

## PCT Programming Guide

This document outlines some of the issues to consider when programming the PCT

### Memory Organization

The PCT has 240 pages of non volatile memory available for your program. There are 3 requirements for the use of these memory pages.

1. Store the instructions for each individual step
2. Save the results data after a step terminates in order to use the virtual console.
3. Hold the memo

Each step will require 1 page to store the instructions for that step. The page number is the same as the step number. So if your program has 20 steps then memory pages 1 to 20 will be used.

The page assigned to save the step results, when the step terminates, is defined in the step instructions. This can be any unused memory page. You cannot assign a memory page required for the step instructions. For example if you are writing a bp3 program with 12 steps, do not assign memory pages 1 – 12 to store the results data from these steps. Assign higher numbers such as 100 +.

The memo is a text listing that can be used to describe and document the operation of the PCT program. It is stored in the program and becomes part of the bp3 file. When the bp3 file is loaded into the PCT the memo will be saved into the PCT memory pages. It is loaded in reverse order starting at page 240 and proceeding forward until the entire memo is saved. Each memory page can store 32 memo characters so you can store quite a detailed memo. However care should be taken to avoid conflict with the other requirements for the available memory. If the memo is too long it will get cut off when the program instructions are loaded. Also the memo may get overwritten during operation of the PCT if the results data are stored in a page that contains memo data. Do not allow this to happen. A good practice is to use pages 1-100 for step instructions, Step 101+ for saved results & the remaining can then be used memory for the memo.

For example if you have a 20 step program: Pages 1-20 will be used for the step instructions. You can then assign pages 101-120 for results data and you can use 120-240 to hold the memo. This will make it easier to design, write & debug your bp3 program.

If you do not require results data for the console then set the Save Data to memory page = 0 . This means “Do not save step results” and will avoid using any memory. This is good practice. Do not save any more data than required.

Other data is also stored by the PCT such as the programming statements, calibration data, battery details, program name etc. etc. However these other items have their own separate memory and are not competing for any of the 240 pages.

### Programming instructions

There are up to 32 routing statements that can be specified for each program. These statements are used to control the operation of the program. They are used to specify how each step will terminate and what will happen when the termination occurs. These routing statements can be assigned to any number of steps. They can be assigned to a particular step as either Termination statements or Conditional statements.

While your PCT program can have up to 32 unique routing statements, you can only assign a maximum of 12 routing (Termination + Conditional) statements per step.

Programming statements can consist of any of the following test items:

Voltage (V)	Current (mA)	Step Time (Min)	Total Time (Min)
Charge (mAH)	Neg Delta V (V)	Temperature (°C)	dT/dt (°C/Min)
IR Test (mOhms)	Counter 1	Counter 2	Counter 3
Counter 4	delta Output (mAH)		

Programming instructions can be more useful if they do not specify a Go To Step (i.e. set Step = 0). Whenever the Go To Step is 0 it instructs the PCT to route to the next numbered step. This is very common and allows you to assign the same program instruction in many different steps. With this strategy you will need fewer program statements. Also the program will be easier to read, understand, modify and debug.

The screenshot shows a window titled "Program Statements" with a table of four conditional routing statements. Each row contains a step number, an "IF" condition, an operator, a value, the action "THEN GO TO", a target step number, a counter value, and a "Clear" checkbox.

No	Parameter	Operator	Value		Step	Counter	Clear
1	Voltage (V)	<	10.5	THEN GO TO	12	0	<input checked="" type="checkbox"/>
2	Charge (mAH)	>=	1200	THEN GO TO	8	1	<input checked="" type="checkbox"/>
3	Step Time (Min)	>=	60	THEN GO TO	18	0	<input checked="" type="checkbox"/>
4	Neg Delta V (V)	>=	.15	THEN GO TO	7	0	<input checked="" type="checkbox"/>

Below the table are buttons for "Previous 4", "Next 4", "Clear", "Print", and "Ok". At the bottom, a note states: "Set Step = 0 to automatically go to the next step".

### **Termination Routing Statements**

A termination statement is a routing statement that describes a condition which will cause the presently running program step to end. The termination statement consists of the condition to test and what action to take when the tested condition is met. Each step is assigned up to 12 possible termination statements. The termination statements are selected from up to 32 different programming statements available for each program. With each step you specify which routing statements to use for step termination. As long as no assigned Conditional Statements are true (see below) then the program routes to the step identified in the assigned Termination statement that caused the step to terminate. The assigned termination statements can also assign one of the 4 counters to increment when this termination statement is executed by the PCT.

### **Conditional Routing Statements**

Users can also specify Conditional statements with each step. This is optional and provides for alternate routing if specific conditions are met. With this feature the routing can branch to different steps depending on various parameters. If no conditional statements are specified then routing will follow to the step specified in the termination step that caused the termination. If conditional statements are specified then routing will continue based on the conditional statement if it is true.

Conditional statements are optional. They are not required but they can be used to provide for more complex program routing that will be based on measured parameters. These are useful for writing programs that stop when some conditions exist or possibly go on to execute more steps or loops when other conditions exist. This can allow you to have the PCT program finish sooner by running for only as long as required. For example if the battery test runs successfully then the program can stop early to save time, however if more cycles or steps are required then the program will only run longer when required.

Conditional statements are only checked once a termination has occurred. This is different than the Termination statements which are continually checked during the step. So once any termination has occurred all conditional statements (if any) will be checked. If any conditional statements is true then the Next Step, Counter and Reset parameters of the true conditional statement will override the Next Step, Counter and Reset parameters in the termination statement that generated the termination. The termination will still occur as per the termination statement but routing will be controlled by the true conditional statement rather than the termination statement.

If there are more than 1 conditional statement then only the conditional statement that is true will be executed. If there are other additional conditional statements that also happen to be true then the last true conditional statement will take

priority. This is the conditional statement with the highest number. Once a termination statement terminates the step, all listed conditional statements are examined. Starting with the lowest number and continuing with the next higher number until all conditional statements are examined. The last conditional statement to be examined, that happens to be true, will be executed. If a conditional statement with a higher number also happens to be true it will over-ride and take precedent. If a conditional statement with a higher number is not true (false) then it will be ignored. If no conditional statements are true then the termination statement that generate the termination will control the routing.

By having the last true conditional statement become the one that controls the routing you can create very powerful selection algorithms to create selective routing. This can be used to implement a lookup table. For example if you want to route to different steps depending on the voltage then you can write the conditional statements to look like:

- (1) If voltage <12.0      Go To Step 16
- (2) If voltage < 11.8      Go To Step 18
- (3) If Voltage < 11.6      Go To Step 20
- (4) If Voltage < 11.4      Go To Step 22
- (5) If voltage < 11.2      Go To Step 24
- (6) If voltage < 11.0      Go To Step 26

Note that this is not the same as:

- (1) If voltage <11.0      Go To Step 26
- (2) If voltage < 11.2      Go To Step 24
- (3) If Voltage < 11.3      Go To Step 22
- (4) If Voltage < 11.6      Go To Step 20
- (5) If voltage < 11.8      Go To Step 18
- (6) If voltage < 12.0      Go To Step 16

Because the last TRUE statement is the one that controls the routing.

### **Hints and tips.**

You do not have to have a *Stop* function in your program. Any step with no termination program statements will not continue anywhere and will act like a *Stop* function. However the *total time* and *step time* will continue to increment. If you need the *total time* to stop increasing then you will need to end the program with a *Stop* function. If you stop a program by not declaring any termination program statements then you will need to do a hardware reset to restart the program for another battery.

#### Hardware Restart

If a *Stop* function is used, then you will need to restart the program to begin another charge or test session. You can do this by powering off/on with the power switch. This is a hardware restart. When the power comes back on the PCT will restart in *Step 1*. This method of restarting the PCT for a new session is really only needed when you cannot, or do not want, to implement an intelligent *Software Restart* (see below) to begin a new session. Normally you would use a software restart to have the final step check for a large voltage drop (battery is removed) and automatically proceeding to Step 1 to get ready for another battery to be connected. Step 1 would be programmed to check for a voltage rise (new battery is connected) and begin the session. However if you were using the PCT to test for an open battery situation then you may not want the open battery (no terminal voltage because battery is defective or battery protection circuitry has been activated) to restart the session.

#### Software Restart

In many applications you can write the program to automatically restart the program intelligently. This means you would implement a restart based on some event occurring. For example you could use a software restart to have the final step check for a large voltage drop (battery is removed) and automatically proceeding to Step 1 to be ready for another battery to be connected. Step 1 would be programmed to check for a voltage rise (new battery is connected) and begin the session. With this type of programming you never have to do a hardware restart. You can just keep reusing the PCT over and over again running the same routine again and again without ever having to power down and up.

A program implementing software restart will never need a *Stop* function.

A *Pause* function will need to have a termination program statement assigned, that ends the *Pause* step. For example you could use Time, Temperature, Voltage etc.. You must use a least one termination statement otherwise the *Pause* will never end.

The 4 counters are useful as a memory device or to flag a particular condition. This is also useful when programming the console. When a condition exists you can set a counter as a flag and then refer to this flag later to display a real time field to report on a previous results.

E.g. In Step 6 you may have: If mAH>1000 then go to step 14 and increment counter2.

Now in future steps you can just check counter 2 to determine if there was a Pass or Fail for display purposes.

Also since Real Time data, in the console, can only be displayed in it's own grid, you can refer to a counter in a save step in order to allow you to display saved data that can be displayed in an alternate grid. However remember that Saved Data (including counters) is not real time. So a saved data counter from a previous step can be different than the actual real time counter now. Only Counter 1 and Counter 2 are available as saved data. So if you need to refer to these for console display purposes you should use Counter 1 or Counter 2. All 4 counters are available in real time to direct control of the program and for real time display. Counter1, 2 & 3 will clear when the reset step is encountered. Counter 4 will not. Counter 4 is a special counter that can be used to count the number of sessions completed. This is useful for keeping track of productivity results.

Alternatively you can just momentarily pass through (Term if Step Time(m)>.1) an interim step in order to record a situation (like a PASS or FAIL). Then in a future step you can perform a calculation on the Saved Data from this interim step to see if the situation occurred. I.e. program it so the interim step will only occur when the condition exists (such as PASS) then in future you can view the saved data from the interim step and if Step Time >0 then display a PASS on the console.

### Session Reset

In the programming options you specify which program step causes a session reset. Once the program encounters the step designated as a reset step then all data and information from the previous session will be cleared. This data will no longer be available. This could be set up as step 1 so that every time you turn on the PCT you will clear the previous session's data. Alternatively if you want to preserve the data for later viewing set the reset step to a higher number so that the new session won't begin until the battery is connected.

A session reset will clear the Total Time timer as well as Counter 1, Counter 2 and Counter 3. Note that Counter 4 is a special counter that will only be cleared on Power Up. This way you can use Counter 4 as a counter that will not get cleared during a session reset step. This is useful for applications such as monitoring production. If you are using your PCT to perform many repetitive tasks then Counter 4 can be used to count the number of sessions. This is ideal for tracking productivity.

### Vector

In the programming options you can specify a Hardware (H/W) vector step. This is the step that the PCT will route to when the rear push-button is pressed. With this feature you could actually have 2 programs loaded in the PCT and the vector button can be used to force the program to the 2<sup>nd</sup> program routine. The vector push switch can be set up to have many different uses. e.g. Restart, reset, continue, stop etc. etc. You just need to specify a step to route to when the vector push button is pressed.

### **Program Debugging & Testing Tips**

When trying to test your new programs you can generate a termination using Step Time. This would be used just for testing and is useful for testing conditional statements. By adjusting the termination statement time interval you can simulate a variety of conditions to ensure that your program is routing as you expect. When you no longer need this test termination, just remove the statement number from the step.

There can be up to 12 program statements assigned to each step. These are assigned as either termination or conditional statements by listing the corresponding program statement number. Any combination up to a total of 12 statements per step is allowed. Remember, if you do not have any termination statements listed, then the step will never end. This acts just like a stop statement. Also remember that conditional statements are only checked once a termination occurs. When at termination occurs as a result of any termination statement then the conditional statements are checked.

When a conditional statement is checked and found to be true, the program will act on this statement and route accordingly. True conditional statements will be ignored and not be acted upon until a termination occurs due to a true termination statement.

The conditional statement, found to be true, will override the GOTO and CLEAR portions of the termination statement that caused the step to terminate. The conditional statement will not override the counter. The counter assigned to the terminating statement and the first true conditional statement will both be incremented.

It is possible to have logical errors in your program that cause the PCT to act in unexpected ways. For example if you have a term statement such as:

If Step Time < 65 then go to Step 0 (next step).

This statement will immediately be true at the beginning of a step causing it to terminate immediately. The same can be true for all statement types. You need to carefully consider the effect of each step and then test your program carefully. Please see the section on Testing Your Program. Failure to test your program can result in the PCT acting incorrectly. This can cause a dangerous situation resulting in battery damage, fire, explosion personal injury and death.

In your program a single programming statement cannot be used as termination statement AND a conditional statement. If you ever need an identical termination and conditional statement then you will need to specify two separate identical programming statements and use one as a term statement and the other as a conditional statement.

With 32 possible programming statements you will probably never need to use this many in one program. You do not need to assign all the programming statements. You can keep them in your program so that your program list can act as a library of available programming statements. This can make programs faster and easier to program and adjust.

~~If you have specified charge pulses during a discharge then you will need to enter a maximum Vreg value. If no charge pulses are specified then a Vreg entry is not required for a discharge function.~~ Feature not available at this time.

A programming statement containing a Counter for the parameter would never be used as a termination statement. The counters only increment upon a termination or condition. You would use programming statement containing a Counter for the parameter as a conditional statement. For example you may increment the counter each time the voltage drops down to a certain value for each discharge. Each time this termination occurs you could route back to a charge function and increment counter 1.

For example during a discharge you may use a termination statement such as:

If Voltage <= 10 then GO TO Step 2 and Increment Counter 1

And a conditional Statement such as

If Counter1 = 3 then Go To Step 6

This would cycle the battery 3 times. After the 3<sup>rd</sup> discharge the conditional statement will be true so the program can break out of the cycling loop and proceed to the rest of the program which might be a Stop step.

\*Note regarding mixing programming statements: Programming statements that use the mA or mAH as the parameter should not be used for both a charge and a discharge function. This is because the downloaded calibration adjustments are different if the program statement is used for a charge or discharge. This should not normally ever be an issue. In the event you do want to use the same programming statement in a discharge as well as a charge step then just define 2 identical programming statements. Use one for all your charge steps and the other for all your discharge steps. This only applies the mA and mAH parameters. All other parameters can be assigned to any type of step function.

Inserting some extra steps can facilitate program building, testing and debugging. If you have some spare steps scattered throughout your program then it becomes easier to build steps without having to readjust the termination and conditional statement numbers of existing steps. You can just use the spare steps.

### **Be careful with the use of the = function**

When programming avoid the use of the = function for terminating values that can change, such as Voltage, current, time, mah etc because the value could be lower then greater and never be equal. For example if the voltage is dropping during a discharge do not attempt to terminate the discharge with a statement like: If Voltage = 10 because the voltage could drop from 10.01 to 9.99 in the time between the PCT checks for a possible termination. Resulting in the discharge continuing right down to 0. This is an example of a logical program error and an example of how all programs must be tested carefully before they can be relied upon.

The correct terminating statement to use would be: If Voltage < 10 then .....

Keep this in mind when terminating on other values as well.

The = function should really only be used when programming statements with counters.

### **Using dT/dt termination**

Allow time for the battery to cool if you need dT/dt termination on two successive steps. You can insert a pause statement to allow the battery to cool and then terminate the step when  $T < \text{Value}$ . This way you can have a pause that lasts just long enough for the temperature to drop to normal.

dT/dt will only detect a positive increase in the rate of temperature rise. If the temperature is dropping then the dT/dt value will be zero.

### **Consider what will happen when the battery is disconnected**

If the battery is disconnected then the voltage that the PCT sees at the terminals may not be zero. If you have a step programmed to charge with  $V_{\text{reg}}=12\text{V}$  and  $I_{\text{reg}}=100\text{mA}$ . Then the PCT will generate either 12V or 100mA whichever is reached first. When no battery is connected, no current will flow and so the PCT will regulate the terminals to 12V. So the PCT will be measuring 12V.

If you are trying to use a step that will detect when the battery is connected then you should use a  $V_{\text{reg}}$  value that is below the expected battery terminal voltage (you could use  $V_{\text{reg}}=0$ ) and then terminate this step on a voltage  $>1\text{V}$ . The voltage can then only be  $>1$  when the battery is detected.

This is also important to consider when trying to detect a disconnected battery at the end or during a session. When the battery is removed the current will stop, so the PCT will regulate on the  $V_{\text{reg}}$  value if it is a Charge step. If you have been charging at constant current then the  $V_{\text{reg}}$  value must have been greater than the battery voltage. The PCT terminal voltage will then rise up to the  $V_{\text{reg}}$  value once the battery is removed. So to detect the disconnected battery you should terminate on the voltage being greater than some value. For example to charge a 6 Cell 7.2 V NiCd battery you could set the  $I_{\text{reg}}$  to 200mA (desired charge current) and set the  $V_{\text{reg}}$  value to 20V. Once the battery is removed the PCT terminal voltage will rise to 20V. The PCT can be program to terminate if the Voltage  $>18\text{V}$

Similarly if you are charging a SLA battery at constant voltage of 14.7V with a  $I_{\text{reg}} = 250\text{mA}$  (Maximum current limit) then when the battery is disconnected the PCT terminal voltage will continue to be 14.7V. In this case you can program the step to terminate (and route to step 1 to start over) when the Current is  $< 15 \text{ mA}$  or some low value that will only occur when a battery is removed. You can also use an extra step to restart the PCT if you think there could be a possibility that the current would drop to less than 15mA normally. In this case you could have the PCT go into an extra step when the current drops below 15mA. This extra step could be programmed with  $I_{\text{reg}}=0$  and  $V_{\text{reg}}=5\text{V}$ . Now this extra step would not provide any charge current because the  $V_{\text{reg}}$  value is below the actual battery voltage. Once the battery is removed the PCT will regulate to 5V and you could terminate this step (detect a removed battery) if the voltage  $< 7$ .

5V is a better choice than using  $V_{\text{reg}}=0$  because the voltage may never actual=0 due to normal offsets. For example if you set  $V_{\text{reg}}=0$  then the PCT may measure 0.05 V or some small value so the statement "If  $V=0$  then" will never be true.

So remember that during a Charge step the terminal voltage will rise to the  $V_{\text{reg}}$  value when the battery is removed. So to detect a battery removed situation during a charge you will need a statement such as:

IF Voltage  $> 10$  then Go To next step ( This has now detected that the battery has been removed )

### **Double testing**

You may want to consider the effects of vibration and user actions on the operation of your program. For example suppose you are performing a Lilon charge and want to terminate the charge when the current drops below a certain value. In the event the user touches, jiggles, shakes, vibrates or momentarily disconnects battery then the step may terminate. If you feel this could occur and you want to protect against this possibility you can do a double test of the current.

To do this route to the next step when the current drops below a certain value. The next step will perform the same function and terminate on step time in 0.1 minutes and route back to the previous step (normal charge step). Include a conditional statement in this second step that will check the current when the step terminates. If the current is less than your minimum then you can assume that the battery has been removed on purpose, however if the current at the end of the 0.1M is not less than the minimum then it was just a momentary drop in current and the program can then route back to the original normal charge step.

If you want to skip the clear then you cannot use step time to create delay with your double testing. In this case you could implement a counter situation. For example set Term If  $V < 6$  Go To next step. On the next step term If  $V < 6$  as well but

go back to a previous step & increment counter 1. On 2<sup>nd</sup> step also have a conditional statement that if counter1>10 then continue on to the 3<sup>rd</sup> step. This will create a 10 step loop to provide for the desired amount of delay for double testing.

### **Debugging**

Often it is very helpful to route to steps that are just used for parameter checking and routing and are not required for any length of time. On some of these types of steps you may choose to terminate on Step time (ex IF Step Time > 0.1 M) . When writing and testing programs with these types of steps you might want to use longer step times while you are building and testing your program. This will allow time to see the step displayed on the PC Monitor to confirm that the step is actually occurring and operating as expected. Once your program is working as required you can then shorten the step times to your desired values.

A nice way to develop your programs is to periodically print the program listing. This will provide a hard copy of everything you have so far and give you a place to mark revisions and plan how to proceed further. The hard copy makes it easier to visualize the operation of the entire program and spot items that need further consideration.

Developing and testing your program will take a bit of time so proceed carefully with thought and planning. Expect to make changes and add new functionality to your program as you proceed. As you program and test you will discover new things that you need to consider and incorporate into your program. Developing a new bp3 file is an iterative process of: Edit – Load – Test, Edit – Load – Test, ..... Edit – Load – Test .... until you are completely happy with the performance in all possible situations.

### **Smart programming**

There are several tricks you can do to perform some complex checking within your program. For example if you want to determine if the battery voltage is between 0.5V and 7.2 V you could have one step to test if  $V \geq 0.5$ . If yes route to next step. The next step is identical in function but has term statement if  $V < 7.2$ . This will only occur now if V is in the range of 0.5 – 7.2 because the previous step determined that it was not  $< 0.5$  V

If you want to route to 1 step if a parameter is > than a certain value and route to an alternate step if the same parameter is < than a certain value you can do this all in one step. To accomplish this, set the term statement to some value that will definitely occur. Ex if step time  $> 0.1$  min. Then have two conditional statements one for Parameter A  $\geq$  number X and the other for Parameter A  $<$  number Y. These two conditional statements will route to different steps. Note in this case the GOTO part of the termination statement will never be used because one of the two conditional statements will always be used.

Alternatively you could use the fact that only the last (higher number) is the routing that takes effect to create a powerful lookup table. Described earlier.

### **Remember the Battery's effect**

The battery of course is connected to the PCT during use. The battery has a voltage and there will be current flow at various times during the operation of your PCT program. You need to think about what the voltage will be at various times and how this may or may not affect the operation of your program. Also the battery voltage takes a finite amount of time to rise and fall once charge current begins and ends. An example is when charging a SLA battery at a constant voltage. You may want to move to a new step once the current drops below a certain point. If the following step is also going to regulate the voltage at some lower level, the current will probably be 0 for some small period of time until the battery voltage naturally drifts downward to the point that current will again start to flow. So if this second step is also going to terminate on some minimum current then it will probably terminate immediately because initially the current will be 0.

Also when using the temperature probe remember that the battery will take a finite amount of time to cool.

An easy way to avoid situations such as this is to insert a pause step as required that will give the battery time to stabilize before proceeding to the next step.

Another effect you should consider is the voltage drop across the leads and connectors as a result of current flow. Whenever current flows through the leads and connectors there is a small resistance and this will generate a voltage drop. When viewing real time data on the PC console & PC monitor, there are correction factors that adjust the voltage reading to remove errors associated with the test leads. However the PCT does not adjust for these values in the routing, conditional or saved data.

You will need to consider the possible effects of voltage errors due to current flow on how you manage these values. In many instances these voltage errors are insignificant and you can forget about them. However if the current is large and the voltage values are critical then compensate for them when you write the PCT program. For example you might want to do a discharge test using a discharge current of 1A and a low voltage cutoff of 10.5V. If the leads have a resistance of 0.1 ohms then there could be a voltage error of 0.1V. This would mean the discharge would terminate when the battery voltage is only 10.6V. So in this case you would change the low voltage cutoff to 10.4 volts knowing the current will be 1A during this portion of the test. When testing your PCT program for the first time you can monitor the actual battery voltage closely to accurately determine how much compensation is required.

Also note that the PCT system can't really do an adjustment to the routing values for lead resistance because the term statement is not dedicated to any particular step. The software cannot figure out what the current would be by examining the program because the same term statement could be used by more than 1 step. So users have no choice but to adjust the Vcut values in their routing statements in order to get the desired Vcut. This is not a problem. Just something that you need to be aware of. This is not a deficiency it is an inherent situation in how the system works and the user is fully expected to work with this knowledge.

The measured values in the Real Time Monitor and Console are adjusted based on the actual measured current. These adjustments assume a certain value for the lead resistance. If your leads are different than the assumed value then the on screen voltage readings can deviate from the actual values.

When testing your programs to see if the Vcut values are correct connect your voltmeter as close to the battery terminal as possible. Remember that you are interested in the actual voltage at the battery terminals and need to consider any voltage errors that the leads may introduce. None of this is a problem. Just something you need to deal with when designing your programs. Also this is easy to test and adjust as required.

### **Step Time vs Total Time**

Remember that Total Time is the time elapsed for all steps since the last session reset. Don't confuse this with Step Time which is the time elapsed for each step. If you specify Total Time in a program statement but actually mean Step Time you will cause the step to terminate prematurely.

### **Consider the User**

Remember that the person using the PCT will not be thinking about how the program is structured. They may remove or reconnect the battery at any time. You need to consider this when writing your program. If you want the PCT to revert back to the start of the program when the battery is removed you need to think about how this will work for all steps. This can be easily accomplished with a bit of planning. Often the program will just proceed through several steps and reach back at the start again. It is possible however that the program could get stalled on a step if you don't consider all possibilities. If the PCT does get stalled on a step during program execution you can always power it on/off (hardware restart) to restart the PCT.

The PCT programming functionality is very powerful. With a bit of thought, planning and testing you will be able to write a routine to accomplish your objectives with a routine that is very robust, easy to use and reliable. Don't be afraid to experiment, test, debug and revise until you are happy with the operation. Of course you must watch your PCT very carefully during program development to avoid dangerous situations of battery over charge, or over discharge. The battery can be damaged, catch fire or even explode if not charged properly. The best way to avoid this is to use a voltmeter and current meter to carefully monitor the voltage and current during program development. Also don't forget to use the monitor feature of the PCTWin to monitor how the PCT is operating.

Only when you have **fully tested** your PCT program should you use it, or make it available for use by others. Also each time you reload the PCT you should monitor the process to ensure that it has been loaded correctly and is working as expected.

### **Mandatory Requirements**

Companies, personnel and individuals who run the programming software PCTWin must agree to label the PCT module correctly and accurately with sufficient detail so that people using the PCT will be sufficiently notified as to the specific application and use for the programmed PCT. This label should include battery type, battery voltage and any specific instructions or warnings that the user should be made aware of.

Furthermore, companies, personnel and individuals who run the programming software PCTWin must agree to sufficiently test the PCT after it has been programmed to ensure that the programmed PCT operates as intended and

correctly for the intended application. This includes running each and every PCT with the application program prior to delivery to a user. This run test should include monitoring of voltage, current and all termination parameters for all steps of the program. The voltage and current should be monitored and measured using suitable current and voltage meters.

### **Miscellaneous Notes**

Do not use quotation ( " ) marks in the program memo. It will corrupt the data field and result in lost text information.

### **Data Preservation**

Note that the PCT routes automatically to step 1 when the PCT is power up. However the previous session's data does not get cleared until the PCT routes to the reset step declared in the Program Options. This give you the opportunity to use the PCT to test a battery. Then remove the battery, turn off the PCT and preserve the session details and data to be viewed at a different time or location. For example you could test a battery at one location, then return to your office or shop to view & print the console information. This allows you to use the PCT as a battery tester in a remote location without requiring a computer. To take advantage of this feature you should write the PCT program to not reset until a battery is connected. This way you can turn on the PCT, view the console details of the previous session on your computer. When you are ready to start a new session simply connect the battery. Ensure the PCT is programmed to detect the battery voltage and route to the Reset Step thereby clearing all the data from the previous session and then the program can proceed with the new session.

### **Consider possible power failures**

If the power fails during a session the PCT will automatically shut down all charge and discharge current. This is a fail safe mechanism designed to protect the battery from any damage due to over charge or over discharge. When the power is restored the PCT will automatically start at step 1 again. If power failures are a concern to you, then you can write the PCT program to perform as required during this situation. For example you could just have the session restart from the beginning. You could also have the PCT halt further action until you intervene. You can also preserve the console details until you get a chance to view the session up until the time of the power outage. The following tips can help you set this up to meet your needs:

- Have step 1 only proceed if the voltage is zero (i.e.  $V < 1\text{Volt}$ ). This will have the effect of preventing a new session from starting until the battery is removed. You can view the console during this step.
- Use step 2 the detect if a battery is connected. So now you will have to remove the battery (step1) and reconnect (step 2) to begin a new session.
- Use step 3 to reset the previous session's data so the new session can now begin.

### **PCT Resources**

The PCT has many resources. These resources are available for custom programming. Any or all of these resources can be employed to create a unique charge or test routine.

- Constant current charging
- Voltage regulated charging
- Constant current discharging
- Session timer
- Program step timer
- mAH measurement for both charging and discharging
- Current measurement
- Voltage measurement
- Temperature measurement (optional)
- Programmable built in beeper for audio indication
- Yellow LED programmable for various flash rates
- Green LED programmable for various flash rates
- -dv measurement for charge termination
- Minimum current measurement for charge termination.
- ~~Programmable pulse current capabilities~~ Feature not available at this time
- Internal resistance measurement
- dT/dT measurement (with optional temperature probe)
- Maximum voltage measurement
- Minimum voltage measurement
- Maximum temperature measurement (with optional temperature probe)
- Programmable termination statements for each program step

- Programmable conditional statements for alternative program routing
- Programmable routing and looping based on program statements
- 4 programmable counters used to control looping or alternate program routing
- Ability to set up conditional routing based on measured parameters

## **Application Ideas**

You can take advantage of the various PCT resources in order to use the PCT in innovative ways. The following examples give you some ideas. You will undoubtedly come up with other ideas of your own. The PCT has a unique design that allows you to program it in an unlimited number of ways for all sorts of applications.

### Decision Making:

For example you can use the PCT to make decisions based on measured parameters. A good example of this would be to create a single program to charge 6V or 12V SLA batteries. The PCT can detect the voltage and proceed to the step that will apply the correct charge parameters for the detected battery voltage. The green LED could illuminate for a 12V SLA and Yellow LED for a 6V SLA

Power Supply: The PCT can be used as a power supply. It can be set up as either a current supply or a current limited voltage supply. It is a very simple program that would provide the specified voltage / current at the output terminals.

Multi-staged Charging: The PCT can be programmed to charge in various stages. This could be a high current initial charge followed by another stage with lower current. Any number of stages can be programmed and rest periods can be inserted anywhere as required.

Function Generator: ~~With the pulse capabilities the PCT can provide current or voltage pulses.~~ Not available yet.

Battery Monitor with Alarm: The PCT can be programmed to monitor a battery voltage. If the voltage gets out of range the PCT can sound the built in alarm or flash a corresponding LED.

Load Module: With discharge capabilities the PCT can be programmed to be a constant current load module. This could also include options to interrupt the load if certain voltage, current or time parameters are reached.

Temperature Alarm: With the optional temperature input and temperature probe the PCT can be programmed to monitor temperature. If the temperature gets out of range the PCT can sound an alarm or flash a corresponding LED. This could be battery temperature, room temperature or any other temperature.

Float Charger: Hold an SLA battery to a float charge.

Field charger with 12V Input: The PCT1020 can be powered by a 12VDC source instead of the external 24VDC supply. With this feature you can power your PCT with a 12V car battery. This is great for turning your PCT into a charger you can use on the go. The maximum output charge voltage will be about 8V when powering the PCT from a 12VDC supply.

Trickle Charger: Great for maintaining a battery in a ready state of charge

Cycler: Program any number of cycles

Conditioner: Exercise a battery

NiCd Refresher : Discharge to a lower voltage per cell to equalize the cells

Timer The built in timer can be used to provide timed charges or even to just sound a beeper after an elapsed time.

Battery Life Cycle testing: The PCT can be programmed to simulate a real world application that a battery may experience. The PCT can be used to see how a battery will respond using this simulation. The simulation can include discharging and charging ~~as well as using pulse currents~~ to simulate changing loads. The LED's and beeper indicators can be programmed to let the user know when certain conditions are met. This can be an invaluable tool for a designer trying to evaluate various batteries and battery requirements.

---

Your imagination is the limit. Think of the PCT as a battery charging & testing module with a built in operating system. This, together with the PCTWin software application, allows you to write your own battery management programs - bp3 files. This puts you in control to build the ideal module for your unique application. Once you have your bp3 you can load or modify it as required. Finally a battery testing system that is not limited to what someone else designed. You can make it run as **you** need it.

---