Keywords: TDC, ultrasonic, metering, heat meter application, energy conservation

APPLICATION NOTE 6107

USING THE MAX35101 IN A HEAT METER APPLICATION

Abstract: In this application note, we will show how the MAX35101 time measurement chip makes an ideal metering device for water-sourced radiator designs.

As energy prices have escalated over the past few decades, energy conservation has become an ever greater concern. And a central component in energy conservation is accuracy and flexibility in accounting and billing for energy use. The reason is obvious: if billing and accounting can be more flexibly applied, economic incentives can be provided to reduce energy usage during periods of highest demand and to shift energy usage to lower-demand periods.

For example, electricity metering is moving from mechanical meters based on a rotating Ferraris wheel to electronic meters that can keep track of energy use based on time of use, power factor, and surge pricing. Utility companies are going to the expense of replacing mechanical meters because, ultimately, there has to be sufficient generating capacity for the peak hours—generating capacity that will be largely idle during off-peak periods. But if they can increase prices during the peak hours and offer discounts for off-peak usage, users with demand flexibility can enjoy significant discounts—and the utility will see the load profile over the course of a day become much smoother, possibly reducing the need for new generating capacity.

Heating and cooling a living space is often the greatest contributor to energy consumption in both commercial and residential applications. It is for this reason that many conservation efforts focus on adding insulation and sealing air leaks in older buildings, and sponsoring public service campaigns to remind utility customers to set their thermostats warmer in the summer and cooler in the winter. Customers who fail to take these measures will see their energy bill inevitably rise.

In multi-unit dwellings, it is not uncommon for heat to be provided by radiators sourced by heated water. There are several reasons for this: heating a large quantity of water and distributing it to radiators in the living spaces is an efficient way to provide heat. Water is an inexpensive working fluid, has a relatively high capacity to carry heat, and the technology for building water radiators, pumps, and distribution systems is mature.

But providing individual billing is more difficult with water-sourced radiator heat than with direct electrical heat or direct gas- or oil-fired heat. With these latter heat sources, it is easy to measure energy usage: just measure the number of kilowatt-hours of electricity or the volume of gas or oil consumed. The radiator is different, because there are two components to the energy usage, the amount of water flowing through the radiator and the temperature drop as the water flows through the radiator.

In this application note, we will show how Maxim Integrated's MAX35101 time measurement chip makes an ideal metering device for water-sourced radiator designs.

Background

In a typical water-sourced radiator system, hot water from a heat source enters the radiator through a valve that can be operated by the user, and exits the radiator to return back to the heat source to be re-heated before making the trip again. The user can open the valve to allow more water to flow and so more heat is rejected into the space, or the user can close the valve to reduce the flow, and consequently, reduces the heat delivered to the space.

To reject a given amount of heat energy into a space, it is necessary to start with a working fluid at a temperature higher than the ambient temperature of the space. As heat from the working fluid is rejected into the space, the working fluid cools toward the ambient temperature of the space and the temperature of the space rises. The enthalpy—that is, the heat—contained in the working fluid can be calculated by the following formula:

$$E = C(t) \times m \times t$$

In the equation, t is the absolute temperature of the working fluid, m is the mass of the working fluid and C is the specific heat capacity of the working fluid. In our analysis, the working fluid is water. Water has a heat capacity of about 4.1813 joules per gram per degree Kelvin at room temperature and varies somewhat over its liquid temperature range. Nominally, water has a density of about one gram per cubic centimeter, but this varies strongly with temperature. Water is densest at 4°C (0.99997/cc) and at its least dense just below the boiling point (0.9584/cc). Since we will be measuring flow volume and not mass, we must apply conversion tables to use the proper heat capacity and volume-to-mass conversion values. To compute the energy rejected into the space, we compute the enthalpy as the working fluid enters the radiator and subtract the enthalpy as it leaves. The difference is the heat rejected into the space.

Armed with this knowledge, we see that to measure the heat energy delivered to the space (see **Figure 1** for a diagram or the arrangement) we need to periodically measure is the inlet temperature, the outlet temperature, and the volume of the water moving through the system. If we are measuring flow through a spool body of known diameter, then the volume of water will be proportional to the velocity of water through the spool body. We can then convert the volume of water to mass of water using lookup tables, multiply the mass by the heat capacity of water at the operating temperature (also obtained from a table), and then multiply the result by the temperature drop (the difference between the inlet temperature and the outlet temperature) across the radiator. The result is the amount of energy dissipated by the radiator.

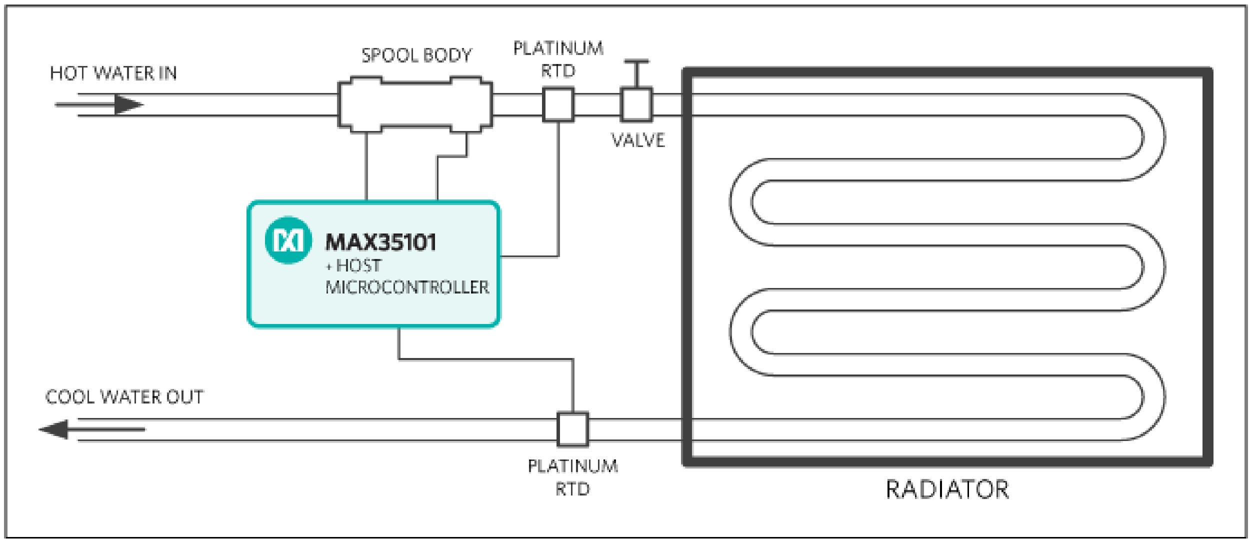

Figure 1. A water-sourced radiator with a heat meter.

Measuring flow

The MAX35101 time measurement device contains drivers for piezoelectric transducers and can launch an acoustic pulse through the working fluid in the upstream direction and then in the downstream direction. By computing the time difference between the time-of-flight in the upstream direction and the time-of-flight in the downstream direction, the velocity of the working fluid in the spool body can be computed. Multiply by the cross-sectional area of the spool body, and the volume of flow per unit of time can be computed.

There are application notes in the library that describe this process in detail. For the purposes of this discussion, we will assume that we have computed the flow rate in cubic centimeters per second.

Temperature measurement

When considering how to measure temperature in an industrial setting, two technologies stand out: thermocouples and resistive temperature detectors (RTDs). There is a place for each in temperature detection, but it usually comes down to this: if the application needs to measure very hot (greater than 600 C) temperatures, a thermocouple is a better choice. In virtually every other situation, however, the RTD is a better choice, and it is our choice here.

An RTD is typically a fine coil of platinum wire or a thin film of platinum metal on a ceramic or other inert base. As the temperature increases, the resistance of the conductor increases; if the temperature range is relatively narrow, the change in resistance is a simple quadratic function with temperature.

Above 0°C, the resistance of a platinum RTD is given by the following formula:

$$R_T = R_0 (1 + AT - BT^2)$$

For typical RTD sensors, A has a value of about 3.9083×10^{3}/°C and B has a value of about -0.5775×10^{6}/°C^2. At 90°C (typical inlet temperature for a water-sourced radiative heat system) a 1,000Ω RTD will exhibit a resistance of 1,347.07Ω. If the outlet temperature is room temperature (it would not be—no radiator is

100% efficient) the 1kΩ RTD will exhibit a resistance of 1,097.35Ω, for a difference of 249.72Ω. This resistance range is easily measured.

One method of determining temperature given the resistance of the RTD is to solve the quadratic equation given above for T and plug in R_T. The problem with this "simple" solution is accurately measuring the resistance of the RTD. Simply put, the MAX35101 is not designed to measure resistance—it is a time measurement circuit.

Fortunately, there is a simple way to convert resistance to time: allow a capacitor to discharge through the resistance and count the amount of time it takes to discharge to a known voltage level. In the MAX35101, there are four ports connected to the appropriate current drivers and switches to first charge an external capacitor, and then discharge the capacitor through the RTD while counting the time required to discharge the capacitor to a particular voltage. Here is how it works:

In **Figure 2**, Q1 turns on to precharge the capacitor prior to the measurement interval. When the capacitor is completely charged, Q1 turns off and Q2 turns on and begins discharging the capacitor at the same time the time measurement logic in the MAX35101 begins counting time. When the capacitor falls below the comparator's threshold level, the comparator switches, notifying internal logic to stop counting and report the count. At the same time, Q2 is turned off and Q1 is turned on to precharge the capacitor again. Since resistance is proportional to temperature, and a higher resistance corresponds to a longer period to discharge the capacitor, a greater count will correspond to a higher temperature. **Figure 3** illustrates the voltage across the capacitor when measuring the capacitor discharge time.

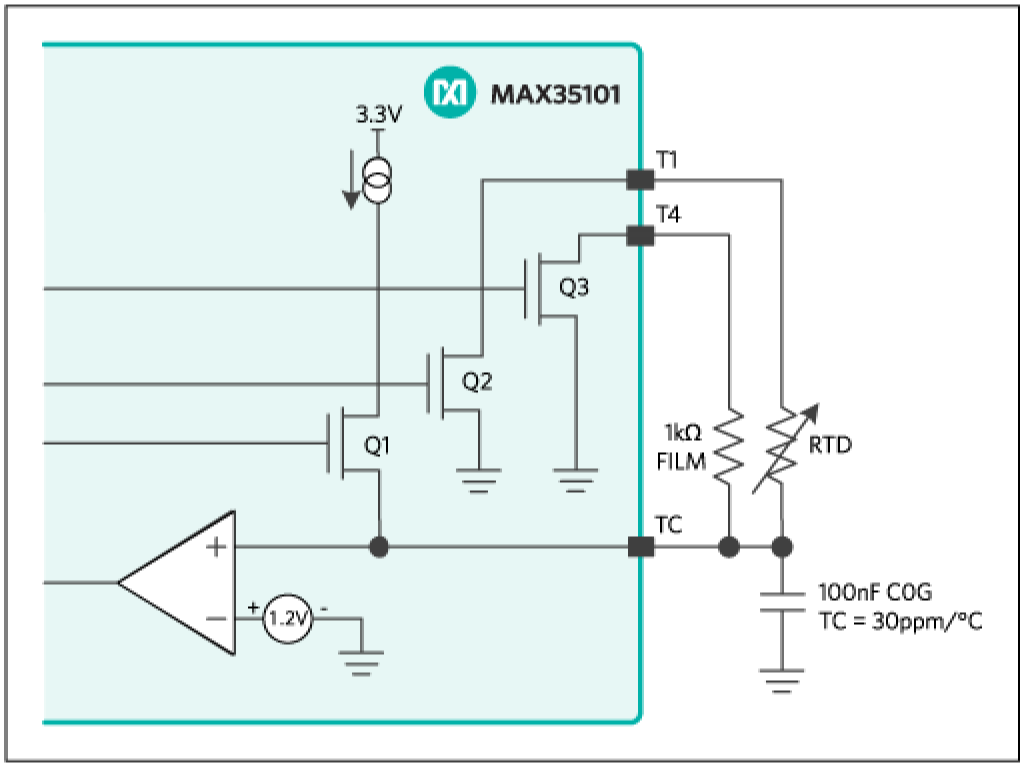

Figure 2. Temperature-to-time conversion circuit.

In Figure 2, a second input—T4—is connected to a 1kΩ metal film resistor. Metal film is a very stable material to use as a resistance element, changing little with temperature and with good aging properties. The purpose of measuring the time required to discharge the capacitor through a very stable, fixed resistor is to eliminate all other possible influences in the circuit. For example, the C0G-type capacitor, while having good stability, does introduce another factor that varies with temperature. Also, the internal voltage reference and comparator—while very good—introduce minor but unwanted variations with temperature and supply voltage. By measuring first the temperature at T1 and then the fixed resistor at T4 and computing the ratio of the two counts, all spurious effects are cancelled, leaving only the effects of temperature on the RTD.

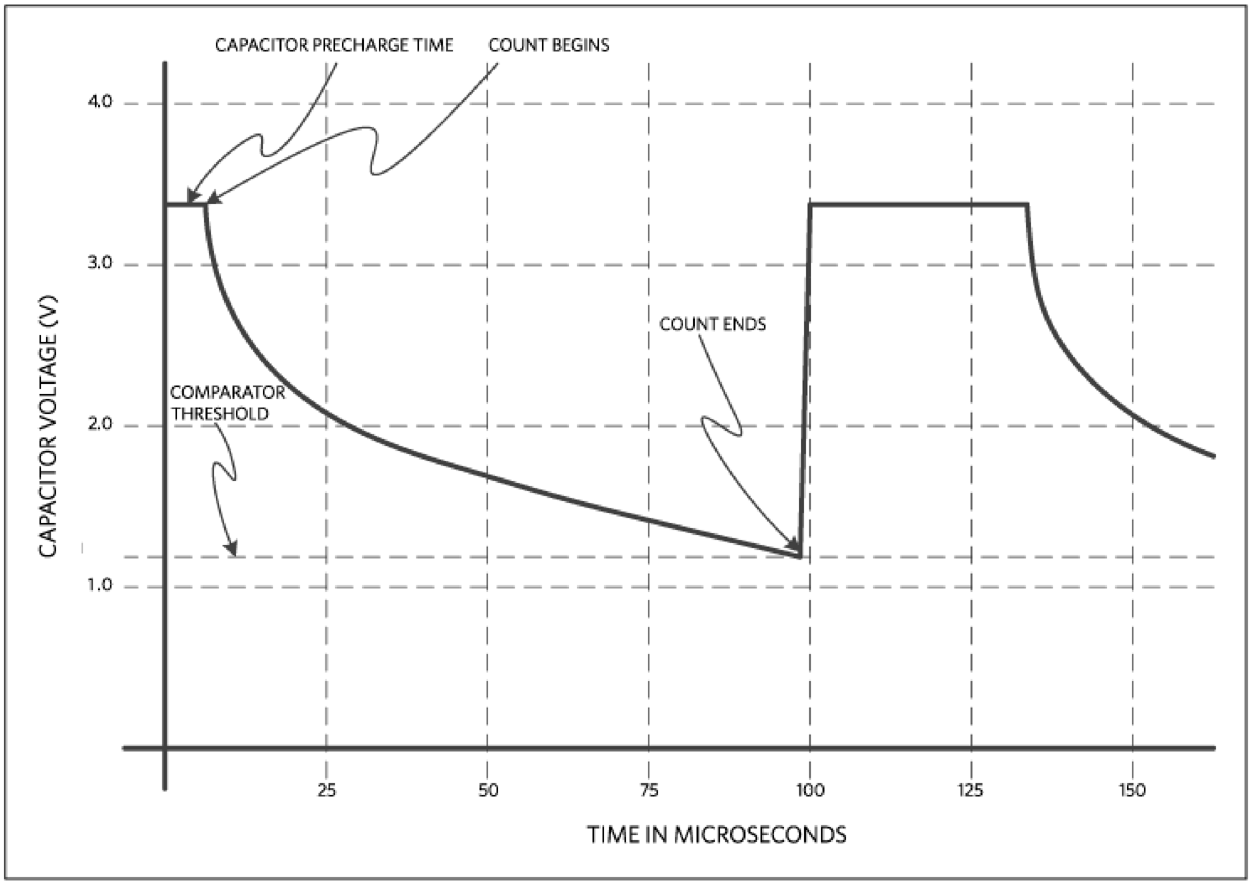

Figure 3. Timing in the temperature-to-time measurement circuit.

The host microcontroller may take the average count from several measurements of the RTD resistance and the reference resistor from the MAX35101, or may use the event management system of the MAX35101 to automatically average several counts. Either way, once the average of the quotients of several RTD resistance counts and reference counts are taken, the temperature can be found by using a lookup table. RTD manufacturers typically provide a table of resistance versus temperature for each of their sensor types.

MAX35101 Register Interface

The MAX35101 contains a SPI interface to connect to a host microcontroller. To start a temperature conversion cycle, the host microcontroller need only write a 0x03 command to the MAX35101, and then read the results over the SPI interface from the results registers. But before you begin that process, you need to set up a few operating parameters. These parameters are configured in the Event Timing 2 register.

EVTIM2 (Write: 0x40 Read: 0xC0)

Bit	15	14	13	12	11	10	9	8
Name	TMM					CAL_USE	CAL_CFG[2:1]	
Access	R/W					R/W	R/W	
Reset	0 0000					0	00	
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	CAL_CFG.0	TP		PRECYC			PORTCYC	
Access	R/W	R/W		R/W			R/W	
Reset	0	00		000			00	

EVTIM2[1:0] - PORTCYC[1:0]

Temperature Port Cycle Time. This field sets up the time between successive temperature measurements in a group.

PORTCYC[1:0]	Time (µs)
0b00	128
0b01	256
0b10	384
0b11	512

The value you enter into this field will depend on the value of the timing capacitor. If you are using the recommended 100nF C0G capacitor, you can leave this field at the default of 0b00.

EVTIM2[4:2] - PRECYC[2:0]

Preamble Temperature Cycle Count. When you initiate a emperature measurement, the MAX35101 can optionally insert dummy cycles at the beginning of the operation to reduce the effect of dielectric absorption in the capacitor. Generally, one or two dummy cycles are sufficient to deal with the issue.

EVTIM2[6:5] - TP[1:0]	**Temperature Port.** This field defines which ports will be measured during one temperature measurement cycle

TP[1:0]	Ports to Measure
0b00	T1, T3
0b01	T2, T4
0b10	T1, T3, T2
0b11	T1, T3, T2, T4

EVTIM2[9:7] - CAL_CFG[2:0]	**Calibration Configuration.** The MAX35101 can be configured to automatically execute a CALIBRATE command when executing event operations. It is not germane to this discussion.
EVTIM2.10 - CAL_USE	**Use Calibration.** When set, the MAX35101 scales all timing results with the data in the CalibrationINT and CalibrationFRAC registers. These registers are outside the scope of this discussion
EVTIM2[15:11] - TMM[4:0]	**Temperature Measurement Cycle Count.** This field establishes the number of temperature measurements taken when event timing is in use. This field is outside the scope of this discussion

Start the temperature measurement cycle by writing a 0x03 command to the SPI port. The MAX35101 will perform the requested temperature measurement cycles, including any dummy cycles requested in the PRECYC field. When finished, the MAX35101 will deposit the accumulated count into the following register pairs:

Register Address	Register Name	Description
0xE7:0xE8	T1Int:T1Frac	Temperature count for input T1
0xE9:0xEA	T2Int:T2Frac	Temperature count for input T2
0xEB:0xEC	T3Int:T3Frac	Temperature count for input T3
0xED:0xEE	T4Int:T4Frac	Temperature count for input T4

The pairs of 16-bit registers can be taken as a single 32-bit register. Considered this way, the count associated with the temperature inputs has a resolution of about 3.81ps and a maximum time measurement period of about 8.19ms. A one microsecond interval would provide a count of 1µs/3.81ps = 262,144 counts. Conversely, a count of 10,000,000 would correspond to a time of 10,000,000 × 3.81ps = 38.1µs.

Now that we know how to convert from raw counts to time, we are ready to convert the time measurement to emperature. To begin, let us assume default values of 1,000Ω for the resistor and 100nF for the capacitor. When the capacitor is fully charged and the switch connects the resistor to ground, the time for the capacitor to discharge from 3.3V to 1.2V is given by:

$t = -RC \ln (VO/VI)$

Evaluating this for the values above, the discharge curve will hit the threshold after 101.16μs. The count registers provide time in units of about 3.81ps, so dividing one by the other we see that the contents of the count register should be about 26.5 million (specifically, 0x0194 A3EF).

That is the value you should obtain when measuring the reference resistor, or when measuring the platinum RTD at its reference temperature of 0°C.

One might think at this point it would make sense to solve the equation above for R and convert the time measurement to a resistance value. But since the time and resistance are proportional, and since we are really not interested in the absolute resistance but rather the ratio of the resistance of the RTD and the fixed reference resistor, we can leave the numbers as raw counts.

Since you probably do not have a floating-point unit in the host microcontroller, an easy way to take the ratio is to do this:

1. Measure the reference resistor
2. Shift the measured value right by (for example) twelve bits
3. Measure the RTD
4. Divide the measured value of the RTD by the shifted value of the reference resistor. Discard the remainder.

If the value of the reference resistor is equal to the value of the RTD, then the result will be exactly 4,096 (implying a temperature at the RTD of 0°C). If the value is greater than 4,096, then the RTD had a value higher than the reference resistor (and the temperature is greater than 0°C); if the value is less than 4,096, then the RTD had a value less than the reference resistor (and the temperature is less than 0°C).

But how much greater (or less)? RTD manufacturers publish tables for each of their sensors that give the resistance of the sensor over their entire recommended operating range. For example, here is an excerpt of a table from a typical sensor, centered on room temperature:

Temperature (°C)	Resistance (Ω)
20	1,077.94
21	1,081.82
23	1,085.70

24	1,093.47
25	1,097.35
26	1,101.23
27	1,105.10
28	1,108.98
29	1,112.86
30	1,116.73

Now, we can extend this table to show the ratio and the expected result from the divide operation:

Temperature (°C)	Resistance (Ω)	Ratio (vs. 1,000Ω)	Result of divide
20	1,077.94	1.07794	4,415
21	1,081.82	1.08182	4,431
22	1,085.70	1.08570	4,447
23	1,089.59	1.08959	4,463
24	1,093.47	1.09347	4,479
25	1,097.35	1.09735	4,495
26	1,101.23	1.10123	4,511
27	1,105.10	1.10510	4,527
28	1,108.98	1.10898	4,543
29	1,112.86	1.11286	4,558
30	1,116.73	1.11673	4,574

In your microcontroller code, you will need only the shaded columns: the temperature and the results of the division—and if you have a table with a fixed temperature increment (like the table above) you can eliminate the temperature column. For example, if you had a static array called temp_table that only contained the rightmost column of the table and that started at 0°C, your conversion subroutine might look like this:

```
int convert_quotient_to_temperature(int quotient)
    {
    int i=0;
    while(temp_table[i] < quotient) i++;
    return i;
    }
```

So if the result of the divide operation was, say, 4,493, one could search the table and determine the nearest value is 4,495 and conclude that the temperature was 25°C. If more precision is required, one could use linear interpolation between the points on the table—if using the nominal value of 4,096, and could extract about one more digit of precision. And if even more accuracy is desired, one could model the RTD itself and use that knowledge to extract even more meaningful resolution from the raw count.

Putting it All Together

Now we have the tools necessary to build a heat meter. We can measure the flow through the radiator, and then measure the inlet and outlet temperature. All that is needed now is to integrate these values over time.

Practical integration over time periodically involves periodically taking samples and making the assumption that the flow rate and temperatures are constant until you take another measurement. Fortunately, with reasonable sample rates that turns out to be a good assumption: bulk temperatures of the working fluid really do not change very frequently, and the flow rate changes only when the valve position is changed.

Assume that the system takes a measurement every minute. At one particular measurement, let us say the MAX35101 senses the inlet temperature at 90°C and the outlet temperature at 50°C, and computes the flow rate at 15cc per second.

At 90°C, water has a density of about 0.965g/cc. So in one second, we observe about 14.48 grams of water pass through the spool body. Also, at 90°C, the specific heat of water is 4.208 joules per gram per degree, and a 50°C, the specific heat of water is 4.182 joules per degree per gram per degree.

We can compute the energy delivered in one second. Note that we are using Celsius and Kelvin temperatures somewhat interchangeably below. In reality, they are not interchangeable, but since we are concerned about differences rather than absolute enthalpy of the medium we can be a little loose with the units.

$$E = (C(T1) \times m \times T1) - (C(T2) \times m \times T2)$$

$$E = 4.208\ \text{J/g/K} \times 14.48\text{g} \times 90°\text{C} - 4.182\ \text{J/g/Kx}\ 14.48\text{g} \times 50°\text{C}$$

$$E = 5{,}481.97\text{J} - 3{,}026.72\text{J}$$

$$E = 2{,}455.25\text{J}$$

So the energy rejected into the space during one second is 2,455.25 joules. Over the one minute sample period, the energy rejected into the space is 2,508.89j/s × 60s = 147.3kJ = 40.92Wh. If that rate of usage persists for an hour, the energy used would be about 2.455kWh.

Enhancements

The MAX35101 time measurement device includes four inputs for temperature sensors. In this application, we have used three so far—one for outlet temperature, one for inlet temperature, and one for the reference resistor. But it would be a simple matter to attach a sensor to the one remaining sensor input to determine the ambient temperature and to provide an actuator to adjust the valve. Then an interface element associated with the control processor could allow the user to directly input the desired temperature with the control processor regulating the valve to achieve the requested temperature. Such a system would represent a complete closed-loop temperature management system: the user selects a temperature, and the control processor drives the valve to deliver more or less heat to the space—all while keeping track of how much heat is actually consumed.

Another enhancement would be remote management and reporting. In this case, the control processor would serve as a data aggregator and switch to permit remote recalibration and reporting of exceptional conditions. Reporting could be performed via wireless protocols (WiFi or cellular modem) or wired protocols (typically powerline networking).

Conclusion

The MAX35101 is an ideal device to measure flow rate in any fluidic system. With the addition of temperature sensors, the MAX35101 can be used to measure energy use in water sourced radiator systems. With the addition of more sensors and a host microcontroller, it can serve as the analog front-end to a complete energy management system.

A similar version of this article appeared June 1, 2015 in *EDN*.

Related Parts		
MAX35101	Time-to-Digital Converter with Analog Front-End	Free Samples
MAX35101EVKIT	Evaluation Kit for the MAX35101	
MAX35102	Time-to-Digital Converter Without RTC	Free Samples
MAX35103	Reduced Power Time-to-Digital Converter with AFE, RTC, and Flash	Free Samples
MAX35103EVKIT	Evaluation Kit for the MAX35103	