WinGPIB USER tab - Configuration Format Guide V3

by Ian Johnston - 4th Feb. 2026

(Requires WinGPIB V4.088+)

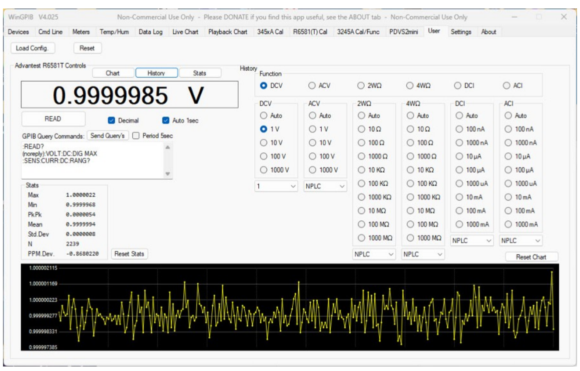
WinGPIB is a Windows-based application designed to control and monitor GPIB-enabled test and measurement instruments. It provides direct communication using standard GPIB and SCPI commands, supporting both manual operation and automation.

This help guide describes the **USER tab**, which allows users to create custom instrument control panels. The USER tab is driven by a simple text-based configuration file that defines both the layout and behaviour of the interface. Typically, one configuration file is used per instrument, although both Dev1 & Dev2 can be interfaced with on the same config file.

When a configuration file is loaded, it is parsed and the defined controls are created and displayed immediately on the USER tab. The configuration system supports a wide range of generic control types, including buttons, displays, and input fields. In addition, more advanced features are available, such as scripting and automated instrument control, all with full GPIB communication support.

Note: This guide follows the newer ‘named’ config file style entries. Old positional only ones have limited usage (legacy).

CONFIG FILES	–	Overview
GPIB ENGINE	–	Specifies which internal engine within WinGPIB is used (native or standalone).
BOOTCOMMANDS	–	Initialize instrument before operating.
DATASOURCE	–	Query producer for a named result stream.
BUTTON	–	Executes a SEND, SENDVALUE, or QUERY action when pressed.
MULTIBUTTON	–	Multi button version as above.
TEXTBOX	–	Displays results or accepts user-entered values.
CHECKBOX	–	Enables optional behaviours (Decimal mode, Auto-read, etc.).
LABEL	–	Displays static text anywhere on the layout. Different font sizes can be set.
DROPDOWN	–	Sends a command immediately when the user selects an item.
RADIOGROUP	–	Creates a grouped set of RADIO options.
RADIO	–	Sends a command when selected within its RADIOGROUP.
SLIDER	–	Adjustable trackbar that sends a scaled value on release.
HLINE	–	Horizontal separator line for visual grouping.
VLINE	–	Vertical separator line for visual grouping.
SPINNER	–	Numeric input control that sends a command when user selects.
LED	–	Indicator that changes colour based on query results.
BIGTEXT	–	Large digital-style display showing the same output as a logical result stream.
TEXTAREA	–	With an associated button, sends a command from each lineMulti-line script box.
TOGGLE	–	Button with on/off states, sending different commands for each.
TOGGLEDUAL	–	Dual button with left/right on/off states, sending different commands for each.
STATSPANEL	–	A small panel with Max, Min, PkPk, Mean, Std.Deviation & PPM Deviation.
HISTORYGRID	–	Rolling history table logging live reading/calculated in a grid format.
INVISIBILITY	–	Hide specific controls via a button, great for overlapping controls on a congested GUI.
TRIGGER	–	Monitors values and fires actions when conditions are met.
CALC	–	Define derived numeric result streams based on existing result names.
DETERMINE PARAM.	–	Match controls with actual instrument status when opening a config.
KEYPAD (POPUP)	–	Numerical keypad popup for use with textboxes.
DATASAVE PARAM.	–	Controls whether USER-tab TEXTBOX values are automatically saved and restored.
TAB PARAM.	–	Multiple sub-tabs.
SCRIPTING (LUA)	–	Run LUA scripts direct from the USER tab or from the Config file.



CONFIG FILES

WinGPIB uses a plain-text configuration file to define the layout, controls, and behaviour of the User tab. Each file is read at runtime and can be edited with any text editor — no programming is required.

```
; dual toggle between NPLC 1 & NPLC 10
TOGGLEDUAL;
   name=NPLCmode;
   caption=;
   device=dev1;
   left=NPLC 1;
   right=NPLC 10;
   on=:VOLT:DC:NPLC 1;
   off=:VOLT:DC:NPLC 10;
   oncolor=yellow;
   x=700;y=460;w=100;h=35;
   determine=:VOLT:DC:NPLC?|1|respnum
```

GENERAL STRUCTURE:

- Each control or directive begins with a keyword (for example BUTTON, RADIO, SLIDER, LUASCRIPTBEGIN).
- Parameters are supplied as key=value pairs.
- Lines starting with a semicolon (;) are comments and are ignored.
- In-line comments must use double semi-colons.
- If a physical line starts with any whitespace (space/tab), it is considered a continuation of the previous logical line (under certain conditions). If you don't have indents where they are necessary the config file won't load properly.

EXAMPLE (single-line form) ---- THIS IS BEING PHASED OUT:

```
BUTTON;caption=Run;device=dev1;command=:READ?;x=20;y=40
```

EXAMPLE (multi-line form):

Configuration entries may span multiple lines for clarity and readability.
Rules for multi-line entries:
The entry starts with the control keyword followed by a semicolon.
Each following line contains one parameter and ends with a semicolon.
Indentation must be 3 spaces.
The entry ends when a new control keyword or comment is encountered.

```
RADIO;
   group=DMMFunction;
   caption=DCV;
   device=dev1;
   command=:CONF:VOLT:DC;
   x=10;
   y=20;
   determine=:CONF?|VOLT:DC|resptext
```

COMMENTS, SPACING & INDENTATION:

All lines should end with a semi-colon unless with a comma (multi-line commands/parameters).
Lines beginning with a semicolon (;) are comments, unless it's within an existing multi-line continuation whereby you need to begin the line with a double semi-colon.
Comments may appear between entries or around multi-line blocks.
In-line comments must use double semi-colons.
Blank lines are ignored.
Parameters under a control are usually indented (3 spaces).

Example:

```
RADIO;
   group=DCRange;
   caption=200 mV;
   device=dev2;
   command=:SENS:VOLT:DC:RANG 0.2;
   x=10;y=45;
   scale=1000;
   when=Range==0.1;
   determine=:VOLT:DC:RANG?|0.2;
   dp=6;
```

Parameter Order:
Parameters may appear in any order.
Only the required parameters for a given control are needed.
Optional parameters may be omitted.

CONFIG FILE EDITOR:
Built-in **User Config Editor** for creating and modifying WinGPIB user layout files.
- Full **syntax highlighting** for block headers, parameters, comments, and section headers.
- **Line numbers** and a **clickable block navigation panel** to quickly jump around large configs.
- **Auto-indent**, smart Enter behaviour, and inline ;; comment support.
- **Right-click editing tools** including duplicate-line, cut/copy/paste, and now Save.
- **Ctrl+S** to save, **Ctrl+F** to find, with a built-in **Find bar** supporting F3 / Shift+F3 navigation.
- Dark theme with clean UI and smooth scrolling preserve cursor and view positions during edits.
- Checks are done on loading the file, these are the current list of checks:
    - Semi-colon check.

SUMMARY:
Config files are human-readable and editable.
Controls may be defined on one line or many.
The multi-line format is recommended for complex controls.
Comments and whitespace are fully supported.
No scripting or compilation is required.

GPIB ENGINE SELECTION

GpibEngineDev1
GpibEngineDev2

FIELD MEANINGS:
GPIBengineDev1 or 2 = Selects which GPIB communication engine WinGPIB will use & device.
native = Uses WinGPIB’s existing built-in GPIB engine with all legacy features (checkbox controls) enabled.
standalone = Uses the new internal SCPI/GPIB engine with no built-in device behaviours.

BEHAVIOUR:
If GPIBengine=native:
• Built-in polling options for dev1/dev2 are active.
• Device-specific features enabled (3457A 7th digit, divide/multiply ×1000, regex cleaning, etc.).
• Terminator options (CR/LF) are applied.
• Keithley 2001 data isolation and letter stripping are active.
• Ignore-errors mode is respected.
• See the Blocking query and response on WinGPIB's main DEVICES tab.

If GPIBengine=standalone:
• Commands are sent exactly as written.
• No legacy or device-specific processing is applied.
• Recommended for users with modern SCPI devices or when compatibility issues arise.

If omitted, GPIBengine defaults to **native**.

EXAMPLES:
For Dev1 (similarly for Dev2).

# Use full WinGPIB legacy engine (QUERY ASYNC / SEND ASYNC)
GpibEngineDev1=native;

# Use modern clean SCPI engine (QUERY BLOCKING / SEND ASYNC) – THIS IS BEING PHASED OUT
GpibEngineDev1=standalone;

BOOTCOMMANDS

FIELD MEANINGS:
BOOTCOMMANDS = Defines a list of SCPI commands that WinGPIB will send automatically immediately after the config file is loaded, before any DATASOURCE, BUTTON, RADIO, or other user controls operate.

device = Selects which GPIB device executes the commands.
dev1 = Device 1
dev2 = Device 2

Add a delay between commands being sent.
DelayPerCmd = time in secs

commandlist = A comma-separated list of SEND commands to execute in order.
Each command is transmitted exactly as written.
Commands are executed sequentially.

BEHAVIOUR:

• Commands are executed once when the config loads.

• Executed before any user-initiated actions or auto refresh tasks.

• Uses the currently selected GPIB engine (native or standalone).

• If a command fails, remaining commands are still attempted.

• Useful for instrument reset, range setup, NPLC, filters, display text, trigger control, etc.

If omitted, no boot commands are executed.

EXAMPLE:

```
BOOTCOMMANDS;
  device=dev2;
  DelayPerCmd=0.05;
  commandlist=*RST,
          :DISP:WIND1:TEXT:DATA "WinGPIB WinGPIB",
          :DISP:WIND1:TEXT:STAT ON,
          :SENS:VOLT:DC:DIG 8.5,
          :SENS:VOLT:DC:RANG 2,
          INIT:CONT OFF,
          :DISP:WIND1:TEXT:STAT OFF;
```

DATASOURCE

DATASOURCE is a non-visual config entry that defines a query “producer” for a named result stream. It does not create any UI controls. Its purpose is to provide query details (device + command + result name + overload detection token) that other features (especially FuncAuto) can use without needing a visible QUERY button.

SYNTAX (NAMED ONLY):
DATASOURCE;name=...;device=...;command=...;result=...;overload=...|text or raw;decimal=...|1 or 0

FIELD MEANINGS:
DATASOURCE = Identifies a non-visual query producer

name = Unique identifier for this DATASOURCE definition
Used internally for reference/debugging. This is NOT the result name.

device = Instrument device to query (dev1, dev2, etc.)

command = Query command to send (e.g. READ?, OUTPUT?, *IDN?)

result = Logical result stream name written by this DATASOURCE – **IMPORTANT** – Must start with a letter, not a number
Any control or function bound to this result name will update.

overload = overload detection token | text or raw GPIB reply to detect
If the incoming value text contains this token, WinGPIB treats the reading as OVERLOAD.
Matching is case-insensitive and substring-based.
Important: the token is checked against the processed display value (after parsing/formatting), not necessarily the instrument’s raw scientific notation string.

decimal = Controls whether numeric values are displayed as decimal or left in the instrument’s original scientific notation.
1 = decimal formatting (default)
0 = scientific notation / raw instrument format

IMPORTANT CONCEPT:
There are TWO names involved:

• name= = the producer ID (DATASOURCE definition name)
• result= = the value stream name (the thing that updates displays)

EXAMPLE:

```
DATASOURCE;
    name=Q_R6581T;
    device=dev1;
    command=READ?;
    result=R6581T;
```

Here:
• Q_R6581T is the producer ID
• R6581T is the result stream

WHAT UPDATES FROM A RESULT STREAM:
Any of the following can consume the result stream named by result= :

• BIGTEXT;name=<result>
• CHART;target=<result>
• STATSPANEL;target=<result>
• HISTORYGRID;result=<result>
• LED;Name=<result>
• SENDVALUE;value=<result> (uses the latest value)

FUNC AUTO INTEGRATION:
FuncAuto checkboxes bind to a result name. When enabled, WinGPIB will:

• look for a DATASOURCE that produces that same result name
• if found, use its device + command to perform repeated queries
• if not found, fall back to the legacy method (using a QUERY button as the source)

This is why DATASOURCE allows fully automatic operation without a visible READ button.

OVERLOAD BEHAVIOUR:
When an overload token matches:
The displayed result becomes OVERLOAD
The numeric value for this result stream is set to NaN (so calcs/triggers don't blow up)
Linked numeric consumers (CHART / HISTORYGRID / STATSPANEL) do not update with the overload value

EXAMPLE (AUTO READ WITHOUT A BUTTON):
```
DATASOURCE;
    name=Q_ReadDMM;
    device=dev1;
    command=READ?;
    result=YourDMM;
BIGTEXT;
    name=YourDMM;
    caption=;
    x=20;
    y=35;
    f=32;
    w=400;
    h=50;
    border=on;
    units=on;
CHECKBOX;
    name=YourDMM;
    caption=Auto Refresh;
    func=FuncAuto;
    param=0.5;
    x=280;
    y=100;

;Optional manual read button (not required):
BUTTON;
    caption=READ;
    action=QUERY;
    device=dev1;
    command=READ?;
    result=YourDMM;
    x=20;
    y=90;
    w=120;
    h=30;
```

EXAMPLE (INSTRUMENT-SPECIFIC OVERLOAD - |TEXT):
Use an overload token that matches the returned value on an overload.
```
DATASOURCE;
    name=Q_K2001;
    device=dev2;
    command=READ?;
    result=K2001Datasource;
    overload=99000000000000000000|text;          ;; text value
    decimal=1;
```

EXAMPLE (INSTRUMENT-SPECIFIC OVERLOAD - |RAW):
Use an overload token that matches the processed value.
```
DATASOURCE;
    name=Q_K2001;
    device=dev2;
    command=READ?;
    result=K2001Datasource;
    overload=9.9E37|raw;
    decimal=1;
```

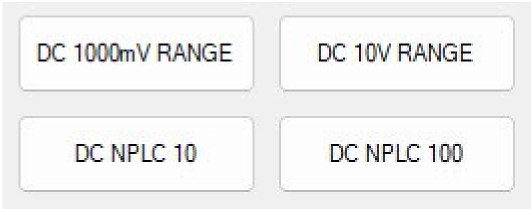
The simplest way to determine which type token (text or raw) you wish to use, query your instrument on the DEVICES tab and observe. Trial and error may be required in the config to achieve OVERLOAD on the USER tab display:
QUERY ASYNC RESPONSE = raw
DECIMAL = text

NOTES:

- DATASOURCE does not send any commands by itself. It only defines what to send when a function (e.g. FuncAuto) triggers it.
- You can define multiple DATASOURCE entries in one file, each producing different result streams.
- Use a naming convention such as name=Q_<instrument> and result=<instrument> to keep roles clear.
- To make overload polarity independent change remove sign, i.e. +9.90000000E+37 to 9.90000000E+37.

BUTTON

FIELD MEANINGS:



BUTTON = Identifies a button entry
caption = Text shown on the button
action = SEND, SENDVALUE, or QUERY
device = dev1, dev2, etc.
command = SCPI/GPIB command (e.g. APPLY DCV 5, APPLY DCV, , USE?, READ?)
sendval = Value source name (for SENDVALUE only)
• This is a logical value name (not necessarily a TEXTBOX)
• Can be driven by a TEXTBOX, DATASOURCE, QUERY, LUA, etc.
result = Result name (QUERY only)
• The instrument reply is written to this named result
• Any control bound to this result name will update (BIGTEXT, CHART, STATSPANEL, HISTORYGRID, LED, TEXTBOX if present)
name = Optional internal ID for the button itself (recommended for debugging / invisibility)

POSITION / SIZE PARAMETERS:

NAMED PARAMETER FORMAT (preferred):
x = Horizontal coordinate
y = Vertical coordinate
w = Width
h = Height

Named parameters may appear in **any order**.
Unspecified parameters use defaults.

ACTION TYPES:
SEND
• Sends command exactly as written.

SENDVALUE
• Sends command followed by the current value from sendval=.

QUERY
• Sends command as a query.
• The instrument reply is written to result=.

EXAMPLE BUTTON:
```
BUTTON;
    caption=READ;
    action=QUERY;
    device=dev2;
    command=READ?;
    sendval=;
    result=K2001Datasource;
    x=20;
    y=120;
    w=80;
    h=30;
```

Sending multiple GPIB commands to instrument from a textarea

```
TEXTAREA;
    name=ScriptBox;
    caption=GPIB Commands:;
    x=700;
    y=200;
    w=200;
    h=80;
    init=:VOLT:DC:DIG MAX|:SENS:VOLT:DC:RANG 1|:VOLT:DC:NPLC 10
BUTTON;
    name=btnSendLines;
    caption=Send;
    action=SENDLINES;
    device=dev1;
    command=;
    sendval=ScriptBox;
    result=;
    x=700;
    y=295;
    w=60;
    h=25;
```

MULTIBUTTON

The MULTIBUTTON control creates a horizontal row of 2 to 10 buttons where **only one button is active at a time**.
Each button sends a different command to the instrument, and the active button reflects the current instrument state.

FIELD MEANINGS:
MULTIBUTTON = Creates a multi-state button group (2–10 buttons)
name = Internal name used by the system
caption = Optional label shown to the left of the button group. If omitted or blank, no label is shown
device = Target device (dev1, dev2, etc.)
items = Comma-separated list of button captions. Number of items must be between 2 and 10
commands = Comma-separated list of commands sent when each button is pressed. Must match the number of items.
If omitted, a single command prefix may be used instead

x = Horizontal coordinate inside the custom GroupBox
y = Vertical coordinate inside the custom GroupBox
w = Width of each button (per button)
h = Height of each button (per button)
gap = Pixel gap between buttons (0 or greater)
oncolor = Colour used for the active button (optional, default = limegreen)
offcolor = Colour used for inactive buttons (optional, default = system default)

DETERMINE (OPTIONAL):

determine = Query used to read back the current instrument state
Format:
determine=<query>||resptext
or
determine=<query>||respnum

Example:
determine=CONF?||resptext

DETMAP (OPTIONAL, MULTIBUTTON ONLY):

detmap = Maps instrument responses to buttons by position.
Used together with determine.
Each entry corresponds to an item by index

Format:
detmap=match1,match2,match3,...

Rules:
• Number of detmap entries must match number of items
• Matching is done left-to-right
• First match wins
• No command is sent during determine (visual update only)

Exact match:
Prefix a detmap entry with = to force exact token matching
This avoids collisions such as RES vs FRES

BEHAVIOUR:

• Clicking a button sends its associated command
• Only one button is active at a time
• Active button is highlighted using oncolor
• Inactive buttons use offcolor
• On config load or auto-refresh, determine + detmap select the active button without sending commands

EXAMPLE – toggle between multiple commands:

```
MULTIBUTTON;
    name=MeasFunc;
    caption=;
    device=dev1;
    items=ACV, DCV, R2W, R4W, FREQ, PER, CONT, DIODE, ACI, DCI;
    commands=:CONF:VOLT:AC, :CONF:VOLT:DC, :CONF:RES, :CONF:FRES, :CONF:FREQ, :CONF:PER, :CONF:CONT, :CONF:DIOD, :CONF:CURR:AC, :CONF:CURR:DC;
    x=450;
    y=500;
    w=50;
    h=28;
    gap=3;
    determine=CONF?||resptext;
    detmap=VOLT:AC, VOLT, RES, FRES, FREQ, PER, CONT, DIOD, CURR:AC, CURR;
```

NOTES:

- MULTIBUTTON is ideal for mode or function selection
- detmap is required when more than two states exist
- Exact matching (=) is recommended when responses overlap, i.e. =FRES
- MULTIBUTTON does not toggle — it selects a specific state

TEXTBOX

FIELD MEANINGS:
TEXTBOX = Identifies a textbox entry
name = Logical name used to bind this textbox to buttons, queries, or other controls
caption = Text label displayed beside the textbox

POSITION / SIZE PARAMETERS:
x= Horizontal coordinate inside the custom GroupBox
y= Vertical coordinate inside the custom GroupBox
w= Width (optional, default applied if omitted)
h= Height (optional, default applied if omitted)

Named parameters may appear in **any order**.
Unspecified parameters use default values.

OPTIONAL PARAMETERS:
readonly = true or false (default = false)

- When true, the textbox is read-only and cannot be edited by the user.
- Commonly used for QUERY result displays.

EXAMPLE TEXTBOXES:

```
TEXTBOX;
   name=YourDMM;
   caption=Set Value:;
   x=200;
   y=20;
   w=120;
   h=25;
TEXTBOX;
   name=TextBoxResult;
   caption=Query Result:;
   x=200;
   y=60;
   w=150;
   h=25;
   readonly=true;
```

CONNECTING A TEXTBOX TO A SENDVALUE BUTTON

EXAMPLE: CONNECTING A TEXTBOX TO A SENDVALUE BUTTON:

STEP 1 — Define the textbox:

```
TEXTBOX;
   name=YourDMM;
   caption=Enter Value:;
   x=200;
   y=20;
   w=120;
   h=25
```

STEP 2 — Define the button:

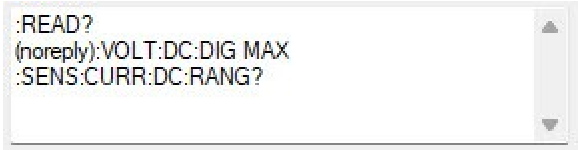
```
BUTTON;
   caption=Set DCV;
   action=SENDVALUE;
   device=dev2;
   command=APPLY DCV, ;
   value=YourDMM;
   x=20;
   y=60;
   w=140;
   h=35;
```

RESULT:
If the user types 1.234, the command sent becomes:
APPLY DCV, 1.234

TEXTAREA (SCRIPT / MULTI-COMMAND)

FIELD MEANINGS:
TEXTAREA = Identifies a multi-line script entry (multiline TextBox)
name = Logical name used to link this TEXTAREA to buttons, timers, or checkboxes
caption = Text label displayed above or beside the textarea

POSITION / SIZE PARAMETERS:
x= Horizontal coordinate inside the custom GroupBox
y= Vertical coordinate inside the custom GroupBox
w= Width of the textarea
h= Height of the textarea



Named parameters may appear in **any order**.
Unspecified parameters use default values.

SCRIPT INITIALISATION:
init = Initial script contents

- Individual commands are separated using the "|" character
- Each entry between "|" becomes one script line

COMMAND RULES:

- Each script line is executed sequentially
- Lines starting with ";" are treated as comments and ignored
- Blank lines are ignored
- TEXTAREA execution **requires the native GPIB engine**
- GpibEngineDev*X* must be set to **native**
- The standalone engine is not supported for TEXTAREA execution

QUERY vs SEND BEHAVIOUR:

- By default, each command line is treated as a QUERY
- A response is expected and logged

To explicitly mark a command as SEND (no response expected), prefix the command with:
(noreply)

COMMAND EXECUTION RULES:

Commands prefixed with (noreply):

- Are sent to the instrument
- Do NOT wait for a response
- Are logged to CSV
- Response field in CSV is blank

Commands without (noreply):

- Are treated as QUERY commands
- A response is awaited
- Response is logged to CSV

CSV LOGGING FORMAT:
Each executed command generates ONE CSV line:

"timestamp","device","command","response"

EXAMPLE:
Mixed QUERY and SEND commands with button and timer:

```
TEXTAREA;
    name=ScriptBox2;
    caption=GPIB Query Commands:;
    x=20;
    y=200;
    w=360;
    h=90;
    init=:READ?|(noreply):VOLT:DC:DIG MAX|:SENS:CURR:DC:RANG?;
BUTTON;
    caption=Send Queries;
    action=QUERIESTOFILE;
    device=dev1;
    script=ScriptBox2;
    result=LastReading;
    x=150;
    y=195;
    w=100;
    h=20;
CHECKBOX;
    name=LastReading;
    caption=Period 5sec;
    func=FuncAuto;
    param=5.0;
    x=260;
    y=198;
```

NOTE:
Maximum period = 2,147,483 seconds
≈ 596 hours
≈ 24.8 days

SAMPLE CSV OUTPUT:
"2025-12-12 17:22:26","dev1",":READ?","0.99999990"
"2025-12-12 17:22:26","dev1",":VOLT:DC:DIG MAX",""
"2025-12-12 17:22:27","dev1",":SENS:CURR:DC:RANG?","0.00000010"
"2025-12-12 17:22:28","dev1",":READ?","0.99999930"
"2025-12-12 17:22:28","dev1",":VOLT:DC:DIG MAX",""
"2025-12-12 17:22:28","dev1",":SENS:CURR:DC:RANG?","0.00000010"

IN THIS EXAMPLE:

- First command queries the instrument
- Second command sends a setup command (no reply)
- Third command queries again
- One button press executes ALL commands in order

TIMER / AUTO EXECUTION:

- TEXTAREA Name may be used as the GroupName for a CHECKBOX with FuncAuto
- When enabled, the entire script is executed periodically
- Query and send behaviour follows the same (noreply) rules

IMPORTANT NOTES:

- TEXTAREA scripts are executed sequentially
- Native engine uses WinGPIB’s internal queued query system
- Standalone engine sends commands directly and must not be used with TEXTAREA

QUERY BUTTONS

STEP 1 — Define a result consumer, any display or logic control may be used, e.g. BIGTEXT, CHART, STATSPANEL, HISTORYGRID, LED.

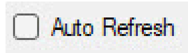
```
BIGTEXT;
    name=YourDMM;
    caption=;
    x=200;
    y=100;
    f=20;
```

STEP 2 — Define a button that performs the query:

```
BUTTON;
    caption=ID Query;
    action=QUERY;
    device=dev2;
    command=*IDN?;
    result=YourDMM;
    x=20;
    y=140;
    w=120;
    h=35;
```

RESULT:
When the button is pressed, the query is sent to the instrument and the response is written to the result name “`YourDMM`”.

CHECKBOX



FIELD MEANINGS:
CHECKBOX = Identifies a checkbox entry
name = Logical name or logical target(s) affected by this checkbox
• May be a single name
• May be a comma-separated list of names
caption = Text shown beside the checkbox
funct = Behaviour assigned to the checkbox
param = Optional parameter for the selected function
x= Horizontal coordinate inside the custom GroupBox
y= Vertical coordinate inside the custom GroupBox

Named parameters may appear in **any order**.
Unspecified parameters use default values.

SUPPORTED CHECKBOX FUNCTIONS:

FuncDecimal

- Converts scientific notation into fixed-point decimal format
  Example conversion:
  "+1.234E-03" → "0.001234"
- Operates on the control specified by Name
- Param must be omitted or left blank

EXAMPLE — Decimal Conversion Checkbox:
CHECKBOX;name=TextBoxResult;caption=decimal:;func=FuncDecimal;x=430;y=440

FuncAuto

- Automatically repeats the most recent QUERY operation
- Uses WinGPIB's internal timer mechanism
- Operates on the logical result stream(s) specified by Name
- Param specifies the repeat interval in seconds
- Fractional seconds are supported (e.g. 0.5, 0.2)
- The minimum interval is 0.001s (1ms)
- Very small intervals or frequent GUI updates (e.g. charts) may impact performance
- Name may specify:
  – A single logical result stream
  – Multiple logical result streams, separated by commas
  (all will be enabled or disabled together)

EXAMPLES — Automatic Query Repeat:

```
CHECKBOX;
    name=YourDMM;
    caption=Auto 0.5s:;
    func=FuncAuto;
    param=0.5;
    x=430;
    y=465;
CHECKBOX;
    name=YourDMM;
    caption=Auto 2s:;
    func=FuncAuto;
    param=2;
    x=430;
    y=490;
CHECKBOX;
    name=YourDMM;
    caption=Auto 5s:;
    func=FuncAuto;
    param=5;
    x=430;
    y=515;
```

EXAMPLE — Automatic Query Repeat (multiple targets):

```
CHECKBOX;
    name=R6581TDatasource,K2001Datasource;
    caption=Auto 2.0s:;
    func=FuncAuto;
    param=2.0;
    x=280;
    y=100;
```

FULL WORKING EXAMPLE:

```
DATASOURCE;
    name=Q_ReadDMM;
    device=dev1;
    command=READ?;
    result=YourDMM;
BUTTON;
    caption=READ;
    action=QUERY;
    device=dev1;
    command=READ?;
    result=YourDMM;
    x=20;
    y=430;
    w=120;
    h=40;
CHECKBOX;
    name=YourDMM;
    caption=Decimal:;
    func=FuncDecimal;
    x=430;
    y=440;
CHECKBOX;
    name=YourDMM;
    caption=Auto 0.5s:;
    func=FuncAuto;
    param=0.5;
    x=430;
    y=465;
```

THIS CREATES:

- A READ query button
- A logical result stream named `YourDMM`
- A checkbox that converts scientific notation to decimal
- A checkbox that automatically re-queries every 0.5 seconds

LABEL

FIELD MEANINGS:
LABEL = Identifies a label entry
caption = Text displayed in the label

POSITION / APPEARANCE PARAMETERS:
x= Horizontal coordinate inside the custom GroupBox
y= Vertical coordinate inside the custom GroupBox
f= Font size (optional, default = 10)

Named parameters may appear in **any order**.
Unspecified parameters use default values.

EXAMPLE LABELS:

```
LABEL;
    caption=R6581T DCV Settings;
    x=20;
    y=10;
    f=14;
LABEL;
    caption=Measurement Options;
    x=20;
    y=160;
    f=12;
LABEL;
    caption=Values are RMS unless stated;
    x=20;
    y=300;
    f=10;
```

DROPDOWN

FIELD MEANINGS:
DROPDOWN = Identifies a dropdown (ComboBox) entry
name = Internal name used by the system
caption = User-facing caption text
device = Target device (dev1, dev2, etc.)

COMMAND PARAMETERS (TWO MODES OF OPERATION):

1. **SINGLE-COMMAND MODE (unchanged functionality)**
   command = GPIB/SCPI command prefix (e.g., :VOLT:DC:NPLC)
   When the user selects an item, the system sends:
   command + " " + SelectedItem

Example:
command = :VOLT:DC:NPLC
Selected = 10
Sent → :VOLT:DC:NPLC 10

2. **MULTI-COMMAND MODE (alternative)**
   commands = Comma-separated list of FULL GPIB commands
   Example:
   commands = :VOLT:DC:NPLC 1,:VOLT:DC:NPLC 10,:VOLT:DC:NPLC 100
   In this mode each dropdown item maps directly to a complete command.
   No prefix is appended.
   No value text is appended.
   Items and commands must align by index.
   Index 1 → first command
   Index 2 → second command
   etc.

OPTIONAL (multi-command mode only)
detmap = value=text,value=text,...
Used only when “commands=” is used.
Provides user-friendly determine/state text mapping.
Ensures UI correctly identifies instrument state.

Example:
detmap = 1=NPLC1,10=NPLC10,100=NPLC100

POSITION / SIZE PARAMETERS:
x = Horizontal coordinate inside the custom GroupBox
y = Vertical coordinate inside the custom GroupBox
w = Width of the dropdown (height is automatic)

ITEM DEFINITIONS:
Items = Comma-separated list of selectable values

OPTIONAL PARAMETERS:
captionpos = Controls how Caption is displayed

captionpos=1 (default):
• Caption is shown left of dropdown
• First dropdown entry is blank

captionpos=2:
• Caption appears as first item inside dropdown
• Acts as placeholder
• No command sent when selected

BEHAVIOUR RULES:

• FIRST dropdown entry NEVER sends a command
captionpos=1 → first entry = blank
captionpos=2 → first entry = caption text

• In SINGLE-COMMAND mode (command=)
Selecting entry N sends:
command + " " + SelectedItem

• In MULTI-COMMAND mode (commands=)
Selecting entry N sends:
commands[N]
No concatenation
No formatting
No modification

• Selecting the first entry does nothing
• No “apply” button required
• Command is sent immediately on selection

EXAMPLES

Standard single-command mode:

```
DROPDOWN;
    name=ComboNPLC;
    caption=NPLC;
    device=dev1;
    command=:VOLT:DC:NPLC ;
    x=400;
    y=40;
    w=120;
    Items=1,10,100;
```

Caption inside dropdown (placeholder):

```
DROPDOWN;
    name=ComboNPLC;
    caption=NPLC;
    device=dev1;
    command=:VOLT:DC:NPLC ;
    x=400;
    y=40;
    w=120;
    Items=1,10,100;
    captionpos=2;
```

Multi-command mode example:

```
DROPDOWN;
    name=ComboNPLC;
    caption=NPLC;
    device=dev1;
    commands=:VOLT:DC:NPLC 1,:VOLT:DC:NPLC 10,:VOLT:DC:NPLC 100;
    x=400;
    y=40;
    w=140;
    Items=1,10,100;
    detmap=1=NPLC1,10=NPLC10,100=NPLC100;
```

Multi-command mode example (multi-line):

```
DROPDOWN;
    name=Combo_Funct;
    caption=MODE;
    device=dev1;
    x=440;
    y=330;
    w=100;
    Items=ACV,DCV,R2W,R4W,FREQ,PER,CONT,DIODE,ACI,DCI;
    commands=:CONF:VOLT:AC,:CONF:VOLT:DC,:CONF:RES,:CONF:FRES,:CONF:FREQ,:CONF:PER,:CONF:CONT,:CONF:DIOD,:CONF:CURR:AC,:CONF:CURR:DC;
    determine=CONF?|resptext;
    detmap=VOLT:AC,VOLT,RES,FRES,FREQ,PER,CONT,DIOD,CURR:AC,CURR:DC;
```

Other examples:

```
DROPDOWN;
    name=ComboRate;
    caption=Rate;
    device=dev1;
    command=:SENS:VOLT:DC:RATE ;
    x=400;
    y=80;
    w=120;
    Items=SLOW,MEDIUM,FAST;

DROPDOWN;
    name=ComboResRange;
    caption=Ω Range;
    device=dev1;
    command=:SENS:RES:RANG ;
    x=400;
    y=120;
    w=140;
    Items=100,1K,10K,100K;
```

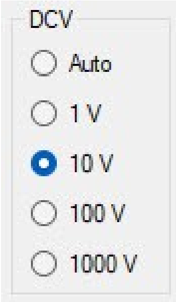
NOTE:

The first dropdown entry always performs NO action, regardless of captionpos or command mode.

RADIOGROUP / RADIO

These controls allow the user to create grouped radio buttons that send GPIB commands immediately when selected.
Each RADIOGROUP creates a framed GroupBox, and each RADIO inside it represents a selectable option.

RADIOGROUP;GroupName=...;Caption=...;x=..;y=..;w=..;h=..



RADIOGROUP FIELD MEANINGS:
RADIOGROUP = Identifies a grouped radio-button container
name = Internal name used by RADIO entries to link to this group
caption = Text displayed on the GroupBox frame

POSITION / SIZE PARAMETERS:
x= Horizontal coordinate inside the custom GroupBox area
y= Vertical coordinate inside the custom GroupBox area
w= Width of the GroupBox
h= Height of the GroupBox

Named parameters may appear in **any order**.
Unspecified parameters use default values.

EXAMPLE RADIOGROUP:
RADIOGROUP;
name=DCRange;
caption=DC Voltage Range;
x=20;
y=200;
w=260;
h=140

RADIO;group=...;caption=...;device=...;command=...;x=..;y=..;scale=...;when=...;determine=...;getrange=.. ;dp=..

RADIO FIELD MEANINGS:
RADIO = Identifies a radio button entry
group = Must match a previously defined RADIOGROUP
caption = Text shown next to the radio button
device = Target device name
command = Command or commands issued when the radio is selected

POSITION PARAMETERS:
x = Horizontal position within the RADIOGROUP
y = Vertical position within the RADIOGROUP

OPTIONAL SCALE PARAMETERS:
scale = Enables manual or automatic scaling behaviour applied to returned values
getrange = Command used to retrieve the currently active instrument range
dp = Controls the number of decimal places displayed

SUPPORTED SCALE MODES:
Fixed scale mode using a constant multiplier
Automatic scaling without querying the instrument
Automatic scaling using instrument range information and a defined range table

EXAMPLE (Advantest R6581 DMM):
scale=auto|:SENS:CURR:DC:RANG?
The meter may return values such as +1.00E-07 (range = 100 nA)

BEHAVIOUR RULES:

- Selecting a RADIO immediately sends its Command
- Only one RADIO in the same GroupName can be active at a time

Scaling behaviour:

- If scale is supplied:
    - WinGPIB stores it as the active conversion factor
    - QUERY results written to the same result will apply this scale
- If scale=auto|query is used:
    - Each QUERY also executes the provided range query
    - Scaling is dynamically updated based on the instrument's response
- If scale is omitted, scale=1 is assumed

RADIO controls:

- Cannot use SENDVALUE or QUERY actions
- Always send their Command immediately when checked

EXAMPLES:
Auto-ranging with instrument query (recommended for DMM's):

```
RADIO;
    group=DCRange;
    caption=Auto;
    device=dev2;
    command=:SENS:VOLT:DC:RANG:AUTO ON;
    x=10;y=20;
    scale=auto;
    getrange=:VOLT:DC:RANG?;                ;; command to get current range from DMM
    range1=0.2,mV,6;                        ;; RANG? reply : units : dp
    range2=2.0,V,7;
    range3=20.0,V,7;
    range4=200.0,V,3;
    range5=1100.0,V,2;
    determine=:VOLT:DC:RANG:AUTO?|1;        ;; I.E. reply would be 1 / 1.0000000 if AUTORANGE MODE is on
    dp=3;                                   ;; fallback only
```

Simple fixed-range radios:

```
RADIO;
    group=DCRange;
    caption=10 V;
    device=dev1;
    command=:SENS:VOLT:DC:RANG 10;
    x=10;
    y=45;
    scale=1;

RADIO;
    group=DCRange;
    caption=100 V;
    device=dev1;
    command=:SENS:VOLT:DC:RANG 100;
    x=10;
    y=70;
    scale=1;
```

Ohms range with fixed scaling:

```
RADIO;
    group=2WRange;
    Caption=100 Ω;
    DeviceName=dev1;
    Command=:SENS:RES:RANG 100;
    x=10;
    y=70;
    scale=0.001;
```

Auto-ranging without meter query:

```
RADIO;
    group=DCIRange;
    Caption=Auto;
    DeviceName=dev1;
    Command=:SENS:CURR:DC:RANG:AUTO ON;
    x=10;
    y=20;
    scale=auto;
```

NOTES:

- Group must be identical for a RADIOGROUP and all its RADIO entries
- Multiple independent RADIOGROUPs may be created anywhere on the form
- scale=auto|<QueryCommand> is the recommended approach for AUTO range modes on multimeters
- Commands are sent immediately when the radio becomes checked

SLIDER

FIELD MEANINGS:
SLIDER = Identifies a slider (TrackBar) entry
name = Internal name used by the system (also used for the value label)
caption = Text displayed as the GroupBox title surrounding the slider
device = Target device (dev1, dev2, etc.)
command = GPIB/SCPI command prefix (e.g., :VOLT:DC:NPLC)

POSITION / SIZE PARAMETERS:
x= Horizontal coordinate inside the custom GroupBox
y= Vertical coordinate inside the custom GroupBox
w= Width of the slider

- The enclosing GroupBox automatically resizes to match the slider width

RANGE / SCALING PARAMETERS:
min= Minimum integer slider value
max= Maximum integer slider value
step= Step size for slider movement
scale= Multiplier applied to the slider value before sending the command

BEHAVIOUR:

- Each SLIDER appears inside its own GroupBox titled with Caption
- A value label appears to the right of the slider
- The value label is BLANK at startup
- The value label updates dynamically while the slider is being moved
- The GPIB command is sent ONLY when the user releases the mouse (MouseUp event)
- No tick marks are shown on the slider
- Background colours follow the GroupBox theme

COMMAND FORMAT:
When the slider is released, the following command is sent:

command + " " + (SliderValue × scale)

EXAMPLES:

SliderValue = 10
scale = 1
→ :VOLT:DC:NPLC 10

SliderValue = 50
scale = 0.1
→ sends 5.0

EXAMPLE SLIDERS:

NPLC selection slider:

```
SLIDER;
    name=SliderNPLC;
    caption=NPLC Setting;
    device=dev1;
    command=:VOLT:DC:NPLC;
    x=400;
    y=40;
    w=220;
    min=1;
    max=100;
    step=1;
    scale=1;
```

→ Generates a slider from 1 to 100 for selecting NPLC

Scaled AC voltage slider:

```
SLIDER;
    name=SliderACV;
    caption=AC Voltage;
    device=dev2;
    command=APPLY ACV;
    x=40;
    y=200;
    w=240;
    min=0;
    max=100;
    step=1;
    scale=0.1;
```

→ Slider range 0–100, scaled to 0–10.0 V
→ Sends: APPLY ACV <volts>

HORIZONTAL RULE (HR)

HR is used to draw a thin horizontal line for visually separating sections.

HR (Named Parameters Only)

HR;x=..;y=..;w=..

FIELD MEANINGS:
HR = Identifies a horizontal rule
x= Horizontal coordinate inside the custom GroupBox
y= Vertical coordinate inside the custom GroupBox
w= Length of the line

BEHAVIOUR:

- Draws a horizontal line
- Height is automatically fixed to **2 pixels**
- No text and no interaction
- Purely cosmetic

EXAMPLE HR:

```
HR;
    x=20;
    y=180;
    w=350;
```

VERTICAL RULE (VR)

VR is used to draw a thin vertical line for visually separating sections.

VR (Named Parameters Only)

VR;x=..;y=..;h=..

FIELD MEANINGS:
VR = Identifies a vertical rule
x= Horizontal coordinate inside the custom GroupBox
y= Vertical coordinate inside the custom GroupBox
h= Height of the line (in pixels)

BEHAVIOUR:

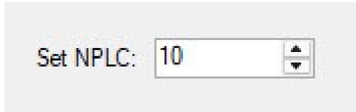
- Draws a vertical line
- Width is automatically fixed to **2 pixels**
- No interaction; purely cosmetic

EXAMPLE VR:

```
VR;
    x=300;
    y=40;
    h=150;
```

SPINNER

FIELD MEANINGS:
SPINNER = Identifies a numeric spinner control (NumericUpDown)
name = Internal name used by the system (and by buttons or queries if required)
caption = Text label displayed beside the spinner
device = Target device (dev1, dev2, etc.)
command = GPIB/SCPI command prefix (e.g., :SENS:VOLT:DC:NPLC)



POSITION / SIZE PARAMETERS:
x= Horizontal coordinate of the spinner caption inside the custom GroupBox
y= Vertical coordinate of the spinner caption inside the custom GroupBox
w= Width of the spinner control

RANGE / SCALING PARAMETERS:
min= Minimum numeric value of the spinner
max= Maximum numeric value of the spinner
step= Increment applied per click or keypress
scale= Multiplier applied to the spinner value before sending

BEHAVIOUR:

- The spinner sends its GPIB command **immediately** when the value changes

- The value sent to the instrument is:

(SpinnerValue × scale)

- The command sent is:

command + " " + ScaledValue

EXAMPLE COMMAND:
:SENS:VOLT:DC:NPLC 10

EXAMPLE SPINNERS:

Basic NPLC spinner:

```
SPINNER;
    name=SpinNPLC;
    caption=Set NPLC:;
    device=dev1;
    command=:SENS:VOLT:DC:NPLC;
    x=40;
    y=300;
    w=80;
    min=1;
    max=100;
    step=1;
    scale=1;
```

Alternative placement:

```
SPINNER;
    name=SpinNPLC;
    caption=Set NPLC:;
    device=dev1;
    command=:SENS:VOLT:DC:NPLC;
    x=400;
    y=80;
    w=80;
    min=1;
    max=100;
    step=1;
    scale=1;
```

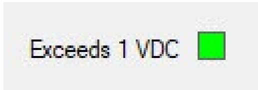
Example with scaling (0.1 resolution):

```
SPINNER;
    name=SpinACV;
    caption=AC Voltage:;
    device=dev2;
    command=APPLY ACV;
    x=40;
    y=200;
    w=80;
    min=0;
    max=200;
    step=1;
    scale=0.1;
```

→ Produces 0.0 → 20.0 V in steps of 0.1 V

→ Sends: APPLY ACV <volts>

LED



FIELD MEANINGS:
LED = Identifies an LED indicator control

name = Internal name of the LED control
Used as a result target for QUERY or DATASOURCE updates.

caption = Text displayed beside the LED

POSITION / APPEARANCE PARAMETERS:
x = Horizontal coordinate inside the custom GroupBox
y = Vertical coordinate inside the custom GroupBox
s = Diameter of the LED circle (in pixels)

OPTIONAL COLOUR PARAMETERS:
on = Colour when LED is ON (default = Lime)
off = Colour when LED is OFF (default = DarkGray)
bad = Colour when the response is invalid or unrecognised (default = Red)

Named parameters may appear in any order.
Unspecified parameters use default values.

BEHAVIOUR:
• The LED listens for updates on the named result specified by Name=
• Any QUERY or DATASOURCE with result=<Name> will drive this LED
• No TEXTBOX is required

LED state is determined from the incoming value as follows:

LED ON when response is:
• 1
• ON
• TRUE
• HIGH
• Any non-zero numeric value

LED OFF when response is:
• 0
• OFF
• FALSE
• LOW
• Empty response

LED BAD when response is:
• Any unrecognised or invalid value

NOTES:
• Numeric responses are parsed using invariant culture
• Whitespace is ignored
• The comparison is case-insensitive
• Non-numeric strings not matching known keywords produce BAD state

EXAMPLES:

Default colours:

```
LED;
    name=LedMAV;
    caption=MAV:;
    x=640;
    y=440;
    s=14;
```

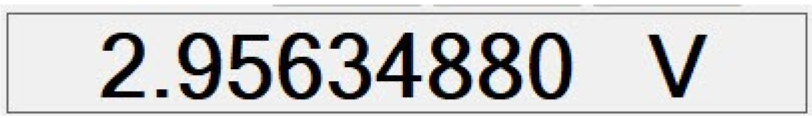
Explicit colours:

```
LED;
    name=LedMAV;
    caption=MAV:;
    x=640;
    y=440;
    s=14;
    on=Lime;
    off=DarkGray;
    bad=Red;
```

Error indicator:

```
LED;
    name=LedERR;
    caption=Error:;
    x=640;
    y=500;
    s=14;
    on=Red;
    off=DarkGray;
```

BIGTEXT



FIELD MEANINGS:
BIGTEXT = Creates a large numeric display box
name = Unique identifier for this display, any QUERY or DATASOURCE with result=<name> will update this BIGTEXT
caption = Optional (currently unused; should be left blank)
units = Controls whether units are shown if supplied by the data source

POSITION / APPEARANCE PARAMETERS:
x = Horizontal coordinate of the BIGTEXT display
y = Vertical coordinate of the BIGTEXT display
f = Font size (default = 28)
w = Width of the surrounding box (default = 200)
h = Height of the surrounding box (default = 60)

OPTIONAL PARAMETERS:
border = Controls display border

- border=on → Draw a frame around the display (default)
- border=off → No border

units = Controls unit display

- units=on → Display units (default)
- units=off → Display numeric value only

BEHAVIOUR:

- On configuration load, BIGTEXT displays "#####" until a value is received
- BIGTEXT automatically links to any control that outputs to the same result
- BIGTEXT normally displays values **with units**, unless units=off is specified
- Units are only shown if supplied by the incoming result value
- Decimal checkbox formatting still applies
- Size and appearance are controlled by f=, w=, h=, and border=

EXAMPLES:

Simple large display (defaults):

```
BIGTEXT;
    name=YourDMM;
    caption=;
    x=160;
    y=220;
    f=32;
```

Custom width:

```
BIGTEXT;
    name=YourDMM;
    caption=;
    x=160;
    y=220;
    f=32;
    w=260;
```

Fully customised size:

```
BIGTEXT;
    name=YourDMM;
    caption=;
    x=160;
    y=220;
    f=32;
    w=260;
    h=80;
```

Border disabled:

```
BIGTEXT;
    name=YourDMM;
    caption=;
    x=160;
    y=220;
    f=32;
    border=off;
```

Units suppressed (numeric only):

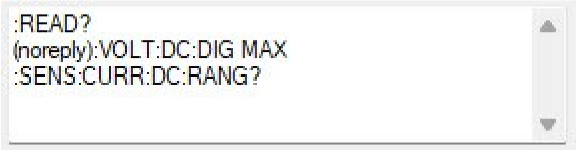
```
BIGTEXT;
    name=YourDMM;
    caption=;
    x=160;
    y=220;
    f=32;
    units=off;
```

2nd BigText as a popup:

```
BIGTEXT;
  name=K34465ADatasource;
  caption=Ian's Workshop;
  x=20;
  y=200;
  f=48;
  w=500;
  h=50;
  border=on;
  units=on;
  format=F6;          ;; normally overriden by RADIO button config if present
  popup=1;
```

TEXTAREA

TEXTAREA;name=...;caption=...;x=..;y=..;w=..;h=..;init=line1|line2|line3|...



FIELD MEANINGS:
TEXTAREA = Creates a multiline text input area
name = Internal name used by SENDLINES buttons or other controls
caption = Text label displayed above the textarea

POSITION / SIZE PARAMETERS:
x= Horizontal coordinate of the textarea caption
y= Vertical coordinate of the textarea caption
w= Width of the textarea
h= Height of the textarea

OPTIONAL PARAMETERS:
init = Initial contents of the textarea

- Multiple lines are separated using the "|" character

BEHAVIOUR:

- The TEXTAREA allows the user to manually edit or replace scripted GPIB commands
- The init= parameter preloads the textarea with one or more lines
- Each "|" creates a new line in the textarea
- Lines beginning with ";" are treated as comments and skipped during execution
- A SENDLINES button can execute each line sequentially

EXAMPLES:

Basic multiline entry box:
```
TEXTAREA;
    name=ScriptBox;
    caption=Startup Script:;
    x=700;
    y=200;
    w=200;
    h=80
```

Preloaded GPIB commands:
```
TEXTAREA;
    name=ScriptBox;
    caption=Startup Script:;
    x=700;
    y=200;
    w=200;
    h=80;
    init=*RST|CONF:VOLT:DC|READ?;
```

Startup script example (using init=):
```
TEXTAREA;
    name=ScriptBox;
    caption=Startup Script:;
    x=700;
    y=200;
    w=200;
    h=80;
    init=:VOLT:DC:DIG MAX|:SENS:VOLT:DC:RANG 1|:VOLT:DC:NPLC 10;
```

IMPORTANT NOTE:
This TEXTAREA type is intended for **editable user scripts** executed via SENDLINES.
For **automated query / logging scripts**, use the dedicated TEXTAREA scripting control documented separately.

SENDLINES BUTTON

FIELD MEANINGS:
BUTTON = Identifies a button entry
name = Unique identifier
action=SENDLINES = Executes multiple GPIB commands from a TEXTAREA
caption = Text shown on the button
device = Target device (dev1, dev2, etc.)
command = Optional text prepended to each command line (usually left blank)
sendval = Name of the TEXTAREA whose lines are executed
result = Not used for SENDLINES (leave blank)

POSITION / SIZE PARAMETERS:
x= Horizontal coordinate inside the custom GroupBox
y= Vertical coordinate inside the custom GroupBox
w= Button width
h= Button height

Named parameters may appear in **any order**.
Unspecified parameters use default values.

BEHAVIOUR:

- When pressed, the button reads all lines from the specified TEXTAREA
- Each non-empty line is sent to the instrument
- Lines beginning with ";" are treated as comments and skipped
- Lines are sent **exactly as typed** unless command is specified
- Commands are executed sequentially
- The user may edit the TEXTAREA live without modifying the configuration file

EXAMPLE — TEXTAREA with associated SENDLINES button:

```
TEXTAREA;
    name=ScriptBox;
    caption=GPIB Commands:;
    x=700;
    y=200;
    w=200;
    h=80;
    init=:VOLT:DC:DIG MAX|:SENS:VOLT:DC:RANG 1|:VOLT:DC:NPLC 10;
BUTTON;
    caption=Send;
    action=SENDLINES;
    device=dev1;
    command=;
    sendval=ScriptBox;
    result=;
    x=700;
    y=295;
    w=60;
    h=25;
```

TOGGLE

FIELD MEANINGS:
TOGGLE = Creates an ON/OFF toggle button
name = Internal name used by the system
caption = Base text displayed on the button
device = Target device (dev1, dev2, etc.)
on = GPIB/SCPI command sent when the toggle is switched ON
off = GPIB/SCPI command sent when the toggle is switched OFF
determine = (Optional) Query used on load to set the initial ON/OFF state
Format:
determine=<query>|<expected>|resptext

Notes:
• Multiple commands can be sent, separate with the § character.
• If "|resptext" is omitted then respnum (numeric compare) is assumed.
• For simple numeric replies you can use:
determine=<query>|<expected>
oncolor = (Optional) Button colour when ON (active)
offcolor = (Optional) Button colour when OFF (inactive)
Supported colour names:
default, limegreen, green, red, orange, yellow, blue

POSITION / SIZE PARAMETERS:
x= Horizontal coordinate inside the custom GroupBox
y= Vertical coordinate inside the custom GroupBox
w= Button width (optional, default applied if omitted)
h= Button height (optional, default applied if omitted)

Named parameters may appear in any order.
Unspecified parameters use default values.

BEHAVIOUR:
• The first click sends the ON command
• The next click sends the OFF command
• Each subsequent click alternates between ON and OFF
• The button illumination changes to show state (colour configurable via oncolor/offcolor)
• If determine is provided, WinGPIB queries the instrument when the config loads and sets the initial state accordingly

EXAMPLES:

Basic output enable toggle:

```
TOGGLE;
    name=TogOutput;
    caption=Output Enable;
    device=dev1;
    on=OUTP ON;
    off=OUTP OFF;
    x=500;
    y=350;
    w=140;
    h=35;
```

Same, but with colours:

```
TOGGLE;
    name=TogOutput;
    caption=Output Enable;
    device=dev1;
    on=OUTP ON;
    off=OUTP OFF;
    x=500;
    y=350;
    w=140;
    h=35;oncolor=limegreen;offcolor=default;
```

Toggle between two NPLC modes (default colours):

```
TOGGLE;
    name=TogNPLC;
    caption=NPLC 1 / 10;
    device=dev1;
    on=:VOLT:DC:NPLC 1;
    off=:VOLT:DC:NPLC 10;
    x=420;
    y=280;
    w=160;
    h=35;
```

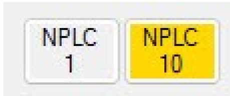
Toggle between two NPLC modes with determine (initial state taken from instrument):

```
TOGGLE;
    name=TogNPLC;
    caption=NPLC 1 / 10;
    device=dev1;
    on=:VOLT:DC:NPLC 1;
    off=:VOLT:DC:NPLC 10;
    x=420;
    y=280;
    w=160;
    h=35;
    determine=:VOLT:DC:NPLC?|1;
    oncolor=limegreen;
    offcolor=default;
```

Sending multiple commands (separated with § symbol):

```
TOGGLE;
  name=TogOnOFF1_2;
  caption=Ch1 && Ch2 ON/OFF;
  device=dev2;
  on=OUTP ON,CH1§OUTP ON,CH2;
  off=OUTP OFF,CH1§OUTP OFF,CH2;
  oncolor=limegreen;
  offcolor=default;
  x=570;
  y=100;
  w=140;
  h=22;
```

TOGGLEDUAL



FIELD MEANINGS:
TOGGLEDUAL = Creates a 2-position “segmented” toggle (two adjacent buttons)
name = Internal name used by the system
caption = (Optional) Label shown to the left of the two buttons
device = Target device (dev1, dev2, etc.)
left = Text shown on the LEFT button (represents the ON state)
right = Text shown on the RIGHT button (represents the OFF state)
on = GPIB/SCPI command sent when the LEFT button is selected (ON)
off = GPIB/SCPI command sent when the RIGHT button is selected (OFF)
determine = (Optional) Query used on load to set the initial LEFT/RIGHT selection
Format:
determine=<query>|<expected>|resptext

Notes:
• If “|resptext” is omitted then respnum (numeric compare) is assumed.
• For simple numeric replies you can use:
determine=<query>|<expected>
oncolor = (Optional) Colour used for the ACTIVE side (whichever side is selected)
offcolor = (Optional) Colour used for the INACTIVE side (the other side)
Supported colour names:
default, limegreen, green, red, orange, yellow, blue

POSITION / SIZE PARAMETERS:
x= Horizontal coordinate inside the custom GroupBox
y= Vertical coordinate inside the custom GroupBox
w= Total width of the two-button control (split automatically into left/right)
h= Button height (optional, default applied if omitted)

Named parameters may appear in any order.
Unspecified parameters use default values.

BEHAVIOUR:
• Clicking the LEFT button:
◦ Sends the ON command
◦ LEFT side becomes ACTIVE (illuminated)
◦ RIGHT side becomes INACTIVE
• Clicking the RIGHT button:
◦ Sends the OFF command
◦ RIGHT side becomes ACTIVE (illuminated)
◦ LEFT side becomes INACTIVE
• The active side is shown by colour (configurable via oncolor/offcolor)
• If determine is provided, WinGPIB queries the instrument when the config loads and sets the initial side accordingly

EXAMPLES:

Basic dual toggle (no determine):

```
TOGGLEDUAL;
    name=TrigMode;
    caption=Trigger:;
    device=dev1;
    left=IMM;
    right=BUS;
    on=TRIG:SOUR IMM;
    off=TRIG:SOUR BUS;
    x=500;
    y=350;
    w=180;
    h=35;
```

Dual toggle with colours:

```
TOGGLEDUAL;
    name=TrigMode;
    caption=Trigger:;
    device=dev1;
    left=IMM;
    right=BUS;
    on=TRIG:SOUR IMM;
    off=TRIG:SOUR BUS;
    x=500;
    y=350;
    w=180;
    h=35;
    oncolor=limegreen;
    offcolor=default;
```

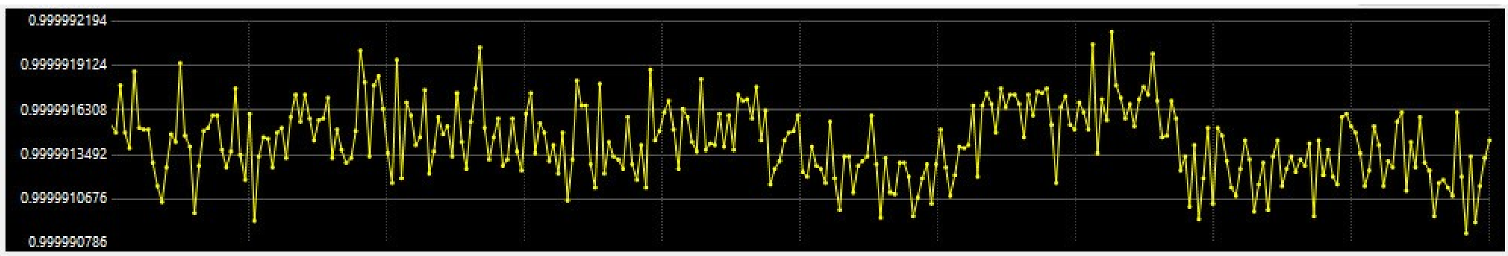
Dual toggle between NPLC 1 and NPLC 10:

```
TOGGLEDUAL;
    name=NPLCmode;
    caption=NPLC:;
    device=dev1;
    left=NPLC 1;
    right=NPLC 10;
    on=:VOLT:DC:NPLC 1;
    off=:VOLT:DC:NPLC 10;
    x=700;
    y=400;
    w=180;
    h=35;
    oncolor=limegreen;
    offcolor=default;
```

Dual toggle with determine (initial state taken from instrument):

```
TOGGLEDUAL;
    name=NPLCmode;
    caption=NPLC:;
    device=dev1;
    left=NPLC 1;
    right=NPLC 10;
    on=:VOLT:DC:NPLC 1;
    off=:VOLT:DC:NPLC 10;
    x=700;
    y=400;
    w=180;
    h=35;
    determine=:VOLT:DC:NPLC?|1;
    oncolor=limegreen;
    offcolor=default;
```

CHART



FIELD MEANINGS:
CHART = Identifies a chart control.

Name = Internal name of the chart control.

Caption = Text label displayed above the chart.

Target = Name of the result value to consume.
Any DATASOURCE or QUERY with result=<target> will feed this chart.

POSITION / SIZE PARAMETERS:
x = Horizontal position of the chart (top-left).

y = Vertical position of the chart (top-left, caption drawn above).

w = Width of the chart area.

h = Height of the chart area.

Y-AXIS PARAMETERS:
ymin = Optional fixed minimum Y-axis value (ignored if autoscale=yes).

ymax = Optional fixed maximum Y-axis value (ignored if autoscale=yes).

autoscale = Controls Y-axis scaling.

autoscale = no (default):
• Uses ymin/ymax if supplied
• If ymin/ymax are omitted, Y-axis auto-ranges once and then remains fixed

labelsize = font size for Y-axis scale labels

autoscale = yes:
• Y-axis min/max are recalculated on every update
• A small padding is automatically added for readability

POPUP or FIXED:
popup = 1 enables a pop-out CHART, otherwise 0 for the fixed version, x, y, w, & h can be used for both popup & fixed.

TRACE / BUFFER PARAMETERS:
xstep = Step between samples on the X-axis (time or index increment).

maxpoints = Rolling buffer length (number of most recent samples retained).

linewidth = Line thickness (1, 2, 3, …).

color = Trace colour name (e.g. Yellow, Lime, Red).

INNER PLOT AREA (PERCENTAGES):
innerX=
Left margin of plot area (0–100 %).

innerY=
Top margin of plot area (0–100 %).

innerW=
Width of plot area (1–100 %).

innerH=
Height of plot area (1–100 %).

NOTES:

Set x/y first (top-left origin), then adjust innerX / innerY / innerW / innerH.

BEHAVIOUR:
- CHART listens for updates on the named result specified by target=.
- No TEXTBOX is required.
- DATASOURCE or QUERY buttons may supply the data.

DATA INTERPRETATION:
If the result value contains a SINGLE numeric value:
→ The value is appended as a new point (streaming / rolling mode).

If the result value contains MULTIPLE numeric values:
→ Values are parsed as a complete buffer.
→ X-axis values are generated as: index × xstep.

SUPPORTED NUMERIC SEPARATORS:
- Comma (,)
- Semicolon (;)
- Space
- Newline (CR/LF)

CHART BEHAVIOUR:
- Only the last maxpoints samples are retained.
- The X-axis automatically scrolls to keep the most recent data visible.
- Grid lines are shown on a dark background.
- Y-axis uses a smaller font with no tick marks.
- X-axis shows grid only (no labels).
- If color= is invalid or omitted, the trace defaults to Yellow.

EXAMPLES:
Fixed Y-axis range:

```
CHART;
    name=ChartDMM;
    caption=DCV;
    target=YourDMM;
    x=20;
    y=355;
    w=990;
    h=160;
    ymin=0.0;
    ymax=1.0;
    xstep=0.1;
    maxpoints=15;
    color=Yellow;
    linewidth=1;
    innerX=7;
    innerY=5;
    innerW=92;
    innerH=92;
```

Auto-scaling Y-axis:

```
CHART;
    name=ChartDMM;
    caption=DCV;
    target=YourDMM;
    x=20;
    y=355;
    w=990;
    h=160;
    autoscale=yes;
    xstep=0.1;
    maxpoints=100;
    color=Lime;
    linewidth=1;
    innerX=7;
    innerY=5;
    innerW=92;
    innerH=92;
```

Auto-scaling Y-axis with popout chart and label size set:

```
CHART;
    name=ChartDMM;
    caption=DCV;
    target=YourDMM;
    x=20;
    y=355;
    w=990;
    h=160;
    autoscale=yes;
    xstep=0.1;
    maxpoints=100;
    color=Lime;
    linewidth=1;
    innerX=7;
    innerY=5;
    innerW=92;
    innerH=92;
    popup=1;
    labelsize=10;
```

RESET CHART BUTTON (CLEARCHART)

The RESET button is implemented as a normal BUTTON with a special action.

BUTTON;caption=...;action=CLEARCHART;target=...;x=..;y=..;w=..;h=..

FIELD MEANINGS:
BUTTON
Identifies a button control

caption
Text shown on the button (e.g. “Reset Chart”)

action=CLEARCHART
Special action that clears chart data

target
Name of the chart(s) to clear
Must match the chart name= used in the CHART definition
Multiple charts separate with the § character, i.e. ChartDMM§ChartDMM2

POSITION / SIZE PARAMETERS:
x = Horizontal coordinate inside the custom GroupBox
y = Vertical coordinate inside the custom GroupBox
w = Button width
h = Button height

Named parameters may appear in any order.
Unspecified parameters use default values.

BEHAVIOUR:
• When clicked, all data points from the chart are cleared
• The chart resumes plotting from an empty buffer when new data arrives
• CLEARCHART does not send any GPIB commands
• CLEARCHART does not use dev1 / dev2 instrument devices

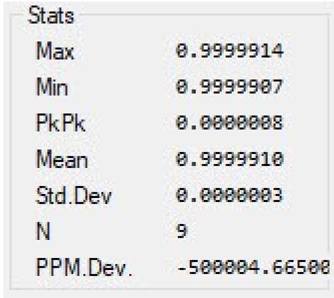
IMPORTANT NOTES:
• The referenced CHART must already exist in the same configuration file
• The CHART must be created before the CLEARCHART button
• All series created by the CHART definition are cleared

EXAMPLE:

Chart definition:
```
CHART;
    name=ChartDMM;
    caption=DCV;
    target=YourDMM;
    x=20;
    y=250;
    w=250;
    h=120;
    autoscale=yes;
    maxpoints=50;
    color=Lime;
    linewidth=2;
```

Reset button:
```
BUTTON;
    caption=Reset Chart;
    action=CLEARCHART;
    target=ChartDMM;
    x=280;
    y=250;
    w=100;
    h=30;
```

STATSPANEL / STAT



FIELD MEANINGS:

STATSPANEL = Identifies a statistics panel control.

name = Internal name of the statistics panel.
Used by STAT definitions and RESETSTATS.

caption = Text shown as the GroupBox title.

target = Name of the result value to consume.
Any DATASOURCE or QUERY with result=<target> will feed this statistics panel.
No TEXTBOX is required.

POSITION / SIZE PARAMETERS:
x = Horizontal coordinate inside the custom GroupBox.

y = Vertical coordinate inside the custom GroupBox.

w = Width of the statistics panel.

h = Height of the statistics panel.

fontsize = Font size

colleftpos = Relative left column position to default

colrightpos = Relative right column position to default

FORMAT PARAMETER:
format = Optional default numeric format for all statistics in this panel.

Supported format values:
• F<n> → Fixed-point decimal (e.g. F6, F7)
• G<n> → General format, automatic decimal/scientific (e.g. G6, G7)
• E<n> → Scientific notation (e.g. E6, E7)

If omitted:
format=G6

BEHAVIOUR:
• STATSPANEL listens for updates on the named result specified by target=.
• Each new numeric value is added as a sample.
• Non-numeric text is ignored.
• Statistics accumulate continuously.
• Accumulation continues until RESETSTATS is pressed.
• Formatting defaults to the panel format unless overridden per STAT row.

STAT (Statistic Row Definition)

STAT;panel=...;label=...;func=...;ref=...;units=...

FIELD MEANINGS:
STAT = Defines a single statistic row inside a STATSPANEL.
panel = Name of the STATSPANEL this statistic belongs to.
label = Text shown on the left (e.g. Min, Max, Mean).
func = Statistic function to compute.

OPTIONAL PARAMETERS:
ref = Reference value used by PPM calculations.

- ref=MEAN → Use running mean (default)
- ref=FIRST → Use first sample
- ref=<number> → Use fixed numeric reference
- ref=@<controlName> → Use the current value of a UI control (typically a TEXTBOX).
  Example: ref=@RefPPM reads the value from control RefPPM.

units = Override numeric format for this statistic only.

- units=decimal → Fixed-point decimal (F-format)
- units=scientific → Scientific notation (E-format)

If omitted:
- Uses STATSPANEL format.

SUPPORTED func VALUES:
MIN → Minimum value seen
MAX → Maximum value seen
PKPK → Peak-to-peak (Max − Min)
MEAN → Running mean
STD → Sample standard deviation (N−1)
LAST → Most recent value
COUNT → Number of samples collected
PPM → PPM deviation of LAST from reference

PPM DEFINITION:
PPM = ((LAST − Reference) / Reference) × 1,000,000

NOTES:
STD – Deviation shows how much the readings are spreading or fluctuating.
Low values indicate a stable signal; high values indicate noise or settling.

PPM – Deviation shows how far the current reading is from a reference value,
expressed in parts-per-million. Useful for observing drift or offset.

RESETSTATS BUTTON (Named Parameters Only)

BUTTON;caption=...;action=RESETSTATS;device=...;x=..;y=..;w=..;h=..

FIELD MEANINGS:
BUTTON = Identifies a button control.

action=RESETSTATS
Clears all accumulated statistics in a STATSPANEL.

caption = Text shown on the button.

device = Name of the STATSPANEL to reset.
Multiple Stats panels separate with the § character, i.e. Stats1§Stats2;

BEHAVIOUR:
- Clears all accumulated samples and statistics for the specified panel.
- Statistics restart from empty on the next incoming value.
- RESETSTATS does not send any GPIB commands.

EXAMPLE:

Statistics Panel with Reset Button & Textbox control for user to set ref value:

```
STATSPANEL;
  name=Stats1;
  caption=Stats;
  target=K34465ADatasource;
  x=20;
  y=220;
  w=160;
  h=145;
  format=F7;
STAT;
  panel=Stats1;
  label=Max;
  func=MAX;
  units=decimal
STAT;
  panel=Stats1;
  label=Min;
  func=MIN;
  units=decimal
STAT;
  panel=Stats1;
  label=PkPk;
  func=PKPK;
  units=decimal
STAT;
  panel=Stats1;
  label=Mean;
  func=MEAN;
  units=decimal
STAT;
  panel=Stats1;
  label=Std.Dev;
  func=STD;
  units=decimal
STAT;
  panel=Stats1;
  label=N;
  func=COUNT;
  units=decimal
STAT;
  panel=Stats1;
  label=PPM.Dev.;
  func=PPM;
  ref=@RefPPM;
  units=decimal
TEXTBOX;
  name=RefPPM;
  caption=;
  x=181;
  y=345;
  w=69;
  h=22;
  default=1.000000
LABEL;
  name=RefPPMlabel;
  caption=PPM Dev Ref;
  x=256;
  y=348;
  f=7
BUTTON;
  name=BtnResetStats;
  caption=Reset Stats;
  action=RESETSTATS;
  device=Stats1;
  command=;
  value=;
  result=;
  x=185;
  y=322;
  w=70;
  h=22;
```

HISTORYGRID

Value	Min	Max	PkPk	Mean	Std	PPM	Count
0.9999908	-0.0000072	9.9999899	9.9999971	2.2562538	2.4288069	-556791.5...	315
0.9999908	-0.0000072	9.9999899	9.9999971	2.2602547	2.4316438	-557576.0...	314
0.9999909	-0.0000072	9.9999899	9.9999971	2.2642811	2.4344889	-558362.7...	313
0.9999909	-0.0000072	9.9999899	9.9999971	2.2683333	2.4373422	-559151.7...	312
0.9999911	-0.0000072	9.9999899	9.9999971	2.2724116	2.4402038	-559942.7...	311
0.9999907	-0.0000072	9.9999899	9.9999971	2.2765161	2.4430736	-560736.4...	310
0.9999910	-0.0000072	9.9999899	9.9999971	2.2806473	2.4459516	-561531.9...	309
0.9999908	-0.0000072	9.9999899	9.9999971	2.2848053	2.4488379	-562329.9...	308
0.9999913	-0.0000072	9.9999899	9.9999971	2.2889903	2.4517325	-563129.9...	307
0.9999910	-0.0000072	9.9999899	9.9999971	2.2932027	2.4546355	-563932.5...	306
0.9999911	-0.0000072	9.9999899	9.9999971	2.2974428	2.4575467	-564737.3...	305
0.9999911	-0.0000072	9.9999899	9.9999971	2.3017107	2.4604663	-565544.4...	304
0.9999907	-0.0000072	9.9999899	9.9999971	2.3060068	2.4633942	-566353.9...	303
0.9999909	-0.0000072	9.9999899	9.9999971	2.3103314	2.4663305	-567165.6...	302

FIELD MEANINGS:
HISTORYGRID = Identifies a rolling history grid control

name = Internal name of the history grid
Used by CLEARHISTORY and internal routing.

caption = Text label displayed above the grid

result = Name of the result value supplying live numeric data
Any DATASOURCE or QUERY with `result=<resultname>` will feed this grid when `HISTORYGRID;...;result=<resultname>` matches.

POSITION / SIZE PARAMETERS:
x = Horizontal position of the grid (top-left; caption drawn above)
y = Vertical position of the grid (top-left)
w = Width of the grid
h = Height of the grid

DATA / FORMAT PARAMETERS:
maxrows = Maximum number of rows to keep (rolling buffer; newest at top)

format = Optional numeric display format applied to all numeric columns
• F<n> → Fixed decimal places (e.g. F6, F7)
• G<n> → Significant digits (e.g. G6, G7)
• E<n> → Scientific notation (e.g. E6, E7)

If omitted:
format=F7

cols = Comma-separated list of column names to display, in order

SUPPORTED COLUMN NAMES:
Value
Time
Min
Max
PkPk
Mean
Std
PPM
Count

PPM REFERENCE PARAMETER:
ppmref= = Reference used for PPM deviation calculation

Supported values:
• MEAN → PPM relative to running mean (default)
• FIRST → PPM relative to first captured value
• <number> → Fixed numeric reference value

BEHAVIOUR:
• HISTORYGRID listens for updates on the named result specified by result=
• Each new numeric value is added as a new row at the TOP of the grid
• When maxrows is exceeded, the oldest row is removed
• The first column (Value) is displayed in bold to highlight the live reading
• The Time column stores the capture timestamp (DateTime)
• All statistics (Min, Max, PkPk, Mean, Std, PPM, Count) are calculated live
• Numeric formatting uses the specified format and invariant culture
• A reset button can be included to clear the grid

NOTES:
• Numeric columns must be bound to numeric data types (Double / Integer) for F / G / E formatting to function correctly
• Column widths are fixed unless AutoSizeColumnsMode is enabled

EXAMPLES:
Basic rolling history (newest at top):

```
HISTORYGRID;
    name=Hist1;
    caption=History;
    result=YourDMM;
    x=200;
    y=150;
    w=500;
    h=180;
    maxrows=15;
BUTTON;
    caption=Clear History;
    action=CLEARHISTORY;
    device=Hist1;
    x=422;
    y=342;
    w=95;
    h=22;
```

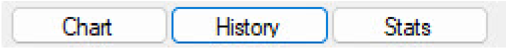
Fixed decimal format with custom columns:

```
HISTORYGRID;
    name=Hist1;
    caption=History;
    result=YourDMM;
    x=200;
    y=150;
    w=500;
    h=180;
    maxrows=20;
    format=F7;
    cols=Value,Min,Max,PkPk,Mean,Std,PPM,Count;
BUTTON;
    caption=Clear History;
    action=CLEARHISTORY;
    device=Hist1;
    x=422;
    y=342;
    w=95;
    h=22;
```

PPM deviation relative to FIRST value:

```
HISTORYGRID;
    name=Hist1;
    caption=History;
    result=YourDMM;
    x=200;
    y=150;
    w=500;
    h=180;
    maxrows=30;
    format=F6;
    cols=Value,Time,PPM,Count;
    ppmref=FIRST;
BUTTON;
    caption=Clear History;
    action=CLEARHISTORY;
    device=Hist1;
    x=422;
    y=342;
    w=95;
    h=22;
```

INVISIBILITY



FIELD MEANINGS:
INVISIBILITY = Defines a visibility-toggle function for one or more controls
targets = Control name(s) to be shown/hidden
• Single control name: targets=ChartDMM
• Multiple controls: targets=ChartDMM,Hist1,Stats1
func = Logical toggle function name (used as a result stream name)
default = Optional startup visibility state for the targets
• default=on → targets start Visible=True
• default=off → targets start Visible=False

SUPPORTED TARGETS:
Any dynamically created control with a name, including:
• CHART
• HISTORYGRID
• STATSPANEL
• BIGTEXT
• TEXTBOX / LABEL / BUTTON / DROPDOWN / SLIDER / LED
• Any other dynamic control created by the config

BEHAVIOUR:
• INVISIBILITY creates no UI by itself
• It registers a toggle function name (func=) and the associated targets
• When the func name is “triggered” (via result=), the target control(s) toggle Visible
• If a control becomes visible, it is brought to the front
• No GPIB/SCPI commands are sent
• Visibility state is not persisted between application runs
• If a target name is not found at runtime, it is silently ignored
• Matching is case-insensitive

DEFAULT VISIBILITY:
• If default= is supplied, targets are set to that visibility state at startup
• If default= is omitted, targets start with their normal default Visible=True

ASSOCIATED CONTROLS / COMPOSITE TARGETS:
• INVISIBILITY operates strictly on the named targets
• Functionally related controls (e.g. a STATSPANEL and its Reset button) are not implicitly linked
• Any related controls that should hide/show together must all be listed in targets=
• Alternatively, place related controls inside a common container (e.g. GROUPBOX) and target that container instead

USAGE WITH BUTTONS (recommended pattern):
Use a BUTTON that writes to result=funcName (no device/command required)
BUTTON;caption=...;result=<funcName>;x=..;y=..;w=..;h=..

EXAMPLES:

Individual hide/show with defaults:

```
;Function to provide invisibility - Individual ID's
INVISIBILITY;
    targets=ChartDMM;
    func=ChartDMMhide;
    default=on;
INVISIBILITY;
    targets=Hist1;
    func=Hist1hide;
    default=off;
INVISIBILITY;
    targets=Stats1;
    func=Stats1hide;
    default=on;
BUTTON;
    caption=Chart;
    result=ChartDMMhide;
    x=150;
    y=10;
    w=80;
    h=20;
BUTTON;
    caption=History;
    result=Hist1hide;
    x=230;
    y=10;
    w=80;
    h=20;
BUTTON;
    caption=Stats;
    result=Stats1hide;
    x=310;
    y=10;
    w=80;
    h=20;
```

Hide/show multiple controls with one button:

```
INVISIBILITY;
    targets=ChartDMM,Hist1,Stats1;
    func=HideAll;
    default=on
BUTTON;
    caption=Toggle All;
    result=HideAll;
    x=150;
    y=40;
    w=120;
    h=20;
```

Explicit grouping of related controls:

```
INVISIBILITY;
    targets=Stats1,BtnResetStats;
    func=Stats1hide;
    default=on;
BUTTON;
    caption=Stats;
    result=Stats1hide;
    x=310;
    y=10;
    w=80;
    h=20;
```

Container-based hiding (preferred when available):

```
GROUPBOX;
    name=StatsGroup;
    caption=Stats;
    x=20;
    y=210;
    w=260;
    h=180;
STATSPANEL;
    name=Stats1;
    parent=StatsGroup;…
BUTTON;
    name=BtnResetStats;
    parent=StatsGroup;…

INVISIBILITY;
    targets=StatsGroup;
    func=Stats1hide;
    default=on;
```

NOTES:

- INVISIBILITY definitions may appear anywhere in the configuration file
- If you hide a container control (e.g. a Panel/GroupBox), all child controls hide with it
- default= applies to all targets listed on that INVISIBILITY line

TRIGGER

Current implementations may also use the simplified event-driven form:
; TRIGGER;name=...;when=...;do=...

FIELD MEANINGS:

TRIGGER = Defines a conditional automation rule evaluated periodically or event-driven
name = Unique trigger name (used for logging and future enable/disable control)
period = Evaluation interval in seconds (floating point allowed)
(Note: current implementation evaluates triggers on the main system timer)
if = Boolean condition expression that must evaluate TRUE
(Alias: when=)
then = One or more actions executed when the trigger fires
(Alias: do=)
cooldown = Minimum time (seconds) before the trigger may fire again
(reserved for future use)
need = Number of consecutive TRUE evaluations required before firing
(reserved for future use)
enabled = Initial enabled state (yes/no, true/false, 1/0)
(reserved for future use)

Named parameters may appear in any order.
Unspecified parameters use default values.

SUPPORTED CONDITION SOURCES:
- stats:PanelName.mean
- stats:PanelName.std
- stats:PanelName.min
- stats:PanelName.max
- stats:PanelName.pkpk
- stats:PanelName.count
- stats:PanelName.PPM
- num:TextBoxName
- bignum:BigTextName
- ResultName (auto-resolved numeric result)
- LedName (LED state as numeric 0 or 1)

SUPPORTED OPERATORS:
- Comparisons: > < >= <= == !=
- Single "=" accepted and normalized to "=="
- Boolean aliases: ON / TRUE / HIGH → 1, OFF / FALSE / LOW → 0
- Logical: AND OR NOT
- Parentheses for grouping
(Logical operators and grouping reserved for future use)

SUPPORTED ACTIONS (then=... / do=...):
- fire:ButtonName → Simulates a button press
- resetstats:PanelName → Resets a statistics panel
- clearchart:ChartName → Clears a chart
- togglevis:Control1,... → Toggles visibility
- show:Control1,... → Forces visible
- hide:Control1,... → Forces hidden
- enabletrig:TriggerName → Enables another trigger
- disabletrig:TriggerName → Disables another trigger
- LED:LedName=ON|OFF|BAD → Sets LED state
- SET:ResultName=Number → Writes to a result stream
- SEND:dev:command → Fire-and-forget GPIB send
- QUERY:dev:command->ResultName → Query instrument and store result

Multiple actions may be chained using | (pipe separator).

BEHAVIOUR:
- Triggers are evaluated on the main system timer

- Triggers are edge-triggered by default
- When the IF / WHEN expression transitions from FALSE to TRUE, the trigger fires
- Triggers do not continuously fire while the condition remains TRUE
- To follow a condition continuously (e.g. LED ON/OFF), paired triggers should be used
- If the IF expression is TRUE, an internal hit counter may increment
- When hit counter reaches "need", the trigger fires
- After firing:
  - hit counter resets
  - cooldown timer starts (if specified)
- If IF expression becomes FALSE, hit counter resets to zero
- Triggers run continuously while enabled

(Some hit-count, cooldown, and enable features are reserved for future expansion.)

EXAMPLES:

- **Simple numeric threshold:**

```
TRIGGER;
    name=HighValue;
    period=0.5;
    if=num:TextBoxTest > 5;
    then=fire:BtnHello;
    need=1;
    enabled=yes;
```

Explanation:
Checks twice per second. Fires when TextBoxTest exceeds 5.

Equivalent event-driven form:

```
TRIGGER;
    when=TextBoxTest > 5;
    do=SEND:dev1:*CLS;
```

- **Time-qualified condition (sustained state):**

```
TRIGGER;
    name=StableHigh;
    period=1.0;
    if=stats:Stats1.std > 0.0005;
    then=fire:BtnHello;
    need=10;
    enabled=yes;
```

Explanation:
Evaluates once per second. Std.Dev must remain above 0.0005 for 10 consecutive seconds before the trigger fires.

- **Settling detector (typical calibration use):**

```
TRIGGER;
    name=Settled;
    period=0.5;
    if=stats:Stats1.std < 0.00001 AND stats:Stats1.count >= 20;
    then=fire:BtnWriteCal;
    cooldown=5;
    need=5;
    enabled=yes;
```

Explanation:
Used when monitoring a reference instrument (e.g. 3458A).
Once enough samples exist and variability drops below threshold, the calibration write command is issued.
Cooldown prevents immediate re-triggering.

- **Manual enable via checkbox:**

```
TRIGGER;
    name=TrigStdHigh;
    period=1.0;
    if=stats:Stats1.std > 0.0005;
    then=fire:BtnHello;need=10;
    enabled=no;
CHECKBOX;
    caption=Enable Trigger;
    func=FuncTrigEnable;
    param=TrigStdHigh;
    x=220;
    y=400;
```

Explanation:
Trigger is disabled on load. User explicitly enables it when ready, preventing accidental automation during setup.

• **Multi-action trigger:**

```
TRIGGER;
    name=Alarm;
    period=0.2;
    if=num:TempValue > 80;
    then=fire:BtnStop|show:AlarmPanel;cooldown=10;
    need=3;
    enabled=yes;
```

Explanation:

If temperature exceeds 80 for three consecutive checks, measurement stops and an alarm panel is shown.

• **Stability detector**

```
BUTTON;
    name=BtnHello;
    caption=TriggerOnStdDev;
    action=SEND;
    device=dev1;
    command=:READ?;
    x=220;
    y=420;
    w=120;
    h=30;
TRIGGER;
    name=TrigStdHigh;
    period=1.0;
    if=stats:Stats1.std > 0.0005;
    then=fire:BtnHello|resetstats:Stats1;
    cooldown=2;
    need=10;
    enabled=no;
TEXTBOX;
    name=TriggerLog;
    caption=Trigger Log:;
    x=220;
    y=455;
    w=500;
    h=220;
    readonly=true;
CHECKBOX;
    caption=Enable Trigger;
    func=FuncTrigEnable;
    param=TrigStdHigh
    ;x=220;
    y=400;
```

Explanation:

Monitors measurement stability using running standard deviation.

When Std.Dev stays ABOVE 0.0005 continuously for approximately 10 seconds, READ? command is issued and the statistics are reset.

Trigger activity is recorded in a dedicated log window.

More detail:

Evaluates once per second (period=1.0)

Each evaluation where Std.Dev > 0.0005 counts as one hit after 10 consecutive hits (need=10 ≈ 10 seconds), the trigger fires the trigger then:

Fires the TriggerOnStdDev button (sending :READ?)

Resets Stats1

Logs the event to textbox

Trigger is disabled on load and must be enabled manually via checkbox

• **Trigger chain**

Trigger chain demo (with LED status):
Stage 1 waits for Std.Dev to rise above threshold (LED goes RED)
Stage 2 then waits for Std.Dev to settle back low (LED goes GREEN)

```
CHECKBOX;caption=Enable Std.Dev Trigger;func=FuncTrigEnable;param=TrigStdHigh;x=380;y=480
LED;name=LedTrig;caption=Trig State:;x=400;y=420;s=14;on=Lime;off=DarkGray;bad=Red
BUTTON;name=BtnHello;caption=TriggerAction;action=SEND;device=dev1;command=:READ?;x=370;y=440;w=140;h=30
```

```
; ---------- Stage 1 ----------
TRIGGER;
    name=TrigStdHigh;period=1.0;
    if=stats:Stats1.std > 0.0005;
    then=LED:LedTrig=BAD|fire:BtnHello|enabletrig:TrigStdLow;
    cooldown=2;
    need=10;
    enabled=yes;


; ---------- Stage 2 ----------
TRIGGER;
    name=TrigStdLow;
    period=1.0;if=stats:Stats1.std < 0.0002;
    then=LED:LedTrig=ON|fire:BtnHello|disabletrig:TrigStdLow;
    cooldown=2;
    need=5;
    enabled=no;
TEXTAREA;
    name=TriggerLog;
    caption=Trigger Log:;
    x=520;
    y=400;
    w=420;
    h=100;
    readonly=true;
```

NOTES:

- TRIGGER may send GPIB directly using SEND or QUERY
- TRIGGER may also reuse existing buttons for identical behaviour
- period × need ≈ minimum time condition must remain TRUE
- Triggers are evaluated independently and may run concurrently

CALC — Derived / Computed Result Streams

CALC allows you to define **derived numeric result streams** based on existing result names (from DATASOURCE, QUERY buttons, or other CALC outputs).
The computed result behaves exactly like a normal result stream and may be displayed or consumed by any control that accepts a result= or name= binding.

CALC;result=...;expr=...

FIELD MEANINGS:
CALC = Defines a calculated result stream
result = Name of the output result produced by this calculation
expr = Arithmetic expression defining how the result is computed
    expr may optionally include an output format suffix using the | character
    Syntax: expr=<expression>|<format>

SUPPORTED INPUTS:
• Any existing result name produced by:
– DATASOURCE
– QUERY buttons
– Other CALC definitions
• Numeric literals (e.g. 1, 0.5, 1000000)

SUPPORTED OPERATORS:
• Addition: +
• Subtraction: −
• Multiplication: *
• Division: /
• Parentheses for grouping: ( )

NUMERIC FORMAT:
• All calculations are performed using Double precision
• Decimal point is always . (invariant culture)

BEHAVIOUR:
• CALC does not communicate with any instrument
• CALC automatically re-evaluates whenever any input result changes
• Calculated results propagate to all bound controls (BIGTEXT, CHART, STATSPANEL, HISTORYGRID, LED, etc.)
• CALC results may be chained (one CALC may depend on another CALC)
• If a required input result is not yet available, evaluation is deferred until it becomes valid

BINDING CALC OUTPUTS:
The calculated result name may be used exactly like a normal result:

BIGTEXT;name=<result>;...
CHART;target=<result>;...
STATSPANEL;target=<result>;...
HISTORYGRID;result=<result>;...
LED;name=<result>;...

EXAMPLE 1 — Simple scaling (divide or multiply)

```
; Create /1000 parameter from existing R6581TDatasource DATASOURCE
CALC;
    result=R6581T_mV;
    expr=R6581TDatasource*1000;
BIGTEXT;
    name=R6581T_mV;
    caption=;
    x=200;
    y=245;
    f=16;
    w=170;
    h=30;
    border=on;
    units=off;
LABEL;
    caption=Divided by 1000;
    x=370;
    y=255;
    f=12;
```

This example:
• Uses R6581TDatasource as the input
• Multiplies the value by 1000
• Displays the derived value in a BIGTEXT control

EXAMPLE 2 — PPM deviation from a fixed reference

```
; Create PPM drift from reference 1.000000 from existing R6581TDatasource DATASOURCE
CALC;
    result=R6581T_Ref;
    expr=1.0000000+0R6581TDatasource;
CALC;
    result=R6581T_PPM;
    expr=(R6581TDatasource-R6581T_Ref)/R6581T_Ref1000000;
BIGTEXT;
    name=R6581T_PPM;
    caption=;
    x=200;
    y=280;
    f=16;
    w=170;
    h=30;
    border=on;
    units=off;
    format=F2;
LABEL;
    caption=PPM Dev from 1.000000;
    x=370;
    y=290;
    f=12;
```

Explanation:
• R6581T_Ref defines a stable reference value (1.0000000)
• The +0*R6581TDatasource ensures the reference is evaluated after data is present
• R6581T_PPM computes PPM deviation relative to the reference
• The BIGTEXT control displays the result with two decimal places

EXAMPLE 3 — Fixed-format derived value (raw formatted text)

```
; Create fixed-decimal readout from a raw/scientific datasource
CALC;
     result=HP3245A_DEC;
     expr=HP3245ADatasource*1|F6;
BIGTEXT;
     name=HP3245A_DEC;
     caption=;
     x=510;
     y=390;
     f=32;
     w=400;
     h=50;
     border=on;
     units=on;
```

This example:

- Uses HP3245ADatasource as the input
- Performs a numeric calculation (×1) to force a CALC result
- Applies a fixed numeric format (F6) to the output text, F=fixed, 6=number of digits.
  Other options:
  G=general.
  N=much like F fixed point but uses thousands as separators.
- Outputs raw formatted text (e.g. 1.000000) while preserving numeric correctness

IMPORTANT NOTES:

- CALC expressions are purely numeric — no strings or units
- Units are applied (or suppressed) only by the display control
- CALC results can be plotted, logged, or statistically analysed like any live measurement
- Circular dependencies are not supported

DETERMINE Parameter (Control Initialisation)

The DETERMINE parameter allows WinGPIB to query an instrument when a User configuration is loaded and automatically initialise a control to match the instrument's current state.

This ensures that buttons, radios, sliders, spinners, and other controls correctly reflect the actual instrument settings before the user interacts with them.

DETERMINE is optional. If it is not specified, the control will start in its default state.

COMPATIBILITY:
DETERMINE works with RADIO, DROPDOWN, SLIDER, SPINNER, TOGGLE, TOGGLEDUAL & MULTIBUTTON.

PURPOSE:
DETERMINE is used to:

• Read back the current instrument mode or setting
• Compare the response with an expected value
• Automatically select or position a control
• Prevent accidental re-sending of commands during startup

It is evaluated once when the configuration is loaded.

GENERAL SYNTAX:
determine=<query>|<expected>[|<format>]

Where:

<query> The SCPI (or instrument) query command used to read the current state. Example:
CONF?, :VOLT:DC:RANG?, :SENS:VOLT:DC:NPLC? <expected> The value or text to compare against the instrument response. Used mainly for RADIO and CHECK-type controls.

<format> (optional)
Specifies how the response should be interpreted.

SUPPORTED FORMATS:
respnum
The response is treated as numeric (default).
Scientific notation and decimals are supported.

resptext
The response is treated as raw text.
Used when the instrument returns mode strings or quoted text.

If <format> is omitted, respnum is assumed.

EXAMPLES:
Radio button initialisation (text response):
```
RADIO;
    group=DMMFunction;
    caption=DCV;
    device=dev1;
    command=:CONF:VOLT:DC;
    determine=CONF?|VOLT:DC|resptext;
```

This queries CONF?.
If the response contains "VOLT:DC", the DCV radio button is selected.

Numeric range selection:
```
RADIO;
    group=DCRange;
    caption=10 V;
    device=dev1;
    command=:SENS:VOLT:DC:RANG 10;
    determine=:VOLT:DC:RANG?|10;
```

If the instrument reports a range of 10, this radio button is selected.

Slider initialisation:

```
SLIDER;
    name=SliderNPLC;
    device=dev1;
    command=:SENS:VOLT:DC:NPLC ;
    min=1;
    max=100;
    determine=:SENS:VOLT:DC:NPLC?|respnum;
```

The slider is positioned to the current NPLC value reported by the instrument.

Spinner initialisation:

```
SPINNER;
    name=SpinNPLC;
    device=dev1;
    command=:SENS:VOLT:DC:NPLC ;
    min=1;
    max=100;
    determine=:SENS:VOLT:DC:NPLC?|respnum;
```

The spinner value is set to match the current instrument NPLC setting.

OPERATIONAL NOTES:

• DETERMINE does not send any write commands to the instrument
• Controls are updated without triggering their action handlers
• DETERMINE queries are performed using the same GPIB engine as the control
• Raw instrument responses are supported when resptext is used
• Numeric responses may be in decimal or scientific notation

Performance Considerations

Some instruments (for example DMMs with high NPLC settings) may respond slowly to certain queries. DETERMINE is executed sequentially during configuration load, so excessive use of DETERMINE on slow queries may increase startup time.

It is recommended to:

• Use DETERMINE only where initial synchronisation is required
• Avoid redundant queries
• Prefer numeric DETERMINE where possible

SUMMARY:

DETERMINE provides automatic synchronisation between the User interface and the instrument state.
It is optional, non-destructive, and evaluated once at configuration load time.
Used correctly, it prevents mismatches between displayed controls and actual instrument settings.

KEYPAD – POPUP (NUMERICAL)

The KEYPAD control creates an on-screen numeric keypad that can type into USER-tab TEXTBOX controls.
Ideal if you want to use WinGPIB on a touchscreen device.
How targeting works:

- The keypad always types into the "current target" TEXTBOX.
- The user sets the target by clicking a TEXTBOX first.
- Only TEXTBOX is supported as a keypad target (for now).

KEYPAD can be FIXED (always visible on the User tab) or POPUP (shown when a TEXTBOX is selected).

CONFIG FORMAT (multi-line or single-line):
KEYPAD;
name=Keypad1;
caption=Keypad;
mode=popup; ; fixed or popup
x=750;y=10;
w=230;h=280;
targetlabel=on

FIELD MEANINGS:
KEYPAD = creates the keypad
name = internal name used by the system (optional; default Keypad1)
caption = title text for the keypad (optional; default "Keypad")
mode = fixed or popup (optional; default fixed)
x,y = position (fixed panel position, or initial popup position)
w,h = keypad size in pixels
targetlabel = on/off (shows the current target TEXTBOX name at the top)

BEHAVIOUR (POPUP MODE):

- Clicking a TEXTBOX sets the keypad target and shows the popup keypad.
- The popup remembers its last moved position within the WinGPIB session.
- If the user closes the popup, WinGPIB treats this as "hide" (the keypad will reappear next time a TEXTBOX is clicked).

KEYS:
0 – 9 = inserts digits at the caret position
. = inserts decimal point
BK = backspace at the caret (or deletes selected text)
CL = clears the entire target TEXTBOX
+/- = toggles the sign of the current entry
OK = no action (reserved for future use)
X = hides the popup keypad (popup mode only)

EXAMPLES:
Fixed keypad on the User tab:

```
KEYPAD;
    name=KeypadFixed;
    caption=Keypad;
    mode=fixed;
    x=700;
    y=20;
    w=230;
    h=280;
    targetlabel=on;
```

Popup keypad (appears when a TEXTBOX is clicked):

```
KEYPAD;
    name=KeypadPopup;
    caption=Keypad;
    mode=popup;
    x=750;
    y=10;
    w=230;
    h=280;
    targetlabel=on;
```

DATASAVE Parameter (TEXTBOX Persistance Control)

The DATASAVE directive controls whether USER-tab TEXTBOX values are automatically saved and restored when a configuration file is loaded.

This is a global configuration directive and applies to the entire config file.

CONFIG FORMAT (single-line directive):

DATASAVE=enabled
or
DATASAVE=disabled

- No semicolons
- No additional parameters
- Case-insensitive
- If omitted, DATASAVE defaults to enabled

FIELD MEANINGS:
DATASAVE = enables or disables automatic saving and restoring of TEXTBOX values

enabled
TEXTBOX contents are saved when the User tab is reset or refreshed, and restored when the same config file is loaded again.

disabled
TEXTBOX contents are not saved and not restored.
TEXTBOX controls always start with their defined default values.

BEHAVIOUR:
When DATASAVE is enabled:

- User-entered TEXTBOX values are stored automatically.
- Values are restored when the same config file is reloaded.
- RESET and REFRESH preserve user-entered values.

When DATASAVE is disabled:

- No TEXTBOX values are saved.
- Previously stored values are ignored.
- RESET and REFRESH do not persist any TEXTBOX data.
- Existing saved data is not deleted.

All other controls and behaviour are unaffected.

PLACEMENT RULES:
- DATASAVE may appear anywhere in the config file (once only).
- It is typically placed near the top for clarity.
- Each config file is evaluated independently.

EXAMPLES:
Disable textbox persistence (defaults always apply):

```
DATASAVE=disabled;
TEXTBOX;
    name=RefPPM;
    caption=PPM Ref;
    x=20;
    y=370;
    w=80;
    h=22;
    default=2.000000;
```

Enable textbox persistence (default behaviour):

```
DATASAVE=enabled;
TEXTBOX;
    name=UserValue;
    caption=Set Value;
    x=20;
    y=40;
    w=100;
    h=22;
    default=0;
```

SUMMARY:

- DATASAVE is a global config directive
- Controls persistence of TEXTBOX values only
- Defaults to enabled for backward compatibility
- No impact on other controls or logic

TAB Parameter (Multiple sub-tabs)

MULTIPLE TABS:
The User Tab supports multiple configurable sub-tabs, allowing complex layouts to be spread over several pages instead of crowding everything onto one screen.

Controls are assigned to tabs in the User configuration text file using the TAB; block.

Everything above the first TAB block automatically appears on a default tab named Main.

Locate your tabs per example below at the foot of your config file. Kepp the default MAIN tab config entries above them.

Each TAB; block defines (or re-selects) a tab, and all following controls belong to that tab until another TAB; block is encountered.

EXAMPLE:

In this example a sub-tab is created named “Mode”.

```
TAB;
    name=Mode;
MULTIBUTTON;
    name=MeasFunc;
    caption=;
    device=dev2;
    items=ACV,DCV,R2W,R4W;
    commands=:CONF:VOLT:DC,:CONF:CURR:DC,:CONF:VOLT:AC,:CONF:CURR:AC,:CONF:RES,:CONF:FRES;
    x=450;y=500;
    w=50;
    h=28;
    gap=3;
    determine=CONF?||resptext;
    detmap=VOLT:DC,CURR:DC,VOLT:AC,CURR:AC,RES,FRES;
```

SCRIPTING (LUA)

Lua scripts allow custom logic, sequencing, looping, conditional control, and data processing beyond fixed UI elements. Scripts can send commands to instruments, read responses, log data, and interact with the application runtime.

See https://www.lua.org for more info including the LUA command set.

```
LUASCRIPTBEGIN;name=LoopBigText
logto("LuaLog", "Ready to go......")

for i = 1, 5 do
    local r = query("dev1", ":READ?")
    settext("BigTextLua", r)
    logto("LuaLog", "Value = " .. r)
    doui()
    sleep(1000)
end

fire("ShowFiles2")   -- fire the "WinGPIBdata" button on the Devices tab

logto("LuaLog", "Done")
LUASCRIPTEND
```

BASIC SYNTAX:

Lua is a lightweight, line-oriented scripting language.

Statements execute top to bottom
Commands are case-sensitive
Comments begin with --
Device 1 name used in scripts = dev1
Device 2 name used in scripts = dev2

HOW TO LOAD SCRIPTS:

LUA scripts can be run from a textarea on the USER tab or from the Config file.
Load R6581T-Demo3.txt or R6581T-Demo4.txt into WinGPIB to demo both methods.

AVAILABLE FUNCTIONS:

The following functions are provided by the host application and are available to all Lua scripts:

Add comments to your scripts.
<u>Example:</u>
-- This is a comment line

send(DeviceName, Command)
Sends a GPIB/SCPI command to the specified device (no reply expected).
<u>Example:</u>
send("dev1", ":CONF:VOLT:AC") *(Advantest R6581(T) demo command)*

query(DeviceName, Command)
Sends a command and returns the response as a string.
<u>Example:</u>
local r = query("dev1", "READ?") *(Advantest R6581(T) demo command)*

log(Text)
Writes text to the application log window.
<u>Example:</u>
logto("LuaLog", "R6581T to 10V range")

fire("ButtonName")
Press a button elsewhere on other tabs in WinGPIB
<u>Example:</u>
fire("ShowFiles2") -- Press the "WinGPIBdata" button on the Devices tab
Note: see WinGPIB_buttons_by_tab.csv for list of available WinGPIB buttons across the app.

VARIABLES:
Variables are created using local (recommended) or global assignment.

Example:
```
local count = 0
local value = 0.0
```

Lua is dynamically typed; variables may contain strings, numbers, or nil.

LOOPS:
for loop
Used for a fixed number of iterations.
<u>Example:</u>
```
for i = 1, 5 do
local r = query("dev1", "READ?")
log(i .. ": " .. r)
end
```

while loop
Used for continuous or long-running operations.
<u>Example:</u>
```
local i = 0
while i < 100 do
i = i + 1
log("Count = " .. i)
end
```

DELAYS:
The sleep(ms) function pauses execution of a Lua script for a specified number of milliseconds.
Apply in any kind of repeated check, a loop is the simplest.
What the loop achieves:
- sleeps in small steps (50ms)
- checks _luaCancelRequested each step (ThrowIfLuaCancelled())
- optionally calls Application.DoEvents() so the UI can process clicks (including Stop)
The script can still be stopped using the **STOPLUA** button during sleep.
<u>Example:</u>
```
send("OUTP ON")
sleep(500)
send("MEAS:VOLT?")
```
or,
```
while true do
   local v = query("READ?")
   print(v)
   sleep(1000)  -- 1 second interval
end
```

STRING FORMATTING:
Formatted numeric output can be generated using string.format.
<u>Example:</u>
```
local v = tonumber(query("dev1", "READ?"))
log(string.format("Value = %.9f", v))
```

COMMENTS:
-- This is a comment

Multi-line comment:
```
--[[
This is a
multi-line comment
]]
```

BEHAVIOUR:

- Scripts execute sequentially when run
- Instrument configuration is persistent unless changed
- Errors in Lua logic do not affect device configuration
- Lua scripts do not perform device discovery or addressing
- All device names (dev1, dev2, etc.) must already be defined by the system

POLYNOMIAL AND METROLOGY FUNCTIONS :
WinGPIB provides a built-in set of polynomial and metrology helper functions which are available directly from LUA scripts. These functions are intended for test, calibration, linearity analysis, and numerical processing of instrument data.

Polynomial Evaluation

The polyeval() function evaluates a polynomial at a given x value.

Form:
polyeval(x, a0, a1, a2, ...)

This evaluates:
$y = a0 + a1 \cdot x + a2 \cdot x^2 + \ldots$

Example:
polyeval(4, 1, 2, 3) evaluates $y = 1 + 2x + 3x^2$ at $x = 4$ and returns 57.

Polynomial Root Solvers

WinGPIB includes closed-form solvers for polynomials up to 4th order.

solve2(a, b, c)
Solves: $a \cdot x^2 + b \cdot x + c = 0$

solve3(a, b, c, d)
Solves: $a \cdot x^3 + b \cdot x^2 + c \cdot x + d = 0$

solve4(a, b, c, d, e)
Solves: $a \cdot x^4 + b \cdot x^3 + c \cdot x^2 + d \cdot x + e = 0$

These functions return arrays of complex roots. Complex values with very small imaginary components may represent real roots.

Real Root Extraction

The realroots() and realrootssorted() functions filter real roots from complex root arrays.

Form:
realrootssorted(roots, tolerance)

This removes roots whose imaginary magnitude exceeds the tolerance and returns real roots sorted in ascending order.

Two-Point Linear Calibration

TwoPointCal performs a classic two-point linear calibration.

Form:
twopointcal(meas1, true1, meas2, true2)

Returns:
{ gain, offset }

Calibration equation:
true = gain·measured + offset

Apply Calibration

applycal applies a previously calculated gain and offset.

Form:
applycal(measured, gain, offset)

This is typically used to correct instrument readings using calibration constants.

PPM Error Calculation

ppmerror calculates parts-per-million error relative to a nominal value.

Form:
ppmerror(measured, nominal)

Formula:
$ppm = (measured - nominal) / nominal \times 10^6$

This is commonly used in DMM and calibrator verification.

Least-Squares Line Fit

linefit performs a least-squares linear regression.

Form:
linefit(xValues, yValues)

Returns:
{ slope, intercept, rSquared }

This is useful for linearity testing, drift analysis, and gain/offset estimation.

Polynomial Curve Fit

polyfit performs a least-squares polynomial fit of arbitrary degree.

Form:
polyfit(xValues, yValues, degree)

Returns:
{ a0, a1, a2, ... }

Resulting polynomial:
$y = a0 + a1 \cdot x + a2 \cdot x^2 + \ldots$

Polynomial Derivative

polyder returns the derivative polynomial coefficients.

Form:
polyder(coeffs)

If:
$y = a0 + a1 \cdot x + a2 \cdot x^2$

Then:
$dy/dx = a1 + 2 \cdot a2 \cdot x$

Polynomial Integral

polyint returns the integral polynomial coefficients.

Form:
polyint(coeffs, constant)

If:
$y = a0 + a1 \cdot x + a2 \cdot x^2$

Then:
$\int y\, dx = C + a0 \cdot x + a1 \cdot x^2/2 + a2 \cdot x^3/3$

Typical Applications

These functions are intended for:
- Instrument calibration
- Gain and offset correction
- Linearity and non-linearity analysis
- Polynomial calibration curves
- PPM error reporting
- Numerical post-processing of measurement data

All calculations use double-precision floating point arithmetic.

EXAMPLES:

Simple measurement loop:

```
log("5 readings...")
for i = 1, 5 do
local r = query("dev1", ":READ?")
logto("LuaLog",i .. ": " .. r)
end
```

Configure once, then read:

```
send("dev1", "DCV 10")
send("dev1", "NPLC 10")

local v = query("dev1", "READ?")
logto("LuaLog","Measured value = " .. v)
```

EXAMPLES (IanJ's):

Setup R6581T DMM and do 5 reads (from User tab text area):

```
logto("LuaLog", "Writing to BigTextLua")
for i = 1, 5 do
   local r = query("dev1", ":READ?")
   settext("BigTextLua", r)
   logto("LuaLog", "Lua value = " .. r)
   doui()
   sleep(1000)
end
logto("LuaLog", "Done")
```

Requires this in Config file:

```
; GPIB engine method dev1 & dev2, standalone or native
GpibEngineDev1=native
GpibEngineDev2=native

; Name
GROUPBOX;caption=Advantest R6581T Controls - LUA demo

; LUA SCRIPTING TEST
TEXTAREA;
     name=LuaTest;
     caption=Lua Script:;
     x=20;
     y=270;
     w=600;
     h=200;
     init=logto("LuaLog","Lua running")|local r=query("dev1",":READ?")|logto("LuaLog",r);

BUTTON;
     caption=Run Lua from test box;
     action=RUNLUA;
     device=;
     command=LuaTest;
     value=;
     result=;
     x=20;
     y=490;
     w=180;
     h=30;
BUTTON;
     caption=Stop Lua;
     action=STOPLUA;
     device=;
     command=;
     value=;
     result=;
     x=220;
     y=490;
     w=180;
     h=30;
```

```
; --- LOG BOXES ---
TEXTAREA;
      name=LuaLog;
      caption=Lua Log:;
      x=650;
      y=270;
      w=350;
      h=100;
      readonly=true;

; Bigtext that LUA can write to directly in the script.
BIGTEXT;
      name=BigTextLua;
      caption=Lua Output:;
      x=20;
      y=35;
      f=32;
      w=600;
      h=50;
      border=on;
      units=off;
      target=TextBoxLua;
```

Read from R6581T DMM to WinGPIB display - 5 reads (Config file scripting):

```
LUASCRIPTBEGIN;name=LoopBigText
logto("LuaLog", "Writing to BigTextLua")
for i = 1, 5 do
   local r = query("dev1", ":READ?")
   settext("BigTextLua", r)
   logto("LuaLog", "Lua value = " .. r)
   doui()
   sleep(1000)
end
fire("ShowFiles2")   -- fire the "WinGPIBdata" button on the Devices tab
logto("LuaLog", "Done")
LUASCRIPTEND

BUTTON;
      caption=Run Lua from config file;
      action=RUNLUA;
      command=LoopBigText;
      x=20;
      y=170;
      w=220;
      h=30;
TEXTAREA;
      name=TriggerLog;
      caption=Lua Log:;
      x=20;
      y=230;
      w=600;
      h=140;
      readonly=true;
BIGTEXT;
      name=BigTextLua;
      caption=Lua Output:;
      x=20;
      y=35;
      f=32;
      w=600;
      h=50;
      border=on;
      units=off;
      target=TextBoxLua;
```

Write to an HP3245A & Read from R6581T DMM to WinGPIB display - (Config file scripting):

```
; GPIB engine method dev1 & dev2, standalone or native
GpibEngineDev1=native;
GpibEngineDev2=native;

; Name
GROUPBOX;
      caption=Advantest R6581T Controls - LUA demo;

LUASCRIPTBEGIN;name=LoopBigText
logto("LuaLog", "Writing to BigTextLua")


-- HP3245A on dev2
logto("LuaLog", "3245A 5.000000VDC")
send("dev2", "DCRES HIGH")
send("dev2", "APPLY DCV 5")

sleep(1000)   -- Allow time for 3245A to settle

-- Setup R6581T
logto("LuaLog", "R6581T to 10V range")
send("dev1", "SENS:VOLT:DC:RANG 10")

-- R6581T on dev1 (5 readings @ 1s)
logto("LuaLog", "Read R6581T")
for i = 1, 5 do
    if shouldstop and shouldstop() then logto("LuaLog","[LUA] Stopped") return end
    local r = query("dev1", ":READ?")
    settext("BigTextLua", r)
    logto("LuaLog", "Lua value = " .. r)
    doui()
    sleep(1000)
end

-- HP3245A on dev2
logto("LuaLog", "3245A 1.000000VDC")
send("dev2", "APPLY DCV 1")

sleep(1000)   -- Allow time for 3245A to settle

-- Setup R6581T
logto("LuaLog", "R6581T to 1V range")
send("dev1", "SENS:VOLT:DC:RANG 1")

-- R6581T on dev1 (5 readings @ 1s)

logto("LuaLog", "Read R6581T")
for i = 1, 5 do
    if shouldstop and shouldstop() then logto("LuaLog","[LUA] Stopped") return end
    local r = query("dev1", ":READ?")
    settext("BigTextLua", r)
    logto("LuaLog", "Lua value = " .. r)
    doui()
    sleep(1000)
end

logto("LuaLog", "Done")
LUASCRIPTEND

BUTTON;
      caption=Run Lua from config file;
      action=RUNLUA;command=LoopBigText;
      x=20;
      y=100;
      w=180;
      h=30;
BUTTON;
      caption=Stop Lua;
      action=STOPLUA;
      device=;
      command=;
      value=;
      result=;
      x=220;
      y=100;
      w=180;
      h=30;
TEXTAREA;
      name=LuaLog;
      caption=Lua Log:;
```

```
    x=20;
    y=140;
    w=300;
    h=360;
    readonly=true;
; Bigtext that LUA can write to directly in the script.
BIGTEXT;
    name=BigTextLua;
    caption=Lua Output:;
    x=20;
    y=35;
    f=32;
    w=600;
    h=50;
    border=on;
    units=off;
    target=TextBoxLua;
```

SUMMARY:

Lua scripts provide flexible, programmable control over instruments, enabling complex test sequences, data logging, and conditional logic beyond fixed UI controls.