Development of Improved Thermal Analysis Capabilities at the NASA Goddard Space Flight Center

Hume L. Peabody[1] and Eric Yee[2]
NASA-GSFC, Greenbelt, MD 20771

Goddard Space Flight Center (GSFC) has been developing a framework of additional analysis capabilities to aid in the verification, development, and execution of thermal models using the OpenTD Application Programming Interface (API). This paper provides a brief overview of the data structures, properties, methods, and relationships between the objects accessible through the current API and describes some of the algorithms necessary to implement the desired functions at GSFC. Some example code snippets are also provided to aid potential users in the development of their own utilities. Following the overview are descriptions and algorithm methodologies of the new capabilities added to the GSFC framework, including: a new PI heater/controller approach for improved steady state predictions, selective copying of symbol over-rides from one source CaseSet to destination CaseSet(s), comparison of submodel object counts between a source and destination model to verify model integration, comparison of thermo-optical and thermo-physical properties between models, and improved display of extracted thermo-optical and thermo-physical properties for documentation.

Nomenclature

API	= *Application Programming Interface*	*NASA*	= *National Aeronautics and Space Administration*
GMM	= *Geometric Math Model*	*SINDA*	= *Systems Integrated Numerical Difference Analyzer*
GSFC	= *Goddard Space Flight Center*	*PID*	= *Proportional-Integral-Derivative*
GUI	= *Graphical User Interface*	*TMM*	= *Thermal Math Model*
MLI	= *Multi Layer Insulation*		

I. Introduction

Thermal Desktop® is an analysis tool commonly used by NASA-GSFC for the thermal modeling of spacecraft and instruments. It utilizes the AutoCAD program as the front-end Graphical User Interface (GUI) to allow analysts to construct geometric math models (GMM, which are used to compute radiative exchange factors and radiative heatloads from celestial sources) as well as generating a network thermal math model (TMM, which is solved to predict temperatures). Thermal Desktop has recently added an Application Programming Interface (API) beginning with version 6.0 and has extended the capabilities of the API with subsequent releases. Since the inclusion of the API, GSFC has utilized the capabilities offered to develop its own framework of thermal analysis utilities[1,2] to interface with model data and to automate repetitive tasks.

With the latest OpenTDv62 API, users now have full access to model data for all surface types, including finite elements, which were not available in previous versions. Leveraging these new additions, GSFC has developed additional utilities and capabilities to add to its existing framework. These new features include: a Proportional-Integral (PI) controller algorithm for steady state SINDA solutions and utilities to aid in the integration and checkout of integrated models. Capabilities of this new code consist of: the ability to compare the submodel level object counts/types across two models, copying selected symbols from a source CaseSet to destination CaseSets, improved comparison of predictions between model outputs, and improved reporting of thermo-optical and thermophysical properties for model documentation. This paper outlines some of the basic usage of the API and how the data can be accessed and describes the GSFC developed utilities and methodologies used to develop them in further detail. It concludes with an updated compilation of the capabilities of the framework and the current GSFC GUI to access the utilities.

[1] Staff Thermal Engineer, Mail Stop 545, Goddard Space Flight Center, Greenbelt MD 20771.
[2] Thermal Engineer, Mail Stop 545, Goddard Space Flight Center, Greenbelt MD 20771.

II. Basic API Usage Overview

In order to describe the methodologies employed to develop the utilities listed in the introduction, a brief overview of the methods and data structures utilized by the API is provided. While code snippets could be provided, they are deemed beyond the scope of this paper. The intent of this section is to provide just enough information for an interested user to get started. The API itself is accessed by including a reference to the OpenTDv62.dll in a user's .NET compatible project. To establish a connection with Thermal Desktop, a variable of `ThermalDesktop` type is created (e.g. `Dim TD as New OpenTDv62.ThermalDesktop`). Once created, the drawing file to be accessed is specified by setting the `TD.ConnectConfig.DwgPathname` property. Following this with a call to the `TD.Connect()` method establishes a link with either an existing AutoCAD application that is already open with the specified drawing file or creates a new instance in which the specified drawing file is subsequently opened. At this point, all of the exposed data is now available for further investigation or manipulation. In general, two types of tasks could be developed: those that alter the execution flow and those that query and/or manipulate data.

Most object types can be retrieved as a List(Of *ObjectType*) by calling the appropriate method [e.g. `TD.GetRectangles()`]. Alternately, a single object may also be retrieved knowing its handle, which is the unique identifier for each object in the AutoCAD database and is generally stored as a 6-character hexadecimal values [e.g. `TD.GetNode(myHandle)`]. Each object type includes `.AttachedZZZHandles` lists, where ZZZ varies by object type (e.g. Conics, Node, HeatLoad, Object, etc) which provides the linking hierarchy between objects (e.g. a Contactor that references a Surface, a HeatLoad assigned to a Node). For example, a node object type includes the `.AttachedConicsHandles` method, which retrieves a list of all the handles for surfaces related to that node, and the `.AttachedObjectHandles` method, which retrieves a list of handles for all the objects (e.g. HeatLoads, Finite Elements, Conductors, etc.) related to it. Using the `.GetEntityTypes` method allows each of these handles to be dereferenced to their object type and the specific instance of each object retrieved for further evaluation. Understanding the relationships between objects through their related handles was crucial to the development of the object count utility as well as any future development across dissimilar object types.

While the function names listed above are specific to the OpenTDv62.dll, the software vendor has made efforts to not "break" the functions with each subsequent release of OpenTD. In fact, future releases would be in their own file, preventing conflicts if the original library continues to be used with the developed code. Only when a new library is introduced, perhaps to take advantage of newer features, does the risk of conflicts arise. In fact, during the development of these tools, upgrading from v6.1 to v6.2 did break some functions that changed between the two libraries, but the updates to fix the code were very minor. That said, it is in the software vendor's own interest to minimize any disruptions to the existing functions in the library to keep end users satisfied and wanting to continue to use the capability while also reducing the amount of support they would need to provide.

Some of the functions developed by GSFC do not directly interact with the objects in the model, but rather alter the model execution portion of the analysis process to inject code for specific purposes. In this case, the API is used modify a case set, execute it as needed, and process the files generated in response to add custom code to perform additional tasks during model execution. This was first used by GSFC to automate the inclusion of Heater, Heat Load, and PID Controller logic between the generation of the CondCap file and the execution of the input file[2]. This approach was also improved to implement the Radk Filtering options developed previously[3]. Most recently, this capability has been expanded to add custom logic for the emulation of PID controllers in steady state solutions to achieve the setpoint, which is described in the following section.

III. Steady State PID Emulation

PID controlled heaters are difficult to model in steady state solutions as there is no time value on which the integrator term can operate to achieve the setpoint. Therefore, the proportional term is the only term that can provide meaningful contributions to the control variable. Some PID algorithms might try to utilize the iteration count as a pseudo-time substitute, but this becomes difficult with simultaneous solutions which advance the entire solution each iteration and no mass to dampen the changes between iterations. For this new approach, a predictor-corrector method was employed to get close to the stable solution before an averaging window is used to generate the final predictor value. The corrector term is then employed based on the setpoint/sense point relationship to gradually increase or decrease the control variable. Each time the setpoint is crossed, the adjustment value is decreased until it reaches a minimal threshold, after which it is set to zero and the duty cycle remains fixed for the remainder of the solution.

The SINDA input deck is first processed to identify all calls to PID controllers and the setpoint, sense point, and control variable registers are identified. Further processing is performed in two additional passes through the input file to identify registers that may reference the control variable but are themselves assigned to the nodal heat

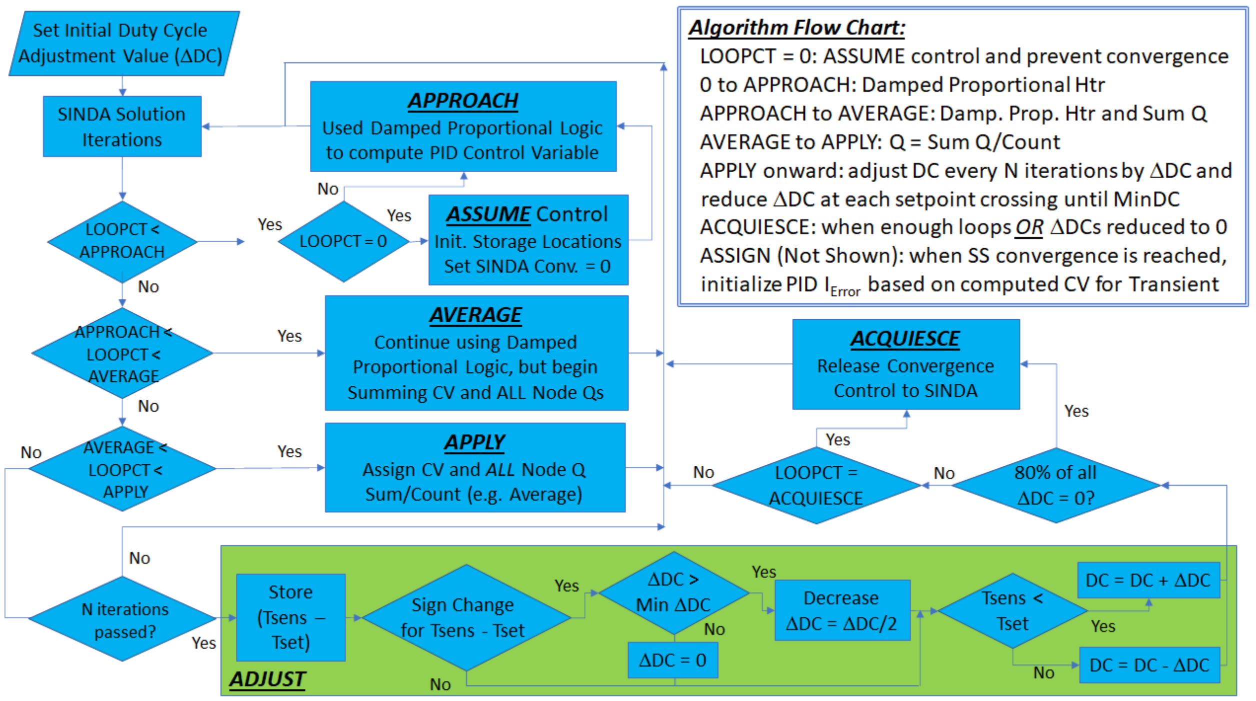

Set Initial Duty Cycle Adjustment Value (ΔDC)
SINDA Solution Iterations
APPROACH
Used Damped Proportional Logic to compute PID Control Variable
LOOPCT < APPROACH
Yes
No
LOOPCT = 0
Yes
No
ASSUME Control
Init. Storage Locations
Set SINDA Conv. = 0
APPROACH < LOOPCT < AVERAGE
AVERAGE
Continue using Damped Proportional Logic, but begin Summing CV and ALL Node Qs
AVERAGE < LOOPCT < APPLY
APPLY
Assign CV and ALL Node Q Sum/Count (e.g. Average)
Algorithm Flow Chart:
LOOPCT = 0: ASSUME control and prevent convergence
0 to APPROACH: Damped Proportional Htr
APPROACH to AVERAGE: Damp. Prop. Htr and Sum Q
AVERAGE to APPLY: Q = Sum Q/Count
APPLY onward: adjust DC every N iterations by ΔDC and reduce ΔDC at each setpoint crossing until MinDC
ACQUIESCE: when enough loops OR ΔDCs reduced to 0
ASSIGN (Not Shown): when SS convergence is reached, initialize PID I_Error based on computed CV for Transient
ACQUIESCE
Release Convergence Control to SINDA
LOOPCT = ACQUIESCE
80% of all ΔDC = 0?
N iterations passed?
Store (Tsens – Tset)
Sign Change for Tsens - Tset
ΔDC > Min ΔDC
ΔDC = 0
Decrease ΔDC = ΔDC/2
Tsens < Tset
DC = DC + ΔDC
DC = DC - ΔDC
ADJUST

Figure 1. Steady State PID Heater Emulation Flowchart: *The algorithm behavior varies with iteration. It first ASSUMES control to prevent premature convergence by SINDA. Next, it APPROACHes the required heater power using a damped, proportional approach. After some iterations, it begins the process of AVERAGE-ing these values, which are the APPLY-ed as a constant averaged value over the next iterations. After applying the average, it ADJUSTs the heater power to more closely match the setpoint, reducing the adjustment amount every time the*

application logic (e.g. PID_1_HeaterPwr = PID_1_DutyCycle * PID_1_AvailablePower). A final pass is made through the file to determine all the nodes to which heat is applied for each PID controller. With the knowledge of the control variable, sensing point, setpoint, and related nodes for each controller, the logic is then be written to the file to be included in the solution.

The complete algorithm (shown in Figure 1) is executed in SINDA and is broken into multiple sequential phases: ASSUME CONTROL, APPROACH, AVERAGE, APPLY, ADJUST, ACQUIESCE, and ASSIGN. The ASSUME CONTROL phase begins by storing all the current convergence criteria and recasting the convergence criteria to zero to prevent solution convergence prior to setpoint achievement. This portion also initializes arrays which store the relative node locations in SINDA and initializes the average value of the control variable to zero. The APPROACH phase utilizes the same logic as a steady state damped, proportional heater generated by Thermal Desktop, with the On/Off range defined as (setpoint) to (setpoint – 4). This phase is used to determine a duty cycle that is generally close to achieving the desired setpoint through the first X iterations but may result in oscillations about the setpoint. The next phase is the AVERAGE phase which computes the sum of the heat applied for ***each individual node*** in the model over the next Y iterations as well as the sum of each control variable. Dividing these sums by Y, results in the average power or average control variable value as the solution moves into the next phase. The APPLY phase assigns the average nodal heat load to each node and the average control variable for the next Z iterations in order to provide stable and constant values to the solution for better likelihood of convergence without the perturbations of PID controllers or other heaters or varying heat in a steady state solution.

The next ADJUST phase is the most crucial, as this is the phase where adjustments are made to the heat applied based on the relationship between the setpoint and the sense point. If the sense point is above the setpoint, then the duty cycle is reduced by a fixed delta beginning with a 4% change (e.g. a duty cycle of 0.33 changes to 0.29 if the sense point is warmer than the setpoint). The opposite is true if the sense point is below the setpoint and the duty cycle is increased. Constraints are applied to ensure than the duty cycle never exceeds 100% or falls below 0%. Earlier versions of this algorithm only adjusted the PID control variable, but other non-PID heaters in the model generally perturbed the solution enough to prevent convergence. Furthermore, previous analysis efforts constrained the nodal heat values to constant (averaged values) for the last part of the run to improve the likelihood of convergence, but without the corrector term, the controllers were unlikely to achieve their setpoints. The final algorithm combines the benefits of both approaches. The relationship between each PID controller and the nodes to which heat is applied must be established in order for this approach to function properly. The heat values at a nodal level are adjusted every N iterations and the data is tracked to see when the sensing point crosses the setpoint. At this crossing event, the amount of the adjustment is reduced by half (e.g. 4% to 2%, 2% to 1%, etc.) and the process continues. The effect is a damping of the overshoot or undershoot about the setpoint until 0.25% is reached, at which point the adjustment is set to zero and the prediction deemed adequate. Throughout the ADJUST phase, the percentage of circuits that have had their adjustment reduced to zero is tracked, and once 80% of the controllers are no longer adjusting, the last ACQUIESCE phase is executed. During this phase, the convergence criteria is set back to the original values, and the solution continues towards convergence with the few remaining controllers still adjusting if needed. Upon convergence, the final ASSIGN phase is executed which calls the PIDINIT function in SINDA with the current setpoint, sense point, and control variable, which initializes the accumulated error term for the start of a subsequent transient solution.

During the development of this technique, other options were explored and found to not perform as well as the final algorithm. This includes only applying this methodology to the control variable, which did not allow for the nodal heat averaging over all nodes and often did not converge. Additionally, the adjustment was originally envisioned as a multiplier, but this had a near negligible effect for very low duty cycles (e.g. a 1.04/0.96 multiplier on a 4% duty cycle would take many adjustments before any meaningful response could be seen.

The overall goal was a reduction in the run time for the Roman Space Telescope observatory model to achieve quasi-steady stability. While the number of iterations needed to reach steady state increased, the ability to start the transient from a closer condition resulted in less solution time needed to reach quasi-stability and an overall reduction in run time. Figure 2 shows the PID emulation for one of the controllers where each phase can be clearly seen. Figure 3 shows a comparison of the transient model performance for a select controller and resulted in a 33% reduction in overall run time.

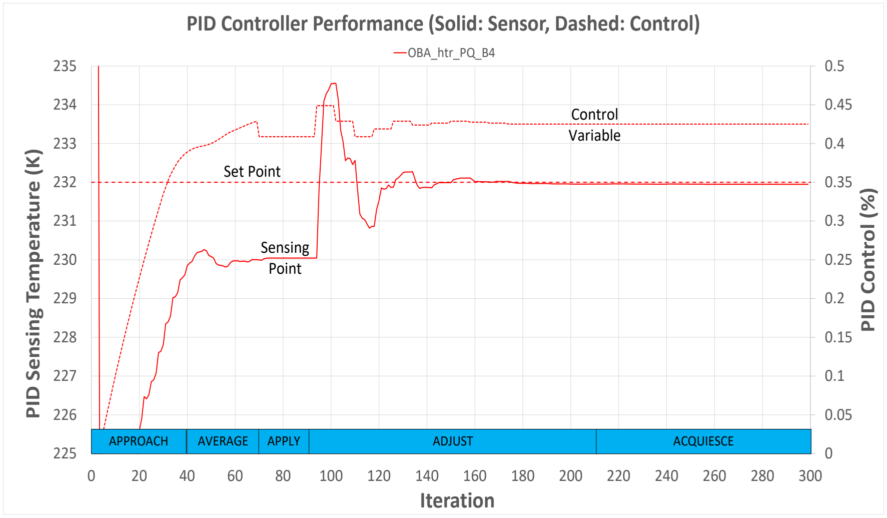


Figure 2. Performance of PID Heater Emulation during Steady State Solution: *APPROACH: 0-40, AVERAGE: 41-70, APPLY: 71-90, ADJUST: 91-210. As the entire solution progresses, full model convergence is achieved around iteration 300 and the sensing point is very close to the setpoint of 232 K*

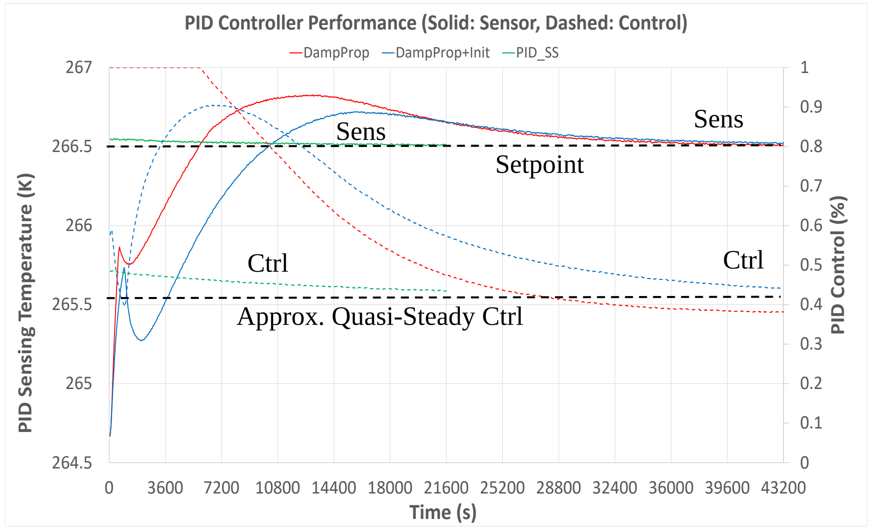


Figure 3. Performance of PID Heater with and without Steady State Emulation: *The red curve represents the previous method using a damped proportional approach for steady state. The blue initialized the PID accumulated error using the damped proportional values estimated in steady state. Lastly, the green used the Predictor/Corrector approach and initialized the accumulated error using the predicted steady state values and was run for half the transient time reaching quasi-stability considerably sooner. By the end of each run, the results are similar, although the new approach reaches that point much sooner.*

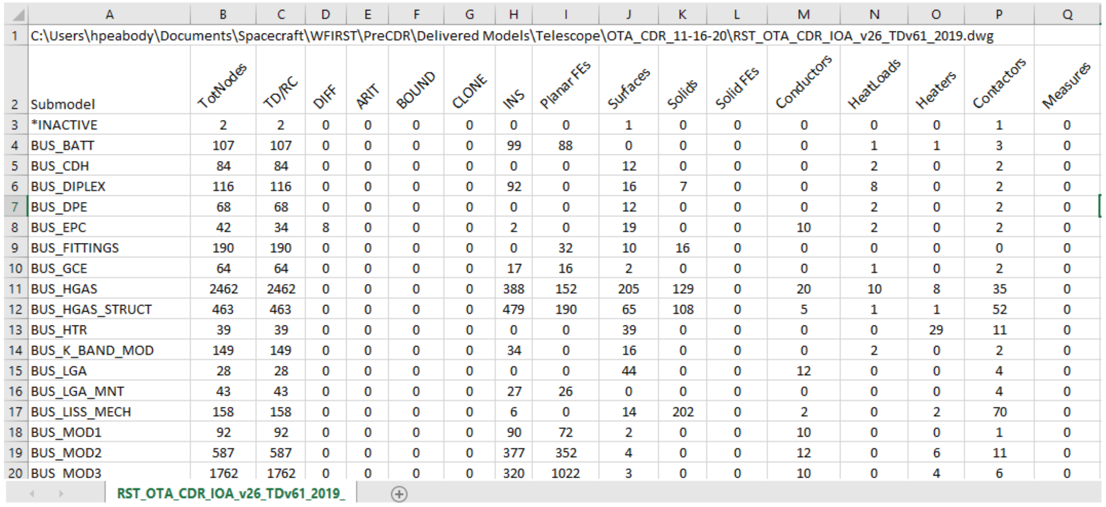
IV. Object Counts

A fairly common model check when integrating a source model into a destination model in Thermal Desktop is to compare the object counts between the two models, usually done at a submodel level. Any discrepancies may indicate differences that should be resolved before concluding the integration was successful. However, for model deliveries with many submodels, the practicality of comparing possibly more than 50 submodels becomes tedious at best. Therefore, the API was utilized to programmatically retrieve this information for each submodel from two models and output the results into a spreadsheet for much easier direct comparison.

The routine begins with evaluating all Domain Tag Sets and storing the information for later use. To optimize the data retrieval, lists of Submodels, Nodes, Conductors, Contactors, HeatLoads, and Heaters for the entire model are retrieved and stored in internal variables. Each submodel is processed and the subset of nodes in that submodel are retrieved using the `.Where` method of the `List(of Node)`. The type of the node (RC/TD, Diffusion, Arithmetic, etc) is readily identifiable through direct properties of a node. Surfaces and elements require iterating that the node's `.AttachedConicsHandles` and `.AttachedObjectHandles` lists.

As these lists are processed for their respective entity types, handles for Surfaces, Solids, Planar Finite Elements, and Solid Finite Elements are identified and stored separately. Upon processing all the nodes in a submodel, these four lists of handles are sorted and compressed to remove duplicate handles. The lists of Surfaces, Solids, and Finite Elements are then processed to identify any assigned MLI nodes, which are not included as unique objects in the List(of Node) previously retrieved. As each object type is identified, the object itself is retrieved, and then the properties defining the application of insulation are queried to identify MLI node numbers and store them in a master insulation node list. This master list is then used to report the number of INS nodes for each submodel.

Lastly, all Conductors, Contactors, HeatLoads, and Heaters are processed. For Conductors, the `.From` connection list is a single node and this handle can be compared to all node handles in the submodel subset list to determine if it matches, and therefore is related to the submodel being processed. The `.To` list for a Conductor may contain either nodes or surfaces. Determination of related nodes is the same as for the `.To` list approach, but for surface types, the comparison is made to the list of surfaces and planar finite elements determined from `.AttachedConicsHandles` and `.AttachedObjectHandles` methods earlier. In the event that a Domain Tag Set is specified, it is replaced with the handles to the objects contained in the Domain Tag Set definition processed earlier in the sequence. If a reference to the currently processed submodel is identified, then the current Conductor is identified as related to the current submodel and the Conductor count is incremented. Each Conductor is processed similarly, and the resulting count of all related entities is stored. A similar process is followed for HeatLoads, Heaters (which also includes the `.ApplyConnections` and `.SensorConnections`) and Contactors (which includes the `.From` and `.To` connections).Once all the data is processed, the results are compiled into a 2D matrix comprised of submodels and object counts for: Total Nodes, Thermal Desktop/RadCAD nodes, Diffusion Nodes, Arithmetic Nodes, Boundary Nodes, Clone Nodes, Insulation Nodes, Planar Finite Elements, Surfaces, Solids, Solid Finite Elements, Conductors, HeatLoads, Heaters, Contactors, and Measures. This matrix is then readily output as a Comma-Separated Values file for subsequent import into a spreadsheet, as shown below in Figure 4.



C:\Users\hpeabody\Documents\Spacecraft\WFIRST\PreCDR\Delivered Models\Telescope\OTA_CDR_11-16-20\RST_OTA_CDR_IOA_v26_TDv61_2019.dwg

Submodel	TotNodes	TD/RC	DIFF	ARIT	BOUND	CLONE	INS	Planar FEs	Surfaces	Solids	Solid FEs	Conductors	HeatLoads	Heaters	Contactors	Measures
*INACTIVE	2	2	0	0	0	0	0	0	1	0	0	0	0	0	1	0
BUS_BATT	107	107	0	0	0	0	99	88	0	0	0	0	1	1	3	0
BUS_CDH	84	84	0	0	0	0	0	0	12	0	0	0	2	0	2	0
BUS_DIPLEX	116	116	0	0	0	0	92	0	16	7	0	0	8	0	2	0
BUS_DPE	68	68	0	0	0	0	0	0	12	0	0	0	2	0	2	0
BUS_EPC	42	34	8	0	0	0	2	0	19	0	0	10	2	0	2	0
BUS_FITTINGS	190	190	0	0	0	0	0	32	10	16	0	0	0	0	0	0
BUS_GCE	64	64	0	0	0	0	17	16	2	0	0	0	1	0	2	0
BUS_HGAS	2462	2462	0	0	0	0	388	152	205	129	0	20	10	8	35	0
BUS_HGAS_STRUCT	463	463	0	0	0	0	479	190	65	108	0	5	1	1	52	0
BUS_HTR	39	39	0	0	0	0	0	0	39	0	0	0	0	29	11	0
BUS_K_BAND_MOD	149	149	0	0	0	0	34	0	16	0	0	0	2	0	2	0
BUS_LGA	28	28	0	0	0	0	0	0	44	0	0	12	0	0	4	0
BUS_LGA_MNT	43	43	0	0	0	0	27	26	0	0	0	0	0	0	4	0
BUS_LISS_MECH	158	158	0	0	0	0	6	0	14	202	0	2	0	2	70	0
BUS_MOD1	92	92	0	0	0	0	90	72	2	0	0	10	0	0	1	0
BUS_MOD2	587	587	0	0	0	0	377	352	4	0	0	12	0	6	11	0
BUS_MOD3	1762	1762	0	0	0	0	320	1022	3	0	0	10	0	4	6	0

RST_OTA_CDR_IOA_v26_TDv61_2019_

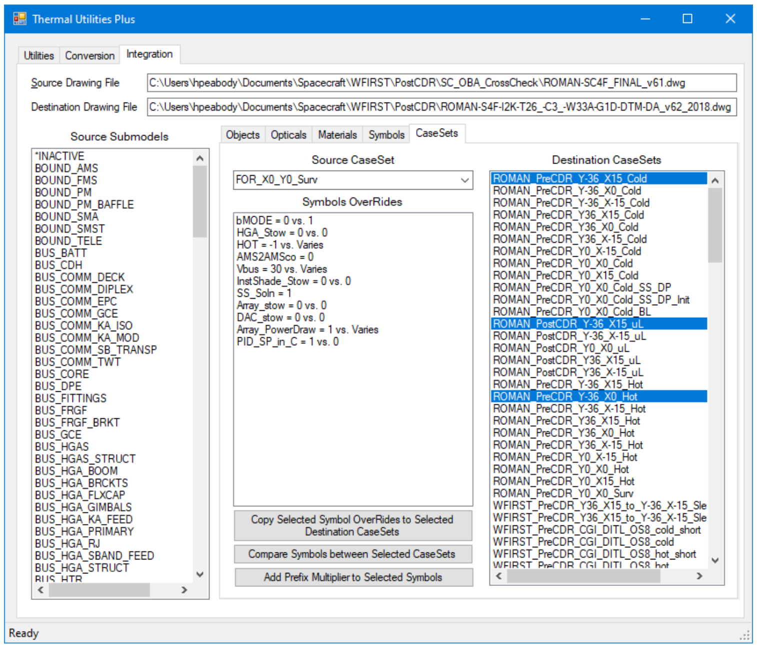
Figure 4. Sample of Object Count Output: *The number of each object type associated with a given submodels is listed for further evaluation or comparison between two or more models*

V. Copy Symbols between CaseSets

One of the more powerful features of Thermal Desktop is the ubiquitous use of symbols to control and configure a model. However, this can also lead to instances where the incorrect setting of a symbol can lead to model errors. Furthermore, incorporating numerous symbol overrides from a given CaseSet in a delivered model into a CaseSet in a destination model can be very time consuming and error prone. Needing to include these symbols in multiple destination CaseSets only compounds this challenge, although the ability to edit multiple case sets does alleviate this a bit. But when needing to integrate multiple subsystem models, each with their own sets of symbols and values, for multiple configuration cases (e.g. hot, cold, survival, stowed, etc), this can quickly be a time-consuming process to establish the correct full set of symbols in the destination model.

Using the API, an interface was developed which allows for the selection of a single CaseSet from the source model and then displays all the symbol overrides, along with their override values. A listing of all the CaseSets from a destination model is also provided, and as a user selects destination CaseSets, the values for each symbol override in those CaseSets are displayed alongside the source case set symbols. If the destination symbol's values are not identical across all selected CaseSets, then the value displayed is "Varies". Use of this interface has reduced the time necessary to construct the CaseSet definition in the destination file considerably by displaying the source values and allowing for a simple button click to transfer the symbol definitions to the destination CaseSets.

Another companion capability to copying symbols between CaseSets was also developed that allows for comparison of symbols between multiple case sets. However, this feature goes deeper than a direct symbol override comparison, evaluating symbols based on based on the direct override, as well as symbols that are dependent on the overridden symbol[1]. Figure 5 shows the graphical interface for copying symbols from one CaseSet to another, while Figure 6 shows the results of a comparison of symbols between multiple case sets.



Source Drawing File: C:\Users\hpeabody\Documents\Spacecraft\WFIRST\PostCDR\SC_OBA_CrossCheck\ROMAN-SC4F_FINAL_v61.dwg

Destination Drawing File: C:\Users\hpeabody\Documents\Spacecraft\WFIRST\PostCDR\ROMAN-S4F-I2K-T26_-C3_-W33A-G1D-DTM-DA_v62_2018.dwg

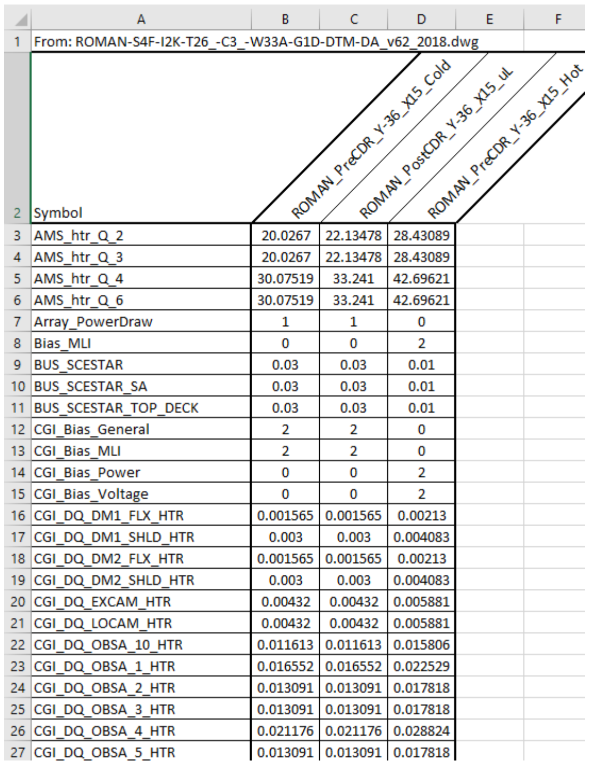
Source Submodels:
*INACTIVE
BOUND_AMS
BOUND_FMS
BOUND_PM
BOUND_PM_BAFFLE
BOUND_SMA
BOUND_SMST
BOUND_TELE
BUS_BATT
BUS_CDH
BUS_COMM_DECK
BUS_COMM_DIPLEX
BUS_COMM_EPC
BUS_COMM_GCE
BUS_COMM_KA_ISO
BUS_COMM_KA_MOD
BUS_COMM_SB_TRANSP
BUS_COMM_TWT
BUS_CORE
BUS_DPE
BUS_FITTINGS
BUS_FRGF
BUS_FRGF_BRKT
BUS_GCE
BUS_HGAS
BUS_HGAS_STRUCT
BUS_HGA_BOOM
BUS_HGA_BRCKTS
BUS_HGA_FLXCAP
BUS_HGA_GIMBALS
BUS_HGA_KA_FEED
BUS_HGA_PRIMARY
BUS_HGA_RJ
BUS_HGA_SBAND_FEED
BUS_HGA_STRUCT
BUS_HTR

Source CaseSet: FOR_X0_Y0_Surv

Symbols OverRides:
bMODE = 0 vs. 1
HGA_Stow = 0 vs. 0
HOT = -1 vs. Varies
AMS2AMSco = 0
Vbus = 30 vs. Varies
InstShade_Stow = 0 vs. 0
SS_Soln = 1
Array_stow = 0 vs. 0
DAC_stow = 0 vs. 0
Array_PowerDraw = 1 vs. Varies
PID_SP_in_C = 1 vs. 0

Destination CaseSets:
ROMAN_PreCDR_Y-36_X15_Cold
ROMAN_PreCDR_Y-36_X0_Cold
ROMAN_PreCDR_Y-36_X-15_Cold
ROMAN_PreCDR_Y36_X15_Cold
ROMAN_PreCDR_Y36_X0_Cold
ROMAN_PreCDR_Y36_X-15_Cold
ROMAN_PreCDR_Y0_X-15_Cold
ROMAN_PreCDR_Y0_X0_Cold
ROMAN_PreCDR_Y0_X15_Cold
ROMAN_PreCDR_Y0_X0_Cold_SS_DP
ROMAN_PreCDR_Y0_X0_Cold_SS_DP_Init
ROMAN_PreCDR_Y0_X0_Cold_BL
ROMAN_PostCDR_Y-36_X15_uL
ROMAN_PostCDR_Y-36_X-15_uL
ROMAN_PostCDR_Y0_X0_uL
ROMAN_PostCDR_Y36_X15_uL
ROMAN_PostCDR_Y36_X-15_uL
ROMAN_PreCDR_Y-36_X15_Hot
ROMAN_PreCDR_Y-36_X0_Hot
ROMAN_PreCDR_Y-36_X-15_Hot
ROMAN_PreCDR_Y36_X15_Hot
ROMAN_PreCDR_Y36_X0_Hot
ROMAN_PreCDR_Y36_X-15_Hot
ROMAN_PreCDR_Y0_X-15_Hot
ROMAN_PreCDR_Y0_X0_Hot
ROMAN_PreCDR_Y0_X15_Hot
ROMAN_PreCDR_Y0_X0_Surv
WFIRST_PreCDR_Y36_X15_to_Y-36_X-15_Sle
WFIRST_PreCDR_Y36_X15_to_Y-36_X-15_Sle
WFIRST_PreCDR_CGI_DITL_OS8_cold_short
WFIRST_PreCDR_CGI_DITL_OS8_cold
WFIRST_PreCDR_CGI_DITL_OS8_hot_short
WFIRST_PreCDR_CGI_DITL_OS8_hot

Figure 5. Graphical User Interface for Symbol Copy between CaseSets: *The GUI displays the symbol overrides values of the selected source as well as the corresponding values from the selected Destination CaseSets*



	A	B	C	D	E	F
1	From: ROMAN-S4F-I2K-T26_-C3_-W33A-G1D-DTM-DA_v62_2018.dwg					
2	Symbol	ROMAN_PreCDR_Y-36_X15_Cold	ROMAN_PostCDR_Y-36_X15_uL	ROMAN_PreCDR_Y-36_X15_Hot		
3	AMS_htr_Q_2	20.0267	22.13478	28.43089		
4	AMS_htr_Q_3	20.0267	22.13478	28.43089		
5	AMS_htr_Q_4	30.07519	33.241	42.69621		
6	AMS_htr_Q_6	30.07519	33.241	42.69621		
7	Array_PowerDraw	1	1	0		
8	Bias_MLI	0	0	2		
9	BUS_SCESTAR	0.03	0.03	0.01		
10	BUS_SCESTAR_SA	0.03	0.03	0.01		
11	BUS_SCESTAR_TOP_DECK	0.03	0.03	0.01		
12	CGI_Bias_General	2	2	0		
13	CGI_Bias_MLI	2	2	0		
14	CGI_Bias_Power	0	0	2		
15	CGI_Bias_Voltage	0	0	2		
16	CGI_DQ_DM1_FLX_HTR	0.001565	0.001565	0.00213		
17	CGI_DQ_DM1_SHLD_HTR	0.003	0.003	0.004083		
18	CGI_DQ_DM2_FLX_HTR	0.001565	0.001565	0.00213		
19	CGI_DQ_DM2_SHLD_HTR	0.003	0.003	0.004083		
20	CGI_DQ_EXCAM_HTR	0.00432	0.00432	0.005881		
21	CGI_DQ_LOCAM_HTR	0.00432	0.00432	0.005881		
22	CGI_DQ_OBSA_10_HTR	0.011613	0.011613	0.015806		
23	CGI_DQ_OBSA_1_HTR	0.016552	0.016552	0.022529		
24	CGI_DQ_OBSA_2_HTR	0.013091	0.013091	0.017818		
25	CGI_DQ_OBSA_3_HTR	0.013091	0.013091	0.017818		
26	CGI_DQ_OBSA_4_HTR	0.021176	0.021176	0.028824		
27	CGI_DQ_OBSA_5_HTR	0.013091	0.013091	0.017818		

Figure 6. Symbol Comparison Output: *The comparison of the three selected Destination CaseSets highlights the differences in symbol values, including the dependencies of the direct overrides.*

VI. Model Prediction Comparison

Beyond the comparison of object counts and symbols between models, a methodology was also sought to easily compare predictions between models. Previous efforts[1] included the capability to output Heater, Heat Load, and PID controller information at each timestep. This approach reads through the CondCap file generated by ThermalDesktop prior to the execution of SINDA. During this read, it identifies SINDA code associated with the application of HeatLoads, Heaters, and PID controllers and extracts the defining parameters (e.g. ,setpoint, heat dissipations, control variable, etc). These values and variables associated with them are then processed to generate specialized logic to be included during the model execution in SINDA, which in turn produces relevant data in the output file related to the HeatLoads, Heaters and PID controllers at every timestep during the solution. After model execution, the output file is then processed to extract the data and imports it into a template Microsoft Excel® workbook for further processing, shown in Figure 7.

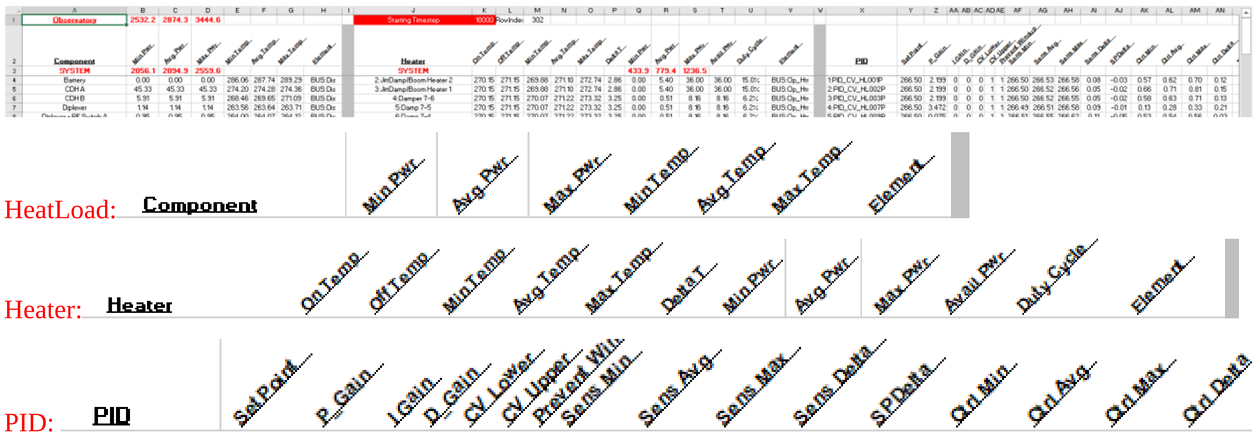
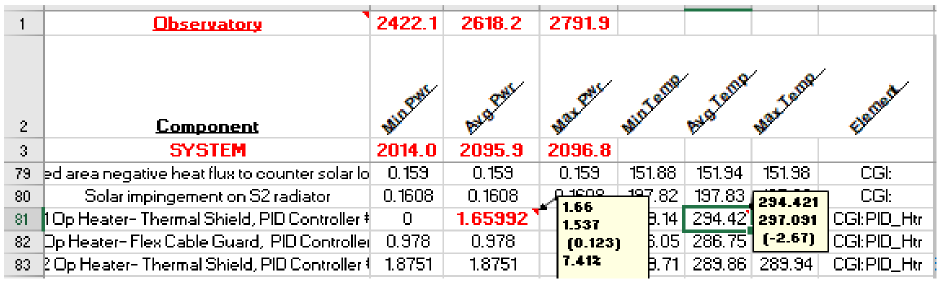


Figure 7. Heater, HeatLoad, PID Controller Summary Sheet: *The Summary sheet shows the critical temperature and power data for all HeatLoads, Heaters, and PID Controllers found in a model (Headers enlarged)*

A new comparison utility was recently added to identify significant differences between two model outputs based on the data on the Summary tab, shown in Figure 8. Two files are supplied to the routine and it evaluates the data, identifying temperature deviations of more than 2 K and any differences in power (Dissipation, Heater, or Control Variable). Power differences of more than 5% are identified by bold, red text; power differences between 2-5% are identified by bold orange text. All differences of consequence are captured as a comment to the worksheet cell, which includes: the base value, compare value, difference, and percent difference (if a power value). Figure 7 shows the Summary_Compare sheet with the comments identifying significant differences. At this time, the comparison is keyed off the name of the HeatLoad, Heater, or PID Controller. Therefore, comparisons are limited based on those names and to the same software, but the general intent is to be able to verify that the performance of a subsystem model is consistent when integrated to the next higher level of assembly, and the renaming of these object types should not be expected. Furthermore, these is no effective way to graphically display these values (e.g. contour plot on a 3D model) as many of the values represent a subset of all nodal values, but do represent critical locations where heat is applied or temperatures are directly impacted by the application of heat. In this sense, tabulation of the data and the differences is judged the best means to display this data.



1	Observatory	2422.1	2618.2	2791.9				
2	Component	Min Pwr	Avg Pwr	Max Pwr	Min Temp	Avg Temp	Max Temp	Element
3	SYSTEM	2014.0	2095.9	2096.8				
79	ed area negative heat flux to counter solar lo	0.159	0.159	0.159	151.88	151.94	151.98	CGI:
80	Solar impingement on S2 radiator	0.1608	0.1608	0.1608	197.82	197.83	[illegible]	CGI:
81	1 Op Heater- Thermal Shield, PID Controller #	0	1.65992	[illegible]	.14	294.42	[illegible]	CGI:PID_Htr
82	Op Heater- Flex Cable Guard, PID Controller	0.978	0.978	[illegible]	.05	286.75	[illegible]	CGI:PID_Htr
83	2 Op Heater- Thermal Shield, PID Controller #	1.8751	1.8751	[illegible]	.71	289.86	289.94	CGI:PID_Htr

1.66
1.537
(0.123)
7.41%

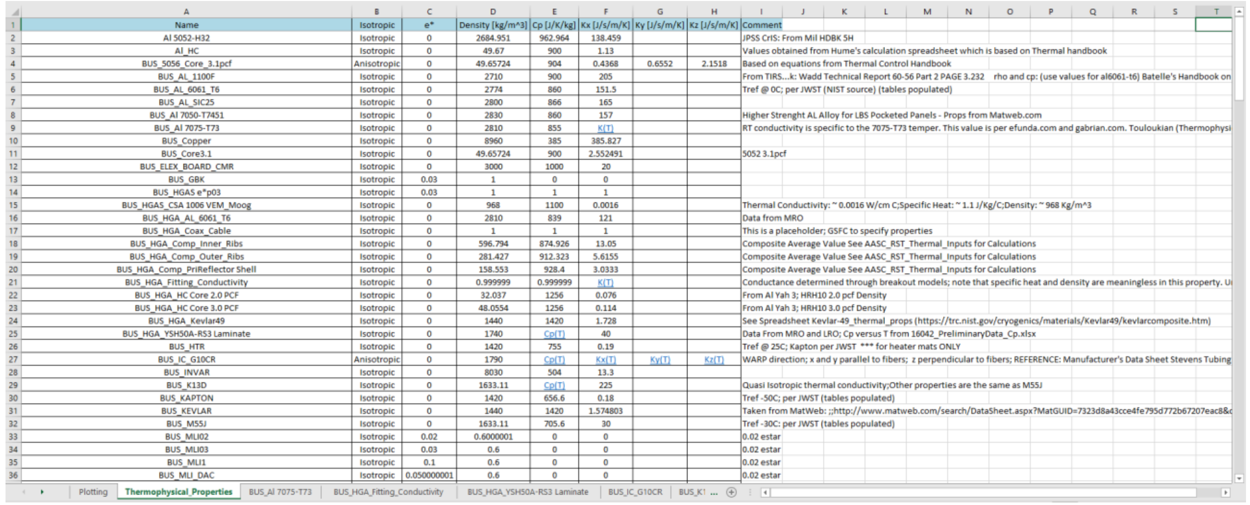
294.421
297.091
(-2.67)

Figure 8. Heater, HeatLoad, PID Controller Summary_Compare Sheet: *The Summary_Compare sheet highlights differences between two files as comments for each cell (including the two values, difference, and percent difference for power values) with a comment in A1 listing the compare filename for reference*

VII. Property Documentation

Thermal Desktop stores optical and material property data in text files and displays the property values in the main GUI. However, the data itself is not easily extracted from these text files without a better understanding of the format nor is it easily exported to other programs from within Thermal Desktop. That said, the API does offer functions for retrieving the data programmatically for further manipulation. Basic functions had been written previously to extract this data, but while the output format was functional, it was not particularly user friendly. A new function was developed to improve the data format for presentations or inclusion in documentation.

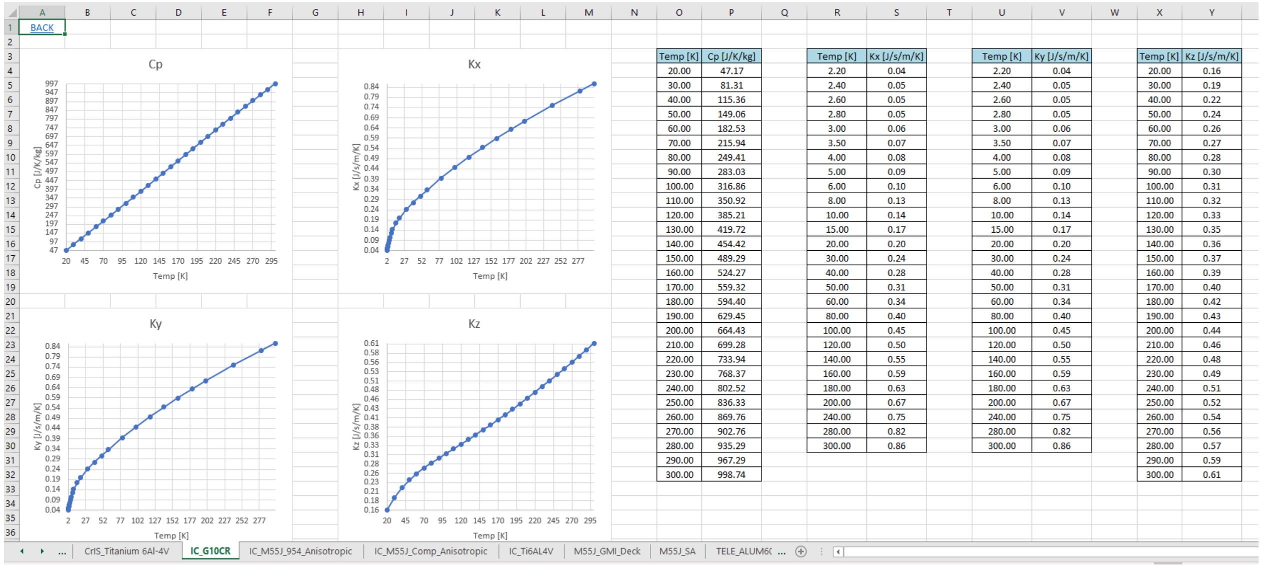
After extracting the properties through the API, a spreadsheet is generated to display the material properties line by line. Pertinent information regarding material name, isotropy, effective emissivity, density, specific heat, thermal conductivity in XYZ directions, and comments are listed in individual columns. If the specific heat or any thermal conductivity are temperature dependent, a new sheet is created with the material name and includes the tabulated values and corresponding plots to visualize the dependence. Hyperlinks are generated to provide greater ease in shifting between the material list and temperature dependent data. Regarding thermo-optical properties, a similar documentation process is done with Beginning-of-Life and End-of-Life as defining categories. Figure 9 shows the thermo-physical property table, while Figure 10 shows an example of a fully populated temperature dependent material sheet.



	Name	Isotropic	e*	Density [kg/m^3]	Cp [J/K/kg]	Kx [J/s/m/K]	Ky [J/s/m/K]	Kz [J/s/m/K]	Comment
2	Al 5052-H32	Isotropic	0	2684.951	962.964	138.459			JPSS CrIS: From Mil HDBK 5H
3	Al_HC	Isotropic	0	49.67	900	1.13			Values obtained from Hume's calculation spreadsheet which is based on Thermal handbook
4	BUS_5056_Core_3.1pcf	Anisotropic	0	49.65724	904	0.4368	0.6552	2.1518	Based on equations from Thermal Control Handbook
5	BUS_AL_1100F	Isotropic	0	2710	900	205			From TIRS...k: Wadd Technical Report 60-56 Part 2 PAGE 3.232 rho and cp: (use values for al6061-t6) Batelle's Handbook on
6	BUS_AL_6061_T6	Isotropic	0	2774	860	151.5			Tref @ 0C; per JWST (NIST source) (tables populated)
7	BUS_AL_SIC25	Isotropic	0	2800	866	165			
8	BUS_Al 7050-T7451	Isotropic	0	2830	860	157			Higher Strenght AL Alloy for LBS Pocketed Panels - Props from Matweb.com
9	BUS_Al 7075-T73	Isotropic	0	2810	855	K(T)			RT conductivity is specific to the 7075-T73 temper. This value is per efunda.com and gabrian.com. Touloukian (Thermophysi
10	BUS_Copper	Isotropic	0	8960	385	385.827			
11	BUS_Core3.1	Isotropic	0	49.65724	900	2.552491			5052 3.1pcf
12	BUS_ELEX_BOARD_CMR	Isotropic	0	3000	1000	20			
13	BUS_GBK	Isotropic	0.03	1	0	0			
14	BUS_HGAS e*p03	Isotropic	0.03	1	1	1			
15	BUS_HGAS_CSA 1006 VEM_Moog	Isotropic	0	968	1100	0.0016			Thermal Conductivity: ~ 0.0016 W/cm C;Specific Heat: ~ 1.1 J/Kg/C;Density: ~ 968 Kg/m^3
16	BUS_HGA_AL_6061_T6	Isotropic	0	2810	839	121			Data from MRO
17	BUS_HGA_Coax_Cable	Isotropic	0	1	1	1			This is a placeholder; GSFC to specify properties
18	BUS_HGA_Comp_Inner_Ribs	Isotropic	0	596.794	874.926	13.05			Composite Average Value See AASC_RST_Thermal_Inputs for Calculations
19	BUS_HGA_Comp_Outer_Ribs	Isotropic	0	281.427	912.323	5.6155			Composite Average Value See AASC_RST_Thermal_Inputs for Calculations
20	BUS_HGA_Comp_PriReflector Shell	Isotropic	0	158.553	928.4	3.0333			Composite Average Value See AASC_RST_Thermal_Inputs for Calculations
21	BUS_HGA_Fitting_Conductivity	Isotropic	0	0.999999	0.999999	K(T)			Conductance determined through breakout models; note that specific heat and density are meaningless in this property. U
22	BUS_HGA_HC Core 2.0 PCF	Isotropic	0	32.037	1256	0.076			From Al Yah 3; HRH10 2.0 pcf Density
23	BUS_HGA_HC Core 3.0 PCF	Isotropic	0	48.0554	1256	0.114			From Al Yah 3; HRH10 3.0 pcf Density
24	BUS_HGA_Kevlar49	Isotropic	0	1440	1420	1.728			See Spreadsheet Kevlar-49_thermal_props (https://trc.nist.gov/cryogenics/materials/Kevlar49/kevlarcomposite.htm)
25	BUS_HGA_YSH50A-RS3 Laminate	Isotropic	0	1740	Cp(T)	40			Data From MRO and LRO; Cp versus T from 16042_PreliminaryData_Cp.xlsx
26	BUS_HTR	Isotropic	0	1420	755	0.19			Tref @ 25C; Kapton per JWST *** for heater mats ONLY
27	BUS_IC_G10CR	Anisotropic	0	1790	Cp(T)	Kx(T)	Ky(T)	Kz(T)	WARP direction; x and y parallel to fibers; z perpendicular to fibers; REFERENCE: Manufacturer's Data Sheet Stevens Tubing
28	BUS_INVAR	Isotropic	0	8030	504	13.3			
29	BUS_K13D	Isotropic	0	1633.11	Cp(T)	225			Quasi Isotropic thermal conductivity;Other properties are the same as M55J
30	BUS_KAPTON	Isotropic	0	1420	656.6	0.18			Tref -50C; per JWST (tables populated)
31	BUS_KEVLAR	Isotropic	0	1440	1420	1.574803			Taken from MatWeb: ;;http://www.matweb.com/search/DataSheet.aspx?MatGUID=7323d8a43cce4fe795d772b67207eac8&c
32	BUS_M55J	Isotropic	0	1633.11	705.6	30			Tref -30C: per JWST (tables populated)
33	BUS_MLI02	Isotropic	0.02	0.6000001	0	0			0.02 estar
34	BUS_MLI03	Isotropic	0.03	0.6	0	0			0.02 estar
35	BUS_MLI1	Isotropic	0.1	0.6	0	0			0.02 estar
36	BUS_MLI_DAC	Isotropic	0.050000001	0.6	0	0			0.02 estar

Plotting
Thermophysical_Properties
BUS_Al 7075-T73
BUS_HGA_Fitting_Conductivity
BUS_HGA_YSH50A-RS3 Laminate
BUS_IC_G10CR
BUS_K1 ...

Figure 9. Thermo-physical Property Sheet: *The excel sheet displays the thermophysical list for further evaluation or comparison between materials. Hyperlinks are highlighted to indicate temperature dependent data is available.*



Temp [K]	Cp [J/K/kg]
20.00	47.17
30.00	81.31
40.00	115.36
50.00	149.06
60.00	182.53
70.00	215.94
80.00	249.41
90.00	283.03
100.00	316.86
110.00	350.92
120.00	385.21
130.00	419.72
140.00	454.42
150.00	489.29
160.00	524.27
170.00	559.32
180.00	594.40
190.00	629.45
200.00	664.43
210.00	699.28
220.00	733.94
230.00	768.37
240.00	802.52
250.00	836.33
260.00	869.76
270.00	902.76
280.00	935.29
290.00	967.29
300.00	998.74

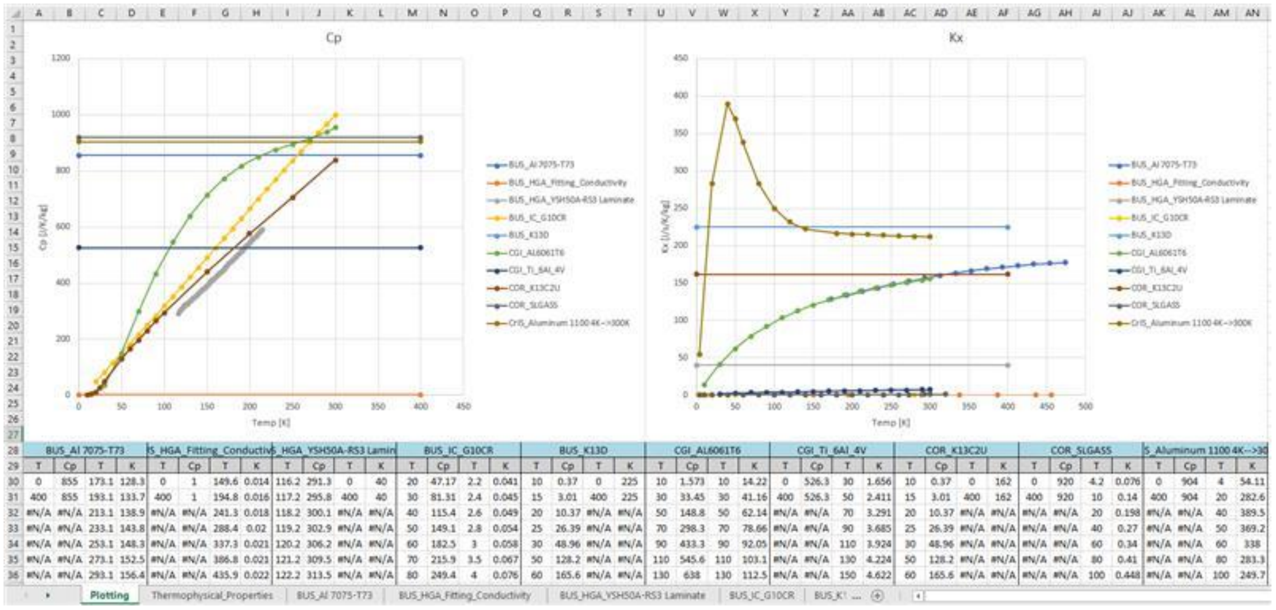
Temp [K]	Kx [J/s/m/K]
2.20	0.04
2.40	0.05
2.60	0.05
2.80	0.05
3.00	0.06
3.50	0.07
4.00	0.08
5.00	0.09
6.00	0.10
8.00	0.13
10.00	0.14
15.00	0.17
20.00	0.20
30.00	0.24
40.00	0.28
50.00	0.31
60.00	0.34
80.00	0.40
100.00	0.45
120.00	0.50
140.00	0.55
160.00	0.59
180.00	0.63
200.00	0.67
240.00	0.75
280.00	0.82
300.00	0.86

Temp [K]	Ky [J/s/m/K]
2.20	0.04
2.40	0.05
2.60	0.05
2.80	0.05
3.00	0.06
3.50	0.07
4.00	0.08
5.00	0.09
6.00	0.10
8.00	0.13
10.00	0.14
15.00	0.17
20.00	0.20
30.00	0.24
40.00	0.28
50.00	0.31
60.00	0.34
80.00	0.40
100.00	0.45
120.00	0.50
140.00	0.55
160.00	0.59
180.00	0.63
200.00	0.67
240.00	0.75
280.00	0.82
300.00	0.86

Temp [K]	Kz [J/s/m/K]
20.00	0.16
30.00	0.19
40.00	0.22
50.00	0.24
60.00	0.26
70.00	0.27
80.00	0.28
90.00	0.30
100.00	0.31
110.00	0.32
120.00	0.33
130.00	0.35
140.00	0.36
150.00	0.37
160.00	0.39
170.00	0.40
180.00	0.42
190.00	0.43
200.00	0.44
210.00	0.46
220.00	0.48
230.00	0.49
240.00	0.51
250.00	0.52
260.00	0.54
270.00	0.56
280.00	0.57
290.00	0.59
300.00	0.61

Figure 10. Temperature Dependent Material Sheet: *The excel sheet displays the temperature dependent properties and associated tabulated values. A back button allows quick return to the table of all properties.*

A material comparison sheet is also provided to compare properties. 10 drop down menus are included, each containing all available materials. Selecting a particular material populates the specific heat and thermal conductivity data table and updates the data displayed in the plots. If the property value is constant, the data is shown as a constant value ranging from a default of 0 to 400. Figure 11 depicts the worksheet after specifying 10 different materials.



28	BUS_Al 7075-T73				S_HGA_Fitting_Conductiv				S_HGA_YSH50A-RS3 Lamin				BUS_IC_G10CR				BUS_K13D			
29	T	Cp	T	K	T	Cp	T	K	T	Cp	T	K	T	Cp	T	K	T	Cp	T	K
30	0	855	173.1	128.3	0	1	149.6	0.014	116.2	291.3	0	40	20	47.17	2.2	0.041	10	0.37	0	225
31	400	855	193.1	133.7	400	1	194.8	0.016	117.2	295.8	400	40	30	81.31	2.4	0.045	15	3.01	400	225
32	#N/A	#N/A	213.1	138.9	#N/A	#N/A	241.3	0.018	118.2	300.1	#N/A	#N/A	40	115.4	2.6	0.049	20	10.37	#N/A	#N/A
33	#N/A	#N/A	233.1	143.8	#N/A	#N/A	288.4	0.02	119.2	302.9	#N/A	#N/A	50	149.1	2.8	0.054	25	26.39	#N/A	#N/A
34	#N/A	#N/A	253.1	148.3	#N/A	#N/A	337.3	0.021	120.2	306.2	#N/A	#N/A	60	182.5	3	0.058	30	48.96	#N/A	#N/A
35	#N/A	#N/A	273.1	152.5	#N/A	#N/A	386.8	0.021	121.2	309.5	#N/A	#N/A	70	215.9	3.5	0.067	50	128.2	#N/A	#N/A
36	#N/A	#N/A	293.1	156.4	#N/A	#N/A	435.9	0.022	122.2	313.5	#N/A	#N/A	80	249.4	4	0.076	60	165.6	#N/A	#N/A

28	CGI_AL6061T6				CGI_Ti_6Al_4V				COR_K13C2U				COR_SLGASS				S_Aluminum 1100 4K-->30			
29	T	Cp	T	K	T	Cp	T	K	T	Cp	T	K	T	Cp	T	K	T	Cp	T	K
30	10	1.573	10	14.22	0	526.3	30	1.656	10	0.37	0	162	0	920	4.2	0.076	0	904	4	54.11
31	30	33.45	30	41.16	400	526.3	50	2.411	15	3.01	400	162	400	920	10	0.14	400	904	20	282.6
32	50	148.8	50	62.14	#N/A	#N/A	70	3.291	20	10.37	#N/A	#N/A	#N/A	#N/A	20	0.198	#N/A	#N/A	40	389.5
33	70	298.3	70	78.66	#N/A	#N/A	90	3.685	25	26.39	#N/A	#N/A	#N/A	#N/A	40	0.27	#N/A	#N/A	50	369.2
34	90	433.3	90	92.05	#N/A	#N/A	110	3.924	30	48.96	#N/A	#N/A	#N/A	#N/A	60	0.34	#N/A	#N/A	60	338
35	110	545.6	110	103.1	#N/A	#N/A	130	4.224	50	128.2	#N/A	#N/A	#N/A	#N/A	80	0.41	#N/A	#N/A	80	283.3
36	130	638	130	112.5	#N/A	#N/A	150	4.622	60	165.6	#N/A	#N/A	#N/A	#N/A	100	0.448	#N/A	#N/A	100	249.7

Figure 11. Material Comparison Sheet: *The excel sheet display graphs of specific heat and thermal conductivity in the x-direction for up to 10 materials. Each material has its respective values beneath the graphs.*

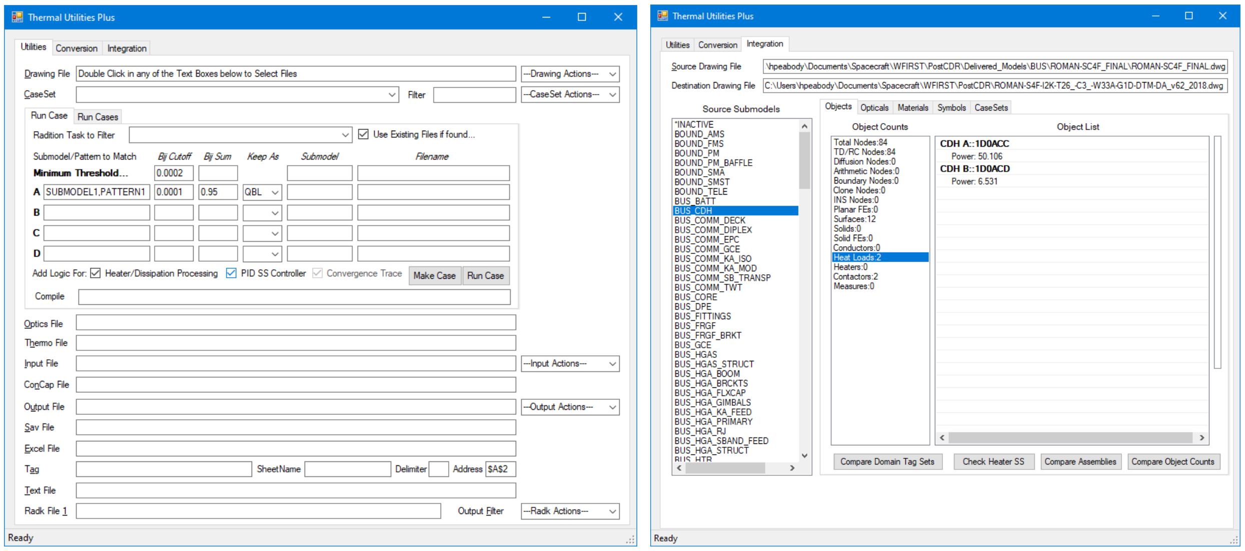
VIII. Framework and Interface

The various capabilities described in this paper were added to the framework[1] developed at the Goddard Space Flight Center. Table 1 lists the new functions added, and a complete list of functions in the framework may be found in the appendix. The goal was to minimize, as much as possible, the need for a graphical interface to access the capabilities of the framework, allowing users to develop their own interface to the framework capabilities. Entries in italics utilize the OpenTD API, while the others are utility functions that do not require a connection to the API.

Framework Function	**Purpose**
GenerateDampedPropHeatersInSSForPIDs	Process SINDA .inp file and add logic to .pid file to apply Predictor/Corrector approach for all PID controllers
GenerateHtrDisSummaryCompare	Generate Compare_Summary sheet highlighting the differences between two HtrDis postprocessing files
WritePropsToXL	*Extract Material and Optical property data and generate Tables in Excel workbook along with temperature dependent material property plots*
GetTDGlobalVisibilityStates	*Return data structure with deterministic global visibility state for each Thermal Desktop object type (e.g. TD/RC nodes, Surfaces, HeatLoads, etc)*
TurnNodeIDsOnForSelectedNodes	*Evaluate user provided node list and turn node number visibility on in GUI*
GetTDObjectCounts	*Retrieve counts for each TD object type for each submodel*
GetTDReferencedProperties	*Retrieve list of Material and Optical properties used by each submodel*
GetTDObjectCountsAndReferencedProperties	*Retrieve counts for each TD object type and lists of Material and Optical properties for each submodel*
GetAllReferencedRCOFiles	*Make list of all Optical files that were referenced throughout all CaseSets*
GetAllReferencedTDPFiles	*Make list of all Material files that were referenced throughout all CaseSets*

Table 1 – List of new Functions added to GSFC Developed Framework

The GUI developed at GSFC was updated to provide a means of access to these functions without the need for a user to write their own code to access the functionality. This GUI is currently split across two separate tabs; one is useful for manipulating and processing files and executing CaseSets. The other is geared toward utilities for model integration/comparison. This GUI is depicted in Figure 12.



Drawing File: Double Click in any of the Text Boxes below to Select Files
CaseSet:
Filter:
Radition Task to Filter:
[✓] Use Existing Files if found...

Submodel/Pattern to Match		Bij Cutoff	Bij Sum	Keep As	Submodel	Filename
Minimum Threshold...		0.0002				
A	SUBMODEL1,PATTERN1	0.0001	0.95	QBL		
B						
C						
D						

Add Logic For: [✓] Heater/Dissipation Processing [✓] PID SS Controller [✓] Convergence Trace
Compile:
Optics File:
Thermo File:
Input File:
ConCap File:
Output File:
Sav File:
Excel File:
Tag: SheetName: Delimiter: Address: A2
Text File:
Radk File 1: Output Filter:

Source Drawing File: \hpeabody\Documents\Spacecraft\WFIRST\PostCDR\Delivered_Models\BUS\ROMAN-SC4F_FINAL\ROMAN-SC4F_FINAL.dwg
Destination Drawing File: C:\Users\hpeabody\Documents\Spacecraft\WFIRST\PostCDR\ROMAN-S4F-I2K-T26_-C3_-W33A-G1D-DTM-DA_v62_2018.dwg

Source Submodels:
```
*INACTIVE
BOUND_AMS
BOUND_FMS
BOUND_PM
BOUND_PM_BAFFLE
BOUND_SMA
BOUND_SMST
BOUND_TELE
BUS_BATT
BUS_CDH
BUS_COMM_DECK
BUS_COMM_DIPLEX
BUS_COMM_EPC
BUS_COMM_GCE
BUS_COMM_KA_ISO
BUS_COMM_KA_MOD
BUS_COMM_SB_TRANSP
BUS_COMM_TWT
BUS_CORE
BUS_DPE
BUS_FITTINGS
BUS_FRGF
BUS_FRGF_BRKT
BUS_GCE
BUS_HGAS
BUS_HGAS_STRUCT
BUS_HGA_BOOM
BUS_HGA_BRCKTS
BUS_HGA_FLXCAP
BUS_HGA_GIMBALS
BUS_HGA_KA_FEED
BUS_HGA_PRIMARY
BUS_HGA_RJ
BUS_HGA_SBAND_FEED
BUS_HGA_STRUCT
BUS_HTR
```

Object Counts:
```
Total Nodes:84
TD/RC Nodes:84
Diffusion Nodes:0
Arithmetic Nodes:0
Boundary Nodes:0
Clone Nodes:0
INS Nodes:0
Planar FEs:0
Surfaces:12
Solids:0
Solid FEs:0
Conductors:0
Heat Loads:2
Heaters:0
Contactors:2
Measures:0
```

Object List:
```
CDH A::1D0ACC
    Power: 50.106
CDH B::1D0ACD
    Power: 6.531
```

Figure 12. Simplistic Graphical User Interface to access framework functions: *A simplistic interface was developed to more easily specify files and parameters to access the framework capabilities*

IX. Conclusions and Path Forward

The continued addition of capabilities to the OpenTD API has allowed users greater access to the internal data in Thermal Desktop models. The Goddard Space Flight Center continues to explore the exposed data, seeking new methods to optimize our thermal analysis process and expand our capabilities. The ability to inject these features into the model execution process has helped streamline the process and minimizes the amount of user intervention required to take advantage of the developed features.

The most recent efforts have focused primarily on improving the model integration and verification part of the process. These added abilities allow a quick check that the same number of objects are present in both the source and integrated models for each submodel and to import symbols from delivered case sets into their counterparts in the integrated model. Further development of the "Compare" capability of the Heat Load, Heater, and PID Controller summary sheets also allows for the verification of consistent predictions between delivered and integrated models. To provide better documentation of the optical and material properties, the output from the previous code was improved to more easily visualize the data extracted from the models. Lastly, the development of a new heater routine for better predictive capabilities of PID controlled heaters is already beginning to pay dividends for the Roman Space Telescope project, with a considerable reduction in the necessary run time to achieve quasi-steady predictions for many of its mission modes. Furthermore, the ability to utilize steady state solution predictions, bypassing the need for longer transient runs, is also being investigated and shows promise at this time. While this code was developed specifically for SINDA, the algorithm is fairly easily convertible to other codes that utilize a similar solution structure as SINDA with convergence and VARIABLES user logic blocks.

The process is currently underway to allow the release of these capabilities to the wider community, both the executable and the source code, with the hope that other organizations will add to these capabilities and also share their developments with the community. At this time, there are no plans to output to other formats, but once the source code is released, any users wishing to expand the output options to other formats (e.g. web-based pages) are free to utilize the framework provided to assist in that effort. As new capabilities are added to the OpenTD API, GSFC will continue to explore the exposed data seeking even more improvements to its analysis processes and work to share them with the thermal community.

Acknowledgments

The primary author would like to thank the Roman Space Telescope project for continuing to support the exploration of the API to improve our processes and reduce the model run time.

References

[1] Peabody, H., "Extending the Capabilities of Thermal Desktop with the OpenTD Application Programming Interface" ICES-2020-297, *50th International Conference on Environmental Systems*, 2020

[2] Peabody, H., "Tracking Critical Thermal Metrics throughout the Life Cycle of a Large Observatory Thermal Model" ICES-2020-298, *50th International Conference on Environmental Systems*, 2020

[3] Peabody, H., Yee, E., “Run Time Improvement Efforts for the Roman Space Telescope Thermal Analysis” ICES-2021-253, *51st International Conference on Environmental Systems*, 2021

Appendix

The table below lists the current functions in the framework. Those with a preceding * are newly added. Those in italics interface directly with Thermal Desktop through the OpenTD API, while the others interface with files that may or may not have been generated using Thermal Desktop and does not require the API.

Framework Function	**Purpose**
GenerateHeaterDissipationLogic	Process SINDA .inp file and add logic to .htr file to output Heater and Dissipation output logic for every timestep
ProcessHeaterDissipationResults	Process SINDA .out file and retrieve output generated from GenerateHeaterDissipationLogic and import into Excel template workbook
*GenerateDampedPropHeatersInSSForPIDs	Process SINDA .inp file and add logic to .pid file to apply Predictor/Corrector approach for all PID controllers
*GenerateHtrDisSummaryCompare	Generate Compare_Summary sheet highlighting the differences between two HtrDis postprocessing files
Write2DArrayToCSVFile	Convert 2D array to comma-separated value output file (can include Header row and Output Mask)
ImportFileIntoExcel	Import specified text file into specified Excel location, parsing on delimiter
ExtractTaggedLinesAndImportIntoExcel	Extract lines beginning with TagID from OutFile and import into specified Excel location, parsing on specified delimiter
**GetTDGlobalVisibilityStates*	*Return data structure with deterministic global visibility state for each Thermal Desktop object type (e.g. TD/RC nodes, Surfaces, HeatLoads, etc)*
**TurnNodeIDsOnForSelectedNodes*	*Evaluate user provided node list and turn node number visibility on in GUI*
**GetTDObjectCounts*	*Retrieve counts for each TD object type for each submodel*
**GetTDReferencedProperties*	*Retrieve list of Material and Optical properties used by each submodel*
**GetTDObjectCountsAndReferencedProperties*	*Retrieve counts for each TD object type and lists of Material and Optical properties for each submodel*
**GetAllReferencedRCOFiles*	*Make list of all Optical files that were referenced throughout all CaseSets*
**GetAllReferencedTDPFiles*	*Make list of all Material files that were referenced throughout all CaseSets*
RemoveRadk1FromRadk2	Outputs file with all the Radks that appear only in Radk File 2
ReplaceRadk2MinusRadk1WithBackloads	Outputs file with inputs and logic to represent all the Radks in Radk2 but not in Radk1 as Backloads along with radiation to sink
ReplaceRadk2MinusRadk1WithHeatFlowsIJ	Outputs file with inputs and logic to represent all the Radks in Radk2 but not in Radk1 as a Heat Flow
EvaluateSymbolsInDWGFile	*Generates temporary case set, outputs CC file and processes this for evaluated symbol values*
EvaluateSymbolsForSpecifiedCaseSet	*Generates temporary case set spawned from user specified case, outputs CC file and processes this for evaluated symbol values*
ExtractSymbolEvaluatedValuesFromCCFile	Process CC file header and retrieves symbol names and evaluated values
GetTDOptProps	*Read TD object and extract optical properties and store in GMM_OpticalProperties collection*
GetTDThermoProps	*Read TD object and extract thermophysical properties and store in GMM_ThermophiysicalProperties collection*
**WritePropsToXL*	*Extract Material and Optical property data and generate Tables in Excel workbook along with temperature dependent material property plots*
GetTDNotes	*Read TD object and extract Notes data and Splice together all Tabs into single output text file*
GetTDRunDirectory	*Return path to either DWG file if No CaseSet specified with UserDirectory, or to UserDirectory*
RunSpecifiedCaseSet	*Execute case set in its own directory with options to add heater dissipation and convergence trace logic*