

Chapter 6

Running Modules

This chapter details both the single-module **Run** and **Run Group** functions available to **Normal** users. Additionally, it describes the **Run State Editor** (available directly on a **Run** form or via the **Configure** submenus) which provides user-defined control scripts for acquiring data from modules operating with either **Run** function. **Chapter 3** only described how to activate these functions from other forms and menus. Here they are described in their fullest functional detail with pictures and examples.

Run is the most useful function in NUSS. It is multifaceted, being able to accomplish, in one single form, many integrated NetScanner module operating features including **Acquire**, **Display**, **Record**, **Playback**, and **Module Control**. Such a form adjusts itself to the number of channels and unique features of a particular model.

Single-module **Run** is not limited to running only one module at a time. Instead, multiple instances of this form may be activated concurrently under NUSS, each for a *different* unique module. This is accomplished from the *context* menu of individual connected modules — or by clicking a special “hidden” control (the **Number of Connected Modules** box (labeled #Con=) on the **Network Status** form).

Run Group is an alternative way of operating *multiple* modules concurrently, as a **Coordinated Group**. It operates mutually-exclusive of the other instances of single-module **Run** forms, and is started from the “home-base” menu’s **Run** function. All its **Acquire**, **Record**, and **Playback** features are centralized on a single **Run Group** form — while data appear on separate **Display** forms, one for each module. **Module Control** features appear only on the separate **Display** forms. This function acts on modules defined in a named group (A-Z or *) that is assigned as NUSS’s **Current Group**.

The **Run State Editor** would not be required if you would be happy with the way NUSS chooses to **Run** modules by default. All *new* modules begin their life under NUSS assigned to a default **Run State 0** (Zero), a special run state not generated by the **Run State Editor**. NUSS creates it when a *new* module is **Run** for the first time. Though it is written to a special “script” file (named <modid>rs0.ini), the **Run State Editor** only allows minor changes to it. Thus, while a module stays assigned to **Run State 0**, its **Run** behavior is relatively slow and predictable. In general, **Run State 0** acquires three autonomous streams from the module assigned to it: One stream acquires E.U. pressure data every 0.5 sec., another acquires “other” pressure data and internal module status every two (2) seconds, and a final stream acquires temperature data every 7.5 seconds. Each stream scans all the

channels of the module. The **Display** feature of **Run** updates appropriate *tabular*, *graphical*, and *status* fields when an arriving stream contains data applicable to it.

NOTE: to insure that your host computer is able to keep up with data acquired by each **Run** form, it is important that you choose it wisely. You should consider a PC or workstation that uses the fastest possible microprocessor chip, has the fastest possible peripheral busses, and has very fast buffered video display capabilities (e.g., AGP). For maximum **recording** capabilities, also consider a hard disk with a buffered controller and the fastest rotational speeds and shortest seek times. The *maximum* number of modules that **Run** can process depends on many variables, and can only be empirically determined.

6.1 Single-Module Run

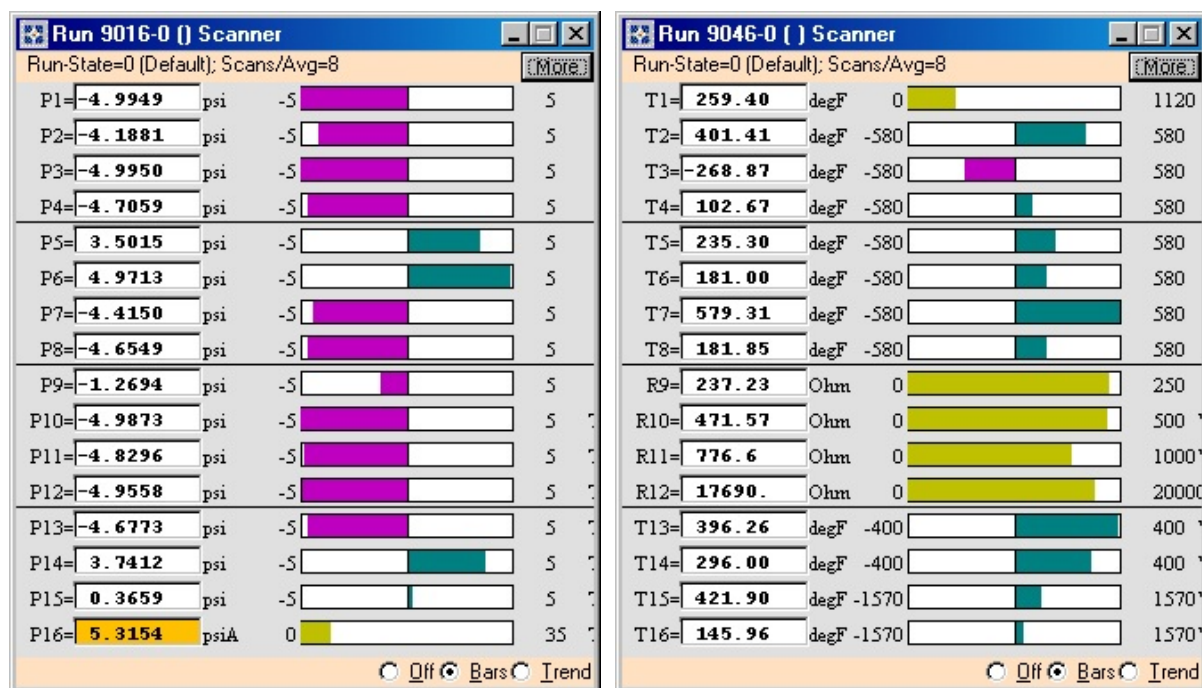
The **Run** function, selected from a module's *context* menu, opens a form titled:

“Run <modid> (<yourmodname>) <modclass>”
(e.g., **“Run 9016-1234 (Fender 2) Scanner”**).

More than one module may have this function operating concurrently, each running in its own separate **Run** form. All module classes, *scanners*, *calibrators*, and *standards*, generally have the same basic **Run** form format, except that the *number of displayed data channels* adjusts to what that particular module has (or is being scanned by its current **Run State**). Also, different module classes often have different *module control functions* (e.g., because of different types of internal valves, etc.). *Calibrator* (model 903x) modules have an addition frame of controls for controlling its *pressure-generation* function that *scanner* modules do not have.

6.1.1 Simple Scanner Run Forms

By default, when a module has never been operated by NUSS before, only a *Simple Run* form initially appears (see examples on next page). It displays continuously-refreshed EU pressure data, using the default **Run State 0**. Every input channel of the selected module is displayed, with no obvious ability to change it. The only other controls visible allow the **Run** form to *exit* by clicking [X] or to change its *Graphical Data* between default *Bar Graphs* and an alternate *Trend Graph* form. The [More] button is described later. In earlier NUSS versions there was an extra [Exit] button, now removed.

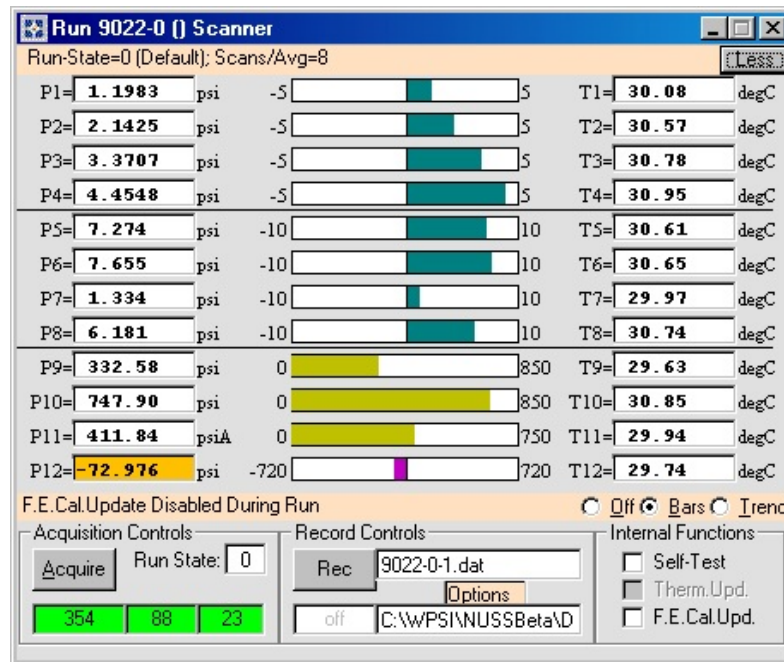
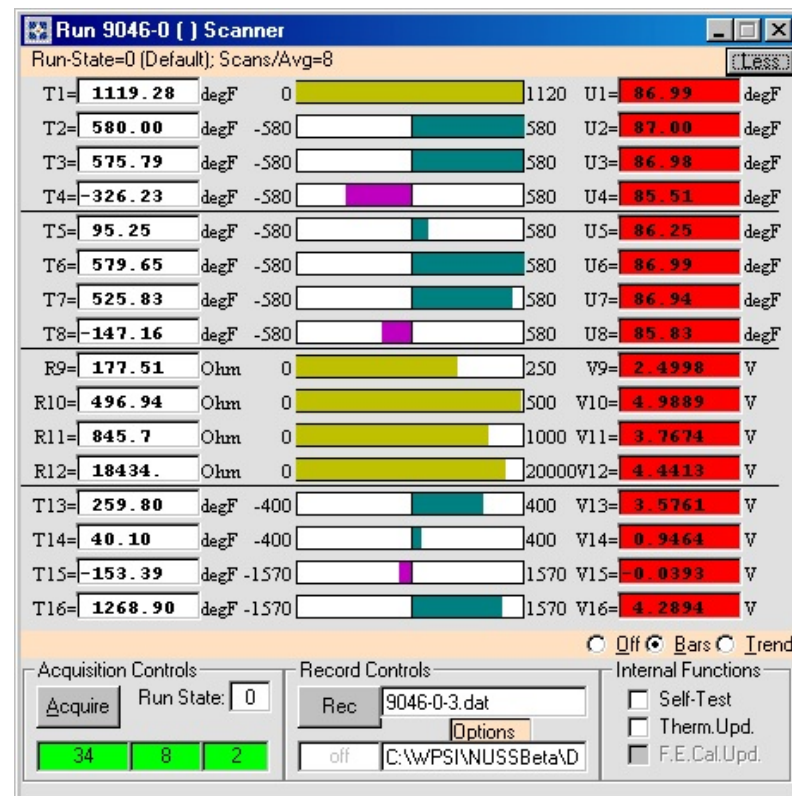
Simple **Run** form for Model 9016 ModuleSimple **Run** form for Model 9046 Module

In these “Simple” examples of similar scanner models (Models 9016 and 9046) there are 16 channels displayed in both tabular and graphic (bar graph) forms. A Model 9816 would have 18 channels, and a Model 902x would have only 12 channels.

6.1.2 Maximum Scanner Run Forms

If the [**More**] button is clicked (on above forms), it widens (at the right) to reveal one “other” datum per channel, and lengthens (at the bottom) to reveal frames of extra functional *controls* that operate additional **Run** features (see example next page). This is known as the **Maximum Run** form, and reveals all the displayable data and additional features. This **Maximum** form replaces the [**More**] button with a [**Less**] button (used to hide these extra features when not needed).

The particular examples (next page) shows a Model 9022 pressure scanner and a Model 9046 temperature/resistance scanner in their **Maximum** form. Each has an **Internal Functions** frame (lower right corner). It appears (by default) in place of an otherwise empty **Valve Controls** frame, since these models have no internal valves. Also notice that the 9022s form is smaller *vertically* since it only needs to display this module’s 12 channels. Both types of module also allow each channel to be separately configurable – by the user plugging in various specific modules.

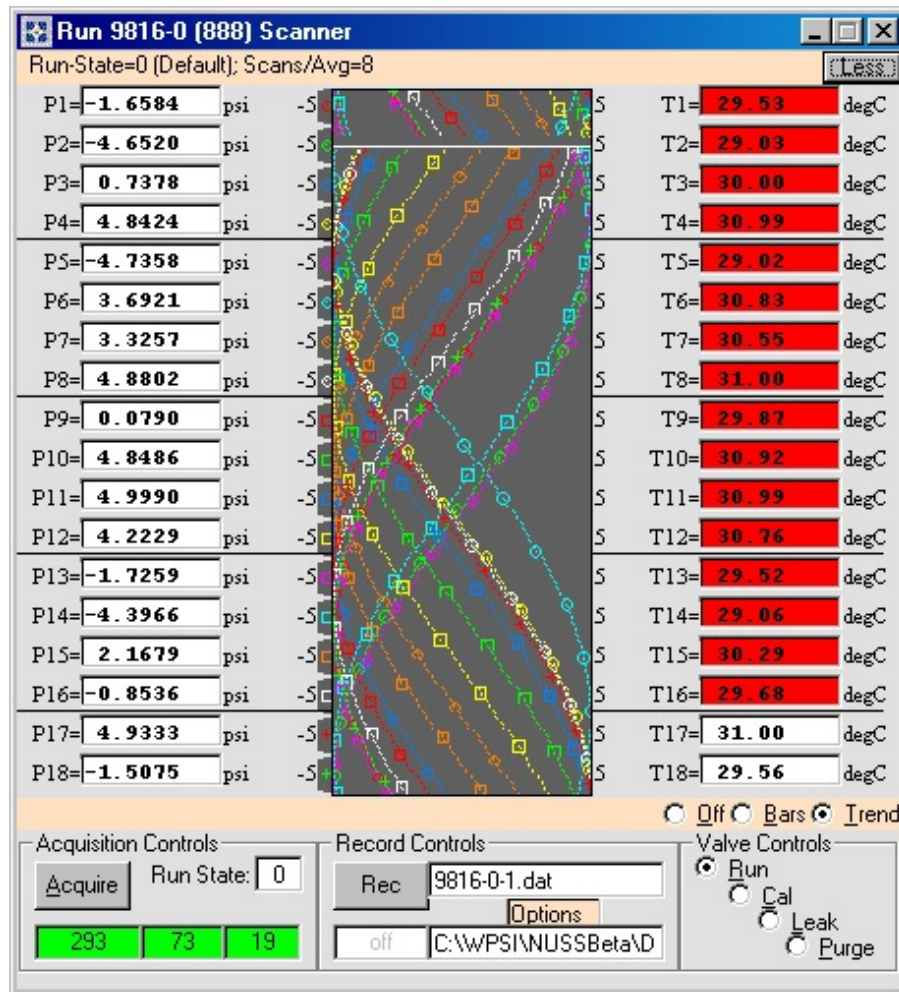
Maximum **Run** form for Model 9022 ModuleMaximum **Run** form for a Model 9046 Module

In the first example above, external Model 9400 pressure transducers of various ranges (and pressure modes) are apparently plugged into every channel of this “simulated” Model 9022 module. In the second example, a simulated Model 9046 temperature/resistance scanner is also configured with various different types of temperature or resistance sensors. When a real module has no transducer configured for a channel, its corresponding EU tabular data column displays *volts* instead of *pressure* (9022) or *temperature/resistance* (9046), and its label has a “V” instead of a “P” (9022) or “T” (for 9046). Other differences between these two types of scanner **Run** forms should be obvious from their examples. The pressure scanner’s primary EU data (left tabular column and bar graph) measure pressure, while the secondary (other) data measure the slower scanned temperature used to compensate the pressure measurements of the unit. The temperature scanner’s primary EU data measure temperature (in °C or °F) or resistance (in ohms), while the secondary (other) data measure the slower scanned Uniform Temperature Reference (UTR) junction temperature used to accurately characterize each thermocouple measurement. Other non-thermocouple channels of the 9046 show an internal Reference Voltage as secondary (other) data.

The Model 9046 example shows *bright red* data in the rightmost column of the form. These “alarms” indicate that its reference junction is “out-of-tolerance” with the average value of junction temperature measured for all other TC channels. Such **Temperature Alarm** status data are acquired periodically in *stream 3* of Run State 0.

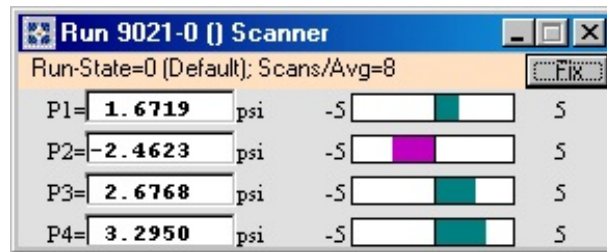
Another example of the *Maximum Run* form is shown on the following page. It is larger vertically because it is for a simulated Model 9816 module with a full 18-pressure channels. Channels P17 and P18 are special S & P channels which are actually located in the 98RK rack but *mapped* to *every* module in the rack. Since a Model 9816 has valves, the **Internal Functions** frame is replaced with a **Valve Controls** frame by default. However, it does have an **Internal Functions** frame, obtained by clicking the frame’s label. In fact, clicking either frame’s label cause the other frame to appear.

This Model 9816 form example also just happens to display a *Trend Graph* rather than the default *Bar Graph*. Also, like the previous Model 9046 example, each *bright red* datum on the rightmost column of this form indicates that its displayed compensation temperature is “out-of-tolerance”. That is, the current temperature being measured is either above the High Temperature Limit or below the Low Temperature Limit. When a channel’s data returns to normal temperature, a *white* background is displayed. Such **Temperature Alarm** status data are acquired periodically in *stream 3* of Run State 0. Notice that the two special “rack” channels (17 and 18) are not in alarm — because such “rack” channels are not limit-checked by the Model 9816's module firmware.

Maximum **Run** from for 9816 Scanner with Trend Graph and Valves

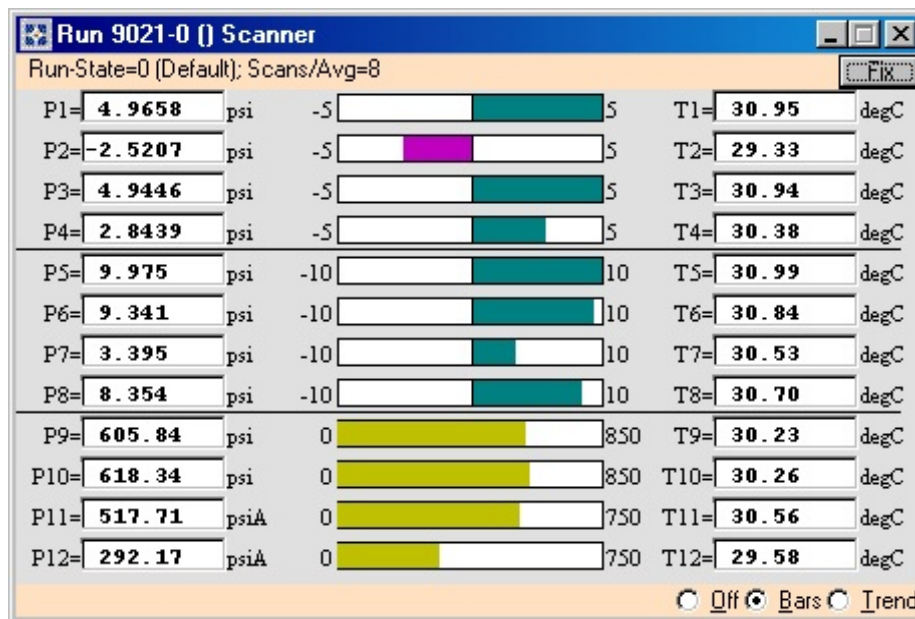
6.1.3 Custom Scanner Run Forms

In addition to the convenient **[More]** and **[Less]** buttons, you may also add or subtract functionality (in finer “custom” increments) on a **Run** form by simply using the standard Windows mouse “drag” functions to change the height and width. To do this first position your mouse carefully on a form’s *border* or *corner*, until a two-headed arrow cursor appears. Then, hold down (left mouse button) while dragging the form’s border/corner (or outline) to change size. With this method, you *reveal* or *hide* the functionality you need. When you *exit* the form, it remembers its current height and width, and restore it the next time **Run** is started, for the same module. A **[Fix]** button replaces the **[More]** or **[Less]** button for any of these intermediate *Custom Run* forms. Clicking **[Fix]** restores the default standard “minimum” **Run** form.

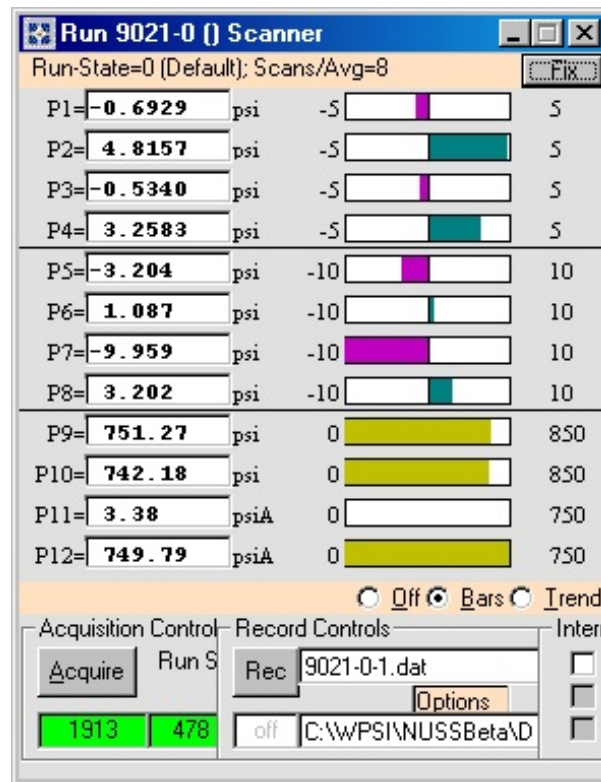
Extreme Custom **Run** Form

There may be many “flavors” of Custom Run forms. If you need neither the bar graphs nor the trend graph, simply drag the right edge of the form to the left with your mouse, until this graphic region is no longer visible (or leave a little bar graph). In the extreme example above only the first four channels of data are left showing

If you may drag the bottom edge up (to hide the controls) you get this example below:

Custom **Run** form showing both data columns (but no controls)

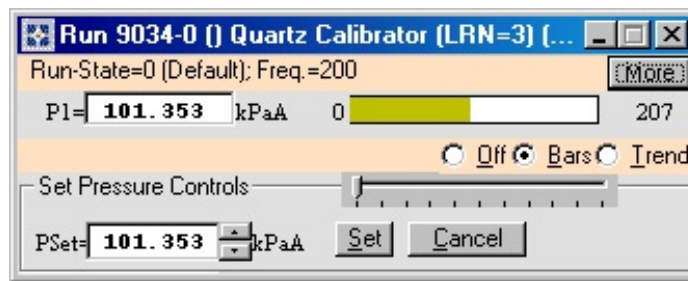
If you drag the right boundary left (to hide the “other” datum column and reduce size of the graph/bar plotting area), the other “bottom” control frames may shrink in width and even shift left, overlapping part of their neighbors. Such controls frames may function fully even when partially hidden in this way. See example on following page.

Custom **Run** from with overlapping controls

Notice that [**Acquire**] button and **Run State** selection functions still work if you click them. Even though only 2 of the 3 *stream counters* are still visible, you can still **Pause** streams by clicking any one of them. The right-most **Internal Functions** frame's three check boxes also still work, even though you can no longer read their labels.

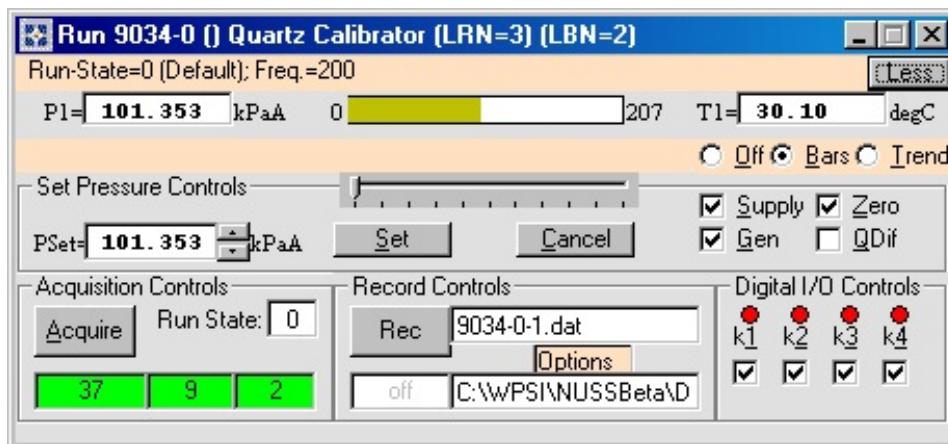
6.1.4 Simple and Maximum Calibrator Run Forms

A *calibrator* module (with only one *input* channel) has a similar but smaller *Simple* form. However, it does have an extra **Set Pressure Controls** frame contains controls for *setting* its *output* pressure. Also note that, in this example of an *Absolute* calibrator, its bar graph scales become *unipolar* (no negative section), with zero pressure at the left end instead of in the middle. A different *gold* color bar is used to indicate such a scale, instead of the two-color (negative=*dark magenta*, positive=*dark cyan*) bipolar bar graph scales we have seen so far. However, such units also have a *Quasi-Differential* mode (selected by *QDif* switch) that reverts its form to a bipolar bar graph when so selected.



Simple **Run** form for Model 9034 “Quartz” Calibrator

The *Maximum* form for a *calibrator* module, also expands to the right and down to reveal its additional features. Its unique internal control “valve states” are operated/indicated by *check box* controls *inside* the expanded **Set Pressure Controls** frame. A **Digital I/O Controls** frame replaces the **Valve Controls** frame. There is also a hidden **Internal Functions** frame if you click the **Digital I/O Controls** frame’s title. Clicking the *frame title* of the other **Acquisition Controls** and **Record Controls** reveals the same “hidden” controls available for *scanner* modules.



Maximum **Run** form for Model 9034 “Quartz” Calibrator

6.1.5 Simple and Maximum Standard/Barometer Run Forms

For a *standard* (or *barometer*) module, the various types of **Run** forms look similar to the *calibrator* forms above, except that the **Set Pressure Controls** frame is mostly **empty**.

Run 9032-0 () Quartz Standard (LBN=1)

Run-State=0 (Default); Freq.=200

P1= 14.700 psiA 0 30 T1= 30.10 degC

Off Bars Irend

Set Pressure Controls

QDif

Custom **Run** form for Model 9032 “Quartz” Absolute Standard

Custom **Run** form examples shown on this page (full-width, bottom frames hidden) illustrate these differences best. For an *absolute* unit (see example above), it has only a **QDif** checkbox for manually putting the unit into its *Quasi-Differential* mode. It is shown running in its “native” *Absolute* mode above and running in its *QDif* mode below.

Run 9032-0 () Quartz Standard (LBN=1)

Run-State=0 (Default); Freq.=200

P1= 0.000 psi -15 15 T1= 30.10 degC

Off Bars Irend

Set Pressure Controls

QDif

Custom **Run** form for Model 9032 Running in *QDif* mode

A “native” *differential* unit has a single **Zero** valve control in this frame (see example below). Notice that for a *Differential* type of *standard* (or *calibrator*), or for an *Absolute* type operating in *QDif* mode the bar graph returns to a *bipolar* (two color) format as was used for *scanners*.

Run 9033-0 () DPT Standard

Run-State=0 (Default); Scans/Avg=20

P1= -0.0002 psi -5 5 T1= 30.10 degC

Off Bars Irend

Set Pressure Controls

Zero

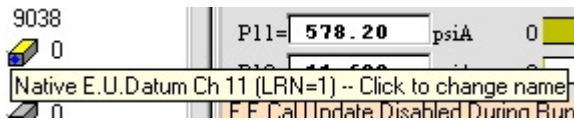
Custom **Run** form for Model 9033 “DPT” Differential Standard

Examples of the *Maximum Run* forms for these units are not shown, as their *bottom* control frames are exactly the same as for the *calibrator* units shown in the previous section.

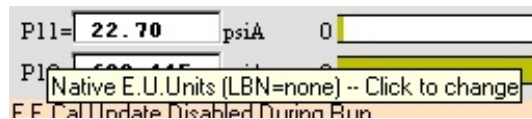
6.1.6 Hidden Features on *Run* Forms Summarized

It may be useful to illustrate several “hidden” features of all *Run* forms, before we proceed with detailed descriptions of them — and all other visible control frames (as described in the previous sections, for specific module classes). These details follow in the next section. All hidden functionality is revealed via pop-up *ToolTips*. Simply allow your mouse cursor to rest (hover) for a few seconds over just about any frame, control, label, or data field and a small rectangular window opens with a message describing that feature. In some cases the message is static, in other cases it (or part of it) is created dynamically.

The following three examples show the *ToolTips* obtained by hovering over three different controls (labels) used for displaying Pressure Channel P11 below.

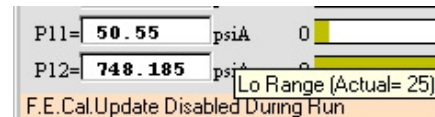


Hovering the *P11=* label itself (note LRN)



Hovering P11's *units* label (note LBN)

The static part of the message, in the first two *ToolTip* examples above, let the user know that clicking that label pops-up a configuration form that can change that channel's default *name* label (P11) or *units* (psiA). Additionally, the dynamic part of the message (in parentheses) shows that channel's current LRN or LBN assignments, if any.



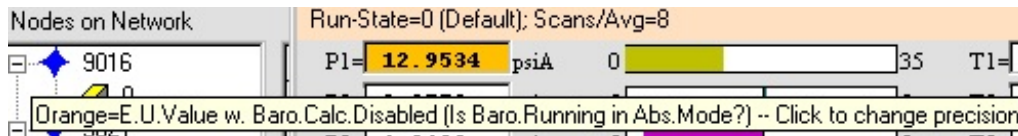
Hovering P11's *Lo Range* bar
graph label (note Actual value)

The third *ToolTip* example above (hovering cursor over a *Lo Range* label of P11's Bar/Trend graph) displays the “actual” *Lo Range* value (25 psiA) of that channel's transducer, even though the actual bar graph *Lo Range* is labeled zero (0). Such a value may differ from the labeled *Lo Range* end of the graph because all Bar/Trend graphs are scaled in one of two ways:

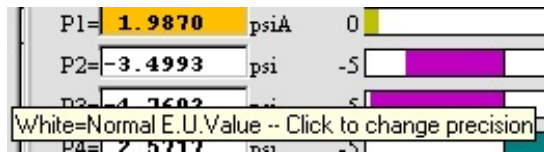
- (1) with the data *zero* value located at the *extreme left end* or
- (2) with the data *zero* value located exactly in the *center*.

Some actual calibrated pressure and temperature ranges of transducers do not fit this restriction. Their *high* and *low* range ends may be *unbalanced* (thus their *zero* data value may appear other than at the center), or there may be no *zero* value at all (as is the absolute pressure example above that is calibrated only from 25 to 750 psiA).

Hovering over the *tabular value* field (between the *name* and *units* labels) for each displayed channel also displays useful information about the color of the datum, which can change dynamically depending upon data quality – and due to special calculations being assigned to replace the normal live datum.



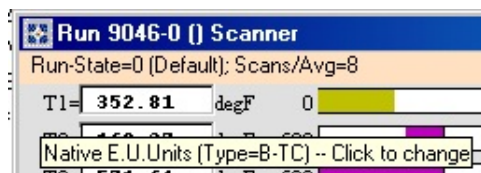
Hovering P1's orange datum value field (note color and precision notice)



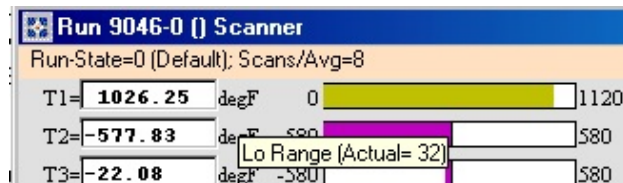
Hovering P2's white datum value field

The trailing static part in both examples above notify the user that clicking this datum field pops-up a configuration form that can change the displayed precision of that channel's data. Additionally, any normal (white) or other (orange, light yellow, or light blue) datum value color is explained in the first part. The light yellow and light blue colors (no examples shown) are used to indicate two types of special barometer calculations described later.

Another example of ToolTips for the **Run** form of a Model 9x46 temperature/resistance scanner are shown below. The *units* are labeled degF and the bar graph *Lo Range* is labeled zero (0) degF. The dynamic (in parentheses) part of the *units* label's ToolTip (left example below) shows this channel to be a "Type B" Thermocouple. The dynamic part of the *Lo Range* label's ToolTip (right example below) shows this channel to actually have a *Lo Range* value of 32 degF. This is because all Type B Thermocouples actually have a range of "0 to 600 degC" as taken from the **SenType9046.Txt** file in the **Ini** subfolder when NUSS starts. Any 9x46 module has a changeable coefficient (common to all channels) that select either degC (default) or degF units for all temperature channels. See **Section 3.3.3.2 in Chapter 3** for more information.



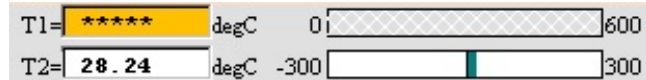
ToolTip for T1's *units* label shows the Sensor Type of a Type B Thermocouple



ToolTip for T1's *Lo Range* label shows the Actual *Lo Range* of 32 degF instead of 0 degF

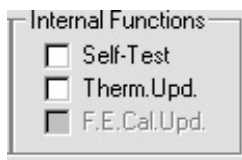
These various ToolTips actually reveal many hidden *interactive configuration features* of this module's **Run** form's **Display Set** (which is independent of, but saved with each numbered **Run State**). We discuss the **Run State** and its “parallel” **Display Set** — and ways to configure special user-defined versions of each in following sections.

The *orange* tabular datum color also shows up when a channel's datum is “over-scale” (*****) or not meaningful for some reason

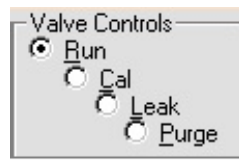


(see datum T1 inset right). Notice that the corresponding bar graph also shows “over-scale” with a *gray-crosshatch* color without any visible bars. Also, in this particular example (for a Model 9046 temperature scanner) the thermal channel (T1) may give the “over-scale” indication due to the module having detected an open (or unplugged) thermocouple. Configuring the internal jumpers wrong for a channel can give the same indication.

All module classes have an **Internal Functions** frame, located in the lower left corner of the full function **Run** form. It contains one of three different miscellaneous module functions that can be executed by simply clicking a control. The content of this frame varies with the module's model number.



Typical for 902x



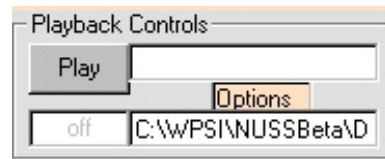
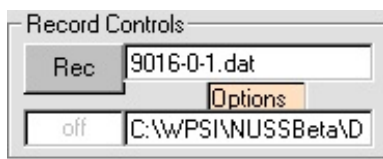
Typical for 9x16



Typical for 903x

The first frame example shows this frame for Model 9022 modules (which have no valves). The second frame example is for Model 9x16 Pressure Scanner modules (which have internal Calibration valves). However, if you click its frame label (**Valve Controls**) it also displays another frame (as shown in the first example). Finally, the last frame example is for any Model 903x Calibrator/Standard modules (which have no valves – but do have digital outputs that could conceivably be interfaced to external valves as needed). Both the *normal* (default) and *hidden* frames of controls are alternately displayed when you click the *frame's title*. More detailed information on these frames is found in **Section 6.1.10**.

Similarly, the **Record Controls** frame hides the **Playback Controls** frame, until its *frame title* is clicked, in which case they swap places.

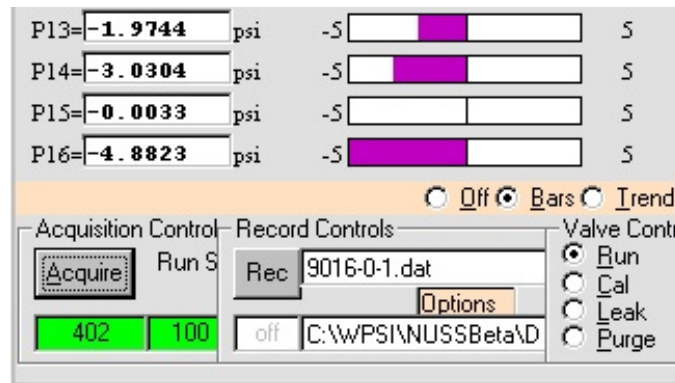


However, the **Playback Controls** frame refuses to appear if *live acquisition (Acquire)* is **On** for the module. Likewise, *Acquire* refuses to start again unless the **Record Controls** frame is visible (instead of the **Playback Controls** frame). Warning messages on the form's *bottom* status line indicate these restrictions if you try it (see example below).



Note warning message in bottom status bar

There is no alternate hidden frame hiding behind the **Acquisition Controls** frame, but if you click its *frame title*, you do get the pop-up **Run State** menu (just as you do by clicking the **Run State** label or its *text box*). Having so many ways to pop up this form might seem unnecessary — but the clicking of the frame is useful when the *label* and *text box* are hidden due to the form's width being reduced (and overlapped by the **Record Controls** frame) on an extreme *custom Run* form).



Scrunched control frames on a width-reduced form

6.1.7 Acquisition Controls (Details)

Refer to both **Acquisition Controls** frame examples below during the following discussion. This frame is located in the *bottom left* corner of a *maximum Run* form, and contains an **[Acquire]** button, a **Run State** display/select field, and three *stream activity* fields (which also serve as “hidden” **Acquire Pause/Resume** controls if you click them).



..showing *Acquire On*



..showing *Acquire Off*

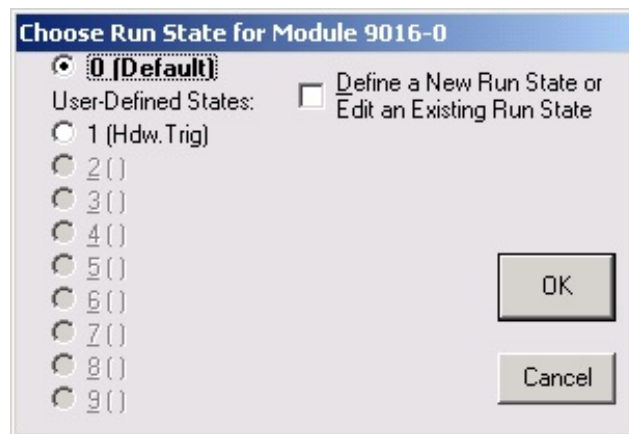
6.1.7.1 Acquire On/Off/Pause/Resume Controls

Acquisition for a module starts automatically if its current **Run State** so dictates. However, a new user-specified **Run State** may be configured for the module that starts **Run** with *acquisition* Off. When *acquisition* is turned Off, a status message “Press [Acquire] to START module” appears on the *bottom* Status Line just above this frame (see *salmon* colored status line in second example above).

Three *stream activity* fields, underneath the **[Acquire]** button, indicate when any of three autonomous streams are *actively being acquired live* by the module. The number displayed in each field shows the *sequence number* of its particular stream (streams 1-3, left-to-right) received while *acquisition* is active. The backgrounds of these fields are *bright green* when *acquisition* is On, *bright yellow* when it is Paused, and *white* when it is Off. On and Off states are obtained by alternate clicks of the **[Acquire]** button. The *Paused* state is obtained by clicking any one of the *stream activity* fields (which freezes the *sequence numbers* displayed, and blinks the value). Clicking any one of them again *Resumes acquisition*. The only difference between the *Off* and *Paused* state, is that the *sequence numbers* are reset (to 0 or 1) when *acquisition* is turned On after being in an *Off* or *Paused* state. These fields continue counting (not reset) after a *Pause* is simply *Resumed*. Natural streams start with *sequence* # = 1. NUSS starts all streams at *sequence* # = 0 if module's **Run State** is configured to have a special “prefix” scan at beginning of *acquisition* (this feature described later). A particular single stream indicator field may change to a white background (with its last *sequence* # frozen) to indicate the *natural end* of a *limited* stream. You cannot restart such *limited* streams (once they have ended) unless you restart the overall acquisition of all streams (with **[Acquire]** button).

6.1.7.2 Run State Change Controls

When a particular module is operated in a **Run** form for the first time, a **Run State** field (to the right of [Acquire] button) contains the number zero (0), indicating that the “*default*” **Run State** is assigned to it. Later, other *user-defined Run States* (1-9), created by the **Run State Editor** (see Section 6.3), may be configured and assigned to the module. Assignment of a *new Run State* (by clicking this field, its label, or the **Acquisition** frame title with the mouse) causes a pop-up form to be displayed with all that module’s **Run State** choices on it (see example below). If no user-defined **Run States** have ever been defined for that module, then only the “*default*” **Run State 0** is available for assignment. In this example **Run State 1 (Faster)** has been defined and is available for selection. Click [OK] to exit with your new selection, or click [Cancel] to ignore any selection change you might have made.



Run State selection form for a module

A *check box* may be clicked to request the **Run State Editor** from this form. When you press [OK], and this box is clicked, a much larger *editor* form pops up (dismissing the form above), and allows any new **Run States** to be defined for the module or any existing ones to be edited. Note, however, that upon returning from that editor, whatever **Run State** was *created* or *edited* by it has *not* been *assigned* to the module. You must click the **Run State** field again (to obtain above form) and then select that new **Run State**. See Section 6.3 for the details of using this *editor* form.

Changing the **Run State** on the **Run** form causes the module to stop all current acquisition functions (per the previous **Run State**), and reconfigure itself (per the new **Run State**). Acquisition then restarts (if new **Run State** so dictates) using possibly new *autonomous stream* formats (with new speeds, triggers, and periods) and with other module Acquisition options changed (e.g., # A/D Scans Averaged). NUSS remembers the **Run State** last assigned to the module when **Run** exits. It optionally stops or starts acquisition (and even

data recording) per that **Run State's Start** parameters the next time **Run** is started for the same module.

A **Run State** is nothing more than a special configuration file (or *internal script*) that, when assigned to a module (for use in a **Run** form), makes the module use any of the capabilities of its *autonomous streams* scanning feature (and other options) that were recorded in that file. The **Run State Editor** allows a user to define each **Run State** with simple point-and-click operations, without having to know any of the low level commands contained in the underlying script. This editor can also be obtained via a **Configure** submenu selection on the module's *context* menu, or on the **home-base** menu. Details of using the **Run State Editor** are deferred to **Section 6.3**.

Selection of any **Run State** for a module also associates a unique **Display Set** for the module. However, the **Display Set** is not configured with the **Run State Editor**, but by *interactive editing features* built into each **Run** form. Exiting the **Run** form not only remembers the **Run State**, but also remembers the user's current **Display Set** parameters in **its own special file**. Details of what constitutes the **Display Set** and how its is configured is described below (in **Section 6.1.7.3**).

Each **Run State** may have a verbose *functional label* (16 characters) assigned to it (by the **Run State Editor**), in addition to its inherent short numeric label (0-9). This *functional label* is the fixed string "(Default)" for fixed default **Run State 0**. The module's current **Run State** number is not only displayed in the **Run State** field on the **Run** form, but is also displayed as a text message (including its *functional label* in parentheses) on the Top (*salmon-colored*) Status Line (located between the **Title Bar** of each **Run** form and the first channel's data display line).

A Bottom (*salmon-colored*) Status Line (immediately following the last channel's data display line) shows other *event* messages and *error* messages. The right end of this line also contains the *Graphic Display* selection options. Its left end also displays any *attention* messages that require operator action. Such messages are often accompanied by a "beep" sound and the "blinking" of its text color (red-to-black-to-red...).

When the "default" fixed **Run State** (0) is assigned to a **Run form**, its module (and underlying NUSS **Run** task) always begins *default (relatively slow) periodic scans* of three (3) *autonomous streams*. It then repetitively *updates its data displays* as it acquires and processes these data streams *continuously* from the module. The default *primary* stream (1) is acquired every half ($\frac{1}{2}$) second, and contains EU *pressure* data only (*volts* appear when a Model 9400 transducer is not installed for any channel of a 902x module). A *secondary* stream (2), acquired every 2 seconds, contains *pressure* data in *counts* and *volts* (just in case any "other" tabular datum field is ever configured to display it). This secondary stream also

acquires any **Valve** status and any **Temperature Alarm** status being scanned by the module's internal firmware. A *tertiary* stream (3), acquired every 7.5 seconds, contains *temperature* data in °C, *counts*, and *volts* (for display as needed).

All three streams contain sequential samples of every channel of the module. When acquisition begins (with this Run State 0) the module also makes a single *prefix* scan of *each stream*, before starting the *repetitive* streams described above. This insures that any displayed “other” datum fields get refreshed *immediately* (instead of waiting as much as 7.5 seconds for its first datum to appear).

This “default” **Run State** (0) also associates a single *default fixed* **Display Set** consisting of *all input channels* of the module. A **Display Set** associated with a user-defined **Run State** (1-9) may specify only a subset of the module's input channels. For all module classes, the engineering-unit (EU) *pressure* data from the primary stream is displayed in both *tabular* (numerical) and *graphic* (bar or trend graph) form. The *precision* of values displayed in the tabular field may be changed by clicking them (more on this interactive **Display Set** configuration capability in **Section 6.1.7.3**).

An “other” datum, displayed as EU temperature (°C) by default, may also be shown for each channel's data line (in tabular form only) — if “other” fields are visible in the **Run** form. Each “other” datum has the same *precision*, *name*, and *units* reconfiguration features, if you click its datum field or labels.

Using *autonomous host streams* makes the NUSS host programs run more efficiently, in that they do not have to continuously send commands to the module to keep it going. Instead, the module, once its acquisition is started (per a defined **Run State**), generates the data streams automatically and continuously (unless a user-defined **Run State** limits them), and the NUSS host forms only need to receive and process the data stream records as they arrive. Higher overall NUSS throughput is possible (especially when operating multiple modules concurrently). Maximum throughput is only achieved when *hardware triggered* streams are defined in a user-defined **Run State**. Such *coordinated acquisition* requires averaged A/D scanning (inside the module) being synchronized with the emission of the streams by the module to the host.

After choosing a new user-defined **Run State** in a **Run** form, data may no longer be generated at a “leisurely pace” (as for the “default” fixed *Run State* 0). Depending on the speed of your particular NUSS host computer and its display card, the **Run** task may even have difficulty keeping up with every acquired stream. If it cannot keep up with all received module data streams, some of them are discarded (though some or all of these might possibly be recorded if the “higher priority” **Record** feature is On).

Any user-defined *Run State* (1-9) may be configured to *Start* or *End* as follows:

- (1) Automatically turn *On or Off* or *Don't Change (default)* the **Acquire** state when that *Run State* is selected to start;
- (2) Automatically turn *On or Off* or *Don't Change (default)* the **Record** state when that *Run State* is selected to start;
- (3) Specify another *Run State* to use (chain to) when the current one ends a *limited* run of all defined streams (or its acquisition is manually stopped).

Run State 0 always begins with *acquisition* On, but does not change the *recording* state set by any previous **Run State**. If there was no previous **Run State** (i.e. the module starts **Run** form with **Run State 0**) then the **Record** feature is always Off.

These extra options of the **Configure/Run-State** menu function (i.e., the **Run State Editor**) are chosen normally only when they make sense. That is, when a *limited* number of scans is specified (for any of the three concurrent streams) — or hardware triggering is being used to actually control acquisition in the module (for all the concurrent streams). In the latter case (hardware trigger) clicking the [**Acquire**] button (or automatic start) simply “arms” or “enables” the hardware trigger capability in the module, and the hardware trigger itself actually controls delivery of *limited* or *continuous* acquired data stream(s) to the host. The hardware trigger must be user-supplied (e.g., via a hard-wired control panel switch or some other synchronization signal generated in the test environment). Users should **not** select *hardware trigger* for any stream definition of a user-defined **Run State** unless they know that the necessary hardware is available. The result would be (what looks like) a failure of the module to acquire any data.

To assist the user in managing a plethora of *displayed* data on the **Run** form, each user-defined **Run State** (1-9) also maintains a unique copy of the **Display Set**. This **Display Set** is a file, saved uniquely with each particular **Run State** definition file, that remembers the current configuration of a module's particular input channels, and how they are displayed — as they were last *interactively configured* inside the **Run** form. Since each **Run State** has its own **Display Set**, changing the **Run State** also changes the **Display Set**. See next section for details of the **Display Set** configuration facilities.

Early versions of NUSS (1.0.4 and earlier) defined **Run State 0** so that it changed the *Scan-per-Average* count in the module to **8** by default. Later versions create **Run State 0** for *new* modules so that it does **not** change the *Scan-per-Average* count in the module. The parameter may be changed by the **Run State Editor**, even for **Run State 0**, with the value zero (0) meaning “do not change” *Scans-per-Average* when the Run State is initiated. Other values (1, 2, 4, ... 64, 128) may be assigned to effect such a change.

6.1.7.3 Interactive Display Set Configuration Controls

The following **Display Set** parameters can only be *configured interactively* in the operating **Run** form (i.e., there is no way to define it or change it from a module's **Configure** menu function). Most options are obtained by clicking a particular label or numerical field on each channel's data display line (see examples in **Section 6.1.6**). Use the pop-up *Tool Tips* to reveal this capability of each field. The following interactive configuration capabilities are provided:

- Assign a *logical name* to each EU pressure datum (e.g., "Psk" or "Tx2") instead of its *natural "channel" name* (e.g., "P1" or "T3");
- Change an EU pressure datum's *displayed precision* (maximum useful precision is 6 to 7 significant digits for any 32-bit internal IEEE float value);
- Change an EU pressure datum's *units* between the default "differential" (e.g., psig) and "absolute" units (e.g., psia). If that channel has been pre-associated with one or more *barometer* (i.e., absolute *standard*) modules, and that channel's natural pressure datum is differential, then the channel's EU datum is obtained by **adding** its value to the absolute value of the associated barometer, to yield an absolute displayed calculation result (displayed light yellow). If that channel's natural pressure datum is absolute, then the channel's EU datum is obtained by subtracting it from the absolute value of the associated barometer, to yield a differential result (displayed light blue). Lacking a barometer association, the fixed value 14.7 psi can be added to or subtracted from the channel's natural datum. Any *barometer* module(s) must be **actively acquiring data** (in a separate **Run** form) for this to work. Else, the original datum (not added or subtracted) is displayed with an orange (warning) color;
- Change any "other" datum's *source* between "reference temperature" (e.g., T1) or "junction temperature" (e.g., U1) and "pressure" (e.g., P1) or "primary temperature" (e.g., T1);
- Change any "other" datum's *units* between counts, volts, and EU. An EU pressure may even display "alternate" units than *natural* module units;
- Include (default) or disable the display of the selected graphic (*bar* or *trend* graph) form of each EU datum, and set various options for any *trend* graph.

The Model 9x46 has most of the same configuration functions described above, but cannot have its primary temperature datum appear in any other units, except for those configured

for the module for all temperature channels (°C or °F) as set by changing a calibration coefficient common to all channels..

Calibrator and *standard* modules may use most of these same *interactive configuration* features. However, the “units” of their displayed data are *unchangeable* (by interactive configuration means), thus always appearing per the *current nature* of their internal *standards*. In other words, an *absolute* unit only displays *absolute* data — except when operating in a *quasi-differential mode* (per its **QDif** check-box setting). A *differential* unit only displays *differential* data.

6.1.7.3.1 Changing E.U. Datum Content and Name

When you click in any of the **Run** form's datum fields (or labels) listed above, a pop-up form (per the several examples shown below) gives you choices you can select, for any of the features already described above. Such changes are then remembered (in an internal file) that defines the **Display Set** of the currently selected **Run State**. This **Display Set** is always saved (as currently configured) when the module exits a **Run** form. When the module restarts in the **Run** form, the last **Run State** it was previously using is restored, and the **Display Set** of that **Run State** is also restored.

The examples below shows the pop-up form obtained by clicking either the *unit* or *name* label of any EU datum (not the datum itself) for both *pressure* and *temperature* scanners. Any changes are stored persistently in the **Display Set** file of the current **Run State** saved when the **Run** form exits for this module — and is thus restored the next time this module uses the **Run** form. For *new* modules operated by NUSS, the first option is chosen by default. Any module can be made new again by deleting the internal (.ini) file that records its **Display Set** (see **Forget** function).

Pressure Scanner E.U. Datum options

Temperature scanner E.U. Datum options

For a pressure scanner, the “**Choose Alternate EU Units**” field (see bottom of left example above) appears only if you have selected the **Data in Alternate EU Units** option. Click the arrow in this *list box* to reveal a list of many “*alternate*” EU *pressure* units to use in displaying the selected EU datum (the current native units are selected by default). Click another list item desired to change this channel's displayed datum to appear in some other non-native units. If that datum is also visible in graphic form (bars or trend graph) then its Low/High scale points are also converted to the new “*alternate*” units (but these are usually rounded up to the nearest integer equivalent). These scale values could be rather large numbers for certain alternate EU pressure units, and this might cause “*crowding*” of this and adjacent fields in the displayed **Run** form.

If the pressure channel selected is associated with a barometer (LBN) – as indicated in the Title Bar of the above pop-up form (see left example), and as also indicated in the *ToolTip* of the *units name* field of the selected channel on the **Run** form – then an optional *checkbox*

also appears to allow *enabling* or *disabling* an optional *barometer calculation*. That checkbox is labeled **Add Baro** if the selected channel's *native pressure mode* is already *differential*. It is labeled **Subtr. Baro** (i.e., *Subtract*) if the selected channel's *native pressure mode* is *absolute* (as are some 940x external transducers that plug into 902x pressure scanner modules). An enabled *barometer calculation* yields a datum for this channel on the **Run** form in the “opposite” *pressure mode* than its selected channel had natively – and the graphic (bar or trend) range for this channel changes accordingly. The new graphic *absolute* range (for the **Add Baro** case) has a high end about 30 psi (or equivalent other native units) *larger* than the selected channels native range, and its low end changes to 0 pressure units. The new graphic *differential* range (for the **Subtr. Baro** case) has a high end about 30 psi (or equivalent other native units) *smaller*, and its low end changes to the *negative* of the same value (0 pressure units are in the center).

The Model 9x46 *temperature/resistance* scanner does not allow changing its primary E. U. datum to alternate units, since each channel can already be re-configured to show various *temperature* or *resistance* data.

The **Change Name** field displays the EU datum's current or default name. Click it to change the name – but keep name short if you want it to fit on the **Run** form.

6.1.7.3.2 Changing Other Datum Content and Name

A similar pop-up occurs if you click any “other” datum's *unit* or *name* label. The following examples illustrate differences for *pressure* and *temperature/resistance* scanners.

Other datum choices for *pressure* scanners

Other choices for *temperature* scanners

The “**Choose Units of EU Pressure**” field (see left example above) appears only on the pressure scanner version, if you have selected the **Pressure in EU** option for a pressure scanner. Click the arrow in this *list box* to reveal a list of many “*alternate*” EU units to use in displaying the selected datum. Click the list item desired to change the units. Even

though data inside the module may be acquired and returned in each stream in native units, selecting an “alternate” EU units for a datum causes its displayed units to be re-scaled to be displayed on the channel’s *Run* form datum field in “alternate” value in other selected units. There is no provision of also displaying an “other” datum in any graphical form.

If the pressure channel selected is associated with a barometer (LBN) – then an optional *checkbox* also appears to allow *enabling* or *disabling* an optional *barometer calculation*. For more information, see the description of this option in the previous section (6.1.7.3.1).

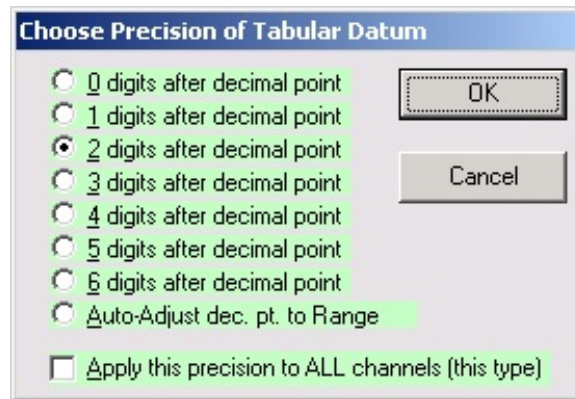
The **Change Name** field displays the datum’s current or default transducer name. Click it to change the name – but keep name short if you want it to fit on the **Run** form.

The other forms of temperature and pressure data you may select above (by clicking one of the top 5 options), are data types that are all natural constituents of the three-streams generated by the default **Run State 0**. However, if you have any of them specified for display by the module, and then you change the **Run State** of the module to a user-defined one in which the streams no longer contain such “other” forms of data, then these data fields *remains blank* as long as that **Run State** is used by the module. However, since a new **Display Set** changes along with a change of the module’s **Run State**, you have the opportunity to configure that new Display set appropriately to the contents of its accompanying Run State. Normally, a user-defined **Display Set** can *inherit* the currently configured data display features of another Display Set that is associated with another existing specified **Run State**.

The above forms for a Model 9x46 temperature/resistance module are slightly different – mostly in the label text. Any “pressure” item is changed into a temperature item for the primary data channels. Any compensation temperature item for secondary data channels is changed into UTR Junction temperatures and other references.

6.1.7.3.3 Changing Datum Precision

This example shows the pop-up form obtained by clicking the *value* field of an EU Datum. The same pop-up appears if you click the “other” datum. Any changes you make appear in the **Display Set** file associated with the current **Run State** that is saved when the **Run** form exits for this module.



Form to change Precision of datum field

The current *precision* (digits after decimal point) is shown (as selected) on this form when it is first displayed. You may then click any other *precision*, and press **[OK]**. The **[Cancel]** button may be used instead if you make a change and then decide NOT to use it.

Note that the last precision choice lets you select a value “suitable” for the particular full scale range of the transducer of the selected channel. This is particularly useful for Model 9021, 9022, or 9x46 modules where each channel is likely to have a different range. You can apply this option to all channels, and each channel has an appropriate precision..

The selection you make only affects the particular datum you clicked — unless you also click the *check box* requesting that your selection be applied to ALL channels of the same datum type. If dissimilar datum types are displayed for certain “other” datum channels (e.g, some in counts, volts, or °C), then only those channels (that have the same type (i.e., units) as the one you clicked) is also affected by this *check box* option.

6.1.7.3.4 Changing Trend Graph Parameters

The example (next page) shows the pop-up form obtained by clicking the graphic area of a displayed Trend Graph. It allows you to reconfigure the graph in many ways. Any changes you make appear in the **Display Set** file associated with the current **Run State** that is saved when the **Run** form exits for this module.

Display/Edit Trend Graph Parameters

☐ Clear Trend Graph Only

30 Time Span in Seconds (top to bot)

40 Plot Symbol Size (0-255)

16 # Points Skipped/Symbol (0-255)

Action on Auto Wrap

☐ Clear Screen ☒ Overwrite Old Data @ cursor

Background Color

☐ White ☐ Lt.Gray ☒ Dk.Gray ☐ Black

OK Cancel

Configuration form for Trend Graph

The **Clear Trend Graph Only** *check box* serves only to clear the current graph and start over again (after you press OK).

The **Time Span** field is useful in setting how much data is crammed onto the Trend graph as points are plotted on the time axis (time advances down the page).

Plot Symbol Size may need to be adjusted on some low-resolution displays to allow one to distinguish the **round** plotting symbols from the **square** ones, or if a large **Time Span** is selected that causes symbols to be plotted too close together.

You might also change **# Points Skipped/Symbol** when symbols overwrite each other. It should normally be set to some multiple of the number of channels displayed. Please note that a high resolution XGA display (at least 1024 X 768 with at least 16-bit or 24-bit color) may be required, before you can distinguish all the colors and symbols for 18 channels. Lacking this, it may be necessary to boost Plot Symbol Size.

The **Action on Auto Wrap** frame contains two mutually-exclusive options. The first (**Clear Screen**) causes the trend graph to erase (entirely) when the plotting symbols reach the bottom of the graph. New data then starts plotting at the top of the graph. The second (**Overwrite Old Data @ Cursor**) causes only a narrow band of the trend graph surface to be erased *forward (down the page)* of where plotting symbols are written on the graph. The leading edge of this *erase band* is indicated by a *white* horizontal line cursor that also moves down the page as new symbols are plotted. This method has the added advantage of leaving as much “historical” data on the plot as possible at all times.

The **Background Color** frame contains four (4) mutually-exclusive options for selecting the background color of the Trend Graph. The rather dull “colors” are chosen to insure a suitable contrast with all the *plotting symbol colors* (assigned automatically) for specific channels of data.

6.1.7.3.5 Bar Graph Parameters

Bar Graphs have no selectable options. Thus, clicking them does not bring up any configuration form. They always scale themselves (both Low and High ends) automatically per the closest *integer* value to their true natural transducer range. Some ranges (e.g., 2.5 psi and below) keep a fractional value for Bar Graph labeling. These ranges are based on the *Range Code* of each DH transducer channel (for pressure scanners) and the *Sensor Type* configured for each channel (for temperature/resistance scanners). In each case these range data come from a text file in the **Ini** subfolder. For Model 9x46 *temperature/resistance* scanners only, the range data is somewhat arbitrary – and you may edit the file (SensorType9046.txt) to change the range of each Sensor Type – and thus the scaling and end labels of bar graphs that display them.

If an “alternate” unit is selected (either for single datum field(s) with NUSS, or by changing the units of all the channels of the module, inside the module itself), the bar graph scales adjust to the closest integer range (rounded up) that results by multiplying the natural range by the new units conversion factor.

When a natural *differential* channel is chosen to be displayed in *absolute* units (causing a particular barometer’s value to be added to it) then the otherwise *bipolar* scales are converted to a *unipolar absolute* scale (Low=0), and the bar’s color changes to *gold* (just as it does for *absolute calibrators*) The High range is then automatically changed to an absolute value (per the barometer used) which is the closest integer range (rounded up) to the sum of the natural range plus 15.0 psi (or equivalent other units).

Differential (bi-polar) data is always displayed on Bar Graphs with *dark-cyan-colored* bars (if zero or above) or *dark-magenta-colored* bars (if less than zero).

6.1.7.3.6 Persistence of Display Set Options and ‘Forget’ Functions

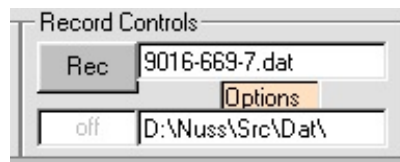
All of the above display characteristics of a Run form for a particular module and each of its defined Run States are remembered in special “historical” files that are saved whenever a Run form is exited. This lets NUSS restore that modules display information whenever you execute its particular **Run** form again later.

This feature is very useful for most scanners with fixed transducers. However, for the Model 902x pressure scanner modules and the Model 9x46 temperature/resistance scanner modules, which have easily re-configurable transducers on a per channel basis, there is a down side. When the user changes the transducer physically or changes its firmware configuration in any way, the Run forms don’t seem to notice – because of these historical files. What is required, is that after changing any transducer it is necessary to delete the historical files – so that NUSS restores them to defaults based on the “new” hardware/firmware configuration of all its channels.

Later versions of NUSS have a ‘**Forget**’ function for each module (available from module’s context menu or from a ‘group’ menu on the Network Status form) which deletes all the **Display Sets** (“historical” files holding display configuration data) for each defined Run State of that module (or current group of modules). This causes a module’s **Run** form to subsequently recognize each “new” transducer that may have been configured. Of course, any custom **Run States**, **Run** form sizes, and **Run** form positions on the screen – must be re-specified after executing ‘**Forget**’. Likewise, any per-channel data precision or data source changes (part of each Run State) must also be re-specified.

6.1.8 Record Controls (Details)

A **Record Controls** frame is shown below. It is located in the *lower center* of a *maximum Run* form. By clicking this frame's title you may alternately swap between the **Record Controls** frame (shown) or the **Playback Controls** frame (illustrated in next section). If the **Options** label is clicked an *Option Selection* form pops-up appropriate to the particular **Rec** or **Play** frame selected. The **Playback** frame can only be selected if live *acquisition* is Off. *Acquisition* does not restart until the **Record Controls** frame is again selected.



..showing *Record Off*

The **Record** feature writes, to a disk data file, all configured streams acquired “live” from the module after the **[Rec]** button is pressed. Recorded stream data are written in a *binary* (32-bit single IEEE float samples, little endian) format, even if they are acquired in some other *decimal* or *hexadecimal* format. Time stamps and other useful header information are added by NUSS to these recorded streams. If any of the module's transducers has a *barometer* (i.e., Logical Barometer Number (LBN)) associated with it, then a special barometer stream is *synthesized* by NUSS and also written to the recorded data file at regular intervals. Special *event* records may be recorded, either manually (by clicking the **Options** label when a file is being recorded) or automatically (to note certain time discontinuities, such as **Record** or **Acquire** pauses, that NUSS detects on its own).

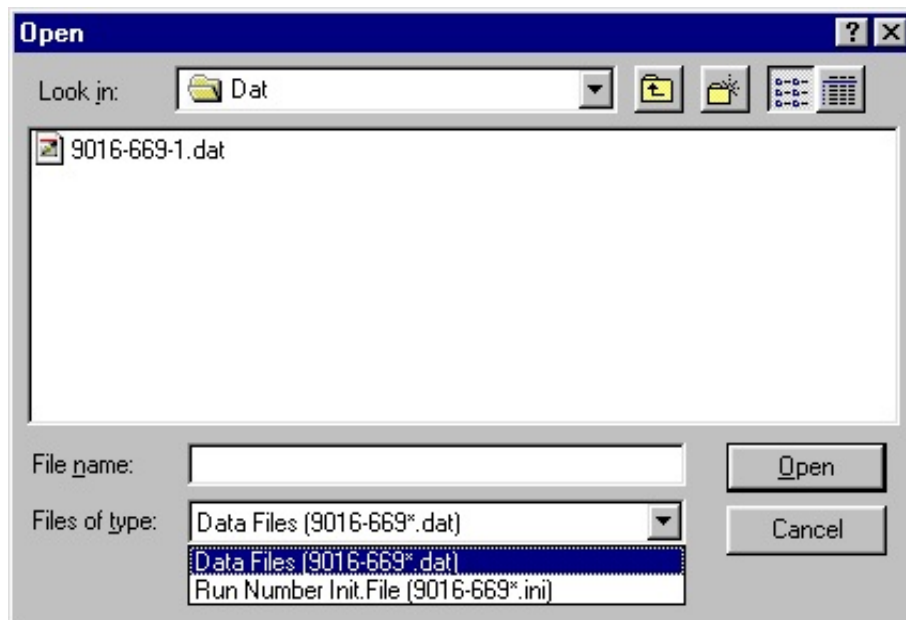
Normally, both **Record** and **Playback** features are disabled, and the **Record Controls** frame displayed only, when a **Run** form is started. However, when a new (or the current) **Run State** starts in the module, it may specify if the **Record** feature is to be turned On, turned Off, or be *unchanged* (i.e., remain in state it was in during its operation in the previous **Run State**). The default **Run State** (0) always starts live data acquisition immediately but does *not* change the *recording* state.

6.1.8.1 Record (and Playback) Data Files

Record and **Playback** feature frames are identical in appearance (see **Record** example on previous page). Each has two unlabeled text boxes displaying a *current file name* (top) and a *current path name* (bottom). A **recorded** file name is set *automatically* by default, using the following general name format:

“<modid>-<testrunnumber>.dat”
(e.g., “9016-141-1234.dat”).

These data files are normally located in a *DAT subfolder* of the *base* folder where NUSS was installed. The suffix “-<testrunnumber>” identifies a unique “test run number” (or version). For the **recorded** file, it is obtained from a special **Test Run Number** (“<modid>**TRN.INI**”) file maintained in the same folder with the data files. Manually changing the file name is possible, **but not recommended**, as it might conflict with the convenient automatic naming feature. Changing or choosing a file or path names is as simple as clicking the *filename* or *pathname* fields. The default path is recommended when maximum data acquisition rates and recording rates are required — as your hard disk is likely to be more capable of faster live recording throughput than most network drives.



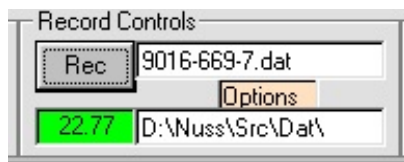
Typical Record/Playback File Dialog Box

Clicking either the *filename* or *pathname* field reveals a pop-up dialog box (shown above) which allows you to navigate to any local or network drive accessible on your system. Note the two filters (in **File of Types** list box). One shows all the data files for the module. The

other shows all the Run Number **.INI** files for the module. The latter should not normally be assigned to the **Record** file name (by pressing [Open]) — though such a file may be highlighted and then right-clicked (to perform file maintenance functions (copy, delete, etc.) one file at a time) per the standard Windows *context* menu.

6.1.8.2 Record On/Off/Pause/Resume Controls

Anytime the **Record Controls** frame is displayed the **Record** feature may be turned *On* or *Off* by depressing the [**Rec**] button. When turned *On*, the current **Record** file name is opened. If this file already exists new data overwrites old data. However, the automatic file naming feature prevents this from happening. Special header records are written immediately at the beginning of this file to “save” the module’s current **Run State** and **Display Set** parameters (needed by playback to know the format of each recorded data stream and each displayed field). The actual real *time* and *date* when the file is opened is also recorded in these header records. The *time of file opening* is used as a *reference base* for *relative time stamps* which are added later to each recorded *data stream* (if **Acquire** is **On** or is eventually turned on).



..showing **Record On**

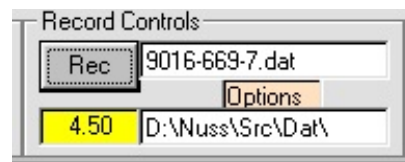
If **acquisition** is running, all *subsequently acquired* data streams from the subject module is written as special binary *data stream* records, until “acquisition stops” (or is stopped manually) or the [**Rec**] button is pressed again to stop **recording**. Each of these *data stream* records contains a *relative time stamp* (described above). To indicate *data stream* recording activity, this *relative time* is initially displayed (on a *bright green* background) in *fractional seconds* in a field just below the [**Rec**] button (see example at time 22.77 sec. above). As the *relative time* value grows (while the **Record** file remains open), a colon is added to display *minutes:seconds* (e.g., 59:59) and the fraction is dropped to save space. Eventually, a second colon is added to display *hours:minutes:seconds* (e.g. 23:59:59) if the recording goes on that long.

If one or more *barometer* modules are associated with any channels of a subject *scanner* module (i.e., used to transform the “differential” scanner channel’s data into “absolute” pressure units), and these barometers are actively running, then special “synthesized” *barometer data stream* records (identified in the file as a special *data stream 4*) are also written to the open file, just after the starting header records, and written every one second

thereafter while the file remains open. When these extra barometer streams are being recorded, the *relative time* field is displayed in **red** text (instead of normal **black** text), but the background remains *bright green*.

If **acquisition** is not running when the **Record** file is initially opened, a special *event* record is written immediately (following the starting header records). This event record indicates (during playback) the *time discontinuity* that exists (between the file header's *base start time* and the first actual data stream (with a relative time stamp) that is eventually recorded — when/if acquisition is finally started. All *event* records also have the current *real time* and *date* recorded in them, just like the starting header records.

A **Pause** feature for *recording* may be activated by clicking the *relative time* indicator field below the **[Rec]** button while recording is On. This stops any active recording of acquired module *data* (and *barometer*) *streams*, thus the *relative time* field stops incrementing (and blinks). Also the field's background turns *bright yellow* instead of *bright green* (see example at time 4.50 sec. below). Clicking **Pause** causes a “start pause” *time discontinuity event* record to also be written to the recorded data file (along with its real time and date).



..showing **Record Paused**

Clicking this *relative time* field again defeats the **Pause** and writes a “end pause” *time discontinuity event* record. All subsequently acquired live *data streams* from the module (and any *barometers*) are again recorded to the file. The *relative record time* field again increments (for each stream recorded) and the background turns *bright green*.

Once a *recording* is started, the **Run** task attempts to *record* as much of the acquired “live” *data streams* as possible, but does not necessarily keep up with the maximum rate the module is delivering such *data streams* to the host. Thus, like the task's display update feature, recording is also a “quick-look” feature, whose capabilities are mostly determined by the speed of your host PC (and network) facilities. However, **recording** is *prioritized* so that it takes precedence over *display update* when the **Run** function is trying to do both while receiving data streams from the module at very high rates. In other words, an attempt to *record* is always done first for each acquired data stream record that has arrived in the host. If there is no indication of data backup in queues, any displayed data is also updated. If queues are backing up, some streams do not update the display, and even some fast *data streams* may not be recorded. These restrictions rarely occur unless a user-defined **Run**

State is being used which has specified unusually fast scanning rates (short stream repetition periods).

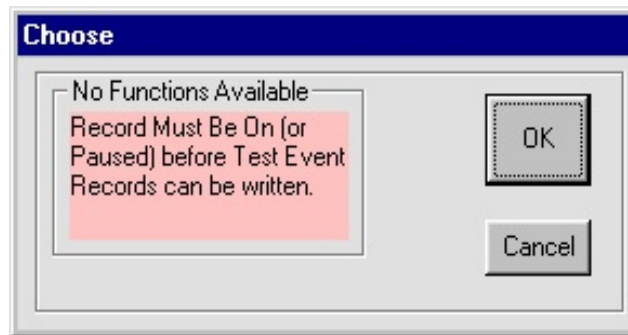
After you turn **recording Off** by clicking the **[Rec]** button (and therefore close the file), a new “unique” file name (using the “next” *<testrunnumber>*) is displayed in the *filename* field and is immediately ready to be opened as soon as you press the **[Rec]** button again.

Later, you may rename any existing recorded *files* or move them out of the base DAT subfolder for archiving purposes. If you do so, you also need to **delete** the special **.INI** file for each module that maintains the next *<testrunnumber>* used for automatic file naming of the module’s data files (i.e., *<modid>TRN.ini*). This insures that this *<testrunnumber>* restarts at 1 again for each module affected. Since when you click the *filename* field, the common dialog box that pops-up has selectable filters that allow either the **.dat** files or these special **.ini** files to be displayed in the form. Thus, you can delete the **.ini** files (one at a time only) within NUSS without having to resort to **My Computer** or **Windows Explorer**. However, if you are already using these more flexible standard Windows applications to do file maintenance, you might as well use them to delete the *<modid>TRN.ini* files too. They have the advantage of being able to process multiple files in selected groups (using the *<Shift>* and *<Ctrl>* keys during mouse selection).

Changing the *current Run State* (and thus the **Display Set**) of a module is permitted, **while a Record file is open**. However, when this happens, a new set of “header” records (similar to the initial header records written at the start of a file) are written in the middle of the file. These remember all the parameters of the new **Run State** (and its **Display Set**) to aid **playback** in processing the possibly redefined data streams subsequently written to the remainder of the file. This is not a “bumpless” process however. It takes a finite amount of time to (1) stop the old **Run State** in the module, (2) restart it for the new **Run State**, (3) resize the **Run** form’s display area for any change of the channels acquired (and any other **Display Set** change) and then (4) write the new mid-file header records. Some time gaps occur in the recorded data streams during such a transition (from old **Run State** to new **Run State**) .

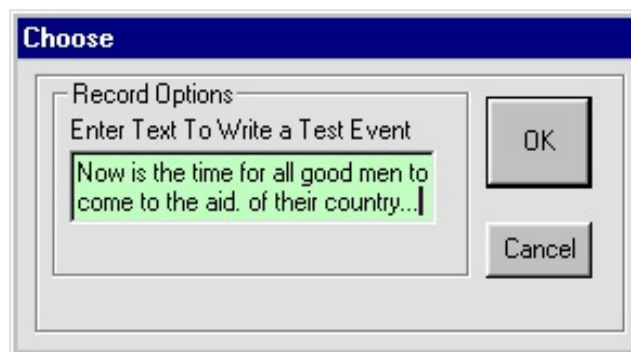
6.1.8.3 Record Options

There are currently no special **Record** options to set when **Record** is Off. If you click the **Option** label the following unsatisfying form pops-up.



Empty **Record Options** form when **Record Off**

However, if **Record** is On, clicking the **Option** label pops-up the following form, to allow you to enter any comment as a *Manual* event. This event is written to the data file only when you press the [OK] button after you finish entering the event's text. During the time you are actually typing the event message into its box, data recording is continuing to occur. You could, of course, *Pause Record* before attempting to write a *Manual* event. However, that would actually write three (3) events sequentially onto the recording file: The *Pause Begin* event, the *Manual* event, and the *Pause End* event.



..showing entry of Test Event text

6.1.9 Playback Controls (Details)

Anytime *Acquire* is Off, the **Playback Controls** frame (see example below) may be displayed instead of the **Record Controls** frame, by clicking the frame's title. Then, the *Playback* feature may be *started* or *stopped* by depressing the **[Play]** button (which replaces the **[Rec]** button).



..showing *Playback Off*

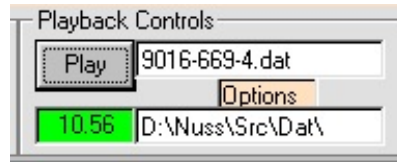
6.1.9.1 Playback Data Files

When *Playback* is active, any recent previously-recorded *record file name*, normally becomes the default *playback file name*. Thus, in switching from **Record** to **Playback**, you may immediately *play back the most recently recorded file* — or you may choose any other pre-recorded file from the DAT folder list (obtained by clicking the *filename* field). If the *filename* field is blank (as in above example), you *must* click it to choose a file before playback can begin. A standard Windows **Open common dialog** then pops-up to allow you to pick a new playback file name in the DAT subfolder (or any subfolder you change it to). See Section 6.1.8.1 for more information. It covers the subject of Record **and** Playback Data Files in one place.

6.1.9.2 Playback On/Off/Pause/Resume Controls

The *Playback* feature (available when *Acquire* is Off) reads the data (**.dat**) files written by **Record**. When the **[Play]** button is pressed, it displays the pre-recorded data it reads from the file, and displays it directly on the **Run** form's *tabular* and *graphic* screen displays — just as when that data were being acquired live. Clicking the **Option** label, before pressing **[Play]**, reveals a dialog that selects readable text reports or spreadsheets, called secondary files, which are generated while data are being played back. See **Section 6.1.9.4**. A Secondary file (with **.txt** or **.csv** extension) can be generated from the recorded data file — during the time *Playback* is reading the pre-recorded data from the file. Many such Secondary files can be created from a single playback file. Text (**.txt**) files are actually readable text Reports. Each may contain all (or only selected) time-regions or all (or only selected) streams of the original **.dat** file. The **.csv** secondary files contain the same data, but formatted so that it may be easily processed, further, by a spreadsheet application (e.g., Microsoft Excel).

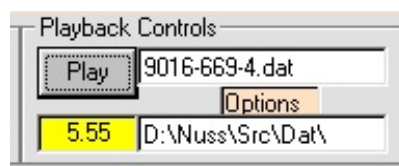
Once **playback** is started, the current *playback file name* is opened at the beginning (i.e., it is “rewound” in a sequential data file sense). Unless you change the default **Playback** editing features to specify new Secondary files to be written (by clicking the **Options box**) any data played back is simply displayed in the **Run** form’s data displays — per the **Display Set** of the **Run** form that existed *when the file was recorded*. Just as during recording, the *activity field* beneath the **[Play]** button turns *bright-green* and shows the *relative time* of each data stream played back (see 10.56 sec. in example below).



..showing **Playback On**

The special header records, recorded at the beginning of this pre-recorded data file, have all the necessary **Run State** and **Display Set** information to allow **Playback** to reformat the display as necessary. If you are surprised by the display size and formats changing when you start **Playback**, this is why. If one or more **Run State** changes occurred during the original recording of this file, other *mid-file header records* may be encountered during playback. These also causes the display to be noticeably *reformatted* as necessary. Such a format change also occurs when playback is done, and you switch the **Playback Controls** frame back to its **Record Controls** format (which restores the live acquisition **Run State** and **Display Set**).

Playback also has a **Pause** feature, just like **Record**, that allows it to be stopped and started without losing the file’s current position. This is accomplished by clicking the *Relative Time* indicator field under the **[Play]** button, while **Playback** is On, which changes the field’s background color to *bright-yellow*, and blinks any current *relative time* displayed in the field (see 5.55 sec. in example below). Clicking the field again *resumes Playback* and change the background color back to *bright-green* (if there is any more data to playback).



..showing **Playback Paused**

6.1.9.3 Playback Auto-Pause

The **Pause** feature is *automatically activated* when any *event* record is encountered during playback of the data file. In addition, a description of the event (including the real *time* and *date* when it was recorded) is written to the *Bottom Status Line* (see example below).

Otherwise, an **auto-Pause** looks just like a **manual Pause** initiated by the user. A similar *event* (with different text) is shown when the user resumes playback, and when the playback file ends. To continue playing back the *remaining* data (recorded in the file beyond the encountered event), simply click the *Relative Time* indicator again, just as you would to **Resume** a manual **Pause**. An **Option** box dialog option can defeat **auto-Pause** if it becomes annoying, but the default is for it to be enabled when a new playback file is chosen.



..showing **Playback** in *Auto-Pause* due to encountering event

6.1.9.4 Playback Options

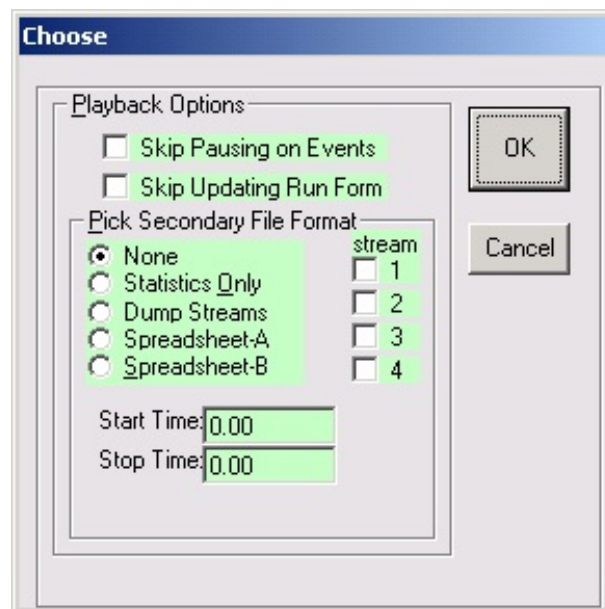
Playback reads and processes any pre-recorded data file (i.e., any with the **.dat** extension) sequentially *record by record*. Since such a file contains *binary* data stream records, a simple *text editor* is unsuitable for reading and displaying it. **Playback** has a *primary* output function that simply displays data streams directly on the **Run** form's data displays, via the **Run State** (and **Display Set**) that is recorded in the header records at the beginning (or middle) of the file. Such data appears just as it did during "live" data acquisition, except that the *rate of display update* may be faster than real time.

Additionally, **Playback** can convert the originally recorded data (**.dat** file) into one or more *readable text reports* (**.txt** files) or *spreadsheets* (**.csv** files) by clicking the **Option box** before you click **[Play]** (see pop-up dialog that results below). These new data are written "silently" (behind the scenes) as *secondary data files* during the time playback is displaying its data on the **Run** form. Many such files may be created from the same recorded data file, by playing it back multiple times, each time specifying a different *relative time* range and/or *stream* content. These files have the same name as their source **.dat** file, but with the string *<secid>* appended to it. See the end of this section for more explicit examples of secondary output file names.

The following Secondary Output File *report* formats are possible:

- None, or
- A Statistics Only format, (Events and Stream Counts between them only) or
- Dumping the recorded data streams into a “readable decimal text” *dump* format with each data group of a stream on a separate line (also called *block* format), or
- Reformatting the data streams into a “spreadsheet exchange” (comma-separated variable) format, in a multiple row (*block*) format (Type A) or in single row (Type B) format.

Other options allow one to edit the original data by *relative time stamp* or *stream content*. Only data in the file between a selected Start Time and Stop Time are included in the secondary output file (set both times to 0.00 (default) to include *all* the data recorded in the data file). Only selected *stream numbers* need be checked for inclusion in the header and body of the output file. Numbers 1-3 are the real data streams from module, number 4 is a special Barometer stream (recorded only when a module has Barometer associations to convert differential data into absolute form). Failure to check any stream number results in a secondary output file *containing only events*.



Typical **Playback Options** dialog

The **Statistics Only** report format is used when you do not wish to see any actual data, but just want to know how many data records exist *for each stream*, and their *starting* and *ending* times, between all other recorded events. Events are always included in all report formats. This is a useful way to preview a long *data* file before deciding what sections to

process into other edited reports of the **Spreadsheet** or **Dump Streams** format. This type of report is named `<modid>-<run#><secid>.txt`.

The **Dump Streams** report format produces a simple text file report in a form that is easily human-readable using a simple text editor. Use it to print out all the data in a file, or print various ranges of the recorded file from specified beginning and ending times. This type of report is named `<modid>-<run#><secid>.txt`. Note that this format has the same naming convention as the **Statistics Only** format, but you can produce as many versions of each as you desire, as they all have different `<secid>` fields. This format prints each *data group* of a stream on separate text lines – sometimes called the stream *block* format.

The **Spreadsheet** report format is like the Dump format, except that its data values are in a “comma-separated” export form that is readily loaded into a spreadsheet application. This type of report is named `<modid>-<run#><secid>.csv` (for comma separated variables). The **Type A spreadsheet** is formatted exactly as the Dump (printable) format (i.e., each *data group* of each stream is written on successive data rows in a *block* format). The **Type B spreadsheet** prints all data fields for all data groups on a “wide” single row, making it easier to plot any datum versus time or versus any other datum – when the spreadsheet is processed by data reduction functions of the spreadsheet application (such as Excel).

The *name of each secondary file report* currently being written (or last written) always appears on the top status line of the **Run** form. Also note that there are two options (at the top of this **Playback Options** dialog) that is not related to report format or editing, but applies to **Playback** in general. The first, the **Skip Pausing on Events check box**, is checked if you want **Playback** to not **Auto-Pause** when it encounters events in the recorded data file. This is particularly useful when the same data file is played back multiple times in order to create several different secondary output files. By default, playback stops whenever each event is processed, and has to be manually resumed. The second, the **Skip Updating Run Forms check box**, is checked if you want **Playback** to generate its specified secondary file while reading the playback data, but NOT update the display windows with the data at the same time. This speeds up the secondary file generation process noticeably on a slower computer with a slower graphics card.

After creating one or more *secondary data files*, you may click the **Status Bar** just under the **home-base** menu in order to view them (or select the home-base **‘File | View Files (NUSS)’** menu item). This pops-up an **Open** dialog form that lets you navigate to the **Dat** subfolder. You can then view all the files in it. For a particular data file (e.g. **9016-012-123.dat**, which is Test Run Number 123 for Module 9016, serial # 012) you also see any *secondary data files* (with **.txt** and **.csv** extensions) that were created from that data file. Such files all begin with the same root name as their source data file, but have a `<secid>` field (e.g., **A-ZZZZ...**) added to the end (e.g., **9016-012-123A.txt** (the first readable stream dump report) or **9016-**

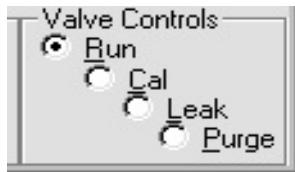
012-123B.csv (the next spreadsheet comma-separated variable import file)). These files are automatically named, using a special *<datafilename>***SID.ini** file created for each *data* file whenever the *first secondary* file is created from it. These files operate similar to the way recorded data files are automatically named using similar *<modid>***TRN.ini** files. See **Section 2.4** and **Appendix F** for more information.

Using the **Open** dialog above (or other more flexible Windows applications like **My Computer** or **Windows Explorer**), you may also send these *secondary data files* to a *printer*, or pop them into a Spreadsheet application for further processing, or copy them elsewhere on your available disk drives. However, if you delete all the data files or secondary output files from the **Dat** subfolder, you should also delete these **.ini** files too. This causes the automatic naming of data files to revert back to Test Run Number 1, and the automatic naming of secondary output files to revert back to Revision A.

6.1.10 Other Module Control Frames (Details)

The *Maximum Run form* has other “live” controls that permit you to control hardware features (in its subject module) or other NUSS features (on behalf of its module) as it operates. These are described fully in this section.

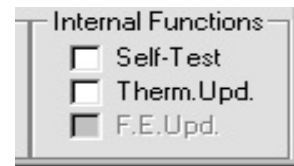
Any *pressure scanner* module (except a Model 9022) has *internal solenoid valves* which may be changed manually by clicking *radio buttons* in its **Valve Controls** frame (see first example below). This 4-state valve may change the module from its normal **Run** state to a **Cal** state, or select other **Purge** or **Leak-Check** functions. Starting any module’s Run form always forces the module into the **Run** state.



some scanners



all 903x modules



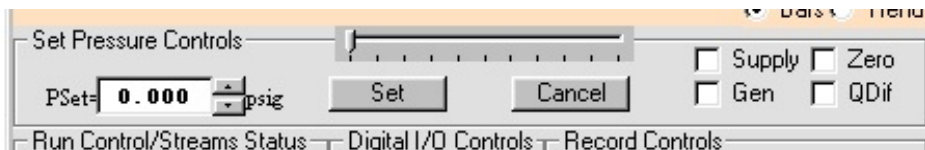
all modules

If *valve sensing* is also available for the module (e.g., Model 9816) then current *valve position* status is acquired periodically (in the secondary stream normally), and is indicated by the *text color* of the *valve labels*. A label’s text changes to a *dark yellow* (*brown* or *gold* on some computers) after its valve setting control is clicked — then changes to *dark green* after its correct state is sensed. Such label text is *black* normally, and remain that way if no sensing data are available (e.g., Model 9016). The original valve position is restored if sensing indicates failure to shift the valve’s position to a new state.

Model 903x *calibrator* and *standard* modules also have special internal valves (described below) but they also have *digital outputs* and *digital inputs* (i.e., presumably for *driving* and *sensing* external user-supplied solenoid valves). The four (4) digital outputs may be changed individually via four (4) *check-box* controls in a frame labeled **Digital I/O Controls** (see second example above). Four (4) digital inputs are displayed above them, as simple *lamps* (grey circles for “off”, red circles for “on”). A particular “input” *lamp* is not necessarily related to the “output” *check box* below it — though you should consider custom-wiring inputs and outputs so they are “related” in this way. Your reward would be a program already equipped to naturally show this relationship.

Every module also has **Internal Functions** (see third example above), which may be revealed by clicking the frame title (of the other examples above). It is the default visible frame for 9022 models. To execute a particular function, simply “check” the *check box* desired. The check mark remains visible *only for the duration of the test initiated* (some tests are almost instantaneous). Clicking this frame title again restores the original **Valve** or **Digital I/O** frame.

All Model 903x modules have a separate **Set Pressure Controls** frame (see example below). This allows you to make a *calibrator* module *manually generate* any suitable *output pressure* at any time. A new *tabular pressure set point* is entered directly into the field labeled “PSet=” (by clicking it to obtain a *text entry cursor*). Alternately, you may set the value “coarsely” with an *analog slider* control (under bar graph) or set it “finely” with an *up/down arrow* control (between PSet value and its units label). Click the **[Set]** button to start setting the pressure (or if you prefer, press <Tab>, then <Enter> after entering the value in the **PSet=** text box).



Controls for Setting Pressures manually for a Calibrator module

Pressure setting progress can be viewed in the displayed EU pressure data fields (tabular and graphic) as the actual measured pressure of the standard moves toward the desired *set point*. A **[Cancel]** button may be clicked to *stop* pressure generation (e.g., should there be no closed volume in which to build up a pressure, and it never converges to setting the correct value). To indicate the status of the pressure generation process, the background color of the **PSET** data field is *white* before the **[Set]** button (or <Tab> key) is pressed, *bright-yellow* while generating the pressure, and *bright-green* after pressure has been set. It also returns to *white* if **[Cancel]** is clicked or a new value is entered to be set.

The various individual *internal control valves* for a *calibrator* module may also be closed (or opened) appropriately by checking (or un-checking) its *check box* control, as needed. Normally, NUSS sets these valves automatically as needed, and the check boxes change automatically to indicate their current positions (for 903x firmware version 1.11 or later). These valves appear in the same **Set Pressure Controls** frame, because they are generally related to pressure generation. For example, the **Supply** and **Gen(erate)** valves are closed *automatically* when the **[Set]** button is pressed (if they are not already in their correct states). The **Zero** valve may be closed or opened *manually* as needed to connect the input to “atmosphere” or some other reference (e.g., for zero check).

An *absolute calibrator* module has an additional *check box* control to enable/disable its operating in an optional “**quasi-differential**” mode. This is not a valve inside the module, but only a module option. The **QDif** mode is used whenever an *absolute* calibrator is used to calibrate *differential* scanner transducers.

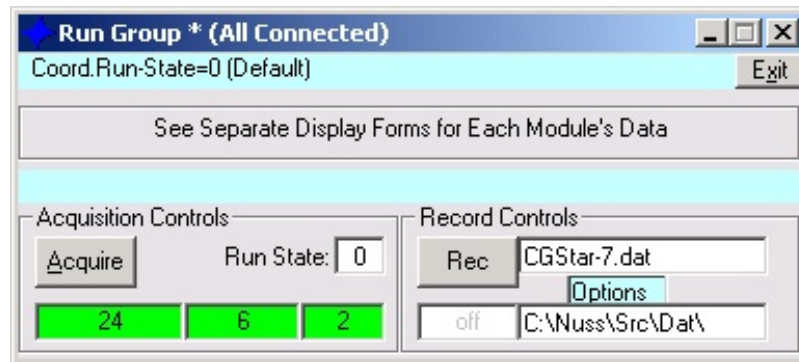
6.1.11 Other Limitations/Considerations for *Run*

Possible limitations in this single-module *Run* forms’s ability to keep up with “live” acquired data from a module can be avoided by *not* specifying **Run State** parameters that deliver data to *your* host faster than it can process it. Such limitations are, of course, very dependent on the computing power of your host PC and the available bandwidth of your network. These limits are of particular importance if you are going to be running this function for several modules concurrently. However, the **Run Group** function, available on the *home-base* menu (and described in the next section), has certain advantages over running multiple instances of this simpler single-module *Run* function. That is because it is able to keep up with higher data rates from its *coordinated* modules. This is particularly true for its **Record** feature, since it writes the data acquired from all modules to *one coordinated data file*. This is considerably more efficient than having multiple files open (independently for each module).

WARNING: NUSS uses several pop-up *Open File* dialog forms (that Windows itself provides), both for the ‘**File | View Files (NUSS)**’ function on the home-base menu, and for the support of many **[View]** buttons on various configuration, calibration, and test forms of NUSS. When these forms are popped-up, they may “defeat” the multiprogramming operations that both single or multiple *Run* forms attempt to achieve. Therefore, you should avoid using these forms, when the **Record** feature of *Run* is on – as live data stream acquisition and recording may be stalled while the forms are active.

6.2 Run Group

The **Run** function, selected from the *home-base* menu, is very similar to the single-module version of **Run** available from a module's *context* menu, but can operate a “coordinated” group of modules together — under the management of a single **Run Group** form.



‘Run | Run Group’ central control form

This common **Run Group** form has both similarities and differences when compared to the single-module **Run** form you are already familiar with. In particular, its size is fixed — though its position is remembered on exit. Thus, it lacks the [More] or [Less] or [Fix] buttons, and has no variable length **Data Display** frame that adjusts to number of channels displayed. Instead, a fixed “central” frame appears, containing only a message — indicating that all data are displayed one **Separate Display Forms** (one for each module in the group). **Acquisition** and **Record/Playback** frames still appear at the bottom of the form. However, there are no **Module Control** features (e.g., no **Valve Controls** or **Digital I/O Controls** or **Internal Functions** frames) as these module-specific controls appear in the above-mentioned **Separate Display Forms**. Top and Bottom Status Lines are retained (and have the same purpose) but they are colored *turquoise* (instead of *salmon*) as a reminder that you are using the **Run Group** function. The **Separate Display Forms** have matching color schemes.

6.2.1 Acquisition Controls of Run Group

The **Acquisition Controls** frame appears identical to single-module **Run**, except its **Run State** field can now indicate *mixed Run States*. This field (or its label or the surrounding frame's title) is still clicked to change the **Run State**, but now you may change it for (1) all modules in the group together or (2) for just one specific single module. You may use it to move from a “mixed” Run State to a “coordinated one”, and vice versa. Thus, it pops-up an extra form (to determine the *scope* of the **Run State** change) before it pops-up the form to select the **Run State** (see both examples next page).

Choose Scope of Run State Change

Modules used in Group

pos.	model	serial
1	9016	669
2	9034	119
3	9816	284

Buttons: Entire Group, Selected Module Only, Cancel

Choose Run State for Module 9016-669

☐ 0 (Default)
 ☐ Define a New Run State or Edit an Existing Run State

User-Defined States:

☒ 1 (Faster)
 ☐ 2 ()
 ☐ 3 ()
 ☐ 4 ()
 ☐ 5 ()
 ☐ 6 ()
 ☐ 7 ()
 ☐ 8 ()
 ☐ 9 ()

Buttons: OK, Cancel

Simply press the [**Entire Group**] button on this extra *scope* form if you want a coordinated **Run State** change. To change **Run State** for only a single module, highlight it first (by clicking it in the **pos.** column), then press the [**Selected Module Only**] button. In either case the *second pop-up form* is then displayed, and it operates essentially the same as for single-module **Run**. Click the [**Cancel**] button (on either form) to abort any *inadvertent Run State* change that you started but no longer wish to complete.

Note that the possible choices (on the second form example above) may be “fewer” for the *coordinated change* (Entire Group) case. When one (or more) particular **Run State(s)** is not defined for *every* module in the group, that **Run State** is dimmed, and thus not allowed. Such selections are dimmed for the single module (Selected Module Only) case only when the **Run State** is not defined for the selected module.

Returning now, to our description of the **Acquisition Controls** frame on the **Run Group** form (example previous page), the three *Stream Activity Indicators* look the same as they did for single-module **Run**, but they do not display the actual *sequence #'s* of any one module being acquired. Instead they each display an *internal counter*, incremented for that stream being acquired for *every* module in the group. Therefore, any one stream’s indicator tends to increment not by one, but by the number of modules in the group generating that stream. Such indicators are initially blank, and always start counting at 1. Their background colors indicate **Acquire** state, and are the same as for single-module **Run**: *bright-green* for *On*, *bright-yellow* for *Pause*, and *white* for *Off*.

6.2.2 Record/Playback Files and Controls of *Run Group*

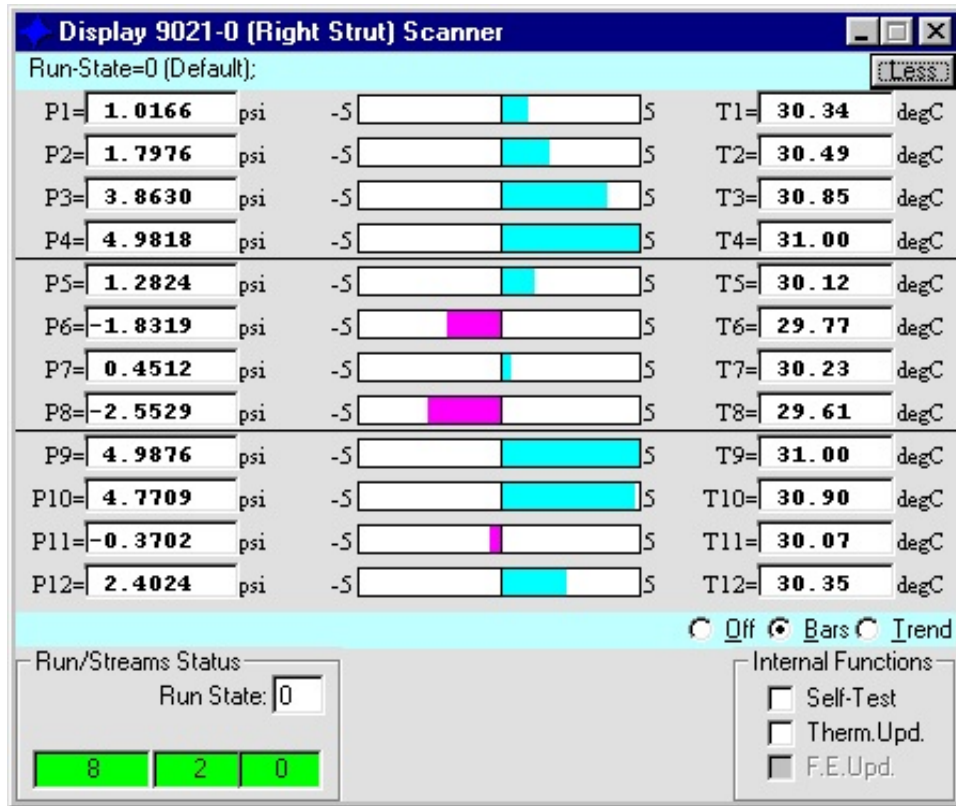
The **Record Controls** frame (and the alternate **Playback Controls** frame that can be selected by clicking the frame's title when acquisition is Off) are identical in appearance and operation to their single-module versions. However, any recorded data files contain the streams acquired by all modules in the group, as well as a the special barometer stream (4) synthesized from any other running barometers (in the group, or possibly running in a single-module **Run** form). The automatic naming convention of such files is similar, but begins with the fixed characters "**CG**" (for Coordinated Group) instead of beginning with a specific module ID (<modid>). Also, the *header records* written at the beginning (and possibly middle) of such files contain **Run State** and **Display Set** information for *every module* in the group.

6.2.3 Displaying Data in Separate Forms for *Run Group*

A *separate display form* appears for *each module* in the group. An example is shown (for one module) on the following page. These **Display** forms look similar to a single-module **Run** form, except that they lack any ability to control **Acquire**, **Record**, or **Playback** (which are now controlled from the central **Run Group** form). They do, however, retain **Module Control** functions such as a **Valve Controls** frame or a **Digital I/O Controls** frame or a **Internal Functions** frame — that a module of that type would have. They also have the same ability to change size as needed, and their positions are remembered when you exit **Run Group**. In fact, the same internal NUSS module files (<modid>**R.INI**) that save size, position, and **Display Set** configuration, are shared with the single-module **Run** function.

Notice that all these separate module **Display** forms have *turquoise*-colored Top and Bottom Status Lines so that you can distinguish them from multiple instances of the Single-Module **Run** function on the screen (which have *salmon*-colored Status Lines).

Both types are mutually-exclusive. That is, NUSS disallows you having a **Run Group Display** form for a particular module on the screen at the same time you have one or more single-module **Run** form(s) for the same module on the screen.



The **Run/Streams Status** frame, although it looks like an **Acquisition Controls** frame, is not exactly. It is thus renamed, because it only *displays* **Run State** and **Stream Sequence #’s** for that module. Also, you cannot change the **Run State** here, thus its *text box* is only an indicator. If you click it you are not allowed to change it (because **Run State** is only controlled (and changeable) from the coordinated **Group Run** form).

You may also click a *stream indicator* to *Pause*, but you are *only suspending the display update function of that module*, and not its **acquisition** (which is only controlled from the coordinated **Group Run** form).

6.2.4 Mixed-vs-Coordinated Run States for *Run Group*

The ***Run Group*** form can operate all the modules (in the selected group) according to the way they are *currently* setup when the function is started (i.e., each module might have been setup using *its own peculiar Run State* when last operated with the single-module ***Run*** function). Thus, some modules in the group may have been using the *default Run State (0)* — meaning that they scan *all channels* in the module periodically according to a default *leisurely-paced* internal clock. Still other modules in the group might have been using different *user-defined Run States (1-9)* — meaning that they generate one or more autonomous data streams to the host that are driven continuously (or in a limited fashion) by periodic or hardware triggered drive mechanisms. Although operating from such diverse data sources (called *mixed Run States*) is possible, it is not the *normal* way this ***Run Group*** function was intended to run. Whenever such a possibly chaotic condition is detected by the start of the ***Group Run*** task, it sets the ***Run State*** field to the word “*Mix*” instead of the usually number, and starts ***Acquire*** anyway.

At any time you may enter a particular *user-defined Run State (1-9)*, or the *default Run State (0)*, by clicking the ***Run State*** field. Then *all modules* in the selected group *may* be assigned one common ***Run State***, thus allowing them to share a “coordinated” setup method when they run. The word “coordinated” is in parentheses here to remind you that you must have planned it that way — when you defined *user-defined Run States* for the individual modules for a particular ***Run State***! Chaos can still reign if you did not plan for like-*user-defined Run States* to work together across multiple modules when you configured them.

6.2.5 Record/Playback for *Run Group*

The same basic **Record** and **Playback** features, available in the single-module version of **Run**, are also available in **Group Run** as well. However, data in the “coordinated” **recorded** file has contributions from possibly many modules. Such data aggregations may all be processed together (for time correlation of output data across many streams generated by many modules), or it may be edited (via **Playback’s secondary output file** feature) so that only one (or more) module’s contributions are output to the secondary file for one or more (specified) streams during one pass of the recorded data file.

Clicking the **Options** box, before or after starting **Record**, yields the same basic pop-up choices available for single-module **Run**.

Clicking the **Options** box, before starting file **Playback**, yields a similar pop-up option selection form, but with extra **module selection** check boxes added at the bottom of the form (see example below).

Playback Options for Group

The length of the **Options** selection form adjusts to accommodate all the modules in the current group (only 2 check boxes are shown in the example below). Thus, you can produce edited reports and spreadsheets containing not only specified single or multiple streams, but these selected streams may be contributed by single or multiple modules, as specified. Of course, the usual *relative time* editing choices are also available. Also note that there is one option (at the top of this form) that is not related to report editing, but applies to **Playback** in

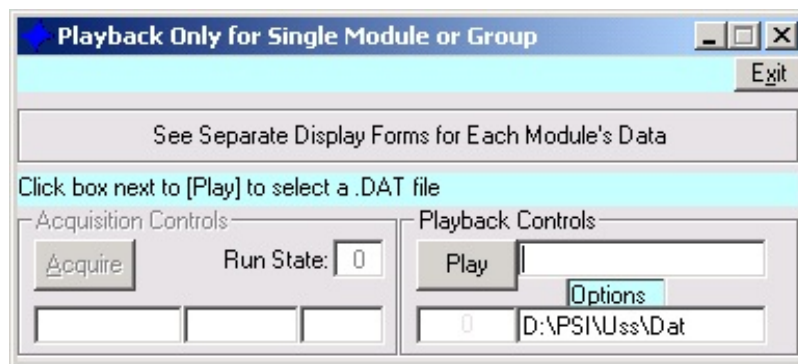
general. If the **Skip Pausing on Events** *check box* is checked, then **Playback** does not **Pause** when it encounters events in the recorded data file.

See **Appendix F** for the detailed *format* of all *files* (including **Record** and **Playback** files), should you wish to process any such files with your own programs. This appendix also details the format and naming conventions for the related *user-defined Run State* files, and their related *Display Set* files. The relationship between the **Record** data files and these other related *specification* files is also explained.

6.2.6 Playback-Only Activation of Run Group

When you select **Run Group** from the home-base **Run** menu item, you get a two-item submenu not present when you select **Run** from a single-module's *context* menu. The first selection on this added submenu (**Run (Group)**) does lead directly to the **Run Group** function with live data acquisition capabilities we have defined throughout this section (see example at beginning of **Section 6.2**).

The second selection on the submenu (**Playback-Only**) also starts the very same **Run Group** form, but starts it immediately with the **Playback** feature turned *On* and the **Acquire** feature turned *Off and disabled* (see example below).



'Run | Playback-Only central control form

This extra selection is useful in the event you have recorded any data from modules that happen to no longer be “available” at the time you wish to playback their data. Because it bypasses all the Current Group's module verification logic of the program, this form of the program allows you to playback any existing recorded data file — regardless of the modules contributing data to it and their availability on the network. Otherwise, **Run Group** would have been unhappy finding missing modules in its group and refused to continue.

You may use the **Playback-Only** feature of **Run Group** to playback data files originally recorded by the **Run Group** function — and by the single-module **Run** function as well.

To start the ***Playback-Only*** version of ***Run Group*** click the empty File box to the right of the **[Play]** button to obtain a dialog box — and use it to select a file to playback. This form does not pop-up any supporting ***Display*** forms until you select a file — and it knows the modules involved (by reading the header record at the beginning of the data file).

Click the Path box below the **Options** label if necessary to change the default path where data files are located, and then select a data file in the File box.

Click the **Options** label to select how the data are processed. Click the **[Play]** button to start playing-back the data in the selected file. If recorded events are encountered in the file, it may pause playback until you click the yellow blinking label below the **[Play]** button to resume playback. Clicking **[Play]** again restarts playback at the beginning of the file instead of resuming it.

Click the File box again to change to a different file, and banish the ***Display*** forms used to show data for the previous file. The new file's header is examined, and new ***Display*** forms appear for the particular modules recorded in the new file.

6.3 Run State Editor

The **Run State Editor** may be activated via the **Configure** menu functions (*home-base* or a module's *context* menu), or directly from the **Run** form of an operating module. Either way the following form appears initially – showing the module's current Run State (Run State 0 in this particular example). If the module is currently using another user-specified Run State (1-9), that one is displayed instead.

Edit Run State 0 Configuration For Module: 9016-0

Pick Run State To View/Edit/Create: Name for Run State:

☐ Inherit Defaults From Run State: Begin This Run State with Acquire: ☐ At End ... Chain To Run State: Begin This Run State with Record:

Enter # A/D Scans to Average: Select Initial Display Set:

Choose Acquire Start Mode
☐ Begin Streams Normally ☒ Begin with 1 of Each Stream Immediately ☐ Delete This Run State

Configure/Undefined 3 Autonomous Streams for Run State

Choose Stream To Configure: ☐ Undefine this Stream

Choose Channels included in each Channel Group of stream

18	16	13	12	9	8	5	4	1
<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Choose Synch. Type
☒ Internal Clock ☐ Hardware Trigger

Enter Stream's Period: (# Hdw. Triggers or Clock Milliseconds)

Choose Stream's Datum Format (* preferred)
☒ Binary Float (*) ☐ Hex32 Float ☐ Hex32 Scaled Integer
☐ Decimal Float ☐ Hex64 Float ☐ Binary Float (reversed)

Enter Total # Streams Acquired: (0=Continuous/Unbounded)

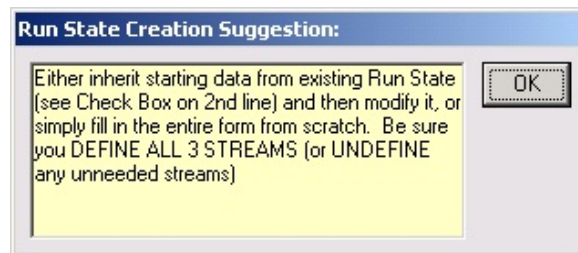
Choose Channel Groups & Status Prefixes of Stream

<input checked="" type="checkbox"/> Press. Data E.U.	<input type="checkbox"/> Temp. Data E.U.	<input type="checkbox"/> Valve Status Prefix	<input type="checkbox"/> Temp. Status Prefix
<input type="checkbox"/> Press. Data Counts	<input type="checkbox"/> Temp. Data Counts	<input type="checkbox"/> Reference Counts	<input type="checkbox"/> Zero Counts
<input type="checkbox"/> Press. Data Volts	<input type="checkbox"/> Temp. Data Volts	<input type="checkbox"/> Reference Volts	<input type="checkbox"/> Zero Volts

If the module's current Run State is not the one you want, then select another one by clicking the *arrow* on the right-side of the first combo-box. This shows all valid Run State numbers (0-9), whether they are created or not. The selected Run State appears on the form, if it exists.

6.3.1 Creating New User-Defined Run State For A Single Module

If the selected Run State does not exist, a pop-up form appears (see first example below). If you chose to create that new **Run State** (2 in this case), by clicking **[OK]**, another form appears (see second example below) to advise you of your basic goals:



After you click **[OK]** on the second form, the entire *new Run State* form is displayed with default parameters already set. You may use most of these parameters, editing only the ones that need to change — or you may first choose to *inherit* all the parameters of another existing **Run State** which is more closely related to (or similar to) the one you wish to define.

All **Run States** look the same on the editor's form, but notice that only the *first (primary) stream* is initially displayed. To inspect/modify the other streams (#2 and #3) you must select them sequentially in a list box at the top of the largest frame on the form. Note that the **Record** and **Playback** sub-functions of **Run** forms often refer to a fourth stream (#4). This is an automatically created Barometer stream created by NUSS itself if you have configured a barometer only, and such a stream cannot be generated by a module. Thus, Stream 4 does not appear as a choice for the **Run State Editor**.

The default **Run State 0** parameters were shown in the example on the previous page. Note that only a single *light-green* parameter named **Enter # A/D Scans To Average** could be changed for it. The other two green fields displayed were there to select the **Run State #** itself, or select each of its particular **Stream Definitions** (1, 2, or 3) that must be sequentially displayed on the form because of lack of room.

Any **User Specified Run State (1-9)** appears similarly, except that all its parameters on the form may be changed, thus they are all *light green*. In the example below (next page) Run State 1 of a Module 9046 module is shown instead. In this case, **Hardware Triggering** mode is selected for the Stream 1 shown instead of the default internal clock synchronization. Also, since this is a temperature/resistance scanner, the names of the various *data groups* at the bottom of the form have changed appropriately (i.e., Pressure data has been replaced by Temperature (or Resistance) data, and Compensation Temperature data has been replaced by UTR Junction Temperature data..

Run State 1 (named Hdw.Trig) of a 9046 Module

Yes, a typical **Run State** has many parameters! These must include the parameters of three (3) fully-defined *autonomous streams* (only the first of which is displayed in the largest *frame* occupying the bottom 2/3 of the form). However, don't be dismayed, as most parameters are set to intelligent defaults for you, requiring that you change only a few of them to obtain a useful new **Run State**. Notice (in above examples) that one field (labeled “**Inherit Defaults From Run State:**”) is disabled (dimmed) to indicate that choice is not available to existing Run States. However, for newly created **Run States** only, it is enabled and highlighted to draw your attention to any important *first feature* you should use — the ability to *inherit* the parameters from an existing **Run State**, before you start editing it. To use it, first click the *arrow* in the *list box* at the *right* end of this field. This reveals a *list* of all “existing” **Run States**. Select the one you want to inherit from by clicking its *list item*. When you start with a new module, only **Run State 0** (permanently named **Default**) may exist in this list. This is a good one to inherit until you have created others. Finally, click the *checkbox* at *left* end of this field to **execute** the *inherit* function. Next, all the fields in the form change to the parameters of the inherited **Run State**, and an *entry cursor* moves to

the field labeled “**Your Name for Run State**”. The name of the inherited **Run State** is highlighted (**Default** in this case) to encourage you to *replace* it — so, just start typing your new unique name (see example below). If this *name* field is blank, simply click it to start a new name entry for your new Run State.

Edit Run State 2 Configuration For Module: 9046-0

Pick Run State To View/Edit/Create: Name for Run State:

☐ Inherit Defaults From Run State: Begin This Run State with Acquire:

☐ At End ... Chain To Run State: Begin This Run State with Record:

Enter # A/D Scans to Average: Select Initial Display Set:

Choose Acquire Start Mode

☐ Begin Streams Normally ☒ Begin with 1 of Each Stream Immediately

☐ Delete This Run State

Top Section of Run State Editor parameters only

First, we concentrate only on the top section of the form (illustrated above). It includes all parameters of the **Run State** *except data streams*. Once you have *named* your **Run State**, the next few fields on the form define the *Begin* conditions and *End* conditions of the **Run State**.

The *Begin* conditions determine whether the **Run** form changes the status of its **Acquire** and **Record** features when it starts using this **Run State** for a particular module, or whenever this **Run State** is later selected on the **Run** form. Each of these fields has a *list box* with three (3) choices in its drop-down list. These choices are *Off*, *On*, or *Null*. The first two cause the affected feature to be *started* or *stopped* when the **Run State** is first used. The *Null* choice (default for both conditions) is used if you want the affected feature to be *unchanged* (from the state it was in) when the **Run State** is first used. Thus, you can ignore the *Begin* conditions, if you want your new Run State to leave its module’s Acquire and Record functions unchanged when it is invoked.

The *End* function has a *list box* (to *configure* it) and a *check box* (to **execute** it). To configure an *End* function, the *combo box* picks another **Run State** for the **Run** form to **automatically chain to** after this particular **Run State** ends (i.e., when all defined but *limited* streams are completed, or the user simply *stops* the **Run State**’s acquisition). Click your choice in the *combo-box* drop-down list (your selection does not have to be currently defined now). Finally, click the *check box*. Like the *Begin* conditions, you should probably not select an *End* condition unless you have an *overpowering need* to use one (i.e., you need a new **Run State** to start automatically when another ends).

The next field on the form (labeled “**Enter # A/D Scans to Average:**”) is a normal **Acquisition** option of any module, and determines how many data scans it makes internally

(of each stream) before the *averaged* datum value of each channel is calculated (i.e., the value for a channel that is returned to host in all data streams). A **Run State** allows you to *change or not change* this option, in the module, when *starting* to use this **Run State**. The factory default is **0** meaning “don’t change it”. The values 1, 2, 4, 8, 16, 32, 64, or 128 are allowed if you want the **Run State** to “change it”. This is the only field you may change for **Run State 0**. A small value yields faster potential data stream throughput at the expense of “noisier” data. A large value limits the maximum possible data throughput of the module, but yield “smoother” data. You can also change averaging it in the module directly with **‘Configure | Other Options’** on a single-module’s context menu, in which case **0** is a better choice for this parameter in Run State definitions. That way, the module’s configuration command controls it exclusively – not the changing of any Run States of that module.

The field labeled “**Select Initial Display Set:**” is currently *unused*, thus is *dim*, and not selectable. Currently NUSS gives a **Run State** only a single **Display Set**, which includes, *for all the data channels scanned in all the streams* of that **Run State**: a display configuration memory (content and precision) for each displayed tabular data item.

The *frame* titled “**Choose Acquire Start Mode**” contains two options, one of which you must choose. The first choice “**Begin Streams Normally**” causes *all* autonomous streams generated by using this **Run State** to act *naturally*. This means, that for streams that repeat less frequently (i.e., have longer periods), its first available scan (sequence #) arrives in the host possibly *many seconds* after **Acquire** starts. If this is not annoying to you, then this is the way you want to define your streams. It is the default mode for streams. The second choice “**Begin with 1 of Each Stream Immediately**”, is what the *default Run State 0* always uses. This mode causes **Acquire** to start with one *immediate (prefix)* scan of *each stream* (designated Sequence # 0), followed by the natural scans of the stream (i.e., starting with Sequence # 1). This “prefix” scan does a better job of “lighting up” all the data displays immediately when **Acquire** starts. If it were not used, no “other” temperature data would appear in the **Run** form until 7.5 seconds after the start of acquisition. Please note that this feature is an *internal* “acquisition initialization” feature of a **Run** form’s software and not a feature of the module itself. The time between the *prefix* scan and the first natural scan of fastest stream is short and unpredictable.

Note the final *check box* labeled “**Delete this Run State**”. Click it if you want to delete the **Run State** being edited (i.e., delete its `<modid>rs<runstate#>.ini` file).

Now let's focus our attention on defining each of the three (3) *autonomous data streams* that a module should generate when it uses this new **Run State**. Again, study the **Run State 0** defaults (in previous and following examples) for clues to a useful stream creation. The **Run State Editor** only has room on its already crowded form to show only *one* of these *three* streams at a time. It begins by always showing only the Primary stream (1) in the

largest frame (last 2/3 of form) which is labeled “**Configure/Undefine 3 Autonomous Streams for Run State**” (see first example below). Other streams are viewed and edited separately by selecting them (one at a time) from the first *list box* in this frame..

Configure/Undefine 3 Autonomous Streams for Run State

Choose Stream To Configure: **Primary Stream (1)** ☐ Undefine this Stream

Choose Channels included in each Channel Group of stream

18	16	13	12	9	8	5	4	1
<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Choose Synch. Type
☒ Internal Clock
☐ Hardware Trigger

Enter Stream's Period: **500** (# Hdw.Triggers or Clock Milliseconds)

Choose Stream's Datum Format (* preferred)
☒ Binary Float (*) ☐ Hex32 Float ☐ Hex32 Scaled Integer
☐ Decimal Float ☐ Hex64 Float ☐ Binary Float (reversed)

Enter Total # Streams Acquired: **0** (0=Continuous/Unbounded)

Choose Channel Groups & Status Prefixes of Stream

<input checked="" type="checkbox"/> Press. Data <u>E</u> .U.	<input type="checkbox"/> Temp. Data <u>E</u> .U.	<input type="checkbox"/> <u>V</u> alve Status Prefix	<input type="checkbox"/> <u>T</u> emp.Status Prefix
<input type="checkbox"/> Press. Data <u>C</u> ounts	<input type="checkbox"/> Temp. Data <u>C</u> ounts	<input type="checkbox"/> Reference Counts	<input type="checkbox"/> Zero Counts
<input type="checkbox"/> Press. Data <u>V</u> olts	<input type="checkbox"/> Temp. Data <u>V</u> olts	<input type="checkbox"/> Reference Volts	<input type="checkbox"/> Zero Volts

Current Stream 1 Frame for PSI Model 9016 Module's Run State 0

The example below shows the third (#3) stream for a temperature/resistance scanner.

Configure/Undefine 3 Autonomous Streams for Run State

Choose Stream To Configure: **Tertiary Stream (3)** ☐ Undefine this Stream

Choose Channels included in each Channel Group of stream

18	16	13	12	9	8	5	4	1
<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Choose Synch. Type
☒ Internal Clock
☐ Hardware Trigger

Enter Stream's Period: **7500** (# Hdw.Triggers or Clock Milliseconds)

Choose Stream's Datum Format (* preferred)
☒ Binary Float (*) ☐ Hex32 Float ☐ Hex32 Scaled Integer
☐ Decimal Float ☐ Hex64 Float ☐ Binary Float (reversed)

Enter Total # Streams Acquired: **0** (0=Continuous/Unbounded)

Choose Channel Groups & Status Prefixes of Stream

<input type="checkbox"/> Temp. Data <u>E</u> .U.	<input checked="" type="checkbox"/> UTR Data <u>E</u> .U.	<input type="checkbox"/> <u>V</u> alve Status Prefix	<input type="checkbox"/> <u>T</u> emp.Status Prefix
<input type="checkbox"/> Temp. Data <u>C</u> ounts	<input checked="" type="checkbox"/> UTR Data <u>C</u> ounts	<input type="checkbox"/> Reference Counts	<input type="checkbox"/> Zero Counts
<input type="checkbox"/> Temp. Data <u>V</u> olts	<input checked="" type="checkbox"/> UTR Data <u>V</u> olts	<input type="checkbox"/> Reference Volts	<input type="checkbox"/> Zero Volts

Current Stream 3 Frame for Model 9046 Module's Run State 1

Notice first the *check box* titled “**Undefine this Stream**”. It is only available for a user-defined Run State (like second example above) – as Run State 0 (first example) cannot be modified. If you click it, the stream is undefined, and no other parameters on the stream frame are pertinent. If you do not click it, then you must define all the other parameters of the stream explicitly, or use the defaults offered or inherited.

Initially the *channels* selected (i.e., checked) to be scanned by *all* streams of this **Run State** include *all those possible* for the particular module class. If any channel #'s are *dim*, and thus cannot be checked, then that module does not have a channel # that high. If you uncheck any of the module's existing channels, then the **Run State** does not scan those channels for *this stream*. However, when you proceed to edit the other (higher numbered) *streams* later, you notice that these have also *inherited* only the channels that were checked for this (lower numbered) *stream*. Practically, however, there is probably no reason for other streams to include a *different* number of channels. Normally, other streams specify different data groups (look ahead to last frame on form), but each group includes the same channels scanned. However, you may violate this rule.

The next frame (labeled “**Choose Synch.Type**”) contains two options, only one of which must be selected. Your choice determines whether this stream is periodically self-generating (i.e. synchronized with an Internal Clock) or generated only by a Hardware Trigger (a digital pulse signal applied directly to the module, and provided by the user). Use the **Internal Clock** choice if you are not sure that a hardware trigger signal is available and connected to your module. However, the **Hardware Trigger** choice is essential when maximum throughput is a requirement, since it is the only way internal scanning (and its module's internal EU data conversion task) are synchronized with stream delivery to the host. When Internal Clock is used, scanning (and EU data conversion) run independently of stream generation inside the module.

The next field of a stream definition is labeled “**Enter Stream's Period:**”. Its meaning depends on the **Synch.Type** you selected. For **Internal Clock** synchronization, it must specify the **Period in Milliseconds** that your stream repeats. For **Hardware Trigger** synchronization, it must specify *how many hardware triggers* to count before generating your next stream repetition.

Guideline: Always make the Primary stream (1) faster than the others, and the Secondary stream (2) faster than the Tertiary stream (3). This is not enforced, but keeps the stream names consistent with their performance. When you find that you don't need the scanning diversity of three different streams for a particular **Run State**, then Undefine the unneeded streams (starting with the Tertiary one).

The next frame (labeled “**Choose Stream’s Datum Format (* preferred)**”) lists several datum format choices, for each datum of each channel of each data group included in the stream. Most of these are *legacy* formats, not very useful and not very efficient, when used with NUSS. Only the first choice (**Binary Float (*)**) is *recommended*, since it causes the module to emit the most compact and efficient datum format for each channel scanned. Not only does the module itself not have to do unnecessary conversions work (reformatting data into the legacy formats from data already naturally scanned as binary float values) but NUSS receives the streams in the form it can immediately use them. Otherwise, NUSS must also convert the data from legacy formatted streams back into the simpler binary form before it can utilize them.

The next field (labeled “**Enter Total # Streams Acquired**”) chooses whether your stream is unbounded (i.e., *continuous*) or whether it stops automatically after a finite number of scans of the stream are generated (i.e., called a *limited* stream) — each time *Acquire* is started for your **Run State** (in a *Run* form of the module). If all defined streams are *limited*, and the **Run State** has an *End* function defined (i.e., another **Run State** is defined to be **chained to...**) then this *End* function is executed when *all* the defined *limited* streams have expired (i.e., have been received and counted by the host). The *End* function is also executed if you simply turn *Acquire Off* manually for that Run State.

The final frame (labeled “**Choose Channel Groups & Status Prefixes of Stream**”) allows you to choose the major content of the stream. **Channel Groups** pick types of data (scanned for *each channel* checked above) to be included in the stream. Notice that pressures scanners and temperature/resistance scanners name their groups differently in the examples above.

The list at the bottom of the frame should be self-explanatory, except that those on the left are natural to all module classes, and those on the right (if they are not *dim*) exist only for certain 903x differential type Calibrator modules with HASS front-ends). **Status Prefixes** are different (i.e., they are fixed (mostly 16 bit) initial stream appendages, whose length doesn’t vary with the # of channels), and not every module class can generate these prefixes, thus one or more may be *dim* for your module, and cannot be selected. Only a Model 9816 module has both **Valve Status** and **Temperature Status** prefixes. Model 903x modules have only the **Valve Status** prefix. Model 9022 has neither. Model 9x46 temperature/resistance scanners have a **Temperature Status** but no **Valve Status**. NUSS includes all these necessary *status* fields (for an appropriate module) in its Secondary stream (#2, not shown above) when it creates the default **Run State 0** for a module.

Also note that there is an optional “byte count” prefix that can optionally start any stream, and a required “stream id” prefix. These are not shown as selections on this form because they are ALWAYS USED by NUSS. See Appendix F for details of group and status prefix formats in *recorded* streams (these have other fields added by NUSS).

Well, you have now defined your Run State's first stream. Be sure you go back to the top of this frame and select each of the other two streams as well. You must either define them with reasonable parameters, retain the parameters inherited, or undefine that stream (per *check box* at top right of frame).

After the last stream is defined or viewed, you can complete your definition of your **Run State** by simply clicking the **[Save]** button before you click **[Exit]**. The **Run State** is saved in a special file in the **Ini** subfolder (of Main Base Path) and is named so as to identify its target module and which **Run State** (0-9) of that module it is. If you are just practicing, and do not really want to save your edit work, press **[Exit]** instead to leave the **Run State Editor**. This warns you that you are trying to exit without saving. Simply click **[Exit]** a second time to leave without saving.

Whenever the **Run** form starts for the module, and this **Run State** was the last one used by that module, it is fetched from its file (like an internal script) and used to initialize the module and all its streams. **Acquisition** then begins (if your *Begin* condition so permits). If you select another **Run State** while in the **Run** form (or existing **Run State** terminates with an *End* function), then this state is discarded, and the module is setup to use a new selected **Run State**.

6.3.2 Creating/Editing Run States For A Group

The **Run State Editor's** Title Bar always identifies the module (or “group” of modules) for which it is configuring a **Run State**. The “group” version has a few added features. It seems to work just as for a single-module, and comes up initially ready to edit the *first stream of the first module* in the *Current Group*.

It has an extra control labeled [**Next Module**] which allows you to quickly move from one module to module within the group – but only if you have not modified any streams of the current module. If you have edited any streams, you are warned to save it (or lose it) before proceeding to the next module.

When you are ready to save a new or modified **Run State**, you press [**Save**] or [**Save to All**] to save it. Both do save the edited Run State in a file, but the latter also *replicates* it (with certain *automatic modifications*) in all the other modules in the *Current Group*.

Obviously, the *replication* feature only works well when the other modules in the group are identical models to the first module edited. If you try to use it with dissimilar modules the results may not be satisfactory.

6.3.3 Other Notes on Run State Editing

If you only view (i.e., don't change) any of the editor's displayed parameters, the various fields retain their *light-green* background color, as it was when the editor starts. However, the first time you change *any parameter on the form* that field changes its background color to *dark-pink*, as an indication of *change*. If there are no *dark-pink* fields displayed when you are ready to exit the editor, then you can be sure that no parameter was changed.

Also, the **[Save]** button is not even enabled unless there are changes to save. Thus, you can usually tell whether you must save or not before clicking the **[Exit]** button. Further, if you press **[Exit]** without saving changes made, you are warned to click **[Save]** once.

If you start modifying parameters of a new stream before you decide to *inherit* another **Run State's** parameters, a pop-up form warns you that other parameters are already modified. Click **[OK]** on the pop-up to continue with the *inherit*. However, after the *inherit*, all the *dark-pink* "change" indicators are returned to their *light-green* "unchanged" states. In other words, a freshly *inherited* form is considered *unchanged*, until another parameters are subsequently changed by the user, and any editing that was done before the *inherit* is lost.

6.4 Limitations and Other Characteristics of Run

Possible limitations in a **Run** form's ability to keep up with "live" acquired data from a module can be avoided by *not* specifying **Run State** parameters that deliver data to *your* host faster than it can process it. Such limitations are, quantitatively, very dependent on the computing power of your host PC and the available bandwidth of your network. These limits are of particular importance if you are going to be running this function for several modules concurrently. However, the "group" **Run** function, available on the *home-base* menu, has certain advantages over this simpler single-module **Run** function. That is because it is able to keep up with higher data rates from its *coordinated* modules. This is particularly true for its **Record** feature, since it writes the data acquired from all modules to *one coordinated data file*. This is considerably more efficient than having multiple files open (independently for each module).

Actually, the data formats for **Recorded Data** files (as detailed in **Appendix F**) are the same for both the single-module **Run** function and the "group" **Run** function. Single-module **Run** simply uses a *subset* of the features of the broader file format definition. In fact, the **Run Group** form's **Playback** feature can read not only the "group" files its own **Record** feature writes, but it can also read any of the recorded data files written by single-module **Run** — and such files do appear in its *filename* dialog box that pops-up when you click its *filename* field. The special header records that are written at the start (and middle) of "group" files are much larger though — since they must record the **Run State** and **Display Set** information for each module in the Group instead of just for a single module.

Each Model 903x *calibrator* and *standard* module has its own standard *recording* function that can be used for *recording* the one-channel data streams generated by that module. However, when one of these is also a *barometer*, its own *recording* function is rarely used, since the *recording* function for the *scanner* module (that needs the barometer's data for use during playback) actually records *synthesized* "Stream 4" records containing the data from these modules. When such a barometer module runs (in its own single-module **Run** form) and it knows that it is assigned to a particular LBN (Logical Barometer Number 1-4) for use by scanner modules, it deposits each EU pressure datum it scans (in its Primary stream) into a special 5-datum "global" pressure buffer dedicated for *synthesized* "Stream 4". The first datum in this buffer contains a fixed 14.7 psi constant. Within the range of the remaining four words (actually 32-bit IEEE single floats) are where each barometer module deposits its datum (indexed by its assigned LBN). The *constant* datum may be used by scanner modules that want *absolute* data displayed, but are not concerned by accuracy or barometer associations. The other data are used by scanner modules who have been assigned real associations with specific real barometers (i.e., specific transducers have been assigned to specific LBNs).

The factory default value assigned to the # **A/D Scans Per Average** parameter of a Model 9x46 temperature/resistance scanner is 64. This results in maximally smooth data at less than maximum possible scanner throughput. You, can reduce this parameter to lower values (powers of 2: 32, 16, 8, 4, 2, or 1) to increase the throughput of the module when data noise is not a concern.

The factory default value assigned to the # **A/D Scans Per Average** parameter of all PSI pressure scanners is 8. You can change its value up or down to increase throughput or get smoother data. Although you can set a Run State to changed the # **A/D Scans Per Average** parameter it is best to avoid that feature (i.e., set it to 0, meaning do not let Run State change it) and use the '**Configure | Other Options**' function instead to change the parameter.

WARNING: NUSS uses several pop-up *Open File* dialog forms (that Windows itself provides), both for the '**File | View Files (NUSS)**' function on the home-base menu, and for the support of many **[View]** buttons on various configuration, calibration, and test forms of NUSS. When these forms are popped-up, they may “defeat” the multiprogramming operations that both single or multiple **Run** forms attempt to achieve. Therefore, you should avoid using these forms, when the **Record** feature of **Run** is on – as live data stream acquisition and recording may be stalled while the forms are active.

NOTICE: The **Run** form normally sets the *range* of each pressure transducer (and indirectly its bar graph scales) according to a “standard” **Range Code** stored in the module for each transducer. This **Range Code** is “looked up” in a special editable text file **RngCodes.txt** which is stored in the **Ini** subfolder of NUSS. The result of the lookup are *low* and *high* end-point values of the transducer’s range and its *units*. A bar graph’s *low* end-point may be *lower* (more negative) than the transducer’s actual *range low* end-point to insure that the *zero* point of all bipolar bar graphs are in the center. If a module contains an *invalid Range Code* for any transducer, or the **RngCodes.txt** file does not contain the code’s low, high, units information, the **Run** display chooses a *default range* of **-r** to **+r** where **r** is the Range Code number itself, and the units are displayed as “????”. See **Appendix F** for more information on the **RngCodes.txt** file and how to edit it to include new range codes. The **Model 9x46** module has a similar lookup file (**SenType9046.txt**) and a Sensor code per channel to find the range and units of its non-pressure transducers (also described in Appendix F).