Contents

1 Introduction 1

2 Code Notes 2
  2.1 Intended audience 2
  2.2 Organization 2
  2.3 Terms and definitions 2
  2.4 Architecture overview 3
  2.5 Motion Controller Introduction 5
  2.6 Block diagrams and Data Flow 7
  2.7 Homing 10
    2.7.1 Homing state diagram 10
    2.7.2 Another homing diagram 11
  2.8 Commands 11
    2.8.1 ABORT 11
      2.8.1.1 Requirements 11
      2.8.1.2 Results 12
    2.8.2 FREE 12
      2.8.2.1 Requirements 12
      2.8.2.2 Results 12
    2.8.3 TELEOP 12
      2.8.3.1 Requirements 12
      2.8.3.2 Results 12
    2.8.4 COORD 13
      2.8.4.1 Requirements 13
      2.8.4.2 Results 13
    2.8.5 ENABLE 13
      2.8.5.1 Requirements 13
      2.8.5.2 Results 13
    2.8.6 DISABLE 13
      2.8.6.1 Requirements 13
2.24.1 Nonrandom Toolchangers ........................................................... 25
2.24.2 Random Toolchangers ............................................................. 26
2.24.2 The Tool Table ........................................................................... 26
2.24.3 Gcodes affecting tools ............................................................... 26
  2.24.3.1 Txxx ....................................................................................... 27
  2.24.3.2 M6 ......................................................................................... 27
  2.24.3.3 G43/G43.1/G49 ................................................................. 27
  2.24.3.4 G10 L1/L10/L11 ............................................................... 28
  2.24.3.5 M61 .................................................................................... 28
  2.24.3.6 G41/G41.1/G42/G42.1 ...................................................... 29
  2.24.3.7 G40 .................................................................................... 29
2.24.4 Internal state variables ............................................................. 29
  2.24.4.1 IO ....................................................................................... 29
  2.24.4.2 interp.................................................................................. 29

3 NML Messages ................................................................................. 31
  3.1 EMC OPERATOR ................................................................. 31
    3.1.1 EMC_OPERATOR_ERROR_TYPE ............................................ 31
    3.1.2 EMC_OPERATOR_TEXT_TYPE .............................................. 31
    3.1.3 EMC_OPERATOR_DISPLAY_TYPE ..................................... 31
  3.2 EMC NULL, SET, DEBUG, & SYSTEM ....................................... 31
    3.2.1 EMC_NULL_TYPE ............................................................... 31
    3.2.2 EMC_SET_DEBUG_TYPE .................................................... 32
    3.2.3 EMC_SYSTEM_CMD_TYPE ................................................. 32
  3.3 EMC AXIS ............................................................................... 32
    3.3.1 EMC_AXIS_SET_AXIS_TYPE .............................................. 32
    3.3.2 EMC_AXIS_SET_UNITS_TYPE ............................................. 32
    3.3.3 EMC_AXIS_SET_GAINS_TYPE ........................................... 32
    3.3.4 EMC_AXIS_SET_CYCLE_TIME_TYPE ................................. 32
    3.3.5 EMC_AXIS_SET_INPUT_SCALE_TYPE ................................. 33
    3.3.6 EMC_AXIS_SET_OUTPUT_SCALE_TYPE ............................. 33
    3.3.7 EMC_AXIS_SET_MIN_POSITION_LIMIT_TYPE .................... 33
    3.3.8 EMC_AXIS_SET_MAX_POSITION_LIMIT_TYPE .................... 33
    3.3.9 EMC_AXIS_SET_MIN_OUTPUT_LIMIT_TYPE ....................... 33
    3.3.10 EMC_AXIS_SET_MAX_OUTPUT_LIMIT_TYPE ..................... 33
    3.3.11 EMC_AXIS_SET_FERROR_TYPE ........................................ 34
    3.3.12 EMC_AXIS_SET_HOMING_VEL_TYPE .............................. 34
    3.3.13 EMC_AXIS_SET_HOME_TYPE ............................................ 34
    3.3.14 EMC_AXIS_SET_HOME_OFFSET_TYPE .............................. 34
3.3.15  EMC_AXIS_SET_MIN_FERROR_TYPE  ......................................................... 34
3.3.16  EMC_AXIS_SET_MAX_VELOCITY_TYPE .................................................... 35
3.3.17  EMC_AXIS_INIT_TYPE ........................................................................... 35
3.3.18  EMC_AXIS_HALT_TYPE ......................................................................... 35
3.3.19  EMC_AXIS_ABORT_TYPE ........................................................................ 35
3.3.20  EMC_AXIS_ENABLE_TYPE ...................................................................... 35
3.3.21  EMC_AXIS_DISABLE_TYPE ..................................................................... 36
3.3.22  EMC_AXIS_HOME_TYPE .......................................................................... 36
3.3.23  EMC_AXIS_JOG_TYPE ............................................................................ 36
3.3.24  EMC_AXIS_INCR_JOG_TYPE .................................................................. 36
3.3.25  EMC_AXIS_ABS_JOG_TYPE .................................................................... 36
3.3.26  EMC_AXIS_ACTIVATE_TYPE .................................................................. 37
3.3.27  EMC_AXIS_DEACTIVATE_TYPE ............................................................... 37
3.3.28  EMC_AXIS_OVERRIDE_LIMITS_TYPE ....................................................... 37
3.3.29  EMC_AXIS_SET_OUTPUT_TYPE ............................................................... 37
3.3.30  EMC_AXIS_SET_STEP_PARAMS_TYPE .................................................... 38
3.3.31  EMC_AXIS_SET_UNITS_TYPE .................................................................. 38
3.3.32  EMC_AXIS_STAT_TYPE .......................................................................... 38
3.3.33  EMC_AXIS_ALTER_TYPE ......................................................................... 38
3.4   EMC TRAJ .............................................................................................. 38
3.4.1  EMC_TRAJ_SET_AXES_TYPE ................................................................. 38
3.4.2  EMC_TRAJ_SET_UNITS_TYPE .................................................................. 38
3.4.3  EMC_TRAJ_SET_CYCLE_TIME_TYPE ....................................................... 39
3.4.4  EMC_TRAJ_SET_MODE_TYPE ................................................................... 39
3.4.5  EMC_TRAJ_SET_VELOCITY_TYPE ............................................................ 39
3.4.6  EMC_TRAJ_SET_ACCELERATION_TYPE ................................................ 39
3.4.7  EMC_TRAJ_SET_MAX_VELOCITY_TYPE ................................................... 39
3.4.8  EMC_TRAJ_SET_MAX_ACCELERATION_TYPE .......................................... 40
3.4.9  EMC_TRAJ_SET_SCALE_TYPE .................................................................. 40
3.4.10 EMC_TRAJ_SET_MOTION_ID_TYPE ......................................................... 40
3.4.11 EMC_TRAJ_INIT_TYPE ............................................................................ 40
3.4.12 EMC_TRAJ_HALT_TYPE ......................................................................... 40
3.4.13 EMC_TRAJ_ENABLE_TYPE ...................................................................... 41
3.4.14 EMC_TRAJ_DISABLE_TYPE .................................................................... 41
3.4.15 EMC_TRAJ_ABORT_TYPE ........................................................................ 41
3.4.16 EMC_TRAJ_PAUSE_TYPE ........................................................................ 41
3.4.17 EMC_TRAJ_STEP_TYPE .......................................................................... 41
3.4.18 EMC_TRAJ_RESUME_TYPE ...................................................................... 42
3.4.19 EMC_TRAJ_DELAY_TYPE ........................................................................ 42
3.4.20 EMC_TRAJ_LINEAR_MOVE_TYPE ................................................. 42
3.4.21 EMC_TRAJ_CIRCULAR_MOVE_TYPE ........................................... 42
3.4.22 EMC_TRAJ_SET_TERM_COND_TYPE ............................................ 42
3.4.23 EMC_TRAJ_SET_OFFSET_TYPE ................................................ 43
3.4.24 EMC_TRAJ_SET_ORIGIN_TYPE ................................................. 43
3.4.25 EMC_TRAJ_SET_HOME_TYPE .................................................. 43
3.4.26 EMC_TRAJ_SET_PROBE_INDEX_TYPE ........................................ 43
3.4.27 EMC_TRAJ_SET_PROBE_POLARITY_TYPE .................................... 43
3.4.28 EMC_TRAJ_CLEAR_PROBE_TRIPPED_FLAG_TYPE .............................. 44
3.4.29 EMC_TRAJ_PROBE_TYPE ..................................................... 44
3.4.30 EMC_TRAJ_SET_TELEOP_ENABLE_TYPE ..................................... 44
3.4.31 EMC_TRAJ_SET_TELEOP_VECTOR_TYPE ..................................... 44
3.4.32 EMC_TRAJ_STAT_TYPE ..................................................... 44

3.5 EMC MOTION ................................................................. 45

3.5.1 EMC_MOTION_INIT_TYPE .................................................... 45
3.5.2 EMC_MOTION_HALT_TYPE ................................................... 45
3.5.3 EMC_MOTION_ABORT_TYPE ................................................ 45
3.5.4 EMC_MOTION_SET_AOUT_TYPE ............................................. 45
3.5.5 EMC_MOTION_SET_DOUT_TYPE ............................................. 45
3.5.6 EMC_MOTION_STAT_TYPE .................................................. 46

3.6 EMC TASK ................................................................. 46

3.6.1 EMC_TASK_INIT_TYPE ..................................................... 46
3.6.2 EMC_TASK_HALT_TYPE ..................................................... 46
3.6.3 EMC_TASK_ABORT_TYPE .................................................. 46
3.6.4 EMC_TASK_SET_MODE_TYPE .............................................. 46
3.6.5 EMC_TASK_SET_STATE_TYPE .............................................. 46
3.6.6 EMC_TASK_PLAN_OPEN_TYPE ............................................. 47
3.6.7 EMC_TASK_PLAN_RUN_TYPE .............................................. 47
3.6.8 EMC_TASK_PLAN_READ_TYPE ............................................. 47
3.6.9 EMC_TASK_PLAN_EXECUTE_TYPE ....................................... 47
3.6.10 EMC_TASK_PLAN_PAUSE_TYPE ........................................... 47
3.6.11 EMC_TASK_PLAN_STEP_TYPE ............................................ 47
3.6.12 EMC_TASK_PLAN_RESUME_TYPE ........................................ 48
3.6.13 EMC_TASK_PLAN_END_TYPE ............................................. 48
3.6.14 EMC_TASK_PLAN_CLOSE_TYPE .......................................... 48
3.6.15 EMC_TASK_PLAN_INIT_TYPE ............................................. 48
3.6.16 EMC_TASK_PLAN_SYNCH_TYPE ........................................ 48
3.6.17 EMC_TASK_STAT_TYPE .................................................. 48

3.7 EMC TOOL ................................................................. 49
3.7.1 EMC_TOOL_INIT_TYPE .......................................................... 49
3.7.2 EMC_TOOL_HALT_TYPE ...................................................... 49
3.7.3 EMC_TOOL_ABORT_TYPE .................................................... 49
3.7.4 EMC_TOOL_PREPARE_TYPE .................................................. 49
3.7.5 EMC_TOOL_LOAD_TYPE ....................................................... 50
3.7.6 EMC_TOOL_UNLOAD_TYPE ................................................... 50
3.7.7 EMC_TOOL_LOAD_TOOL_TABLE_TYPE .................................... 50
3.7.8 EMC_TOOL_SET_OFFSET_TYPE .............................................. 50
3.7.9 EMC_TOOL_STAT_TYPE ....................................................... 50
3.8 EMC AUX ......................................................................... 51
3.8.1 EMC_AUX_INIT_TYPE .......................................................... 51
3.8.2 EMC_AUX_HALT_TYPE ........................................................ 51
3.8.3 EMC_AUX_ABORT_TYPE ...................................................... 51
3.8.4 EMC_AUX_DIO_WRITE_TYPE ............................................... 51
3.8.5 EMC_AUX_AIO_WRITE_TYPE ............................................... 51
3.8.6 EMC_AUX_ESTOP_ON_TYPE ................................................. 51
3.8.7 EMC_AUX_ESTOP_OFF_TYPE ................................................ 52
3.8.8 EMC_AUX_STAT_TYPE ........................................................ 52
3.9 EMC SPINDLE ................................................................. 52
3.9.1 EMC_SPINDLE_INIT_TYPE .................................................... 52
3.9.2 EMC_SPINDLE_HALT_TYPE .................................................. 52
3.9.3 EMC_SPINDLE_ABORT_TYPE ................................................ 52
3.9.4 EMC_SPINDLE_ON_TYPE ..................................................... 52
3.9.5 EMC_SPINDLE_OFF_TYPE ..................................................... 53
3.9.6 EMC_SPINDLE_FORWARD_TYPE .............................................. 53
3.9.7 EMC_SPINDLE_REVERSE_TYPE ............................................ 53
3.9.8 EMC_SPINDLE_STOP_TYPE ................................................... 53
3.9.9 EMC_SPINDLE_INCREASE_TYPE ............................................ 53
3.9.10 EMC_SPINDLE_DECREASE_TYPE ......................................... 53
3.9.11 EMC_SPINDLE_CONSTANT_TYPE ......................................... 54
3.9.12 EMC_SPINDLE_BRAKE_RELEASE_TYPE .................................. 54
3.9.13 EMC_SPINDLE_BRAKE_ENGAGE_TYPE .................................... 54
3.9.14 EMC_SPINDLE_ENABLE_TYPE ............................................. 54
3.9.15 EMC_SPINDLE_DISABLE_TYPE ............................................ 54
3.9.16 EMC_SPINDLE_STAT_TYPE ................................................ 54
3.10 EMC COOLANT ............................................................... 55
3.10.1 EMC_COOLANT_INIT_TYPE ............................................... 55
3.10.2 EMC_COOLANT_HALT_TYPE .............................................. 55
3.10.3 EMC_COOLANT_ABORT_TYPE ............................................. 55
Chapter 1

Introduction

This handbook is a work in progress. If you are able to help with writing, editing, or graphic preparation please contact any member of the writing team or join and send an email to emc-users@lists.sourceforge.net.

Copyright © 2000-2015 LinuxCNC.org

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

LINUX® is the registered trademark of Linus Torvalds in the U.S. and other countries. The registered trademark Linux® is used pursuant to a sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the mark on a world-wide basis.

The LinuxCNC project is not affiliated with Debian®. Debian is a registered trademark owned by Software in the Public Interest, Inc.

The LinuxCNC project is not affiliated with UBUNTU®. UBUNTU is a registered trademark owned by Canonical Limited.
Chapter 2

Code Notes

2.1 Intended audience

This document is a collection of notes about the internals of LinuxCNC. It is primarily of interest to developers, however much of the information here may also be of interest to system integrators and others who are simply curious about how LinuxCNC works. Much of this information is now outdated and has never been reviewed for accuracy.

2.2 Organization

There will be a chapter for each of the major components of LinuxCNC, as well as chapter(s) covering how they work together. This document is very much a work in progress, and its layout may change in the future.

2.3 Terms and definitions

- **AXIS** - An axis is one of the nine degrees of freedom that define a tool position in three-dimensional Cartesian space. Those nine axes are referred to as X, Y, Z, A, B, C, U, V, and W. The linear orthogonal coordinates X, Y, and Z determine where the tip of the tool is positioned. The angular coordinates A, B, and C determine the tool orientation. A second set of linear orthogonal coordinates U, V, and W allows tool motion (typically for cutting actions) relative to the previously offset and rotated axes. Unfortunately, “axis” is also sometimes used to mean a degree of freedom of the machine itself, such as the saddle, table, or quill of a Bridgeport type milling machine. On a Bridgeport this causes no confusion, since movement of the table directly corresponds to movement along the X axis. However, the shoulder and elbow joints of a robot arm and the linear actuators of a hexapod do not correspond to movement along any Cartesian axis, and in general it is important to make the distinction between the Cartesian axes and the machine degrees of freedom. In this document, the latter will be called **joints**, not axes. (The GUIs and some other parts of the code may not always follow this distinction, but the internals of the motion controller do.)

- **JOINT** - A joint is one of the movable parts of the machine. Joints are distinct from axes, although the two terms are sometimes (mis)used to mean the same thing. In LinuxCNC, a joint is a physical thing that can be moved, not a coordinate in space. For example, the quill, knee, saddle, and table of a Bridgeport mill are all joints. The shoulder, elbow, and wrist of a robot arm are joints, as are the linear actuators of a hexapod. Every joint has a motor or actuator of some type associated with it. Joints do not necessarily correspond to the X, Y, and Z axes, although for machines with trivial kinematics that may be the case. Even on those machines, joint position and axis position are fundamentally different things. In this document, the terms **joint** and **axis** are used carefully to respect their distinct meanings. Unfortunately that isn’t necessarily true everywhere else. In particular, GUIs for machines with trivial kinematics may gloss over or completely hide the distinction between joints and axes. In addition, the ini file uses the term **axis** for data that would more accurately be described as joint data, such as input and output scaling, etc.
• **POSE** - A pose is a fully specified position in 3-D Cartesian space. In the LinuxCNC motion controller, when we refer to a pose we mean an EmcPose structure, containing six linear coordinates (X, Y, Z, U, V, and W) and three angular ones (A, B, and C).

### 2.4 Architecture overview

There are four components contained in the LinuxCNC Architecture: a motion controller (EMCMOT), a discrete IO controller (EMCIO), a task executor which coordinates them (EMCTASK) and several text-mode and graphical User Interfaces. Each of them will be described in the current document, both from the design point of view and from the developers point of view (where to find needed data, how to easily extend/modify things, etc.).
LinuxCNC software architecture. At the coarsest level, LinuxCNC is a hierarchy of three controllers: the task level command handler and program interpreter, the motion controller, and the discrete I/O controller. The discrete I/O controller is implemented as a hierarchy of controllers, in this case for spindle, coolant, and auxiliary (e.g., estop, lube) subsystems. The task controller coordinates the actions of the motion and discrete I/O controllers. Their actions are programmed in conventional numerical control "G and M code" programs, which are interpreted by the task controller into NML messages and sent to either the motion or discrete I/O controllers at the appropriate times.

### 2.5 Motion Controller Introduction

The motion controller receives commands from user space modules via a shared memory buffer, and executes those commands in realtime. The status of the controller is made available to the user space modules through the same shared memory area. The motion controller interacts with the motors and other hardware using the HAL (Hardware Abstraction Layer). This document assumes that the reader has a basic understanding of the HAL, and uses terms like HAL pins, HAL signals, etc, without explaining them. For more information about the HAL, see the HAL Manual. Another chapter of this document will eventually go into the internals of the HAL itself, but in this chapter, we only use the HAL API as defined in src/hal/hal.h.
2.6 Block diagrams and Data Flow

The following figure is a block diagram of a joint controller. There is one joint controller per joint. The joint controllers work at a lower level than the kinematics, a level where all joints are completely independent. All the data for a joint is in a single joint structure. Some members of that structure are visible in the block diagram, such as coarse_pos, pos_cmd, and motor_pos_fb.

Joint Controller Block Diagram

The above figure shows five of the seven sets of position information that form the main data flow through the motion controller. The seven forms of position data are as follows:

1. `emcmotStatus->carte_pos_cmd` - This is the desired position, in Cartesian coordinates. It is updated at the traj rate, not the servo rate. In coord mode, it is determined by the traj planner. In teleop mode, it is determined by the traj planner? In free mode, it is either copied from actualPos, or generated by applying forward kins to (2) or (3).

2. `emcmotStatus->joints[n].coarse_pos` - This is the desired position, in joint coordinates, but before interpolation. It is updated at the traj rate, not the servo rate. In coord mode, it is generated by applying inverse kins to (1). In teleop mode, it is generated by applying inverse kins to (1). In free mode, it is generated from (3), I think.

3. `emcmotStatus->joints[n].pos_cmd` - This is the desired position, in joint coords, after interpolation. A new set of these coords is generated every servo period. In coord mode, it is generated from (2) by the interpolator. In teleop mode, it is generated from (2) by the interpolator. In free mode, it is generated by the free mode traj planner.

4. `emcmotStatus->joints[n].motor_pos_cmd` - This is the desired position, in motor coords. Motor coords are generated by adding backlash compensation, lead screw error compensation, and offset (for homing) to (3). It is generated the same way regardless of the mode, and is the output to the PID loop or other position loop.
5. `emcmotStatus->joints[n].motor_pos_fb` - This is the actual position, in motor coords. It is the input from encoders or other feedback device (or from virtual encoders on open loop machines). It is "generated" by reading the feedback device.

6. `emcmotStatus->joints[n].pos_fb` - This is the actual position, in joint coordinates. It is generated by subtracting offset, lead screw error compensation, and backlash compensation from (5). It is generated the same way regardless of the operating mode.

7. `emcmotStatus->carte_pos_fb` - This is the actual position, in Cartesian coordinates. It is updated at the traj rate, not the servo rate. Ideally, `actualPos` would always be calculated by applying forward kinematics to (6). However, forward kinematics may not be available, or they may be unusable because one or more axes aren’t homed. In that case, the options are: A) fake it by copying (1), or B) admit that we don’t really know the Cartesian coordinates, and simply don’t update `actualPos`. Whatever approach is used, I can see no reason not to do it the same way regardless of the operating mode. I would propose the following: If there are forward kins, use them, unless they don’t work because of unhomed axes or other problems, in which case do (B). If no forward kins, do (A), since otherwise `actualPos` would never get updated.
2.7 Homing

2.7.1 Homing state diagram
2.7.2 Another homing diagram

![Diagram showing the homing sequence]

2.8 Commands

This section simply lists all of the commands that can be sent to the motion module, along with detailed explanations of what they do. The command names are defined in a large typedef enum in emc2/src/emc/motion/motion.h, called cmd_code_t. (Note that in the code, each command name starts with `EMCMOT_`, which is omitted here.)

The commands are implemented by a large switch statement in the function emcmotCommandHandler(), which is called at the servo rate. More on that function later. There are approximately 44 commands - this list is still under construction.

2.8.1 ABORT

The ABORT command simply stops all motion. It can be issued at any time, and will always be accepted. It does not disable the motion controller or change any state information, it simply cancels any motion that is currently in progress.\(^1\)

2.8.1.1 Requirements

None. The command is always accepted and acted on immediately.

\(^1\) It seems that the higher level code (TASK and above) also use ABORT to clear faults. Whenever there is a persistent fault (such as being outside the hardware limit switches), the higher level code sends a constant stream of ABORTs to the motion controller trying to make the fault go away. Thousands of ‘em… That means that the motion controller should avoid persistent faults. This needs to be looked into.
2.8.1.2 Results

In free mode, the free mode trajectory planners are disabled. That results in each joint stopping as fast as its accel (decel) limit allows. The stop is not coordinated. In teleop mode, the commanded Cartesian velocity is set to zero. I don’t know exactly what kind of stop results (coordinated, uncoordinated, etc), but will figure it out eventually. In coord mode, the coord mode trajectory planner is told to abort the current move. Again, I don’t know the exact result of this, but will document it when I figure it out.

2.8.2 FREE

The FREE command puts the motion controller in free mode. Free mode means that each joint is independent of all the other joints. Cartesian coordinates, poses, and kinematics are ignored when in free mode. In essence, each joint has its own simple trajectory planner, and each joint completely ignores the other joints. Some commands (like JOG) only work in free mode. Other commands, including anything that deals with Cartesian coordinates, do not work at all in free mode.

2.8.2.1 Requirements

The command handler applies no requirements to the FREE command, it will always be accepted. However, if any joint is in motion (GET_MOTION_INPOS_FLAG() == FALSE), then the command will be ignored. This behavior is controlled by code that is now located in the function set_operating_mode() in control.c, that code needs to be cleaned up. I believe the command should not be silently ignored, instead the command handler should determine whether it can be executed and return an error if it cannot.

2.8.2.2 Results

If the machine is already in free mode, nothing. Otherwise, the machine is placed in free mode. Each joint’s free mode trajectory planner is initialized to the current location of the joint, but the planners are not enabled and the joints are stationary.

2.8.3 TELEOP

The TELEOP command places the machine in teleoperating mode. In teleop mode, movement of the machine is based on Cartesian coordinates using kinematics, rather than on individual joints as in free mode. However the trajectory planner per se is not used, instead movement is controlled by a velocity vector. Movement in teleop mode is much like jogging, except that it is done in Cartesian space instead of joint space. On a machine with trivial kinematics, there is little difference between teleop mode and free mode, and GUIs for those machines might never even issue this command. However for non-trivial machines like robots and hexapods, teleop mode is used for most user commanded jog type movements.

2.8.3.1 Requirements

The command handler will reject the TELEOP command with an error message if the kinematics cannot be activated because the one or more axes have not been homed. In addition, if any joint is in motion (GET_MOTION_INPOS_FLAG() == FALSE), then the command will be ignored (with no error message). This behavior is controlled by code that is now located in the function set_operating_mode() in control.c. I believe the command should not be silently ignored, instead the command handler should determine whether it can be executed and return an error if it cannot.

2.8.3.2 Results

If the machine is already in teleop mode, nothing. Otherwise the machine is placed in teleop mode. The kinematics code is activated, interpolators are drained and flushed, and the Cartesian velocity commands are set to zero.
2.8.4 COORD

The COORD command places the machine in coordinated mode. In coord mode, movement of the machine is based on Cartesian coordinates using kinematics, rather than on individual joints as in free mode. In addition, the main trajectory planner is used to generate motion, based on queued LINE, CIRCLE, and/or PROBE commands. Coord mode is the mode that is used when executing a G-code program.

2.8.4.1 Requirements

The command handler will reject the COORD command with an error message if the kinematics cannot be activated because the one or more axes have not been homed. In addition, if any joint is in motion (GET_MOTION_INPOS_FLAG() == FALSE), then the command will be ignored (with no error message). This behavior is controlled by code that is now located in the function set_operating_mode() in control.c. I believe the command should not be silently ignored, instead the command handler should determine whether it can be executed and return an error if it cannot.

2.8.4.2 Results

If the machine is already in coord mode, nothing. Otherwise, the machine is placed in coord mode. The kinematics code is activated, interpolators are drained and flushed, and the trajectory planner queues are empty. The trajectory planner is active and awaiting a LINE, CIRCLE, or PROBE command.

2.8.5 ENABLE

The ENABLE command enables the motion controller.

2.8.5.1 Requirements

None. The command can be issued at any time, and will always be accepted.

2.8.5.2 Results

If the controller is already enabled, nothing. If not, the controller is enabled. Queues and interpolators are flushed. Any movement or homing operations are terminated. The amp-enable outputs associated with active joints are turned on. If forward kinematics are not available, the machine is switched to free mode.

2.8.6 DISABLE

The DISABLE command disables the motion controller.

2.8.6.1 Requirements

None. The command can be issued at any time, and will always be accepted.

2.8.6.2 Results

If the controller is already disabled, nothing. If not, the controller is disabled. Queues and interpolators are flushed. Any movement or homing operations are terminated. The amp-enable outputs associated with active joints are turned off. If forward kinematics are not available, the machine is switched to free mode.
2.8.7  ENABLE_AMPLIFIER

The ENABLE_AMPLIFIER command turns on the amp enable output for a single output amplifier, without changing anything else. Can be used to enable a spindle speed controller.

2.8.7.1 Requirements

None. The command can be issued at any time, and will always be accepted.

2.8.7.2 Results

Currently, nothing. (A call to the old extAmpEnable function is currently commented out.) Eventually it will set the amp enable HAL pin true.

2.8.8  DISABLE_AMPLIFIER

The DISABLE_AMPLIFIER command turns off the amp enable output for a single amplifier, without changing anything else. Again, useful for spindle speed controllers.

2.8.8.1 Requirements

None. The command can be issued at any time, and will always be accepted.

2.8.8.2 Results

Currently, nothing. (A call to the old extAmpEnable function is currently commented out.) Eventually it will set the amp enable HAL pin false.

2.8.9  ACTIVATE_JOINT

The ACTIVATE_JOINT command turns on all the calculations associated with a single joint, but does not change the joint’s amp enable output pin.

2.8.9.1 Requirements

None. The command can be issued at any time, and will always be accepted.

2.8.9.2 Results

Calculations for the specified joint are enabled. The amp enable pin is not changed, however, any subsequent ENABLE or DISABLE commands will modify the joint’s amp enable pin.

2.8.10 DEACTIVATE_JOINT

The DEACTIVATE_JOINT command turns off all the calculations associated with a single joint, but does not change the joint’s amp enable output pin.

2.8.10.1 Requirements

None. The command can be issued at any time, and will always be accepted.
2.8.10.2 Results

Calculations for the specified joint are enabled. The amp enable pin is not changed, and subsequent ENABLE or DISABLE commands will not modify the joint’s amp enable pin.

2.8.11 ENABLE WATCHDOG

The ENABLE WATCHDOG command enables a hardware based watchdog (if present).

2.8.11.1 Requirements

None. The command can be issued at any time, and will always be accepted.

2.8.11.2 Results

Currently nothing. The old watchdog was a strange thing that used a specific sound card. A new watchdog interface may be designed in the future.

2.8.12 DISABLE WATCHDOG

The DISABLE WATCHDOG command disables a hardware based watchdog (if present).

2.8.12.1 Requirements

None. The command can be issued at any time, and will always be accepted.

2.8.12.2 Results

Currently nothing. The old watchdog was a strange thing that used a specific sound card. A new watchdog interface may be designed in the future.

2.8.13 PAUSE

The PAUSE command stops the trajectory planner. It has no effect in free or teleop mode. At this point I don’t know if it pauses all motion immediately, or if it completes the current move and then pauses before pulling another move from the queue.

2.8.13.1 Requirements

None. The command can be issued at any time, and will always be accepted.

2.8.13.2 Results

The trajectory planner pauses.

2.8.14 RESUME

The RESUME command restarts the trajectory planner if it is paused. It has no effect in free or teleop mode, or if the planner is not paused.
2.8.14.1 Requirements

None. The command can be issued at any time, and will always be accepted.

2.8.14.2 Results

The trajectory planner resumes.

2.8.15 STEP

The STEP command restarts the trajectory planner if it is paused, and tells the planner to stop again when it reaches a specific point. It has no effect in free or teleop mode. At this point I don’t know exactly how this works. I’ll add more documentation here when I dig deeper into the trajectory planner.

2.8.15.1 Requirements

None. The command can be issued at any time, and will always be accepted.

2.8.15.2 Results

The trajectory planner resumes, and later pauses when it reaches a specific point.

2.8.16 SCALE

The SCALE command scales all velocity limits and commands by a specified amount. It is used to implement feed rate override and other similar functions. The scaling works in free, teleop, and coord modes, and affects everything, including homing velocities, etc. However, individual joint velocity limits are unaffected.

2.8.16.1 Requirements

None. The command can be issued at any time, and will always be accepted.

2.8.16.2 Results

All velocity commands are scaled by the specified constant.

2.8.17 OVERRIDE_LIMITS

The OVERRIDE_LIMITS command prevents limits from tripping until the end of the next JOG command. It is normally used to allow a machine to be jogged off of a limit switch after tripping. (The command can actually be used to override limits, or to cancel a previous override.)

2.8.17.1 Requirements

None. The command can be issued at any time, and will always be accepted. (I think it should only work in free mode.)

2.8.17.2 Results

Limits on all joints are over-ridden until the end of the next JOG command. (This is currently broken… once an OVERRIDE_LIMITS command is received, limits are ignored until another OVERRIDE_LIMITS command re-enables them.)
2.8.18 HOME

The HOME command initiates a homing sequence on a specified joint. The actual homing sequence is determined by a number of configuration parameters, and can range from simply setting the current position to zero, to a multi-stage search for a home switch and index pulse, followed by a move to an arbitrary home location. For more information about the homing sequence, see the homing section of the Integrator Manual.

2.8.18.1 Requirements

The command will be ignored silently unless the machine is in free mode.

2.8.18.2 Results

Any jog or other joint motion is aborted, and the homing sequence starts.

2.8.19 JOG_CONT

The JOG_CONT command initiates a continuous jog on a single joint. A continuous jog is generated by setting the free mode trajectory planner’s target position to a point beyond the end of the joint’s range of travel. This ensures that the planner will move constantly until it is stopped by either the joint limits or an ABORT command. Normally, a GUI sends a JOG_CONT command when the user presses a jog button, and ABORT when the button is released.

2.8.19.1 Requirements

The command handler will reject the JOG_CONT command with an error message if machine is not in free mode, or if any joint is in motion (GET_MOTION_INPOS_FLAG() == FALSE), or if motion is not enabled. It will also silently ignore the command if the joint is already at or beyond its limit and the commanded jog would make it worse.

2.8.19.2 Results

The free mode trajectory planner for the joint identified by emcmotCommand->axis is activated, with a target position beyond the end of joint travel, and a velocity limit of emcmotCommand->vel. This starts the joint moving, and the move will continue until stopped by an ABORT command or by hitting a limit. The free mode planner accelerates at the joint accel limit at the beginning of the move, and will decelerate at the joint accel limit when it stops.

2.8.20 JOG_INCR

The JOG_INCR command initiates an incremental jog on a single joint. Incremental jogs are cumulative, in other words, issuing two JOG_INCR commands that each ask for 0.100 inches of movement will result in 0.200 inches of travel, even if the second command is issued before the first one finishes. Normally incremental jogs stop when they have traveled the desired distance, however they also stop when they hit a limit, or on an ABORT command.

2.8.20.1 Requirements

The command handler will silently reject the JOG_INCR command if machine is not in free mode, or if any joint is in motion (GET_MOTION_INPOS_FLAG() == FALSE), or if motion is not enabled. It will also silently ignore the command if the joint is already at or beyond its limit and the commanded jog would make it worse.
2.8.20.2 Results

The free mode trajectory planner for the joint identified by emcmotCommand->axis is activated, the target position is increment-decremented by emcmotCommand->offset, and the velocity limit is set to emcmotCommand->vel. The free mode trajectory planner will generate a smooth trapezoidal move from the present position to the target position. The planner can correctly handle changes in the target position that happen while the move is in progress, so multiple JOG_INCR commands can be issued in quick succession. The free mode planner accelerates at the joint accel limit at the beginning of the move, and will decelerate at the joint accel limit to stop at the target position.

2.8.21 JOG_ABS

The JOG_ABS command initiates an absolute jog on a single joint. An absolute jog is a simple move to a specific location, in joint coordinates. Normally absolute jogs stop when they reach the desired location, however they also stop when they hit a limit, or on an ABORT command.

2.8.21.1 Requirements

The command handler will silently reject the JOG_ABS command if machine is not in free mode, or if any joint is in motion (GET_MOTION_INPOS_FLAG() == FALSE), or if motion is not enabled. It will also silently ignore the command if the joint is already at or beyond its limit and the commanded jog would make it worse.

2.8.21.2 Results

The free mode trajectory planner for the joint identified by emcmotCommand->axis is activated, the target position is set to emcmotCommand->offset, and the velocity limit is set to emcmotCommand->vel. The free mode trajectory planner will generate a smooth trapezoidal move from the present position to the target position. The planner can correctly handle changes in the target position that happen while the move is in progress. If multiple JOG_ABS commands are issued in quick succession, each new command changes the target position and the machine goes to the final commanded position. The free mode planner accelerates at the joint accel limit at the beginning of the move, and will decelerate at the joint accel limit to stop at the target position.

2.8.22 SET_LINE

The SET_LINE command adds a straight line to the trajectory planner queue.

(More later)

2.8.23 SET_CIRCLE

The SET_CIRCLE command adds a circular move to the trajectory planner queue.

(More later)

2.8.24 SET_TELEOP_VECTOR

The SET_TELEOP_VECTOR command instructs the motion controller to move along a specific vector in Cartesian space.

(More later)

2.8.25 PROBE

The PROBE command instructs the motion controller to move toward a specific point in Cartesian space, stopping and recording its position if the probe input is triggered.

(More later)
2.8.26 CLEAR_PROBE_FLAG

The CLEAR_PROBE_FLAG command is used to reset the probe input in preparation for a PROBE command. (Question: why shouldn’t the PROBE command automatically reset the input?)

(More later)

2.8.27 SET_xix

There are approximately 15 SET_xxx commands, where xxx is the name of some configuration parameter. It is anticipated that there will be several more SET commands as more parameters are added. I would like to find a cleaner way of setting and reading configuration parameters. The existing methods require many lines of code to be added to multiple files each time a parameter is added. Much of that code is identical or nearly identical for every parameter.

2.9 Backlash and Screw Error Compensation

+  

2.10 Task controller (EMCTASK)

2.10.1 State

Task has three possible internal states: E-stop, E-stop Reset, and Machine On.

2.11 IO controller (EMCIO)

+  

[Diagram showing the state transitions of the task controller]
2.12 User Interfaces

2.13 libnml Introduction

libnml is derived from the NIST rcslib without all the multi-platform support. Many of the wrappers around platform specific code has been removed along with much of the code that is not required by LinuxCNC. It is hoped that sufficient compatibility remains with rcslib so that applications can be implemented on non-Linux platforms and still be able to communicate with LinuxCNC.

This chapter is not intended to be a definitive guide to using libnml (or rcslib), instead, it will eventually provide an overview of each C++ class and their member functions. Initially, most of these notes will be random comments added as the code scrutinized and modified.

2.14 LinkedList

Base class to maintain a linked list. This is one of the core building blocks used in passing NML messages and assorted internal data structures.

2.15 LinkedListNode

Base class for producing a linked list - Purpose, to hold pointers to the previous and next nodes, pointer to the data, and the size of the data.

No memory for data storage is allocated.

2.16 SharedMemory

Provides a block of shared memory along with a semaphore (inherited from the Semaphore class). Creation and destruction of the semaphore is handled by the SharedMemory constructor and destructor.

2.17 ShmBuffer

Class for passing NML messages between local processes using a shared memory buffer. Much of internal workings are inherited from the CMS class.

2.18 Timer

The Timer class provides a periodic timer limited only by the resolution of the system clock. If, for example, a process needs to be run every 5 seconds regardless of the time taken to run the process, the following code snippet demonstrates how:

```cpp
main()
{
    timer = new Timer(5.0);   /* Initialize a timer with a 5 second loop */
    while(0) {
        /* Do some process */
        timer.wait();           /* Wait till the next 5 second interval */
```
2.19 Semaphore

The Semaphore class provides a method of mutual exclusions for accessing a shared resource. The function to get a semaphore can either block until access is available, return after a timeout, or return immediately with or without gaining the semaphore. The constructor will create a semaphore or attach to an existing one if the ID is already in use.

The Semaphore::destroy() must be called by the last process only.

2.20 CMS

At the heart of libnml is the CMS class, it contains most of the functions used by libnml and ultimately NML. Many of the internal functions are overloaded to allow for specific hardware dependent methods of data passing. Ultimately, everything revolves around a central block of memory (referred to as the message buffer or just buffer). This buffer may exist as a shared memory block accessed by other CMS/NML processes, or a local and private buffer for data being transferred by network or serial interfaces.

The buffer is dynamically allocated at run time to allow for greater flexibility of the CMS/NML sub-system. The buffer size must be large enough to accommodate the largest message, a small amount for internal use and allow for the message to be encoded if this option is chosen (encoded data will be covered later). The following figure is an internal view of the buffer space.

```
Buffer Name [32 char]

CMS Header {
    Read,
    Message ID
    Message Size
}

Data Space
```

CMS buffer The CMS base class is primarily responsible for creating the communications pathways and interfacing to the O.S.

2.21 Configuration file format

NML configuration consists of two types of line formats. One for Buffers, and a second for Processes that connect to the buffers.
2.21.1 Buffer line

The original NIST format of the buffer line is:

- \textit{B name type host size neut RPC# buffer# max_procs key [type specific configs]}
- B - identifies this line as a Buffer configuration.
- \textit{name} - is the identifier of the buffer.
- \textit{type} - describes the buffer type - SHMEM, LOCMEM, FILEMEM, PHANTOM, or GLOBMEM.
- \textit{host} - is either an IP address or host name for the NML server
- \textit{size} - is the size of the buffer
- \textit{neut} - a boolean to indicate if the data in the buffer is encoded in a machine independent format, or raw.
- \textit{RPC#} - Obsolete - Place holder retained for backward compatibility only.
- \textit{buffer#} - A unique ID number used if a server controls multiple buffers.
- \textit{max_procs} - is the maximum processes allowed to connect to this buffer.
- \textit{key} - is a numerical identifier for a shared memory buffer

2.21.2 Type specific configs

The buffer type implies additional configuration options whilst the host operating system precludes certain combinations. In an attempt to distill published documentation into a coherent format, only the \textbf{SHMEM} buffer type will be covered.

- \textit{mutex=\textit{os\_sem}} - default mode for providing semaphore locking of the buffer memory.
- \textit{mutex=\textit{none}} - Not used
- \textit{mutex=\textit{no\_interrupts}} - not applicable on a Linux system
- \textit{mutex=\textit{no\_switching}} - not applicable on a Linux system
- \textit{mutex=\textit{mao\_split}} - Splits the buffer into half (or more) and allows one process to access part of the buffer whilst a second process is writing to another part.
- \textit{TCP=(\textit{port number})} - Specifies which network port to use.
- \textit{UDP=(\textit{port number})} - ditto
- \textit{STCP=(\textit{port number})} - ditto
- \textit{serialPortDevName=(\textit{serial port})} - Undocumented.
- \textit{passwd=file\_name.pwd} - Adds a layer of security to the buffer by requiring each process to provide a password.
- \textit{bsem} - NIST documentation implies a key for a blocking semaphore, and if bsem=-1, blocking reads are prevented.
- \textit{queue} - Enables queued message passing.
- \textit{ascii} - Encode messages in a plain text format
- \textit{disp} - Encode messages in a format suitable for display (???)
- \textit{xdr} - Encode messages in External Data Representation. (see rpc/xdr.h for details).
- \textit{diag} - Enables diagnostics stored in the buffer (timings and byte counts ?)
2.21.3 Process line

The original NIST format of the process line is:

\texttt{P name buffer type host ops server timeout master c_num [type specific configs]}

- \texttt{P} - identifies this line as a Process configuration.
- \texttt{name} - is the identifier of the process.
- \texttt{buffer} - is one of the buffers defined elsewhere in the config file.
- \texttt{type} - defines whether this process is local or remote relative to the buffer.
- \texttt{host} - specifies where on the network this process is running.
- \texttt{ops} - gives the process read only, write only, or read/write access to the buffer.
- \texttt{server} - specifies if this process will running a server for this buffer.
- \texttt{timeout} - sets the timeout characteristics for accesses to the buffer.
- \texttt{master} - indicates if this process is responsible for creating and destroying the buffer.
- \texttt{c_num} - an integer between zero and (max_procs -1)

2.21.4 Configuration Comments

Some of the configuration combinations are invalid, whilst others imply certain constraints. On a Linux system, GLOBMEM is obsolete, whilst PHANTOM is only really useful in the testing stage of an application, likewise for FILEMEM. LOCMEM is of little use for a multi-process application, and only offers limited performance advantages over SHMEM. This leaves SHMEM as the only buffer type to use with LinuxCNC.

The neut option is only of use in a multi-processor system where different (and incompatible) architectures are sharing a block of memory. The likelihood of seeing a system of this type outside of a museum or research establishment is remote and is only relevant to GLOBMEM buffers.

The RPC number is documented as being obsolete and is retained only for compatibility reasons.

With a unique buffer name, having a numerical identity seems to be pointless. Need to review the code to identify the logic. Likewise, the key field at first appears to be redundant, and it could be derived from the buffer name.

The purpose of limiting the number of processes allowed to connect to any one buffer is unclear from existing documentation and from the original source code. Allowing unspecified multiple processes to connect to a buffer is no more difficult to implement.

The mutex types boil down to one of two, the default "os_sem" or "mao split". Most of the NML messages are relatively short and can be copied to or from the buffer with minimal delays, so split reads are not essential.

Data encoding is only relevant when transmitted to a remote process - Using TCP or UDP implies XDR encoding. Whilst ASCII encoding may have some use in diagnostics or for passing data to an embedded system that does not implement NML.

UDP protocols have fewer checks on data and allows a percentage of packets to be dropped. TCP is more reliable, but is marginally slower.

If LinuxCNC is to be connected to a network, one would hope that it is local and behind a firewall. About the only reason to allow access to LinuxCNC via the Internet would be for remote diagnostics - This can be achieved far more securely using other means, perhaps by a web interface.

The exact behavior when timeout is set to zero or a negative value is unclear from the NIST documents. Only INF and positive values are mentioned. However, buried in the source code of rcslib, it is apparent that the following applies:

- \texttt{timeout > 0} Blocking access until the timeout interval is reached or access to the buffer is available.
- \texttt{timeout = 0} Access to the buffer is only possible if no other process is reading or writing at the time.
- \texttt{timeout < 0} or \texttt{INF} Access is blocked until the buffer is available.
2.22 NML base class

Expand on the lists and the relationship between NML, NMLmsg, and the lower level cms classes.

Not to be confused with NMLmsg, RCS_STAT_MSG, or RCS_CMD_MSG.

NML is responsible for parsing the config file, configuring the cms buffers and is the mechanism for routing messages to the correct buffer(s). To do this, NML creates several lists for:

- cms buffers created or connected to.
- processes and the buffers they connect to
- a long list of format functions for each message type

This last item is probably the nub of much of the malignment of libnml/rcslib and NML in general. Each message that is passed via NML requires a certain amount of information to be attached in addition to the actual data. To do this, several formatting functions are called in sequence to assemble fragments of the overall message. The format functions will include NML_TYPE, MSG_TYPE, in addition to the data declared in derived NMLmsg classes. Changes to the order in which the formatting functions are called and also the variables passed will break compatibility with rcslib if messed with - There are reasons for maintaining rcslib compatibility, and good reasons for messing with the code. The question is, which set of reasons are overriding?

2.22.1 NML internals

2.22.1.1 NML constructor

NML::NML() parses the config file and stores it in a linked list to be passed to cms constructors in single lines. It is the function of the NML constructor to call the relevant cms constructor for each buffer and maintain a list of the cms objects and the processes associated with each buffer.

It is from the pointers stored in the lists that NML can interact with cms and why Doxygen fails to show the real relationships involved.

Note

The config is stored in memory before passing a pointer to a specific line to the cms constructor. The cms constructor then parses the line again to extract a couple of variables... It would make more sense to do ALL the parsing and save the variables in a struct that is passed to the cms constructor - This would eliminate string handling and reduce duplicate code in cms... .

2.22.1.2 NML read/write

Calls to NML::read and NML::write both perform similar tasks in so much as processing the message - The only real variation is in the direction of data flow.

A call to the read function first gets data from the buffer, then calls format_output(), whilst a write function would call format_input() before passing the data to the buffer. It is in format_xxx() that the work of constructing or deconstructing the message takes place. A list of assorted functions are called in turn to place various parts of the NML header (not to be confused with the cms header) in the right order - The last function called is emcFormat() in emc.cc.

2.22.1.3 NMLmsg and NML relationships

NMLmsg is the base class from which all message classes are derived. Each message class must have a unique ID defined (and passed to the constructor) and also an update(*cms) function. The update() will be called by the NML read/write functions when the NML formatter is called - The pointer to the formatter will have been declared in the NML constructor at some point. By virtue of the linked lists NML creates, it is able to select cms pointer that is passed to the formatter and therefor which buffer is to be used.
2.23 Adding custom NML commands

LinuxCNC is pretty awesome, but some parts need some tweaking. As you know communication is done through NML channels, the data sent through such a channel is one of the classes defined in emc.hh (implemented in emc.cc). If somebody needs a message type that doesn’t exist, he should follow these steps to add a new one. (The Message I added in the example is called EMC_IO_GENERIC (inherits EMC_IO_CMD_MSG (inherits RCS_CMD_MSG)))

1. add the definition of the EMC_IO_GENERIC class to emc2/src/emc/nml_intf/emc.hh
2. add the type define: #define EMC_IO_GENERIC_TYPE ((NMLTYPE) 1605)
   a. (I chose 1605, because it was available) to emc2/src/emc/nml_intf/emc.hh
3. add case EMC_IO_GENERIC_TYPE to emcFormat in emc2/src/emc/nml_intf/emc.cc
4. add case EMC_IO_GENERIC_TYPE to emc_symbol_lookup in emc2/src/emc/nml_intf/emc.cc
5. add EMC_IO_GENERIC::update function to emc2/src/emc/nml_intf/emc.cc

Recompile, and the new message should be there. The next part is to send such messages from somewhere, and receive them in another place, and do some stuff with it.

2.24 The Tool Table and Toolchanger

LinuxCNC interfaces with toolchanger hardware, and has an internal toolchanger abstraction. LinuxCNC manages tool information in a tool table file.

2.24.1 Toolchanger abstraction in LinuxCNC

LinuxCNC supports two kinds of toolchanger hardware, called nonrandom and random. The ini setting [EMCIO]RANDOM_TOOLCHANGER controls which of these kinds of hardware LinuxCNC thinks it’s connected to.

2.24.1.1 Nonrandom Toolchangers

Nonrandom toolchanger hardware puts each tool back in the pocket it was originally loaded from. Examples of nonrandom toolchanger hardware are the "manual" toolchanger, lathe tool turrets, and rack toolchangers.

When configured for a nonrandom toolchanger, LinuxCNC does not change the pocket number in the tool table file as tools are loaded and unloaded. Internal to LinuxCNC, on tool change the tool information is copied from the tool table’s source pocket to pocket 0 (which represents the spindle), replacing whatever tool information was previously there.

Note

In LinuxCNC configured for nonrandom toolchanger, tool 0 (T0) has special meaning: "no tool". T0 may not appear in the tool table file, and changing to T0 will result in LinuxCNC thinking it’s got an empty spindle.
2.24.1.2 Random Toolchangers

Random toolchanger hardware swaps the tool in the spindle (if any) with the requested tool on tool change. Thus the pocket that a tool resides in changes as it is swapped in and out of the spindle.

An example of random toolchanger hardware is a carousel toolchanger.

When configured for a random toolchanger, LinuxCNC swaps the pocket number of the old and the new tool in the tool table file when tools are loaded. Internal to LinuxCNC, on tool change, the tool information is swapped between the tool table’s source pocket and pocket 0 (which represents the spindle). So after a tool change, pocket 0 in the tool table has the tool information for the new tool, and the pocket that the new tool came from has the tool information for the old tool (the tool that was in the spindle before the tool change), if any.

**Note**
In LinuxCNC configured for random toolchanger, tool 0 (T0) has no special meaning. It is treated exactly like any other tool in the tool table. It is customary to use T0 to represent “no tool” (ie, a tool with zero TLO), so that the spindle can be conveniently emptied when needed.

2.24.2 The Tool Table

LinuxCNC keeps track of tools in a file called the tool table. The tool table records the following information for each tool:

**tool number**
An integer that uniquely identifies this tool. Tool numbers are handled differently by LinuxCNC when configured for random and nonrandom toolchangers:

- When LinuxCNC is configured for a nonrandom toolchanger this number must be positive. T0 gets special handling and is not allowed to appear in the tool table.
- When LinuxCNC is configured for a random toolchanger this number must be non-negative. T0 is allowed in the tool table, and is usually used to represent "no tool", ie the empty pocket.

**pocket number**
An integer that identifies the pocket or slot in the toolchanger hardware where the tool resides. Pocket numbers are handled differently by LinuxCNC when configured for random and nonrandom toolchangers:

- When LinuxCNC is configured for a nonrandom toolchanger, the pocket number in the tool file can be any positive integer (pocket 0 is not allowed). LinuxCNC silently compactifies the pocket numbers when it loads the tool file, so there may be a difference between the pocket numbers in the tool file and the internal pocket numbers used by LinuxCNC-with-nonrandom-toolchanger.
- When LinuxCNC is configured for a random toolchanger, the pocket numbers in the tool file must be between 0 and 55, inclusive. Pockets 1-55 are in the toolchanger, pocket 0 is the spindle.

**diameter**
Diameter of the tool, in machine units.

**tool length offset**
Tool length offset (also called TLO), in up to 9 axes, in machine units. Axes that don’t have a specified TLO get 0.

2.24.3 Gcodes affecting tools

The gcodes that use or affect tool information are:
2.24.3.1 Txxx

Tells the toolchanger hardware to prepare to switch to a specified tool xxx.
Handled by Interp::convert_tool_select().

1. The machine is asked to prepare to switch to the selected tool by calling the Canon function SELECT_POCKET() with the pocket number of the requested tool.
   a. (saicanon) No-op.
   b. (emccanon) Builds an EMC_TOOL_PREPARE message with the requested pocket number and sends it to Task, which sends it on to IO. IO gets the message and asks HAL to prepare the pocket by setting iocontrol.0.tool-prep-pocket, iocontrol.0.tool-prep-number, and iocontrol.0.tool-prep-prepare. IO then repeatedly calls read_tool_inputs() to poll the HAL pin iocontrol.0.tool-prep-prepare, which signals from the toolchanger hardware, via HAL, to IO that the requested tool prep is complete. When that pin goes True, IO sets emcioStatus.tool.pocketPrepped to the requested tool’s pocket number.

2. Back in interp, settings->selected_pocket is assigned the pocket number of the requested tool xxx.

2.24.3.2 M6

Tells the toolchanger to switch to the currently selected tool (selected by the previous Txxx command).
Handled by Interp::convert_tool_change().

1. The machine is asked to change to the selected tool by calling the Canon function CHANGE_TOOL() with settings->selected_pocket.
   a. (saicanon) Sets sai’s _active_slot to the passed-in pocket number. Tool information is copied from the selected pocket of the tool table (ie, from sai’s _tools[_active_slot]) to the spindle (aka sai’s _tools[0]).
   b. (emccanon) Sends an EMC_TOOL_LOAD message to Task, which sends it to IO. IO sets emcioStatus.tool.toolInSpindle to the tool number of the tool in the pocket identified by emcioStatus.tool.pocketPrepped (set by Txxx aka SELECT_POCKET()). It then requests that the toolchanger hardware perform a tool change, by setting the HAL pin iocontrol.0.tool-change to True. Later, IO’s read_tool_inputs() will sense that the HAL pin iocontrol.0.tool_changed has been set to True, indicating the toolchanger has completed the tool change. When this happens, it calls load_tool() to update the machine state.
      i. load_tool() with a nonrandom toolchanger config copies the tool information from the selected pocket to the spindle (pocket 0).
      ii. load_tool() with a random toolchanger config swaps tool information between pocket 0 (the spindle) and the selected pocket, then saves the tool table.

2. Back in interp, settings->current_pocket is assigned the new tool from settings->selected_pocket (set by Txxx). The relevant numbered parameters (#5400-#5413) are updated with the new tool information from pocket 0 (spindle).

2.24.3.3 G43/G43.1/G49

Apply tool length offset. G43 uses the TLO of the currently loaded tool, or of a specified tool if the H-word is given in the block. G43.1 gets TLO from axis-words in the block. G49 cancels the TLO (it uses 0 for the offset for all axes).
Handled by Interp::convert_tool_length_offset().

1. It starts by building an EmcPose containing the 9-axis offsets to use. For G43.1, these tool offsets come from axis words in the current block. For G43 these offsets come from the current tool (the tool in pocket 0), or from the tool specified by the H-word in the block. For G49, the offsets are all 0.

2. The offsets are passed to Canon’s USE_TOOL_LENGTH_OFFSET() function.
a. (saicanon) Records the TLO in _tool_offset.

b. (emccanon) Builds an EMC_TRAJ_SET_OFFSET message containing the offsets and sends it to Task. Task copies the offsets to emcStatus->task.toolOffset and sends them on to Motion via an EMC_MOT_SET_OFFSET command. Motion copies the offsets to emcmotStatus->tool_offset, where it gets used to offset future motions.

3. Back in interp, the offsets are recorded in settings->tool_offset. The effective pocket is recorded in settings->tool_offset_index, though this value is never used.

2.24.3.4 G10 L1/L10/L11

Modifies the tool table.

Handled by Interp::convert_setup_tool().

1. Picks the tool number out of the P-word in the block and finds the pocket for that tool:

   a. With a nonrandom toolchanger config this is always the pocket number in the toolchanger (even when the tool is in the spindle).

   b. With a random toolchanger config, if the tool is currently loaded it uses pocket 0 (pocket 0 means "the spindle"), and if the tool is not loaded it uses the pocket number in the tool changer. (This difference is important.)

2. Figures out what the new offsets should be.

3. The new tool information (diameter, offsets, angles, and orientation), along with the tool number and pocket number, are passed to the Canon call SET_TOOL_TABLE_ENTRY().

   a. (saicanon) Copy the new tool information to the specified pocket (in sai’s internal tool table, _tools).

   b. (emccanon) Build an EMC_TOOL_SET_OFFSET message with the new tool information, and send it to Task, which passes it to IO. IO updates the specified pocket in its internal copy of the tool table (emcioStatus.tool.toolTable), and if the specified tool is currently loaded (it is compared to emcioStatus.tool.toolInSpindle) then the new tool information is copied to pocket 0 (the spindle) as well. (FIXME: that’s a buglet, should only be copied on nonrandom machines.) Finally IO saves the new tool table.

4. Back in interp, if the modified tool is currently loaded in the spindle, and if the machine is a non-random toolchanger, then the new tool information is copied from the tool’s home pocket to pocket 0 (the spindle) in interp’s copy of the tool table, settings->tool_table. (This copy is not needed on random toolchanger machines because there, tools don’t have a home pocket and instead we just updated the tool in pocket 0 directly.)

5. The relevant numbered parameters (#5400-#5413) are updated from the tool information in the spindle (by copying the information from interp’s settings->tool_table to settings->parameters). (FIXME: this is a buglet, the params should only be updated if it was the current tool that was modified).

6. If the modified tool is currently loaded in the spindle, and if the config is for a nonrandom toolchanger, then the new tool information is written to the tool table’s pocket 0 as well, via a second call to SET_TOOL_TABLE_ENTRY(). (This second tool-table update is not needed on random toolchanger machines because there, tools don’t have a home pocket and instead we just updated the tool in pocket 0 directly.)

2.24.3.5 M61

Set current tool number. This switches LinuxCNC’s internal representation of which tool is in the spindle, without actually moving the toolchanger or swapping any tools.

Handled by Interp::convert_tool_change().

Canon: CHANGE_TOOL_NUMBER()

settings->current_pocket is assigned the pocket number currently holding the tool specified by the Q-word argument.
2.24.3.6 **G41/G41.1/G42/G42.1**

Enable cutter radius compensation (usually called *cutter comp*).

**Handled by** `Interp::convert_cutter_compensation_on()`.

No Canon call, cutter comp happens in the interpreter. Uses the tool table in the expected way: if a D-word tool number is supplied it looks up the pocket number of the specified tool number in the table, and if no D-word is supplied it uses pocket 0 (the spindle).

2.24.3.7 **G40**

Cancel cutter radius compensation.

**Handled by** `Interp::convert_cutter_compensation_off()`.

No Canon call, cutter comp happens in the interpreter. Does not use the tool table.

### 2.24.4 Internal state variables

This is not an exhaustive list! Tool information is spread throughout LinuxCNC.

#### 2.24.4.1 IO

`emcioStatus` is of type `EMC_IO_STAT`

- `emcioStatus.tool.pocketPrepped`
  When IO gets the signal from HAL that the toolchanger prep is complete (after a `Txxx` command), this variable is set to the pocket of the requested tool. When IO gets the signal from HAL that the tool change itself is complete (after an `M6` command), this variable gets reset to -1.

- `emcioStatus.tool.toolInSpindle`
  Tool number of the tool currently installed in the spindle. Exported on the HAL pin `iocontrol.0.tool-number` (s32).

- `emcioStatus.tool.toolTable[]`
  An array of `CANON_TOOL_TABLE` structures, `CANON_POCKETS_MAX` long. Loaded from the tool table file at startup and maintained there after. Index 0 is the spindle, indexes 1-(CANON_POCKETS_MAX-1) are the pockets in the toolchanger. This is a complete copy of the tool information, maintained separately from Interp’s `settings.tool_table`.

#### 2.24.4.2 interp

`settings` is of type `settings`, which is struct `setup_struct`. Defined in `src/emc/rs274ngc/interp_internal.hh`.

- `settings.selected_pocket`
  Pocket of the tool most recently selected by `Txxx`.

- `settings.current_pocket`
  Original pocket of the tool currently in the spindle. In other words: which toolchanger pocket the tool that’s currently in the spindle was loaded from.

- `settings.tool_table[]`
  An array of tool information. The index into the array is the "pocket number" (aka "slot number"). Pocket 0 is the spindle, pockets 1 through (CANON_POCKETS_MAX-1) are the pockets of the toolchanger.
settings.tool_offset_index
Unused. FIXME: Should probably be removed.

settings.toolchange_flag
Interp sets this to true when calling Canon’s CHANGE_TOOL() function. It is checked in Interp::convert_tool_length_offset() to decide which pocket to use for G43 (with no H-word): settings->current_pocket if the tool change is still in progress, pocket 0 (the spindle) if the tool change is complete.

settings.random_toolchanger
Set from the ini variable [EMCIO]RANDOM_TOOLCHANGER at startup. Controls various tool table handling logic. (IO also reads this ini variable and changes its behavior based on it. For example, when saving the tool table, random toolchanger save the tool in the spindle (pocket 0), but non-random toolchanger save each tool in its “home pocket”.)

settings.tool_offset
This is an EmcPose variable.

• Used to compute position in various places.
• Sent to Motion via the EMCMOT_SET_OFFSET message. All motion does with the offsets is export them to the HAL pins motion.0.tooloffset.[xyzabcuvw]. FIXME: export these from someplace closer to the tool table (io or interp, probably) and remove the EMCMOT_SET_OFFSET message.

settings.pockets_max
Used interchangably with CANON_POCKETS_MAX (a #defined constant, set to 56 as of 2012 December 30). FIXME: This settings variable is not currently useful and should probably be removed.

settings.tool_table
This is an array of CANON_TOOL_TABLE structures (defined in src/emc/nml_intf/emctool.h), with CANON_POCKETS_MAX entries. Indexed by “pocket number”, aka “slot number”. Index 0 is the spindle, indexes 1-(CANON_POCKETS_MAX-1) are the pockets in the tool changer. On a random toolchanger pocket numbers are meaningful. On a nonrandom toolchanger pockets are meaningless; the pocket numbers in the tool table file are ignored and tools are assigned to tool_table slots sequentially.

settings.tool_change_at_g30, settings.tool_change_quill_up, settings.tool_change_with_spindle_on
These are set from ini variables in the [EMCIO] section, and control how tool changes are performed.
Chapter 3

NML Messages

3.1 EMC OPERATOR

3.1.1 EMC_OPERATOR_ERROR_TYPE
Description, NML Type: textual error message to the operator, 11
Written From: emccanon.cc, iosh.cc
Read To: emctaskmain.cc, emcsh.cc
Parameter, Type: [error, char[LINELEN]]

3.1.2 EMC_OPERATOR_TEXT_TYPE
Description, NML Type: textual information message to the operator, 12
Written From: emctaskmain.cc
Read To: emctaskmain.cc, emcsh.cc
Parameter, Type: [text, char[LINELEN]]

3.1.3 EMC_OPERATOR_DISPLAY_TYPE
Description, NML Type: URL or filename of a document to display, 13
Obs: not used, only read
Written From: none
Read To: emctaskmain.cc, emcsh.cc
Parameter, Type: [display, char[LINELEN]]

3.2 EMC NULL, SET, DEBUG, & SYSTEM

3.2.1 EMC_NULL_TYPE
Description, NML Type: used to reset serial number to original, 21
Written From: thisQuit (emcsh.cc)
Read To: emctaskmain.cc
Parameter, Type: none
3.2.2 EMC_SET_DEBUG_TYPE
Description, NML Type: sets debug level, 22
Written From: emcIoSetDebug (iotaskintf.cc), sendDebug (emcsh.cc)
Read To: emctaskmain.cc, ioControl.cc
Parameter, Type: [debug, int]

3.2.3 EMC_SYSTEM_CMD_TYPE
Description, NML Type: execute a system command, 30
Written From: user_defined_add_m_code (emctask.cc)
Read To: emcSystemCmd (emctaskmain.cc)
Parameter, Type: [string, char[LINELEN]]

3.3 EMC AXIS

3.3.1 EMC_AXIS_SET_AXIS_TYPE
Description, NML Type: axis type to linear or angular, 101
Obs: not used
Written From: none
Read To: none
Parameter, Type: [axis (in EMC_AXIS_CMD_MSG), int] [axisType, unsigned char]

3.3.2 EMC_AXIS_SET_UNITS_TYPE
Description, NML Type: units conversion factor, 102
Obs: not used
Written From: none
Read To: none
Parameter, Type: [axis (in EMC_AXIS_CMD_MSG), int] [units, double]

3.3.3 EMC_AXIS_SET_GAINS_TYPE
Description, NML Type: Set the PID gains, 103
Obs: currently not used in EMC2, needs to go to HAL
Written From: none
Read To: emctaskmain.cc
Parameter, Type: [axis (in EMC_AXIS_CMD_MSG), int] [p, double] [i, double] [d, double] [ff0, double] [ff1, double] [ff2, double] [backlash, double] [bias, double] [maxError, double]

3.3.4 EMC_AXIS_SET_CYCLE_TIME_TYPE
Description, NML Type: cycle time for the servo task, 104
Written From: none
Read To: emctaskmain.cc
Parameter, Type: [axis (in EMC_AXIS_CMD_MSG), int] [cycleTime, double]
3.3.5 EMC_AXIS_SET_INPUT_SCALE_TYPE
Description. NML Type: scale factor and offset for the position input, 105
Obs: currently if 0’ed, used only directly from iniaxis
Written From: none Read To: emcTaskIssueCommand (emctaskmain.cc) calls emcAxisSetInputScale (minimillbridgeporttaskintf.cc) which sends EMCMOT_SET_INPUT_SCALE
Parameter, Type: [axis (in EMC_AXIS_CMD_MSG), int] [scale, double] [offset, double]

3.3.6 EMC_AXIS_SET_OUTPUT_SCALE_TYPE
Description. NML Type: scale factor and offset for the position output, 106
Obs: currently if 0’ed, used only directly from iniaxis
Written From: none
Read To: emcTaskIssueCommand (emctaskmain.cc)
Parameter, Type: [axis (in EMC_AXIS_CMD_MSG), int] [scale, double] [offset, double]

3.3.7 EMC_AXIS_SET_MIN_POSITION_LIMIT_TYPE
Description. NML Type: sets min limit, 107
Obs: also handled by iniaxis which directly calls emcAxisSetMinPositionLimit
Written From: none Read To: emcTaskIssueCommand (emctaskmain.cc) calls emcAxisSetMinPositionLimit (taskintf.cc) which sends EMCMOT_SET_POSITION_LIMITS
Parameter, Type: [axis (in EMC_AXIS_CMD_MSG), int] [limit, double]

3.3.8 EMC_AXIS_SET_MAX_POSITION_LIMIT_TYPE
Description. NML Type: sets max limit, 108
Obs: also handled by iniaxis which directly calls emcAxisSetMaxPositionLimit
Written From: none Read To: emcTaskIssueCommand (emctaskmain.cc) calls emcAxisSetMaxPositionLimit (taskintf.cc) which sends EMCMOT_SET_POSITION_LIMITS
Parameter, Type: [axis (in EMC_AXIS_CMD_MSG), int] [limit, double]

3.3.9 EMC_AXIS_SET_MIN_OUTPUT_LIMIT_TYPE
Description. NML Type: -, 109
Obs: not used
Written From: none
Read To: none
Parameter, Type: [axis (in EMC_AXIS_CMD_MSG), int] [limit, double]

3.3.10 EMC_AXIS_SET_MAX_OUTPUT_LIMIT_TYPE
Description. NML Type: -, 110
Obs: not used
Written From: none
Read To: none
Parameter, Type: [axis (in EMC_AXIS_CMD_MSG), int] [limit, double]
3.3.11 **EMC_AXIS_SET_FERROR_TYPE**

Description, NML Type: sets max following error, 111

Obs: also handled by iniaxis which directly calls emcAxisSetFerror

Written From: none Read To: emcTaskIssueCommand (emctaskmain.cc) calls emcAxisSetFerror (taskintf.cc) which sends EMCMOT_SET_MAX_FERROR

Parameter, Type: [axis (in EMC_AXIS_CMD_MSG), int] [ferror, double]

3.3.12 **EMC_AXIS_SET_HOMING_VEL_TYPE**

Description, NML Type: -, 112

Obs: in EMC2 those are SET_HOMING_PARAMS double home, double offset, double search_vel, double latch_vel, int use_index, int ignore_limits,

Written From: none

Read To: none

Parameter, Type: [axis (in EMC_AXIS_CMD_MSG), int] [ferror, double]

3.3.13 **EMC_AXIS_SET_HOME_TYPE**

Description, NML Type: -, 113

Written From: none

Read To: none

Parameter, Type: [axis (in EMC_AXIS_CMD_MSG), int] [homingVel, double]

3.3.14 **EMC_AXIS_SET_HOME_OFFSET_TYPE**

Description, NML Type: -, 114

Written From: none

Read To: none

Parameter, Type: [axis (in EMC_AXIS_CMD_MSG), int] [home, double]

3.3.15 **EMC_AXIS_SET_MIN_FERROR_TYPE**

Description, NML Type: sets min following error, 115

Obs: also handled by iniaxis which directly calls emcAxisSetMinFerror

Written From: none

Read To: emcTaskIssueCommand (emctaskmain.cc) calls emcAxisSetMinFerror (taskintf.cc) which sends EMCMOT_SET_MIN_FERROR

Parameter, Type: [axis (in EMC_AXIS_CMD_MSG), int] [offset, double]
3.3.16 **EMC_AXIS_SET_MAX_VELOCITY_TYPE**

Description, NML Type: sets max. velocity, 116
Obs: not used
Written From: none
Read To: none
Parameter, Type: [axis (in EMC_AXIS_CMD_MSG), int] [vel, double]

3.3.17 **EMC_AXIS_INIT_TYPE**

Description, NML Type: -, 118
Obs: not used
Written From: none
Read To: none
Parameter, Type: [axis (in EMC_AXIS_CMD_MSG), int]

3.3.18 **EMC_AXIS_HALT_TYPE**

Description, NML Type: -, 119
Obs: not used, only read
Written From: none
Read To: emcTaskIssueCommand (emctaskmain.cc)
calls emcAxisHalt (taskintf.cc)
Parameter, Type: [axis (in EMC_AXIS_CMD_MSG), int]

3.3.19 **EMC_AXIS_ABORT_TYPE**

Description, NML Type: aborts motion on an axis (e.g. GUI jogs), 120
Obs: used from the GUI when stopping a manual jog
Written From: sendJogStop (emcsh.cc)
Read To: emcTaskIssueCommand (emctaskmain.cc)
calls emcAxisAbort (taskintf.cc)
which sends EMCMOT_AXIS_ABORT
Parameter, Type: [axis (in EMC_AXIS_CMD_MSG), int]

3.3.20 **EMC_AXIS_ENABLE_TYPE**

Description, NML Type: enables axis, 121
Obs: not used from tkemc & mini
Written From: sendAxisEnable (emcsh.cc)
Read To: emcTaskIssueCommand (emctaskmain.cc)
calls emcAxisEnable (taskintf.cc)
which sends EMCMOT_ENABLE_AMPLIFIER
Parameter, Type: [axis (in EMC_AXIS_CMD_MSG), int]
3.3.21 EMC_AXIS_DISABLE_TYPE

Description, NML Type: disable axis, 122
Obs: not used from tkemc & mini
Written From: sendAxisDisable (emcsh.cc)
Read To: emcTaskIssueCommand (emctaskmain.cc)
calls emcAxisDisable (taskintf.cc)
which sends EMCMOT_DISABLE_AMPLIFIER
Parameter, Type: [axis (in EMC_AXIS_CMD_MSG), int]

3.3.22 EMC_AXIS_HOME_TYPE

Description, NML Type: home an axis at current position, 123
Obs: used from tkemc & mini through emc_home
Written From: sendHome (emcsh.cc)
Read To: emcTaskIssueCommand (emctaskmain.cc)
calls emcAxisHome (taskintf.cc)
which sends EMCMOT_HOME
Parameter, Type: [axis (in EMC_AXIS_CMD_MSG), int]

3.3.23 EMC_AXIS_JOG_TYPE

Description, NML Type: jogs an axis continuously, 124
Obs: used on jogging
Written From: sendJogCont (emcsh.cc)
Read To: emcTaskIssueCommand (emctaskmain.cc)
calls emcAxisJog (taskintf.cc)
which sends EMCMOT_JOG_CONT
Parameter, Type: [axis (in EMC_AXIS_CMD_MSG), int] [vel, double]

3.3.24 EMC_AXIS_INCR_JOG_TYPE

Description, NML Type: jogs an axis with an increment, 125
Obs: used on jogging
Written From: sendJogIncr (emcsh.cc)
Read To: emcTaskIssueCommand (emctaskmain.cc)
calls emcAxisIncrJog (taskintf.cc)
which sends EMCMOT_JOG_INCR
Parameter, Type: [axis (in EMC_AXIS_CMD_MSG), int] incr, double] [vel, double]

3.3.25 EMC_AXIS_ABS_JOG_TYPE

Description, NML Type: jogs an axis with an absolute value, 126
Obs: not used, only read
Written From: none
Read To: emcTaskIssueCommand (emctaskmain.cc)
calls emcAxisAbsJog (taskintf.cc)
which sends EMCMOT_JOG_ABS Parameter, Type: [axis (in EMC_AXIS_CMD_MSG), int] [pos, double] [vel, double]
3.3.26 EMC_AXIS_ACTIVATE_TYPE

Description, NML Type: -, 127
Obs: not used
Written From: none
Read To: none
Parameter, Type: [axis (in EMC_AXIS_CMD_MSG), int]

3.3.27 EMC_AXIS_DEACTIVATE_TYPE

Description, NML Type: -, 128
Obs: not used
Written From: none
Read To: none
Parameter, Type: [axis (in EMC_AXIS_CMD_MSG), int]

3.3.28 EMC_AXIS_OVERRIDE_LIMITS_TYPE

Description, NML Type: overrides min/max limits during homing, 129
Obs: used from tkemc & mini through emc_override_limit
Written From: sendOverrideLimits (emcsh.cc)
Read To: emcTaskIssueCommand (emctaskmain.cc)
calls emcAxisOverrideLimits (taskintf.cc)
which sends EMCMOT_OVERRIDE_LIMITS
Parameter, Type: [axis (in EMC_AXIS_CMD_MSG), int]

3.3.29 EMC_AXIS_SET_OUTPUT_TYPE

Description, NML Type: sets an DAC output value, 130
Obs: currently not used in EMC2, needs to go to HAL
Written From: sendAxisSetOutput (emcsh.cc)
Read To: emcTaskIssueCommand (emctaskmain.cc)
calls emcAxisSetOutput (taskintf.cc)
which sends EMCMOT_DAC_OUT
Parameter, Type: [axis (in EMC_AXIS_CMD_MSG), int] [output, double]

3.3.30 EMC_AXIS_LOAD_COMP_TYPE

Description, NML Type: loads compensation values from a file, 131
Obs: currently usrmotLoadComp if 0'ed in EMC2
Written From: sendAxisLoadComp (emcsh.cc)
Read To: emcTaskIssueCommand (emctaskmain.cc)
calls emcAxisLoadComp (minimillbridgeporttaskintf.cc)
which calls usrmotLoadComp
Parameter, Type: [axis (in EMC_AXIS_CMD_MSG), int] [file, char[LINELEN]]
3.3.31 **EMC_AXIS_ALTER_TYPE**

Description, NML Type: loads the alter value to modify the axis position, 132
Written From: sendAxisAlter (emcsh.cc)
Read To: emcTaskIssueCommand (emctaskmain.cc)
calls emcAxisAlter (taskintf.cc)
which calls usrmotAlter

Parameter, Type: [axis (in EMC_AXIS_CMD_MSG), int] [alter, double]

3.3.32 **EMC_AXIS_SET_STEP_PARAMS_TYPE**

Description, NML Type: was used to set step related params, 133
Obs: currently not used in EMC2, needs to go to HAL
(maybe directly from the ini, not through NML)
Written From: none
Read To: emcTaskIssueCommand (emctaskmain.cc)
calls emcAxisSetStepParams (taskintf.cc)
which sends EMCMOT_SET_STEP_PARAMS

Parameter, Type: [axis (in EMC_AXIS_CMD_MSG), int] [setup_time, double] [hold_time, double]

3.3.33 **EMC_AXIS_STAT_TYPE**

Description, NML Type: status for axis, not sent as a message but used as is, 199
Written From: none
Read To: none

Parameter, Type: [a HUGE load of params]

3.4 **EMC TRAJ**

3.4.1 **EMC_TRAJ_SET_AXES_TYPE**

Description, NML Type: -, 201
Obs: not used
Written From: none
Read To: none

Parameter, Type: [axes, int]

3.4.2 **EMC_TRAJ_SET_UNITS_TYPE**

Description, NML Type: -, 202
Obs: not used
Written From: none
Read To: none

Parameter, Type: [linearUnits, double] [angularUnits, double]
3.4.3  **EMC_TRAJ_SET_CYCLE_TIME_TYPE**

Description, NML Type: -, 203
Obs: not used
Written From: none
Read To: none
Parameter, Type: [cycleTime, double]

3.4.4  **EMC_TRAJ_SET_MODE_TYPE**

Description, NML Type: -, 204
Obs: not used
Written From: none
Read To: none
Parameter, Type: [mode, enum EMC_TRAJ_MODE_ENUM]

3.4.5  **EMC_TRAJ_SET_VELOCITY_TYPE**

Description, NML Type: sends a request to set the vel, which is in internal units/sec, 205
Written From: sendVelMsg (emccanon.cc)
Read To: emcTaskIssueCommand (emctaskmain.cc)
calls emcTrajSetVelocity (minimill | bridgeporttaskintf.cc)
which sends EMCMOT_SET_VEL
Parameter, Type: [velocity, double]

3.4.6  **EMC_TRAJ_SET_ACCELERATION_TYPE**

Description, NML Type: -, 206
Obs: not used
Written From: none
Read To: none
Parameter, Type: [acceleration, double]

3.4.7  **EMC_TRAJ_SET_MAX_VELOCITY_TYPE**

Description, NML Type: -, 207
Obs: not used
Written From: none
Read To: none
Parameter, Type: [velocity, double]
3.4.8 EMC_TRAJ_SET_MAX_ACCELERATION_TYPE

Description, NML Type: -, 208
Obs: not used
Written From: none
Read To: none
Parameter, Type: [acceleration, double]

3.4.9 EMC_TRAJ_SET_SCALE_TYPE

Description, NML Type: set the feed override to be the percent value, 209
Obs: used for feed override messages
Written From: sendFeedOverride (emcsh.cc)
Read To: emcTaskIssueCommand (emctaskmain.cc)
calls emcTrajSetScale (taskintf.cc)
which sends EMCMOT_SCALE
Parameter, Type: [scale, double]

3.4.10 EMC_TRAJ_SET_MOTION_ID_TYPE

Description, NML Type: -, 210
Obs: not used
Written From: none
Read To: none
Parameter, Type: [id, int]

3.4.11 EMC_TRAJ_INIT_TYPE

Description, NML Type: -, 211
Obs: not used
Written From: none
Read To: none
Parameter, Type:

3.4.12 EMC_TRAJ_HALT_TYPE

Description, NML Type: -, 212
Obs: not used
Written From: none
Read To: none
Parameter, Type:
3.4.13 EMC_TRAJ_ENABLE_TYPE

Description, NML Type: -, 213
Obs: not used
Written From: none
Read To: none
Parameter, Type:

3.4.14 EMC_TRAJ_DISABLE_TYPE

Description, NML Type: -, 214
Obs: not used
Written From: none
Read To: none
Parameter, Type:

3.4.15 EMC_TRAJ_ABORT_TYPE

Description, NML Type: causes traj to abort ?, 215
Obs: not used, only read
Written From: none
Read To: emcTaskIssueCommand (emctaskmain.cc)
calls emcTrajAbort (taskintf.cc)
which sends EMCMOT_ABORT
Parameter, Type:

3.4.16 EMC_TRAJ_PAUSE_TYPE

Description, NML Type: causes traj to pause ?, 216
Obs: not used, only read
Written From: none
Read To: emcTaskIssueCommand (emctaskmain.cc)
calls emcTrajPause (minimill | bridgeporttaskintf.cc)
which sends EMCMOT_PAUSE
Parameter, Type:

3.4.17 EMC_TRAJ_STEP_TYPE

Description, NML Type: -, 217
Obs: not used
Written From: none
Read To: none
Parameter, Type:
3.4.18 **EMC_TRAJ_RESUME_TYPE**

Description, NML Type: causes traj to resume ?, 218

Obs: not used, only read

Written From: none

Read To: emcTaskIssueCommand (emctaskmain.cc)
calls emcTrajResume (minimill | bridgeporttaskintf.cc)
which sends EMCMOT_RESUME

Parameter, Type:

3.4.19 **EMC_TRAJ_DELAY_TYPE**

Description, NML Type: sets a delay in the task execution, 219

Obs: used with dwelling

Written From: DWELL (emccanon.cc)

Read To: emcTaskIssueCommand (emctaskmain.cc)

Parameter, Type: [delay, double]

3.4.20 **EMC_TRAJ_LINEAR_MOVE_TYPE**

Description, NML Type: sends a linear move from the interp to motion, 220

Obs: used

Written From: STRAIGHT_TRAVERSE, ARC_FEED (emccanon.cc)

Read To: checkInterpList, emcTaskIssueCommand (emctaskmain.cc)
calls emcTrajLinearMove (minimill | bridgeporttaskintf.cc)
which sends EMCMOT_SET_LINE

Parameter, Type: [end, EmcPose]

3.4.21 **EMC_TRAJ_CIRCULAR_MOVE_TYPE**

Description, NML Type: sends a circular move from the interp to motion, 221

Obs: used

Written From: ARC_FEED (emccanon.cc)

Read To: checkInterpList, emcTaskIssueCommand (emctaskmain.cc)
calls emcTrajCircularMove (minimill | bridgeporttaskintf.cc)
which sends EMCMOT_SET_CIRCLE Parameter, Type: [end, EmcPose] [center, PM_CARTESIAN] [normal, PM_CARTESIAN] [turn, int]

3.4.22 **EMC_TRAJ_SET_TERM_COND_TYPE**

Description, NML Type: chooses between blending or exact path mode, 222

Obs: used, seems the interp knows exact PATH, STOP and BLEND, motion however knows only BLEND or STOP

Written From: SET_MOTION_CONTROL_MODE (emccanon.cc)

Read To: emcTaskIssueCommand (emctaskmain.cc)
calls emcTrajSetTermCond (minimill | bridgeporttaskintf.cc)
which sends EMCMOT_TERM_COND_STOP or EMCMOT_TERM_COND_BLEND

Parameter, Type: [cond, int]
3.4.23 EMC_TRAJ_SET_OFFSET_TYPE

Description, NML Type: is used for tool length offset, 223
Obs: used, the message could transport more than just Z offset used for tool length
Written From: USE_TOOL_LENGTH_OFFSET (emccanon.cc)
Read To: emcTaskIssueCommand (emctaskmain.cc)

`remembers the origin offset into emcStatus->task.origin`

Parameter, Type: [offset, EmcPose]

3.4.24 EMC_TRAJ_SET_ORIGIN_TYPE

Description, NML Type: sets the origin coords ?, 224
Obs: used
Written From: SET_ORIGIN_OFFSETS (emccanon.cc)
Read To: emcTaskIssueCommand (emctaskmain.cc)

`remembers the tool length offset`

Parameter, Type: [origin, EmcPose]

3.4.25 EMC_TRAJ_SET_HOME_TYPE

Description, NML Type: -, 225
Obs: not used
Written From: none
Read To: none

Parameter, Type: [home, EmcPose]

3.4.26 EMC_TRAJ_SET_PROBE_INDEX_TYPE

Description, NML Type: sends the index pin used for probing, 226
Obs: should get obsolete, probin pin should get routed by HAL
Written From: sendSetProbeIndex (emcsh.cc)
Read To: emcTaskIssueCommand (emctaskmain.cc)
calls emcTrajSetProbeIndex (minimill | bridgeporttaskintf.cc)
which sends EMCMOT_SET_PROBE_INDEX

Parameter, Type: [index, int]

3.4.27 EMC_TRAJ_SET_PROBE_POLARITY_TYPE

Description, NML Type: sends the polarity for the pin used for probing, 227
Obs: should get obsolete, probin pin polarity should get routed by HAL
Written From: sendSetProbePolarity (emcsh.cc)
Read To: emcTaskIssueCommand (emctaskmain.cc)
calls emcTrajSetProbePolarity (minimill | bridgeporttaskintf.cc)
which sends EMCMOT_SET_PROBE_POLARITY

Parameter, Type: [polarity, int]
3.4.28 EMC_TRAJ_CLEAR_PROBE_TRIPPED_FLAG_TYPE
Description, NML Type: clears the probe tripped, 228
Obs: used
Written From: TURN_PROBE_ON (emccanon.cc) sendClearProbeTrippedFlag (emcsh.cc)
Read To: emcTaskIssueCommand (emctaskmain.cc) calls emcTrajClearProbeTrippedFlag (minimill | bridgeporttaskintf.cc) which sends EMCMOT_CLEAR_PROBE_FLAGS
Parameter, Type:

3.4.29 EMC_TRAJ_PROBE_TYPE
Description, NML Type: performs a straight probe move, 229
Obs: used
Written From: STRAIGHT_PROBE (emccanon.cc) sendProbe (emcsh.cc)
Read To: emcTaskIssueCommand (emctaskmain.cc) calls emcTrajProbe (minimill | bridgeporttaskintf.cc) which sends EMCMOT_PROBE
Parameter, Type: [pos, EmcPose]

3.4.30 EMC_TRAJ_SET_TELEOP_ENABLE_TYPE
Description, NML Type: sets the traj mode to teleop, 230
Obs: used
Written From: sendSetTeleopEnable (emcsh.cc)
Read To: emcTaskIssueCommand (emctaskmain.cc) calls emcTrajSetMode (minimill | bridgeporttaskintf.cc) which sends EMCMOT_TELEOP
Parameter, Type: [enable, int]

3.4.31 EMC_TRAJ_SET_TELEOP_VECTOR_TYPE
Description, NML Type: jogs in teleop mode, 231
Obs: used for jogging in teleop mode
Written From: sendJogCont (emcsh.cc)
Read To: emcTaskIssueCommand (emctaskmain.cc) calls emcTrajSetTeleopVector (minimill | bridgeporttaskintf.cc) which sends EMCMOT_SET_TELEOP_VECTOR
Parameter, Type: [vector, EmcPose]

3.4.32 EMC_TRAJ_STAT_TYPE
Description, NML Type: status for traj, not sent as a message but used as is, 299
Written From: none
Read To: none
Parameter, Type: [a HUGE load of params]
3.5 EMC MOTION

3.5.1 EMC_MOTION_INIT_TYPE

Description, NML Type: -, 301
Obs: not used
Written From: none
Read To: none
Parameter, Type:

3.5.2 EMC_MOTION_HALT_TYPE

Description, NML Type: -, 302
Obs: not used
Written From: none
Read To: none
Parameter, Type:

3.5.3 EMC_MOTION_ABORT_TYPE

Description, NML Type: -, 303
Obs: not used
Written From: none
Read To: none
Parameter, Type:

3.5.4 EMC_MOTION_SET_AOUT_TYPE

Description, NML Type: sets an analog output value coordinated with motion, 304
Obs: emccanon.cc currently lacks this in EMC2, not used in EMC2, needs to go to HAL
Written From: none
Read To: emcTaskIssueCommand (emctaskmain.cc)
calls emcMotionSetAout (minimill | bridgeporttaskintf.cc)
which sends EMCMOT_SET_AOUT Parameter, Type: [index, unsigned char] [start, double] [end, double] [now, unsigned char]

3.5.5 EMC_MOTION_SET_DOUT_TYPE

Description, NML Type: sets a digital output value coordinated with motion, 305
Obs: emccanon.cc currently lacks this in EMC2, not used in EMC2, needs to go to HAL
Written From: none
Read To: emcTaskIssueCommand (emctaskmain.cc)
calls emcMotionSetDout (minimill | bridgeporttaskintf.cc)
which sends EMCMOT_SET_DOUT
Parameter, Type: [index, unsigned char] [start, double] [end, double]
3.5.6 **EMC_MOTION_STAT_TYPE**

Description, NML Type: status for motion, not sent as a message but used as is, 399  
Written From: none  
Read To: none Parameter, Type: [heartbeat, unsigned long int] [traj, EMC_TRAJ_STAT] [axis[EMC_AXIS_MAX], EMC_AXIS_STAT]

3.6 **EMC TASK**

3.6.1 **EMC_TASK_INIT_TYPE**

Description, NML Type: calls the Task init(), 501  
Obs: not used, emcTaskInit called directly from emctask_startup()  
Written From: none  
Read To: emcTaskIssueCommand (emctaskmain.cc) calls emcTaskInit()  
Parameter, Type:

3.6.2 **EMC_TASK_HALT_TYPE**

Description, NML Type: -, 502  
Written From: none  
Read To: none  
Parameter, Type:

3.6.3 **EMC_TASK_ABORT_TYPE**

Description, NML Type: aborts task, cleans up, 503  
Obs: used on shutdown  
Written From: sendAbort (emcsh.cc)  
Read To: emcTaskIssueCommand (emctaskmain.cc) aborts all  
Parameter, Type:

3.6.4 **EMC_TASK_SET_MODE_TYPE**

Description, NML Type: sets current TASK mode, MANUAL, MDI, AUTO, 504  
Obs: used for switching the current mode  
Written From: sendManual sendMdi sendAuto (emcsh.cc)  
Read To: emcTaskIssueCommand (emctaskmain.cc) calls emcTaskSetMode (emcTask.cc)  
Parameter, Type: [mode, enum EMC_TASK_MODE_ENUM]

3.6.5 **EMC_TASK_SET_STATE_TYPE**

Description, NML Type: , 505  
Written From: none  
Read To: none  
Parameter, Type: [state, enum EMC_TASK_STATE_ENUM]
3.6.6  EMC_TASK_PLAN_OPEN_TYPE

Description, NML Type: , 506
Written From: none
Read To: none
Parameter, Type: [file, char[LINELEN]]

3.6.7  EMC_TASK_PLAN_RUN_TYPE

Description, NML Type: , 507
Written From: none
Read To: none
Parameter, Type: [line, int]

3.6.8  EMC_TASK_PLAN_READ_TYPE

Description, NML Type: , 508
Written From: none
Read To: none
Parameter, Type:

3.6.9  EMC_TASK_PLAN_EXECUTE_TYPE

Description, NML Type: , 509
Written From: none
Read To: none
Parameter, Type: [command, char[LINELEN]]

3.6.10  EMC_TASK_PLAN_PAUSE_TYPE

Description, NML Type: , 510
Written From: none
Read To: none
Parameter, Type:

3.6.11  EMC_TASK_PLAN_STEP_TYPE

Description, NML Type: , 511
Written From: none
Read To: none
Parameter, Type:
3.6.12 **EMC_TASK_PLAN_RESUME_TYPE**

Description, NML Type: , 512
Written From: none
Read To: none
Parameter, Type:

3.6.13 **EMC_TASK_PLAN_END_TYPE**

Description, NML Type: , 513
Written From: none
Read To: none
Parameter, Type:

3.6.14 **EMC_TASK_PLAN_CLOSE_TYPE**

Description, NML Type: , 514
Written From: none
Read To: none
Parameter, Type:

3.6.15 **EMC_TASK_PLAN_INIT_TYPE**

Description, NML Type: , 515
Written From: none
Read To: none
Parameter, Type:

3.6.16 **EMC_TASK_PLAN_SYNCH_TYPE**

Description, NML Type: , 516
Written From: none
Read To: none
Parameter, Type:

3.6.17 **EMC_TASK_STAT_TYPE**

Description, NML Type: , 599
Written From: none
Read To: none
Parameter, Type: [heartbeat, unsigned long int] [a HUGE load of params]
3.7 EMC TOOL

3.7.1 EMC_TOOL_INIT_TYPE

Description, NML Type: starts TOOL init, 1101
Obs: used for initializing the IO stuff, should load the tool table too
Written From: emcIoInit (iotaskintf.cc)
Read To: main (ioControl.cc simIoControl.cc)
emc_io_get_command (iosh.cc)
Parameter, Type:

3.7.2 EMC_TOOL_HALT_TYPE

Description, NML Type: stops TOOL, 1102
Obs: used for stopping IO, doesn’t actually do anything so far, in EMC1 it was send to subordinates too (spindle, aux, coolant, lube)
Written From: emcIoHalt (iotaskintf.cc)
Read To: main (ioControl.cc simIoControl.cc)
emc_io_get_command (iosh.cc)
Parameter, Type:

3.7.3 EMC_TOOL_ABORT_TYPE

Description, NML Type: aborts TOOL, 1103
Obs: used for aborting IO, doesn’t actually do anything so far, in EMC1 it was send to subordinates too (spindle, aux, coolant, lube)
Written From: emcIoAbort (iotaskintf.cc)
Read To: main (ioControl.cc simIoControl.cc)
emc_io_get_command (iosh.cc)
Parameter, Type:

3.7.4 EMC_TOOL_PREPARE_TYPE

Description, NML Type: prepares a tool for tool changing, 1104
Obs: loads the prep tool in emcioStatus.tool.toolPrepped, should go to PLC and make it move the desired tool in the toolchanging position
Written From: SELECT_TOOL (emccanon.cc)
Read To: emcTaskIssueCommand (emctaskmain.cc)
calls emcToolPrepare (iotaskintf.cc)
which sends it to the IO controller
Parameter, Type: [tool, int]
3.7.5 EMC_TOOL_LOAD_TYPE

Description, NML Type: changes the current tool with the prepared tool, 1105
Obs: loads the actual tool, makes toolprepped=0
Written From: CHANGE_TOOL (emccanon.cc)
Read To: emcTaskIssueCommand (emctaskmain.cc)
calls emcToolLoad (iotaskinf.cc)
which sends it to the IO controller,
main (simIoControl.cc ioControl.cc)
Parameter, Type:

3.7.6 EMC_TOOL_UNLOAD_TYPE

Description, NML Type: unloads the current tool from the spindle, 1106
Obs: unloads the current tool, not written in EMC2 only read
Written From: none
Read To: emcTaskIssueCommand (emctaskmain.cc)
calls emcToolUnLoad (iotaskinf.cc)
which sends it to the IO controller,
main (simIoControl.cc ioControl.cc)
Parameter, Type:

3.7.7 EMC_TOOL_LOAD_TOOL_TABLE_TYPE

Description, NML Type: loads the tool table, without this tool comp. can’t be made, 1107
Written From: sendLoadToolTable (emcsh.cc)
Read To: none
Parameter, Type: [file, char[LINELEN]]

3.7.8 EMC_TOOL_SET_OFFSET_TYPE

Description, NML Type: , 1108
Written From: none
Read To: none
Parameter, Type: [tool, int] [length, double] [diameter, double]

3.7.9 EMC_TOOL_STAT_TYPE

Description, NML Type: , 1199
Written From: none
Read To: none
Parameter, Type:
3.8 EMC AUX

3.8.1 EMC_AUX_INIT_TYPE

Description, NML Type: , 1201
Written From: none
Read To: none
Parameter, Type:

3.8.2 EMC_AUX_HALT_TYPE

Description, NML Type: , 1202
Written From: none
Read To: none
Parameter, Type:

3.8.3 EMC_AUX_ABORT_TYPE

Description, NML Type: , 1203
Written From: none
Read To: none
Parameter, Type:

3.8.4 EMC_AUX_DIO_WRITE_TYPE

Description, NML Type: , 1204
Written From: none
Read To: none
Parameter, Type: [index, int] [value, int]

3.8.5 EMC_AUX_AIO_WRITE_TYPE

Description, NML Type: , 1205
Written From: none
Read To: none
Parameter, Type: [index, int] [value, double]

3.8.6 EMC_AUX_ESTOP_ON_TYPE

Description, NML Type: , 1206
Written From: none
Read To: none
Parameter, Type:
3.8.7 EMC_AUX_ESTOP_OFF_TYPE

Description, NML Type: 1207
Written From: none
Read To: none
Parameter, Type:

3.8.8 EMC_AUX_STAT_TYPE

Description, NML Type: 1299
Written From: none
Read To: none Parameter, Type: [estop, int] [estopIn, int] [dout, unsigned char[EMC_AUX_MAX_DOUT]] [din, unsigned char[EMC_AUX_MAX_DIN]] [aout, double[EMC_AUX_MAX_AOUT]] [ain, double[EMC_AUX_MAX_AIN]]

3.9 EMC SPINDLE

3.9.1 EMC_SPINDLE_INIT_TYPE

Description, NML Type: 1301
Written From: none
Read To: none
Parameter, Type:

3.9.2 EMC_SPINDLE_HALT_TYPE

Description, NML Type: 1302
Written From: none
Read To: none
Parameter, Type:

3.9.3 EMC_SPINDLE_ABORT_TYPE

Description, NML Type: 1303
Written From: none
Read To: none
Parameter, Type:

3.9.4 EMC_SPINDLE_ON_TYPE

Description, NML Type: 1304
Written From: none
Read To: none
Parameter, Type: [speed, double]
3.9.5  EMC_SPINDLE_OFF_TYPE
Description, NML Type: , 1305
Written From: none
Read To: none
Parameter, Type:

3.9.6  EMC_SPINDLE_FORWARD_TYPE
Description, NML Type: , 1306
Written From: none
Read To: none
Parameter, Type: [speed, double]

3.9.7  EMC_SPINDLE_REVERSE_TYPE
Description, NML Type: , 1307
Written From: none
Read To: none
Parameter, Type: [speed, double]

3.9.8  EMC_SPINDLE_STOP_TYPE
Description, NML Type: , 1308
Written From: none
Read To: none
Parameter, Type: [speed, double]

3.9.9  EMC_SPINDLE_INCREASE_TYPE
Description, NML Type: , 1309
Written From: none
Read To: none
Parameter, Type: [speed, double]

3.9.10  EMC_SPINDLE_DECREASE_TYPE
Description, NML Type: , 1310
Written From: none
Read To: none
Parameter, Type:
3.9.11 EMC_SPINDLE_CONSTANT_TYPE
Description, NML Type: , 1311
Written From: none
Read To: none
Parameter, Type:

3.9.12 EMC_SPINDLE_BRAKE_RELEASE_TYPE
Description, NML Type: , 1312
Written From: none
Read To: none
Parameter, Type:

3.9.13 EMC_SPINDLE_BRAKE_ENGAGE_TYPE
Description, NML Type: , 1313
Written From: none
Read To: none
Parameter, Type:

3.9.14 EMC_SPINDLE_ENABLE_TYPE
Description, NML Type: , 1314
Written From: none
Read To: none
Parameter, Type:

3.9.15 EMC_SPINDLE_DISABLE_TYPE
Description, NML Type: , 1315
Written From: none
Read To: none
Parameter, Type:

3.9.16 EMC_SPINDLE_STAT_TYPE
Description, NML Type: , 1399
Written From: none
Read To: none Parameter, Type: [speed, double] [direction, int] [brake, int] [increasing, int] [enabled, int]
3.10 EMC COOLANT

3.10.1 EMC_COOLANT_INIT_TYPE

Description, NML Type: initializes the COOLANT controller (currently part of the IO controller), 1401
Obs: not written in EMC2, only read, in EMC1 it was sent when TOOL_INIT was sent
Written From: none
Read To: main (ioControl.cc simIoControl.cc)
  emc_io_get_command (iosh.cc)
Parameter, Type: none

3.10.2 EMC_COOLANT_HALT_TYPE

Description, NML Type: stops the COOLANT, 1402
Obs: not written in EMC2, only read, in EMC1 it was sent when TOOL_HALT was sent
Written From: none
Read To: main (ioControl.cc simIoControl.cc)
  emc_io_get_command (iosh.cc)
Parameter, Type: none

3.10.3 EMC_COOLANT_ABORT_TYPE

Description, NML Type: aborts the COOLANT, 1403
Obs: not written in EMC2, only read, in EMC1 it was sent when TOOL_ABORT was sent
Written From: none
Read To: main (ioControl.cc simIoControl.cc)
  emc_io_get_command (iosh.cc)
Parameter, Type: none

3.10.4 EMC_COOLANT_MIST_ON_TYPE

Description, NML Type: starts MIST coolant, 1404
Obs: used, written by emccanon.cc
Written From: MIST_ON (emccanon.cc) sendMistOn (emcsh.cc)
Read To: emcTaskIssueCommand (emctaskmain.cc)
calls emcCoolantMistOn (iotaskintf.cc)
which sends it to the IO controller,
main (simIoControl.cc ioControl.cc) iosh.cc
Parameter, Type: none
### 3.10.5 EMC_COOLANT_MIST_OFF_TYPE

Description, NML Type: stops MIST coolant, 1405  
Obs: used, written by emccanon.cc  
Written From: MIST_OFF (emccanon.cc) sendMistOff (emcsh.cc)  
Read To: emcTaskIssueCommand (emctaskmain.cc)  
calls emcCoolantMistOff (iotaskintf.cc)  
which sends it to the IO controller,  
main (simIoControl.cc ioControl.cc) iosh.cc  
Parameter, Type: none

### 3.10.6 EMC_COOLANT_FLOOD_ON_TYPE

Description, NML Type: starts FLOOD coolant, 1406  
Obs: used, written by emccanon.cc  
Written From: FLOOD_ON (emccanon.cc) sendFloodOn (emcsh.cc)  
Read To: emcTaskIssueCommand (emctaskmain.cc)  
calls emcCoolantFloodOn (iotaskintf.cc)  
which sends it to the IO controller,  
main (simIoControl.cc ioControl.cc) iosh.cc  
Parameter, Type: none

### 3.10.7 EMC_COOLANT_FLOOD_OFF_TYPE

Description, NML Type: stops FLOOD coolant, 1407  
Obs: used, written by emccanon.cc  
Written From: FLOOD_OFF (emccanon.cc) sendFloodOff (emcsh.cc)  
Read To: emcTaskIssueCommand (emctaskmain.cc)  
calls emcCoolantFloodOff (iotaskintf.cc)  
which sends it to the IO controller,  
main (simIoControl.cc ioControl.cc) iosh.cc  
Parameter, Type: none

### 3.10.8 EMC_COOLANT_STAT_TYPE

Description, NML Type: status for coolant, not sent as a message but used as is, 1499  
Written From: none  
Read To: none  
Parameter, Type: [mist, int] [flood, int]

### 3.11 EMC LUBE

#### 3.11.1 EMC_LUBE_INIT_TYPE

Description, NML Type: initializes the LUBE controller (currently part of the IO controller), 1501
Obs: not written in EMC2, only read, in EMC1 it was sent when TOOL_INIT was sent
Written From: none
Read To: main (ioControl.cc simIoControl.cc)
emc_io_get_command (iosh.cc)
Parameter, Type: none

3.11.2 EMC_LUBE_HALT_TYPE

Description, NML Type: stops the LUBE, 1502
Obs: not written in EMC2, only read, in EMC1 it was sent when TOOL_HALT was sent
Written From: none
Read To: main (ioControl.cc simIoControl.cc)
emc_io_get_command (iosh.cc)
Parameter, Type: none

3.11.3 EMC_LUBE_ABORT_TYPE

Description, NML Type: aborts the LUBE, 1503
Obs: not written in EMC2, only read, in EMC1 it was sent when TOOL_ABORT was sent
Written From: none
Read To: main (ioControl.cc simIoControl.cc)
emc_io_get_command (iosh.cc)
Parameter, Type: none

3.11.4 EMC_LUBE_ON_TYPE

Description, NML Type: starts LUBE, 1504
Obs: written only by the GUIs (emcsh.cc)
Written From: sendLubeOn (emcsh.cc)
Read To: emcTaskIssueCommand (emctaskmain.cc)
calls emcLubeOn (iotaskintf.cc)
which sends it to the IO controller,
main (ioControl.cc simIoControl.cc)
emc_io_get_command (iosh.cc)
Parameter, Type: none

3.11.5 EMC_LUBE_OFF_TYPE

Description, NML Type: stops LUBE, 1505
Obs: written only by the GUIs (emcsh.cc)
Written
From: sendLubeOff (emcsh.cc)
Read To: emcTaskIssueCommand (emctaskmain.cc)
calls emcLubeOff (iotaskintf.cc)
which sends it to the IO controller,
main (ioControl.cc simIoControl.cc)
emc_io_get_command (iosh.cc)
Parameter, Type: none
3.11.6 EMC_LUBE_STAT_TYPE
Description, NML Type: status for LUBE, not sent as a message but used as is, 1599
Written From: none
Read To: none
Parameter, Type: [on, int] [level, int]

3.12 EMC SET

3.12.1 EMC_SET_DIO_INDEX_TYPE
Description, NML Type: , 5001
Obs: not used
Written From: none
Read To: none
Parameter, Type: [value, int] [index, int]

3.12.2 EMC_SET_AIO_INDEX_TYPE
Description, NML Type: , 5002
Obs: not used
Written From: none
Read To: none
Parameter, Type: [value, int] [index, int]

3.12.3 EMC_SET_POLARITY_TYPE
Description, NML Type: , 5003
Obs: not used
Written From: none
Read To: none
Parameter, Type: [value, int] [polarity, int]

3.13 EMC IO

3.13.1 EMC_IO_INIT_TYPE
Description, NML Type: , 1601
Obs: not written in EMC2, only read
Written From: none
Read To: main (ioControl.cc simIoControl.cc)
emc_io_get_command (iosh.cc)
Parameter, Type:
3.13.2 EMC_IO_HALT_TYPE

Description, NML Type: , 1602
Obs: not used
Written From: none
Read To: none
Parameter, Type:

3.13.3 EMC_IO_ABORT_TYPE

Description, NML Type: , 1603
Obs: not used
Written From: none
Read To: none
Parameter, Type:

3.13.4 EMC_IO_SET_CYCLE_TIME_TYPE

Description, NML Type: , 1604
Obs: not used
Written From: none
Read To: none
Parameter, Type: [cycleTime, double]

3.13.5 EMC_IO_STAT_TYPE

Description, NML Type: status for IO, not sent as a message but used as is, 1699
Written From: none
Read To: none
Parameter, Type: [heartbeat, unsigned long int] [tool, EMC_TOOL_STAT] [spindle, EMC_SPINDLE_STAT] [coolant, EMC_COOLANT_STAT] [aux, EMC_AUX_STAT] [lube, EMC_LUBE_STAT]

3.14 EMC_INIT, HALT, & ABORT

3.14.1 EMC_INIT_TYPE

Description, NML Type: , 1901
Obs: not used
Written From: none
Read To: none
Parameter, Type:
3.14.2  EMC_HALT_TYPE

Description, NML Type: , 1902
Obs: not used
Written From: none
Read To: none
Parameter, Type:

3.14.3  EMC_ABORT_TYPE

Description, NML Type: , 1903
Obs: not used
Written From: none
Read To: none
Parameter, Type:

3.15  EMC LOG

3.15.1  EMC_LOG_OPEN_TYPE

Description, NML Type: opens the log file, 1904
Obs: not used in EMC2, it was used in EMC[1] from emclog.tcl
Written From: sendLogOpen (emcsh.cc)
Read To: emcTaskIssueCommand (emctaskmain.cc)
calls emcLogOpen (taskintf.cc)
which sends EMCMOT_OPEN_LOG Parameter, Type: [file, char[LINELEN]] [type, int] [size, int] [skip, int] [which, int] [triggerType, int] [triggerVar, int] [triggerThreshold, double]

3.15.2  EMC_LOG_START_TYPE

Description, NML Type: starts logging, 1905
Obs: not used in EMC2, it was used in EMC[1] from emclog.tcl
Written From: sendLogStart (emcsh.cc)
Read To: emcTaskIssueCommand (emctaskmain.cc) calls
emcLogStart (taskintf.cc)
which sends EMCMOT_START_LOG
Parameter, Type: none

3.15.3  EMC_LOG_STOP_TYPE

Description, NML Type: stops logging, 1906
Obs: not used in EMC2, it was used in EMC[1] from emclog.tcl
Written From: sendLogStop (emcsh.cc)
Read To: emcTaskIssueCommand (emctaskmain.cc) calls
emcLogStop (taskintf.cc)
which sends EMCMOT_STOP_LOG
Parameter, Type: none
3.15.4 **EMC_LOG_CLOSE_TYPE**

Description, NML Type: closes the log file, 1907

Obs: not used in EMC2, it was used in EMC[1] from emclog.tcl

Written From: sendLogClose (emcsh.cc)

Read To: emcTaskIssueCommand (emctaskmain.cc)
calls emcLogClose (taskintf.cc)
which sends EMCMOT_CLOSE_LOG

Parameter, Type: none

3.16 **EMC_STA**

3.16.1 **EMC_STAT_TYPE**

Description, NML Type: aggregation of all the status messages, not sent as a message but used as is all over the place, 1999

Written From: none

Read To: none

Parameter, Type: [task, EMC_TASK_STAT]
[motion, EMC_MOTION_STAT]
[io, EMC_IO_STAT]
[logFile, char[LINELEN]]
[logType, int]
[logSize, int]
[logSkip, int]
logOpen, int] [logStarted, int] [logPoints, int]
Chapter 4

Coding Style

This chapter describes the source code style preferred by the LinuxCNC team.

4.1 Do no harm

When making small edits to code in a style different than the one described below, observe the local coding style. Rapid changes from one coding style to another decrease code readability.

Never check in code after running “indent” on it. The whitespace changes introduced by indent make it more difficult to follow the revision history of the file.

Do not use an editor that makes unneeded changes to whitespace (e.g., which replaces 8 spaces with a tabstop on a line not otherwise modified, or word-wraps lines not otherwise modified)

4.2 Tab Stops

A tab stop always corresponds to 8 spaces. Do not write code that displays correctly only with a differing tab stop setting.

4.3 Indentation

Use 4 spaces per level of indentation. Combining 8 spaces into one tab is acceptable but not required.

4.4 Placing Braces

Put the opening brace last on the line, and put the closing brace first:

```c
if (x) {
    // do something appropriate
}
```

The closing brace is on a line of its own, except in the cases where it is followed by a continuation of the same statement, i.e. a while in a do-statement or an else in an if-statement, like this:

```c
do {
    // something important
} while (x > 0);
```
and

```c
if (x == y) {
    // do one thing
} else if (x < y) {
    // do another thing
} else {
    // do a third thing
}
```

This brace-placement also minimizes the number of empty (or almost empty) lines, which allows a greater amount of code or comments to be visible at once in a terminal of a fixed size.

## 4.5 Naming

C is a Spartan language, and so should your naming be. Unlike Modula-2 and Pascal programmers, C programmers do not use cute names like ThisVariableIsATemporaryCounter. A C programmer would call that variable `tmp`, which is much easier to write, and not the least more difficult to understand.

However, descriptive names for global variables are a must. To call a global function `foo` is a shooting offense.

GLOBAL variables (to be used only if you really need them) need to have descriptive names, as do global functions. If you have a function that counts the number of active users, you should call that function `count_active_users()` or similar, you should not call it `cntusr()`.

Encoding the type of a function into the name (so-called Hungarian notation) is brain damaged - the compiler knows the types anyway and can check those, and it only confuses the programmer. No wonder Microsoft makes buggy programs.

LOCAL variable names should be short, and to the point. If you have some random integer loop counter, it should probably be called `i`. Calling it `loop_counter` is non-productive, if there is no chance of it being misunderstood. Similarly, `tmp` can be just about any type of variable that is used to hold a temporary value.

If you are afraid to mix up your local variable names, you have another problem, which is called the function-growth-hormone-imbalance syndrome. See next chapter.

## 4.6 Functions

Functions should be short and sweet, and do just one thing. They should fit on one or two screenfuls of text (the ISO/ANSI screen size is 80x24, as we all know), and do one thing and do that well.

The maximum length of a function is inversely proportional to the complexity and indentation level of that function. So, if you have a conceptually simple function that is just one long (but simple) case-statement, where you have to do lots of small things for a lot of different cases, it’s OK to have a longer function.

However, if you have a complex function, and you suspect that a less-than-gifted first-year high-school student might not even understand what the function is all about, you should adhere to the maximum limits all the more closely. Use helper functions with descriptive names (you can ask the compiler to in-line them if you think it’s performance-critical, and it will probably do a better job of it that you would have done).

Another measure of the function is the number of local variables. They shouldn’t exceed 5-10, or you’re doing something wrong. Re-think the function, and split it into smaller pieces. A human brain can generally easily keep track of about 7 different things, anything more and it gets confused. You know you’re brilliant, but maybe you’d like to understand what you did 2 weeks from now.

## 4.7 Commenting

Comments are good, but there is also a danger of over-commenting. NEVER try to explain HOW your code works in a comment: it’s much better to write the code so that the working is obvious, and it’s a waste of time to explain badly written code.
Generally, you want your comments to tell WHAT your code does, not HOW. A boxed comment describing the function, return value, and who calls it placed above the body is good. Also, try to avoid putting comments inside a function body: if the function is so complex that you need to separately comment parts of it, you should probably re-read the Functions section again. You can make small comments to note or warn about something particularly clever (or ugly), but try to avoid excess. Instead, put the comments at the head of the function, telling people what it does, and possibly WHY it does it.

If comments along the lines of /* Fix me */ are used, please, please, say why something needs fixing. When a change has been made to the affected portion of code, either remove the comment, or amend it to indicate a change has been made and needs testing.

4.8 Shell Scripts & Makefiles

Not everyone has the same tools and packages installed. Some people use vi, others emacs - A few even avoid having either package installed, preferring a lightweight text editor such as nano or the one built in to Midnight Commander.

gawk versus mawk - Again, not everyone will have gawk installed, mawk is nearly a tenth of the size and yet conforms to the Posix AWK standard. If some obscure gawk specific command is needed that mawk does not provide, than the script will break for some users. The same would apply to mawk. In short, use the generic awk invocation in preference to gawk or mawk.

4.9 C++ Conventions

C++ coding styles are always likely to end up in heated debates (a bit like the emacs versus vi arguments). One thing is certain however, a common style used by everyone working on a project leads to uniform and readable code.

Naming conventions: Constants either from #defines or enumerations should be in upper case through out. Rationale: Makes it easier to spot compile time constants in the source code. e.g. EMC_MESSAGE_TYPE

Classes and Namespaces should capitalize the first letter of each word and avoid underscores. Rationale: Identifies classes, constructors and destructors. e.g. GtkWidget

Methods (or function names) should follow the C recommendations above and should not include the class name. Rationale: Maintains a common style across C and C++ sources. e.g. get_foo_bar()

However, boolean methods are easier to read if they avoid underscores and use an is prefix (not to be confused with methods that manipulate a boolean). Rationale: Identifies the return value as TRUE or FALSE and nothing else. e.g. isOpen, isHomed

Do NOT use Not in a boolean name, it leads only leads to confusion when doing logical tests. e.g. isNotOnLimit or is_not_on_limit are BAD.

Variable names should avoid the use of upper case and underscores except for local or private names. The use of global variables should be avoided as much as possible. Rationale: Clarifies which are variables and which are methods. Public: e.g. axislimit Private: e.g. maxvelocity_

Specific method naming conventions

The terms get and set should be used where an attribute is accessed directly. Rationale: Indicates the purpose of the function or method. e.g. get_foo set_bar

For methods involving boolean attributes, set & reset is preferred. Rationale: As above. e.g. set_amp_enable reset_amp_fault

Math intensive methods should use compute as a prefix. Rationale: Shows that it is computationally intensive and will hog the CPU. e.g. compute_PID

Abbreviations in names should be avoided where possible - The exception is for local variable names. Rationale: Clarity of code. e.g. pointer is preferred over ptr compute is preferred over cmp compare is again preferred over cmp.

Enumerates and other constants can be prefixed by a common type name e.g. enum COLOR { COLOR_RED, COLOR_BLUE };

Excessive use of macros and defines should be avoided - Using simple methods or functions is preferred. Rationale: Improves the debugging process.
Include Statements Header files must be included at the top of a source file and not scattered throughout the body. They should be sorted and grouped by their hierarchical position within the system with the low level files included first. Include file paths should NEVER be absolute - Use the compiler -I flag instead. Rationale: Headers may not be in the same place on all systems.

Pointers and references should have their reference symbol next to the variable name rather than the type name. Rationale: Reduces confusion. e.g. float *x or int &i

Implicit tests for zero should not be used except for boolean variables. e.g. if (spindle_speed != 0) NOT if (spindle_speed)

Only loop control statements must be included in a for() construct. e.g. sum = 0; for (i = 0; i < 10; i++) { sum += value[i]; } NOT for (i = 0, sum =0; i < 10; i++) sum += value[i];

Likewise, executable statements in conditionals must be avoided. e.g. if (fd = open(file_name) is bad.

Complex conditional statements should be avoided - Introduce temporary boolean variables instead.

Parentheses should be used in plenty in mathematical expressions - Do not rely on operator precedence when an extra parentheses would clarify things.

File names: C++ sources and headers use .cc and .hh extension. The use of .c and .h are reserved for plain C. Headers are for class, method, and structure declarations, not code (unless the functions are declared inline).

4.10 Python coding standards

Use the PEP 8 style for Python code.

4.11 Comp coding standards

In the declaration portion of a .comp file, begin each declaration at the first column. Insert extra blank lines when they help group related items.

In the code portion of a .comp file, follow normal C coding style.
Chapter 5

Adding Configuration Selection Items

Example Configurations can be added to the Configuration Selector by two methods:

- Auxiliary applications — Applications installed independently with a deb package can place configuration subdirectories in a specified system directory. The directory name is specified using the shell linuxcnc_var script:

  ```bash
  $ linuxcnc_var LINUXCNC_AUX_EXAMPLES
  /usr/share/linuxcnc/aux_examples
  ```

- Runtime settings — the configuration selector can also offer configuration subdirectories specified at runtime using an exported environmental variable (LINUXCNC_AUX_CONFIGS). This variable should be a path list of one or more configuration directories separated by a (:). Typically, this variable would be set in a shell starting linuxcnc or in a user’s ~/.profile startup script. Example:

  ```bash
  export LINUXCNC_AUX_CONFIGS=~/myconfigs:/opt/otherconfigs
  ```
Chapter 6

Contributing to LinuxCNC

6.1 Introduction

This document contains information for developers about LinuxCNC infrastructure, and describes the best practices for contributing code and documentation updates to the LinuxCNC project.

Throughout this document, "source" means both the source code to the programs and libraries, and the source text for the documentation.

6.2 Communication among LinuxCNC developers

The two main ways that project developers communicate with each other are:

- Via IRC, at #linuxcnc-devel on FreeNode.
- Via email, on the developers’ mailing list: https://lists.sourceforge.net/lists/listinfo/emc-developers

6.3 The LinuxCNC Source Forge project

We use Source Forge for mailing lists: http://sourceforge.net/p/emc/mailman/

6.4 The git Revision Control System

All of the LinuxCNC source is maintained in the revision control system git.

6.4.1 LinuxCNC official git repo

The official LinuxCNC git repo is at http://github.com/linuxcnc/linuxcnc/

Anyone can get a read-only copy of the LinuxCNC source tree via git:

git clone https://github.com/linuxcnc/linuxcnc linuxcnc-dev

If you are a developer with push access, then follow github’s instructions for setting up a repository that you can push from.

1 http://git-scm.com/
Note that the clone command put the local LinuxCNC repo in a directory called \texttt{linuxcnc-dev}, instead of the default \texttt{linuxcnc}. This is because the LinuxCNC software by default expects configs and G-code programs in a directory called \texttt{HOME/\linuxcnc}, and having the git repo there too is confusing.

Issues and pull requests are welcome on github: https://github.com/LinuxCNC/linuxcnc/issues https://github.com/LinuxCNC/linuxcnc/pulls

### 6.4.2 Use of git in the LinuxCNC project

We use the "merging upwards" and "topic branches" git workflows described here:

https://www.kernel.org/pub/software/scm/git/docs/gitworkflows.html

We have a development branch called \texttt{master}, and one or more stable branches with names like 2.6 and 2.7 indicating the version number of the releases we make from it.

Bugfixes go in the oldest applicable stable branch, and that branch gets merged into the next newer stable branch, and so on up to \texttt{master}. The committer of the bugfix may do the merges themselves, or they may leave the merges for someone else.

New features generally go in the \texttt{master} branch, but some kinds of features (specifically well isolated device drivers and documentation) may (at the discretion of the stable branch release managers) go into a stable branch and get merged up just like bugfixes do.

### 6.4.3 git tutorials

There are many excellent, free git tutorials on the internet.

The first place to look is probably the "gittutorial" manpage. This manpage is accessible by running "man gittutorial" in a terminal (if you have the git manpages installed). The gittutorial and its follow-on documentation are also available online here:

- git tutorial: https://www.kernel.org/pub/software/scm/git/docs/gittutorial.html
- git tutorial 2: https://www.kernel.org/pub/software/scm/git/docs/gittutorial-2.html
- Everyday git with 20 commands or so: https://www.kernel.org/pub/software/scm/git/docs/giteveryday.html

For a more thorough documentation of git see the "Pro Git" book: http://git-scm.com/book

Another online tutorial that has been recommended is “Git for the Lazy”: http://wiki.spheredev.org/Git_for_the_lazy

### 6.5 Overview of the process

The high-level overview of how to contribute changes to the source goes like this:

- Communicate with the project developers and let us know what you’re hacking on
- Clone the git repo
- Make your changes in a local branch, making sure you "sign off" your commits according to our signed-off-by policy (see below).
- Adding documentation and tests is an important part of adding a new feature. Otherwise, others won’t know how to use your feature, and if other changes break your feature it can go unnoticed without a test.
- Share your changes with the other project developers in one of these ways:
  - Push your branch to github and create a github pull request to https://github.com/linuxcnc/linuxcnc (this requires a github account)
– Push your branch to a publicly visible git repo (such as github, bitbucket, your own publicly-accessible server, etc) and share that location on the emc-developers mailing list, or
– Email your commits to the emc-developers mailing list (use `git format-patch` to create the patches)

**Advocate for your patch**

– Explain what problem it addresses and why it should be included in LinuxCNC
– Be receptive to questions and feedback from the developer community.
– It is not uncommon for a patch to go through several revisions before it is accepted.

### 6.6 git configuration

In order to be considered for inclusion in the LinuxCNC source, commits must have correct Author fields identifying the author of the commit. A good way to ensure this is to set your global git config:

```
git config --global user.name "Your full name"
git config --global user.email "you@example.com"
```

Use your real name (not a handle), and use an unobfuscated e-mail address.

### 6.7 Effective use of git

#### 6.7.1 Commit contents

Keep your commits small and to the point. Each commit should accomplish one logical change to the repo.

#### 6.7.2 Write good commit messages

Keep commit messages around 72 columns wide (so that in a default-size terminal window, they don’t wrap when shown by `git log`).

Use the first line as a summary of the intent of the change (almost like the subject line of an e-mail). Follow it with a blank line, then a longer message explaining the change. Example:

```
Get rid of RTAPI_SUCCESS, use 0 instead
```

The test "retval < 0" should feel familiar; it’s the same kind of test you use in userspace (returns -1 for error) and in kernel space (returns -ERRNO for error)

#### 6.7.3 Commit to the proper branch

Bugfixes should go on the oldest applicable branch. New features should go in the master branch. If you’re not sure where a change belongs, ask on irc or on the mailing list.

#### 6.7.4 Use multiple commits to organize changes

When appropriate, organize your changes into a branch (a series of commits) where each commit is a logical step towards your ultimate goal. For example, first factor out some complex code into a new function. Then, in a second commit, fix an underlying bug. Then, in the third commit, add a new feature which is made easier by the refactoring and which would not have worked without fixing that bug.

This is helpful to reviewers, because it is easier to see that the "factor out code into new function" step was right when there aren’t other edits mixed in; it’s easier to see that the bug is fixed when the change that fixes it is separate from the new feature; and so on.
6.7.5 Follow the style of the surrounding code

Make an effort to follow the prevailing indentation style of surrounding code. In particular, changes to whitespace make it harder for other developers to track changes over time. When reformatting code must be done, do it as a commit separate from any semantic changes.

6.7.6 Simplify complicated history before sharing with fellow developers

With git, it’s possible to record every edit and false start as a separate commit. This is very convenient as a way to create checkpoints during development, but often you don’t want to share these false starts with others.

Git provides two main ways to clean history, both of which can be done freely before you share the change:

- `git commit --amend` lets you make additional changes to the last thing you committed, optionally modifying the commit message as well. Use this if you realized right away that you left something out of the commit, or if you typo’d the commit message.

- `git rebase --interactive upstream-branch` lets you go back through each commit made since you forked your feature branch from the upstream branch, possibly editing commits, dropping commits, or squashing (combining) commits with others. Rebase can also be used to split individual commits into multiple new commits.

6.7.7 Make sure every commit builds

If your change consists of several patches, `git rebase -i` may be used to reorder these patches into a sequence of commits which more clearly lays out the steps of your work. A potential consequence of reordering patches is that one might get dependencies wrong - for instance, introducing a use of a variable, and the declaration of that variable only follows in a later patch.

While the branch HEAD will build, not every commit might build in such a case. That breaks `git bisect` - something somebody else might use later on to find the commit which introduced a bug. So beyond making sure your branch builds, it is important to assure every single commit builds as well.

There’s an automatic way to check a branch for each commit being buildable - see [http://dustin.sallings.org/2010/03/28/git-test-sequence.html](http://dustin.sallings.org/2010/03/28/git-test-sequence.html), and the code at [https://github.com/dustin/bindir/blob/master/git-test-sequence](https://github.com/dustin/bindir/blob/master/git-test-sequence). Use as follows (in this case testing every commit from origin/master to HEAD, including running regression tests):

```
cd linuxcnc-dev
$ git-test-sequence origin/master.. '(cd src && make && ../scripts/runtests)'
```

This will either report All’s well or Broke on <commit>

6.7.8 Renaming files

Please use the ability to rename files very cautiously. Like running indent on single files, renames still make it more difficult to follow changes over time. At a minimum, you should seek consensus on irc or the mailing list that the rename is an improvement.

6.7.9 Prefer "rebase"

Use `git pull --rebase` instead of bare `git pull` in order to keep a nice linear history. When you rebase, you always retain your work as revisions that are ahead of origin/master, so you can do things like `git format-patch` them to share with others without pushing to the central repository.
6.8 Other ways to contribute

There are many ways to contribute to LinuxCNC, that are not addressed by this document. These ways include:

- Answering questions on the forum, mailing lists, and in IRC
- Reporting bugs on the bug tracker, forum, mailing lists, or in IRC
- Helping test experimental features
Chapter 7

Glossary

A listing of terms and what they mean. Some terms have a general meaning and several additional meanings for users, installers, and developers.

**Acme Screw**
A type of lead-screw that uses an Acme thread form. Acme threads have somewhat lower friction and wear than simple triangular threads, but ball-screws are lower yet. Most manual machine tools use acme lead-screws.

**Axis**
One of the computer controlled movable parts of the machine. For a typical vertical mill, the table is the X axis, the saddle is the Y axis, and the quill or knee is the Z axis. Angular axes like rotary tables are referred to as A, B, and C. Additional linear axes relative to the tool are called U, V, and W respectively.

**Axis(GUI)**
One of the Graphical User Interfaces available to users of LinuxCNC. It features the modern use of menus and mouse buttons while automating and hiding some of the more traditional LinuxCNC controls. It is the only open-source interface that displays the entire tool path as soon as a file is opened.

**Gmoccapy (GUI)**
A Graphical User Interfaces available to users of LinuxCNC. It features the use and feel of an industrial control and can be used with touch screen, mouse and keyboard. It support embedded tabs and hal driven user messages, it offers a lot of hal beens to be controled with hardware. Gmoccapy is highly cusomizable.

**Backlash**
The amount of "play" or lost motion that occurs when direction is reversed in a lead screw, or other mechanical motion driving system. It can result from nuts that are loose on leadscrews, slippage in belts, cable slack, "wind-up" in rotary couplings, and other places where the mechanical system is not "tight". Backlash will result in inaccurate motion, or in the case of motion caused by external forces (think cutting tool pulling on the work piece) the result can be broken cutting tools. This can happen because of the sudden increase in chip load on the cutter as the work piece is pulled across the backlash distance by the cutting tool.

**Backlash Compensation**
Any technique that attempts to reduce the effect of backlash without actually removing it from the mechanical system. This is typically done in software in the controller. This can correct the final resting place of the part in motion but fails to solve problems related to direction changes while in motion (think circular interpolation) and motion that is caused when external forces (think cutting tool pulling on the work piece) are the source of the motion.

**Ball Screw**
A type of lead-screw that uses small hardened steel balls between the nut and screw to reduce friction. Ball-screws have very low friction and backlash, but are usually quite expensive.

**Ball Nut**
A special nut designed for use with a ball-screw. It contains an internal passage to re-circulate the balls from one end of the screw to the other.
CNC
Computer Numerical Control. The general term used to refer to computer control of machinery. Instead of a human operator turning cranks to move a cutting tool, CNC uses a computer and motors to move the tool, based on a part program.

Comp
A tool used to build, compile and install LinuxCNC HAL components.

Configuration(n)
A directory containing a set of configuration files. Custom configurations are normally saved in the users home/linuxcnc/\-configs directory. These files include LinuxCNC's traditional INI file and HAL files. A configuration may also contain several general files that describe tools, parameters, and NML connections.

Configuration(v)
The task of setting up LinuxCNC so that it matches the hardware on a machine tool.

Coordinate Measuring Machine
A Coordinate Measuring Machine is used to make many accurate measurements on parts. These machines can be used to create CAD data for parts where no drawings can be found, when a hand-made prototype needs to be digitized for moldmaking, or to check the accuracy of machined or molded parts.

Display units
The linear and angular units used for onscreen display.

DRO
A Digital Read Out is a system of position-measuring devices attached to the slides of a machine tool, which are connected to a numeric display showing the current location of the tool with respect to some reference position. DROs are very popular on hand-operated machine tools because they measure the true tool position without backlash, even if the machine has very loose Acme screws. Some DROs use linear quadrature encoders to pick up position information from the machine, and some use methods similar to a resolver which keeps rolling over.

EDM
EDM is a method of removing metal in hard or difficult to machine or tough metals, or where rotating tools would not be able to produce the desired shape in a cost-effective manner. An excellent example is rectangular punch dies, where sharp internal corners are desired. Milling operations can not give sharp internal corners with finite diameter tools. A wire EDM machine can make internal corners with a radius only slightly larger than the wire’s radius. A sinker EDM can make internal corners with a radius only slightly larger than the radius on the corner of the sinking electrode.

EMC
The Enhanced Machine Controller. Initially a NIST project. Renamed to LinuxCNC in 2012.

EMCIO
The module within LinuxCNC that handles general purpose I/O, unrelated to the actual motion of the axes.

EMCMOT
The module within LinuxCNC that handles the actual motion of the cutting tool. It runs as a real-time program and directly controls the motors.

Encoder
A device to measure position. Usually a mechanical-optical device, which outputs a quadrature signal. The signal can be counted by special hardware, or directly by the parport with LinuxCNC.

Feed
Relatively slow, controlled motion of the tool used when making a cut.

Feed rate
The speed at which a cutting motion occurs. In auto or mdi mode, feed rate is commanded using an F word. F10 would mean ten machine units per minute.

Feedback
A method (e.g., quadrature encoder signals) by which LinuxCNC receives information about the position of motors.
Feedrate Override
A manual, operator controlled change in the rate at which the tool moves while cutting. Often used to allow the operator to adjust for tools that are a little dull, or anything else that requires the feed rate to be “tweaked”.

Floating Point Number
A number that has a decimal point. (12.300) In HAL it is known as float.

G-Code
The generic term used to refer to the most common part programming language. There are several dialects of G-code, LinuxCNC uses RS274/NGC.

GUI
Graphical User Interface.

General
A type of interface that allows communications between a computer and a human (in most cases) via the manipulation of icons and other elements (widgets) on a computer screen.

LinuxCNC
An application that presents a graphical screen to the machine operator allowing manipulation of the machine and the corresponding controlling program.

HAL
Hardware Abstraction Layer. At the highest level, it is simply a way to allow a number of building blocks to be loaded and interconnected to assemble a complex system. Many of the building blocks are drivers for hardware devices. However, HAL can do more than just configure hardware drivers.

Home
A specific location in the machine’s work envelope that is used to make sure the computer and the actual machine both agree on the tool position.

ini file
A text file that contains most of the information that configures LinuxCNC for a particular machine.

Instance
One can have an instance of a class or a particular object. The instance is the actual object created at runtime. In programmer jargon, the Lassie object is an instance of the Dog class.

Joint Coordinates
These specify the angles between the individual joints of the machine. See also Kinematics.

Jog
Manually moving an axis of a machine. Jogging either moves the axis a fixed amount for each key-press, or moves the axis at a constant speed as long as you hold down the key. In manual mode, jog speed can be set from the graphical interface.

kernel-space
See real-time.

Kinematics
The position relationship between world coordinates and joint coordinates of a machine. There are two types of kinematics. Forward kinematics is used to calculate world coordinates from joint coordinates. Inverse kinematics is used for exactly the opposite purpose. Note that kinematics does not take into account, the forces, moments etc. on the machine. It is for positioning only.

Lead-screw
An screw that is rotated by a motor to move a table or other part of a machine. Lead-screws are usually either ball-screws or acme screws, although conventional triangular threaded screws may be used where accuracy and long life are not as important as low cost.

Machine units
The linear and angular units used for machine configuration. These units are specified and used in the ini file. HAL pins and parameters are also generally in machine units.
MDI
Manual Data Input. This is a mode of operation where the controller executes single lines of G-code as they are typed by
the operator.

NIST
National Institute of Standards and Technology. An agency of the Department of Commerce in the United States.

NML
Neutral Message Language provides a mechanism for handling multiple types of messages in the same buffer as well as
simplifying the interface for encoding and decoding buffers in neutral format and the configuration mechanism.

Offsets
An arbitrary amount, added to the value of something to make it equal to some desired value. For example, gcode programs
are often written around some convenient point, such as X0, Y0. Fixture offsets can be used to shift the actual execution
point of that gcode program to properly fit the true location of the vise and jaws. Tool offsets can be used to shift the
“uncorrected” length of a tool to equal that tool’s actual length.

Part Program
A description of a part, in a language that the controller can understand. For LinuxCNC, that language is RS-274/NGC,
commonly known as G-code.

Program Units
The linear and angular units used in a part program. The linear program units do not have to be the same as the linear
machine units. See G20 and G21 for more information. The angular program units are always measured in degrees.

Python
General-purpose, very high-level programming language. Used in LinuxCNC for the Axis GUI, the Stepconf configuration
tool, and several G-code programming scripts.

Rapid
Fast, possibly less precise motion of the tool, commonly used to move between cuts. If the tool meets the workpiece or
the fixturing during a rapid, it is probably a bad thing!

Rapid rate
The speed at which a rapid motion occurs. In auto or mdi mode, rapid rate is usually the maximum speed of the machine.
It is often desirable to limit the rapid rate when testing a g-code program for the first time.

Real-time
Software that is intended to meet very strict timing deadlines. Under Linux, in order to meet these requirements it is
necessary to install a realtime kernel such as RTAI and build the software to run in the special real-time environment. For
this reason real-time software runs in kernel-space.

RTAI
Real Time Application Interface, see https://www.rtai.org/, the real-time extensions for Linux that LinuxCNC can use to
achieve real-time performance.

RTLINUX
See https://en.wikipedia.org/wiki/RTLinux, an older real-time extension for Linux that LinuxCNC used to use to achieve
real-time performance. Obsolete, replaced by RTAI.

RTAPI
A portable interface to real-time operating systems including RTAI and POSIX pthreads with realtime extensions.

RS-274/NGC
The formal name for the language used by LinuxCNC part programs.

Servo Motor
Generally, any motor that is used with error-sensing feedback to correct the position of an actuator. Also, a motor which
is specially-designed to provide improved performance in such applications.

Servo Loop
A control loop used to control position or velocity of an motor equipped with a feedback device.
Signed Integer
A whole number that can have a positive or negative sign. In HAL it is known as s32. (A signed 32-bit integer has a usable range of -2,147,483,647 to +2,147,483,647.)

Spindle
The part of a machine tool that spins to do the cutting. On a mill or drill, the spindle holds the cutting tool. On a lathe, the spindle holds the workpiece.

Spindle Speed Override
A manual, operator controlled change in the rate at which the tool rotates while cutting. Often used to allow the operator to adjust for chatter caused by the cutter’s teeth. Spindle Speed Override assumes that the LinuxCNC software has been configured to control spindle speed.

Stepconf
An LinuxCNC configuration wizard. It is able to handle many step-and-direction motion command based machines. It writes a full configuration after the user answers a few questions about the computer and machine that LinuxCNC is to run on.

Stepper Motor
A type of motor that turns in fixed steps. By counting steps, it is possible to determine how far the motor has turned. If the load exceeds the torque capability of the motor, it will skip one or more steps, causing position errors.

TASK
The module within LinuxCNC that coordinates the overall execution and interprets the part program.

Tcl/Tk
A scripting language and graphical widget toolkit with which several of LinuxCNCs GUIs and selection wizards were written.

Traverse Move
A move in a straight line from the start point to the end point.

Units
See "Machine Units", "Display Units", or "Program Units".

Unsigned Integer
A whole number that has no sign. In HAL it is known as u32. (An unsigned 32-bit integer has a usable range of zero to 4,294,967,296.)

World Coordinates
This is the absolute frame of reference. It gives coordinates in terms of a fixed reference frame that is attached to some point (generally the base) of the machine tool.
Chapter 8

Legal Section

8.1 Copyright Terms

Copyright (c) 2000-2015 LinuxCNC.org

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

8.2 GNU Free Documentation License

GNU Free Documentation License Version 1.1, March 2000

Copyright © 2000 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary...
Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:
A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission. B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five). C. State on the Title page the name of the publisher of the Modified Version, as the publisher. D. Preserve all the copyright notices of the Document. E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices. F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below. G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice. H. Include an unaltered copy of this License. I. Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence. J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission. K. In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein. L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles. M. Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version. N. Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.
You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document’s Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright (c) YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have no Invariant Sections, write "with no Invariant Sections" instead of saying which ones are invariant. If you have no Front-Cover Texts, write "no Front-Cover Texts" instead of "Front-Cover Texts being LIST"; likewise for Back-Cover Texts.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.
Chapter 9

Index

A
acme screw, 72
axis, 72

B
backlash, 72
backlash compensation, 72
ball nut, 72
ball screw, 72

C
CNC, 73
comp, 73
coordinate measuring machine, 73

D
display units, 73
DRO, 73

E
EDM, 73
EMC, 73
EMCMOT, 73
encoder, 73

F
feed, 73
feed rate, 73
feedback, 73
feedrate override, 74

G
G-Code, 74
GUI, 72, 74

H
HAL, 74
home, 74

I
INI, 74
Instance, 74

J
jog, 74
joint coordinates, 74

K
kinematics, 74

L
lead screw, 74
loop, 75

M
machine units, 74
MDI, 75

N
NIST, 75
NML, 75

O
offsets, 75

P
part Program, 75
program units, 75

R
rapid, 75
rapid rate, 75
real-time, 75
RS274NGC, 75
RTAI, 75
RTAPI, 75
RTLINUX, 75

S
servo motor, 75
Signed Integer, 76
spindle, 76
stepper motor, 76

T
TASK, 76
Tk, 76
Traverse Move, 76
**U**
units, 76
Unsigned Integer, 76

**W**
world coordinates, 76