

GSP - G&M codes extension to ACSPL+

Reference Guide

Version 1.0

Jan 2014



Table of Contents

| | | |
|---|--------------------------------------------------|---|
| 1 | INTRODUCTION | 3 |
| 2 | GSP ADAPTATION TO DIFFERENT G-CODE DIALECTS..... | 3 |
| 3 | GSP ESSENTIALS..... | 4 |

Notice

The information in this document is deemed to be correct at the time of publishing. ACS Motion Control reserves the right to change specifications without notice. ACS Motion Control is not responsible for incidental, consequential, or special damages of any kind in connection with using this document.

Revision History

| Date | Revision | Description |
|--------------|----------|-----------------|
| Dec. 10 2013 | 1.0 | Initial version |
| | | |
| | | |

Trademarks

ACS Motion Control, SPiiPlus, PEG, ServoBoost, MotionBoost and NetworkBoost are trademarks of ACS Motion Control Ltd.

Windows and Visual Basic are trademarks of Microsoft Corporation.

EtherCAT is registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany.

Any other companies and product names mentioned herein may be the trademarks of their respective owners.

ACS Motion Control Ltd
1 Hataasia St
Ramat Gabriel Industrial Park
Migdal Ha'Emek 2307037 Israel
T +972 4 654 6440
F +972 4 654 6443
www.acsmotioncontrol.com
support@acsmotioncontrol.com

1 Introduction

This document defines the G and M codes extension of ACSPL+ for the SPiiPlus NT controllers, hereafter abbreviated as GSP.

G-code is a common name for the international standard RS-274D/ISO-6983. The standard defines the command language for numerically controlled machines.

G-code cannot be considered as a specific language, but rather as a collection of incompatible dialects with numerous additions/modifications. The proposed G-code extension (GSP) covers common needs, but is not intended to be compatible with any specific CNC machine.

2 GSP Adaptation to Different G-Code Dialects

The SPiiPlus NT controller receives information from the G/M-code program which has been automatically generated by the specific CAM SW tool, and in most cases has the ability to be adapted for the specific CNC machine. This capability is provided by a CAM SW post-processor, which can be adapted to use only those G/M-codes, which are supported by the specific CNC. So, if using SPiiPlus NT, the post-processor needs to be adapted to use only G/M commands, as defined in this guide.

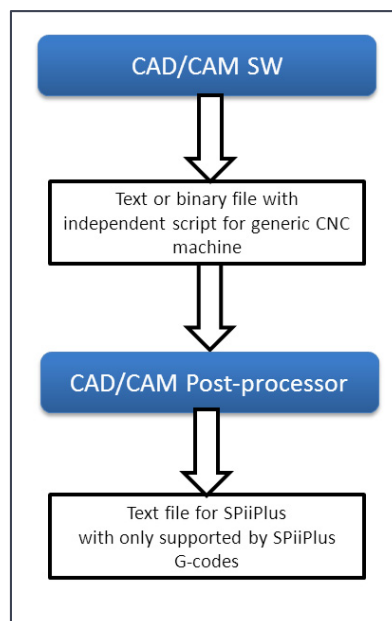


Figure 1: GSP adaptation to G-code dialects

In some cases, the ability to adapt the CAM post-processor may not be available. In these cases, GSP provides the ability to extend a supported G/M code-list by defining new G/M codes which are implemented by means of GSP subroutines. See [Subroutines](#) for details.

3 GSP essentials

3.1 Loading G/M Code Program

The G/M code program that is either generated by the CAD SW tools or written manually, should be saved as a text file. This text file can be loaded into the **SPiiPlus** controller buffer, either using the Program Manager of **SPiiPlus MMI Application Studio** or by the user host program using special functions of the ACS Library. Refer for the corresponding guides for details.

3.2 Compilation and Execution G/M Code Program

Once the G/M program is loaded into the **SPiiPlus** controller buffer, it is compiled. If there are unsupported G/M codes or if there is a syntax error, the controller generates an error and the Program Manager highlights the program line containing the error.

When the G/M program was successfully compiled, it can be executed exactly the same way as a regular ACSPL+ program is executed. The debugging capabilities (breakpoints, step-by-step executions, etc.) of the Program Manager are also supported for G/M programs.

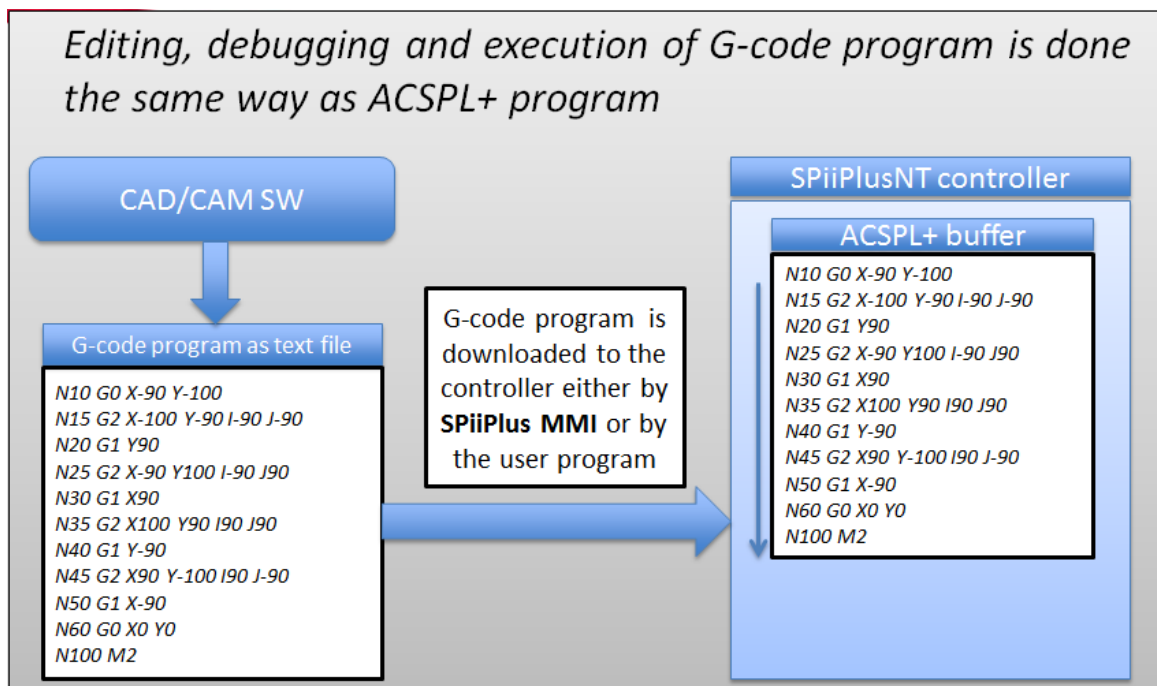


Figure 2: Compilation and execution of G/M code

3.3 GSP and ACSPL+

G/M code program can be used in any SPiiPlus program buffer. A program buffer can contain ACSPL+ program, pure GSP code, or any mix of ACSPL+ and GSP lines.

Mixing of GSP with ACSPL+ provides additional advantages to the user, such as:

- The ability to handle I/O synchronously with the motion

- The ability to check different conditions, like motor position, number of executed segments, etc., during the G/M program execution and perform any user-defined actions accordingly
- Activate specific SPiiPlus functions as PEG, MARK, etc.

The ACSPL+ commands or several commands are added to the G/M code program using Buffer Editor of SPiiPlus MMI Program Manager, exactly the same way as to regular ACSPL+ program:

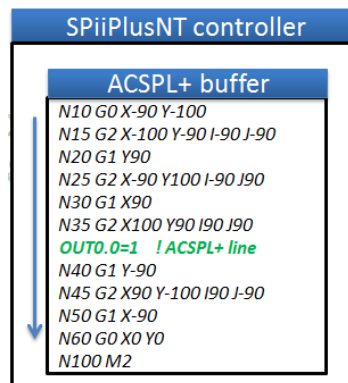


Figure 3: GSP and ACSPL+

The program that combines GSP and ACSPL+ commands should meet the following rules:

- GSP and ACSPL cannot be mixed within a single line; i.e. one code line can contain either a GSP statement or ACSPL statement..
- A GSP line starts with an N address followed by a number; e.g. N635 (N-address). The ACSPL compiler recognizes the N address signature and interprets the rest of line as GSP code.
- One GSP statement spans one line; the statement cannot continue on the next line; two statements cannot be placed in one line.

Note: Special attention should be taken if a user defines an ACSPL variable that resembles the N-address signature, like

```
int N01
```

Note: In this case, if the buffer contains an ACSPL line starting with N01=1000, the controller will recognize the line as GSP code, and will probably return an error.

Therefore, defining such variable names is not recommended in a buffer containing GSP lines, and in the D buffer if GSP is used in any buffer.

If a customer does not use GSP, he is not limited in variable names.

3.3.1 Addresses and Values

The GSP line consists of terms separated by spaces. Each term starts with a one-letter address followed by value. No space between the address and the value is allowed. The value can be either a literal constant or expression in parenthesis.

3.3.2 Value as Literal Constant

Literal constant values in GSP lines can be specified in either integer or real format. Any numerical format allowed in ACSPL statements can be used in GSP statements.

The following example of GSP line contains only literal constant values:

```
N15 G2 X-100 Y-90 R10
```

3.3.3 Value as Expression

A left square bracket placed immediately after an address letter opens expression value. The expression spans to the next right square bracket. If required, the expression may include nested parenthesis.

The expression syntax and semantics are similar to ACSPL expressions syntax.

The controller calculates expressions before executing containing the G code line. Within one G code line, expressions are calculated from left to right.

Expressions are most useful with coordinate addresses X, Y, Z, etc.

The following addresses do not allow expressions and must be followed by literal constant:

N – Line number.

G – Preparatory function.

M – Miscellaneous function.

P – When specifying digital output after M61/M62. However, P accepts expression if designates velocity, dwell, or subroutine parameter.

3.3.4 Using ACSPL Variables

ACSPL variables can be used in G code lines, but only within expressions. If a single variable is required to supply value to an address, the variable name should be enclosed with square brackets to make the controller treat it as trivial expression.

For example, in the following two lines use an ACSPL+ variable VEL that defines the default velocity for the axis with index 0. The first G1 motion will be executed using default X velocity divided by 2, but the second one will use the default X velocity. (Division by 60 is intended to convert units/sec to units/min.):

```
N15 G1 X-100 F[VEL(0)/60/2] ; VEL(0) is ACSPL+ variable  
N20 G1 X100 F[VEL(0)/60] ; VEL(0) is ACSPL+ variable
```

Either predefined standard variables or user-defined variables can be used in GSP expressions. User-defined variables should obey usual ACSPL scope rules. As such, to be used in a G code expression, the variable should be declared either in the same buffer (as local or global) or in D buffer.

3.3.5 User Units

SPiiPlus user units also affect GSP statements. A GSP value related to the axis position or velocity is measured in units specified in SPiiPlus for this axis. For example, if user unit of X axis is one millimeter, then GSP specifies X position in mm, and X velocity in mm/min.

The SPiiPlus units are initially defined during the axis setup and it is not recommended to switch the units during machine operation. For this reason, GSP does not support unit specification commands G20/21.

3.3.6 Comments

Different dialects of G code use different syntaxes to introduce comments.

GSP supports two forms of comments, both can appear only within G code line:

- Semicolon: marks the rest of G code line as comment
- Parenthesis: marks the enclosed text as comment. (Be careful, parenthesis within an ACSPL expression does not enclose the comment.)

The following G code lines illustrate two forms of comments:

```
N10 G0 X-100 Y-100 ; the rest of the line is comment
N15 G1 X[RPOS(0)+100] (Former parenthesis contain RPOS index)
```

3.3.7 The% Symbol

In many G code dialects, % symbol marks the beginning and the end of G code program.

In GSP, a line starting with % symbol is considered a G code line. GSP does not make any use of such line, but simply ignores its content. The rule applies only if % symbol is the first non-space character in a line; the same character between other characters may have other meaning.

3.4 Axis Mapping and Trajectory Planes

G-code uses 9 letter addresses (X, Y, Z, U, V, W, A, B, C) to designate axes.

A circular arc can be specified in planes XY, XZ, or YZ only. Function G17, G18, or G19 selects one of the planes.

If none of the axes X, Y, Z, U, V, W, A, B, C are defined in the controller, GSP implies that the default mapping of X, Y, Z, U, V, W, A, B, C axes to the first nine SPiiPlus axes (0 to 8). However, if the controller redefines one or more of X, Y, Z, U, V, W, A, B, C axes, then GSP follows the existing definitions. For example, if the controller's D buffer contains the statement

```
axisdef X=4,Y=12,A=0
```

GSP maps X to the 4th physical axis, Y to the 12th, and A to axis 0. Mapping of other axes does not change. The user should avoid conflicts: in the example above, X definition overlaps with the V default mapping (axis 4). Therefore V axis should not be used in GSP statements.

In GSP, plane XY is the default trajectory plane: lines and arcs specified in this plane build up a continuous trajectory; the controller applies a look-ahead algorithm to the trajectory. Any motion specified for other axes is considered as an independent PTP motion.

Axis mapping can be changed freely at any time as follows: G17 selects trajectory plane XY, G18 selects XZ, and G19 selects YZ.

Axes of the current trajectory plane (XY, XZ, or YZ) can be used with motion functions G0, G1, G2, or G3. Other axes can be used with motion function G0 and G1.

3.5 GSP Execution

The controller executes the GSP program exactly in same manner as regular ACSPL+ buffer, meaning that for each controller cycle one line of GSP program is executed. Most of G functions do not cause any motion; they define internal calculations or settings. Such functions are executed synchronously with the program execution.

More complex execution rules govern the motion functions G0, G1, G2, and G3. The motion commanded by G0, G1, G2, G3 does not always start when the line with G0, G1, G2, G3 is executed; actual motion usually starts once the next motion is commanded.

If a program contains GSP lines only, the G-code program execution is not affected by the specified above rules. However, these rules should be taken into account if GSP commands are mixed with ACSPL+.

For example, the following mix of GSP and ACSPL may show unexpected behavior:

`N10 G00 X1000 Y1000` ; Actual motion start will occur only once line N20 is executed

`TILL RPOS(X)>500` ; Infinite wait; the condition cannot be met because the motion has not started yet

`N20 G00 X2000 Y2000` ; Program execution never reaches this line as the program waits at the previous line

To fix the problem, reorder the lines as follows:

`N10 G00 X1000 Y1000` ; Actual motion start will occur only once line N20 is executed

`N20 G00 X2000 Y2000` ; N10 motion starts here; N20 motion will wait for the next motion command

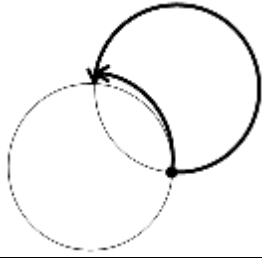
`TILL RPOS(X)>500` ; The condition will be met when X axis position reaches 500

Pay attention, due to G-code modality, a GSP line may omit motion function; however, if any of the addresses X, Y, Z, U, V, W, A, B, C is specified in the line, the last specified motion function is implied.

3.6 GSP Commands Descriptions

| Code | Name | Addresses | Address meaning | Command description |
|----------------|----------------|---------------------------|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| G00 | Positioning | X, Y, Z, U, V, W, A, B, C | Axis Position | <p>Function G0 (Rapid Motion) is equivalent to PTP motion with suffix M. Any combination of X, Y, Z, U, V, W, A, B, C addresses can be specified with G0. All specified axes are used to initiate a single PTP motion. The suffix M causes the maximum vector velocity to comply with axis velocity VEL for each involved axis.</p> <p>If an XSEG motion in a trajectory plane has been initiated by previous commands, the XSEG motion is terminated in a normal way, and PTP starts once the XSEG motion finishes.</p> <p>SPiiPlus buffer execution is delayed in the G0 line waiting for the motion termination. In general, G0 operation is similar to the following action sequence:</p> <p>If an XSEG motion is active in a trajectory plane, execute ends Wait for motion termination for all X, Y, Z, U, V, W, A, B, C axes Start PTP/M motion for all specified axes Wait for the PTP motion termination</p> |
| G01 (in-plane) | Linear segment | X, Y, Z, U, V, W, A, B, C | Axis Position | <p>If the statement specifies one or two axes, and all specified axes belong to the trajectory plane, the function adds a linear segment to XSEG motion. In this case, function execution includes the following actions:</p> <p>If no XSEG motion is currently active, a new XSEG motion in the current trajectory plane is initiated</p> <p>A new linear segment is added to the XSEG motion</p> <p>Usually, the function does not introduce a delay in the program execution. However, usual rules of motion queuing and segment queuing do apply; e.g., if the segment queue is full, the program is delayed until the new segment can be added.</p> |
| | | F | Feedrate in units/min | |

| | | | | |
|-----------------------|-------------------------------|---------------------------|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| G01 (out-of-plane) | Positioning | X, Y, Z, U, V, W, A, B, C | Axis Position | <p>If the statement specifies one or more axes that do not belong to the trajectory plane, the function acts similar to G0, but instead of maximizing velocity it uses the specified vector velocity. The vector velocity can be specified in the same statement (F address), or in a previous statement as G-code modal rules imply. In this case, function execution includes the following actions:</p> <p>If an XSEG motion is active in a trajectory plane, execute ENDS Wait for motion termination for all X, Y, Z, U, V, W, A, B, C axes Start PTP motion for all specified axes Wait for the PTP motion termination Like G0 function, out-of-plane G1 delays program execution until the motion termination.</p> |
| | | F | Feedrate in units/min | |
| G02 | Clockwise arc segment | X, Y, Z | Axis Position | <p>The functions can refer to trajectory plane only; only axes X, Y and Z can appear with functions G1, G2, G3. The function execution includes the following actions:</p> <p>If no XSEG motion is active, a new XSEG motion in the current trajectory plane is initiated New arc segment is added to the XSEG motion Usually, the function does not introduce a delay in the program execution. However, usual rules of motion queuing and segment queuing do apply; e.g., if the segment queue is full, the program is delayed until the new segment can be added. Functions G02 and G03 cannot change the trajectory plane; use G17, G18, or G19 for this purpose. For example, if the current trajectory plane is YZ, specification G2 with address X is an error. R can be specified with a sign: a positive value defines a smaller arc (less than or equal to 180°); a negative value defines a larger arc (more than 180°):</p> |
| | | I, J, K | Circle center offset from the current location in X, Y and Z direction respectively | |
| G03 | Counter-clockwise arc segment | R | Arc radius | |
| | | F | Feedrate in units/min | |

| | | | | |
|-----|-----------------------------------|---------------------------|-----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | | | |  |
| G04 | Dwell | P | Dwell time in seconds | <p>The function causes a delay for the specified number of seconds. It executes as follows:</p> <p>If an XSEG motion is active in a trajectory plane, the execute ENDS</p> <p>Waits for motion termination of any of X, Y, Z, U, V, W, A, B, C axis</p> <p>Waits for the specified number of seconds</p> |
| G09 | Exact stop | | | <p>The function specifies deceleration to zero in the end of the segment. The function is not modal and should be specified in-line with one of segment-definition functions G1÷G3.</p> |
| G10 | Coordinate origin | X, Y, Z, U, V, W, A, B, C | Coordinate origin | <p>The function specifies the coordinate origin. The origin is changed only for the specified axes; other axes retain their previous origin. If absolute programming is used, the value of origin is added to the coordinate values specified in subsequent G0÷G3 functions..</p> |
| G17 | Trajectory plane XY | N/A | N/A | The function selects the XY trajectory plane. |
| G18 | Trajectory plane XZ | N/A | N/A | The function selects the XZ trajectory plane. |
| G19 | Trajectory plane YZ | N/A | N/A | The function selects the YZ trajectory plane. |
| G40 | Tool radius compensation off | N/A | N/A | The function cancels tool radius compensation. |
| G41 | Tool radius compensation left | D | Cutter radius | The function initiates building left equidistant trajectory of the cutter center. |
| G42 | Tool radius compensation right | D | Cutter radius | The function initiates building right equidistant trajectory of the cutter center. |
| G43 | Tool offset compensation positive | H | Tool length | The tool length value supplied by the H address is added to the perpendicular axis position. The perpendicular axis is Z, Y, or X depending on trajectory plane. |

| | | | | |
|-----|-----------------------------------|-----|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| G44 | Tool offset compensation negative | H | Tool length | The tool length value supplied by H address is subtracted from the perpendicular axis position. The perpendicular axis is Z, Y, or X depending on trajectory plane. |
| G49 | Tool offset compensation cancel | N/A | N/A | The function cancels tool offset compensation. |
| G53 | Machine coordinate system | N/A | N/A | The function instructs to ignore previously specified axis origins (G10) for the current line only. |
| G61 | Modal exact stop | N/A | N/A | The function specifies deceleration to zero at the end of the segment. Unlike G09, the function is modal and works for the current statement and all subsequent statements. |
| G64 | Cancel G61 | N/A | N/A | The function cancels the G61 action and returns to default segment processing. |
| G90 | Absolute programming | N/A | N/A | The function defines absolute programming in the current line and subsequent lines. The specified axis position value is absolute related to axis origins. |
| G91 | Incremental programming | N/A | N/A | The function defines incremental programming in the current line and subsequent lines. The specified axis position value is incremental related to the last specified position. |
| M00 | Stop | N/A | N/A | The function terminates program execution. The action is equivalent to the STOP command in ACSPL+. |
| M61 | Set digital output | P | Digital output | The function sets the digital outputs to 1 (high level) one or more. The function should be followed by one or more P addresses. Each P address specifies one digital output to be set. The output is specified as pair of output variable index and output bit, separated by dot: P0.1. If M61 is specified in line with one of G0÷G4 functions, setting the digital outputs is executed synchronously with the start of motion or dwell. If no G0÷G4 function is specified in the line, the setting is not synchronized to motion and occurs once the controller executes the line |

| | | | | |
|-----|----------------------|---|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| M62 | Reset digital output | P | Digital output | <p>The function resets to 0 (low level) one or more digital outputs. The function should be followed by one or more P addresses. Each P address specifies one digital output to be reset. The output is specified as pair of output variable index and output bit, separated by dot: P0.1.</p> <p>If M62 is specified in line with one of G0÷G4 functions, setting the digital outputs is executed synchronously with the start of motion or dwell. If no G0÷G4 function is specified in the line, the setting is not synchronized to motion and occurs once the controller executes the line..</p> |
|-----|----------------------|---|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

3.7 GSP supported Addresses

The following table specifies one-letter addresses used in GSP:

| | |
|---------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| N | Line number |
| G | Preparatory function |
| M | Miscellaneous function |
| X, Y, Z, U, V, W, A, B, C | Axis position |
| F | Feedrate in units/min |
| P | <p>Dwell time in seconds if used after G4 function.</p> <p>Final feedrate in units/min if used after G0÷G4 functions.</p> <p>Digital output to set/reset if used after M61/M62 functions.</p> <p>Subroutine parameter, if used after G or M function interpreted as subroutine.</p> |
| I, J, K | Arc center position |
| D | Cutter radius offset |
| H | Tool length offset |
| R | Arc radius |
| S, Q, T | Can only be used as parameters in subroutine call |

Addresses not specified in the table can be used in subroutine call to define subroutine parameters. If used outside the subroutine call, not specified address is discarded and its value is ignored.

3.8 Subroutines

GSP supports subroutine calls. The subroutines are used if there is a need to define a new G or M code command, which is not listed in the GSP predefined commands list.

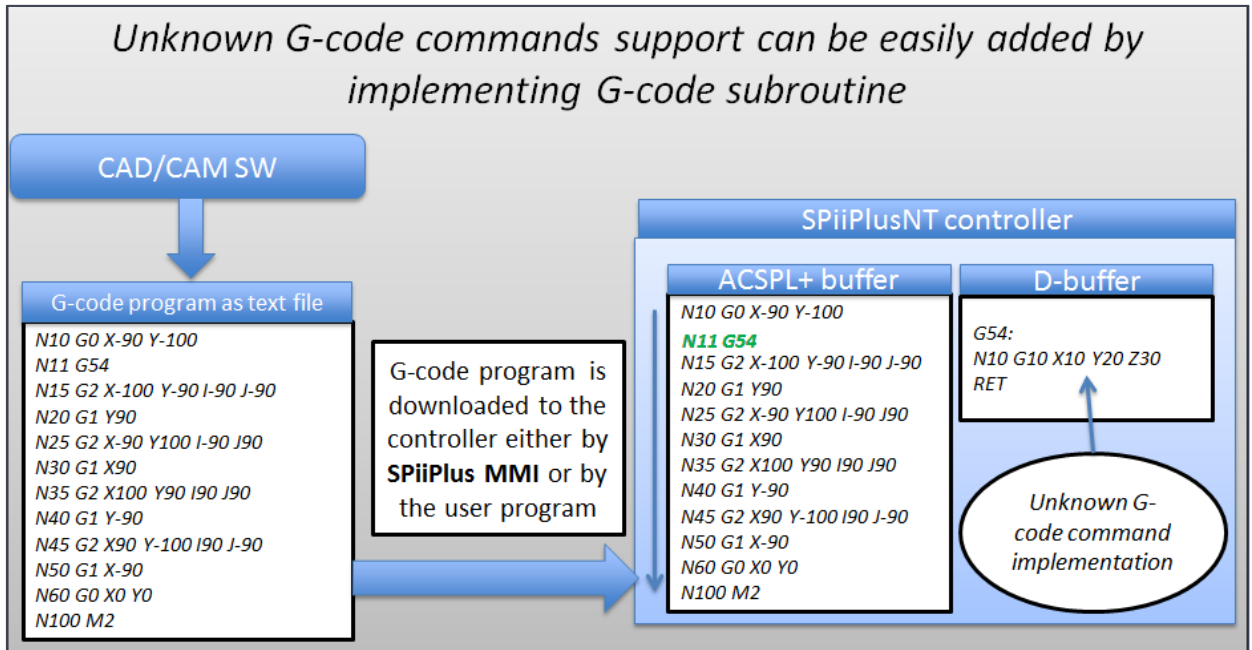


Figure 4: GSP and ACSPL+

3.8.1 Subroutine Declaration

The subroutine declaration follows ACSPL+ syntax. The subroutine starts with a label and ends with ret command. The subroutine body can contain either ACSPL lines or G code lines of any combination..

Not every subroutine can be called from a G code line. To be callable from G code, subroutines must have a special label: callable labels from G code consists of G or M letters followed by digits. For example, label G59 can be called from G code line; however, label G59D cannot.

3.8.2 Subroutine Call

GSP interprets any G or M address as a potential subroutine call.

Once a G or M address is encountered, GSP first checks if the specified value falls within the predefined codes (see [GSP commands description](#)). If the value does not appear among predefined functions, GSP searches through available subroutines. GSP first examines subroutines in the current buffer; if not found, then subroutines in D buffer are examined.

If a subroutine with an appropriate label exists in either a current or D buffer, GSP executes call to the subroutine. If no predefined function exists and no appropriate subroutine was found, GSP reports an error.

This way, predefined G/M codes cannot be overridden with subroutines. Even if a G01 subroutine is defined in a buffer, code G01 will be interpreted as Linear Segment, but not as subroutine call.

There is a slight difference in value interpretation in two cases:

- Predefined codes are interpreted by arithmetic value. For example, addresses G1, G01, and G001 are considered identical and are all interpreted as Linear Segment.
- Subroutine calls follow literal specification, so that G54 calls subroutine G54, but not G054 or G0054. Therefore, addresses G54, G054, and G0054 may call different subroutines.

If a line contains a subroutine call, the whole line is interpreted as subroutine call. All other addresses on the line are interpreted as subroutine parameters accessible through functions gParamAddr/gParamValue. Therefore a subroutine call and its parameters should appear in a separate line. No other subroutine call or predefined G/M code is possible on the same line.

Once a subroutine call occurs, the controller executes the subroutine body until the ret command. Then the controller returns to the line next to the subroutine call, and continues executing the program.

3.8.3 G/M Specification with Digital Point

GSP also supports calling ACSPL subroutines using G/M specification with decimal point.

Once GSP encounters a G or M value with a decimal point, the following processing occurs:

- The construction is interpreted as subroutine call
- All characters left of the decimal point including the letters G or M constitutes the base subroutine name
- The base is supplemented with three positions taken from the right to decimal point. If less than three symbols are to the right of the decimal point, the missing positions are filled with 0.
- The resulting name is used to call ACSPL subroutine

For example, address G5.1 calls subroutine G5100; address M80.123 calls subroutine M80123, and so on.

The following limitations apply:

- Only one decimal point is allowed in a G/M value
- Maximum 10 digits are allowed left to decimal point
- Maximum 3 digits are allowed right to decimal point

3.8.4 Access to Parameters

Once a G or M address causes subroutine call, subsequent addresses can be used as subroutine parameters. Usually, address P supplies parameters to subroutine; however, other addresses can be used if necessary. For example, the following line calls G500 subroutine with three parameters 2, 300.25, and -1.7:

```
N10 G500 P2 P300.25 P-1.7
```

To access parameters from the subroutine body, two new embedded ACSPL functions are implemented:

- gParamAddr has one integer argument and returns the address of the G function call parameter. This argument supplies the parameter number: argument 1 corresponds to

the G address itself, argument 2 corresponds to the first address after the G function call, 3 – to the second, and so on.

Argument 0 can be used if necessary to access address before the G/M function address. The function returns the parameter address as its ASCII code; e.g., if the address is P, the function returns 80.

- gParamValue has one integer argument and returns the value of the G function call parameter. This argument supplies the parameter number: argument 1 corresponds to G address itself, argument 2 corresponds to the first address after the G function call, 3 – to the second, and so on. For example, in the function G500 called by the above line, gParamValue (2) returns 2, gParamValue (3) returns 300.25, and gParamValue (4) returns -1.7.

3.8.4.1 gParamAddr

Description

gParamAddr can be used within an ACSPL subroutine called from G code. The function provides access to the parameters specified after the subroutine call and returns ASCII code of the corresponding parameter address.

Syntax

gParamAddr (Index)

Arguments

| | |
|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Index | An integer number that specifies the position of the parameter after the function call. Value 1 specifies the G/M address itself, value 2 specifies first address after the function call, 3 – second address after the function call, and so on. The value 0 is used to access parameters specified before the G/M command. |
|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Return value

The function returns an integer ASCII code of the corresponding address. For example, if the first address after the function call is P0.01, function **gParamAddr (2)** in the subroutine body returns 80, which is ASCII code of P.

The function returns zero in the following cases:

- The argument is negative
- The argument value exceeds the number of addresses in G code line after the function call
- The function was called not from within the G code

3.8.4.2 gParamValue

Description

gParamValue can be used within the ACSPL subroutine and is called from G code. The function provides access to the parameters specified after the subroutine call. The function returns the value specified in the corresponding parameter address.

Syntax

gParamValue(Index)

Arguments

| | |
|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Index | An integer number that specifies the position of the parameter after the function call. Value 1 specifies the G/M address itself, value 2 specifies first address after the function call, 3 – second address after the function call, and so on. The value 0 is used to access parameters specified before the G/M command. |
|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Return value

The function returns a real value of the corresponding parameter. For example, if the first address after the function call is P0.01, function gParamValue(2) in the subroutine body returns 0.01.

The function returns zero in the following cases:

- Argument is negative
- Argument value exceeds the number of addresses in G code line after the function call
- The function was called not from the G code

Example

The following example subroutine implies to be called from G code line like

```
N100 G655 P0.001 P2.25
```

where two parameters after G655 call supply target values for X and Y:

```
G655:                                ! Subroutine label
if (gParamAddr (2)<>80) | (gParamAddr (3)<>80)
    disp "Wrong parameters"         ! Two P addresses expected
    stop
end
ptp (0), gParamValue(2)             ! Move X axis to required position
ptp (1), gParamValue(3)             ! Move Y axis to required position
ret
```

3.8.5 Subroutine Scope

The G subroutine can be declared either in regular buffer, or in D buffer.

When G subroutine is declared in a regular buffer it has a local scope and can be called from the same buffer only.

When G subroutine is declared in a D buffer it has a global scope and can be called from any buffer.

Once GSP encounters a potential subroutine call, GSP first looks for a subroutine in the current buffer where the calling line is located and if not found, GSP looks in D buffer. If there is no suitable label in the current buffer nor in the D buffer, GSP tries interpreting the G/M address as standard G/M function.

The G subroutine body can access any object accessible in the buffer where G subroutine is declared. G subroutine is declared in a regular buffer and can use any local or global variable declared in this buffer and also global variables declared in D buffer. G subroutine is declared in D buffer can use only variables declared in D buffer.

3.9 Modality

GSP follows usual G-code modality rules. For example, if a GSP statement includes any axis addresses, but lacks G function that requires axis addresses (G0÷G3), the controller implies the last specified function.

Modality is local to a buffer; i.e., modal implications are restricted to a buffer, but do not affect programs in other buffers.

3.10 GSP and XSEG

GSP statements initiate the XSEG command which support Extended Segmented Motion without suffixes.

The first specified axis address defines the leading axis. Corresponding motion uses the motion parameters of the leading axis and applies automatic corner processing.

Additional information can be found in the **ACSPL+ Programmer's Guide** and **SPiiPlus Command & Variable Reference Guide**.