

This Application Note is pertinent to the  
Unidrive SP, Commander GP20, SK and Affinity Families

## Obtaining Temperature Readouts in Fahrenheit

Our drives contain internal thermistors<sup>1</sup> which provide the drives microcomputer with temperature information. The drive uses this information to protect itself from damage to due excessive heat in certain key areas throughout the electronics. These temperatures are monitored and the drives microcomputer in some cases will begin to warn of an overtemperature condition by flashing **Hot** as an alarm. In other cases, the drive may reduce the PWM switching frequency to reduce transistor power dissipation. In other cases, the fan speed may be increased to a full speed. Should these temperatures persist and continue to rise to an unacceptable level, the microcomputer will cause the drive to trip and display an Overtemperature condition.

These temperatures could be monitored and displayed by HMI units such as our CTIU or CTVue keypad/displays. One could decide to include these in a Filtered List of common diagnostic registers that one may wish to review periodically. Click here→ [CTAN289](#) for more details on this topic.

### Temperature Registers ( internal Test Points )

#### Commander SK

Parameter #7.04 displays the drives Heatsink Temperature in °C  
Parameter #7.05 displays the drives Power Circuit 2 Temperature in °C  
Parameter #7.34 displays the drives IGBT Junction Temperature<sup>2</sup> in °C  
Parameter #7.36 displays the drives Input Rectifier Temperature in °C SK6 only

#### Unidrive SP, CommanderGP20, Affinity

Parameter #7.04 displays the drives Heatsink Temperature in °C  
Parameter #7.05 displays the drives Power Circuit 2 Temperature in °C  
Parameter #7.06 displays the drives Control Board Temperature in °C  
Parameter #7.34 displays the drives IGBT Junction Temperature<sup>2</sup> in °C  
Parameter #7.36 displays the drives Input Rectifier Temperature in °C SP6, SPMA

These can be useful in general diagnosis. Obviously if one notes that the Heatsink temperature is running higher than usual, this could mean a fan has failed or is failing ( bearing wore out ) or perhaps the heatsink is getting filled with material – needs blown out.

<sup>1</sup> A **thermistor** is temperature transducer. It is a resistor whose resistance changes as a function of its temperature. Most are designed to provide a degree of linearity of a workable temperature range.

<sup>2</sup> The **IGBT** ( Insulated Gate Bipolar Transistor ) **junction temperature** is derived mathematically by the drives microcomputer. It is not an actual temperature measurement but rather an estimate.

These temperatures are useful but for many the meaning of Celcius is not as readily apparent as it's Fahrenheit counterpart. This application note illustrates how to convert these temperature register from Celcius to Fahrenheit for the convenience of your customer. In the case of Unidrive SP, Commander GP20 and Affinity drives, this can be accomplished at no added cost.

For Commander SK the low cost logic stick would be required.



## **Implementation**

In order to convert Celcius ( used to be called Centigrade ) to Fahrenheit , one must look up the conversion formula if they don't already know it.

$$\text{Fahrenheit or } F = \frac{9}{5} \times \text{Celcius} + 32 \quad \text{or} \quad F = 9/5 C + 32 \quad \text{or} \quad F = 1.8 * C + 32$$

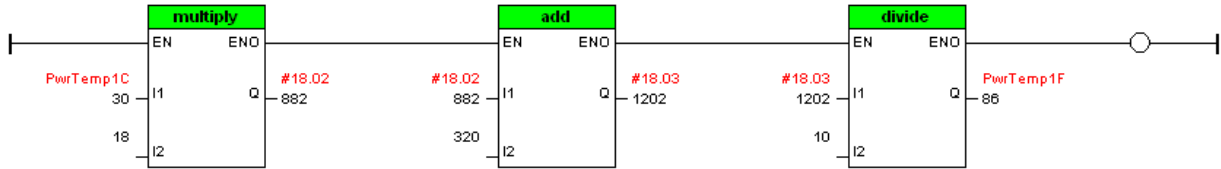
Note that the Celcius temperature must be multiplied by 9/5 (or 1.8 ) before adding 32

Once we know the formula we can let SyPT Lite perform the repetitive and mundane calculation for us.

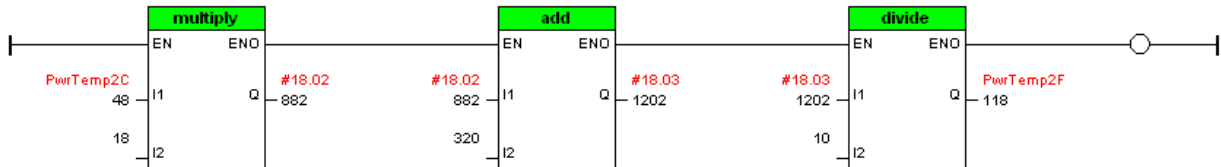
## A Conversion Program

The program below was written for Unidrive SP, Commander GP20 or Affinity.

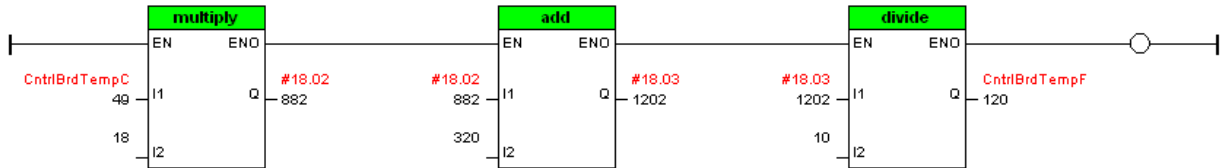
(<sup>o</sup> Celcius to Fahrenheit Conversion \*)



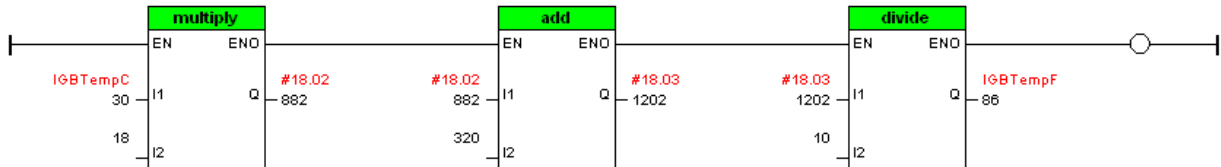
(<sup>o</sup> Celcius to Fahrenheit Conversion \*)



(<sup>o</sup> Celcius to Fahrenheit Conversion \*)



(<sup>o</sup> Celcius to Fahrenheit Conversion \*)



The following aliases were assigned to the appropriate internal drive registers:

### Celcius registers

PwrTemp1C #07.04  
 PwrTemp2C #07.05  
 CntrlBrdTempC #07.06  
 IGBTTempC #07.34

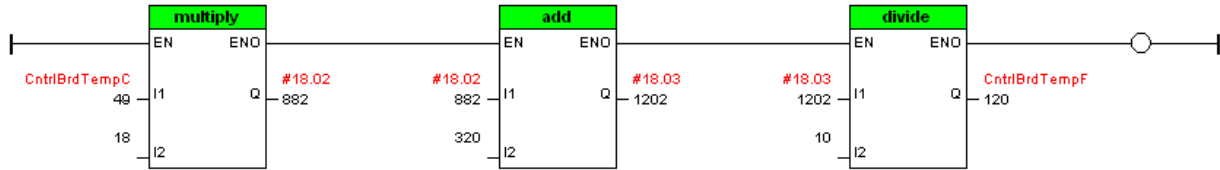
### Converted Fahrenheit registers

PwrTemp1F #19.02  
 PwrTemp2F #19.03  
 CntrlBrdTempF #19.04  
 IGBTTempF #19.05

## How does this work ?

Since each conversion is repetitive let us just examine one case

(° Celcius to Fahrenheit Conversion °)



SyPT Lite uses integer mathematics. This essentially means only whole numbers can be used ( no decimals ) . We would like to multiply by 1.8 , rounding to a 2 would be inaccurate so we can multiply by 18 and just keep the factor of 10 in mind. Then we need to add 32. But since the multiplication was 10x what it should have been we must add 10x as much here also. So we add 320. Then we can get rid of the scale factors by dividing the resultant by 10 and place the converted result into a free register within the drive that can be observed as Fahrenheit for this particular temperature.

It should be noted that #18.02 and #18.03 are free registers within the drive that are used over and over for each conversion calculation rung. These are only intermediate results and are being used as “scratchpad” registers.

Once the program executes the entire 4 rungs in this example, the program starts again from the top and continues – continuously performing the conversions.

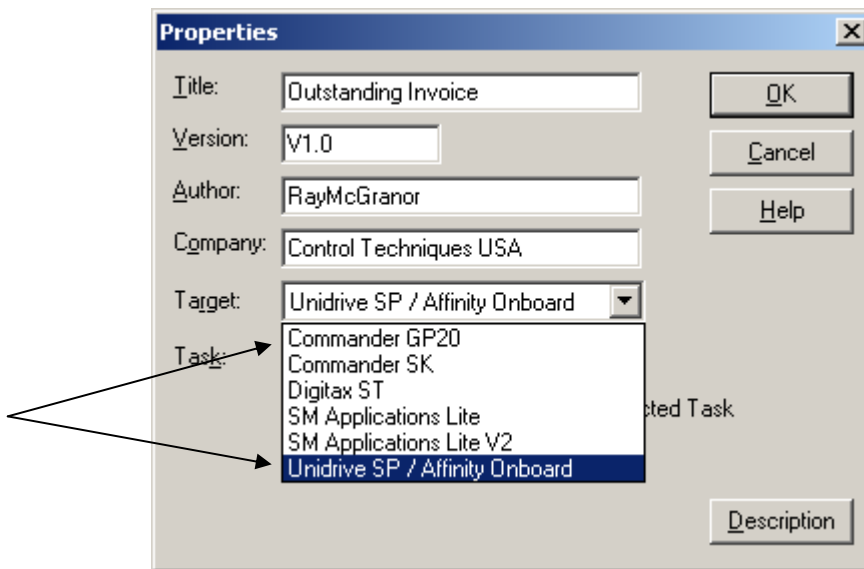
To obtain a copy of this example program click here → [CTSL010.dpl](#)

## Using this program

One would merely open this program using SyPT Lite and select the proper Drive Target type after clicking on the rightmost Toolbar icon.



This would be: **Unidrive SP/Affinity On Board** or **Commander GP20** or **Commander SK**



Then you would merely click on the Lightning Bolt to compile and download into the drive.

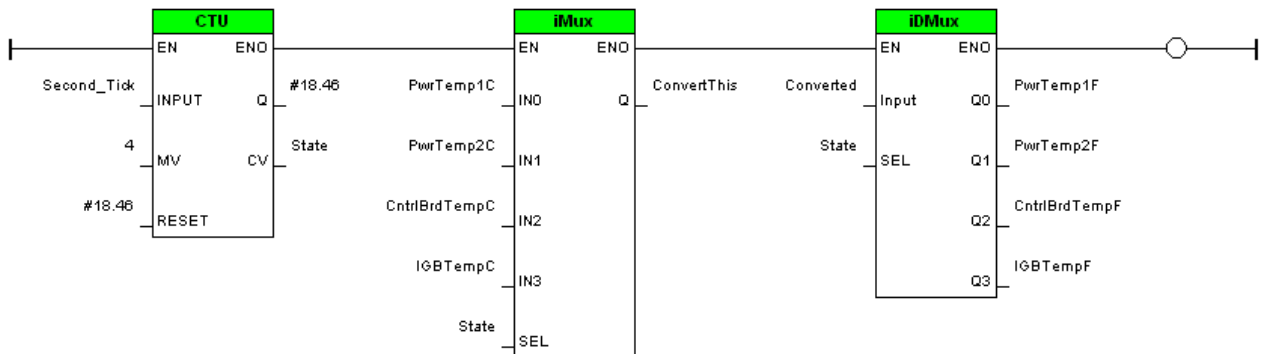


## An Alternate Program

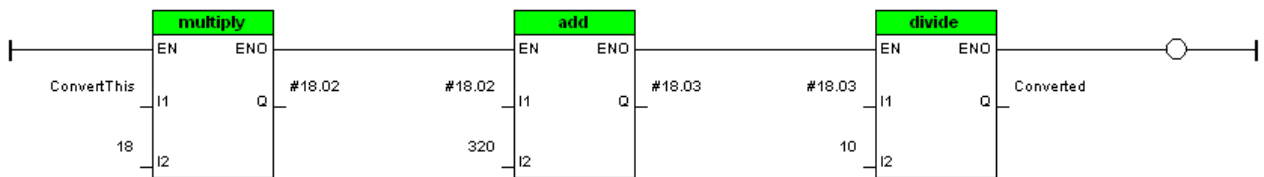
The previous program was effective but not very elegant. It was rather easy to think about and create. Once I made one rung that worked, I just had to copy and paste 3 or more others and change the registers to be converted etc.

When things become repetitive often times it is indicative that a more structured way of programming might be more efficient or elegant. The solution below is more elegant albeit a bit more complex to study but perhaps worth the effort if for nothing more than an academic exercise.

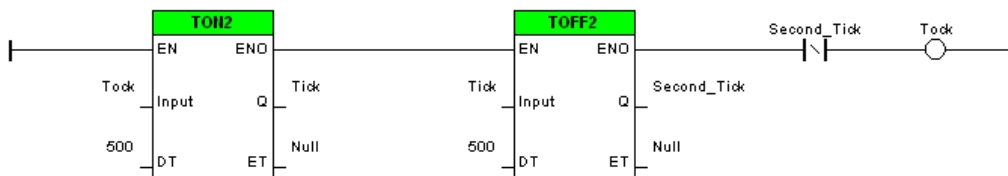
(<sup>o</sup> State Generator and Multiplexer/Demultiplexer<sup>o</sup>)



(<sup>o</sup> Celcius to Fahrenheit Conversion<sup>o</sup>)



(<sup>o</sup> Seconds Clock Generator<sup>o</sup>)



The above program utilizes State Machine methodology. A State Machine is merely a method of organizing a program to execute in a certain manner depending on what state the machine is- at various times. The state in this program is advanced as a counter ( up counter –1<sup>st</sup> rung) gets incremented by a 1 second pulse or tick ( see the last rung and [CTSL001](#) for more details). The state of the counter is used to determine which input to a Multiplexer is to become its output. In this example the inputs are our various Celcius temperatures to be converted. The selected output I've called **ConvertThis**. The counter resets when it gets to the 4<sup>th</sup> state.

**ConvertThis** is presented to the 2<sup>nd</sup> rung which you should recognize as the Celcius to Fahrenheit conversion formula we used in each rung in the first example. The result of this conversion I called **Converted** which gets presented to a Demultiplexer ( back on the 1<sup>st</sup> rung ) which depending on the current state gets stuffed into its' appropriate corresponding Fahrenheit register.

To obtain a copy of this example program click here → [CTSL010a.dpl](#)

## **Summary**

Although the second example is a worthy academic exercise and illustrates a more advanced technique of using a state machine, sometimes a simple straight forward approach is better.

In this example, the solution is certainly more complex looking ( not as easy to understand as the first example ) but in addition, it is somewhat less flexible as well. I've used all 4 inputs. If I wanted another temperature conversion I would have to think of another solution. Whereas in the first example I would merely Copy & Paste another rung like all the others.

In addition, this method although slick, takes more execution resources both in code space and execution time due to the more complex function blocks.

Granted, the last rung, which is nothing more than a 1 second pulse source, could be used for other purposes in my program, in which case this scheme may hold more merit. In addition, if we had been performing the same calculation on a greater number of inputs, we would want to create a “subroutine” or similar method to reduce the redundant code to execute.

Regardless, these two examples illustrate the power of SyPT Lite and the embedded PLC within our drives to solve miscellaneous application problems – often with no additional cost.

To obtain the Free software SyPT Lite program that can be utilized by our Unidrive SP, Commander SK, Commander GP20 and Affinity drives click here→ [SyPT Lite](#)



**Questions ?? Ask the Author:**

**Author:** Ray McGranor  
(716)-774-0093  
(800)-367-8067

**e-mail :** <mailto:ray.mcgranor@emerson.com>