



WALKER SYSTEMS

building
intelligence

TECHNICAL DESCRIPTION

Group Points AND Animation

(PROGRAMMING GROUP POINTS FOR
MASS ALARM PROCESSING AND
ANIMATED DISPLAYS)

FORM # WSC99-001C

REVISION 1

CONTENTS

Summary	1
Background	2
Functionality Changes	3
Building VA and VB Variables Version 40.12 and earlier	3
Building VA and VB Variables Version 41.02 and later	3
Building the Group Point	4
Using the MS15 Message Point	7
Building the Alarm Point	7
Procedure	9
Building a Basic Group Point Application	10
Optional Step: The MS15 Point	13
Examples	14
Example 1: GP with Digital Variables and Basic Alarm Acknowledgement	14
Example 2: GP with Analog Variables and Advanced Alarm Acknowledgement	16
Example 3: Using the MS15 technique	18
Group Point Animation	20
Block Animation Functionality	20
Tank Level Animation	20
Vertical Thermometer Animation	23
Zone Temperature Display	25
Trouble Shooting and Error Messages	28
Appendix A: Display Colours	29
Appendix B: Valid ASCII Characters	30

SUMMARY

The Group Point (GP) became part of the Walker controls system in WS16XX firmware version 40.08. Group Point applications can be programmed in WS1600 and WS1616 panels only.

Group Points provide the ability to efficiently control large groups of similar points with very little system overhead. This increases the alarm processing capabilities of the Walker system significantly and allows for numerous new control techniques.

Group Points provide new display functionality that includes blinking text, blinking text backgrounds and colour programmability that allows colours to change on the screen automatically to reflect conditions changing in the system. Group Point display technology can be used with GCL techniques to produce animations in the Connect-500 GRAPHICS window.

A discussion of the new system functionality is provided, as are programming procedures and examples. Animation programs are also provided that can be copied and used in Walker systems.

BACKGROUND

Group Points are used primarily to control a group or set of identical devices or points and trigger an alarm when one in the group switches or exceeds a limit. This allows the system to effectively process an extremely large number of alarms using only one Alarm Point. There are many Energy Management operations that include the control of numerous identical devices, and the Group Point provides an efficient set of tools for handling these situations with low system overhead.

New classes of analog and digital variables are assigned values from processes in the system, and these variables are grouped together for display in custom colour arrangements on a Connect-500 graphic. The alarm priority, colour scheme, text information and the limits for the group of variables are programmed into the system through a set of instructions or 'Rules' inside the Group Point. These Rules are simple and highly flexible, leaving the options for application wide open.

Group Points are similar to the original Display Points in that they use link files to group points together for graphic display in Connect-500. The display capabilities are significantly different, however, as Group Point displays can include flashing text, flashing background displays and up to 16 colours that can automatically change to reflect changing conditions in the system. This technology allows different conditions in the system to report on the graphic display with instant visual recognition for the operator. Countless new system administration techniques are possible, such as maintenance alarms for groups of equipment and colour coded displays for supplemental building systems. Group Points are highly effective when used to handle incoming data from external systems interfaced to Walker through the Walker ASCII Protocol.

Different levels of alarm acknowledgement functionality can be implemented in Group Point applications. Implementing basic functionality causes the Group Point present itself for acknowledgement when one of the variables contained inside it causes an alarm, and advanced functionality causes the individual variable that caused the alarm to present itself.

Group Points can also be used in concert with some GCL techniques to create animated graphic displays. Animated graphics can consist of a simple animated display with blinking text and background, or more sophisticated animations simulating tank levels rising, thermometers with moving fluids or colour coded floor plans with areas that change colour automatically to indicate actual temperatures.

This document provides procedures and examples for programming Group Point applications with WS16XX firmware versions 40.08 and up. Also included are sample animations that can be modified or used directly in the Walker applications.

FUNCTIONALITY CHANGES

WS1600/WS1616 firmware version 41.02 brought system-wide changes to the menu penetration paths. This was primarily due to the COV trending functionality implemented in this release, and the addition of the Add and Copy commands to the standard Build menu throughout the product.

Group Point functionality became available in version 40.08, and all of the menu selections required to program Group Point applications are different between v40.08 to 40.12 and v41.02. Procedural differences are noted throughout this section.

Note In many cases, the only difference in the menu penetration is the need to enter Program, Database ('PD) in v41.00 and up instead of just Program ('P) to access the features under the program menu. When this is the only difference, it is noted inside the section where it applies.

Building VC and VB Variables — VERSION 40.12 AND EARLIER

Building variables with v40.08 to v40.12 requires the use of Buffer, Database, Add ('BDA) from the main system menu. Upload the database ('BDU) and use Buffer, Database, Add ('BDA) to add the number of VB or VC points that you will need.

Once the points are added they exist in the database as null points. If required, they can be revised using Program, Variables, V-vb or V-vc, Revise ('PDVB or 'PDVC). In many cases the variables will not need to be revised because they will be assigned some value in a GCL program.

Building VC and VB Variables — VERSION 41.02 AND LATER

As of WS1616/WS1600 firmware version 41.02, the menu structure is Program, Database, Variable, ('PDV). When 'PDV is selected, the operator is prompted to select Variables-standard (V), V-vb (B), V-vc (C) or VF Float (F). The options for VB points ('PDVB) and VC points ('PDVC) are the selections you use to build VB and VC points. The option ('PDVF) for floating point variables is not used in Group Point applications.

If you access the SAC menus with 'PDVB or 'PDVC, the system prompts you with the standard line that is seen when building all Walker point types:

Build Add Copy Revise Erase Download Group Select

If you select the Build option the system will prompt you with "use Add prompt to create points".

ADD: If you select the Add option then the system will prompt you with:

```
Point additions --- PA106
Point to copy from or (CR)   ?
Number to add ---           ?
```

- "Point additions for PA0106": states the panel number the point will be added to.
- "Point to copy from or (CR)": if starting with no points, hit <enter>, otherwise type in a VB or VC point which has already been created.
- "Number to add": type in the number of VB or VC points you require, then hit <enter>.

COPY: If there are already existing points, the Copy command can be used to copy points or to change the current configuration of points. When selecting Copy, the system prompts as follows:

```
Point to copy from or (CR)   ? UA1 1
Point to start copy at      ? UA20
Number to copy ---         ? 5
```

- "Points to copy from or (CR)": type in the point to use as a source to copy from. If the enter key is hit with no point referenced, the system prompts with "Number to copy". The number entered here is the number of null points the system will build.
- "Point to start copy at": will start copying the point referenced above beginning at the point entered here. **WARNING:** If the point entered already exists, the system will build the new point over the existing point.
- "Number to copy —": enter the number of points you want built.

Building the Group Point

The SAC menu structure for building a Group Point is Program, Database, Displays, Group ('PDDG) for v41.00 and up, and Program, Displays, Group ('PDG) for v40.08 to 40.12. At this point, the system presents the standard Build menu. In newer versions, using Add will create null points and Copy will create points identical to a source point copied from. Use Build to create new points in all versions. When building a Group Point, the system prompts as follows:

- STATE: auto/manual control <default is auto>.
- ALARM: on/off. This option is only available when the point is in manual control state. As with other Walker point types, this element provides an override to test your control algorithms. When in auto state (the point is under GCL command), the alarm on/off element provides the flag that is used to trigger an associated AL point.

At this point you begin entering the 'rules' that the Group Point will follow.

Every Group Point requires at least one set of rules to determine text, colours, and points which are included in the group. The format for the set of rules for the Analog VC points in a Group Point and the set of rules for a Digital VB point are very similar, but have some differences. The following are the different formats for VC and VB points:

VB Point Rules

%RULES	:the beginning of the set of rules for a digital point
"VBm	:the start of VBs; always prefaced with a quotation mark
"VBn	:and the end of VBs; always prefaced with a quotation mark
_%_AS_AM_AP_CB_C	:determines characteristics of the points
%	:end of rule
VBm	:list all the VB points in the Group Point in numerical order; no quotation marks used
VBn	:the end of the list described above

VC Point Rules

%RULES	:the beginning of the set of rules for a analog point which includes;
"VCm	:the start of VCs; always prefaced with a quotation mark
"VCn	:the end of VCs; always prefaced with a quotation mark
&LIMIT_Hx,Ly,Dz	:sets up the limits for determining the state of alarm
_%_AS_AM_AP_CB_CF	:determines characteristics of the points
%	:end of rule
VCm	:list all the VC points in the Group Point in numerical order; no quotation marks used
VCn	:end of the list described above

The elements of the above rules are as follows:

%RULES	The marker to indicate that there is a set of instructions or rules to follow.
VBm or VCm	The beginning of the range of variables to be included in the Group Point (literal point name; VCm or Vbm), where m represents the beginning of the range.
VCn or VBn	The end of the range of variables to be included in the Group Point (literal point name; VCn, or Vbn), where n represents the end of the range.
&LIMIT	For analog points only. The alarm limits in the format &LIMIT_Hx,Ly,Dz (numeric values; H for high limit, L for low limit and D for dwell or hysteresis) . The values x, y and z have to be in a format with a decimal point. For example, if the high limit was 25, the low limit was 10 and the dwell was 1, then the string would be as follows: &LIMIT_H25.0,L10.0,D1.0.
%_AS_AM_AP_BC_FC	String to indicate different characteristics of a VB or VC point when it meets a certain condition. The fields in this string are described in the following paragraphs.
AS ALARM STATE	Conditions for the VC point are HIGH, LOW, OFF, or PAST. The Alarm State for the VB point can be any value that will be used in the system to indicate a condition. This value can be numeric (0 or 1), or an ASCII character preceded by the pound symbol (#), or the asterisk character (*) to indicate a default value to be used if no other conditions listed are met.
Note	The pound symbol # is used in to indicate that an ASCII character follows. The system actually reads the decimal equivalent of the ASCII character entered; for example, #A would be interpreted by the system as 65 (the decimal equivalent of the ASCII character A is 65). Refer to the ASCII conversion table appended to this document.
AM ALARM MESSAGE	To display text, using a backslash (\) as a space delimiter.
AP ALARM PRIORITY	(A0 to A7), A0 disables the alarm and A7 is the highest priority.
BC BACKGROUND COLOUR	Colour of the background (see Colour Options in Appendix A for available colours).
FC FOREGROUND COLOUR	Foreground colour or display (text) colour (see Colour Options in Appendix A for available colours).
%	Indicates end of rules
VCm or VBm	The last step in creating a Group Point is to list all VB or VC points which are going to be grouped into it. There can be up to 80 points listed. Start

the list with the first VB or VC and list them in numerical sequence.

VCn or VBn The last VB or VC included in the Group Point. This is the end of the list described above.

Note BC (background colour) and FC (foreground colour) are optional fields. If you do not assign a value to these fields, the display will blink between red text on white and white text on red. When the point is OFF the text is red on a white background.

Using the MS15 Message Point

To conserve database space, all of the rules that will be used in Group Point programming for a given SAC can be input into the panel's MS15 point. The system searches the MS15 point for Group Point rules based on pointers included inside the GP. This technique is particularly useful when memory consumption in a particular SAC must be limited. Many operators prefer to put all the rules for all of their GPs into the MS15 point, and then call lines of the MS15 point from within the various Group Points. Each line of the MS15 point is referred to within the Group Point using the format "STx", where x is the line number in the MS15 point. The Group Point is built as before, but the first rule string immediately following the range statements is replaced by the ST number corresponding to the line inside the MS15 point. ST may be considered as an abbreviation for STRING.

Building the MS15 The MS15 point is built by selecting Program, Database, Strings ('PDS) in v41.00 and up, and Program, Strings ('PS) in v40.08 to 40.12. To build MS points, the Build or Add commands can be used to create the 15 MS points you require. Only MS15 can be used to hold GP rules, so MS1 through MS14 will exist in the database as null points.

When using the ADD command, follow the same procedures as mentioned earlier when building VC and VB points. When using v40.12 or earlier, use Buffer, Database, Add ('BDA) to create the 15 points.

After the 15 MS points are constructed, revise MS15 and add the rule strings that are required.

Building the Alarm Point

There are two procedures to associate alarms with GP points. The first method uses the GP point and the second uses a high unused VB point as the 'alarm point' data element in the AL point. In both cases the alarm point being used has to be configured as a digital alarm point. When the variables inside the GP point go into an alarm, the GRAPHIC window displays the alarm message which is configured in the Group Point's rules. The HISTORY window then shows the printer message which is created in the AL point.

Group Point as the 'Alarm Point' data element

In order for the Group Point to trigger an alarm in the system, the Group Point has to be included inside an AL point as the 'alarm point' data element.

The AL point constructed has to be a digital AL point, as the Group Point will send either a 0 or 1 to indicate that it is either in alarm or not in alarm. When any of the variables inside the Group Point trigger an alarm condition (as defined in the rules of the GP), the Group Point's alarm ON/OFF data element immediately switches to ON. When this occurs, the 'alarm point' element in the AL point also goes ON (or 1) and the alarm is triggered. In order for this to work properly, the 'alarm reference' element in the AL point is set to 0 (0 is the same as OFF).

The alarm will then be displayed in the HISTORY window and on the Connect-500 header. The display in the HISTORY window will resemble the following:

Alarm {date} PA102-AL1 PA102-GP1 ON.

{alarm message}

"High Unused VB Point" as the 'Alarm Point' data element

In order for the variable points within the Group Point to report in the HISTORY window when switching in and out of alarm, it is necessary for the Group Point to contain reference to itself and to an associated alarm point. These two points, the AL and the GP are placed before the %RULES field inside the Group Point.

When building the associated AL point, enter VBx in the 'Alarm Point' data element where x is a large number which would never be used in the system. Use the same VB point for the 'Alarm Point' data element for all the Group Points. For example, use VB2000 to reference alarms from GP1, GP2 and GP3. Do not build up the high unused VB point in the database; it must remain unbuilt in order for this technique to work.

With the alarm point configured in this way, the alarms can be acknowledged individually, or globally acknowledged by clicking on the alarm point or the Group Point in the graphic display. The display in the HISTORY window will resemble the following:

Alarm {date} PA102-AL1 PA102-VB1 ON

{Alarm Message}

Note Only high unused VB points may be used in this way. You cannot use high unused VC points because a digital point is required in the AL point's 'Alarm Point' field. The point you choose, such as VB2000, must never be built up in the system.

PROCEDURE

There are essentially 5 steps to building a basic GP application:

- identify how many VB or VC variables are required, and build them up in the database
- identify the rules that will be set out for the variables to follow, build a Group Point and code these rules into it
- build the AL alarm point that triggers an alarm when the one of the points in the group changes state or exceeds a limit
- create a GCL routine that will set the values of the VB or VC variables, based on the actual physical process
- create or modify the appropriate graphic link file to display the data in Connect-500

The way you go about the above steps will change depending on the functionality you want your GP application to have. The following questions should be considered before programming begins:

- Are the points you are grouping digital or analog?

digital points require VB variables; analog points require VC variables

- Will there be a need to build several point groups that will follow the same rules, or will some common rules be shared by different GPs?

the MS15 Message Point can be used to conserve SAC memory by holding common rules that will be shared by several GPs

- When alarms occur, do you want the system to indicate simply that the Group Point is in alarm, or do you want to be able to acknowledge the individual variable point inside the group that caused the alarm?

the way GP and AL points are built will differ slightly depending on the level of functionality you require

Following is a general procedure for building GP applications. As the procedures you follow will change depending on your answers to the above questions, different examples are provided in the next section.

There are additional steps to take if you are using the GP to create animations. This is discussed in Section II: Group Point Animation.

Building a Basic Group Point Application

1. Select the WS1616 or WS1600 panel you want to add the GP to, and upload the database with Buffer, Database, Upload ('BDU).

Build the Variables

Examine the points that you want to group and control with the GP, decide whether you need digital VBs or analog VCs, and identify how many you require.

2. Using v41.00 and up, penetrate the WS16XX menu with Program, Database, Variable ('PDV), and choose (B) for VB points or (C) for VC points. Select Add (A) from the subsequent menu, and the "Point to copy from or (CR)" prompt is provided to allow you to copy variables from a point already in the database, if one exists. If no suitable variables exist in the system to copy from, hit <enter>, and specify the number of new points to add at the next prompt.

Note If variables already exist in the database that are suitable to copy from, selecting Copy ('PDVC) instead of Add ('PDVA) provides the same functionality as specifying a "Point to copy from" after the Add command.

If you are using v40.08 to 40.12, add the points to the database with Buffer, Database, Add, ('BDA).

Build the Group Point

3. If you are using v41.00 and up, select Program, Database, Displays, Group, Build ('PDDGB).

If you are using v40.08 to 40.12, select Program, Displays, Group, Build ('PDGB).

Answer the prompts provided for STATE and ALARM (refer to the "Functionality Changes" section for details if required).

At this point you enter the rules for the Group Point. Refer to the previous sections for syntax descriptions.

Note How you build the Group Point will differ depending on what level of alarm acknowledgment you want. If you want just the Group Point to report when an alarm occurs, no special input is required when programming. If the individual variables inside the Group Point are to report when in alarm, you need to add a reference to the name of the GP you are building, and the associated AL point you will build later, inside of the Group Point. These references are placed just before the RULES, as follows:

```

Point define --- PA102-GP1

STATE      0-manual  1-auto    (1)      ?
ALARM      0-off    1-on      (1)
STATUS    --condition of alarm
---        ? GP1      ←-----
---        ? AL2
---        ? %RULES
---        ? "VB1
---        ? "VB5
    
```

In the above screen capture, the point GP1 is being built, so both GP1 and AL2 are input before the RULES begin. AL2 is now the point that must be used as the associated alarm point for the GP.

Build Associated Alarm Point

4. The AL point that is triggered by the Group Point must now be built. The alarm point must be a digital AL point, regardless of the types of variables inside the Group Point. Further, the Alarm Reference element in the AL must be set to 0 in all cases.

What you input into the 'Alarm Point' data element will change depending on the alarm functionality you want. If you want just the Group Point to report when one of the variables goes into alarm, enter the Group Point as the Alarm Point element (ie GP1). If the individual variables inside the Group Point are to report when alarms occur, the Alarm Point element must be set to a high unused VB point, such as VB2000. As explained in "Functionality Changes", the same high unused VB or VC should be used as the Alarm Point for all of the Group Points you build in the panel.

If you chose to have individual variables inside the Group Point present themselves for acknowledgment when alarms occur, note that the number of the AL point you build (ie AL1) has to correspond to the number you entered inside the Group Point in Step 3.

When you have finished building your AL point, download the database with 'BDD'.

Create GCL for the Variables

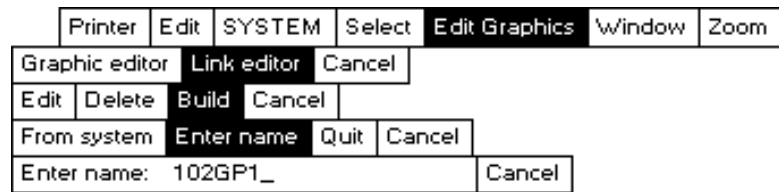
5. The next step involves creating GCL that will set the values of the variables in the Group Point. How this is done will vary greatly depending on the characteristics of your system.

What you need to do is somehow connect the variables with the physical devices they represent. Go through the examples presented in the next chapter if you are unclear about this step.

Create Link File

6. The final step involves creating the Connect-500 link file so that the data within the Group Point will be displayed in the GRAPHICS window. A new link file must be built up for each Group Point, and the variables within the Group Point must be positioned on the graphic as desired.

From within the GRAPHICS window, stop the current graphic display and select Edit Graphics, Link Editor and Build. As shown below, choose Enter name, and enter the point name of the GP.



Once the link file is built up, all of the VB or VC variables inside the GP are automatically added to the display. VB or VC variables can be repositioned as required using either the link editor or a text editor. The link editor is also used at this time to change the background graphic, add other points or otherwise modify the display.

Note If you want to line up VB or VC points in perfect vertical or horizontal lines, use the text editor. Load the link file and change the positional values for each point you want to line up. Each link element line contains an X and Y positional value in the fourth and fifth fields, respectively. For example, the following link element is positioned at X=589, Y=277:

```
:PICTURE [PICT] PROG01 589 277 9 0 |_[PICT] p b ;
```

Optional Step: the MS15 Point

If you have identified that common rules exist that will be used by more than one Group Point, the MS15 Message point is used to hold these rules and thus conserve database memory.

1. Upload the WS16XX database with Buffer, Database, Upload ('BDU) from the main menu.
2. If you are using v41.02 or later, build the MS15 point by selecting Program, Database, Strings ('PDS) from the main menu, then select Add (A).

You are prompted with the standard Add menu:

```
Point additions --- PA106
      Point to copy from or (CR)      ?
      Number to add ---                ?
```

Hit <enter> at the "Point to copy from or (CR)" prompt, and type in <15> at the "Number to add" prompt. You need to add 15 MS points because only MS15 can be used to hold GP rules. After this procedure is completed, MS1 through MS14 remain in the database as null points.

If you are using v40.08 to 40.12, you can build the points by selecting Program, Strings, Build ('PSB) from the main system menu, and building 15 consecutive points. Alternately, you can use Buffer, Database, Add ('BDA) and add 15 MS points.

3. Select Revise (R) from the menu that appears immediately following the previous step (if you have already left the MS point menu area, return to the Revise command by selecting Program, [Database], Strings, Revise ('P[D]SR) from the main menu).

Select MS15 from the list of MS points to revise. At this point, enter the Group Point rules into MS15 as required.

Note Keep track of which rules you input into which lines of the MS15 point. You may want to print out the MS15 point at this time and mark each line with its string number, starting at 1 for the first line, and so on.

4. You can now revise existing Group Points, or build new ones, using the string numbers inside the MS15 point in place of the rules statements they represent. For each rule that is inside MS15, input the corresponding string number into the rules of the GP. Use the format STx, where x is the line number inside the MS15.

For example, if the rule that you need to put inside of GP1 is &LIMIT_H7.2,L3.2,D1.0, and you have already input this statement as the first line inside of MS15, you would now enter ST1 inside GP1.

EXAMPLES

As GP applications can vary considerably, different examples are presented in an attempt to cover several of the possibilities. Application assistance is available from Walker Systems Corp.

Example 1: GP with Digital Variables and Basic Alarm Acknowledgement

Overview A Group Point is to be used to group 10 room occupancy sensors that report either 0 for "room empty" or 1 for "room occupied".

The job requires that an alarm reports at the front end if any of the rooms become occupied, but there is no need to acknowledge which of the rooms has caused the alarm. For this reason, only basic alarm acknowledgment is required. We also know that the alarms are very low priority; probably 1 on a scale of 0 to 7.

The GP will be built in panel 106 of the Walker controls system, and each of the sensors are connected directly to this panel's inputs, from IP1 through IP10. The job specification details that the front end should report "ROOMS UNOCCUPIED" when the sensors detect empty rooms, and "ROOM OCCUPIED", flashing in red letters, if any of the sensors detect movement.

As this application requires only a single GP, so there is no need to use the MS15 memory conservation technique.

- Procedure**
1. Upload the database for panel 106 by selecting the panel ('S), and then selecting Buffer, Database, Upload ('BDU) from the main menu.
 2. Build up the 10 variables required to represent the 10 room occupancy sensors. As the sensors are digital, you will use the digital VB variables.

If you are using v41.02 or later, select Program, Database, Variables, V-vb, Add ('PDVBA), and add 10 VB variables.

If you are using v40.08 to 40.12, select Buffer, Database, Add ('BDA) and add 10 VB points.

3. Revise the panel's PG1 program point to include the code required to set the conditions of the variables. As each sensor is a digital input, from IP1 to IP10, the GCL code must read the IP and set the corresponding VB to the value indicating on (room occupied) or off (room unoccupied).

You can optionally use ASCII characters to represent the on and off conditions, so let A = on and B = off (valid ASCII characters for use with GPs are presented in Appendix B). The decimal values of the ASCII characters A and B are 65 and 66 respectively, so the GCL

code would resemble the following:

```
IF IP1 = 1 THEN VB1 = 65 ELSE VB1 = 66
IF IP2 = 1 THEN VB2 = 65 ELSE VB2 = 66
...
IF IP10 = 1 THEN VB10 = 65 ELSE VB10 = 66
```

4. Next build the Group Point by selecting Program, Database, Displays, Group, Build ('PDDGB) from the main system menu. Note that 'Database' is removed from the menu penetration if you are using v40.08 to 40.12 (ie 'PDGB).

From the above steps we know that the condition A represents "on", and the display for this condition must read "ROOM OCCUPIED" in flashing red letters. Further, B represents "off", and the corresponding display is "ROOMS UNOCCUPIED", with no specified colour.

The rules for the GP can now be coded, and will look like this:

```
%RULES
"VB1
"VB10
%_#A_ROOMOCCUPIED_A1_MBWHITE_RED
%_#B_ROOMSUNOCCUPIED_A0_BWHITE_RED
%
VB1
VB2
VB3
VB4
VB5
VB6
VB7
VB8
VB9
VB10
```

5. The next step is to build the Alarm Point that will send the alarm if the GP's rules are broken. Penetrate the main system menu with Program, Database, Utility, Alarm, Build ('PDUAB). Note that 'Database' is removed from the menu penetration if you are using v40.08 to 40.12 (ie 'PUAB).

Enter the data required for the AL point, noting the following:

- the AL's 'Point Type' must be digital
- the 'Alarm Point' must be the GP name (in this case GP1)
- the 'Alarm Reference' must be 0

Note The strings entered into the Alarm Point's 'Alarm Message' field will display in the HISTORY window when the alarm occurs. The display in the GRAPHICS window is dictated by the Group Point rules in step 4.

6. Remember to download the database to the panel after all of the point additions and revisions are completed, using Buffer, Database, Download ('BDD).

7. Create the link in Connect-500 that will allow the GP to display in the GRAPHICS window. This is accomplished by using the Connect-500 link editor, choosing Build from, choosing Enter name and entering 106GP1.
8. Use the Connect-500 link editor or a text editor to position the VB or VC variables as desired on the display. Use the link editor to add a graphic background (choose File from the link editor menu, then Graphic File) or any other element you want to add to the display.

Example 2: GP with Analog Variables and Advanced Alarm Acknowledgement

Overview A Group Point is to be used to group 5 fuel tank level sensors. These sensors detect the fluid level in five different fuel tanks and report the levels to the system in a decimal number format.

Alarms must occur if the level in any of the tanks goes below 4 units or above 8 units. As the sensors are in different tanks, it is important to know which of the tanks caused the alarm for the group. For this reason we require more advanced alarm acknowledgment functionality than in the previous example. The alarms are to carry a high priority (5 on a scale of 0 to 7), and the tank sensors should be polled for input every 60 seconds.

As with Example 1, this is a relatively small project and only one Group Point is required. The MS15 memory conservation technique with therefore not be used.

- Procedure**
1. Select the WS1600 or WS1616 panel that will be used and upload its database with Buffer, Database, Upload ('BDU).
 2. Build up the VC variables that will carry the values of the tank sensors.

If you are using v41.02 or later, penetrate the main menu with Program, Database, Variables, V-vc, Add ('PDVCA). Add a quantity of five VC variables to the system.

If you are using v40.08 to 40.12, select Buffer, Database, Add ('BDA) from the main menu and add five VC points.

3. Assign the VC variables to the sensors with some simple GCL coding in the panel's PG1 program point. In the case of this example, the sensors are analog inputs attached to panel 102 as IP1 through IP5. The GCL coding required will resemble the following:

```
DO-EVERY 60S
VC1 = 102-IP1
VC2 = 102-IP2
VC3 = 102-IP3
VC4 = 102-IP4
VC5 = 102-IP5
END-DO
```

4. Build the GP to group the VC variables together. For v41.02 or later firmware, penetrate the main menu with Program, Database, Displays, Groups, Build ('PDDGB). For v40.08 to 40.12, the penetration is Program, Displays, Group, Build ('PDGB).

We have been told that the High limit is 8 and the Low limit is 4; but in order to set the rules for the sensors, we must decide on the allowable Dwell (hysteresis or tolerance) for the limits statement. The Dwell is the value inside of the limits that the variable must return to before the alarm becomes a past alarm. For example, with the limits of 8 and 4, a Dwell of 1 would mean that a high alarm would not become a past alarm until the value was reduced to 7 or less, and a low alarm would not become a past alarm until a value of 5 or above was reached.

Using a Dwell of 0 can result in alarms occurring and becoming past alarms with unacceptable frequency. For this reason we will assign a Dwell of 0.5 for this application.

To complete the rules we must also determine what messages we want displayed for the alarm conditions of HIGH, LOW, OFF and PAST. For this example we will use the following:

HIGH condition: "HIGH TANK LEVEL"
LOW condition: "LOW TANK LEVEL"
PAST condition: "TANK LEVEL ALARM PAST"
OFF condition: "TANK LEVELS NORMAL"

Finally, we must recall that in order to achieve the advanced alarm acknowledgment functionality we must include reference to both the associated AL point and the GP itself.

We now know everything that must be coded into the GP:

```
GP1
AL1
%RULES
"VC1
"VC5
&LIMIT_H8.0,L4.0,D0.5
%_HIGH_HIGH\tANK\LEVEL_A5
%_LOW_LOW\tANK\LEVEL_A5
%_PAST_TANK\LEVEL\ALARM\PAST_A1
%_OFF_TANK\LEVELS\NORMAL_A0
%
VC1
VC2
VC3
VC4
VC5
```

Note No colours for foreground (text) or backgrounds were specified in the GP. Recall that these fields are optional, and a default setting applies if no entries are made (refer to "Functionality Changes").

-
- Next build the AL point. The first important thing to note here is that the point AL1 was included inside the GP to facilitate the advanced alarm acknowledgment functionality. For this reason the alarm point associated with the GP, and the one you must now build, has to be AL1.

The next important consideration is the use of the "high unused VB point" as the Alarm Point element. The value entered into the Alarm Point field should be higher than you would conceivably use in all of your Group Point applications; for example, VB2000 (this VB point must be so high that it will never be built up in the system).

Penetrate the main system menu with Program, Database, Utility, Alarm, Build ('PDUAB). Note that 'Database' is removed from the menu penetration if you are using v40.08 to 40.12 (ie 'PUAB). Enter the data required for the AL point, noting the following:

- the AL's 'Point Type' must be digital
 - the 'Alarm Point' must be a high unused variable, like VB2000
 - the 'Alarm Reference' must be 0
- Remember to download the database to the panel after all of the point additions and revisions are completed, using Buffer, Database, Download ('BDD).
 - Build the 102GP1 link file in Connect-500 and position the VCs on the graphic display as desired.

Example 3: Using the MS15 Technique

Overview In this example we will use the MS15 point to hold the rules for all of the Group Points in a panel. For brevity we will assume that all of the rules for the GPs are already established, and all that is required is the creation of the MS15 point and the building of the GPs. The rules that will be used by several different GPs in the panel are as follows:

- &LIMIT_H10.0,L1.0,D0.5
- _%_#A_BUILDING\ON\FIRE_A7_MBWHITE_RED
- _%_#B_ALL\SYSTEMS\NORMAL_A0_BWHITE_RED
- _%_*_UNKNOWN\STATE_A1_CYAN_BLACK

- Procedure**
- First upload the database from the panel you will be working with using 'BDU.
 - If you are using v41.02 or later, build the MS15 point by selecting Program, Database, Strings, Add ('PDSA) from the main system menu. Add a quantity of 15 MS points to the system.

If you are using v40.08 to 40.12, use Program, Strings ('PS) and select Build (B) 15 times.
 - From the menu that appears immediately after Step 2, select Revise

(R) and select MS15 from the list of points available for revision. At this point you enter the rules into MS15:

```
Point revision --- PA102-MS15
--- ? &LIMIT_H10.0,L1.0,D0.5
--- ? %_#A_BUILDING\ON\FIRE_A7_MBWHITE_RED
--- ? %_#B_ALL\SYSTEMS\NORMAL_A0_BWHITE_RED
--- ? %_*_UNKNOWN\STATE_A1_CYAN_BLACK
```

These rules are now available for GPs in the panel.

4. Revise or build the GPs you need, entering the string number of the rule inside MS15 that you want to use. Recall that the format is STx, where x is the number of the line inside MS15. A GP designed to make use of the last three rules that we put into MS15 in step 3 would look like this:

```
Point define --- PA102-GP1

STATE      0-manual  1-auto      (1)          ? 1
ALARM      0-off     1-on        (1)
STATUS     --condition of alarm
---       ? %RULES
---       ? "VB1
---       ? "VB5
---       ? ST2
---       ? ST3
---       ? ST4
---       ? %
---       ? VB1
---       ? VB2
---       ? VB3
---       ? VB4
---       ? VB5
```

Any Group Point in the panel can use the rules inside MS15 by putting the ST value in the rules as shown.

GROUP POINT ANIMATION

GP applications can include a variety of animated displays. This section presents techniques you can use to create simple animations in the Connect-500 GRAPHICS window.

You can copy the applications directly and implement them into your systems, or modify what is presented here to better suit your designs. In any case, working through the following applications will provide you with the skills to create GP animations.

Block Animation Functionality

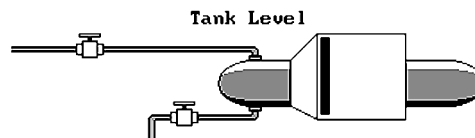
The space delimiter character, the backslash (\), is used extensively in GP animations. This character represents a block of pixels on the screen, 9 pixels wide and 10 pixels high. As you will see in the following applications, the backslash is used in the GP rules to create blocks of colour on the display, and placing VB or VC points next to each other creates the illusion of blocks of pixels moving, shrinking or stretching.

In the GP rules, where you would normally enter the text to display, a backslash character is used to fill one 9x10 pixel block with a particular colour. By adding more backslashes, the block of colour will appear larger.

When the variables inside the GP are positioned in the GRAPHIC link file, the area that would normally include some text message now displays blocks of colour. The variables are positioned on the graphic and coloured such that different values for the variables results in moving blocks of colour, thus creating the animated effect.

Tank Level Animation

This GP application provides the visual of a tank level automatically changing. In a working system the tank level would be an analog input, but for the purposes of demonstration a VA variable point is used to simulate the analog input rising and falling.



In the above screen capture, the tank, lines and valves are static graphics created with the graphics editor. The blocks that "move" and create the animation are VB points from within a GP that are positioned inside the square area in the middle of the tank on the graphic display.

Building the Animation

This animation is created using 12 VB points, 1 Group Point, a graphic image of a tank and about 20 lines of GCL code. The VBs are built up as usual (refer to Section I of this document), and the GP is built as follows:

```

Point define --- PA101-GP1

STATE      0-manual  1-auto      (1)          ? 1
ALARM      0-off    1-on        (1)
STATUS     --condition of alarm
---        ? %RULES
---        ? ''VB1
---        ? ''VB10
---        ? %_#A_\\\\\\\\\\\\\\\\_A0_BLUE
---        ? %_#B_\\\\\\\\\\\\\\\\_A0_WHITE
---        ? %
---        ? VB1
---        ? VB2
---        ? VB3
---        ? VB4
---        ? VB5
---        ? VB6
---        ? VB7
---        ? VB8
---        ? VB9
---        ? VB10
---        ? %RULES
---        ? ''VB24
---        ? ''VB25
---        ? %_#C_\\\\_A0_RED
---        ? %_#Q_\\\\_A0_GREEN
---        ? %
---        ? VB24
---        ? VB25

```

The graphic of the tank is built such that 10 rows (the 10 VBs) of 7 blocks (the 7 backslash characters) represent the contents of the tank. As the above rules show, this area inside of the tank will be 10 rows of blocks of either blue or white depending on the values of the VB variables.

The tank graphic also includes 2 valves that can change between red and green to signify open or closed. These valves each have one block that is set by VB24 and VB25.

The PG program points in the panel have to be edited to include the code that will set the VB values. As the above GP shows, each VB must be assigned either the value 'A' to indicate a row of blue or 'B' to indicate a row of white. In the actual system, the blue condition (A) might represent a sensor that sees liquid at a particular level in the tank, and the white condition (B) might mean no fluid at that sensor's level.

The GCL used to set the VBs is as follows. Note that the decimal equivalents of ASCII characters A, B, C and Q are used in the code (GCL code can use only the decimal equivalents, ASCII characters will either be mistaken for local variables or cause a syntax error). Note also that the code may have to change to reflect your system.

```

GCL Capture 95-Sep-28 14:42:13
**PA101

**PG1 - Point selected.   Label: PG1
**BEGIN
[ TANK LEVEL ANIMATION WHERE VA1 SIMULATES AN ANALOG INPUT ]
[ RAISING FROM 0 - 100 AND LOWERING FROM 100-0 ]
[ SETTING VB1 - VB10 TO 65 (ASCII CHARACTER A) FORCES THE VB'S ]
[ TO DISPLAY A RECTANGLE IN DARK BLUE 7 BLOCKS IN WIDTH ]
[ SETTING VB1 - 10 TO 66 (ASCII CHARACTER B) FORCES THE VB'S ]
[ TO DISPLAY A RECTANGLE IN LIGHT GREY 7 BLOCKS IN WIDTH ]
[ COLOURS AND WIDTHS OF RECTANGLES ARE DEFINED IN MS15 ]
[ THIS GIVES THE EFFECT OF A TANK FILLING WITH LIQUID DISPLAYED ]
[ IN THE GRAPHICS WINDOW USING A GP (GROUP POINT) ]
[ SETTING VB24 -VB25 TO 81 ( GREEN ) OR 67 ( STOP ) FOR VALVES ]

DO-EVERY 3S
IF VA2 = 0 THEN VA1 = VA1 * ( VA1 < 100 ) + 5 , \C
    VB24 = 81 , VB25 = 67 , VB26 = 82 , \C
    \ IF VA1 = 100 THEN VA2 = 1
IF VA2 = 1 THEN VA1 = VA1 * ( VA1 > 0 ) - 5 , \C
    VB24 = 67 , VB25 = 81 , VB26 = 83 , \C
    \ IF VA1 = 0 THEN VA2 = 0
END-DO

**PAGE2
IF VA1 <= 9.9 THEN VB10 = 66 ELSE VB10 = 65
IF VA1 <= 19.9 THEN VB9 = 66 ELSE VB9 = 65
IF VA1 <= 29.9 THEN VB8 = 66 ELSE VB8 = 65
IF VA1 <= 39.9 THEN VB7 = 66 ELSE VB7 = 65
IF VA1 <= 49.9 THEN VB6 = 66 ELSE VB6 = 65
IF VA1 <= 59.9 THEN VB5 = 66 ELSE VB5 = 65
IF VA1 <= 69.9 THEN VB4 = 66 ELSE VB4 = 65
IF VA1 <= 79.9 THEN VB3 = 66 ELSE VB3 = 65
IF VA1 <= 89.9 THEN VB2 = 66 ELSE VB2 = 65
IF VA1 <= 99.9 THEN VB1 = 66 ELSE VB1 = 65
**END

```

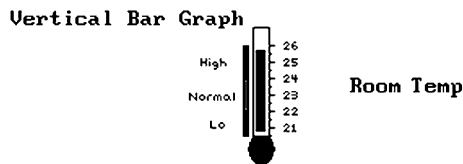
Putting It Together

After building variables and the GP, and generating the GCL code, the GP link file has to be created in Connect-500, and the VBs have to be positioned on the graphic display. Some experimentation will be required to get the VBs lined up and coloured to display the animation as desired. Use the text editor (as described at the bottom of page 10) to line up the VBs perfectly.

When you are in the link editor you will have turn off all of the attributes for the VB points, such as the Label, Units, Status, etc.

Vertical Thermometer Animation

This GP application simulates a liquid-filled thermometer rising and falling in 0.5° increments.



Building the Animation

This animation requires 11 VB points, 1 Group Point, a graphic image of a thermometer and about 10 lines of GCL code. The GP looks like this:

```

Point define --- PA101-GP2

STATE      0-manual  1-auto    (1)           ?
ALARM      0-off    1-on      (1)
STATUS     --condition of alarm
--- ?      %RULES
--- ?      **VB11
--- ?      **VB22
--- ?      %_#C_ \_A0_RED
--- ?      %_#D_ \_A0_BWHITE
--- ?      %
--- ?      **VB11
--- ?      **VB12
--- ?      **VB13
--- ?      **VB14
--- ?      **VB15
--- ?      **VB16
--- ?      **VB17
--- ?      **VB18
--- ?      **VB19
--- ?      **VB20
--- ?      **VB21
--- ?      **VB22

```

Note The above display shows quotation marks (") before each of the VBs listed at the end of the GP. These marks are NOT input when the point is built, but may display when the Revise (R) command is used.

The GCL required for this animation checks an input and sets the value of each VB to A or B depending on the actual temperature:

```

**BEGIN

[ // VERTICAL THERMOMETER DISPLAY ANIMATION // ]
[ .5 DEG C ACCURACY IN DISPLAY ]
IF IP1 <= 20.9 THEN VB11 = 68 ELSE VB11 = 67
IF IP1 <= 21.4 THEN VB12 = 68 ELSE VB12 = 67
IF IP1 <= 21.9 THEN VB13 = 68 ELSE VB13 = 67
IF IP1 <= 22.4 THEN VB14 = 68 ELSE VB14 = 67
IF IP1 <= 22.9 THEN VB15 = 68 ELSE VB15 = 67
IF IP1 <= 23.4 THEN VB16 = 68 ELSE VB16 = 67

```

```
IF IP1 <= 23.9 THEN VB17 = 68 ELSE VB17 = 67
IF IP1 <= 24.4 THEN VB18 = 68 ELSE VB18 = 67
IF IP1 <= 24.9 THEN VB19 = 68 ELSE VB19 = 67
IF IP1 <= 25.4 THEN VB20 = 68 ELSE VB20 = 67
IF IP1 <= 25.9 THEN VB21 = 68 ELSE VB21 = 67
IF IP1 <= 26.4 THEN VB22 = 68 ELSE VB22 = 67
**END
```

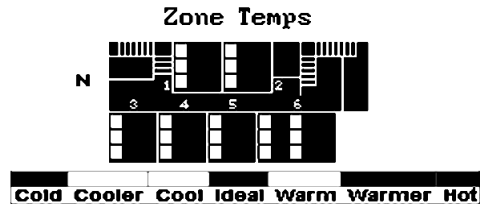
Putting It Together

The final procedure is building the GP link file and positioning the VBs on the display. As with any GP animation effect, the positioning and colouring of the VBs on the screen often requires some experimentation.

The intent with this animation is to place the VBs in a vertical column, with VB11 on the bottom, and each sequential VB directly on top of the last one. As only the "text" is to display (the block that will change colour), use the commands in the link editor to turn off the Label, Status, etc until only the block remains.

Zone Temperature Display

This animation takes a building floor plan and breaks it into the different controlled zones. The temperature in each zone is represented on the screen in an appropriate colour, and the colours automatically change to indicate temperature changes.



Building the Animation

This animation requires 6 VB points, 1 Group Point, the floor plate graphic and about 50 lines of GCL code. The GP looks like this:

```

Point define --- PA102-GP3

STATE      0-manual  1-auto      (1)      ?
ALARM      0-off    1-on        (1)
STATUS     --condition of alarm
---        ? %RULES
---        ? "VB28
---        ? "VB33
---        ? %_#U_ \\ \ A0_RED
---        ? %_#V_ \\ \ A0_BROWN
---        ? %_#W_ \\ \ A0_YELLOW
---        ? %_#X_ \\ \ A0_GREEN
---        ? %_#Y_ \\ \ A0_LGREEN
---        ? %_#Z_ \\ \ A0_LCYAN
---        ? %_#[_ \\ \ A0_BLUE
---        ? %
---        ? VB28
---        ? VB29
---        ? VB30
---        ? VB31
---        ? VB32
---        ? VB33

```

The GCL required for this animation checks the actual temperature for each zone, then compares this value to each zone's setpoint. Colours for each zone are then dictated by how far away from setpoint each zone's temperature is:

```

**BEGIN
[ // ZONE TEMPERATURE DISPLAY ANIMATION // ]
[ WHERE THE TEMPERATURE OF A ZONE IS REPRESENTED ]
[ BY AN ASSOCIATED COLOUR ]
[ VA4, VA5, VA6, VA7, VA8, VA9 ARE THE SETPOINTS ]
[ FOR THE DIFFERENT ZONES ]

```

```

IF ( IP2 >= VA4 - 0.9 ) OR \C
  ( IP2 <= VA4 + 0.9 ) THEN VB28 = 88
IF IP2 <= ( VA4 - 1.0 ) THEN VB28 = 89
IF IP2 <= ( VA4 - 2.0 ) THEN VB28 = 90
IF IP2 <= ( VA4 - 3.0 ) THEN VB28 = 91
IF IP2 >= ( VA4 + 1.0 ) THEN VB28 = 87
IF IP2 >= ( VA4 + 2.0 ) THEN VB28 = 86
IF IP2 >= ( VA4 + 3.0 ) THEN VB28 = 85
IF ( IP3 >= VA5 - 0.9 ) OR \C
  ( IP3 <= VA5 + 0.9 ) THEN VB29 = 88
IF IP3 <= ( VA5 - 1.0 ) THEN VB29 = 89
IF IP3 <= ( VA5 - 2.0 ) THEN VB29 = 90
IF IP3 <= ( VA5 - 3.0 ) THEN VB29 = 91
IF IP3 >= ( VA5 + 1.0 ) THEN VB29 = 87
IF IP3 >= ( VA5 + 2.0 ) THEN VB29 = 86
IF IP3 >= ( VA5 + 3.0 ) THEN VB29 = 85

```

**PAGE2

```

IF ( IP4 >= VA6 - 0.9 ) OR \C
  ( IP4 <= VA6 + 0.9 ) THEN VB30 = 88
IF IP4 <= ( VA6 - 1.0 ) THEN VB30 = 89
IF IP4 <= ( VA6 - 2.0 ) THEN VB30 = 90
IF IP4 <= ( VA6 - 3.0 ) THEN VB30 = 91
IF IP4 >= ( VA6 + 1.0 ) THEN VB30 = 87
IF IP4 >= ( VA6 + 2.0 ) THEN VB30 = 86
IF IP4 >= ( VA6 + 3.0 ) THEN VB30 = 85
IF ( IP5 >= VA7 - 0.9 ) OR \C
  ( IP5 <= VA7 + 0.9 ) THEN VB31 = 88
IF IP5 <= ( VA7 - 1.0 ) THEN VB31 = 89
IF IP5 <= ( VA7 - 2.0 ) THEN VB31 = 90
IF IP5 <= ( VA7 - 3.0 ) THEN VB31 = 91
IF IP5 >= ( VA7 + 1.0 ) THEN VB31 = 87
IF IP5 >= ( VA7 + 2.0 ) THEN VB31 = 86
IF IP5 >= ( VA7 + 3.0 ) THEN VB31 = 85

```

**PAGE3

```

IF ( IP6 >= VA8 - 0.9 ) OR \C
  ( IP6 <= VA8 + 0.9 ) THEN VB32 = 88
IF IP6 <= ( VA8 - 1.0 ) THEN VB32 = 89
IF IP6 <= ( VA8 - 2.0 ) THEN VB32 = 90
IF IP6 <= ( VA8 - 3.0 ) THEN VB32 = 91
IF IP6 >= ( VA8 + 1.0 ) THEN VB32 = 87
IF IP6 >= ( VA8 + 2.0 ) THEN VB32 = 86
IF IP6 >= ( VA8 + 3.0 ) THEN VB32 = 85
IF ( IP7 >= VA9 - 0.9 ) OR \C
  ( IP7 <= VA9 + 0.9 ) THEN VB33 = 88
IF IP7 <= ( VA9 - 1.0 ) THEN VB33 = 89
IF IP7 <= ( VA9 - 2.0 ) THEN VB33 = 90
IF IP7 <= ( VA9 - 3.0 ) THEN VB33 = 91
IF IP7 >= ( VA9 + 1.0 ) THEN VB33 = 87
IF IP7 >= ( VA9 + 2.0 ) THEN VB33 = 86
IF IP7 >= ( VA9 + 3.0 ) THEN VB33 = 85

```

**END

Putting it Together

This animation was designed with the particular floor plate graphic in mind, and the technician knew that groups of three blocks of each colour would work. This is why there is three backslashes in each of the GP rules. The application could just as easily have been built with only one block of colour in each rule string.

Whether one, three or ten backslashes are input into the GP rules, the individual VBs may have to be placed on the display more than once each, as the blocks may not be big enough to fill the desired area. This is okay; there is no limitation to the number of times a single VB point can be placed on the graphic, provided that the total number of points on the display is kept under 80. This point illustrates a valuable technique: build up the application to include the basic blocks you want, and then copy them as required to create the animation effect.

To further illustrate the repeated use of VBs on a display, observe the following excerpt from the link file for this animation. This shows four VBs used three times each in the same display:

```

:POINT      101-VB28  ZONE1          102 106  7 15 0.0_-          p ;
:POINT      101-VB28  ZONE1          102 114  7 15 0.0_-          d ;
:POINT      101-VB28  ZONE1          102 122  7 15 0.0_-          d ;
:POINT      101-VB29  ZONE2          128 106  7 15 0.0_-          p ;
:POINT      101-VB29  ZONE2          128 114  7 15 0.0_-          d ;
:POINT      101-VB29  ZONE2          128 122  7 15 0.0_-          d ;
:POINT      101-VB30  ZONE3           68 143  7 15 0.0_-          p ;
:POINT      101-VB30  ZONE3           68 151  7 15 0.0_-          d ;
:POINT      101-VB30  ZONE3           68 159  7 15 0.0_-          d ;
:POINT      101-VB31  ZONE4           94 143  7 15 0.0_-          p ;
:POINT      101-VB31  ZONE4           94 151  7 15 0.0_-          d ;
:POINT      101-VB31  ZONE4           94 159  7 15 0.0_-          d ;

```

TROUBLE SHOOTING AND ERROR MESSAGES

Problem All points on the display appear in a solid black bar.

Solution 1 The variable's ranges are not defined from beginning to end.

Solution 2 Check to ensure that the proper format for the rules are followed.

Problem Some points do not appear to follow the given rules.

Solution Ensure that the proper format for the rules are followed beginning with a %RULES and ending with a % symbol. Also, ensure that the range of variables effected by the rule is only in that particular rule.

Problem If points display manual off.

Solution Check to ensure that all the points are created.

Problem Points are to receive an alarm but do not.

Solution Ensure that the AL point is built up and that the Alarm priorities defined in the rules have at least an A1 for the alarm priority.

Problem When using analog VC variables, and the values change such that they should go into alarm, but no alarm occurs.

Solution Check to ensure that there is an ampersand character (&) before the LIMITS statement (i.e., &LIMITS).

APPENDIX A : DISPLAY COLOURS

Colour Options The colours available for use with GP programming are as follows. Using the prefix 'M' will cause any of the colours used for backgrounds to blink steadily on the screen. Note that the colours below can be used as either backgrounds or foregrounds.

BLACK

BLUE

BROWN

BWHITE (bright white)

CYAN

DGREY (dark grey)

GREEN

LBLUE (light blue)

LCYAN (light cyan)

LGREEN (light green)

LPURPLE (light purple)

LRED (light red)

PURPLE

RED

WHITE

YELLOW

APPENDIX B: VALID ASCII CHARACTERS

The following ASCII characters can be used with Group point applications. Do not use ASCII characters from the *IBM family character set* other than those displayed in the following table.

VALID ASCII CHARACTERS FOR GROUP POINT APPLICATIONS

!	"	#	\$	%	&	()	*	+
,	-	.	0	1	2	3	4	5	6
7	8	9	;	<	=	>	?	@	A
B	C	D	E	F	G	H	I	J	K
L	M	N	O	P	Q	R	S	T	U
V	W	X	Y	Z	[\]	^	