Summary: This Study Guide will serve as a more detailed explanation of how to program these timers and schedules. As always, it is strongly recommended to follow along with the recorded video of this session and/or the Slide Deck for this session.

This is Precision Digital's FIRST product to feature a real-time clock that can be used for control and alarming, so it is a very exciting topic, and one that I think helps make the ConsoliDator+ really shine!

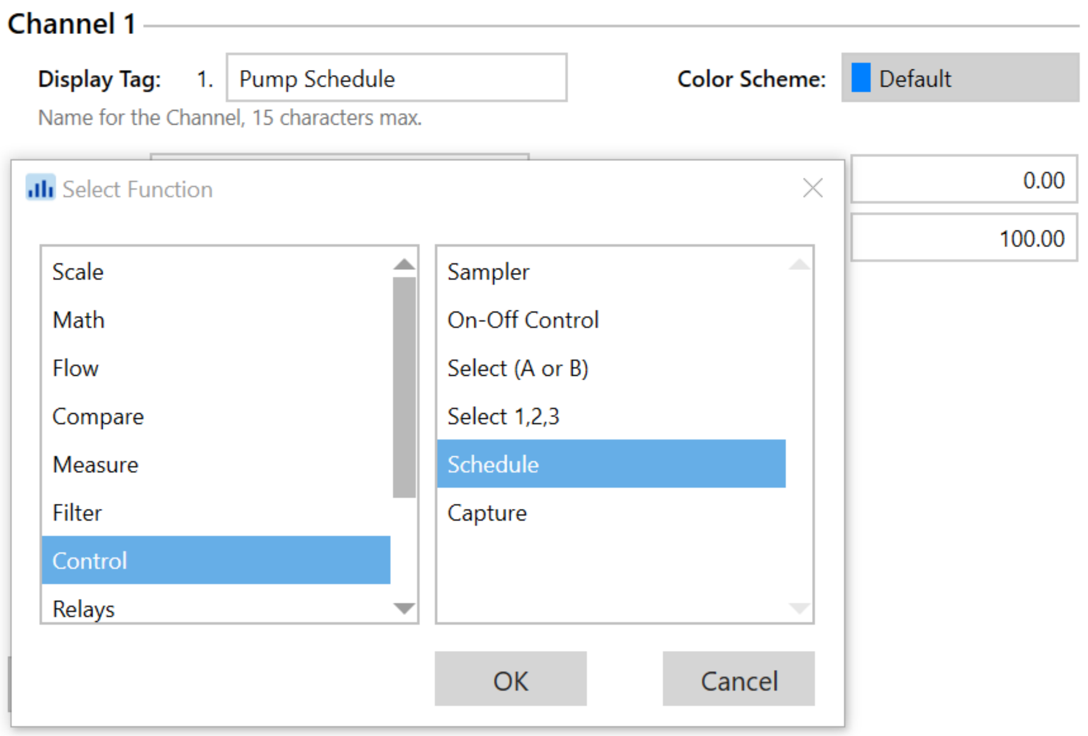
How to Schedule Events Based on Day and Time: We often see requests from customers who are looking to alarm or control something based on day and time. For example, one time a customer wanted to be able to have a pump turn on at Noon on Sunday and run for just a half hour so it doesn't sit all weekend long without running.

Unfortunately, when this request came in, we were unable to provide a solution since none of our legacy products have a real-time clock which can be used for controlling or scheduling anything!

Now that the ConsoliDator+ exists, we absolutely CAN provide a solution for that type of customer request, and can do a lot more than just that!

Actually, let's configure the ConsoliDator+ for that application we were unable to fulfill before! That will be a fun exercise!

To do this, we just need to create a new "Channel" (surprise, surprise, yeah?) and the "Function" we want can be found in the "Control" category, and it's called, "Schedule".


Channel 1

Display Tag: 1. Pump Schedule

Name for the Channel, 15 characters max.

Color Scheme: Default

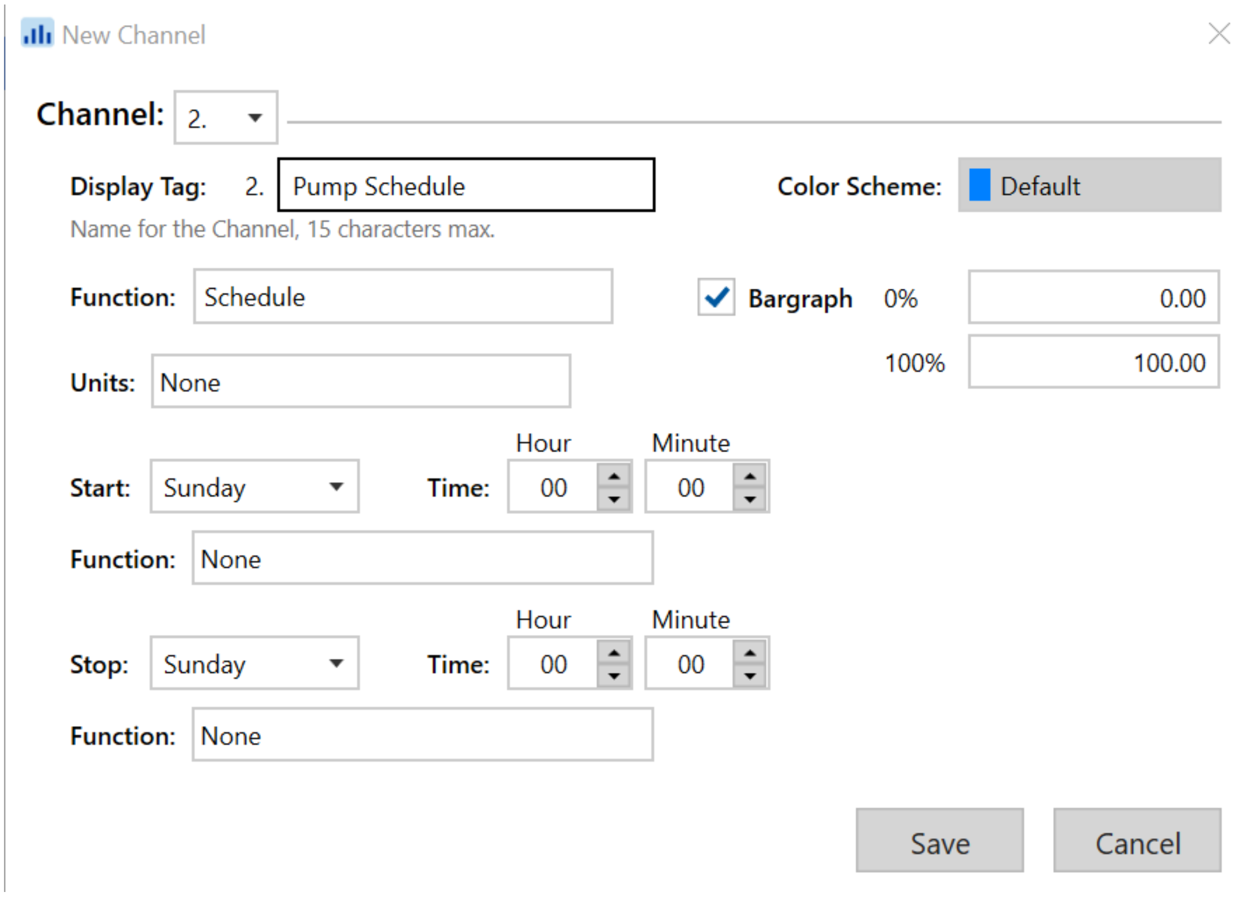
0.00

100.00

When you select “Schedule” as your channel function, there are only a few other fields we need to fill out for this particular application!

Firstly, we have to communicate with the ConsoliDator+ the day of the week and time of day we want it to perform some task. Then, we must tell it when we would like for it to STOP performing said task.

While we are at it, we also have to tell the ConsoliDator+ which task we want it to perform! And we are going to do ALL of that from just ONE setup window!



Channel: 2.

Display Tag: 2. Pump Schedule

Name for the Channel, 15 characters max.

Color Scheme: Default

Function: Schedule

✓ Bargraph 0% 0.00

100% 100.00

Units: None

Start: Sunday Time: Hour 00 Minute 00

Function: None

Stop: Sunday Time: Hour 00 Minute 00

Function: None

Units: For this type of channel setup, units is not going to be needed, for the most part. This is the type of channel that I would not personally add to a screen for viewing. Since this is more of a control Channel, this can run in the “background” without being visible on a Screen.

Start: This field tells the ConsoliDator+ which day to START the task we want it to perform. Your options here are each day of the week, “weekdays”, and “every day”.

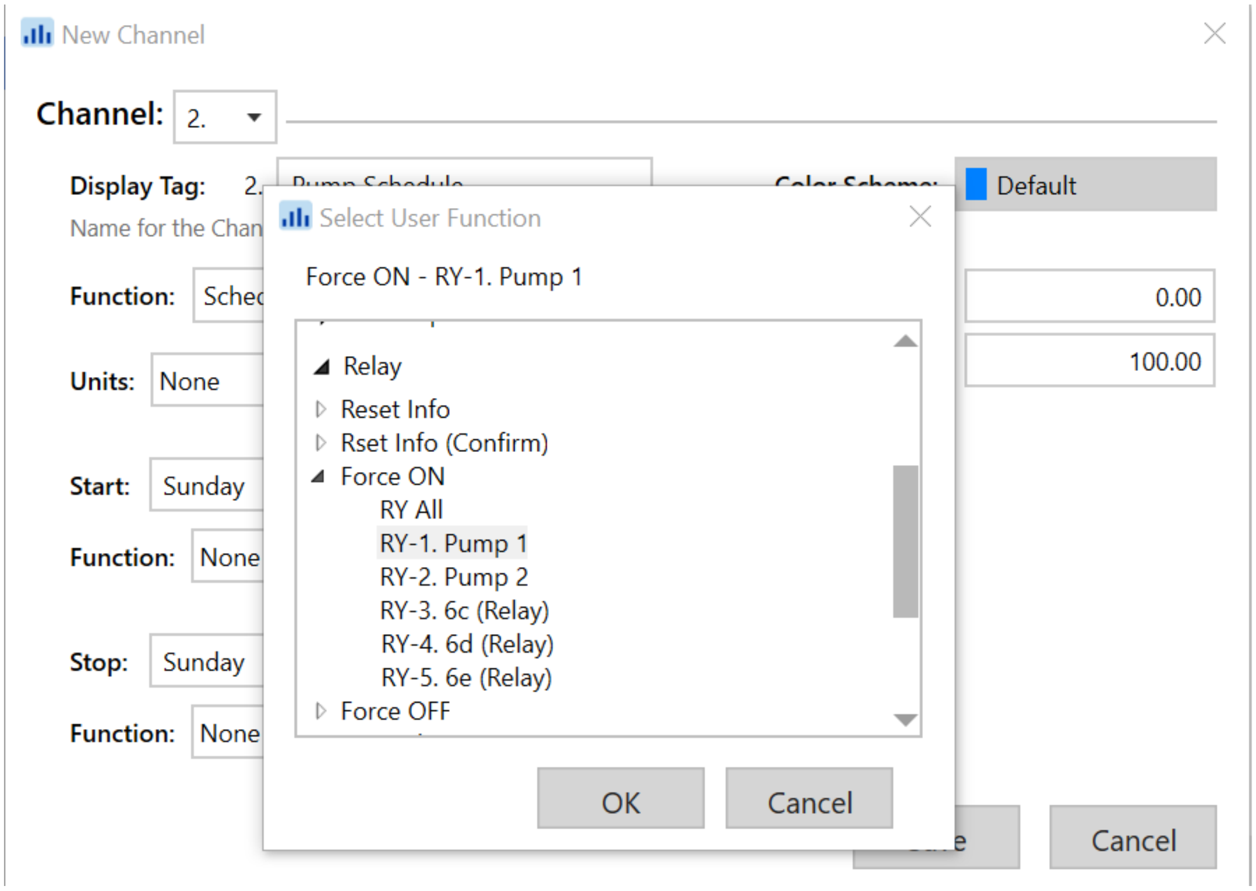
Time: This field tells the ConsoliDator+ at which time of the day we selected to perform the task. KEEP IN MIND this is a 24-hour clock (“military time”) and we wouldn’t want pumps turning on at 4 in the morning, right? Or… maybe we do!

Function: Now this is where we tell the ConsoliDator+ which task we want it to perform for us! You will notice, the available options are going to be VERY similar to how we program our “Soft Keys”. In fact, it’s almost EXACTLY the same, so we don’t have to go through all of that here.

For this particular application, we want to turn a pump on, and since that pump is tied to a relay output on the ConsoliDator+, we are going to want to select a function in the "Relay" category.

From there, we want to find "FORCE ON"

Then, we want to select the relay we want to energize. See below:


New Channel

Channel: 2.

Display Tag: 2. Pump Schedule

Name for the Chan

Color Scheme: Default

Function: Schec

Units: None

0.00

100.00

Start: Sunday

Function: None

Stop: Sunday

Function: None

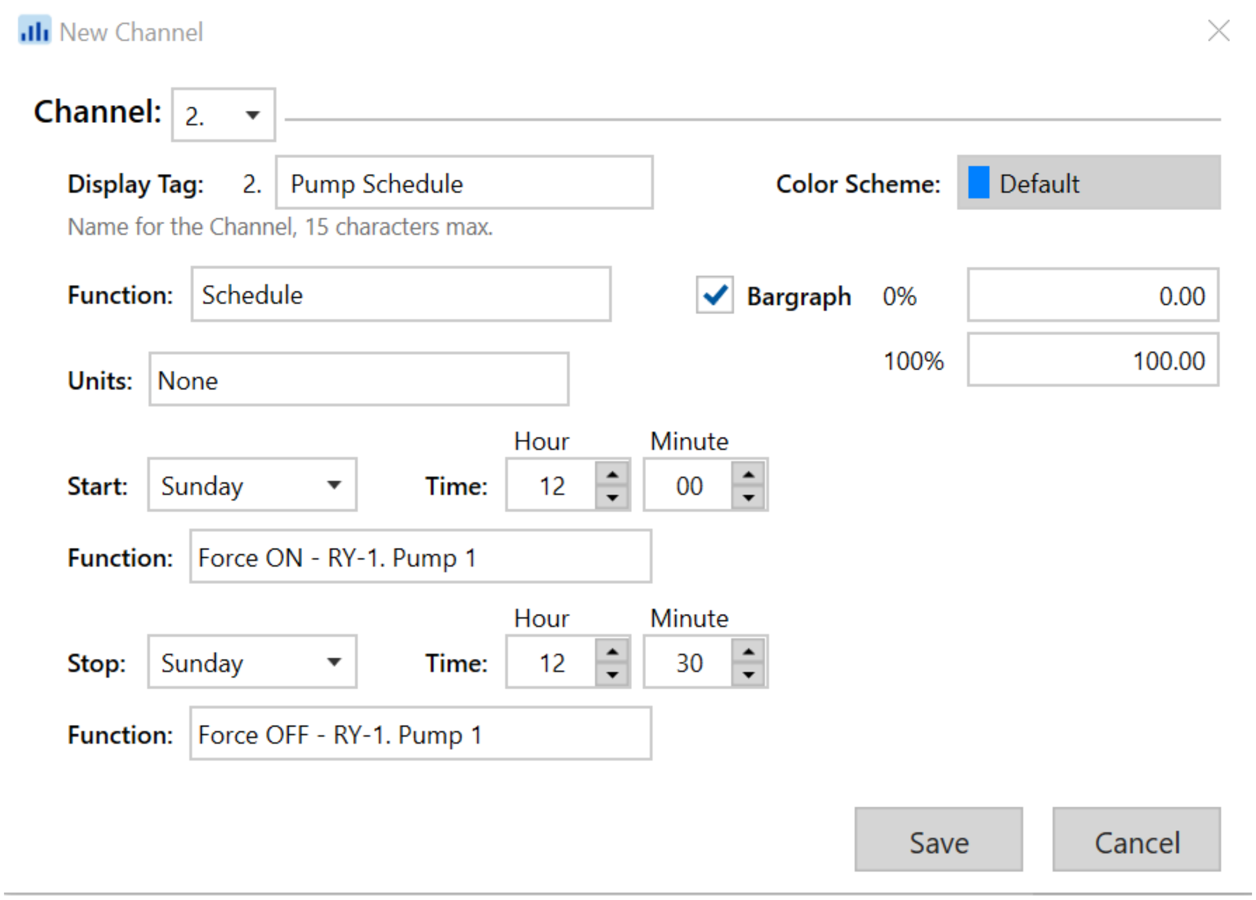
Select User Function

Force ON - RY-1. Pump 1

- Relay
  - Reset Info
  - Rset Info (Confirm)
  - Force ON
    - RY All
    - RY-1. Pump 1
    - RY-2. Pump 2
    - RY-3. 6c (Relay)
    - RY-4. 6d (Relay)
    - RY-5. 6e (Relay)
  - Force OFF

You can follow the exact same steps for STOPPING the task, except you want to make sure your "Function" is to "FORCE OFF" the correct relay.

Now that we have the entire thing figured out, let's take a look at what the completed setup screen will look like: (next page)



New Channel

Channel: 2.

Display Tag: 2. Pump Schedule

Name for the Channel, 15 characters max.

Color Scheme: 🟦 Default

Function: Schedule

✓ Bargraph 0% 0.00

100% 100.00

Units: None

Start: Sunday Time: Hour 12 Minute 00

Function: Force ON - RY-1. Pump 1

Stop: Sunday Time: Hour 12 Minute 30

Function: Force OFF - RY-1. Pump 1

Save Cancel

We are telling the ConsoliDator+ the following "Command" with this configuration:

"On Sunday at 12:00 pm, force on my "Pump 1" relay. On Sunday at 12:30 pm, force OFF my "Pump 1" relay"

If the customer wanted the pump to run for a whole hour instead, we would simply change the "Stop" time to "13:00" (remember, it's a 24-hour clock!).

Once that is completed, you have successfully configured the ConsoliDator+ to turn a pump on for 30 minutes every Sunday with no human intervention!

SIDE NOTE:

I have mentioned several times before that there are several different ways to accomplish the same goal with the ConsoliDator+. Much like with configuring different alarms, we can setup this type of schedule a different way, but it has its disadvantages.

For example, we COULD configure a new "Alarm" and have the "Type" be "Day & Time". From there, we can tell it which day and time to trigger, and for how long the alarm should remain active.

Here's the issue:

You would then need to tie the pump relay to THAT alarm. Since a relay cannot be assigned to multiple "Objects" (AND/OR Alarms being an exception to this rule) if you were to configure this schedule with an Alarm, then the pump will ONLY be controlled by that Alarm and will not turn on and off according to process conditions.

How to Alarm / Control Based on Elapsed Time: Since we can control something based on Day and Time, what's stopping us from controlling something based solely on elapsed time?

The answer is nothing! Nothing will stop us from controlling something based on elapsed time!

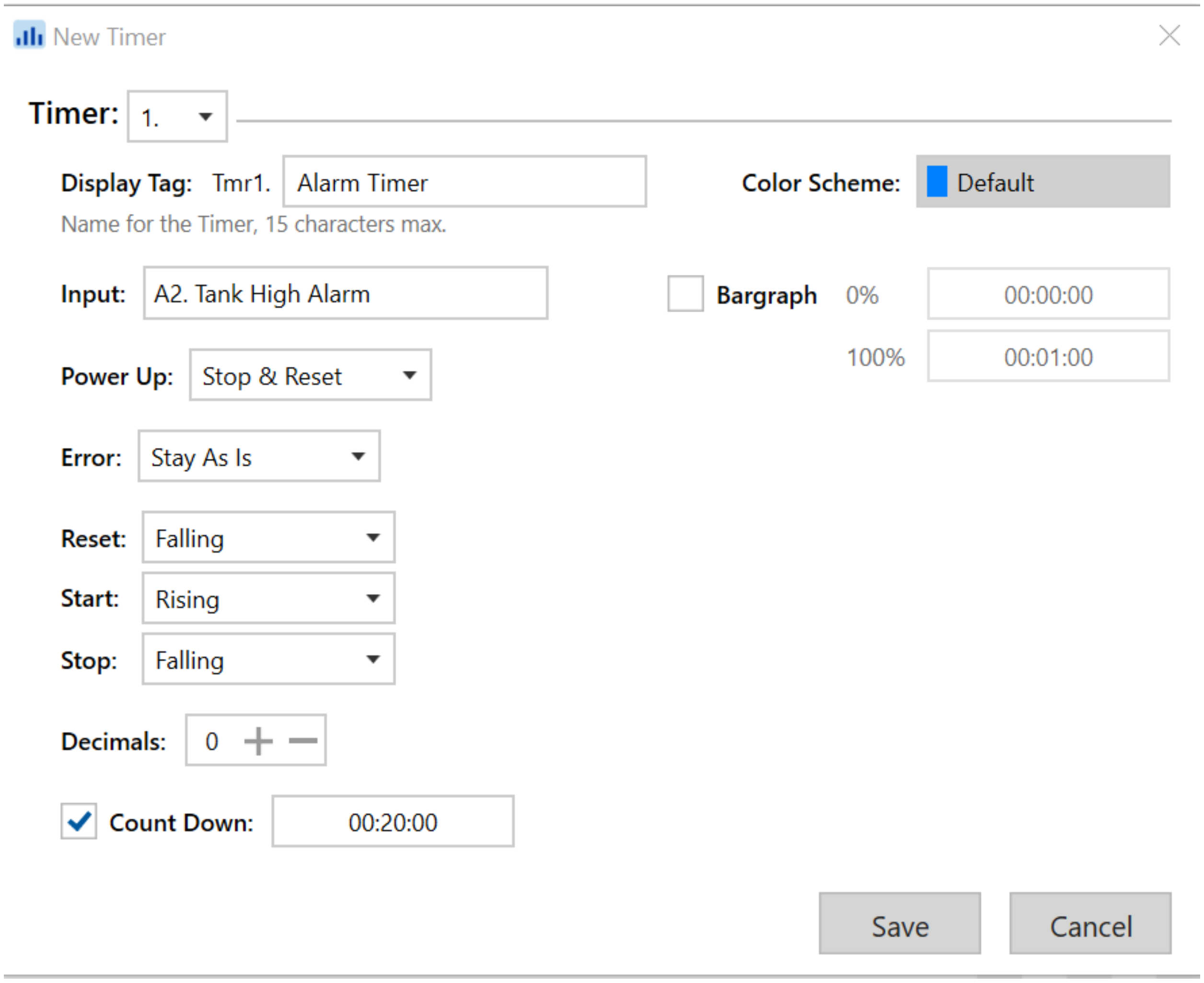
For this example, let's pretend that a customer has an issue with plant operators silencing or ignoring high level alarms, and then forgetting to go back and fix the issue which causes a ton of issues in their process. Their solution is to find a way to make sure the alarms are not ignored for too long!

I actually gave you a quick example of this type of application during one of our classes, so now let's learn how I did it!

We have a "Tank Channel" that is scaled from 0-100% capacity

We also have an alarm on that tank that activates when the tank gets to be 80% full and unfortunately, that alarm has a "silence" button on it which is training the plant operators to ignore the high level alarm! This does NOT make the alarm go away and it does not clear the alarm status. It simply turns off the annoying horn!

Now that the scene is set, we will need to create a new "Timer".


New Timer

Timer: 1.

Display Tag: Tmr1. Alarm Timer

Name for the Timer, 15 characters max.

Color Scheme: Default

Input: A2. Tank High Alarm

Bargraph 0% 00:00:00

100% 00:01:00

Power Up: Stop & Reset

Error: Stay As Is

Reset: Falling

Start: Rising

Stop: Falling

Decimals: 0

Count Down: ✓ 00:20:00

Because we are trying to monitor how long the high level alarm has been active, the “Input” to this timer should be the high alarm configured for the tank. This means the alarm is the one driving the timer, not the tank channel itself (although, we COULD do it that way but it’s just more steps).

The rest of the fields we need to fill out are for telling the timer when it should start, stop, and reset. We communicate this to the ConsoliDator+ with words like, “Rising” and “Falling” instead of giving it a particular setpoint!

You can think of it this way:

Our alarm will be active when the tank **fills to** 80% capacity and will turn off when it **empties to** 50% capacity.

So, we want the timer to START as the level of the tank RISES to 80% capacity, and we want it to STOP and RESET as the level of the tanks FALLS to 50% capacity.

For example, let’s pretend this were for a LOW level alarm. In that case, we would want the alarm to START as the tank level FALLS to X% capacity and RESET as the level of the tank RISES back to X% capacity.

“Count Down” simply allows you to start the timer from a specific hour/minute/second and the timer will count down to “00:00:00” instead of STARTING at “00:00:00” and counting up to infinity.

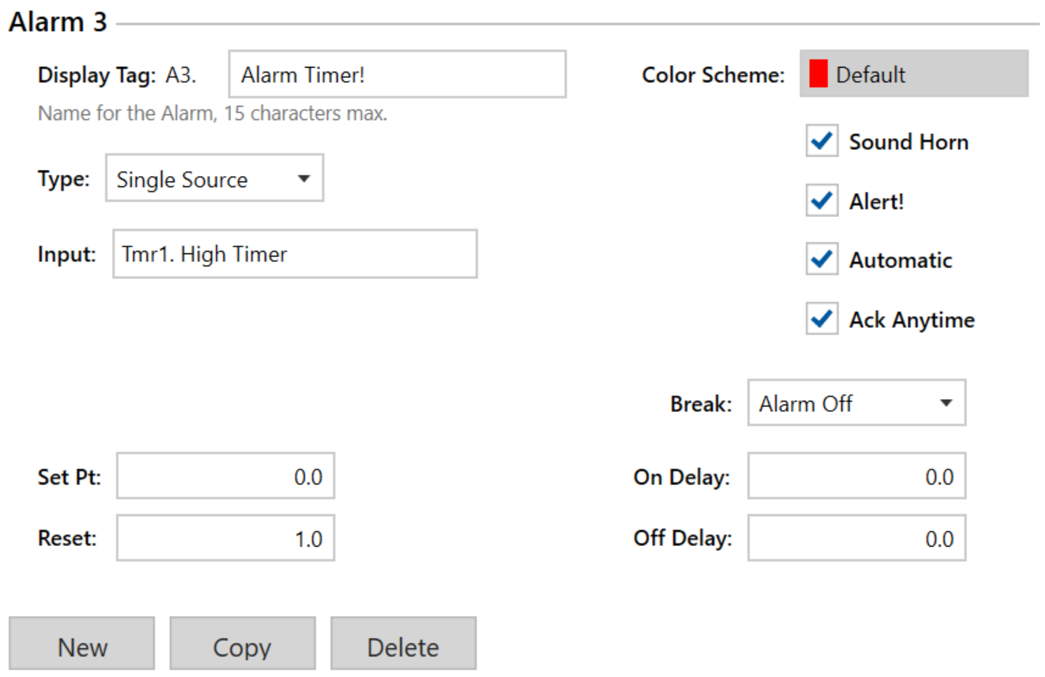
In our example, when the tank climbs to 80% capacity, our timer will start at 20 minutes, and it will count down to “00:00:00”. If the tank empties to 50% capacity before the timer gets to “0”, then nothing will happen.

However, if the timer reaches “0” before the alarm condition is cleared, we want to energize a relay which will then override the silence button, turn on a larger horn, or maybe even send a text to the boss (with a third-party piece of equipment. Precision Digital does not support email or text message notifications).

To do that, we are going to create yet another “Single Source” alarm.

However, setting up this particular alarm is not going to be as intuitive as our other alarms, and that is because the “Input” to the alarm is going to be a “Timer”. So, we will have to communicate using logic (binary logic).

Here’s what I mean:


Alarm 3

Display Tag: A3. Alarm Timer!
Name for the Alarm, 15 characters max.

Type: Single Source

Input: Tmr1. High Timer

Color Scheme: 🟥 Default

✓ Sound Horn
✓ Alert!
✓ Automatic
✓ Ack Anytime

Break: Alarm Off

Set Pt: 0.0
Reset: 1.0
On Delay: 0.0
Off Delay: 0.0

This alarm type is a “Single Source” alarm, and the “Input” to this alarm is going to be the timer we just created. The tricky part comes in with the setpoint and reset point of this alarm.

You see, USUALLY we have the setpoint and reset points being part of a scaled channel, sort of like the high alarm on this tank (meaning, our “setpoint” for that alarm is 80% and the reset is 50%).

However, how are we supposed to communicate a set and reset point when our alarm is based off elapsed time?

Well, like I said before, we need to think of it in terms of binary logic.

0 = off and 1 = on

While the timer is counting down, we can consider that to be “ON” or “1”

When the timer reaches “00:00:00” we can consider that to be “OFF” or “0”

So, we want this alarm to trigger when our timer turns OFF (reaches “00:00:00”) and then go away when our timer is ON (the timer is technically “on” as it is counting, or when it is reset to 20 minutes).

The last part of this is simple! You can just tie a relay to this alarm, and now it is complete!

That said, clearly you can expand on this capability! For this example, we are just alarming based off elapsed time, but if you tied a relay to something else, like a pump, then you can see how we can easily control something based off elapsed time.

Keep in mind, we can also combine timers and combine alarms to really get as intricate as you need to! This example was meant to keep things simple, but hopefully you can see how easily this type of stuff can be expanded.

How to Configure Tank Filling Applications: Using timers allows us to really expand the capabilities of the ConsoliDator+, and one of the really interesting applications it can be used for is monitoring tank filling applications.

For example, let's say your customer has a GIANT tank, and it would take hours for it to fill up all the way. Well, rather than standing there for hours waiting to turn it off, the operator would like to be able to go do something else while the tank fills, but make sure to be back before the tank could fill up too much and spill over.

Well, to do that type of calculation, we need to know how quickly the tank is filling, the current level reading of the tank, and we also need to know the maximum capacity of the tank.

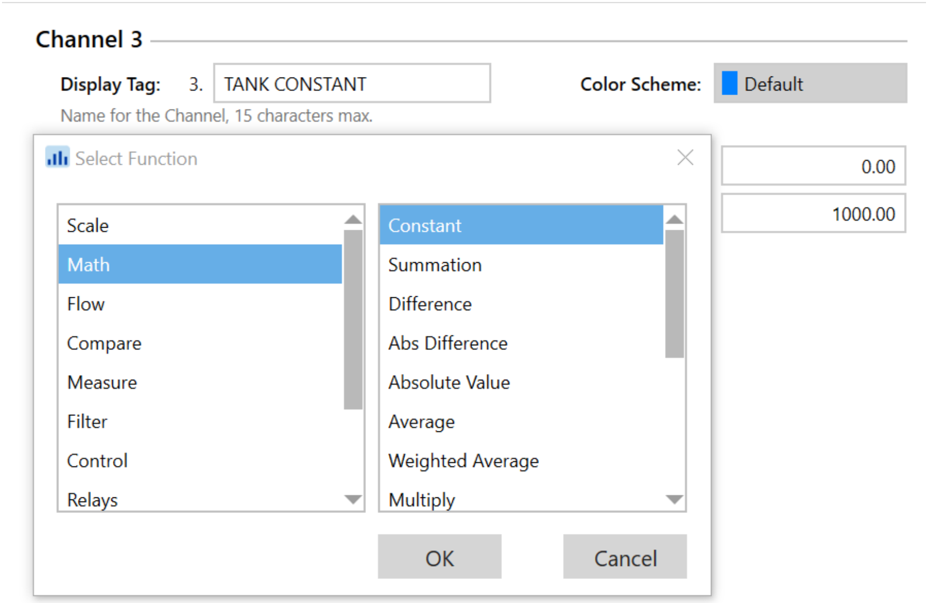
Now, you may think to know how quickly the tank is filling we will have to take an input from a flow meter. Well, we COULD do that, but let's just assume your customer doesn't measure the flow rate with a flow meter. Well, that's okay. The ConsoliDator+ can still work for this type of application!

That said, we are going to have to create quite a few "Objects", and we are going to have to use some of the basic math functions of the ConsoliDator+ - basic arithmetic.

The FIRST thing to do is create a new "Channel" which will represent your tank level reading. To keep things simple, you can just scale your "Tank" Channel for 0-1,000 gallons (most tanks that would use this type of setting will be MUCH larger, but this is just an example).

Next, we need to create a "Constant" channel. What I mean is, we need to know the maximum capacity of the tank and we need that information to do the math needed to make this whole thing work.

So, create another new "Channel", and for the "Function", we want to use the "Math" category, and then look for "Constant".


Channel 3

Display Tag: 3. TANK CONSTANT

Name for the Channel, 15 characters max.

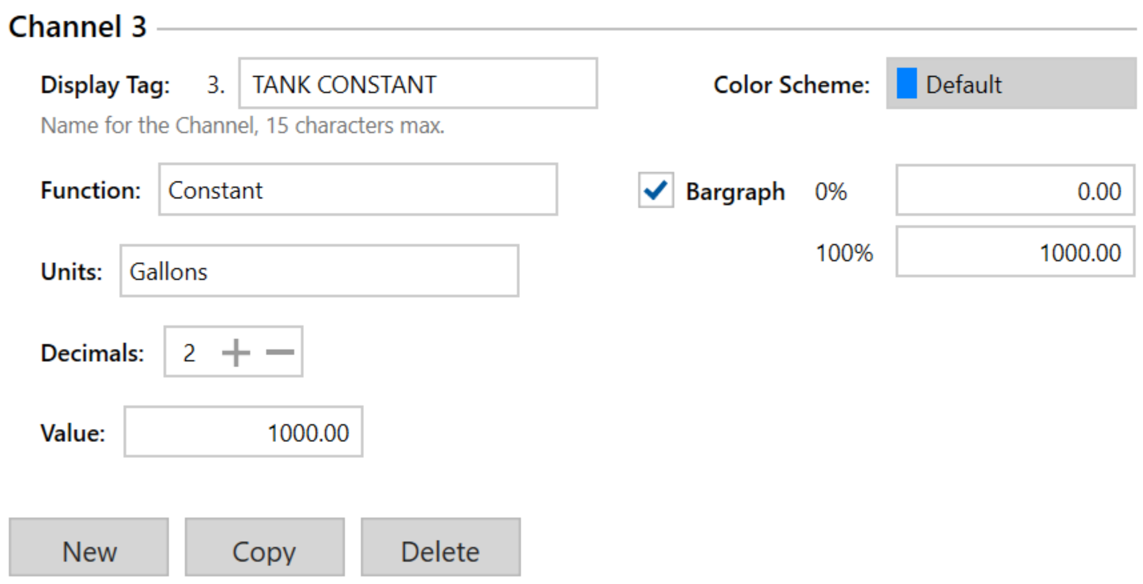
Color Scheme: Default

0.00

1000.00

Since our Tank channel is scaled from 0-1,000 gallons, we want the constant on this channel to be 1,000 gallons. The value of this channel will NEVER change. It will ALWAYS be at 1,000 gallons unless you decide to change it.

To do this, you simply choose “Gallons” as your “Units”, and then input “1000” as the “Value” – that is going to be the “Constant”.



Channel 3

Display Tag: 3. TANK CONSTANT
Name for the Channel, 15 characters max.
Color Scheme: Default
Function: Constant
✓ Bargraph 0% 0.00
100% 1000.00
Units: Gallons
Decimals: 2
Value: 1000.00

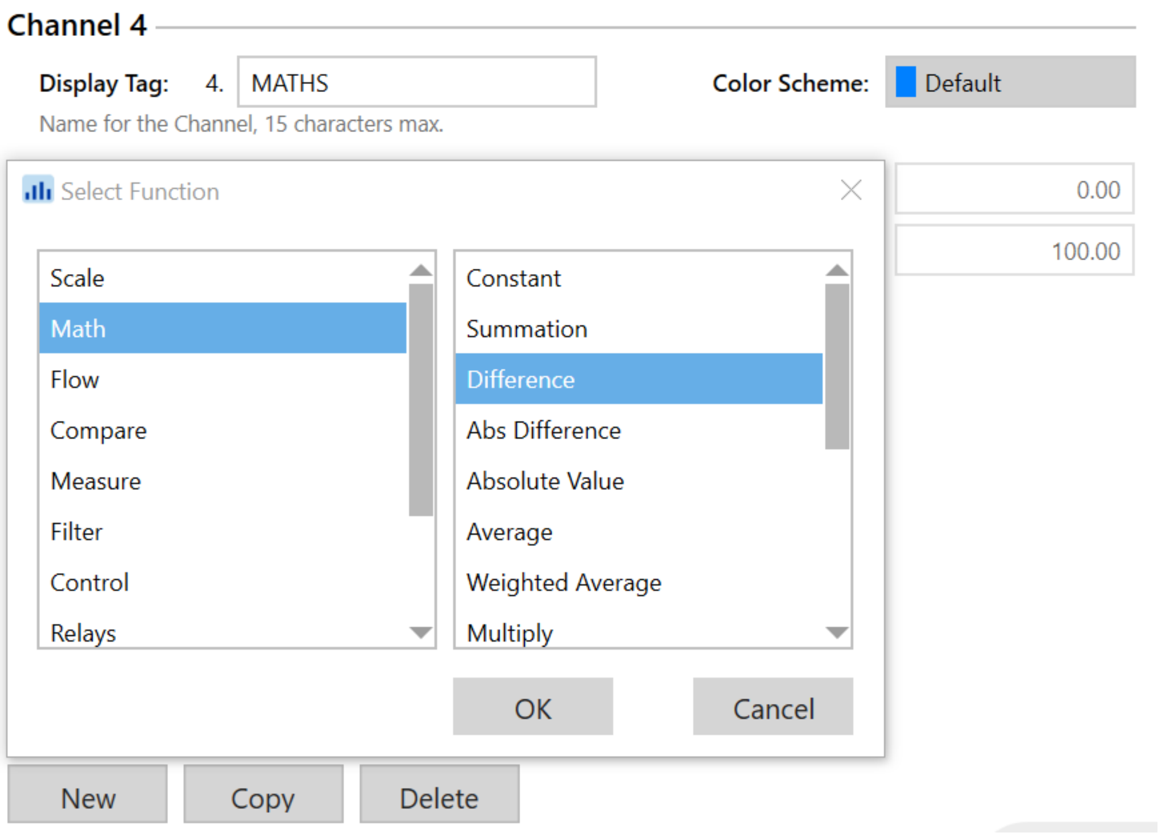
This channel is NOT going to be displayed on the screen, so you don’t have to worry about the decimals, bargraph, or color scheme. If your customer WANTS this channel on a screen, there’s nothing wrong with that, but it’s just going to be a static number and this channel does NOT need to be displayed on a screen for this to work.

Now that we have created these two “Objects”, we are going to create our first MATH channel! The math we need to do is quite simple, but just think about it from a logical standpoint.

If we are trying to figure out how long it’s going to take to fill the tank, then we are going to need to know how much volume is left to fill, right?

So, we want to create a channel that will take our “Constant” and subtract the current real-time level reading. The difference between these two values will be some number between 0 and 1,000. As the tank fills, the value of this math channel will approach 0.

Let’s create a new Channel, and for the “Function”, we want to use the “Math” category, and look for “Difference” (NOT “absolute difference” – that’s for something else entirely).



When you select this function, the ConsoliDator+ software will allow you to subtract up to 3 separate values, or Channels. For this particular application we only have to subtract 2 values, but if you ever need to use all three, then please be mindful of "order of operations" when using this!

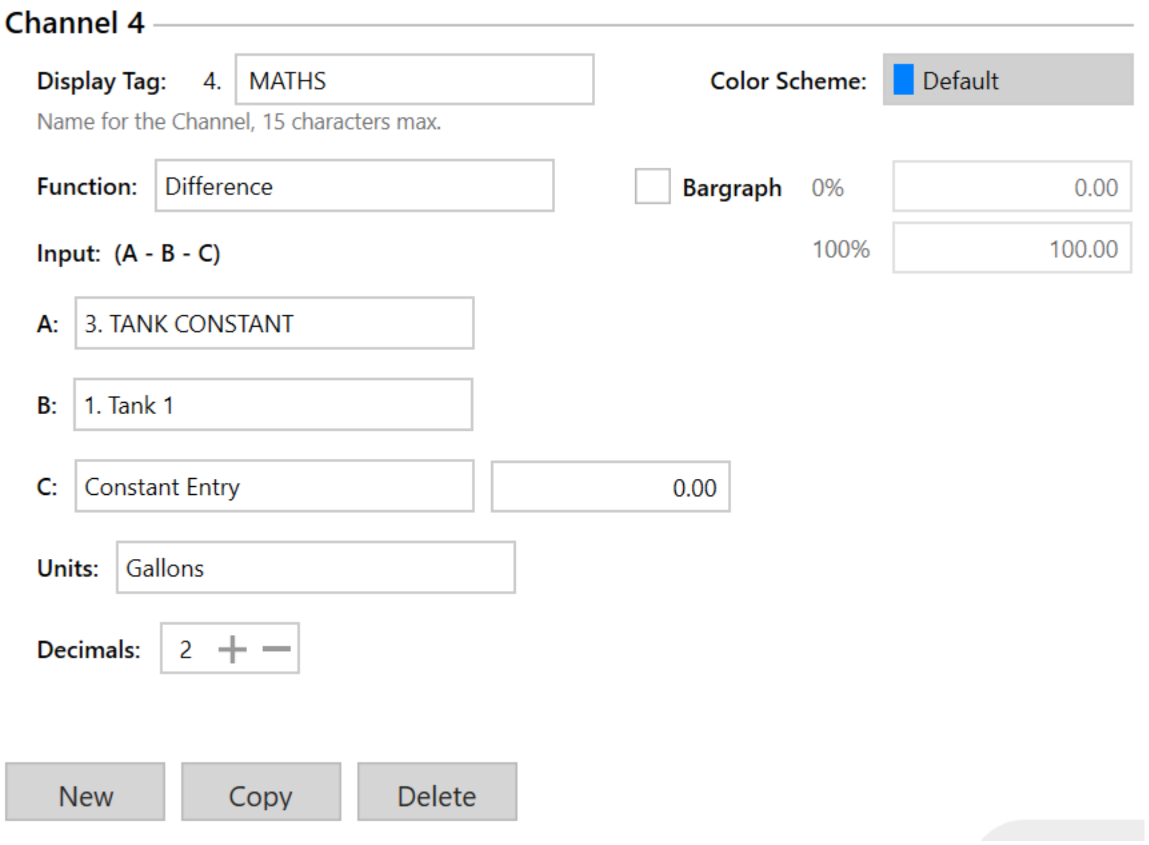
You will notice that formula given is (A-B-C).

For this channel, we want to have A = to our "Tank Constant" Channel

For this channel, we want to have B = to the current level reading Channel.

The default values are always going to be "0". So, you can just leave "C" alone!

Lastly, we want to make sure our "Units" are in "Gallons"!


Channel 4

Display Tag: 4. MATHS

Name for the Channel, 15 characters max.

Color Scheme: Default

Function: Difference

Bargraph ☐ 0% 0.00

100% 100.00

Input: (A - B - C)

A: 3. TANK CONSTANT

B: 1. Tank 1

C: Constant Entry 0.00

Units: Gallons

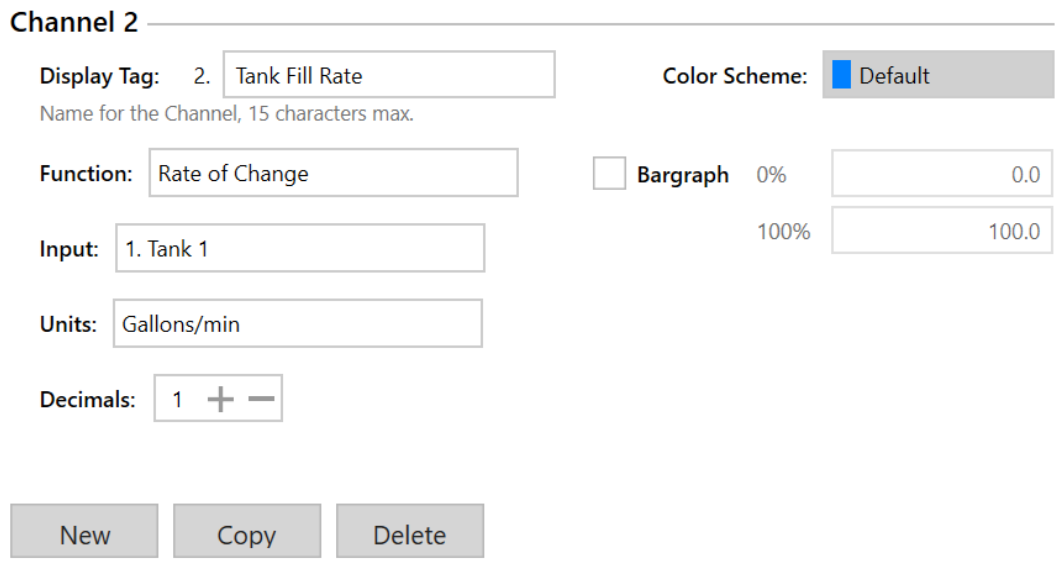
Decimals: 2

Again, this channel does NOT need to appear on the screen to work properly, so you do not have to worry about the decimals, bargraph, or color scheme unless for some reason the customer does want this channel to be visible on a screen!

The NEXT Channel we want to create (see, I told you we would be creating quite a few objects for this one!) is going to be a “Rate of Change” channel. You learned how to configure one of these channels when we went over “How to Configure For Leak Detection” last week, so you can review the Study Guide from Week 9 if you need a refresher on how this channel is created!

For this “Rate of Change” Channel, we want our input to be the real-time level reading, and we want our “Units” to be in “Gallons/min” – the time base is sort of critical for this. Since most flow rates are in GPM, and tank filling is measured in minutes or hours, it wouldn’t make sense to use “Days” or “Seconds”.

Below is an example of what this type of channel will look like when it is complete!


Channel 2

Display Tag: 2. Tank Fill Rate

Name for the Channel, 15 characters max.

Color Scheme: Default

Function: Rate of Change

Bargraph ☐ 0% 0.0 100% 100.0

Input: 1. Tank 1

Units: Gallons/min

Decimals: 1

Now, this channel would be great to have on a screen, so let's make sure we name it something meaningful like, "Tank Fill Rate", and let's also get rid of the bargraph. The bargraph is not needed for this, and it will also get confusing since the customer will most likely want the bargraph shown for the actual tank reading.

With that out of the way, we can now finish this application up with one last math channel!

Now, let's think of this from a logical standpoint. As our tank level increases, the time it will take to fill up completely will decrease. However, at the same time, if our fill RATE increases, the time it will take to fill up will decrease even more.

Or, if our fill rate decreases, that will affect the time it will take to fill in the opposite way.

So, how do we make sure our "Time to Fill" channel is going to be accurate while the fill rate and the level of the tank is constantly changing over time?

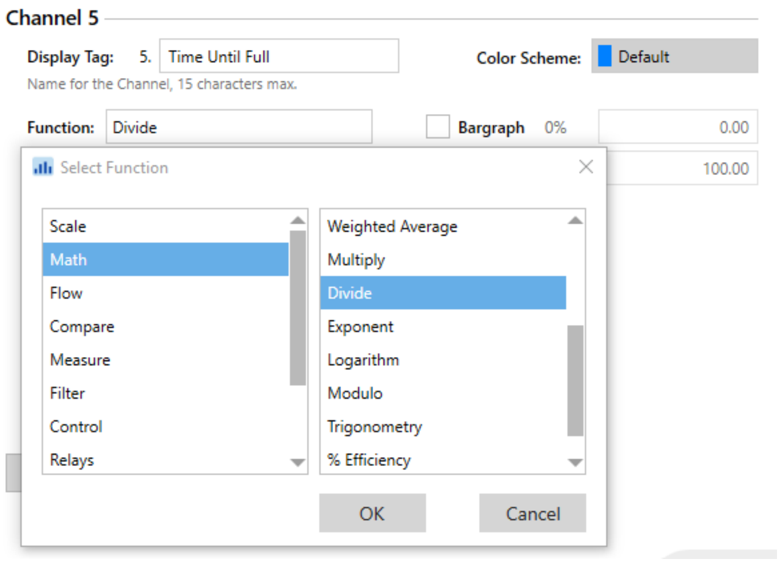
Well, I thought of this problem as if I were driving in a car using my GPS to tell me how long it will take until I get to my destination. I figured that the GPS knows my current position, my change in position over time, and the fixed position of my destination.

So, if I were to subtract my current position from my destination position, and divide that over my change in position (or, my velocity) over time, the result of that calculation will be the amount of time until I reach my destination.

With that, if my change in position (velocity – or, tank fill rate) increases or decreases, our "Time to Fill" answer will automatically compensate for that change and will still give us an accurate "ETA".

Now let's take a look at how we configure a channel to make this calculation!

Well, we want a new "Channel" with the "Function" category being "Math", and our operator being "Divide" since we want to divide these channels!


Channel 5

Display Tag: 5. Time Until Full

Name for the Channel, 15 characters max.

Color Scheme: Default

Function: Divide

Bargraph 0% 0.00

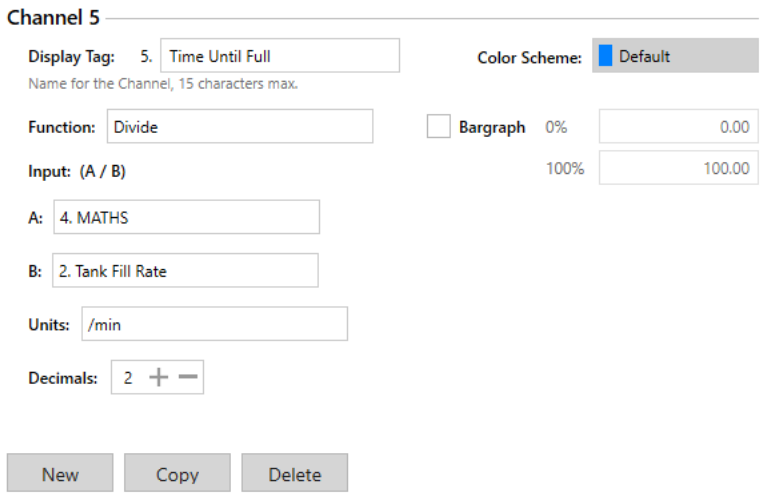
100.00

Select Function

Scale	Weighted Average
Math	Multiply
Flow	Divide
Compare	Exponent
Measure	Logarithm
Filter	Modulo
Control	Trigonometry
Relays	% Efficiency

Once we select this Function, we will then need to tell the ConsoliDator+ which Channels we want to divide!

Now, it is very important to make sure we put the correct channels in the correct spot. Otherwise, if we divide the wrong channels, the answer we get will be of no use to us at all!


Channel 5

Display Tag: 5. Time Until Full

Name for the Channel, 15 characters max.

Color Scheme: Default

Function: Divide

Bargraph 0% 0.00

100% 100.00

Input: (A / B)

A: 4. MATHS

B: 2. Tank Fill Rate

Units: /min

Decimals: 2

Let me explain what's going on here:

My "A" is named, "MATHS", and that channel is simply the difference between the maximum capacity of the tank (our constant channel) and the current level reading of the tank.

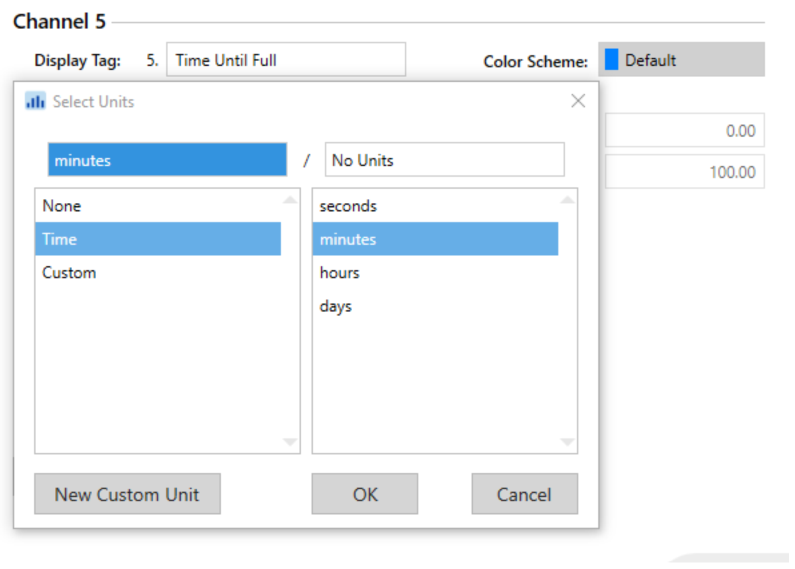
My "B" is named, "Tank Fill Rate", and that channel is quite literally what it sounds like. It's our "Rate of Change" channel that we created at the beginning of this.

Now, for our "Units" it will be a bit tricky, but let me explain what's going on here.

Ideally, you always want your Units to match in math, right? Well, if you look at the two channels we are dividing, one of them is measured in "Gallons/minute" and the other channel is going to be measured in just "Gallons".

However, we don't want the value of this channel to be in "Gallons", because that wouldn't make sense, right? We don't want to know how many GALLONS until our tank is full, we want to know how many MINUTES until our tank is full.

So, let's click on the "Units" box, and figure out how to make our "Units" something useable.



Once you select the correct unit of time, you can save this channel, and now you have successfully programmed this ConsoliDator+ to let us know how long until our tank is going to be at maximum capacity.

There are some laws and regulations out there which dictate a tank operator must be present for the first X number of minutes when the tank is filling, and X number of minutes before the tank is at full capacity.

If I remember correctly, the operator can walk away and do something else WHILE the tank is filling, but once our "Time Until Full" value gets to 15 minutes, the operator must stand by and monitor the last bit of tank filling to make sure nothing over spills.

So, the ConsoliDator+ can actually throw an alarm when that happens and that will let an operator know they need to get back and monitor the tank for the last few minutes!

You just need to create an alarm with the "Time Until Full" Channel as the "Input"!

You don't need my help to do that as this point!

How to Keep Track of Pump Runtimes: In previous sessions, I had shown you how we can view the accumulated pump run times and cycle counts if we add a "Relay" channel to a screen.

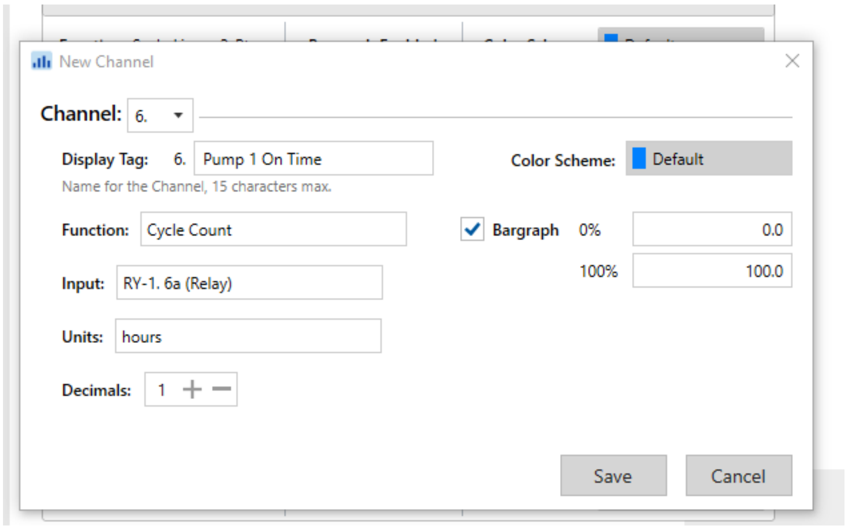
When we do that it will show us the relay status, it will show us the pump run time and the cycle counts which can be reset if need be.

However, we could also create a channel that shows nothing BUT the pump run times.

You may be thinking why that's even necessary, but it could be used to alert someone it is time to service the pump or something! Let's just assume that a pump must be broken down and maintained for every 100 hours of run time.

This is going to be an extremely simple exercise, and I'm confident you could probably figure it out without reading through this Study Guide.

So, let's create a new "Channel" with the "Function" being in the "Relays" category, we want to select "Runtime".


New Channel

Channel: 6.

Display Tag: 6. Pump 1 On Time

Name for the Channel, 15 characters max.

Color Scheme: Default

Function: Cycle Count

✓ Bargraph 0% 0.0

100% 100.0

Input: RY-1. 6a (Relay)

Units: hours

Decimals: 1

NOTE: *I quite literally just discovered a minor bug with the software. Although I selected "Runtime", the software shows "Cycle Count" as the "Function". However, when you load this configuration onto a ConsoliDator+, it will show you the Runtime. This is just a labeling error in the software.*

After you select your function, you can simply choose which relay you want to see the runtime for, and then select your unit of time. Again, if we want to alarm based off 100 hours of pump runtime, then it would be best to select "Hours" as your unit!

The last part of this would be to setup an alarm for when the pump reaches X hours of runtime to alert an operator that it is time to service the pump, or whatever they need to do to the pump.

In a sense, you can think of this as preventative maintenance, yeah? Rather than waiting for a pump to die before fixing it, you can create your own maintenance schedule, and the ConsoliDator+ can help with that!

With everything we have gone over in just the last three weeks, there is basically nothing YOU couldn't figure out on this device yourself. I gave you very basic examples of how to program the device, and I did that intentionally so that it would be easier to "transfer" these concepts to a real-life application you're working on!

Sincerely,

"Professor" Devin Gates
Cell: (508) 683-9034
Email: dgates@predig.com