>LCD to ON</td>
</tr>
<tr>
<td>SW3</td>
<td>3.3 V Position</td>
</tr>
</tbody>
</table>
4. Connect the CY8CKIT – 036 to PORT A of the CY8CKIT – 001 as shown in Figure 39.
5. Power on the CY8CKIT – 001; the LCD should display as shown in Figure 40. The ambient temperature is in Celsius.

Example 2: Interfacing Two TMP05 Sensors

In this example, both the TMP05 sensors in CY8CKIT – 036 are interfaced to the CY8C28 device.

**Project Settings:**
1. Set the TEMPERATURE_SENSOR macro to TMP05
2. Set the TMP_MODE macro to ‘DAISY_CHAIN’.
3. Set the NO_OF_SENSORS macro to ‘2’
4. Check that the PWM_IN pin is at Port 4 Pin 1 and CONV_IN pin is at Port 4 Pin 0.
5. Generate and build the project.
6. Download the ‘hex’ file into the CY8CKIT – 020 CY8C28 processor module using MiniProg3 or MiniPorg1.

**Hardware Settings:**
1. Set the jumper settings for the CY8CKIT – 036 as shown in Table 4.

<table>
<thead>
<tr>
<th>Jumper</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>J2</td>
<td>1-2 (PMW_TMP to Dual)</td>
</tr>
<tr>
<td>J3</td>
<td>2-3 (VDDIO to 3.3 V)</td>
</tr>
<tr>
<td>J9</td>
<td>2-3 (12 V to 12 V_DVK)</td>
</tr>
</tbody>
</table>

2. Disconnect the fan connectors FAN-1, FAN-2, FAN-3 and FAN-4.
3. Set the jumper settings for the CY8CKIT – 001 are shown in Table 3.
4. Connect the CY8CKIT – 036 to PORT A of the CY8CKIT – 001 as shown in Figure 39.
5. Power on the CY8CKIT – 001. The LCD displays the ambient temperature in Celsius as shown in Figure 41.
Summary

Using PSoC, TMP05 sensor can quickly and easily be designed in thermal monitoring and management solutions providing support for up to 255 TMP05 temperature sensors.

PSoC’s unique ability to combine custom digital logic, analog signal chain processing and an MCU in a single device enables system designers to integrate many external fixed-function ASSPs. This powerful integration capability not only reduces BOM cost but also results in PCB board layouts that are less congested and more reliable.

About the Author

Name: Meenakshi Sundaram R
Title: Applications Engineer Senior
Background: Bachelor of Engineering in Electronics and Communication from College of Engineering - Guindy, Chennai, India.
Contact: msur@cypress.com
## Document History

Document Title: PSoC® 1 – Temperature-Sensing Solution Using a TMP05/TMP06 Digital Temperature Sensor – AN78737

Document Number: 001-78737

<table>
<thead>
<tr>
<th>Revision</th>
<th>ECN</th>
<th>Orig. of Change</th>
<th>Submission Date</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>**</td>
<td>3606242</td>
<td>MSUR</td>
<td>05/02/2012</td>
<td>New Application Note</td>
</tr>
<tr>
<td>*A</td>
<td>3671548</td>
<td>MSUR</td>
<td>07/10/2012</td>
<td>Document revised to improve clarity in images.</td>
</tr>
<tr>
<td>*B</td>
<td>4371515</td>
<td>MSUR</td>
<td>05/06/2014</td>
<td>Updated references for Figures 40 and 41.</td>
</tr>
</tbody>
</table>
Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

Automotive: cypress.com/go/automotive
Clocks & Buffers: cypress.com/go/clocks
Interface: cypress.com/go/interface
Lighting & Power Control: cypress.com/go/powerpsoc
Memory: cypress.com/go/memory
Optical Navigation Sensors: cypress.com/go/ons
PSoC: cypress.com/go/psoc
Touch Sensing: cypress.com/go/touch
USB Controllers: cypress.com/go/usb
Wireless/RF: cypress.com/go/wireless

PSoC® Solutions

PSoc.cypress.com/solutions
PSoC 1 | PSOC 3 | PSOC 5

Cypress Developer Community

Community | Forums | Blogs | Video | Training

technical Support

cypress.com/go/support

PSoC® 1 is a registered trademark of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are the property of their respective owners.