

CPC – Fast-Tasking discussions

Rev: June 16, 2013 DMason

This document describes the effective use of Fast-Tasking in CPC.

Table of Contents

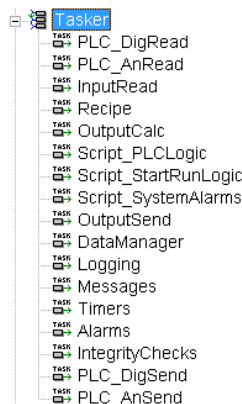
CPC – Fast-Tasking discussions.....	1
Table of Contents	1
What's a tasker?.....	1
The “main tasker” defined	1
Performance of taskers	2
What scan rate is fast enough for good control?	3
What scan rate is needed for time-proportioned outputs?	4
What if the main tasker is not fast enough?	4
What tasks should a Fast Tasker perform?	4
Do I have to have a separate FastInputs and FastOutputs object?.....	5
Should I have separate PLC transfer objects for specific fast inputs and outputs?	5
Multiple task objects under a fast tasker	5
Using a script to consolidate task methods.....	6
How to configure time-proportioned fast tasking.....	7

What's a tasker?

A Tasker is an object that functions as an internal sequencer. The Tasker groups multiple Task objects together as children and performs sequential processing of each task in the list. Taskers are regulated by the CPC Server, which attempts to give each tasker in the objects collection successive processing thread usage.

The “main tasker” defined

Most CPC implementations utilize a configured main tasker to perform the standard control and logic functions of the specific equipment being controlled.



Depending on the size of equipment, the number of inputs and outputs, and the mode of communication to the PLC or I/O hardware, the main tasker can perform a complete scan of it's children task list in under ½ second. On larger systems, this scan rate can grow to 1-2 seconds.

Performance of taskers

Each tasker includes a number of properties and settings to regulate the scan rate and performance.

Name	Tasker
Enabled	True
Count	3
DelayTime	
TotalTime	0.255144997499883
Performance	0.255123765906319
GuaranteedTimeSlice	
MinTaskFrequency	
AllTasksInSlice	

The Performance property indicates the amount of time required to scan through all tasks. The TotalTime is the performance plus any delay time configured.

An unregulated tasker is one in which the properties DelayTime, GuaranteedTimeSlice, and MinTaskFrequency are blank.

Speeding up a tasker

Although it's rarely required, you can speed up a tasker by configuring the GuaranteedTimeSlice property. This property will instruct the tasker to keep and maintain the server's processing thread for a defined amount of time before releasing the thread back to the server. Normally, the tasker returns the processing thread to the server after each task or subtask is completed.

The use of the guaranteed time slice will greatly improve processing performance of a single tasker, but it will adversely affect the speed of other taskers and can also cause a sluggishness of client activities (ie. screen updates, keyboard response, etc.) The example below guarantees that the tasker will have 20msec (.02 sec) before returning the thread. The result is a scan rate improvement from .255 secs to .166 secs.

Name	Tasker
Enabled	True
Count	3
DelayTime	
TotalTime	0.166247944813222
Performance	0.166237608529627
GuaranteedTimeSi	.02
MinTaskFrequency	
AllTasksInSlice	

Slowing down a tasker

You can slow down a tasker by setting the MinTaskFrequency property. This setting will instruct the tasker to regulate the number of tasks and subtasks it executes in each second. By slowing down a tasker, the performance time will increase and more server processing power will be used on other unregulated taskers.

Name	Tasker
Enabled	True
Count	3
DelayTime	
TotalTime	0.934134087059647
Performance	0.928230822784826
GuaranteedTimeSlice	
MinTaskFrequency	200
AllTasksInSlice	

Delaying a Tasker

You can use the Delay property to delay the repeating of a tasker. When a delay is entered, once a tasker completes its task list, the CPC server will not restart the tasker until the delay time has occurred. Note that the delay time does not affect the performance of the tasker while operating, but only delays the repeat.

Name	Tasker
Enabled	True
Count	3
DelayTime	.75s
TotalTime	1.00833038846031
Performance	0.257366229780018
GuaranteedTimeSlice	
MinTaskFrequency	
AllTasksInSlice	

What scan rate is fast enough for good control?

In a normal equipment application such as an autoclave or oven, the scan rate does not have to be very quick. A scan rate of 1-2 seconds is probably sufficient to control a process, especially if the equipment has proportional operating valves.

If an the equipment includes on/off or deviation control of specific sub-systems (ie. Pressure, Vacuum, etc.) or it has time-proportioned outputs, then a quicker scan rate might be required. For On/off control applications, you can judge how fast a scan rate is required by evaluating the rate of change of the process and the tolerance of the control you're trying to meet. For example, if a process changes at 20psig/min maximum and you'd like to control to +/- 1psig, then you'll need a minimum of: $60 \text{ sec/min} / 20 \text{ psig/min} \times 1 \text{ psig} = 3 \text{ second}$ scan rate to open and then close an on-off valve. Remember that the scan rate includes the reading of the sensor, the logic, and the output to the valve.

On some applications which utilize vacuum receivers, the vacuum level might change 10 "Hg in 5 seconds, which would be a rate of .5"Hg per second. In this case, if you want to control to +/- 1"Hg, you'll have to have a scan rate of at least 1 second to accomplish good control.

What scan rate is needed for time-proportioned outputs?

For time-proportioned outputs, the required resolution of time proportioning determines the scan rate requirements. When time-proportioning a discrete output, CPC divides the output % by the configured time-proportioning interval to determine the ON-time of the output. So, for example, if the output is 18% and the time-proportion interval is 5 seconds, then CPC would attempt to turn the discrete output on for $.18 \times 5 = .9$ seconds and off for $5 - .9 = 4.1$ seconds.

Depending on the scan rate of the tasker, CPC might not be able to hit the .9 seconds exactly. The reason for this is that the scan rate of a tasker determines how often CPC actually writes to the I/O or PLC output. So, if the scan rate is .55 seconds, then it can only turn ON or OFF an output every .55 seconds. The result of having a slower tasker is that the resolution of setting the output will be at sub-intervals of .55 seconds. That is, it will set the output at .55, and 1.10, and 1.65, and 1.20, and 1.75, etc. So, in the example above, 18% output might result in either .55 seconds ON or 1.10 seconds ON, both of which are inaccurate outputs for 18%.

So, in general, we'd like time-proportioning have a resolution of 1-2%, especially when the time-proportioning is used for heater control. That means that the scan rate should be 1/100 of the time-proportioning interval. You can achieve this by either increasing the interval setting or by reducing the scan rate. Since you should usually limit the interval to no more than 10 seconds, you should try to get a scan rate of .1 second or quicker.

What if the main tasker is not fast enough?

If the main tasker is not scanning fast enough to perform good repeatable control of fast-acting or time-proportioning sub-systems, then you'll need to implement an independent faster tasker. A "fast tasker" is a tasker that performs a limited number of tasks, but does them very quickly and at a fast scan-rate.

What tasks should a Fast Tasker perform?

If a fast tasker is used for a time-proportioning system, then the tasker should perform the following tasks:

- Perform a Send method of the time-proportioned outputs. This will set the On/Off value based on the current interval and then send those values to the internal table used for PLC transfers.
- Transfer the internal table values to the PLC.

If a fast tasker is used for on/off control of fast process sub-systems (ie. vacuum receiver), then the tasker should perform the following tasks:

- Transfer fast analog inputs from the PLC to the internal CPC tables.
- Perform a Read method of the specific inputs in CPC.
- Perform a Calc method of the specific outputs.
- Perform scripting to validate and interlock control operation for the fast tasked outputs.
- Perform a Send method of the specific outputs. This will send those output values to the internal table used for PLC transfers.
- Transfer the internal table values to the PLC.

It's important to realize that the Send method does not really send to the PLC. On CompactLogix systems, the Send and Read methods merely sends to or reads from internal CPC tables. These tables are not written to the PLC hardware until a Transfer method is executed.

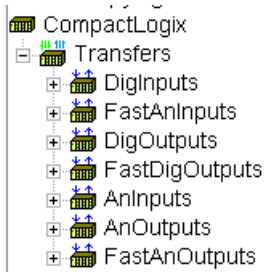
Do I have to have a separate FastInputs and FastOutputs object?

No, this is not required, however if you don't have a separate object then you'll want to make sure each fast output you use is configured to work only for the specific fast tasker. Each output has a property called "TaskerForSend" and "TaskerForCalc" which can be used to restrict the output to work from a specific tasker only. If you don't identify a specific tasker, then the fast tasker and the main tasker will calculate and send the outputs which is not what you want. Double taskers operating on the same output object causes problems, especially if there is fast scripting that overrides operation.

Should I have separate PLC transfer objects for specific fast inputs and outputs?

Yes, this is highly recommended. Don't use the same DigInputs, DigOutputs, AnInputs, and AnOutputs table for fast transfers. There is no use in fast-transferring dozens of inputs and output when all you really need is one or two.

For this reason, it's best to setup the PLC RsLogix program to have a fast inputs and fast outputs table when using fast analog or discrete values. For time-proportioned outputs, I would write a single integer register and then use the specific bit (ie. Bit0) in the PLC to turn On and Off the actual output.



Multiple task objects under a fast tasker

There are two ways of approaching a fast tasker. You can put individual task objects below the tasker, or you can assemble all methods into a script and run the script.

If you utilize multiple task objects below your fast tasker, it's important that you optimize the performance of the multiple tasks by using the properties "ConnectToNext" and "AllSubMethodsInSlice"

Name	FastOutputs
Enabled	True
Method	.Autoclave.FastOutputs\Do_Send
AfterTaskComplete	
ConnectToNext	True
AllSubMethodsInSlice	True
Performance	01.650972
StartTime	0

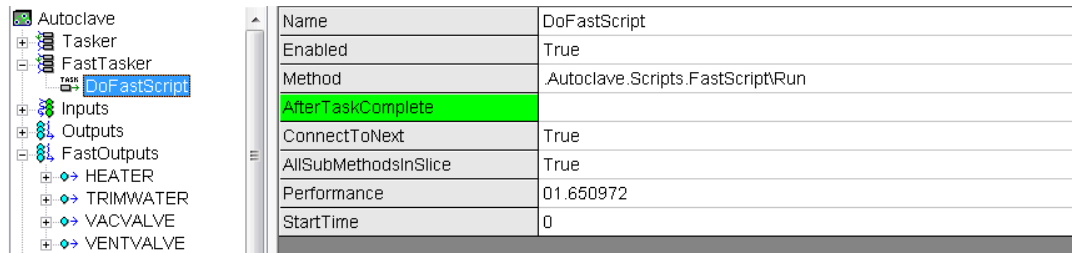
ConnectToNext instructs the task object to not relinquish its processing thread on task completion, but rather give the thread to the next task in the list. This effectively links all tasks into one processing thread slice and speeds the overall performance.

AllSubMethodsInSlice instructs the task object to guarantee that all underlying sub-methods are performed all at once without returning the processing thread. This property only affects methods of Inputs, Outputs, and other similar objects which automatically return the processing thread after each child operation. The affect is to lump all sub-methods into one processing thread slice, improving overall performance.

Using a script to consolidate task methods

Rather than have multiple task objects under the fast tasker, you can also consolidate all actions into a single script and then run the script in the fast tasker.

Here's a single task pointing to the Run method of a FastScript object.



The screenshot shows a tree view on the left with 'Autoclave' expanded, showing 'Tasker', 'FastTasker', and 'DoFastScript'. The 'DoFastScript' object is selected. On the right, a table displays the configuration for 'DoFastScript'.

Name	DoFastScript
Enabled	True
Method	.Autoclave.Scripts.FastScript.Run
AfterTaskComplete	
ConnectToNext	True
AllSubMethodsInSlice	True
Performance	01.650972
StartTime	0

Note: The fast outputs include time-proportion and fast control outputs.

Here's the script which performs transfers, reads, logic, sends, and transfers in one time slice. This script will execute as fast as possible. Notice that the CompactLogix includes fast analog inputs and fast analog outputs transfer tables. These tables only transfer one or two real/single values.

```

CPC Script Editor
Sub FastScript ()

    Alias VacCntrl = .Autoclave.Outputs.VACCNTRL
    Alias VacValve = .Autoclave.FastOutputs.VACVALVE\OutputValue
    Alias VentValve = .Autoclave.FastOutputs.VENTVALVE\OutputValue

    ' ** This logic performs fast control of proportional vacuum receiver
    .CompactLogix.Transfers.FastAnInputs\Transfer      ' ** get Sys Vacuum PLC input and transfer it to internal table
    .Autoclave.Inputs.SVAC\Do_Read                    ' ** transfer from internal table to Input object
    .Autoclave.Outputs.VACCNTRL\Do_Calc                ' ** calculate the PID output

    ' ** check outer extremes of setpoint
    If VacCntrl\MainSetpoint = 0 Then
        VacCntrl\OutputValue = 100
    ElseIf VacCntrl\MainSetpoint <= -30 Then
        VacCntrl\OutputValue = 0
    EndIf

    ' ** set valve outputs
    VentValve = (VacCntrl\OutputValue - 50) * 2
    VacValve = (VacCntrl\OutputValue - 50) * -2

    VentValve = Limit (VentValve, 0, 100)
    VacValve = Limit (VacValve, 0, 100)

    ' ** send valve outputs to the internal table
    .Autoclave.FastOutputs.VACVALVE\Do_Send
    .Autoclave.FastOutputs.VENTVALVE\Do_Send

    ' ** transfer internal table to PLC outputs
    .CompactLogix.Transfers.FastAnOutputs\Transfer

EndSub

```

How to configure time-proportioned fast tasking

You'll see below that we have a FastTasker with two tasks under it. We also have a FastOutputs object with two time-proportioned children outputs (HEATER and TRIMWATER).

Autoclave	Name	FastTasker
Tasker	Enabled	False
FastTasker	Count	0
DoFastOutputs	DelayTime	
DoFastTransfer	TotalTime	1.64309821487404
Inputs	Performance	1.65102575905621
Outputs	GuaranteedTimeSlice	
FastOutputs	MinTaskFrequency	100
HEATER	AllTasksInSlice	
TRIMWATER		
RecipeProcessor		
Internalscrubbing		

The first Task object (see below) will perform the Do_Send of the FastOutputs object. The Task object is specifically configured to complete the Do_Send of all children in one processors thread slice. The property "AllSubMethodsInSlice" force the task to complete all sub-methods before relinquishing the processor thread. Also, the "ConnectToNext" property instructs the task to move the processing thread to the next sibling task without returning it to the parent tasker. These two properties guarantee that all methods and tasks will be finished in one processor thread slice.

Note that the Do_Send method does not send the values to the PLC, but only to an internal table within CPC.

Name	FastOutputs
Enabled	True
Method	.Autoclave.FastOutputs\Do_Send
AfterTaskComplete	
ConnectToNext	True
AllSubMethodsInSlice	True
Performance	01.650972
StartTime	0

The second task performs a transfer of the FastDigOutputs from the internal CPC table to the CompactLogix PLC. Here's the task. Note that we don't have to use the ConnectToNext because this task is the last task in the tasker list. Also, there is no need to do AllSubMethodsInSlice, because the PLC_Transfer object doesn't have sub-methods.

Name	DoFastTransfer
Enabled	True
Method	.CompactLogix.Transfers.FastDigOutputs\Transfer
AfterTaskComplete	
ConnectToNext	False
AllSubMethodsInSlice	False
Performance	01.650972
StartTime	0

Lastly, here's what we would have in the PLC_Transfers. We've added a "FastDigOutputs" transfer that transfers only 1 integer word to the PLC. This word would include the discrete output for the HEATER and TRIMWATER valve.

