# M Codes

Table of Contents

[1. M Code Quick Reference Table](http://linuxcnc.org/docs/html/gcode/m-code.html%22%20%5Cl%20%22cha%3Am-codes)

[2. M0, M1 Program Pause](http://linuxcnc.org/docs/html/gcode/m-code.html%22%20%5Cl%20%22mcode%3Am0-m1)

[3. M2, M30 Program End](http://linuxcnc.org/docs/html/gcode/m-code.html%22%20%5Cl%20%22mcode%3Am2-m30)

[4. M60 Pallet Change Pause](http://linuxcnc.org/docs/html/gcode/m-code.html%22%20%5Cl%20%22mcode%3Am60)

[5. M3, M4, M5 Spindle Control](http://linuxcnc.org/docs/html/gcode/m-code.html%22%20%5Cl%20%22mcode%3Am3-m4-m5)

[6. M6 Tool Change](http://linuxcnc.org/docs/html/gcode/m-code.html%22%20%5Cl%20%22mcode%3Am6)

[6.1. Manual Tool Change](http://linuxcnc.org/docs/html/gcode/m-code.html%22%20%5Cl%20%22_manual_tool_change)

[6.2. Tool Changer](http://linuxcnc.org/docs/html/gcode/m-code.html%22%20%5Cl%20%22_tool_changer)

[7. M7, M8, M9 Coolant Control](http://linuxcnc.org/docs/html/gcode/m-code.html%22%20%5Cl%20%22mcode%3Am7-m8-m9)

[8. M19 Orient Spindle](http://linuxcnc.org/docs/html/gcode/m-code.html%22%20%5Cl%20%22mcode%3Am19)

[9. M48, M49 Speed and Feed Override Control](http://linuxcnc.org/docs/html/gcode/m-code.html%22%20%5Cl%20%22mcode%3Am48-m49)

[10. M50 Feed Override Control](http://linuxcnc.org/docs/html/gcode/m-code.html%22%20%5Cl%20%22mcode%3Am50)

[11. M51 Spindle Speed Override Control](http://linuxcnc.org/docs/html/gcode/m-code.html%22%20%5Cl%20%22mcode%3Am51)

[12. M52 Adaptive Feed Control](http://linuxcnc.org/docs/html/gcode/m-code.html%22%20%5Cl%20%22mcode%3Am52)

[13. M53 Feed Stop Control](http://linuxcnc.org/docs/html/gcode/m-code.html%22%20%5Cl%20%22mcode%3Am53)

[14. M61 Set Current Tool](http://linuxcnc.org/docs/html/gcode/m-code.html%22%20%5Cl%20%22mcode%3Am61)

[15. M62 - M65 Digital Output Control](http://linuxcnc.org/docs/html/gcode/m-code.html%22%20%5Cl%20%22mcode%3Am62-m65)

[16. M66 Wait on Input](http://linuxcnc.org/docs/html/gcode/m-code.html%22%20%5Cl%20%22mcode%3Am66)

[17. M67 Analog Output,Synchronized](http://linuxcnc.org/docs/html/gcode/m-code.html%22%20%5Cl%20%22mcode%3Am67)

[18. M68 Analog Output, Immediate](http://linuxcnc.org/docs/html/gcode/m-code.html%22%20%5Cl%20%22mcode%3Am68)

[19. M70 Save Modal State](http://linuxcnc.org/docs/html/gcode/m-code.html%22%20%5Cl%20%22mcode%3Am70)

[20. M71 Invalidate Stored Modal State](http://linuxcnc.org/docs/html/gcode/m-code.html%22%20%5Cl%20%22mcode%3Am71)

[21. M72 Restore Modal State](http://linuxcnc.org/docs/html/gcode/m-code.html%22%20%5Cl%20%22mcode%3Am72)

[22. M73 Save and Autorestore Modal State](http://linuxcnc.org/docs/html/gcode/m-code.html%22%20%5Cl%20%22mcode%3Am73)

[22.1. Selectively Restoring Modal State](http://linuxcnc.org/docs/html/gcode/m-code.html%22%20%5Cl%20%22_selectively_restoring_modal_state)

[23. M100 - M199 User Defined Commands](http://linuxcnc.org/docs/html/gcode/m-code.html%22%20%5Cl%20%22mcode%3Am100-m199)

## 1. M Code Quick Reference Table

| Code  | Description |
| --- | --- |
| [M0 M1](http://linuxcnc.org/docs/html/gcode/m-code.html%22%20%5Cl%20%22mcode%3Am0-m1) | Program Pause |
| [M2 M30](http://linuxcnc.org/docs/html/gcode/m-code.html%22%20%5Cl%20%22mcode%3Am2-m30) | Program End |
| [M60](http://linuxcnc.org/docs/html/gcode/m-code.html%22%20%5Cl%20%22mcode%3Am60) | Pallet Change Pause |
| [M3 M4 M5](http://linuxcnc.org/docs/html/gcode/m-code.html%22%20%5Cl%20%22mcode%3Am3-m4-m5) | Spindle Control |
| [M6](http://linuxcnc.org/docs/html/gcode/m-code.html%22%20%5Cl%20%22mcode%3Am6) | Tool Change |
| [M7 M8 M9](http://linuxcnc.org/docs/html/gcode/m-code.html%22%20%5Cl%20%22mcode%3Am7-m8-m9) | Coolant Control |
| [M19](http://linuxcnc.org/docs/html/gcode/m-code.html%22%20%5Cl%20%22mcode%3Am19) | Orient Spindle |
| [M48 M49](http://linuxcnc.org/docs/html/gcode/m-code.html%22%20%5Cl%20%22mcode%3Am48-m49) | Feed & Spindle Overrides Enable/Disable |
| [M50](http://linuxcnc.org/docs/html/gcode/m-code.html%22%20%5Cl%20%22mcode%3Am50) | Feed Override Control |
| [M51](http://linuxcnc.org/docs/html/gcode/m-code.html%22%20%5Cl%20%22mcode%3Am51) | Spindle Override Control |
| [M52](http://linuxcnc.org/docs/html/gcode/m-code.html%22%20%5Cl%20%22mcode%3Am52) | Adaptive Feed Control |
| [M53](http://linuxcnc.org/docs/html/gcode/m-code.html%22%20%5Cl%20%22mcode%3Am53) | Feed Stop Control |
| [M61](http://linuxcnc.org/docs/html/gcode/m-code.html%22%20%5Cl%20%22mcode%3Am61) | Set Current Tool Number |
| [m62-m65](http://linuxcnc.org/docs/html/gcode/m-code.html%22%20%5Cl%20%22mcode%3Am62-m65) | Output Control |
| [M66](http://linuxcnc.org/docs/html/gcode/m-code.html%22%20%5Cl%20%22mcode%3Am66) | Input Control |
| [M67](http://linuxcnc.org/docs/html/gcode/m-code.html%22%20%5Cl%20%22mcode%3Am67) | Analog Output Control |
| [M68](http://linuxcnc.org/docs/html/gcode/m-code.html%22%20%5Cl%20%22mcode%3Am68) | Analog Output Control |
| [M70](http://linuxcnc.org/docs/html/gcode/m-code.html%22%20%5Cl%20%22mcode%3Am70) | Save Modal State |
| [M71](http://linuxcnc.org/docs/html/gcode/m-code.html%22%20%5Cl%20%22mcode%3Am71) | Invalidate Stored Modal State |
| [M72](http://linuxcnc.org/docs/html/gcode/m-code.html%22%20%5Cl%20%22mcode%3Am72) | Restore Modal State |
| [M73](http://linuxcnc.org/docs/html/gcode/m-code.html%22%20%5Cl%20%22mcode%3Am73) | Save Autorestore Modal State |
| [M100-M199](http://linuxcnc.org/docs/html/gcode/m-code.html%22%20%5Cl%20%22mcode%3Am100-m199) | User Defined M-Codes |

## 2. M0, M1 Program Pause

* M0 - pause a running program temporarily. LinuxCNC remains in the Auto Mode so MDI and other manual actions are not enabled. Pressing the resume button will restart the program at the following line.
* M1 - pause a running program temporarily if the optional stop switch is on. LinuxCNC remains in the Auto Mode so MDI and other manual actions are not enabled. Pressing the resume button will restart the program at the following line.

|  |  |
| --- | --- |
| Note | It is OK to program M0 and M1 in MDI mode, but the effect will probably not be noticeable, because normal behavior in MDI mode is to stop after each line of input anyway. |

## 3. M2, M30 Program End

* M2 - end the program. Pressing Cycle Start ("R" in the Axis GUI) will restart the program at the beginning of the file.
* M30 - exchange pallet shuttles and end the program. Pressing Cycle Start will start the program at the beginning of the file.

Both of these commands have the following effects:

1. Change from Auto mode to MDI mode.
2. Origin offsets are set to the default (like G54).
3. Selected plane is set to XY plane (like G17).
4. Distance mode is set to absolute mode (like G90).
5. Feed rate mode is set to units per minute (like G94).
6. Feed and speed overrides are set to ON (like M48).
7. Cutter compensation is turned off (like G40).
8. The spindle is stopped (like M5).
9. The current motion mode is set to feed (like G1).
10. Coolant is turned off (like M9).

|  |  |
| --- | --- |
| Note | Lines of code after M2/M30 will not be executed. Pressing Cycle Start will start the program at the beginning of the file. |

|  |  |
| --- | --- |
| Warning | Using % to wrap the G code does not do the same thing as a Program End. See [File Requirements](http://linuxcnc.org/docs/html/gcode/overview.html%22%20%5Cl%20%22gcode%3Afile-requirements) for more information on what using % does not do. |

## 4. M60 Pallet Change Pause

* M60 - exchange pallet shuttles and then pause a running program temporarily (regardless of the setting of the optional stop switch). Pressing the cycle start button will restart the program at the following line.

## 5. M3, M4, M5 Spindle Control

* M3 - start the spindle clockwise at the S speed.
* M4 - start the spindle counterclockwise at the S speed.
* M5 - stop the spindle.

It is OK to use M3 or M4 if the [S](http://linuxcnc.org/docs/html/gcode/other-code.html%22%20%5Cl%20%22sec%3Aset-spindle-speed) spindle speed is set to zero. If this is done (or if the speed override switch is enabled and set to zero), the spindle will not start turning. If, later, the spindle speed is set above zero (or the override switch is turned up), the spindle will start turning. It is OK to use M3 or M4 when the spindle is already turning or to use M5 when the spindle is already stopped.

## 6. M6 Tool Change

### 6.1. Manual Tool Change

If the HAL component hal\_manualtoolchange is loaded, M6 will stop the spindle and prompt the user to change the tool based on the last T- number programmed. For more information on hal\_manualtoolchange see the [Manual Tool Change](http://linuxcnc.org/docs/html/gui/axis.html%22%20%5Cl%20%22sec%3Amanual-tool-change) section.

### 6.2. Tool Changer

To change a tool in the spindle from the tool currently in the spindle to the tool most recently selected (using a T word - see Section [Select Tool](http://linuxcnc.org/docs/html/gcode/other-code.html%22%20%5Cl%20%22sec%3Aselect-tool)), program M6. When the tool change is complete:

* The spindle will be stopped.
* The tool that was selected (by a T word on the same line or on any line after the previous tool change) will be in the spindle.
* If the selected tool was not in the spindle before the tool change, the tool that was in the spindle (if there was one) will be placed back into the tool changer magazine.
* If configured in the .ini file some axis positions may move when a M6 is issued. See the [EMCIO section](http://linuxcnc.org/docs/html/config/ini-config.html%22%20%5Cl%20%22sec%3Aemcio-section) for more information on tool change options.
* No other changes will be made. For example, coolant will continue to flow during the tool change unless it has been turned off by an M9.

|  |  |
| --- | --- |
| Warning | The tool length offset is not changed by M6, use G43 after the M6 to change the tool length offset. |

The tool change may include axis motion. It is OK (but not useful) to program a change to the tool already in the spindle. It is OK if there is no tool in the selected slot; in that case, the spindle will be empty after the tool change. If slot zero was last selected, there will definitely be no tool in the spindle after a tool change. The tool changer will have to be setup to perform the tool change in hal and possibly classicladder.

## 7. M7, M8, M9 Coolant Control

* M7 - turn mist coolant on. M7 controls iocontrol.0.coolant-mist pin.
* M8 - turn flood coolant on. M8 controls iocontrol.0.coolant-flood pin.
* M9 - turn both M7 and M8 off.

Connect one or both of the coolant control pins in HAL before M7 or M8 will control an output. M7 and M8 can be used to turn on any output via G code.

It is OK to use any of these commands, regardless of the current coolant state.

## 8. M19 Orient Spindle

* M19 R- Q- [P-]
* R Position to rotate to from 0, valid range is 0-360 degrees
* Q Number of seconds to wait until orient completes. If motion.spindle.is\_oriented does not become true within Q timeout an error occurs.
* P Direction to rotate to position.
	+ 0 rotate for smallest angular movement (default)
	+ 1 always rotate clockwise (same as M3 direction)
	+ 2 always rotate counterclockwise (same as M4 direction)

M19 is cleared by any of M3,M4,M5.

Spindle orientation requires a quadrature encoder with an index to sense the spindle shaft position and direction of rotation.

INI Settings in the [RS274NGC] section.

ORIENT\_OFFSET = 0-360 (fixed offset in degrees added to M19 R word)

HAL Pins

* motion.spindle-orient-angle (out float) Desired spindle orientation for M19. Value of the M19 R word parameter plus the value of the [RS274NGC]ORIENT\_OFFSET ini parameter.
* motion.spindle-orient-mode (out s32) Desired spindle rotation mode. Reflects M19 P parameter word, Default = 0
* motion.spindle-orient (out bit) Indicates start of spindle orient cycle. Set by M19. Cleared by any of M3,M4,M5. If spindle-orient-fault is not zero during spindle-orient true, the M19 command fails with an error message.
* motion.spindle-is-oriented (in bit) Acknowledge pin for spindle-orient. Completes orient cycle. If spindle-orient was true when spindle-is-oriented was asserted, the spindle-orient pin is cleared and the spindle-locked pin is asserted. Also, the spindle-brake pin is asserted.
* motion.spindle-orient-fault (in s32) Fault code input for orient cycle. Any value other than zero will cause the orient cycle to abort.
* motion.spindle-locked (out bit) Spindle orient complete pin. Cleared by any of M3,M4,M5.

## 9. M48, M49 Speed and Feed Override Control

* M48 - enable the spindle speed and feed rate override controls.
* M49 - disable both controls.

It is OK to enable or disable the controls when they are already enabled or disabled. See the [Feed Rate](http://linuxcnc.org/docs/html/gcode/machining-center.html%22%20%5Cl%20%22sub%3Afeed-rate) Section for more details.

## 10. M50 Feed Override Control

* M50 <P1> - enable the feed rate override control. The P1 is optional.
* M50 P0 - disable the feed rate control.

While disabled the feed override will have no influence, and the motion will be executed at programmed feed rate. (unless there is an adaptive feed rate override active).

## 11. M51 Spindle Speed Override Control

* M51 <P1> - enable the spindle speed override control. The P1 is optional.
* M51 P0 - disable the spindle speed override control program. While disabled the spindle speed override will have no influence, and the spindle speed will have the exact program specified value of the S-word (described in [Spindle Speed](http://linuxcnc.org/docs/html/gcode/other-code.html%22%20%5Cl%20%22sec%3Aset-spindle-speed) Section).

## 12. M52 Adaptive Feed Control

* M52 <P1> - use an adaptive feed. The P1 is optional.
* M52 P0 - stop using adaptive feed.

When adaptive feed is enabled, some external input value is used together with the user interface feed override value and the commanded feed rate to set the actual feed rate. In LinuxCNC, the HAL pin motion.adaptive-feed is used for this purpose. Values on motion.adaptive-feed should range from 0 (feed hold) to 1 (full speed).

## 13. M53 Feed Stop Control

* M53 <P1> - enable the feed stop switch. The P1 is optional. Enabling the feed stop switch will allow motion to be interrupted by means of the feed stop control. In LinuxCNC, the HAL pin motion.feed-hold is used for this purpose. A true value will cause the motion to stop when M53 is active.
* M53 P0 - disable the feed stop switch. The state of motion.feed-hold will have no effect on feed when M53 is not active.

## 14. M61 Set Current Tool

* M61 Q- - change the current tool number while in MDI or Manual mode. One use is when you power up LinuxCNC with a tool currently in the spindle you can set that tool number without doing a tool change.

It is an error if:

* Q- is not 0 or greater

## 15. M62 - M65 Digital Output Control

* M62 P- - turn on digital output synchronized with motion. The P- word specifies the digital output number.
* M63 P- - turn off digital output synchronized with motion. The P- word specifies the digital output number.
* M64 P- - turn on digital output immediately. The P- word specifies the digital output number.
* M65 P- - turn off digital output immediately. The P- word specifies the digital output number.

The P-word ranges from 0 to a default value of 3. If needed the the number of I/O can be increased by using the num\_dio parameter when loading the motion controller. See the [Motion Section](http://linuxcnc.org/docs/html/config/core-components.html%22%20%5Cl%20%22sec%3Amotion) for more information.

The M62 & M63 commands will be queued. Subsequent commands referring to the same output number will overwrite the older settings. More than one output change can be specified by issuing more than one M62/M63 command.

The actual change of the specified outputs will happen at the beginning of the next motion command. If there is no subsequent motion command, the queued output changes won’t happen. It’s best to always program a motion G code (G0, G1, etc) right after the M62/63.

M64 & M65 happen immediately as they are received by the motion controller. They are not synchronized with movement, and they will break blending.

|  |  |
| --- | --- |
| Note | M62-65 will not function unless the appropriate motion.digital-out-nn pins are connected in your hal file to outputs. |

## 16. M66 Wait on Input

M66 P- | E- <L->

* P- - specifies the digital input number from 0 to 3.
* E- - specifies the analog input number from 0 to 3.
* L- - specifies the wait mode type.
	+ Mode 0: IMMEDIATE - no waiting, returns immediately. The current value of the input is stored in parameter #5399
	+ Mode 1: RISE - waits for the selected input to perform a rise event.
	+ Mode 2: FALL - waits for the selected input to perform a fall event.
	+ Mode 3: HIGH - waits for the selected input to go to the HIGH state.
	+ Mode 4: LOW - waits for the selected input to go to the LOW state.
* Q- - specifies the timeout in seconds for waiting. If the timeout is exceeded, the wait is interrupt, and the variable #5399 will be holding the value -1. The Q value is ignored if the L-word is zero (IMMEDIATE). A Q value of zero is an error if the L-word is non-zero.
* Mode 0 is the only one permitted for an analog input.

M66 Example Lines

M66 P0 L3 Q5 (wait up to 5 seconds for digital input 0 to turn on)

M66 wait on an input stops further execution of the program, until the selected event (or the programmed timeout) occurs.

It is an error to program M66 with both a P-word and an E-word (thus selecting both an analog and a digital input). In LinuxCNC these inputs are not monitored in real time and thus should not be used for timing-critical applications.

The number of I/O can be increased by using the num\_dio or num\_aio parameter when loading the motion controller. See the [Motion Section](http://linuxcnc.org/docs/html/config/core-components.html%22%20%5Cl%20%22sec%3Amotion) for more information.

|  |  |
| --- | --- |
| Note | M66 will not function unless the appropriate motion.digital-in-nn pins or motion.analog-in-nn pins are connected in your hal file to an input. |

Example HAL Connection

net signal-name motion.digital-in-00 <= parport.0.pin10-in

## 17. M67 Analog Output,Synchronized

M67 E- Q-

* M67 - set an analog output synchronized with motion.
* E- - output number ranging from 0 to 3.
* Q- - is the value to set (set to 0 to turn off).

The actual change of the specified outputs will happen at the beginning of the next motion command. If there is no subsequent motion command, the queued output changes won’t happen. It’s best to always program a motion G code (G0, G1, etc) right after the M67. M67 functions the same as M62-63.

The number of I/O can be increased by using the num\_dio or num\_aio parameter when loading the motion controller. See the [Motion Section](http://linuxcnc.org/docs/html/config/core-components.html%22%20%5Cl%20%22sec%3Amotion) for more information.

|  |  |
| --- | --- |
| Note | M67 will not function unless the appropriate motion.analog-out-nn pins are connected in your hal file to outputs. |

## 18. M68 Analog Output, Immediate

M68 E- Q-

* M68 - set an analog output immediately.
* E- - output number ranging from 0 to 3.
* Q- - is the value to set (set to 0 to turn off).

M68 output happen immediately as they are received by the motion controller. They are not synchronized with movement, and they will break blending. M68 functions the same as M64-65.

The number of I/O can be increased by using the num\_dio or num\_aio parameter when loading the motion controller. See the [Motion Section](http://linuxcnc.org/docs/html/config/core-components.html%22%20%5Cl%20%22sec%3Amotion) for more information.

|  |  |
| --- | --- |
| Note | M68 will not function unless the appropriate motion.analog-out-nn pins are connected in your hal file to outputs. |

## 19. M70 Save Modal State

To explicitly save the modal state at the current call level, program M70. Once modal state has been saved with M70, it can be restored to exactly that state by executing an M72.

A pair of M70 and M72 instructions will typically be used to protect a program against inadvertant modal changes within subroutines.

The state saved consists of:

* current G20/G21 settings (imperial/metric)
* selected plane (G17/G18/G19 G17.1,G18.1,G19.1)
* status of cutter compensation (G40,G41,G42,G41.1,G42,1)
* distance mode - relative/absolute (G90/G91)
* feed mode (G93/G94,G95)
* current coordinate system (G54-G59.3)
* tool length compensation status (G43,G43.1,G49)
* retract mode (G98,G99)
* spindle mode (G96-css or G97-RPM)
* arc distance mode (G90.1, G91.1)
* lathe radius/diameter mode (G7,G8)
* path control mode (G61, G61.1, G64)
* current feed and speed (F and S values)
* spindle status (M3,M4,M5) - on/off and direction
* mist (M7) and flood (M8) status
* speed override (M51) and feed override (M50) settings
* adaptive feed setting (M52)
* feed hold setting (M53)

Note that in particular, the motion mode (G1 etc) is NOT restored.

current call level means either:

* executing in the main program. There is a single storage location for state at the main program level; if several M70 instructions are executed in turn, only the most recently saved state is restored when an M72 is executed.
* executing within a G-code subroutine. The state saved with M70 within a subroutine behaves exactly like a local named parameter - it can be referred to only within this subroutine invocation with an M72 and when the subroutine exits, the parameter goes away.

A recursive invocation of a subroutine introduces a new call level.

## 20. M71 Invalidate Stored Modal State

Modal state saved with an M70 or by an M73 at the current call level is invalidated (cannot be restored from anymore).

A subsequent M72 at the same call level will fail.

If executed in a subroutine which protects modal state by an M73, a subsequent return or endsub will not restore modal state.

The usefulness of this feature is dubious. It should not be relied upon as it might go away.

## 21. M72 Restore Modal State

[Modal state saved with an](http://linuxcnc.org/docs/html/gcode/m-code.html%22%20%5Cl%20%22mcode%3Am70-saved-state) M70 code can be restored by executing an M72.

The handling of G20/G21 is specially treated as feeds are interpreted differently depending on G20/G21: if length units (mm/in) are about to be changed by the restore operation, 'M72 'will restore the distance mode first, and then all other state including feed to make sure the feed value is interpreted in the correct unit setting.

It is an error to execute an M72 with no previous M70 save operation at that level.

The following example demonstrates saving and explicitely restoring modal state around a subroutine call using M70 and M72. Note that the imperialsub subroutine is not "aware" of the M7x features and can be used unmodified:

**O<showstate> sub**

*(DEBUG, imperial=*#<\_imperial> *absolute=*#<\_absolute> *feed=*#<\_feed> *rpm=*#<\_rpm>*)*

**O<showstate> endsub**

**O<imperialsub> sub**

**g20** *(imperial)*

**g91** *(relative mode)*

**F5** *(low feed)*

**S300** *(low rpm)*

*(debug, in subroutine, state now:)*

**o<showstate> call**

**O<imperialsub> endsub**

*; main program*

**g21** *(metric)*

**g90** *(absolute)*

**f200** *(fast speed)*

**S2500** *(high rpm)*

*(debug, in main, state now:)*

**o<showstate> call**

**M70** *(save caller state in at global level)*

**O<imperialsub> call**

**M72** *(explicitely restore state)*

*(debug, back in main, state now:)*

**o<showstate> call**

**m2**

## 22. M73 Save and Autorestore Modal State

To save modal state within a subroutine, and restore state on subroutine endsub or any return path, program M73.

Aborting a running program in a subroutine which has an M73 operation will not restore state .

Also, the normal end (M2) of a main program which contains an M73 will not restore state.

The suggested use is at the beginning of a O-word subroutine as in the following example. Using M73 this way enables designing subroutines which need to modify modal state but will protect the calling program against inadvertant modal changes. Note the use of [predefined named parameters](http://linuxcnc.org/docs/html/gcode/overview.html%22%20%5Cl%20%22gcode%3Apredefined-named-parameters) in the showstate subroutine.

**O<showstate> sub**

*(DEBUG, imperial=*#<\_imperial> *absolute=*#<\_absolute> *feed=*#<\_feed> *rpm=*#<\_rpm>*)*

**O<showstate> endsub**

**O<imperialsub> sub**

**M73** *(save caller state in current call context, restore on return or endsub)*

**g20** *(imperial)*

**g91** *(relative mode)*

**F5** *(low feed)*

**S300** *(low rpm)*

*(debug, in subroutine, state now:)*

**o<showstate> call**

*; note - no M72 is needed here - the following endsub or an*

*; explicit 'return' will restore caller state*

**O<imperialsub> endsub**

*; main program*

**g21** *(metric)*

**g90** *(absolute)*

**f200** *(fast speed)*

**S2500** *(high rpm)*

*(debug, in main, state now:)*

**o<showstate> call**

**o<imperialsub> call**

*(debug, back in main, state now:)*

**o<showstate> call**

**m2**

### 22.1. Selectively Restoring Modal State

Executing an M72 or returning from a subroutine which contains an M73 will restore [all](http://linuxcnc.org/docs/html/gcode/m-code.html%22%20%5Cl%20%22mcode%3Am70-saved-state) modal state saved.

If only some aspects of modal state should be preserved, an alternative is the usage of [predefined named parameters](http://linuxcnc.org/docs/html/gcode/overview.html%22%20%5Cl%20%22gcode%3Apredefined-named-parameters), local parameters and conditional statements. The idea is to remember the modes to be restored at the beginning of the subroutine, and restore these before exiting. Here is an example, based on snippet of nc\_files/tool-length-probe.ngc:

**O<measure> sub** *(measure reference tool)*

*;*

#<absolute> **=** #<\_absolute> *(remember in local variable if G90 was set)*

*;*

**g30** *(above switch)*

**g38.2** z0 **f15** *(measure)*

**g91** **g0**z.2 *(off the switch)*

#1000**=**#5063 *(save reference tool length)*

*(print,reference length is #1000)*

*;*

**O<restore\_abs> if** **[**#<absolute>**]**

 **g90** *(restore G90 only if it was set on entry:)*

**O<restore\_abs> endif**

*;*

**O<measure> endsub**

## 23. M100 - M199 User Defined Commands

M1-- <P- Q->

* M1-- - an integer in the range of 100 - 199.
* P- - a number passed to the file as the first parameter.
* Q- - a number passed to the file as the second parameter.

|  |  |
| --- | --- |
| Note | After creating a new M1nn file you must restart the GUI so it is aware of the new file, otherwise you will get an Unkown m code error. |

The external program named M100 through M199 (no extension and a capitol M) is executed with the optional P and Q values as its two arguments. Execution of the G code file pauses until the external program exits. Any valid executable file can be used. The file must be located in the search path specificed in the ini file configuration. See the [Display Section](http://linuxcnc.org/docs/html/config/ini-config.html%22%20%5Cl%20%22sec%3Adisplay-section) for more information on search paths.

|  |  |
| --- | --- |
| Warning | Do not use a word processor to create or edit the files. A word processor will leave unseen codes that will cause problems and may prevent a bash or python file from working. Use a text editor like Gedit in Ubuntu or Notepad++ in other operating systems to create or edit the files. |

The error Unknown M code used denotes one of the following

* The specified User Defined Command does not exist
* The file is not an executable file
* The file name has an extension
* The file name does not follow this format M1nn where nn = 00 through 99
* The file name used a lower case M

For example to open and close a collet closer that is controlled by a parallel port pin using a bash script file using M101 and M102. Create two files named M101 and M102. Set them as executable files (typically right click/properties/permissions) before running LinuxCNC. Make sure the parallel port pin is not connected to anything in a HAL file.

M101 Example File

#!/bin/bash

# file to turn on parport pin 14 to open the collet closer

halcmd setp parport.0.pin-14-out True

exit 0

M102 Example File

#!/bin/bash

# file to turn off parport pin 14 to open the collet closer

halcmd setp parport.0.pin-14-out False

exit 0

To pass a variable to a M1nn file you use the P and Q option like this:

M100 P123.456 Q321.654

M100 Example file

#!/bin/bash

voltage=$1

feedrate=$2

halcmd setp thc.voltage $voltage

halcmd setp thc.feedrate $feedrate

exit 0

To display a graphic message and stop until the message window is closed use a graphic display program like Eye of Gnome to display the graphic file. When you close it the program will resume.

M110 Example file

#!/bin/bash

eog /home/john/linuxcnc/nc\_files/message.png

exit 0

To display a graphic message and continue processing the G code file suffix an ampersand to the command.

M110 Example display and keep going

#!/bin/bash

eog /home/john/linuxcnc/nc\_files/message.png &

exit 0

Last updated 2020-01-24 07:10:42 MST