Table of Contents

An introduction to MAP

Map is a specialist interpreted reduced instruction set programming language for industrial applications including motion control and CNC. Because the language has a reduced instruction set it is fast and efficient. The difference between versions of map is normally the number of data fields which are present in each version. Common to all versions of map is a 32-bit Command structure, this includes an 8-bit code number, a four bit condition code which affords up to 16 options for each command, the remainder of the 32 bits is divided up into identifiers which identify the state of each field whether they are immediate data or register fields. In four axis system there will be one command structure and four 32-bit data fields making 20 bytes per command in total.

Map data fields are designed to be flexible and in general there are six different types of data field these are **register** only where the value entered is an integer value representing the number of a register whose contents will be referenced, **long** data which is a 32-bit integer value, **float or register** which is the integer value of a register or a constant float value, a **dimensioned float or register** this means that the value can represent metric or imperial or be an integer value representing a register, **long integer or register** and finally there is a type that represents **strings** such as file names.

Map is designed to be as safe as possible every effort is made to try to prevent the operator from making erroneous data inputs. When the operator has finished editing a program the system will attempt to simulate the program to ensure that it will operate safely. Obviously simulation is not 100% and can only therefore afford a degree of code safety. Because map is so flexible it is very easy for the user to write values into registers which can be used to position an axis, during simulation some values are taken from a simulation menu. For example during running a program it may be that a particular position is calculated from an analogue input. During simulation the analogue input may not operate since the processes not operational. The analogue input is therefore simulated by using a value from the simulation menu.

It is possible for the user to use the graphics tools to simulate a program on the screen and obviously this has the same limitations as a simulation check e.g. that it is not possible to fully simulate the process. Where a G code command is used, the G code is imported and then is converted to map, normally in G code the positions are calculated as immediate data and therefore it is possible to fully simulate the program. Immediate data means that the value is written into the data field as a constant. For example: MOVE, 100, 150.1, 55.1 is an example of immediate data meaning that the data has been written as a series of constants. It is possible to write the same command using registers, where the position has been calculated or previously written into the register and example of this is MOVE, R000, R001, R002, where the positions are taken from registers zero through to 2.

This manual assumes that the programme has a reasonable level of programming ability there is a more in-depth manual for those who require more help. The intention of this manual is to provide a short description of each of the commands, and to provide some information about how they will be simulated to assist a programmer in writing successful map programs.

Map Programmers Quick reference guide

Abs mode(Absolute mode)

This command is used to set the motion controller into the absolute measurement mode. Which means that the datums are ignored therefore if we assume that the the origins are set to X=500 Y=500 a move of X=100 y=100; would give a position of 100 + 500 = 600 for both the X and Y axis. After executing this command the position achieved by the command move of X=100 y=100 will be X = 100 and Y = 100 Note that this command only persists for 1 move command and is automatically cancelled after every move.

Condition codes:
0 Datum : Use Origins (datums) this is the normal mode
1 Absol : Measure from 0 the datums will be ignored and dimensions will be measured from axis zeros

ADC (get an analogue to digital reading)

Map ADC command where a single or range of registers may be used to collect the results Every time the command is called the index is incremented and the result written from the start to the end register when the the end register is reached the index is set to the start register so that the range of registers is preserved enabling easy averaging of the results using the average command.

Data field useage:
Data[0] = 0-7 on the SMC or 0-3 on the PMC4 data is a constant long
Data[1] = the start register data is a constant long
Data[2] = the end resister to store the data data is a constant long

ADDMULTIPL (Add to multiple registers)

MAP ADDMULTIPL (ADD multiple) command is designed to allow the user to add a value to a range of registers. Typical uses are for updating registers used to hold values or positions. It can also be used to subtract a value by adding a negative value. The start register must have a lower value than the end register or the command will fail.

Data field useage:
Data[0] = Start Register, data a constant register number representing the start register in the range.
Data[1] = END Register, data a constant register number representing the end register in the range.
Data[2] = Data Register, is the data to be added where data is a constant float or register number.

AIRDRILL (drill holes or weld a stud command)

MAP DRILL command air operated drill moves at full speed to compensated X, Y position then operates the head solenoids.

Data field useage:
Data[0] = the X position of the drill point displayed in metric or imperial, data is a constant float or register
Data[1] = the Y position of the drill point displayed in metric or imperial, data is a constant float or register
Data[2] = the head number on the range 0-2 data is a constant long or register.

ANALOG_OP (analogue output)

Map ANALOG_OP command run time used to set the analogue op in % where 100% = 10volts and -100% =-10 volts if the selected channel is in use by the motion controller the command will exit function has no effect during simulation.

Data field useage:
Data[0] = ADC channel the data is constant long
Data[1] = the analogue output value in percent data is a constant float or register

ARC (product circular movement)

Map ARC command is use to produce circular motion The positions are relative to the current axis positions The command uses three data fields these are X,Y centre points and the angle in degrees for anti clockwise motion use +ve numbers else -ve.

Data field useage:
Data[0] = the centre X relative position displayed as imperial or metric, in constant floating or register
Data[1] = the centre Y relative position displayed as imperial or metric, in constant floating or register
Data[2] = the angle in degrees, in constant floating or register

AVERAGE (average a range of registers)

Map AVERAGE command where a range of registers are averaged and the results written to a result register.

Data field useage:
Data[0] = the start register data is a constant long
Data[1] = the end resister to store the data is a constant long
Data[2] = result register where the results are stored

BEGIN (mark the beginning of a subroutine)

MAP BEGIN COMMAND the command is used to indicate the start of a subroutine and the end of main code block it behaves just like and END command

Data field useage:
Data[0] = the subroutine number allocated by the user the data is a constant

BRANCH (branch if not equal)

MAP BRANCH_NE the command is used to undertake a comparison and then branch by the number of lines specified by DATA[2] if the equation is not true.

Data field useage:
Data[0] the base register to test data is a constant long register number
Data[1] is a constant data value or register number to test against the base register
Data[2] is the number of lines to skip if the equation is not true.

Condition code gives the test options:

0	==	: Equal.
1	!=	: Not Equal.
2	>	: Greater than.
3	>=	: Greater or Equal.
4	<	: Less than.
5	<=	: Less or Equal.
6	BitL	: Register Bit = 0.
7	BitH	: Register Bit = 1.
8	PortL	: Input Bit = 0.
9	PortH	: Input Bit = 1.
10	FALSE	: Always Jump FALSE.
11	ONKEY	: OnKey Function activated branch if a function was called.
12	IpAct	: Input Bit Active (if an input is active eg if PNP and input is logic high or NPN and input is logic low) branch on false.
13	InMot	: In motion, an axis is moving.

CALL (call subroutine)

MAP CALL SUBRT call subroutine function FUNCTION COMMAND calls a subroutine written in the same map file using a begin command Subroutines are numbered based on the Begin command that was used to start the subroutine in the code with numbers starting with 0 to 99.

Data field useage:
Data[0] = the number of the subroutine to call.

CALL MAP (call a map program as a child process)

Call MAP function Loads a map function from memory and runs it allocates memory and copies the registers to a set reserved for the child process on ending the process these registers will all be copied back to the parent process if the CopyBack command has been called the child process registers will be copied to the parent process as specified by the CopyBack command. End is used to return back to the parent process The data fields are joined together to form the file name.

Data field useage:
Data[3:0] = file name.

CHANGE_OP (change output)

Map CHANGE_OP command run time function Allows the user to change a single bit of the output port does nothing during simulation.

Data field useage:
Data[0] = the number of the bit to change data is constant long or register
Data[1] = the value of the state where 0 = reset the bit and none zero is set data is constant long or register

CIRCLE (draw a filled or unfilled circle of the canvas)

Draws a filled or unfilled circle on the canvas.

Data field useage:
Data[0] is the x start point in percent data is constant or register number.
Data[1] is the y start point in percent data is constant or register number.
Data[2] in the radius percent data is constant or register number.
Data 3 is the bit field quadrants command in constant bit field, where bit 0 = 0-90 bit 1 = 90-180 bit 2 = 180-270 bit 3 = 270-360

Condition code:

```
0     Fcir  : Filled circle.
1     Cir   : Circle.
```

CLR_CANVAS (clear the canvas command)

MAP CLR_CANVAS command used to clear the canvas of all data as a clear screen command Draws a rectangle to full size if the canvas in the selected background colour. The data fields are not used by the command itself and therefore are available for a 16 character comment. The message is displayed only in the edit mode, does nothing during simulation or on the screen during run.

Close File (closes an open file)

Closes a file that was opened using open file. Note no memory associated with the file is freed and this must be freed using the free command after closing the file.

Data field useage:
Data[0] = The register that holds the file handle to the opened file channel.

COLOR (set the drawing and background color)

Map Set_Colors command used to set the back ground and foreground colors for the current canvas colors are expressed as 16 bits per pixel RRRRR,GGGGGG,BBBBB where black is 0 and peak white is 0xFFFF Peak RED = 0xF800, Peak GREEN = 0x07E0, Peak Blue = 0x001F.

Data field useage:
Data[0] = Foreground color expressed in constant HEX long
Data[1] = background color expressed in constant HEX long

Colour definitions (note 0x means hexidecimal)

BLACK	0x0000
BLUE	0x000F
GREEN	0x03E0
TURQUOISE	0x03EF
RED	0x7800
MAGENTA	0x780F
YELLOW	0x3BE0
LIGHT_GREY	0xCE7B
GREY	0x7BEF
Peak BLUE	0x001F
Peak GREEN	0x07E0
CYAN	0x07FF
Peak RED	0xF800
Peak MAGENTA	0xF81F
Peak YELLOW	0xFFE0
Peak WHITE	0xFFFF
Peak PURPLE	0xC3BE
MID_RED	0xC800
DARK_GREEN	0x0300
BOTTLE_GREEN	0x0100
WHITE	0xE77C
CYAN	0x27F7

COOLANT (sets the coolant to off, spray or flood)

MAP Coolant command, this command is used to set the coolant in 3 modes off, mist, or flood the command has no data and uses the condition code data. Since the data fields are not used by the command itself and therefore are available for a 16 character comment. The message is displayed only in the edit mode, does nothing during simulation or on the screen during run.

Condition codes:
0 M9: Coolant off.
1 M7: Coolant Spray.
2 M8: Coolant Flood.

COPYBACK (set the registers to copy back to the parent process)

Map CopyBack command the function sets the values of the start and end register to copy back to the parent process used with the call map function this command must be used in each child process if the entire register range is not to be copied to the parent on return to the parent process.

Data field useage:
Data[0] = the start register number to copy back note this must be the lowest value
Data[1] = the end register number to copy back note this must be the highest value
example if Data[0] =100 and Data[1] = 200 registers from 100 to 200 will be copied back to the parent process.

COSINE (compute the cosine of a value)

Map cosine command calculates the cosine of a register or value in Data[1] and stores the result in Data[0].

Data field useage:
Data[0] is a constant giving the result register computed to a floating point and converted to Data 0 type.
Data[1] is a constant float or register containing the operand value.

DECJMP (decrement a register and jump if not zero)

Map DECJMP command is use to repeat an action register content times similar to a for loop is other languages is the value of the register specified by Data[0] is not zero the register will be decremented and the process will jump to the line number specified in Data[1]. The command will exit when the register changes from 1 to zero after being decremented.

Data field useage:
Data[0] = the constant register number of the counter register.
Data[1] = the constant value of the line number to jump to.

DO (marks the start of a do-while loop)

MAP DO command used to set the start of a do while loop a error is generated if map does not encounter an equal number of do and while commands There are no data fields associated with this command, and therefore the data fields, therefore are available for a comment of up to 16 characters. The message is displayed only in the edit mode does nothing during simulation or on the screen.

DO MOVE (start motion and enter a do while loop)

MAP DO_MOVE command is used to undertake motion and start a while loop at the same time The while loop is the same as a normal while except that it is terminated if the condition becomes false or the motion completes. If a move is terminated early by the action of the while loop, the position is taken when the axis stops and written into the destination register so that the next move will correctly start from the point where the command terminated there may be a small error due to the granularity of the encoder resolution. In relative axis this error may build up where there are large numbers of do moves consideration should be given to this effect
if an axis does not need to move.

Data field useage:
Data[0] = X position constant or register (float).
Data[1] = Y position or register (float).
Data[2] = Z position or register (float).
Data[3] = axis 3 position or register (float) (PMC4 only).

Drawcan (Draw Canvas command)

MAP DRAW_CANVAS command used to set what is shown on the canvas were bits equal to 1 are on shown and 0 is off not shown the command has a two modes set in the Condition code via a wizard these are sets options and redraw or set options only. This command is supported by two help wizards there is a wizard to aid setting the bit options and a wizard to set the condition code.

Data field useage:
Data[0] is a bit field where:
Bit 0 = live draw e.g. move etc. causes a trace on the TPI showing where trace the movement has passed.
Bit 1 = feed rate is visible on the canvas.
Bit 2 = power bars on the canvas screen.
Bit 3 = tool visibility, e.g. the tool number is show on the canvas.
Bit 4 = RPM visibility, e.g. the RPM meter is show on the canvas.
Bit 5 = TPI visibility, e.g. show the Tool Path Image on the canvas.
Bit 6 = DRO visibility, e.g. show the active DROs on the canvas.
Bit 7 = information bar, e.g. show or not map line numbers
Bit 8 = DRO 0 visibility, setting the bit forces DRO visibility of DRO X axis
Bit 9 = DRO 1 visibility, setting the bit forces DRO visibility of DR1 Y axis
Bit 10 = DRO 2 visibility, setting the bit forces DRO visibility of DR2 Z axis
Bit 11 = DRO 3 visibility, setting the bit forces DRO visibility of DR3 axis 3

Note the clearing any of the respective bits does not clear the item from the screen therefore to present a canvas with all of the above disabled set the data fields to zero and after executing the show command redraw the canvas. This command is often used in tool commands to stop the movements needed to load the tool change child processes from drawing on the canvas and spoiling to tool path image.

Force DRO bits:
Bits 9 to 12 are used to force the DRO display for unused motion axis for example setting bit 10 Z on a two axis system will force the Z DRO to be active. But setting bit8 & 9 will have no effect sine they on the example 2 axis system will already be displayed

Condition code:

```
0     O->D  :     Set the option flags and then redraw the screen.
1     Data  :     Set the option flags only.
2     NNNN  :     Update the DRO positions etc ignore the data in Data[0].
```

DRAW LINE (draw a line PMC4 version)

MAP DRAW LINE command draws a line from Data[0] X start to Data[2] the X count
and Data[1] Y start to Data[3] for the Y count uses the background color already set. If the start position + the count exceeds 100% the width will be reduced to fit.

Data field useage:
Data[0] = X start position data is constant float or register.
Data[1] = Y start position data is constant float or register.
Data[2] = X count e.g. the length of the line data is constant float or register.
Data[3] = Y count e.g. the height of the line data is constant float or register

DRAW_POS (set the drawing position on the canvas)

Map DRAW_POS command used to position the drawing pointer on the canvas in X, Y position in percent if X or Y are set to more than 100% the values are set to 100
Data[0] = X position in percentage of the canvas date constant float or register.
Data[1] = Y position in percentage of the canvas date constant float or register.
DRAW RECT (draw rectangle)
MAP DRAW RECT command draws a filled rectangle from Data[0] X start to Data[2] the X count and Data[1] Y start to Data[3] for the Y count uses the background color already set if the start position + the count exceeds 100% the width will be reduced to fit.

Data field useage:
Data[0] = X start position data is constant float or register.
Data[1] = Y start position data is constant float or register.
Data[2] = X count e.g. the length of the rectangle data is constant float or register.
Data[3] = Y count e.g. the height of the rectangle data is constant float or register.

Drill (Equivalent to G81 cycle drilling command)

Map drill command is used for drilling applications operates in the following manner in the relative mode the index value is used as a loop counter In the absolute mode works as follows: 1) does a rapid move to position X,Y 2) does a rapid move to position R 3) does a feed move to Z
In the relative mode works as follows: 1) does a rapid move to relative position X,Y (e.g. current and X and Y position + X and Y respectively) 2) does a rapid move to position R 3) does a feed move to Z 4) if L is greater than 1 decrements L and repeats lines 1 to 4.

Data field useage:
Data[0] = The X position.
Data[1] = The Y position.
Data[2] = The Z position.
Data[3] = The Start position of the feed move e.g. drilling action

Condition code:

```
0     Abs:  Absolute mode
1     Rel:  Relative
```

END (end of map or a subroutine)

Map END command which is used to terminate a child process or signify the end of the primary MAP program these are the uses for END with the condition code set to 0 normal end With the condition code set to FUNCTION end END is used to signify the end of a sub routine initiated using a Call Subroutine command This command tries to set conditions as they were when the program started therefore the speed will be set to the defaults as set in the speed menu the canvas will be redrawn the torque will be set to 100% there are not data fields associated with END var.

ErrorBox (Draw an error box with a message) Added to version 1.16.4

This command is used to display an error message in red message box. The command waits for a button to be pressed.

Data field useage:
Data[0] = 16 bytes of string for the message.

EXIT (exit from a process)

MAP EXIT causes map to exit from the current process if the process is a child then map returns to the calling process if the process is the main process stops execution of MAP. The data fields are not used by the command itself and therefore are available for a comment. The message is displayed only in the edit mode The command has two condition code options 0 = shows E–> and from the selector box Exit program, exits all map programs 1 = shows C–> and from the selector Exit child process, exits the current child program.

Condition code:
0 E--> : Exit program (exit map altogether)
1 C--> : Exit child process.

Export (export data on the serial port)

MAP EXPORT command used to export data via the serial port. The data is a series of registers in the range data[0] to data[1] note data[0] must contain a lower value than data[1] normally the recipient is another controller or controllers for data sharing.

Data field useage:
Data[0] = The constant register number of the first register in the range to be exported.
Data[1] = The constant register number of the last register in the range to be exported.
Data[2] = The constant slave number range 1 to 254 of the intended slave if the controllers are working on CAN or RS484 has no effect on RS232

FEED RATE (set the axis speed to global speed in inches or mm per minute)

MAP Feed Rate sets the feed rate in feet per minute or metres per minute according to the data in data[0] Automatically adjusts the speed depending up the measurement standard as the database is stored in mm per minute but displayed in inches or MM.

Data field useage:
Data[0] = the feed rate in constant float or register representing travel in mm or inches per minute

FILE_STATS (provides file statistics for the open file)

MAP FILE_STATS is used to load file data and characteristics into a register or registers The function should be used immediately after opening a file to ensure that the latest file handle is captured. Once the file handle has been determined the function can be used at any time with the condition code set to one or more to look at a parameter or parameters. With the exception of the file handle, any data that is not required should have absolute data entered in the register number position, any number will do zero is preferred. When the system does not discover a register it will not write the data to the appropriate register thereby providing determinable data.

Data field useage:
Data[0] = the destination register number for Filehandle CC0 or the Filehandle to use if CC!=0
Data[1] = the destination register number for the pointer to the memory buffer or write a absolute number (0) for not needed if the value is not 0 the register type will be set as a pointer
Data[2] = the destination register number for file pointer or write a absolute number (0) for not needed
Data[3] = the destination register number for the file size or write a absolute number (0) for not needed

Condition code:

```
0     Last  : Use last opened file data.                    /*
1     File  : Use FileHandle.
```

FIND MAX (find the maximum in a range of registers)

Map FIND MAX command is used to find the resister that contains the highest value of a series of registers where some registers contain the same value find maximum will return the first e.g. the lowest number register number containing that value note that the start must hold a lower value than the END.

Data field useage:
Data[0] = The number of the first register in the series START the value is a constant long
Data[1] = The number of the last register in the series END the value is a constant long
Data[2] = The number of the results that hold the number of the register containing the maximum value data is a constant long.

Find Min (find the minimum in a range of registers)

Map FIND MIN command is used to find the resister that contains the lowest value of a series of registers where some registers contain the same value find minimum will return the first e.g the lowest number register number containing that value note that the start must hold a lower value than the END.

Data field useage:
Data[0] = The number of the first register in the series START the value is a constant long.
Data[1] = The number of the last register in the series END the value is a constant long.
Data[2] = The number of the results that hold the number of the register containing the minimum value data is a constant long.

FontSize (allows the user to set the size of the font)

Fontsize command allows the user to set the font size of the true type fonts which are set in Data[0] or the fixed font which is set in the CC.

Data field useage:
Data[0] = The true type font size this is a future option.

Condition Code:
0 Small: Small Font
1 Large: Large Font

G2 (G code G2 clock wise arc)

Runs the G code arc command G2 clock wise arc.

Data field useage:
Data[0] = x constant or register (float) centre point in the normal mode or end point in r mode.
Data[1] = y constant or register (float) centre point in the normal mode or end point in r mode.
Data[2] = i constant or register (float) end point in normal mode or r the radius in the r mode.
Data[3] = j constant or register (float) end point in normal mode or z the height change in the r mode

G3 (G code G3 counter clock wise arc)

Runs the G code arc command G2 counter clock wise arc.

Data field useage:
Data[0] = X constant or register (float) centre point in the normal mode or end point in r mode.
Data[1] = y constant or register (float) centre point in the normal mode or end point in r mode.
Data[2] = i constant or register (float) end point in normal mode or r the radius in the r mode.
Data[3] = j constant or register (float) end point in normal mode or z the height change in the r mode.

GCODE (load and run a G code file)

MAP GCODE command executes the g-code script menu. Below are some of the g codes supported.

F (Feed rate parameter)

Creates a Feed Rate command in map and passes the number returns the length of the number to increment the buffer pointer by note the feed rate remains the same has no effect.

G0 (Rapid Move)

Sets the system in the G0 Rapid move mode this is a persistent and remains until another command is enabled.

G1 (Feed Move)

Sets the system in the G1 move mode this is a persistent and remains until another command is enabled.

G2 (Clockwise Arc)

Sets the system in the G2 clock wise arc mode this is a persistent and remains until another command is enabled.

G3 (Counter Clockwise Arc)

Sets the system in the G3 counter clockwise arc mode this is a persistent and remains until another command is enabled

G4 (Dwell)

G4 is a dwell command implemented immediately and non persistent G code command such as G4 dwell have a delay in seconds after the command such as G4 P0.5 for 0.5 seconds this function searches for the P and then the delay value on success waits and returns the number of chars to advance the buffer pointer if the P is not found or the delay is negative returns is a fail.

G7 (diameter mode)

Program G7 to enter the diameter mode for axis X on a lathe. When in the diameter mode the X axis moves on a lathe will be 1/2 the distance to the center of the lathe. For example X1 would move the cutter to 0.500” from the center of the lathe thus giving a 1” diameter part.

G8 (Radius Mode)

Program G8 to enter the radius mode for axis X on a lathe. When in Radius mode the X axis moves on a lathe will be the distance from the center. Thus a cut at X1 would result in a part that is 2 in diameter. G8 is default at power up.

G17 (XY Planes)

Sets the system to XY arc planes.

G18 (XZ Planes)

Sets the system to XZ arc planes .

G19 (YZ Planes)

Sets the system to YZ arc planes.

G20 (use imperial)

Program G20 use inches for measurement.

G21 (uses Metric)

Program G20 use MM for measurement.

G28 (park)

Function G28 causes the system to park at the location set in the G Code menu then calls the map PARK command to park the axis and stop the machine to allow operator access to the work piece

G40 (Radius compensation off)

Program G40 turns off the radius compensation (radius compensation is not active currently)

G49 (tool length compensation cancel)

Tool length offset compensation cancel.

G50 (scaling off)

Scaling function cancel.

G53(absolute Mode on)

Sets the controller is an absolute mode where the datums are ignored for example if the X datum is 150 and X50 will cause a X position of 150 + 50 After this command X50 will result in a position of 50 because the datums will be ignored for 1 move This command is non Modal and must be called each time an absolute move is required.

G64 (exact path mode)

Exact path mode. G61 visits the programmed point exactly, even though that means temporarily coming to a complete stop.

G64 (performance)

P - motion blending tolerance
Q - naive cam tolerance
G64 - best possible speed.
G64 P- <Q- > blending with tolerance.
G64 - without P means to keep the best speed possible, no matter how far away from the programmed point you end up.
G64 P- Q- - is a way to fine tune your system for best compromise between speed and accuracy. The P- tolerance means that the actual path will be no more than P- away from the programmed endpoint. The velocity will be reduced if needed to maintain the path. In addition, when you activate G64 P- Q- it turns on the naive cam detector; when there are a series of linear XYZ feed moves at the same feed rate that are less than Q- away from being collinear, they are collapsed into a single linear move. On G2/G3 moves in the G17 (XY)

plane when the maximum deviation of an arc from a straight line is less than the G64 P-tolerance the arc is broken into two lines (from start of arc to midpoint, and from midpoint to end). those lines are then subject to the naive cam algorithm for lines. Thus, line-arc, arc-arc, and arc-line cases as well as line-line benefit from the naive cam detector. This improves contouring performance by simplifying the path. It is OK to program for the mode that is already active. See also the Trajectory Control Section for more information on these modes. If Q is not specified then it will have the same behaviour as before and use the value of P-.

G80 (cancel canned cycles)

Cancels canned cycles

G81 (cycle drilling command)

G81 cycle drilling command Data X,Y,Z,R,L where X = the Absolute x position or the X increment in the relative mode Y = the Absolute Y position or the Y increment in the relative mode Z = the Absolute Z position R = the Absolute R position this is the point at which the Z axis changes from rapid move to feed move L is the number of loops or repetitions works only in the relative mode
In the absolute mode works as follows: 1) does a rapid move to position X,Y 2) does a rapid move to position R 3) does a feed move to Z
In the relative mode works as follows: 1) does a rapid move to relative position X,Y (eg current and X and Y position + X and Y respectively) 2) does a rapid move to position R 3) does a feed move to Z 4) if L is greater than 1 decrements L and repeats lines 1 to 4

G82 (cycle drilling command)

G82 cycle drilling command Data X,Y,Z,R,L where X = the Absolute x position or the X increment in the relative mode Y = the Absolute Y position or the Y increment in the relative mode Z = the Absolute Z position R = the Absolute R position this is the point at which the Z axis changes from rapid move to feed move L = the number of loops or repetitions works only in the relative mode P = the dwell time after the drill has reached final position in seconds.
In the absolute mode works as follows: 1) does a rapid move to position X,Y 2) does a rapid move to position R 3) does a feed move to Z
In the relative mode works as follows: 1) does a rapid move to relative position X,Y (eg current and X and Y position + X and Y respectively) 2) does a rapid move to position R 3) does a feed move to Z 4) if L is greater than 1 decrements L and repeats lines 1 to 4

G90 (absolute measurement mode)

Absolute distance mode In absolute distance mode, axis numbers (X, Y, Z, A, B, C, U, V, W) usually represent positions in terms of the currently active coordinate system. Any exceptions to that rule are described explicitly

G90_1 (absolute distance mode)

Absolute distance mode for I, J & K offsets. When G90.1 is in effect I and J both must be specified with G2/3 for the XY plane or J and K for the XZ plane or it is an err.

G91 (incremental distance mode)

Incremental distance mode In incremental distance mode, axis numbers usually represent increments

from the current coordinate buffer

G91_1

Incremental distance mode for I, J & K offsets. G91.1 Returns I, J & K to their default behaviour.

G96 (Constant surface speed)

Sets the constant surface speed mode. G96 D- S- - selects constant surface speed of S feet per minute (if G20 is in effect) or millimetres per minute (if G21 is in effect). D- is optional. When using G96, ensure that X0 in the current coordinate system (including offsets and tool lengths) is the center of rotation or will not give the desired spindle speed. G96 is not affected by radius or diameter mode.

G97 (spindle operational mode)

Spindle RPM mode.

I (x offset)

I DIMENSION USED AS X OFFSET IN ARCS AND IN CANNED CYCLES I immediate command.

J (Y offset)

J DIMENSION USED AS Y OFFSET IN ARCS AND IN CANNED CYCLES this command is an immediate.

M0 (program pause)

M0 Program Pause causes a map press_run command to be written .

M2 (end of program)

End Program M2 or M30.

M3 (spindle clock wise start)

M3 turn spindle clock wise.

M4 (spindle counter clock wise start)

M4 Turn spindle counter clock wise.

M5 (stop spindle)

M5 Stop spindle.

M6 (change tool)

Creates a tool command in map and passes the number to the tool changer or invokes a map tool program.

M7 (Mist coolant on)

M8 (Turn flood coolant on)

M9 (Turn all coolant off)

M60 (Pallet Change Pause)

N (line numbers)

Line (block) numbers: Optional, so often omitted. Necessary for certain tasks, such as M99 P address (to tell the control which block of the program to return to if not the default one) or GoTo statements (if the control supports those). N numbering need not increment by 1 (for example, it can increment by 10, 20, or 1000) and can be used on every block or only in certain spots throughout a program. System parameter number: G10 allows changing of system parameters under program control.

P (dwell or number of turns parameter)

P reads the P parameter used for the number of turns in the G2/3 command or G4 dwell.

R (normally used for radius)

R DIMENSION Radius arc immediate command.

S (spindle speed)

Reads the S parameter used spindle speed mainly in the G97.

T (Tool number)

T reads the T parameter used in tool or analogue commands.

X (X position parameter)

X DIMENSION this command is an immediate command used to set the var x.

Y (Y position parameter)

Y DIMENSION this command is an immediate command used to set the var y.

Z (Z position parameter)

Z DIMENSION this command is an immediate command used to set the var Z.

HALT TIMER (clears a timer channel)

Map HALT TIMER command run time used to clear a timer channel that was previously allocated to a register. Has no effect during simulation.

Data field useage:
Data[0] = timer channel 1 to 8 where the data is constant long.

HELIX (cause helical motion)

Map helix command command G2 clock wise arc or G3 counter clock wise arc where the distance prescribed in the height field is divided by the angle such that the z and angular motion are interpolated.

Data field useage:
Data[0] = x constant or register (float) relative X centre point.
Data[1] = y constant or register (float) relative Y centre point.
Data[2] = Angle constant or register (float) angle of rotation.
Data[3] = Height constant or register (float) the relative height change.

HOME (make the axis move home)

Map HOME command used to cause the system to force the system to home or re-home the axis will move towards real zero position of the axis in the order specified in the home order menu The is no data associated with this command. MAP will auto home if a home command is not discovered in the data base.

IF (if compare with absolute jump)

MAP IF command is used to undertake a comparison and then jump to a line number specified by DATA[2] if the equation is not true.

Data field useage:
Data[0] the base register to test data is a constant long register number
Data[1] is a constant data value or register number to test against the base register
Data[2] is the number of to line to jump to if the equation is not true

Condition code gives the test options:

0	==	: Equal.
1	!=	: Not Equal.
2	>	: Greater than.
3	>=	: Greater or Equal.
4	<	: Less than.
5	<=	: Less or Equal.
6	BitL	: Register Bit = 0.
7	BitH	: Register Bit = 1.
8	PortL	: Input Bit = 0.
9	PortH	: Input Bit = 1.
10	FALSE	: Always Jump FALSE.
11	ONKEY	: OnKey Function activated branch if a function was called.
12	IpAct	: Input Bit Active (if an input is active eg if PNP and input is logic high or NPN and input is logic low) branch on false.
13	InMot	: In motion, an axis is moving.

IMPORT (import data from the serial port)

MAP IMPORT command used to import data via the serial port to load all the registers in the range data[0] to data[1] note data[0] must contain a lower value than data[1] normally the recipient is another controller or controllers for data sharing.

Data field useage:
Data[0] = The constant register number of the first register in the range to be imported.
Data[1] = The constant register number of the last register in the range to be imported.

INDEX (move in relative mode)

MAP INDEX is a relative MOVE command run time function sets the relative position mode and calls move Data[0] = X position constant or register (float).

Data field useage:
Data[1] = Y position or register (float)
Data[2] = Z position or register (float)
Data[3] = axis 3 position or register (float)

INPUT (test a port bit and loop until true)

Map INPUT command run time function Allows the user to monitor an input when the input condition is true the program will continue when false the command will remain in a loop does nothing during simulation.

Data field useage:
Data[0] = the number of the bit to test where the data is constant long
Data[1] = the value of the state where 0 = test for a zero value on the port pin and 1 = test for a 1 on the port pin.

JOG (Call the Jog and redraw the canvas)

This command is used to call the jog menu. After which it will draw the canvas again restoring the screen. When used with OSCILLATING AXIS a MOVE 0, 0, 0, 0 command should be used first this will call home ensuring proper setup of the motion axis The data fields are not used by the command itself and therefore are available for a comment. The message is displayed only in the edit mode.

JogAxis (a single axis jog move)

A special instruction that allows a single axis to be run is a JOG mode. This allows the destination or speed to be changed at any time. The function can be used on absolute or relative axis. If the command is run an an axis set up as a relative axis then the STOP AXIS command can be used to reset the position counter and stop motion. Using STOP AXIS with a absolute command will not reset the position counters. uses the current feed rate or speed and moves to positions specified in the 2 data fields on completion continues to next instruction.

Data field useage:
Data[0] = Axis number 0 to 3/
Data[1] = Position or register (float).

Jog Vel (set jog velocity)

Jog Speed command this allows the speed to be set for each axis sets the speed up to 100% for each axis in percent of maximum speed. Works with JogAxis

Data field useage:
Data[0] = X speed constant or register (float)
Data[1] = Y speed or register (float)
Data[2] = Z speed or register (float)
Data[3] = axis 3 speed or register (float)

JPEG (draw a JPEG image)

MAP JPEG Image command draws an image on the canvas starting at draw point X,Y set with the DRAW POS command The data fields are joined together to form the stored file name on the SMC this should be no more than 12 characters and on the PMC4 16
Data string file name from the file manager .

JPEG_SIZE (set the drawing space for JPEG drawing)

MAP Jpeg Size sets the size of a JPEG image in the X and Y planes in percentage of the canvas area.

Data field useage:
Data[0] = X plane size maximum value 100% constant float or register.
Data[1] = Y plane size maximum value 100% constant float or register.

JUMP (test a port input and jump absolute if true)

Map JUMP command run time function used to test a bit and jump to a line if the condition is TRUE if the bit number is greater than the maximum of 15 for the SMC and 19 for the PMC 4 always jumps function does take the jump during simulation.

Data field useage:
Data[0] = the bit number of the port to test constant long.
Data[1] = the line number to jump to constant long.

Condition code:
0 = JUMP if LOW
1= JUMP if HIGH

KEYPRESS (wait for the user to press a key on the keypad)

Map PRSRUN command run time function Causes the process to wait for the user to press a key pressing the ESC or F8 causes the process to terminate otherwise any key press causes the command to stop looping permitting MAP to continue. The data fields are not used by the command itself and therefore are available for a comment. The message is displayed only in the edit mode does nothing during simulation or on the screen.

LOG TASK (start a data logger task)

MAP LOG TASK command data logger function it is designed to start the logging task and is called from MAP Therefore it has to allocate memory for the results and Start the timer.

Data field useage:
Data[0] is the number of samples in unsigned long.
Data[1] is a constant float or register number representing the Sample period in seconds float minimum is 0.01 seconds.

LD_PAR_FILE (load a parameter file)

MAP LD_PAR_FILE command reads a parameter file and loads it into a buffer ready to be parsed. The file can be loaded for either the serial port or the file directory if the file directory is loaded the file wizard will be called with the extension c:*.dat There are no data fields.

Condition codes:

```
0     RS232 : Load from Serial Port.
1     Disc  : load from File.
```

LOAD Svars (load system vars. calibration data etc. into a register)

MAP Load SVars command allows the user access to system variables such as calibration data the command uses the condition codes to allow access to different groups of system data there are two data parameters used by the command the number of the results register and the index. The index is is offset to the data for example to get the calibration length of the Z axis use an index of 2.

Data field useage:
Data[0] = the number of the results register data is constant long.
Data[1] = the index to the data being sought data is constant long.

Condition codes:

0	C_L	: Calibration length data this is the length of each axis as entered by the installer.
1	T_H	: load register a tool height from tools offset menu.
2	T_D	: load register a tool diameter height from tools offset menu.
3	TCX	: Tool changer position X (index is used to select tool position).
4	TCY	: Tool changer position Y (index is used to select tool position).
5	TCZ	: Tool changer position Z (index is used to select tool position).
6	VEL	: Calibration speed setting.
7	Cnt	: The data logger samples counter.
8	Org	: load the origins.
9	Mptr	: last allocated memory pointer.
10	DIS	: Axis displacement (this is added to origin to cause a displacement from 0).
11	Aseq	: Axis sequence flags.

LED (output to the Keypad light emitting Diodes)

Map LED command outputs the data in Data[0] to the Keypad LED register where 0 turns all LEDS off and 0xFF all on. The power LED is bit[0] and KEY is bit[7].

Data field useage:
Data[0] = constant integer or register number to uses an output from a register

LOAD (load a register with a constant or variable data)

Map VAR_LOAD command used to load a value into a register from a selection of sources where data is loaded into the register specified by data[0]
Data[0] = Constant register number of the register to be loaded (destination register).
Data[1] = is either the value to load if CC = VorR or an index for other condition code values this may be a constant float or register.

Data field useage:
Data[2] = Constant register number of the second register to be loaded (destination register).
Data[3] = is either the value to load if CC = VorR or an index for other condition code values this may be a constant float or register.

Condition codes:

0	VorR	: load data. Loads a constant or the contents of a register.
1	Apos	: load actual position in MM for the indexed axis.
2	Encr	: loads the encoder position register data for the indexed channel.
3	Ferr	: Load the sign adjusted positive following error from the indexed axis.
4	Torq	: Torque output this is the motion controller demand signal for the indexed axis.
5	Port	: Load input port. Loads the contents of the input port into the destination register. Data[1] has no effect .
6	IP.b	: Load input port bit, where data[1]= is the number of the bit to test.
7	Ikey	: Load the last key value from the keypad. if the Key is ESC or F8 the program aborts. Data[1] is not used and has no effect
8	Jpos	: Load the jog position register to the destination register. Data[1] specifies which register SMC values are 0-1 PMC4 0-2
9	T->R	: Load the current tool number into the destination register. Data[1] is unused.
10	(R)R	: Load register indirect, the contents of a register pointed to by the value contained in another register are loaded into the destination register. Data[1] is ignored.
11	RPOS	: Actual position relative to the origin example if the absolute position is 120 and the origin is 100 this will return 20.
12	AOPC	: Axis operation count: every time an axis completes a movement the counter is incremented mainly used with valved axis.
13	Dkey	: The debounced key the command automatically eliminates multiple key presses etc.

LD_COUNTER (load hardware counter)

Map LD_COUNTER command is used to load a value into the Jog wheel registers or the Encoder registers where the value loaded is long signed data for SMC there are two channels 0 to 1 and for the PMC4 there are 3 channels 0 to 2 Note if an axis is in use unless it is a relative axis the command will cause an error.

Data field useage:
Data[0] = the constant long data for the channel.
Data[1] = The data to load into the jog encoder register.

Condition Code:
0 = Jog encoders (auxiliary encoders)
1 = Encoders (main servo or stepper encoders).

LD_MULTIPL (load multiple registers)

MAP LD_MULTIPL (LOAD multiple) command is designed to allow the user to load the same value into a range of registers Typical uses is in the start up code before a MAINSECTION the start register must have a lower value than the end register or the command will fail.

Data field useage:
Data[0] = Start Register, data a constant register number representing the start register in the range.
Data[1] = END Register, data a constant register number representing the end register in the range.
Data[2] = Data Register, is the data to be loaded where data is a constant float or register number.

LOGIC_OP (logic operations)

Map LOGIC_OP logic operations command used to facilitate logic operations on the register specified by constant register number Data[0] where the value of Data[1] as a constant long or register is logically operated on the data register example register[0] contains 2 data[0]=0 data[1]=2 shift left the result in register 0 post command will be 8 logic operations only work on integer registers.

Data field useage:
Data[0] = register number of the register to perform logical operation on.
Data[1] = constant integer value or register number.

Condition code:
0 = Rotate right LOGIC_ROTR:
1 = Rotate left LOGIC_ROTL:
2 = logic OR LOGIC_OR:
3 = logic AND LOGIC_AND:
4 = logic XOR LOGIC_XOR:
5 = logic NOT LOGIC_NOT:

MAINSECTION (marks the end of the initialize section of the program)

MAP MAINSECTION Command this command marks the lines that proceed it as run once which means that on start up the lines preceding MAINSECTION are run this is normally use for code that initializes the process such as loading registers and outputs to ensure pneumatic actuators are homed etc. The data fields are not used by the command itself and therefore are available for a 16 character comment. The message is displayed only in the edit mode, does nothing during simulation or on the screen during run.

Condition codes:
0 0->1 : Run from start.
1 !Home : Run if not Homed.

This allows the code before MAINSECTION to be run either every time on start up or only if the axis have not been homed.

MATH (mathematical operations)

MAP MATH command Data[0] is the results register. The contents of this register will have the mathematical operation performed on using the data or contents of the register specified by Data[1] or if Data[2] is not zero then the result will equal the Data[1] math operand Data[2] e.g. A = B (operand) B There are various options which are set using the condition codes via a wizard.

Data field useage:
Data[0] is the results register.
Data[1] Primary operator value.
Data[2] Seconadry operator value.

Condition Codes:
0 = MATH_ADD: The contents of the destination register has the operator Data[1] added to it.
1 = MATH_SUB: The contents of the destination register has the operator Data[1] subtracted from it.
2 = MATH_DIV: The contents of the destination register is divided by the operator Data[1].
3 = MATH_MUL: The contents of the destination register is multiplied by the operator Data[1].
4 = MATH_NOT: The contents of the destination register is subtracted from zero inverting the value. Data[1] is not used.

Memory allocate (allocate a block of memory)

The Map Memory allocate command is used to allocate memory it works in the same way as New in C++ or Malloc in C. The function calls the OS command Malloc_ddr which allocate blocks of memory from the DDR heap. operates in the following manner:

Data field useage:
Data[0] = The register which will be set as a pointer if.
Data[1] = The size of the memory to be allocated.

Data is allocated in blocks of 65536 bytes so size of 128 allocates 65536 bytes and 100,000 allocates 131072 bytes if memory is able to be allocated the register will be returned as a pointer with a value greater then 0 if the allocation failed the register will be returned with a contents of 0 note it is a good idea to preserve the contents of this register to deallocate the memory

Memory to Register (Mem to Reg)

Map copy to command is used to copy a block of data from the allocated memory block to registers operates in the following manner.

Data field useage:
Data[0] = The register that holds the start register for the transfer of data.
Data[1] = The register that holds the end register for the transfer of data.
Data[2] = The register that holds the pointer to the address of the block of data.
Data[3] = The register or index value reference the pointer that holds the start register for the transfer of data.

The value of the contents of the index register is size adjusted (multiplied by 4) and added to the value of the pointer register If the index is a register it's value is incremented by the number of registers processed e.g. Example: Start register = Reg1 End register = Reg10 Pointer = Reg0 Index = Reg11 with a value of 0 after the command is run reg11 will contain 10 and register 1 to 10 will hold the respective values of the memory contents of the allocated memory.

Modsvar modify system variables

MAP modify system variables command allows the user modify to system variables such as origin or axis displacement data the command uses the condition codes to allow access to different groups of system data there are two data parameters used by the command the number of the results register and the index. The index is is offset to the data for example to get the calibration length of the Z axis use an index of 2 The command allows two variables to be modified concurrently but both have a common condition code so address the same variable array and work in the same way but if the index in data[3] is the same or lower then data[1] the second data set has no effect EG modsvar 0,0,0,1 ORG zeros origins for X [0] and Y [1].

Data field useage:
Data[0] = the number of the data register data or a constant value.
Data[1] = the index to the data being sought data is constant long.
Data[2] = the number of the data register data or a constant value for the second parameter.
Data[3] = the index to the data being sought data is constant long for the second parameter.

Condition codes:
0 = ORIGIN: write to an origin where data[0] is written to the data[1] element of origin array
1 = ORIGIN: add to an origin where data[0] is added to the data[1] element of origin array
2 = Axis displacement: writes data[0] to the data[1] element of displacement array (this is added with origin to cause a displacement from 0)
3 = Axis displacement: Adds data[0] to the data[1] element of displacement array (this is added with origin to cause a displacement from 0)

MOVE (linear interpolated move)

MOVE instruction run time function uses the current feed rate or speed and moves to positions specified in the 4 data fields on completion continues to next instruction.

Data field useage:
Data[0] = X position constant or register (float).
Data[1] = Y position or register (float).
Data[2] = Z position or register (float).
Data[3] = axis 3 position or register (float) (PMC4 only)

MESSAGE (print a message in the message box)

MAP MESSAGE command run time function does nothing during simulation in normal use an editable message saved in the environment will be displayed in the canvas message box

Data field useage:
Data[0] = message number data field is a constant long or variable
The condition code can be set to normal or erase message

Nest (step and repeat command)

MAP Nest command allows the user to replicate a part in a jig, fixture or cut from a sheet. The programmer can select the number in the of repetition is the X and Y planes. There is also a displacement for the X and Y dimensions. If there is a loop counter value greater than 0 the command will decrement the counter on each pass. While the counter is greater then zero the command will behave like an end command and control will not be passed back to the parent process. The command can be used with zero offsets and displacements when it is used as a end command to prevent the control from being passed back to a parent process. note the global X and Y displacement registers are expected to be zero when the command is first encountered. This command works on variable only mainly the AXIS displacement registers it has no other effects and is classified as a registers and data command.

Data field useage:
Data[0] = the number of the data register data or a constant value representing the X row count
Data[1] = the number of the data register data or a constant value representing the X displacement (e.g. the X step)
Data[2] = the number of the data register data or a constant value representing the Y row count
Data[3] = the number of the data register data or a constant value representing the Y displacement (e.g. the Y step)

Condition codes:
0 0–>C : On Zero X,Y count quit.
1 0–>0 : On Zero X,Y continue loop.
2 0->Pk : On Zero X,Y counts Park.
explanation
0 = On completion pass to following line (branch on none zero)
1 = On completion jump to first run line after main section (branch always)
2 = On completion park and on restart jump to first run line after main section (Park on zero count)

OnKey (On function key match call a function) Added on version 1.16.4

Call MAP function if the global key matches the value loaded into condition code Loads a map function from memory and runs it allocates memory and copies the registers to a set reserved for the child process on ending the process these registers will all be copied back to the parent process if the CopyBack command has been called the child process registers will be copied to the parent process as specified by the copyBack command. End is used to return back to the parent process The data fields are joined together to form the file name.

Data field useage:
Data[3:0] = filename.

OPEN FILE (open a binary or text file)

Opens a file. The function opens a file, it can be used in 1 of 4 modes these are:

- Read a named file (the file name is written as a string to Data[0]).
- Write to a named file (the file name is written as a string to Data[0]).
- Reads a file by calling the file browser.
- Write a file for writing by calling the file browser.

All modes make the file status available via the FILE_STATS command which should be called as soon as possible after opening a file to get the file handle (the number of the file that was opened). If the command was a READ, then the system automatically allocates memory in blocks of 64K bytes and a pointer to the block of memory that was allocated. After memory is allocated, the file is then read into the memory and the file is left open.
If the command was a WRITE then memory will need to be allocated to allow data to be written use the MEM_ALLOC command to allocate memory Memory can be allocated at any time prior to writing to the file operates in the following manner.

Data field useage:
Data[0] = string filename from the file manager

Condition codes:
0 = Read a fixed named file.
1 = Write a fixed named file.
2 = Read a file by calling the file browser.
3 = write a file by calling the file browser.

OUTPUT (output data to the outputs)

Map Output command run time function Allows the user to change the contents of the output port does nothing during simulation.

Data field useage:

Data[0] = the port output data which is constant long or register.

ORIGIN (origin axis)

Map ORIGIN command run time function used to set the axis origins for either G code or MAP origin is normally the work piece start point.

Data field useage:
Data[0] = axis0 X origin in dimensioned e.g. Imperial/Metric constant float or register.
Data[1] = axis1 Y origin in dimensioned e.g. Imperial/Metric constant float or register.
Data[2] = axis2 Z origin in dimensioned e.g. Imperial/Metric constant float or register.
Data[3] = axis3 origin in dimensioned e.g. Imperial/Metric constant float or register .

Osc_axis (oscillate valved axis)

MAP Oscillate command is used to control valve driven pneumatic or hydraulic axis. this type of axis uses a switch at each end of the axis when the switch is made and the input goes low valves are reversed and the axis is send in the other direction. The system uses plug reverse e.g. the valves are swapped over concurrently. The values for data[2] and data[3].

Data field useage:
Data[0] = Direction where less than zero is reverse or greater then zero is forward
Data[1] = The number of the axis e.g. 0 to 3
Data[2] = the number of the Forward & reverse switches input in hexadecimal e.g. 0x00010002 = forward switch 1 and reverse is 2
Data[3] = the number of the forward and reverse in hexadecimal e.g. 0x00010002 = forward output 1 and reverse is 2.

Condition codes:
0 = Stop
1 = oscillate
2 = after encountering the reverse switch stop
3 = after encountering the forward switch stop

Parameter (sets machine parameters)

This command is used to set the machine currently this is used to set the dwell time for the DRILL command. The parameter is in seconds and works down to milliseconds EG. 0.001 = 1 ms; The command will persist until another command is encountered.

Condition codes:
None for future use

Park (Park an axis at a safe location and power down)

MAP Park command used to park the machine safely at the predetermined location. On commencement of the command the spindle is turned off by disabling the forward/ reverse outputs and setting the speed to 0. The heads are then moved to a safe area determined by the data fields Data[3] to Data[0]. After parking the axis the outputs are set to one of two modes, these are the default where all outputs are turned off or the preserve outputs mode where the spindle and power enable modes are disabled, but the other outputs remain. When stopped the emergency stop circuit is not monitored permitting the operator to enter the machine guarded area safely to reload the machine. NOTE the positions in the data fields are absolute this means they ignore the origin! because the park position should be a designated safe area on the machine and origins can be set to any location. When the operator is ready to start the machine they press the F7 key and then the start button The controller will enable the drives but will not restart the spindle or turn any outputs that were on back on. It is the responsibility of the programmer to ensure that the heads are safe to move before using the park command. The Z axis will be moved first and then all the other axis will move. In the case where a slotting tool has been used this may be dangerous because a x or Y move is required before the z axis is moved this must be done before the PARK is called.

Data field useage:
Data[0] The absolute park position of axis X where no origin is applied.
Data[1] The absolute park position of axis Y where no origin is applied.
Data[2] The absolute park position of axis Z where no origin is applied this axis is moved first.
Data[3] The absolute park position of axis 3 sometimes marked C where no origin is applied.
If an axis is unused or a parallel axis is set up the data in that field is ignored.

Condition Code:
0 = O=0: Outputs = 0 all outputs are set to zero.
1 = O=R: Outputs are preserved with the exception of spindle and power enable all outputs persist.

Parse (parse a data file)

MAP PARSE_FILE command is used to parses a file that was loaded into memory by Load File the command reads data from the file and copies it into the registers specified by Data[0] to data[2] after parsing the file pointer is set to the next value in the series the system will not read past the end of the file.

Data field useage:
Data[0] The register number of the first register to be written by the parser.
Data[1] The register number of the second register to be written by the parser.
Data[2] The register number of the third register to be written by the parser.

PLANES (set the arc planes)

MAP Planes command used by to the set the arc planes. The data fields are not used by the command itself and therefore are available for a 16 character comment. The message is displayed only in the edit mode, does nothing during simulation or on the screen during run.

Condition code:
0 = XY are the set planes.
1 = XZ are the set planes.
2 = YZ are the set planes.

POWER (raise a number to the power)

Map power command calculates the number raised to the power of a register or value to the register or value example if register 0 contains 2 and register 1 contains 4 the result will yield 16. Data[0] is a constant value representing the number of the result register, computed to a floating point and converted to Data 0 type.

Data field useage:
Data[1] is the operand constant float or register value.
Data[2] is a constant float or value to power to raise the Data[1] to .
the results are summed as a float this the size of the data that can be summed due to float limits.

PowerMode (turns the power on or off)

MAP power mode command is designed to allow the user to turn the power on or off. If a power command is found prior to a motion control command in a MAP program the power will not be turned on and the system will not home unless a motion control command is encountered or a power command is used to turn the power on. Once the power is turned on if the has not been homed it will home after turning the power on the power will not automatically be turned on if a motion control command is encountered and the power is off. The data fields are not used by the command itself and therefore are available for a 16 character comment. The message is displayed only in the edit mode, does nothing during simulation or on the screen during run.

Condition code power modes (with a TRM interface card)

0	POWER OFF	: The power will be turned off and the emergency stop will be active and MAP will exit if the Emergency stop is activated.
1	POWER OFF SAFE	: Power will be turned off and the emergency stop will be ignored and MAP will continue to run if the E Stop is active. The output drivers will be turned off also but their state is stored and can be reinstated by mode 3.
2	POWER ON	: Power is turned on if off and the drivers will be cleared will obey the emergency stops.
3	POWER ON Restore	: Power is turned on if off and the drivers will be restored to the state as per the last output state prior to power off. Map will obey the will obey the emergency stops.

Performance (set the operating mode and system precision)

Map Performance command is used to set the precision or operating mode. Mode o is used to reset all modes to the MAP defaults. Mode 1 is used to set the precision limits which means that the controller will endeavour to reach the destination set in move commands before it continues to the next move. This is useful when the motors are under heavy load and may need to dwell for a time when they reach their command destination. This dwell time allows the integral to grow to force the motor to the exact position required, should the system fail to achieve the command position after data[1] time in seconds the program will halt and report an error. If the condition code is set to 2 the motion controller will commence continuous motion.

Data field useage:
Data[0] = the value of the precision.
Data[1] = time to achieve the precision or quit

Condition codes:
0 = reset precision and continuous mode
1 = Set precision
2 = set continuous mode.

Print register (print the contents of a register on the canvas)

Map print register command the command prints the value of a register on the canvas The system takes care of the register type e.g. floats are printed with a precision such as 123.456 where the precision is set by the precision in the options menu and integer values are printed without a decimal fraction.

Data field useage:
Data[0] = the number of the register to print the value of.
Data[1] = the x position on the canvas in percent as a constant or floating point from a register.
Data[2] = the Y position on the Canvas in percent as a constant or floating point from a register.
Data[3] = The number of digits and the precision used e.g. 2.3 shows 00.123

PWM (output to the Pulsed Width Modulator output)

Map PWM command run time function used to set the controller into the PWM mode where the user can control the frequency of the output and the duty cycle using the condition code the user can select to set output bit as PWM or as an output returning the system to it's default setting. frequency_divider is an 8 bit divider where 0 = clock rate is 320mHZ giving 1250khz and 1 625k minimum frequency is 4882 HZ.

Data field useage:

Data[0] = The duty cycle data where 0 = off and 255 = on 100% 128 = 50% data is constant long or register.

Data[1] = The frequency divider 0 = maximum frequency and 255 = is minimum data is constant long or register.

Question (question box PMC4 version)

Question command run time function loads timer_simulation[8] during simulation loads a question in the message box and waits for the user to enter the data which is loaded with regard the the data type and the maximum and minimum values if the data width is based on the maximum or minimum value e.g. if the maximum value is 1000 four digits may be entered where a negative value is entered for minimum negative values may be entered data entered by the user is tested against the minimum and maximum values and if outside of the range and error message is displayed if there is no error is the data entered the result is stored and the program continues if zero is for minimum and maximum there is no data check and the data width is set to 6 if the result register is floating point the current precision is used to allow the user to enter places of decimals.

Data field useage:
Data[0] = result register constant register number Data (0:255).
Data[1] = message number constant (0:32) or register number.
Data[2] = maximum value constant floating point or or register number.
Data[3] = minimum value constant floating point or or register number.

Condition codes
0 [FP] : Fixed placement message. Displays a user message at a fixed position.
1 [XY] : Canvas draw point Message. Displays a user message at a canvas draw point.
2 ____ : Input only option. Inputs a question under the X Y cursor position.
3 NNN : Print value and input. Prints a value and inputs a question under the X Y cursor.

Rapid Index (rapid Index Relative move at full speed)

MAP Rapid INDEX is a relative MOVE at full speed command where the move is relative to the current axis position for example if the current position is 100, 50, 10 Index 50,0,0 will cause axis 0 (x) to move to 150 no other axis will move The function actually works by setting the relative axis flag and then calls the MOVE function .

Data field useage:
Data[0] = X position constant or register (float)
Data[1] = Y position or register (float)
Data[2] = Z position or register (float)
Data[3] = axis 3 position or register (float)

Rapid move (rapid move causes the axis to move at full speed)

Rapid MOVE instruction run time function sets the speed to 100% and moves on completion returns the speed to original setting calculate new command positions modified written Data[0] = X position constant or register (float).

Data field useage:
Data[1] = Y position or register (float)
Data[2] = Z position or register (float)
Data[3] = axis 3 position or register (float)

Read (read a register file)

MAP READ_FILE used to read a file and write the contents to the registers the file name is a:output##.mdf where the ## are replaced with a number from 0 to 10
Data[0] = the number to substitute ## in the file name.

Reg to MEM (copy registers to memory)

Map copy to command is used to copy a block of registers to the allocated memory operates in the following manner.

Data field useage:
Data[0] = The register that holds the start register for the transfer of data.
Data[1] = The register that holds the end register for the transfer of data.
Data[2] = The register that holds the pointer to the address of the block of data.
Data[3] = The register or index value reference the pointer that holds the start register for the transfer of data.

The value of the contents of the index register is size adjusted (multiplied by 4) and added to the value of the pointer register If the index is a register it's value is incremented by the number of registers processed e.g. Example: Start register = Reg1 End register = Reg10 Pointer = Reg0 Index = Reg11 with a value of 0 after the command is run reg11 will contain 10 and memory contents of block 0 to 9 will hold the respective values of the registers 1 to 10 respectively

Repeat (repeat if true command)

Map REPEAT IF command which is similar to the MAP WHILE command except that it can not be nested and it uses branch which are relative jumps The command performs a logic compare and if TRUE will run the lines below the command when the last line is run e.g. the last one before the branch the command will branch back to the compare and the cycle will continue until another instruction or the equation is FALSE. A test False causes a branch outside the loop.

Data field useage:
Data[0] = the base resister and is a constant register number
Data[1] = the compare register and is a constant long or register number to compare
Data[2] = the constant long number of lines to skip

Condition code gives the test options:

0	==	: Equal.
1	!=	: Not Equal.
2	>	: Greater than.
3	>=	: Greater or Equal.
4	<	: Less than.
5	<=	: Less or Equal.
6	BitL	: Register Bit = 0.
7	BitH	: Register Bit = 1.
8	PortL	: Input Bit = 0.
9	PortH	: Input Bit = 1.
10	FALSE	: Always Jump FALSE.
11	ONKEY	: OnKey Function activated branch if a function was called.
12	IpAct	: Input Bit Active (if an input is active eg if PNP and input is logic high or NPN and input is logic low) branch on false.
13	InMot	: In motion, an axis is moving.

REVOLVE (cause an axis to revolve)

Map REVOLVE command run time function Causes an axis to revolve as long as it is set up as relative axis without limits in the calibration menu.

Data field useage:
Data[0] = the axis to revolve data is constant long
Data[1] = the speed to revolve in percentage of maximum speed data is constant float or register

Rewind (rewind a file to the beginning)

MAP REWIND command used to rewind the file pointer to the start of the buffer on files loaded by load file.

REG_TYPE (specify register type)

Map REG_TYPE command where a range of registers are designated as a particular type.

Data field useage:
Data[0] = the start register data is a constant long
Data[1] = the end resister to store the data data is a constant long

Condition code :
0 = Long,
1 = Unsigned long.
2 = float.

Sine (compute a sine)

Map sine command calculates the sine of a register or value in Data[1] and stores the result in Data 0.

Data field useage:
Data[0] is a constant giving the result register computed to a floating point and converted to Data 0 type.
Data[1] is a constant float or register containing the operand value.

SPEED (set the speed of the axis)

Speed command instruction run time function sets the speed up to 100% for each axis in percent of maximum speed Data[0] = X speed constant or register (float).

Data field useage:
Data[1] = Y speed or register (float).
Data[2] = Z speed or register (float).
Data[3] = axis 3 speed or register (float).

Spindle (set up a spindle and operate it)

MAP Spindle command this is designed to start and stop the spindle of the machine the first parameter is the spindle speed The speed of the spindle is determined by the variable S, in either revolutions per minute (G97 mode; default) or surface feet per minute or [surface] meters per minute (G96 mode [CSS] under either G20 or G21). The right-hand rule can be used to determine which direction is clockwise and which direction is counter-clockwise. Spindle ON CW rotation.

Data field useage:
Data[0] = RPM setting from a constant or a register number.
Data[1] = Maximum speed in the CSS mode only from a constant or a register number.

SQUARE ROOT (computes the square root of a register or value)

Map Square root calculates the square root of a register or value in Data[1] and stores the result in Data 0.
Example if Data[1] contains 9 the result of 3 will be stored in register Data 0.

Data field useage:
Data[0] is a constant giving the result register computed to a floating point and converted to Data 0 type.
Data[1] is a constant float or register containing the operand value.

Startlog (start the data logging process)

This function is the MAP data logger function it is designed to start the logging task and is called from MAP Therefore it has to allocate memory for the results and Start the timer

Data field useage:
Data[0] is the number of samples in unsigned long
Data[1] is a constant float or register number representing the Sample period in seconds float minimum is 0.01 seconds.

STOP AXIS (stop a revolving axis)

Map STOP axis used to stop an axis started using revolve command run time function. if this function is called but the axis is not running MAP will terminate. Has no effect during simulation.

Data field useage:
Data[0] = the number of the axis that is revolving data is constant long.

Stoplog (stop the data logging)

This function is the MAP data logger stop function it is designed to stop the logging task and write the result to the flash stick is called from MAP it will if successful it will delete the allocated memory The timer is stopped. The data fields are not used by the command itself and therefore are available for a 16 character comment. The message is displayed only in the edit mode, does nothing during simulation or on the screen during run.

Sum (compute the sum of a range of registers)

Sum a range of registers in the range of a start register that must be the lowest value to the highest value for example: Start [0] end [10] would sum 11 registers starting with 0 and ending with 10.

Data field useage:
Data[0] is a constant value indicating the start of the series of registers.
Data[1] is a constant value indicating the end marker and is included in the summed result.
Data[2] is a constant value indicating the result register.
all the results are summed as a float this the size of the data that can be summed due to float limits.

Table (create and manage a table) Added to version 1.16.4

This command is used to produce a data input table on the canvas. The table will be populated on the basis of rows * columns if the number of members is not divisible by the number of columns then the start will occur towards the right hand side eg. if Cols=4 and members =5 then two rows will be generated.
Data[0] is the number of columns in the X direction the maximum is 16, however this is more than practical.
Data[1] is the number of members and is limited to 16 * 16 again this may be more than practical as the display currently does not scroll. The number of members can be set to a negative value in which case if the number of members is not directly divisible by the number of rows, then the display will show the uneven rows at the bottom of the display padded towards the left, when Number of members is +VE then any incomplete rows will start at the beginning of the table and will be right padded.

Data[2] is the number of the start register for data entry. Registers can be set up as a mixture of floats, longs or unsigned longs as required and the display will indicate to the user if the input is floating point or integer, where the register is a floating point the input box will include a space for the decimal point '.' and the fractional part of the input. Eg. entering 4.2 via the **Tablim** command Data[2] will cause integer inputs to input 4 digits and floating point to input 4 digits and 2 digits of fraction. The next Data is Data[3] this controls the number of lines that are called after the the table command, these lines are designed to be check code or OnKey commands that allow the user to run the data that they have written via the menu. The table command works very much as a while loop and so lines of check code is the number of lines that are run after the table command. In reality setting this value to 0 will only allow the user to enter data but nothing. If any of the OnKey functions are called that destroy the screen then before exiting a child process draw the canvas this will clear the draw flags and the table command will redraw the table on taking back control from the child process. Note do note use a repeat commands in the check code as these may overwrite the table loop commands**.**

Data field useage:
Data[0] = X columns
Data[1] = is the number of members
Data[2] = Start register
Data[3] = lines of limit check code

Tablimit (set table limits) Added to version 1.16.4

This command is used to set the limits for the table command. Data[0] is the maximum value that can be entered in any data entry box. Data[1] is the minimum value that can be entered in any one data box if the data may be entered this value should be negative. Data[2] is the data entry and display number of digits and the precision or example 4.2 give a display of 1000.12. Data[3] is an error register for future use.

Data field useage:
Data[0] = maximum limit.
Data[1] = minimum limit.
Data[2] = precision 3.2 = 3 digits and 2 decimal places.
Data[3] = error register.

TabPars (set the table paramteres) Added to version 1.16.4

This command is used to set the position of the first data input and set the space between the X columns. The Y spacing is the space between Rows.
It is always a good idea to set the the first data position However the spacing is automatically set if the spacing set in Data[2] and Data[3] are set to 0.

Data field useage:
Data[1] = X start in %
Data[1] = Y start in %
Data[2] = x spacing in %
Data[3] = Y spacing in %

Tangent (compute the tangent of a register content or constant)

Map tangent command calculates the tangent of a register or value in Data[1] and stores the result in Data[0].

Data field useage:
Data[0] is a constant giving the result register computed to a floating point and converted to Data 0 type.
Data [1] is a constant float or register containing the operand value.

Text (print text or used as a comment)

This command is used to display a message in the code as a comment or to write a line of text to the screen. Too allow longer messages to be displayed there are two mode these are Text which positions the cursor to the current X and Y locations set using the DRAW POS command. The Prnt mode is designed to allow messages to be made up from multiple commands, this mode send the text to the display without calling the draw pointer.

Data[0] 16 bytes of string message.

Condition codes:

0	Comt:	Comment (puts a commant in the code).
1	Text:	Printable Text (positions the cursor and send the text to the screen).
2	Prnt:	Continue Print (This mode allows printing to continue from the last character draw or current cursor).

TIMER (associate a register to a timer)

Map TIMER command run time used to set a register to a timer channel and to set the initial time the register designated in Data[0] is decremented or incremented on every clock time out which is set at 10ms the condition code is used to set the mode of up or down counter when set as a down counter the count stops at 0 there is no role over check for the up counter has no effect during simulation.

Data field useage:
Data[0] = the number of the register to use 0-255 data is a constant long.
Data[1] = the Timer channel 1 to 8 data is a constant long.
Data[2] = the timer interval in 10ms slots e.g. 100 = 1 second data is a constant long.

Condition code:
0 = down counter.
1 = up counter.

Tool (calls a map tool program or allows manual tool change)

Map TOOL command Loads a map tool function from memory and runs it.

Torque (limit the voltage outputs to limit motor torque)

Map Torque command run time used to set the torque limit in % for the designated axis function has no effect during simulation. Note this command only works if the amplifiers are set in the constant current mode. If the amplifiers or drives are set in any other mode this command may not work. Typically most drives support current mode or velocity mode or constant velocity mode in these modes the velocity of the motor is determined by the voltage output by the controller. It may be useful to limit to limit the output speed but the feed rate or speed command should be used for this requirement.

Data field useage:
Data[0] = the axis to apply the torque limit to in % data is constant long.
Data[1] = the percentage of torque to apply data is constant float or register.

Wait (wait for the time to expire then continue with next command)

Map WAIT command run time function Causes the execution of MAP to pause for a time set by Data[0] in seconds does nothing during simulation.

Data field useage:
Data[0] = time where the minimum time is 0.01 seconds the data is constant float or register.

While (part or a do while loop)

Map WHILE command run time used to do logic compare and jump has two functions when preceded by do if true will jump to the line in the do_vars array else will loop locally until the term is false.

Data field useage:
Data[0] = the base resister and is a constant register number
Data[1] = the compare register and is a constant long or register number to compare.

Condition Codes:

0	==	: Equal.
1	!=	: Not Equal.
2	>	: Greater than.
3	>=	: Greater or Equal.
4	<	: Less than.
5	<=	: Less or Equal.
6	BitL	: Register Bit = 0.
7	BitH	: Register Bit = 1.
8	PortL	: Input Bit = 0.
9	PortH	: Input Bit = 1.
10	FALSE	: Always Jump FALSE.
11	ONKEY	: OnKey Function activated branch if a function was called.
12	IpAct	: Input Bit Active (if an input is active eg if PNP and input is logic high or NPN and input is logic low) branch on false.
13	InMot	: In motion, an axis is moving.

Write Binary File (WrtBinFile)

writes a map binary file written by map from allocated memory. Since map does not know the contents of the memory block the size is set in bytes Note if numeric data has been written the size of the data will need to be multiplied by 4 as 4 is the size of most of the data eg size of a register is 4 bytes.

Data field useage:
Data[0] = The register that holds the file handle to the opened file channel.
Data[1] = The register that holds the pointer to the address of the block of data.
Data[2] = The register or index value reference the pointer that holds the start register for the transfer of data.
Data[3] = The register or that holds the size of the data in BYTES.

WRITE_FILE (used to write the registers to a file)

MAP WRITE_FILE used to write the registers to a file as a binary the file name is a:output##.mdf where the ## are replaced with a number from 0 to 10

Data[0] = the number to substitute ## in the file name.

YESNO (call a yes/no box)

MAP YES_NO command uses a question box to load a message and requests a yes or no answer from the user if the user prompts NO to the question the program branches to the line number specified in DATA[1], on yes the program continues on the next line after the YESNO command.

Data field useage:
Data[0] = the message number data is constant long.
Data[1] = the number of lines to skip on a NO response data is a constant long.

Map command groups (from the pull down menu system)

Motion control	Machine Specific	Inputs	Outputs	Program flow	Math	Timers	Registers & data	Canvas Drawing	Data Recording	Memory allocation	File Handling	Table Functions
Arc	Absloute Mode	ADC analogue in	Analogue out	Begin	ADDMULTIPL	Halt	Copy Back	Draw Canvas	Data Log Task	Allocate Memory	Close File	On Key
Do Move	Air Drill	Import	Change Output	Branch	Average	Timer	Load Register	Clear Canvas	Start Recording	Copy From Memory	Load File Data	Table
Helix	Coolant	Input	Export	Call MAP	Cosine	Wait	Load Counter	Draw Pointer	Stop Recording	Copy To Memory	Load Parameter	Table Limits
Home	Drilling	Key Press	LED	Call Subroutine	Find Max		Load Multiple	Circle		Free Memory	Open Bin File	Table Parameters
Index	Feed Rate	Question	Message	Dec Jump	Find Min		Load System Var	Draw Rectangle			Parse File	
Move	G Code	Read File	Output	Do	Logic		Modify System vars	Set Colors			Rewind File Ptr.	
Origin	Jog (call jog menu)	YesNo Box	Print Reg	END	Math		Nest	Draw Line			Write To Binary File	
Rapid Index	Park & power down		PWM	EXIT	Power		Parameters (Machine)	J Peg Image				
Rapid Move	Planes		Write	IF	Sine		Register Type	J Peg Size				
Revolve	Precision			JUMP	Square Root			Work Size				
Speed	Spindle			Main SECTION	Sum			Font Size				
Stop Axis	Tool			Repeat if true	Tangent							
Torque Limit				While								

Importing map programs written in text

Map now supports a text import facility, this is similar to the G code support the controller has supported for many years. However it is now possible to import programs of up to 1000 lines directly in to the editor. Text programs may support comments which will not be copied into the map imported into the map editor. This facility gives programmers the option of writing their programs off-line and adding considerable commenting. Programs are imported via the option menu in the main edit create screen. To import a file, select "**Import a file**" the controller will display files from the C drive which is the flash stick. The files may have any extension but map will default to file-name.CNC. Once a file has been selected push the ENTER key and the file will be Parsed. Warnings and any errors will be displayed on the screen and should be checked. Note it is possible to shorten command names, but this may result in conflicts for example if you type "load" this will be accepted but in reality you could mean load register or load system the parser will sellect the first. In reality it is better to type the command name in full. However partial matches may be used at your own discretion. It is better to always adopt a programming style and maintain this normally the command will appear on the left-hand side, personally I prefer to use tabs or spaces so that all the data statements appear in the same place an example of this is shown below:

```
SetColors     data[0]0x001F          data[1]0xFFFF
DrawCanv      data[0]0
DrawRect      data[0]0               data[1]0          data[2]100.00       data[3]08.00
DrawPos       data[0]33.00           data[1]0
SetColors     data[0]0xFFFF          data[1]0x0000
Text          str[Save Data Menu]    CC1
```

In the above example the map commands are written on the left-hand side data[0] is the next column data[1] the next column and so on. The condition code (CC) is written on the right-hand side this makes it easier to see. If you look at the command text it does not have a data zero it has string instead (S TR). This means string and the string is embedded between [] in reality this is the same as data[0]. Data[n] is actually a keyword where the number and is equal to the column number therefore on the pmc4 the maximum column number is three. The value immediately following is a numeric value to be entered. This value may be written as a floating point such as 123. 12 and is signed so it may be positive or negative, negative values have a minus sign written in front of them such as -123.12, integer values, that is values that are whole numbers can be signed such as -123 or unsigned such as 123 they can be written as a decimal or hexadecimal number. Hexadecimal numbers are preceded by 0X such as 0XFFFF, value is written maybe uppercase or lowercase. The parser will automatically sort out whether the value should be floating point or integer and correct any errors.

Unlike G code no values persist therefore any values that are not written will be set to 0, for example if we write move data [2] 100, this will make axis three the Z axis move to 100 and the other axis to 0. It is therefore not necessary to write zero values, however it is good practice to write zeros as well where commands can be misinterpreted such command normally involve motion control. There is nothing worse than not being able to remember if the writer intended a value to be zero not. For colours and other non-essential commands there is really no fixed rule.

G code commands can also be included with map commands, it is important to remember that G

code persists so if you write G0 this command will persist until another command written in G code or map is encountered. Map commands do not persist and must be written on each line. Text commands can be written with strings as long as 48 characters and the parser will automatically apply multiple lines to display the message this makes it much easier to write text strings that need to be printed to the screen. If you wish to write a text message longer than 16 characters write text with the message of up to 48 characters followed by a CC1 and the interpreter will automatically populate multiple text messages to facilitate a text you intend to print.

It is wise to add comments to your code in map unused fields may be used for comments of up to 16 characters. It is wise to use these fields to assist programmers in the field that may not have access to your text source files. However the parser supports three different types of comment these are the G code ( ) style the single line ; style and C++// style the latter two operate on only one line and are terminated by a line feed. Note files can be written in either Windows style with a carriage return line feed on each line or in LINUX style with a line feed only on each line. A line is only completed when the parser encounters a line feed (decimal 10). You may include as many comments as you wish to make your code easier to understand. Note that if another language is selected the commands must be written in that language. The controller uses the internal text to select the command and therefore programs written in Spanish must be parsed in Spanish, it is also possible to write a program in one language and translated to another by selecting a different language before exporting the program.

Exporting existing map files

Existing map files can be exported to a text file by selecting the options menu from the edit create screen, select the “export to a file”and press enter, the file save box will appear, type the file name you would like a new file to be called and press enter. When the file box disappears file has been written.

Keywords:

All commands shown on the preceding pages are keywords these may be inappropriate language such as English or Spanish, supported G code shown above in this manual are also keywords the map specific data str and CC are specific keywords to map:

Data[n] Is a numerical data input where n represents the number of the column.

STR[ ] Is a keyword to import a string.

CC n The keyword CC is followed by a number not exceeding 15. CC means condition code and where the value is zero commands which use a condition code will not export a value for CC. The value for CC is sequential and is shown in the commands above.

CODEn Followed by the number (n) can be used inplace of the name of a command. Only for use by the brave!

Map commands keywords listed by text and number

```
End             // 0      Move            // 1
Index           // 2      Speed           // 3
Wait            // 4      KeyPress        // 5
Input           // 6      Output          // 7
ChangeOp        // 8      Message         // 9
Jump            // 10     DecJump         // 11
LoadReg         // 12     Math            // 13
Origin          // 14     Revolve         // 15
StopAxis        // 16     TorqueLim       // 17
AnalogOp        // 18     Timer           // 19
HaltTimer       // 20     If              // 21
Do              // 22     While           // 23
Led             // 24     LdCounter       // 25
PrintReg        // 26     Question        // 27
Import          // 28     Export          // 29
DoMove          // 30     Home            // 31
G_Code          // 32     Begin           // 33
CallSubrt       // 34     BrancFalse      // 35
Arc             // 36     ReadFile        // 37
WriteFile       // 38     ADC             // 39
Average         // 40     RegType         // 41
MainSect        // 42     RapidMove       // 43
AirDrill        // 44     LdMultipl       // 45
DrawCanv        // 46     ClrCanvas       // 47
DrawPos         // 48     DrawRect        // 49
SetColors       // 50     DrawLine        // 51
ParseFile       // 52     Rewind          // 53
LdParFile       // 54     RepeatIf        // 55
FileStats       // 56     Exit            // 57
LogicOp         // 58     JpegImage       // 59
JpegSize        // 60     Coolant         // 61
FeedRate        // 62     Spindle         // 63
PWM             // 64     CallMap         // 65
Tool            // 66     YesNoBox        // 67
Planes          // 68     FindMax         // 69
FindMin         // 70     WorkSize        // 71
LoadSvars       // 72     Park            // 73
CopyBack        // 74     Helix           // 75
G2              // 76     G3              // 77
Sum             // 78     Sine            // 79
Cosine          // 80     Tangent         // 81
SqRoot          // 82     Power           // 83
LogTask         // 84     StartLog        // 85
StopLog         // 86     Fcircle         // 87
Precision       // 88     RapidIndex      // 89
Drill           // 90     MemAlloc        // 91
FreeMem         // 92     MemToReg        // 93
RegToMem        // 94     OpenFile        // 95
WrtBinFile      // 96     CloseFile       // 97
AddMultiple     // 98     PowerMode       // 99
ModSysVars      // 100    Nesting         // 101
OscAxis         // 102    Text            // 103
JogMenu         // 104    AbsMode         // 105
Parameter       // 106    Table           // 107
```

```
TabLimit            // 108          TabPars             // 109
OnKey               // 110          ErrorBox            // 111
JogAxis             // 112          JogVel              // 113
FontSize            // 114
```