Engine Parameters

Mark DiVecchio
markd@silogic.com
http://www.silogic.com/trains/ADPCM.html

7. Feb. 2021

Web pages:

http://www.silogic.com/trains/RTC_Running.html
http://www.silogic.com/trains/OOK_Radio_Support.html
http://www.silogic.com/trains/ADPCM.html
http://www.silogic.com/trains/RFID.html
http://www.silogic.com/trains/RTC_Control_Language.html

Over the past several years, I've been working on software that lets me control my trains from my computer. It is based on work done by Mike Hewett and others. Mike first figured out how to snoop on the wired communications between the Remote and the TIU. I took that a step further and snooped out the radio communications protocol. I used that knowledge to develop my Remote Train Control program (RTC) that runs on Windows based computers. A few years later, I extended that work with a new program ADPCM that plays the sounds in a sound file. The ADPCM program developed into a program that allows you to add and delete sounds to the sound file.

Many users of ADPCM have asked me for a way to change engine parameters. We don't, though, really know what they are. We are fairly sure that values in the sound file include parameters that describe the physical mechanism in the engine. Parameters such as gear-ratio and wheel diameter must be in there somewhere.

I have no information about these parameters. I can look at the sound file and I can make guesses. Hopefully educated guesses that might let us learn about the engines.

First, everything you see here was developed from publicly available documentation. I've gotten hints from the users of my programs, documents I found on public internet sites and much of this comes from editing sound files and downloading them into an engine to see what happens. I've received help from many others who took the time to download sound files to check the effect of changes. I thank everyone for their help.

Engines used for testing

I have 18 engines in my yard and I used all of them at one time or another for this experiment. To keep this document readable, I'll only show data from these two engines. In all cases, though, I've confirmed my conclusions by examining other engines.

My Engine #	Stock #	Short Description
15	20-20484-1	#2060 P&LE Diesel
11	30-1660-1	#9060 P&LE 0-6-0 Steam

Baby Steps

I looked at many, many sound files. I already knew a lot about the sound file format from my work on ADPCM. I knew where the sound information was. But there was an area at the start of the sound file that was unknown to me. It stretched from 0x00 to 0xFF – 256 bytes. Having been in the programming business for decades, I could make an educated guess that this area would logically contain important engine information. As you will see, it does.

RAM Mapping

I started a ~~months~~ years long effort to determine what was in that beginning section of the sound file.

For my RTC program, I already had figured out how to read up the RAM memory from the engine. With that method, I could change engine settings, then read up the RAM and look at how it's values changed. For example, I changed each of the 5 volume setting, read up the RAM and I could see 5 – 16 bit words that changed with the volume. I could set the engine speed, read up the RAM and I could see the speed value. Going further, I found a 16 bit word whose bits told you which lights were on/off and which features (like smoke volume) were set. I could see where the trip odometer (DTO), odometer (DOD) and chronometer (DCH) values were stored.

In Feb of 2019. I published this RAM Mapping document on my web page.

I am guessing that the first 256 bytes of the sound file are loaded into RAM memory whenever the engine is started. Somehow, this information is saved from one operating session to the next.

There were many 16 bit words in that area whose meaning I did not know. Many were just 0x00, or 0xFF or were the same in all engine sound files that I looked at. There were a few that seemed to change between sound files. Some were the same in all diesel engines and some were the same for all steam engines. Maybe these were the elusive engine parameters.

Here are the RAM addresses that caught my eye:

RAM Address in Hex (and the same address in the sound file)
0x04-0x05
0x06-0x07
0x08-0x09
0x0A-0x0B
0x0C-0x0D
0x18-0x19
0x22-0x23
0x64-0x65

As I've seen in this RAM, each address points to one byte. Two bytes make up a 16 bit word and those words are stored in big endian format.

Tach Counts

I am fairly certain that all physical calculations in the engine revolve around "tach counts". One motor flywheel in every engine has a white paper strip glued around it with black stripes. In every case that I looked at, there are 24 stripes but I don't know if that is universal.

Mounted next to the flywheel is an infrared based photo detector pointed at the tach strip. This detector generates an electronic pulse when a stripe passes underneath it. The software in the engine can count these and when 24 stripes pass by, it knows that the motor made one revolution.

The simple things first – RAM 0x06-0x07

The engine gear ratio is an important engine parameter. It seems logical that the software in the engine needs this information so it is a prime candidate to be in this RAM area.

Fortunately, it is easy for us to determine the gear ratio. Use our finger to twirl the tach wheel until the driving wheel goes around once and count the tach wheel revolutions. Simple! I did this on the engines that I own and I got help from others who did the same.

Diesel Engines	Everyone I checked :10.5:1
Steam Engines	P&LE 0-6-0 : 18:1 P&LE 0-8-0 : 14:1 P&LE 2-8-0 : 18:1 P&LE Berkshire : 16:1 Jake’s 2-8-0 : 28:1

Looking for patterns

I looked at the values in the RAM addresses that I listed above. I used a spreadsheet to tabulate those values and to look for patterns. Almost everything that I describe in this document is the result of looking for patterns.

The results you see here did not magically appear in seconds but only after weeks and months of examination.

Gear Ratio

I noticed (again, after weeks) that the values in RAM 0x06-0x07 seemed to follow the value of the gear ratio. Again after much reflection, I saw that if I divided the RAM value by 48, I got the gear ratio!

Engine # (Steam Engines in Bold)	Engine Type	0x06	0x07	16 bit Decimal	Measured Gear Ratio	Calculated Gear Ratio value/48
2	0-8-0	2	A0	672	14.00	14.00
4	Diesel	1	F8	504	10.50	10.50
5	2-8-0 H9	3	60	864	18.00	18.00
6	Berkshire	3	0	768	16.00	16.00
8	Diesel	1	F8	504	10.50	10.50
9	Diesel	1	F8	504	10.50	10.50
10	Diesel	1	F8	504	10.50	10.50
11	0-6-0	3	60	864	18.00	18.00
12	Diesel	1	F8	504	10.50	10.50
13	Diesel	1	F9	505	10.50	10.52
14	Diesel	1	F8	504	10.50	10.50
15	Diesel	1	F8	504	10.50	10.50
Wabash E8	Diesel	1	F8	504	--	10.50
Jake's 2-8-0	2-8-0	5	40	1344	28.00	28.00

Since I believe that all engine calculations run off of tach counts, I'm guessing that this is a value for tach counts per revolution of the driving wheels which can be used to directly calculate the gear ratio.

But what was this 48? In my analysis, I used factors that might be expected to be used in engine calculations. For example, 48 was the "O" scale ratio. I used other factors like 5280 feet/mile, 12 inches/foot, 25.4 mm/inch, 110 feet/Smile (fans of Frank Ellison will recognize that a Smile is a scale mile, in "O" scale that would be 110 feet).

I'm going to jump ahead a bit and let you know that eventually I realized that the 48 in this equation is NOT the "O" scale ratio. Stay tuned for the real information.

The Trip Odometer and RTC – RAM 0x08-0x09

The DTO command reads out the trip odometer from the engine. The Remote converts that to a value in Smiles and displays it. In my RTC program I can look at the raw value as returned by the engine.

When I was developing the RTC program, I figured out how to take the DTO raw value and convert it to Smiles. This was mostly done by trial and error. By looking at the radio transmission between the Remote and the TIU, I could see where the DTO raw value was read (from 0x3D-0x3F). I also saw the Remote read up another 16 bit value from RAM. I looked at the sound file and I could see that that value was stored at locations 0x08-0x09.

I found that I could match the DTO value presented by the Remote if the RTC program would take that raw value and divide it by the value in RAM 0x08-0x09 multiplied by 11. Why? I don't know but the equation worked. Keep on reading….

The Trip Odometer and RAM

On my layout, I measured out 110 feet. That is a Smile. I read up the trip odometer and these important RAM addresses. I then ran the engine for a Smile. I again read up the trip odometer and the RAM. Then I started looking for a pattern.

I'll use the results for my P&LE #2060 diesel here but I confirmed this for several diesel and steam engines.

First, I noticed that the value in RAM 0x08-0x09 was constant (for each engine), it never changed. For this engine, the value was 21637.

I took the ending DTO raw value and subtracted the beginning raw value. I got 237012.

I did some math first. Look at this constants table (for diesel engines):

I measured the driving wheel diameter as well as I could	0.889 inches
Calculate the circumference (C = pi *d)	2.795 inches / revolution
I know the gear ratio from my finger twirling	10.5 : 1
I know the number of stripes (NS) on the tach wheel, I counted them.	24
I know feet/mile	5280 feet/mile
I know feet/Smile	110 feet/Smile
I know inches/foot	12 inches/foot
I know inches/Smile	1320 inches/Smile
Calculate revolutions of driving wheel/Smile (revs = inches/Smile / circumference)	472.23 revolutions of driving wheel over 1 Smile
Calculate stripes/Smile (472.23 * 10.5 * 24)	119000 stripes/Smile

Interesting – the calculated stripes/Smile value of 119000 is almost exactly one-half of the measured DTO raw value delta of 237012 (tach counts/Smile).

I concluded that the DTO value is the stripes/Smile value times two and that the factor of two is because a tach count occurs on both the leading and trailing edge of a stripe. I believe that my measured tach counts are a little off due to my measurement of the 110 foot Smile and my (in)ability to stop the engine exactly at the 110 foot mark.

Taking a step back right now, this showed me that the divisor of 48 in the calculation of gear ratio is most likely two times the number of stripes (NS = 24) which is 48 (NOT the "O" scale factor).

I'm guessing that this (2 * NS) value is important throughout these calculations. I call this value "tach counts" and is equal to 48 counts for each revolution of the tach wheel.

I returned to looking at the RAM values to see if there was a location which contains a value that can relate the tach counts to distance. We want a RAM value that when multiplied or divided by some value based on the values listed above would give the DTO tach counts/Smile value. I immediately went back to the RAM 0x08-0x09 used by RTC to calculate the trip odometer Smile value.

One thing I learned in college (Carnegie-Mellon, BSEE '70) was when working with an equation, you need to have the units work out correctly. This concept really helps you build an equation.

So what equation could I put together that lets me relate Smiles to tach counts and has the same form that I developed for the RTC program. My guess for units:

DTO tach counts/Smile = RAM 0x08-0x09 tach counts/foot * 110 feet/Smile / 10

or

Smile = DTO tach Counts / (RAM 0x08-x0x09 tach counts/foot * 110 feet/Smile / 10)

so tach counts units cancel out, feet units cancel out and we are left with Smiles.

RAM 0x08-0x09 has to have units of “tach counts/10 feet” in order for the units to come out right and for the 110/10 to come out as eleven as I found in the RTC program.

So there was the next important engine parameter, “tach counts/10 feet of distance”.

We can work this result from another direction:

(12 inches/foot) / Circumference in inches) * gear ratio * 2 * NS tach counts * 110 feet/Smile
= 238002 Counts/Smile

then

238002 Counts/Smile * (10 feet/110 feet/Smile)
= 21637 Counts per 10 feet

That is the value in RAM 0x08-0x09. Whew!

Now to think about this completely backwards. We measured the driving wheel diameter to use in the above equations. The engine needs to be told the wheel diameter.

So we take the equation right above and we solve it for diameter given that we have the value in RAM 0x08-0x09.

(12 inches/foot) / (wheel diameter * PI) * gear ratio * 2 * NS tach counts * 110 feet/Smile
= Y Counts/Smile

and

Y Counts/Smile = Ram 0x08-0x09 / 10 feet/110 feet/Smile)

Two equations in two unknowns and solve for wheel diameter:

(12 inches/foot) / (RAM 0x08-0x09 * PI) * gear ratio * 2 * NS tach counts * 110 feet/Smile * 10 feet/110 feet/Smile = wheel diameter in inches

wheel diameter in inches = 0.888 inches
or
wheel diameter in mm = 22.56 mm

Tolerably close to my measured value shown above.

Now lets look at what I have so far:

RAM Address in Hex	
0x06-0x07	Tach Counts/ Driving Wheel Revolution used to calculate Gear Ratio
0x08-0x09	Tach Counts/10 feet

RAM 0x18-0x19

Here I got some help. Another experimenter thought this value had something to do with the chuffs from the steam engine. He noted that this value was zero for most but not all diesel engines and non-zero for steam engines.

I found that the values in this field sometimes matched the value in the 0x06-0x07 tach counts/driving wheel revolution field. So this field had to have a relationship to one driving wheel revolution. Closer examination of several steam engines showed that this value was sometimes one-half of the 0x06-0x07 value, sometimes one quarter, sometimes 1/8. That certainly fit into chuffs /driving wheel revolution. I experimented with changing the chuff rate using the Remote and then reading up this RAM location with RTC. The value was always ½, ¼, 1/8, etc of the tach count per driving wheel revolution.

As an example, for my engine #11, the value of RAM 0x06-0x07 is 864. I found that with a chuff rate of 2 set, the value of 0x18-0x19 was 432, exactly half. The engine would chuff every 432 tach counts or twice per driving wheel revolution.

RAM 0x64-0x65.

These values looked completely random. In all of the other locations that I looked at, there were always some values that were the same. For example, gear ratios for diesel engines had the same value for all diesel engines that I looked at.

When I see a value that looks completely random, the first thought that always comes to mind is Checksum! So what if this value was a checksum over the part of the sound file that contained the Engine Parameters? The simplest checksum would be just a 16 bit sum over the 16 bit words from 0x00-0x01 through 0x62-0x63 (total of 51 - 16bit words). I wrote some code in the ADPCM program to test this and that is exactly what it turned out to be.

RAM 0x0C-0x0D

I could read up this RAM word after the engine was added and assigned a DCS number. The value of this word is the DCS engine number which is one greater than the engine number as displayed in the Remote or on the RTC program screen.

The value in a virgin sound file is 1. This reflects engine number 0 which is a factory reset engine. When the engine is added to the Remote or to RTC, it is given a “real” engine number and that value is stored here.

It may be possible to set a value into this location and when the sound file is loaded into an engine, that engine will have its correct engine number right off. I’ve not tried this yet.

RAM 0x22-0x23

Looking at a lot of sound files makes me think that this word specifies the type of engine and maybe the features of the engine.

Byte 0x22: I can't make sense of this byte.

Byte 0x23 : This byte seems to define the engine type : 0x00 for steam, 0x05/0x85 for Diesel and Gas/Electric, 0x25 for electric

---

RAM 0x04-0x05

The meaning of this RAM word is still a mystery.

What is missing that the engine would need to calculate distance? I'm thinking that the wheel diameter is the missing parameter but I can't relate that to the values I see in this RAM word seen below. I notice that the diesel engines all have a value right around 666. Steam engines have values of 604, 574, 704, 456 and 357.

Engine # (Steam Engines in Bold)	Engine Type	0x04	0x05	16 bit Dec	Wheel Diameter measured (mm)
2	0-8-0	2	5C	604	27.55
4	Diesel	2	9A	666	22.56
5	2-8-0	2	3e	574	33.05
6	Berkshire	2	C0	704	36.27
8	Diesel	2	9B	667	22.56
9	Diesel	2	9A	666	22.56
10	Diesel	2	95	661	22.56
11	0-6-0	1	C8	456	27.11
12	Diesel	2	9B	667	22.56
13	Diesel	2	9A	666	22.56
14	Diesel	2	9B	667	22.56
15	Diesel	2	9B	667	22.56
Wabash E8	Diesel	2	9A	666	22.56
Jake's 2-8-0	2-8-0	1	65	357	32.00

Maybe you can see something that I'm missing.

The 16-bit values seem to be around 666 for diesel engines. But the value gets smaller as the wheel diameter gets larger for steam engines so I may be way off on my thinking of what this value means. The Berkshire's value is large and it has larger diameter wheels.

One interesting relationship that I noticed : The value in this location times the value in locations 0x08-0x09 (tach counts per 10 feet) equals 1,443,555 within about 0.1 percent.

Here is a summary of everything I could figure out:

RAM Address in Hex	
0x04-0x05	Unknown
0x06-0x07	Tach Counts/ Driving Wheel revolution to calculate Gear Ratio
0x08-0x09	Tach Counts/10 feet
0x0A-0x0B…...	Unknown
0x0C-0x0D	DCS Engine Number
0x18-0x19	Tach Counts/Chuff
0x22-0x23	Engine Type
0x64-0x65	Checksum

The meaning of RAM 0x0A-0x0B is not known or guessed at.

Copying a Sound File from one Engine to Another

What does this mean if you want to copy a sound file from a different engine into your engine?

I believe that all of these RAM locations need to be copied from the old sound file into the new sound file and then the checksum needs to be recalculated.

These RAM locations are related to physical attributes of the engine. They must stay the same when you use a different sound file. So look at the original sound file, write down the values (not the checksum). Use ADPCM to then insert the original values into the new sound file. ADPCM will calculate the correct checksum for you automatically. Then download the edited new sound file into your engine.

I have modified the ADPCM program with an option which will copy the entire 256 bytes at the beginning of a sound file to another sound file. Look for ADPCM v1.6.1 or later on my ADPCM web page. Here I'm assuming that copying all of the Engine Parameters from the old sound file is better than copying just a few.

If you try it, let me know if it works.

Addendum: My new favorite engine 30-20349-1 Wabash E-8 AA Diesel Engine Set w/PS 3.0 RailKing Catalog EL 2016 Volume 1 Sept 2016
