# Motor Record and related software

*Tim Mooney, Joe Sullivan, Ron Sluiter*

---

## Contents



## Overview

This documentation describes version R6-3 of the EPICS motor record, and related EPICS software required to build and use it. Version R6-3 of the motor record is compatible with EPICS base R3.14.9 and above.

The motor record is intended to support motors of all kinds, but currently supports only the following variety of motor controllers (in addition to Soft Channel support):

- Oregon Micro Systems, Inc. (OMS) models; VME8, VME44, VME58, VS4, VX2, MAXv, PC68 and PC78.
- Highland Technologies model V540.
- Newport models MM3000, MM4000/5/6, PM500, ESP300/100 and XPSC8.
- Intelligent Motion Systems, Inc. (IMS) models IM483 and MDrive.
- Advanced Control Systems, Corp. model MCB-4B.
- Mclennan models PM304 and PM600.
- Physik Instrumente (PI) GmbH & Co. model C-630, C-844, C-848, C-862, E-662 and E-710.
- MicroMo model MVP 2001 B02.
- Micos model MoCo dc controller.
- Delta Tau PMAC2-VME controller.
- Faulhaber MCDC2805 servo controller.
- Parker Hannifin, Compumotor Division, 6K Series controllers.
- New Focus, models; 8750 and 8752.
- ACS Motion Control, SPiiPlus model.
- Spectra-Physics, Encoder Mike Controller, Model 18011.
- Thorlabs, Piezo Controller, Model MDT695.
- Animatics Corporation SmartMotor.
- piezosystem jena GmbH EDS data interface module.
- Kohzu SC-800 stepper motor controller.

The record maintains two coordinate systems for motor position ("user" and "dial " coordinates); displays drive and readback values; enforces limits to motor motion and maintains those limits in both coordinate systems; displays the states of limit switches; can use a home switch; optionally takes out backlash in a user-defined direction; and provides a mechanism by which the user and other EPICS records can recalibrate the motor position in either coordinate system. The record also supports "tweak", "jog", and "home" motions, and supports both absolute and relative motions in user coordinates. Two "stop" switches are provided: a simple one for use by other records and by channel-access clients, and a more versatile one for interactive use.

Except where specified otherwise, fields associated with the motor position and its derivatives take values in user-specified "engineering units", such as degrees; the engineering unit name is contained in the field EGU. Thus, generally, speeds are expressed in EGU's per second. Accelerations, however, are expressed as the

number of seconds taken to accelerate to full speed. However, additional fields are provided so that the motor position can be specified in steps and the speed in revolutions per second, and so that the step size can be set by specifying the number of steps per revolution and the number of EGU's per revolution.

The motor record can read motor position from the controller's readback register or encoder register, or from any other EPICS record, via an EPICS input link. While the motor is moving, the record can trigger an output link periodically, to send readback information to other EPICS records. When a complete motion (possibly including backlash takeout) is finished, the record can trigger a forward link to process other EPICS records.

The motor record can force its drive fields to agree with its readback fields, and it does so in a variety of circumstances (e.g., when the user tells a motor to stop, and when a limit switch is hit). Therefore, if you are driving the motor record's VAL or DVAL field with the output of another record, and you want that record always to contain the same value as the motor record, you must handle this behavior in the database. One way to do this is to forward link the motor record to a soft analog output record, and to cause that AO record to grab the motor record's VAL or DVAL field and poke it into your record.

The motor record is unlike most other EPICS records in that its processing is neither "synchronous" nor "asynchronous", as these terms are used in the EPICS Record Reference Manual. Currently, the PACT field is always FALSE after record processing has completed, even though a motor motion may be in progress. This means the record always responds to channel-access puts, and can be stopped or retargeted at any time. The record's forward link is not executed until the motor has stopped and no motion requests are pending.

## Field Descriptions

In addition to fields common to all record types (see the EPICS Record Reference Manual for these) the motor record has the fields described below.



---

## Alphabetical list of record-specific fields

NOTE: Hot links in this table take you only to the *section* in which the linked item is described in detail. You'll probably have to scroll down to find the actual item.

| Name | Access | Prompt | Data type | Comment |
|---|---|---|---|---|
| ACCL | R/W | Seconds to Velocity | DOUBLE | acceleration time |
| ATHM | R | At HOME | SHORT | uses the HOME switch |
| BACC | R/W | BL Seconds to Veloc. | DOUBLE | backlash acceleration time |
| BDST | R/W | BL Distance (EGU) | DOUBLE | backlash distance |
| BVEL | R/W | BL Velocity (EGU/s) | DOUBLE | backlash speed |
| CARD | R | Card Number | SHORT | EPICS card # |

| | | | | |
|---|---|---|---|---|
| CBAK | None | Callback structure | NOACCESS | |
| CDIR | R | Raw commanded direction | SHORT | |
| CNEN | R/W | Enable control | RECCHOICE | (0:"Disable", 1:"Enable") |
| DCOF | R/W | Derivative Gain | DOUBLE | |
| DHLM | R/W* | Dial High Limit | DOUBLE | |
| DIFF | R | Difference dval-drbv | DOUBLE | |
| DINP | R/W | DMOV Input Link | INLINK | |
| DIR | R/W* | User Direction | RECCHOICE | (0:"Pos", 1:"Neg") |
| DLLM | R/W* | Dial Low Limit | DOUBLE | |
| DLY | R/W | Readback settle time (s) | DOUBLE | |
| DMOV | R | Done moving to value | SHORT | The "done" flag |
| DOL | R | Desired Output Loc | INLINK | only for closed-loop mode |
| DRBV | R | Dial Readback Value | DOUBLE | |
| DVAL | R/W* | Dial Desired Value | DOUBLE | |
| EGU | R/W | Engineering Units | STRING | |
| ERES | R/W* | Encoder Step Size (EGU) | DOUBLE | |
| FOF | R/W | Freeze Offset | SHORT | |
| FOFF | R/W | Offset-Freeze Switch | RECCHOICE | (0:"Variable", 1:"Frozen") |
| FRAC | R/W | Move Fraction | DOUBLE | |
| HHSV | R/W* | Hihi Severity | GBLCHOICE | |
| HIGH | R/W* | High Alarm Limit | DOUBLE | |
| HIHI | R/W* | Hihi Alarm Limit | DOUBLE | |
| HLM | R/W* | User High Limit | DOUBLE | |
| HLS | R | At High Limit Switch | SHORT | |
| HLSV | R/W* | HW Lim. Violation Svr | GBLCHOICE | |
| HOMF | R/W* | Home Forward | SHORT | |
| HOMR | R/W* | Home Reverse | SHORT | |
| HOPR | R/W | High Operating Range | DOUBLE | |
| HSV | R/W* | High Severity | GBLCHOICE | |
| HVEL | R/W* | Home Velocity | DOUBLE | |
| ICOF | R/W | Integral Gain | DOUBLE | |
| INIT | R/W | Startup commands | STRING | |
| JAR | R/W | Jog Acceleration (EGU/s^2) | DOUBLE | |
| JOGF | R/W* | Jog motor Forward | SHORT | careful! |
| JOGR | R/W* | Jog motor Reverse | SHORT | careful! |
| JVEL | R/W | Jog Velocity | DOUBLE | |
| LDVL | R | Last Dial Des Val | DOUBLE | |
| LLM | R/W* | User Low Limit | DOUBLE | |
| LLS | R | At Low Limit Switch | SHORT | |
| LLSV | R/W* | Lolo Severity | GBLCHOICE | |

| | | | | |
|---|---|---|---|---|
| LOCK | R/W* | Soft Channel Position Lock | RECCHOICE | (0:"No", 1:"Yes") |
| LOLO | R/W* | Lolo Alarm Limit | DOUBLE | |
| LOPR | R/W | Low Operating Range | DOUBLE | |
| LOW | R/W* | Low Alarm Limit | DOUBLE | |
| LRLV | R | Last Rel Value | DOUBLE | |
| LRVL | R | Last Raw Des Val | DOUBLE | |
| LSPG | R | Last SPMG | RECCHOICE | (See SPMG) |
| LSV | R/W* | Low Severity | GBLCHOICE | |
| LVAL | R | Last User Des Val | DOUBLE | |
| LVIO | R | Limit violation | SHORT | |
| MIP | R | Motion In Progress | SHORT | |
| MISS | R | Ran out of retries | SHORT | |
| MMAP | R | Monitor Mask | ULONG | |
| MOVN | R | Motor is moving | SHORT | Don't confuse with DMOV |
| MRES | R/W* | Motor Step Size (EGU) | DOUBLE | |
| MSTA | R | Motor Status | ULONG | |
| NMAP | R | Monitor Mask | ULONG | |
| NTM | R/W* | New Target Monitor | RECCHOICE | (0:"No", 1:"Yes") |
| OFF | R/W | User Offset (EGU) | DOUBLE | |
| OMSL | R/W | Output Mode Select | GBLCHOICE | |
| OUT | R/W | Output Specification | OUTLINK | |
| PCOF | R/W | Proportional Gain | DOUBLE | |
| PERL | R/W | Periodic Limits | RECCHOICE | (0:"No", 1:"Yes") |
| POST | R/W | Post-move commands | STRING | |
| PP | R | Post process command | SHORT | |
| PREC | R/W | Display Precision | SHORT | |
| PREM | R/W | Pre-move commands | STRING | |
| RBV | R | User Readback Value | DOUBLE | |
| RCNT | R | Retry count | SHORT | |
| RDBD | R/W | Retry Deadband (EGU) | DOUBLE | |
| RDBL | R | Readback Location | INLINK | |
| RDIF | R | Difference rval-rrbv | LONG | |
| REP | R | Raw Encoder Position | DOUBLE | |
| RHLS | R | Raw High Limit Switch | SHORT | |
| RINP | R/W | RMP Input Link | INLINK | |
| RLLS | R | Raw Low Limit Switch | SHORT | |
| RLNK | R | Readback OutLink | OUTLINK | |
| RLV | R/W* | Relative Value | DOUBLE | |
| RMP | R | Raw Motor Position | DOUBLE | |
| RRBV | R | Raw Readback Value | DOUBLE | |

| | | | | |
|---|---|---|---|---|
| RRES | R/W | Readback Step Size (EGU) | DOUBLE | |
| RTRY | R/W | Max retry count | SHORT | |
| RVAL | R/W* | Raw Desired Value | DOUBLE | |
| RVEL | R | Raw Velocity | LONG | |
| S | R/W | Speed (RPS) | DOUBLE | |
| SBAK | R/W | BL Speed (RPS) | DOUBLE | |
| SBAS | R/W | Base Speed (RPS) | DOUBLE | |
| SET | R/W | Set/Use Switch | RECCHOICE | (0:"Use", 1:"Set") |
| SMAX | R/W | Max Velocity (RPS) | DOUBLE | |
| SPMG | R/W* | Stop/Pause/Move/Go | RECCHOICE | (0:"Stop", 1:"Pause", 2:"Move", 3:"Go") |
| SREV | R/W* | Steps per Revolution | LONG | |
| SSET | R/W | Set SET Mode | SHORT | |
| STOO | R/W | STOP OutLink | OUTLINK | |
| STOP | R/W* | Stop | SHORT | |
| STUP | R | Status Update Request | RECCHOICE | ON(1) |
| SUSE | R/W | Set USE Mode | SHORT | |
| TDIR | R | Direction of Travel | SHORT | |
| TWF | R/W* | Tweak motor Forward | SHORT | |
| TWR | R/W* | Tweak motor Reverse | SHORT | |
| TWV | R/W* | Tweak Step Size (EGU) | DOUBLE | |
| UEIP | R/W* | Use Encoder If Present | RECCHOICE | (0:"No", 1:"Yes") |
| UREV | R/W* | EGU's per Revolution | DOUBLE | |
| URIP | R/W* | Use RDBL Link If Present | RECCHOICE | (0:"No", 1:"Yes") |
| VAL | R/W* | User Desired Value | DOUBLE | |
| VBAS | R/W | Base Velocity (EGU/s) | DOUBLE | |
| VELO | R/W | Velocity (EGU/s) | DOUBLE | |
| VERS | R | Code Version | DOUBLE | e.g., "1.95" |
| VMAX | R/W | Max Velocity (EGU/s) | DOUBLE | |
| VOF | R/W | Variable Offset | SHORT | |

Note: In the **Access** column above:

| | |
|---|---|
| R | Read only |
| R/W | Read and write are allowed |
| R/W* | Read and write are allowed; write triggers record processing if the record's SCAN field is set to "Passive." |
| N | No access allowed |

Note: In the **Prompt** column above:

| | |
|---|---|
| EGU | Engineering Units |
| RPS | Revolutions Per Second |

## Calibration-related fields

| Name | Access | Prompt | Data type | Comments |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| DIR | R/W* | User Direction | RECCHOICE | (0:"Pos", 1:"Neg") |
| User and dial values are related by the equation<br>`userVAL = DialVAL * DIR + OFFset`<br>This field is the "DIR" in the above equation. | | | | |
| | | | | |
| OFF | R/W | User Offset (EGU) | DOUBLE | User and dial coordinates can differ by a sign (the DIR field) and an offset (OFF), according to the following equation:<br>`userVAL = DialVAL * DIR + OFFset`<br>This field is "OFFset" in the above equation. It is not normally written to directly by the user. |
| FOFF | R/W | Offset-Freeze Switch | RECCHOICE | (0:"Variable", 1:"Frozen") |
| VOF | R/W | Variable Offset | SHORT | Set Offset switch (FOFF) to "Variable". |
| FOF | R/W | Freeze Offset | SHORT | Set Offset switch (FOFF) to "Frozen". |
| The user can cause the difference between user and dial coordinates to remain fixed (i.e., the record will not change it, although the user may) by setting FOFF to "Frozen." The fields VOF and FOF are intended for use in backup/restore operations; any write to them will drive the FOFF field to "Variable" (VOF) or "Frozen" (FOF). | | | | |
| | | | | |
| SET | R/W | Set/Use Switch | RECCHOICE | (0:"Use", 1:"Set") |
| SSET | R/W | Set SET Mode | SHORT | Set Set/Use switch to "Set". |
| SUSE | R/W | Set USE Mode | SHORT | Set Set/Use switch to "Use". |
| SET is a toggle switch used in calibrating the motor's user and dial positions:<br><br>When SET = 0 ("Use"), writes to the user-coordinate drive field (VAL) cause the dial-coordinate drive field (DVAL) to change, and the motor to move. Writes to the dial-coordinate drive field (DVAL) cause the user-coordinate drive field (VAL) to change, and the motor to move.<br><br>When SET = 1 ("Set"), writes to the dial-coordinate drive field (DVAL) and to the raw drive field (RVAL) cause a new raw motor position to be loaded into the hardware without any change to the user-coordinate drive field (VAL). Writes to other fields that would normally move the motor, change the user-coordinate drive field (VAL), and the offset between user and dial coordinates (the OFF field), with corresponding changes in the user-coordinate limit fields (HLM and LLM). When the offset is frozen (FOFF=1), writes to any drive field affect both user and dial values, and also load the hardware position register. | | | | |
| | | | | |

# Motor-resolution fields

| Name | Access | Prompt | Data type | Comments |
|---|---|---|---|---|
| MRES | R/W* | Motor Step Size (EGU) | DOUBLE | May be positive or negative |
| SREV | R/W* | Steps per Revolution | LONG | Must be strictly positive |

| | | | | |
|---|---|---|---|---|
| UREV | R/W* | EGU's per Revolution | DOUBLE | May be positive or negative |

MRES, and (SREV,UREV) represent two ways of specifying the motor resolution--the distance or angle, in engineering units (EGU's), associated with a single motor step. The equation relating these quantities is "MRES = UREV/SREV ". Initially, SREV has the value 200, the number of full steps per revolution for most stepper motors, and the record never changes this field. Only the user can change it.

When MRES is changed, the motor record sets UREV = MRES*SREV. When UREV or SREV is changed, the motor sets MRES = UREV/SREV. In all cases, the effect of a motor resolution change on the reported motor position depends in a simple way on the value of the SET field:
If (SET = 1), new user and dial values (VAL, DVAL) are calculated from the existing raw value (RVAL).
If (SET = 0), a new raw value is calculated from the existing dial value.
The motor doesn't move in either case; neither does the actual motor speed (in revolutions per second) change.

If either MRES or UREV is changed, motor speeds that are expressed in engineering units per second (i.e., those whose names contain the letter 'V': VELO, BVEL, VMAX and VBAS) are automatically adjusted by the motor record according to the following equations: VELO = UREV * S; BVEL = UREV * SBAK; VMAX = UREV * SMAX; VBAS = UREV * SBAS. Motor speeds that are expressed in revolutions per second (S, SBAK, and SBAS) are independent of changes to MRES or UREV. In contrast, when SREV is changed, only MRES is adjusted by the motor record, thus allowing all other fields to remain unaffected.

Currently, changes to motor-resolution fields have no effect on the values of limit fields (although they should).

MRES or UREV allow negative values so that the user/dial coordinate systems can be configured to the opposite polarity of the motor controller's.

| | | | | |
|---|---|---|---|---|
| | | | | |
| ERES | R/W* | Encoder Step Size (EGU) | DOUBLE | Encoder resolution: the distance or angle, in engineering units, associated with a single encoder step. ERES may be positive or negative. If the user sets ERES to zero, the record will overwrite it with MRES. |
| RRES | R/W | Readback Step Size (EGU) | DOUBLE | Readback-device resolution: the distance or angle, in engineering units, associated with a unit change of the number retrieved via the readback- location input link (RDBL). RRES may be either positive or negative. |
| | | | | |
| UEIP | R/W* | Use Encoder If Present | RECCHOICE (0:"No", 1:"Yes") | Switch: nonzero value tells the record to read the encoder (if the hardware indicates an encoder is present) and to ignore the value read back from the hardware's step-count register.<br><br>The state of the UEIP, together with the MSTA encoder indicator, determine:<br><br>1. if the RRBV is set to either the feedback (REP) or the command (RMP) position.<br>2. whether absolute or relative position commands are used. |
| URIP | R/W* | Use RDBL Link If Present | RECCHOICE (0:"No", 1:"Yes") | Switch: nonzero value tells the record to get the motor position from the readback-location link (RDBL) (if it contains valid EPICS link information, and if no error occurs in the attempt to read |

| | | | | |
|---|---|---|---|---|
| | | | | from the link) and to ignore values read back from the hardware's step-count and encoder registers. |

These switches also direct the record to calculate destinations in relative, rather than absolute, terms, since the ratio of encoder and readback units to motor steps may not actually be constant.

## Motion-related fields

| Name | Access | Prompt | Data type | Comments |
|---|---|---|---|---|
| VMAX | R/W | Max Velocity (EGU/s) | DOUBLE | Valid range; 0 <= VMAX. VMAX=0 disables maximum velocity range checking. |
| SMAX | R/W | Max Velocity (RPS) | DOUBLE | Valid range; 0 <= SMAX |
| VBAS | R/W | Base Velocity (EGU/s) | DOUBLE | Valid range; 0 <= VBAS |
| SBAS | R/W | Base Speed (RPS) | DOUBLE | Valid range; 0 <= SBAS |

Range checking is done in such a way that any minimum (i.e., VBAS/SBAS) or maximum (i.e., VMAX/SMAX) value entered is valid. For example, if the minimum is entered and it exceeds the maximum, then the maximum is set to the new minimum value. A VMAX value of zero disables maximum velocity range checking.

At boot-up, if one field of a field pair (i.e., VMAX/SMAX, VBAS/SBAS, VELO/S, BVEL/SBAK) is zero and the other field is nonzero, the nonzero field takes precedence. If both fields of a given field pair are nonzero, the RPS member of the field pair (i.e., SMAX, SBAS, S, SBAK) takes precedence.

Slew (VELO/S) and backup (BVEL/SBAK) velocity fields are silently forced by the motor record to be within the range set by VMAX/SMAX and VBAS/SBAS, inclusively.

Those who use both BURT and VMAX (i.e., nonzero VMAX) should insure that VMAX and VBAS are placed before VELO and BVEL in their BURT request files.

The intent of VBAS/SBAS is to prevent the motor from moving at speeds slow enough to excite its resonance, which can cause the motor to miss steps. The motor is expected to accelerate from a stand-still to VBAS in one motor pulse.

| | | | | |
|---|---|---|---|---|
| VELO | R/W | Velocity (EGU/s) | DOUBLE | Valid range; VBAS <= VELO <= VMAX |
| S | R/W | Speed (RPS) | DOUBLE | Valid range; SBAS <= S <= SMAX |

VELO is the speed, in engineering units per second, at which the motor is moved after the acceleration phase of a motion is finished. S is the same speed expressed in revolutions per second. The record makes sure that VELO and S are consistent, using the equation S = VELO/UREV.

| | | | | |
|---|---|---|---|---|
| HVEL | R/W | Velocity (EGU/s) | DOUBLE | Homing velocity; valid range; VBAS <= HVEL <= VMAX |

| | | | | |
|---|---|---|---|---|
| ACCL | R/W | Seconds to Velocity | DOUBLE | The length, in seconds, of the acceleration and deceleration phases of a motor motion. |
| The motor record expects the hardware to produce a trapezoidal speed profile. That is, the motor speed is expected to increase linearly with time from the base speed, VBAS, to the full speed, VELO, in ACCL seconds. At the end of a motion, the speed is expected to decrease similarly to VBAS. | | | | |
| JVEL | R/W | Jog Velocity (EGU/s) | DOUBLE | Valid range; VBAS <= VELO <= VMAX |
| JAR | R/W | Jog Acceleration (EGU/s^2) | DOUBLE | Default value: VELO / ACCL |
| With the OMS and IMS device drivers, jog velocity can be changed on-the-fly. The velocity will accelerate to the new velocity based on the JAR field. | | | | |
| BDST | R/W | BL Distance (EGU) | DOUBLE | The signed distance, in dial coordinates, used for backlash takeout. |
| The algorithm used in moves to a (dial-coordinate) position called "TARGET" follows:<br><br>1) If the motor is to move a distance greater than the magnitude of BDST, or if the motor is to move in a direction opposite to the sign of BDST, then the motor will move first to position (TARGET-BDST), at an acceleration specified by ACCL and speed VELO, and then to position TARGET, at an acceleration specified by BACC and speed BVEL.<br><br>2) If the motor is to move a distance smaller than the magnitude of BDST, and if the motor is to move in the same direction as the sign of BDST, then backlash is assumed already to have been taken out, and the motor will move to position TARGET at an acceleration specified by BACC and speed BVEL. | | | | |
| | | | | |
| BVEL | R/W | BL Velocity (EGU/s) | DOUBLE | |
| SBAK | R/W | BL Speed (RPS) | DOUBLE | |
| BVEL is the speed, in engineering units per second, at which the motor is move after the acceleration phase of a backlash-takeout motion is finished. SBAK is the same speed expressed in revolutions per second. Neither BVEL nor SBAK may be negative or zero. | | | | |
| | | | | |
| BACC | R/W | BL Seconds to Veloc. | DOUBLE | The length, in seconds, of the acceleration and deceleration phases of a backlash-takeout motion. See discussion of the acceleration field ACCL for more specific information. |
| | | | | |
| FRAC | R/W | Move Fraction | DOUBLE | |
| This field supports closed-loop control of pathological devices for which drive values are not expected to compare reproducibly with readback values. (Inchworms and other friction-driven devices are good examples: the number of steps taken by an inchworm motor is a very poor indicator of the distance it has traveled.) | | | | |

| | | | | |
|---|---|---|---|---|
| | | | | In a move from position CURRENT to position TARGET, the motor record will ask hardware to move a distance FRAC*(TARGET-CURRENT). When that motion is complete, the record will request a motion of FRAC*(remaining distance), and so on until the target position has been reached. |
| | | | | |
| RDBD | R/W | Retry Deadband (EGU) | DOUBLE | When the motor has finished a complete motion, possibly including backlash takeout, the motor record will compare its current position with the desired position. If the magnitude of the difference is greater than RDBD, the motor will try again, as if the user had requested a move from the now current position to the desired position. Only a limited number of retries will be performed (see RTRY). |
| RTRY | R/W | Max retry count | SHORT | The maximum number of times the motor record will try again to move to the desired position. When the retry limit is reached, the motor record will declare the motion finished. If the desired position was not reached, the field MISS will be set to 1. |

## Link-related fields

| Name | Access | Prompt | Data type | Comments |
|---|---|---|---|---|
| OUT | R/W | Output Specification | OUTLINK | If Soft Channel device support is specified, this field is an EPICS link; each time DVAL is changed, device support puts DVAL to this link. Otherwise, this field specifies the hardware to be controlled. |
| RDBL | R | Readback Location | INLINK | This field specifies the field (of this or any other EPICS record) from which the motor's current position is to be read when the field URIP (Use Readback If Present) has the value "Yes" (1). If this field does not contain a valid EPICS link, the URIP may as well have the value "No" (0). If Soft Channel device support is specified, this field is monitored for value changes by a CA event task. |
| DOL | R | Desired Output Loc | INLINK | If this field contains a valid EPICS link, and the OMSL field has the value "closed_loop" (1), then every time the motor record is processed, it will get a value for the VAL field from the link and move to that location, ignoring all other drive fields. Closed-loop mode has not been tested extensively. |
| OMSL | R/W | Output Mode Select | GBLCHOICE | (0:"supervisory", 1:"closed_loop")<br>If this field has the value "closed_loop" (1), and the field DOL contains a valid EPICS link, then every time the motor record is processed, it will get a value for the VAL field from the link and move to that location, ignoring all other drive fields. Closed-loop mode has not been tested extensively. |
| RLNK | R | Readback OutLink | OUTLINK | If this field contains a valid EPICS link, then every time the motor record is processed, it will put the (engineering-unit) readback value RBV to that link. |
| DINP | R/W | DMOV Input Link | INLINK | If Soft Channel device support is specified, the value specified by this link is used to set the DONE bit in the MSTA field; which in turn sets the DMOV field. |

| | | | | |
|---|---|---|---|---|
| RINP | R/W | RMP Input Link | INLINK | If Soft Channel device support is specified, the value specified by this link is used to set the RMP field. |
| STOO | R/W | STOP OutLink | OUTLINK | If Soft Channel device support is specified, a one is written to the specified link each time the STOP_AXIS motor command is issued. |

**Soft Channel Device Driver**

The Soft Channel database links (i.e., DINP, RINP and STOO) are only processed when the Soft Channel device driver is selected. These links are ignored when using any other Motor Record device driver.

The input links (i.e., DINP, RDBL and RINP) are monitored for value changes by a CA event task. Users must choose either a dial input link (RDBL) or a raw input link (RINP), but not both. At this time, the above links are **not** dynamically retargetable.

Note that Soft Channel device support resets the target position (VAL/DVAL/RVAL) to the actual position (RBV/DRBV/RRBV) the first time, and only the first time, that the PV pointed to by RDBL is posted. Hence, Soft Channel device support requires that the RDBL PV have a valid value the first time it is posted.

Note that JOG[F/R] does not work with the Soft Channel device driver.

## Limit-related fields

| Name | Access | Prompt | Data type | Comments |
|---|---|---|---|---|
| HLM | R/W* | User High Limit | DOUBLE | The maximum allowed value of the VAL field. If HLM changes so that VAL is no longer less than HLM, then the record will set the field LVIO to 1. If the DIR field has the value "Pos", then HLM will always be consistent with DHLM, otherwise HLM will always be consistent with DLLM. |
| LLM | R/W* | User Low Limit | DOUBLE | The minimum allowed value of the VAL field. If LLM changes so that VAL is no longer greater than LLM, then the record will set the field LVIO to 1. If the DIR field has the value "Pos", then LLM will always be consistent with DLLM, otherwise LLM will always be consistent with DHLM. |
| DHLM | R/W* | Dial High Limit | DOUBLE | The maximum allowed value of the DVAL field. If DHLM changes so that DVAL is no longer less than DHLM, then the record will set the field LVIO to 1.. If the DIR field has the value "Pos", then DHLM will always be consistent with HLM, otherwise DHLM will always be consistent with LLM. |
| DLLM | R/W* | Dial Low Limit | DOUBLE | The minimum allowed value of the DVAL field. If DLLM changes so that DVAL is no longer greater than DLLM, then the record will set the field LVIO to 1. If the DIR field has the value "Pos", then DLLM will always be consistent with LLM, otherwise DLLM will always be consistent with HLM. |
| LVIO | R | Limit violation | SHORT | A value of 1 indicates that the dial-value drive field, DVAL, or the dial-value readback field, DRBV, is outside of the limits (DHLM, DLLM), and this prevents the motor from moving. If the backlash distance, BDST, is non-zero, it further restricts the allowable range of DVAL. When a JOG button is hit, LVIO goes to 1 and |

| | | | | |
|---|---|---|---|---|
| | | | | stops the motor if/when DVAL gets to within one second's travel time of either limit. |
| PERL | R/W | Periodic Limits | RECCHOICE | (0:"No", 1:"Yes")<br>Not implemented. Originally intended to support periodic "limits" on the VAL field (such as those associated with a rotation stage--e.g., [0...360] or [-180...180]) independently of the actual soft limits HLM and LLM. |
| HOPR | R/W | High Operating Range | DOUBLE | Not used. See HLM and DHLM. |
| LOPR | R/W | Low Operating Range | DOUBLE | Not used. See LLM and DLLM. |
| | | | | |
| HLS | R | At High Limit Switch | SHORT | |
| RHLS | R | Raw High Limit Switch | SHORT | |
| If either of these fields is nonzero, then the motor is at the positive-limit switch, where the positive sense is that of the user-coordinate system for HLS, and that of the raw (step-number) coordinate system for RHLS. | | | | |
| | | | | |
| | | | | |
| LLS | R | At Low Limit Switch | SHORT | |
| RLLS | R | Raw Low Limit Switch | SHORT | |
| If either of these fields is nonzero, then the motor is at the negative-limit switch, where the positive sense is that of the user-coordinate system for LLS, and that of the raw (step-number) coordinate system for RLLS. | | | | |
| | | | | |

# Command-button fields

| Name | Access | Prompt | Data type | Comments |
|---|---|---|---|---|
| SPMG | R/W* | Stop/Pause/Move/Go | RECCHOICE | (0:"Stop", 1:"Pause", 2:"Move", 3:"Go") |
| This field is intended primarily for interactive use, and normally has the value "Go."<br><br>If the user sets this field to "Stop," the motor will decelerate to a stop, the VAL field will be set equal to the | | | | |

RBV field, and the DVAL field will be set equal to the DRBV field. (These actions ensure that the motor will not start moving again until a drive field is changed.) In any case, the motor will not move while SPMG has the value "Stop" or "Pause."

If "SPMG" has the value "Move," the motor record will reset SPMG to "Pause" when a motion completes. This behavior supports users who want a motor to sit still until they say "Move", no matter what changes occur in the drive fields.

| | | | | |
|---|---|---|---|---|
| STOP | R/W* | Stop | SHORT | |

When this field is set to 1, the record will immediately reset it to 0, and the motor will decelerate to a stop. When the motor has stopped, VAL will be set equal to RBV, and DVAL will be set equal to DRBV. (This ensures that the motor will not start moving the next time the record is processed, unless a drive field is explicitly changed. If you want the motor to pause, use the SPMG field.)

| | | | | |
|---|---|---|---|---|
| HOMF | R/W* | Home Forward | SHORT | |
| HOMR | R/W* | Home Reverse | SHORT | |

When one of these fields is set to 1, the motor will decelerate to a stop if already moving, move in the indicated direction (in *dial* coordinates) at the acceleration specified by ACCL and a speed specified by HVEL, until the hardware detects the "home" switch has become active. Then the hardware will do something hardware dependent in response to its "home" command, if any. (The OMS hardware causes the motor to decelerate to a stop.) When the motor stops, the VAL field will be set equal to the RBV field, and the DVAL field will be set equal to the DRBV field. These fields can be set to 1, but setting either field to 0 results in an error. The record sets HOM[F/R] to zero when the homing procedure is either completed or aborted.

| | | | | |
|---|---|---|---|---|
| JOGF | R/W* | Jog motor Forward | SHORT | |
| JOGR | R/W* | Jog motor Reverse | SHORT | |

When one of these fields is set to 1, the motor will decelerate to a stop if already moving, and move in the indicated direction (in user coordinates) at an acceleration specified by ACCL and speed VELO, until the field goes to 0. Then the motor will set VAL to RBV and DVAL to DRBV, decelerate to a stop, and execute a (backlash-corrected, if BDST is nonzero) move to the position at which the field went to 0.

These fields are dangerous when used over channel access, because the motor does not stop moving until a second message is received. If a very busy network should cause that second message to be lost, the motor will travel to its limit switch or hard stop.

| | | | | |
|---|---|---|---|---|
| TWF | R/W* | Tweak motor Forward | SHORT | |
| TWR | R/W* | Tweak motor Reverse | SHORT | |

When one of these fields is set to 1, the record will immediately reset it to 0, and the motor will move (with backlash takeout if BDST is nonzero) by a distance TWV (in user coordinates) at the acceleration specified by ACCL and at speed VELO.

| | | | | |
|---|---|---|---|---|
| TWV | R/W* | Tweak Step Size (EGU) | DOUBLE | This field contains the distance the motor is to move in response to the TWF and TWR buttons. |

## Drive fields

| Name | Access | Prompt | Data type | Comments |
|---|---|---|---|---|
| VAL | R/W* | User Desired Value | DOUBLE | This is the desired position in user coordinates. When this field is written to, DVAL and RVAL will be changed correspondingly, and the motor will move (with backlash takeout if BDST is nonzero) to the newly written position. |
| DVAL | R/W* | Dial Desired Value | DOUBLE | This is the desired position in dial coordinates. When this field is written to, VAL and RVAL will be changed correspondingly, and the motor will move (with backlash takeout if BDST is nonzero) to the newly written position. |
| RVAL | R/W* | Raw Desired Value | DOUBLE | This is the desired position in raw coordinates. When this field is written to, VAL and DVAL will be changed correspondingly, and the motor will move (with backlash takeout if BDST is nonzero) to the newly written position. |
| RLV | R/W* | Relative Value | DOUBLE | When this field is changed, its value will be added to VAL, the field itself will immediately be reset to 0, and the motor record will behave as though the VAL field had been changed directly. |

## Status/readback fields

| Name | Access | Prompt | Data type | Comments |
|---|---|---|---|---|
| RBV | R | User Readback Value | DOUBLE | The current motor position, in user coordinates, from the motor hardware (default), or from the encoder supported by the motor-controller hardware (if UEIP is nonzero), or from the readback link RDBL (if URIP is nonzero). |
| DRBV | R | Dial Readback Value | DOUBLE | The current motor position, in dial coordinates, from the motor hardware (default), or from the encoder supported by the motor-controller hardware (if UEIP is nonzero), or from the readback link RDBL (if URIP is nonzero). |
| DMOV | R | Done moving to value | SHORT | |
| This field is set to 0 when the motor record begins a motion, and remains 0 through any retries and backlash corrections that may be required until the motor record has completely finished that motion, whereupon the field is set to 1. DMOV is guaranteed to execute and post a 1/0/1 pulse when the motor is commanded to move--even if no motion actually occurs because the motor was commanded to move to its current position. | | | | |
| MOVN | R | Motor is moving | SHORT | This field is set to 1 while the record believes that the motor actually is moving. This field is not the inverse of DMOV, since it may go to zero during a complex motion, if that motion includes a momentary stop. |

| | | | | |
|---|---|---|---|---|
| DLY | R/W | Readback settle time (s) | DOUBLE | Delay (in seconds) the time between motor controller done and motor record done (i.e., DMOV). |
| DIFF | R | Difference dval-drbv | DOUBLE | DIFF is the difference, in engineering units, between the desired motor position, and the readback device's report of the current position. RDIF is the same difference in "raw" units (normally, steps). |
| RDIF | R | Difference rval-rrbv | LONG | |
| RRBV | R | Raw Readback Value | DOUBLE | The current position of the motor, encoder, or readback link, as received from whatever source has been selected to provide position information. The units associated with this field depend on the source. |
| RMP | R | Raw Motor Position | DOUBLE | The contents of the hardware's step-count register. This field contains the same information as the dial value, but in steps, rather than in engineering units. |
| REP | R | Raw Encoder Position | DOUBLE | The contents of the hardware's encoder-count register. Ideally, this field contains the same information as the dial value, but in encoder counts, rather than in engineering units. |
| MSTA | R | Motor Status | ULONG | The motor status as received from the hardware. The MSTA bits are defined as follows:<br>1. DIRECTION: last raw direction; (0:Negative, 1:Positive)<br>2. DONE: motion is complete.<br>3. PLUS_LS: plus limit switch has been hit.<br>4. HOMELS: state of the home limit switch.<br>5. Unused<br>6. POSITION: closed-loop position control is enabled.<br>7. Unused<br>8. HOME: if at home position.<br>9. PRESENT: encoder is present.<br>10. PROBLEM: driver stopped polling.<br>11. MOVING: non-zero velocity present.<br>12. GAIN_SUPPORT: motor supports closed-loop position control.<br>13. COMM_ERR: Controller communication error.<br>14. MINUS_LS: minus limit switch has been hit. |
| TDIR | R | Direction of Travel | SHORT | The direction in which the motor is currently traveling (or was most recently traveling), as received from the hardware. If 0, the raw readback value should be decreasing. |
| ATHM | R | At HOME | SHORT | The state of the hardware's "home" switch. If 1, the motor has hit the switch. |
| RCNT | R | Retry count | SHORT | The number of times the motor record has detected failure of the motor to land within the retry-deadband distance of the desired position. |
| MISS | R | Ran out | SHORT | If 1, the motor has failed to land on the desired position more |

| | | | | |
|---|---|---|---|---|
| | | of retries | | than the allowed number of times. This field will be reset the next time the motor succeeds in reaching the desired position. |
| RVEL | R | Raw Velocity | LONG | Speed in steps per second that the motor actually is moving. |
| STUP | R/W | Status Update | RECCHOICE | The STUP field functions as follows;<br>• Valid values for STUP are OFF(0), ON(1) and BUSY(2).<br>• A Channel Access (CA) client writes ON(1) to the STUP field which causes the motor record to set STUP to BUSY(2) and request a single controller status update. After the status is updated the record sets STUP to OFF(0).<br>• CA clients are restricted to writing ON(1) to STUP only when STUP is OFF(0).<br>• It is the responsibility of the user to restrict the frequency (and thus the incurred overhead) at which the CA client writes ON(1) to STUP.<br>With the STUP field it is possible to have another EPICS record periodically write ON(1) to the motor record's STUP field. This would result in continuous, periodic status updates. |

## Servo fields

PID related record fields accept only normalized values (i.e., 0.0 <= value <= 1.0). Before sending them to the motor controller, device support scales the record fields to valid motor controller parameters,. Let the motor controller PID parameters be represented by CKP, CKI and CKD; then the PID coefficients are converted from motor record fields to controller parameters as follows:

For the MM4000; CKP = PCOF, CKI = ICOF and CKD = DCOF.
For all OMS controllers; CKP = 1999.9 * PCOF, CKI = 1999.9 * ICOF, CKD = 1999.9 * DCOF.

Note the following:

- When commanded to move the OMS control law is a PD loop, when it is holding a position it is a PID loop.
- The Proportional Gain cannot be turned off (i.e., set to zero) in an OMS controller. The minimum value is PCOF = 0.00005 (CKP = 0.1). If the user sets PCOF < 0.00005 for an OMS controller, device support silently resets it to it's minimum value of 0.00005.

| Name | Access | Prompt | Data type | Comments |
|---|---|---|---|---|
| CNEN | R/W | Enable control | RECCHOICE | (0:"Disable", 1:"Enable") Enable/Disable closed-loop position control. This field is active only if the GAIN_SUPPORT bit in the MSTA is true. This field is set by both the user and device support. CNEN is set to *Disable* by device support when it detects a motion controller error; e.g. maximum following error exceeded. |
| PCOF | R/W | Proportional Gain | DOUBLE | Valid range; 0.0 <= PCOF <= 1.0 |

| | | | | |
|---|---|---|---|---|
| ICOF | R/W | Integral Gain | DOUBLE | Valid range; 0.0 <= ICOF <= 1.0 |
| DCOF | R/W | Derivative Gain | DOUBLE | Valid range; 0.0 <= DCOF <= 1.0 |

## Alarm fields

| Name | Access | Prompt | Data type | Comments |
|---|---|---|---|---|
| HIHI | R/W* | Hihi Alarm Limit | DOUBLE | |
| LOLO | R/W* | Lolo Alarm Limit | DOUBLE | |
| HIGH | R/W* | High Alarm Limit | DOUBLE | |
| LOW | R/W* | Low Alarm Limit | DOUBLE | |
| HHSV | R/W* | Hihi Severity | GBLCHOICE | Not used. |
| LLSV | R/W* | Lolo Severity | GBLCHOICE | Not used. |
| HSV | R/W* | High Severity | GBLCHOICE | Not used. |
| LSV | R/W* | Low Severity | GBLCHOICE | Not used. |
| HLSV | R/W* | HW Limit Switch Violation Severity | GBLCHOICE | |

## Miscellaneous fields

| Name | Access | Prompt | Data type | Comments |
|---|---|---|---|---|
| PREC | R/W | Display Precision | SHORT | The number of digits to the right of the decimal that are to be displayed by MEDM and other channel-access clients. |
| EGU | R/W | Engineering Units | STRING | String sent to channel-access clients who ask for engineering units. |
| VERS | R | Code Version | DOUBLE | Version number of the recMotor.c code. |
| CARD | R | Card Number | SHORT | For VME based devices (i.e., OMS VME8/44, OMS VME58 and V544) this is the VME card number, derived from the output link. Cards are numbered from zero according to their VME addresses. Oregon Micro Systems series VME8 and VME44 cards occur in the same series, since they are handled by the same driver. Oregon Micro Systems VME58 cards are numbered separately, as are Highland Technology V540 cards. This field is set to -1 for non-VME based device/drivers. |

**Command Primitives**

The following three fields comprise the Command Primitives feature. The command primitive record fields are available to the user to send ASCII command primitives to the motor control board at fixed, predefined, times. Each of the fields is defined as a character string. Consult the motor controller manual for command

protocols. No error checking is done by the motor record or the device driver to insure that the command strings are valid. Each field is *terminated* by the device driver according to the command protocol. Command primitives that result in a response from the motion control board are valid, but the response is not processed. This feature is currently only available with OMS VME8/44/58 or Newport MM4000 device support.

**Device Directives**

- Valid only in the INIT, PREM and POST fields.
- Must be identified by the following;
    - First character of a device directive string must be a '@'.
    - One or more characters followed by a terminating '@'; i.e., device directives must have nonzero length.
    - Valid device directives:
        - In the INIT field; only "DPM_ON".
        - In the PREM field;, only "PUT(*pvname*, *pv-value*, *delay in seconds*)".
        - In the POST field; only "PUT(*pvname*, *pv-value*)".

      See *Driver Power Monitoring* below for the DPM_ON directive. The PUT directive supports changing the value of a database variable. Note that the PREM supports a delay argument, but that POST does not. The *Readback settle time field* (DLY) should be used to create a time delay after the PV specified in the POST field is written.
- Device directive strings are stripped of valid device directives (including @'s) and tested for nonzero length before being sent to the controller. For example, given the INIT string, "@DPM_ON@HE", the device directive @DPM_ON@ is stripped out before HE is sent to the controller.

**Driver Power Monitoring**

- This feature is only available with the OMS VME58 device support.
- The 8 User I/O signals are assigned to the 8 possible VME58 axes as follows:

```
User I/O #0 <> X axis
 "    "    1 <> Y  "
.....................................
"    "    7 <> S  "
```

- Drive-power monitoring defaults to disabled at boot-up. Request enabling drive-power monitoring by entering the device directive "@DPM_ON@" command into the motor record initialization field (i.e., INIT). The INIT field is processed at record initialization (i.e., bootup), hence if there are no errors, drive-power monitoring will be enabled after the next bootup.
- Whenever a request is made to enable drive-power status monitoring, an error check is made (using the VME58 "RB" command) to verify that the User I/O has been configured as an input. The following message will appear in the error log if a configuration error is detected; "Invalid VME58 configuration; RB = 0x####", where "####" is the VME58's response to the RB command.
- When drive-power status monitoring is enabled and a power failure is detected, the device driver will respond by activating the RA_OVERTRAVEL bit in the MSTA. This results in either HLS or LLS being activated depending on the DIR field. In addition, the following message will appear in the error log; "Drive power failure at VME58 card#?? motor#??".

| | | | | |
|---|---|---|---|---|
| INIT | R/W | Startup commands | STRING | Sent at record initialization. |
| PREM | R/W | Pre-move commands | STRING | Sent before every command string that causes motion. |

| | | | | |
|---|---|---|---|---|
| POST | R/W | Post-move commands | STRING | Sent after a complete motion is finished or when an overtravel limit switch is detected. |
| | | | | |
| LOCK | R/W | Soft Channel Position Lock | RECCHOICE | After a "hard" motor initiated move, Soft Channel device support automatically synchronizes a "soft" motor's target position with a "hard" motor's readback position. Soft Channel device support detects that the "hard" motor has initiated the move when it sees the PV pointed to by DINP go false, while the "soft" record's DMOV is true. Automatic synchronization is undesirable in a multi-axis application. Hence, setting the LOCK field to "Yes" disables this automatic synchronization. |
| NTM | R/W | New Target Monitor | RECCHOICE | The requirements on how the motor record processes a new target position while the motor is in motion are as follows;<br>Case #1: The motor record is given a new position, which is in the opposite direction from the current motor motion. If NTM is "Yes", the motor is immediately stopped and given a motion command to the new position. If NTM is "No ", the motor completes the previous move before it is given a motion command to the new position.<br>Case #2: The motor record is given a new position, which is in the same direction as the current motor motion, but the new position is closer to the motor's current position than the original target position. If NTM is "Yes", the motor is stopped after it has gone past the new position; then a command is given to return to the new position. If NTM is "No", the motor completes the previous move before it is given a motion command to the new position.<br>Case #3: The motor record is given a new position, which is in the same direction as the current motor motion, but the new position is further from the motor's current position than the original position. After the motor reaches the original target position and stops, a command is given to the new target position. This case is independent of NTM.<br>NTM defaults to "Yes". It should be set to "No" only for soft motors. Soft motors should be configured with NTM set to "No" to prevent Case #1 above from interfering with "hard" motors using backlash correction. |

## Private fields

| Name | Access | Prompt | Data type | Comments |
|---|---|---|---|---|
| CBAK | None | Callback structure | NOACCESS | |
| LVAL | R | Last User Des Val | DOUBLE | |
| LDVL | R | Last Dial Des Val | DOUBLE | |
| LRVL | R | Last Raw Des Val | DOUBLE | |
| LRLV | R | Last Rel Value | DOUBLE | |
| CDIR | R | Raw commanded direction | SHORT | |

| | | | | |
|---|---|---|---|---|
| PP | R | Post process command | SHORT | |
| MIP | R | Motion In Progress | SHORT | |
| MMAP | R | Monitor Mask | ULONG | |
| NMAP | R | Monitor Mask | ULONG | |
| LSPG | R | Last SPMG | RECCHOICE | (see SPMG) |

# Files, device support

The following table briefly describes all of the files required to implement and use the motor record. The reader is assumed to be familiar with the EPICS Application Source/Release Control document which describes how to build an EPICS application tree into which these files are to be placed, and how to run "makesdr" and "gnumake" to build the record support. These files can all be obtained from the EPICS Software Distribution (in the custom-software section ).

| **SOURCE CODE<br>files to be placed in `&lttop>/&ltapp&gtApp/src/`** | |
|---|---|
| motorRecord.dbd | Database Definition file for motor record. |
| motorRecord.c | Record support for the motor record |
| motor.h | Header included by all .c files |
| | |
| motordevCom.c | Device support common to all motor Record device drivers. |
| motordevCom.h | Device support header file. |
| motordrvCom.c | Driver support common to all motor Record device drivers. |
| motordrvCom.h | Driver support header file. |
| motordrvComCode.h | Local variables common to all motor Record drivers. |
| | NOTE: All of the above files are required for any and all motor Record device drivers. |
| | |
| devOmsCom.c | Device support common to all Oregon Micro Systems device drivers. |
| devOmsCom.h | Device support header file common to all Oregon Micro Systems device drivers. |
| drvOmsCom.h | Driver support header file common to all Oregon Micro Systems device drivers. |
| | NOTE: The above files are required for any and all Oregon Micro Systems device drivers. |
| | |
| devOms.c | Device support for Oregon Micro Systems VME8 and VME44 series boards |
| drvOms.c | Driver for Oregon Micro Systems VME8 and VME44 series boards |

| | |
|---|---|
| drvOms.h | Header included by devOms.c and drvOms.c |
| | |
| devOms58.c | Device support for Oregon Micro Systems VME58 series boards |
| drvOms58.c | Driver for Oregon Micro Systems VME58 series boards |
| drvOms58.h | Header included by devOms58.c and drvOms58.c |
| | |
| devV544.c | Device support for Highland Technology boards. |
| drvV544.c | Driver for Highland Technology boards. |
| drvV544.h | Header included by devV544.c and drvV544.c |
| | |
| drvMMCom.h | Common header included by all Newport Motion Master device drivers. |
| | NOTE: The above files are required for any and all Newport Motion Master device drivers. |
| devMM3000.c | Device support for Newport MM3000. |
| drvMM3000.c | Driver for Newport MM3000. |
| devMM4000.c | Device support for Newport MM4000/40005. |
| drvMM4000.c | Driver for Newport MM4000/40005. |
| devPM500.c | Device support for Newport PM500. |
| drvPM500.c | Driver for Newport PM500. |
| devESP300.c | Device support for Newport ESP300. |
| drvESP300.c | Driver for Newport ESP300. |
| | |
| devSoft.c | Soft Channel device support. |
| devSoftAux.c | Soft Channel device support (Note: CA and record access code cannot both reside in the same file; each defines (redefines) the DBR's. Hence, functions are split between this and the above file base on whether they are record oriented (devSoft.c) or CA oriented (devSoftAux.c). |
| devSoft.h | Header included by devSoft.c and devSoftAux.c |
| | |
| gpibIO.h | GPIB communication include file. |
| gpibIO.c | GPIB interface via Hideos. |

| | |
|---|---|
| serialIO.h | Serial communication include file. |
| serialIOMPF.cc | Serial communication interface via MPF. |
| | |
| drvIM483.h | Common header included by all IMS device drivers. |
| devIM483PL.c | Device support for IM483 in *party line* communication mode. |
| devIM483SM.c | Device support for IM483 in *single mode* communication mode. |
| drvIM483PL.c | Driver for IM483 in *party line* communication mode. |
| drvIM483SM.c | Driver for IM483 in *single mode* communication mode. |
| | |

**MEDM DISPLAY SCREENS**
**files to be placed in `&lttop>/&ltapp&gtApp/op/adl/`**

| | |
|---|---|
| motorx.adl | Small motor-control screen |
| motorx_tiny.adl | Tiny motor-control screen |
| motorx_more.adl | Medium motor-control screen |
| motorx_setup.adl | Setup screen for a single motor |
| motorx_all.adl | Debug screen for a single motor |

These files build `medm` screens to access the motor record. To use one of them from the command line, type, for example

```
medm -x -macro "P=XXX:,M=m1" motorx.adl
```

where `XXX:m1` is the name of a motor record in an IOC.

These files can also be used as related displays from other `medm` screens by passing the argument `"P=XXX:,M=m1"` .

**EPICS STARTUP FILES**
**files to be placed in `&lttop>/ioc/ioc&ltname>/`**

| | |
|---|---|
| st.cmdmv167 | Startup script |

A sample startup script, containing excerpts relevant to motors, is included in the distribution. Here is an annotated copy:

```
############################################################################

      # vxWorks startup script to load and execute system (iocCore) software.

Load standard EPICS software

# the following should be loaded first - BEGIN
ld < targetmv167/
      iocCore
```

```
ld < targetmv167/drvSup
ld < targetmv167/devSup
ld <
      targetmv167/recSup
# the following should be loaded first - END
```

### Load custom EPICS software (including motor support)

```
ld < ../../stdApp/src/O.mv167/stdlib.o
```

### Motor-related debug switches

```
recMotordebug = 0

#OMS vme8/vme44 debug switches
devOMSdebug = 0

      drvOMSdebug = 0

#OMS vme58 debug switches
devOms58debug = 0

      drvOms58debug = 0

#Highland Technology V544 debug switches
devV544ebug
      = 0
drvV544debug = 0
```

### Motor-related databases

```
# load this before loading any databases
dbLoad "../../default.dctsdr"


      #allstop
dbLoadRecords("../../stdApp/gDb/allstop.db","P=tmm:")
#motors

      dbLoadRecords("../../stdApp/gDb/m16.db","P=tmm:")
```

### Specify motor-controller board address, interrupt vector, etc.

```
  omsSetup(nCards, baseAddress, intVectBase, intLevel, pollRate)
oms58Setup(nCards, baseAddress, intVectBase, intLevel, pollRate)

MM3000Setup(nCards, pollRate)
MM3000Config(card#, portName, GPIB#)

MM4000Setup(nCards, pollRate)
MM4000Config(card#, portName, GPIB#)

IM483SMSetup(nCards, pollRate)
IM483SMConfig(card#, portName)

IM483PLSetup(nCommNet, pollRate)
IM483PLConfig(CommNet#, portName)

ESP300Setup(nCards, pollRate)
ESP300Config(card#, portName, GPIB#)
```

| | |
|---|---|
| nCards | the number of cards or controllers; may be less, but not greater than this value. |
| nCommNet | the number of Communication networks (e.g., the number of RS-422 networks for a IM483PL device). |
| baseAddress | the base address of the first card of a series. This must agree with address jumpers on the actual card(s).<br><br>OMS VME8, VME44, and VMEX cards are all of one series, with a base address in the short address space, on a 16-byte (0x10) boundary. (I.e., these cards require 16 bytes each, and must all be addressed contiguously as, e.g., 0xFC00, 0xFC10).<br><br>OMS VME58-x cards are in their own series, also in the short address space, on a 4k-byte (0x1000) boundary.<br><br>Highland Technology V544 cards are in their own series, in the short address space, on a 128-byte (0x80) boundary. |
| intVectBase | the interrupt vector that will be loaded into the first card of a series. Succeeding cards will be loaded with intVectBase+1, intVectBase+2, etc. Set to "0 " to disable interrupt generation; otherwise, stay in the range [64..255]. |
| intLevel | the VME-interrupt level (in [1..6]) the cards will use. This must agree with jumper settings on the cards. |
| pollRate | the rate (in Hz.) at which the driver will interrogate a card when one of its motors is moving. This is also the rate at which channel-access monitors will be posted; to avoid saturating the network with motor-readback information, don't set pollRate much higher than 10 Hz. pollRate must be in the range [1..60]. |
| portName | ASYN port name. |
| link | GPIB link or MPF server CPU location |
| GPIB# | GPIB address. |

```
# OMS VME8, VME44, VMEX driver setup parameters:
omsSetup(5, 0xFC00, 180, 5, 10)

# OMS VME58 driver setup parameters:
oms58Setup(5, 0x2000, 190, 5, 10)

#Start EPICS
iocInit
```

**BACKUP/RESTORE (BURT) REQUEST FILES**
**files to be placed in `&lttop>/&ltapp&gtApp/op/burt/`**

| | |
|---|---|
| settings.req | sample request file to save settings of all motors. Edit this file, supplying names of the motor records whose settings you want saved. (The sample file also saves the states of other records in the sample database, m16.db, that enable or disable the motor records.) |
| yyMotorSettings.req | save settings of a specified motor. This file is `#include` 'd (once for each motor) by |

| | |
|---|---|
| | settings.req. |
| positions.req | sample request file to save positions of all motors. Edit this file, supplying names of the motor records whose positions you want saved. |
| yyMotorPositions.req | save position of a specified motor, This file is `#include` 'd (once for each motor) by positions.req. |

These files tell the backup/restore tool how to save motor settings and positions. To use them from the command line, type, for example

```
burtrb -f settings.req -o myMotorSettings.snap

burtrb -f positions.
      req -o myMotorPositions.snap
```

To restore the motor settings and positions saved by the above commands, type

```
burtwb -f myMotorSettings.snap

burtwb -f myMotorPositions.snap
```

---

# Restrictions

You *must* not change the motor resolution while the motor is moving, but the record currently does not defend itself against this.

There is no way to ask for a reading of the motor's position register, encoder, or limit switches. These are read only while the motor is moving.

Because of the way limit-switch information is conveyed by the OMS hardware, the motor record cannot know the states of both limit switches at the same time. It only knows the state of the switch toward which the motor is moving.

If a move is requested while the motor already is in motion, the original move will still receive a (useless) backlash correction.

Changing MRES (the motor resolution) should change VELO (the speed in engineering units per second), and leave S (the speed in revolutions per second) unchanged. Currently, neither VELO nor S appears to change, but the record behaves as though S had been changed to agree with VELO. A similar thing happens with SBAK and SBAS.

---

# Examples

### Command Primitives

This example assumes the controller is an MM4000 and that only one motor in the controller is moved at a time (MM4000 power on/off effects all motors). If the user wishes the system to perform as follows:

1. Boot-up with motor power off.
2. Turn motor power on before motor motion.
3. Turn motor power off after motor motion.

Then set fields as follows:

- INIT = "MF"
- PREM = "MO"
- POST = "MF"

**Soft Channel**

The following is a simple example of using soft channel device support to allow the user to transform rotary position commands into linear moves. This example assumes the system consist of a linear stage driving a tangent arm. The user commands the system in terms of the angle the arm makes with the vertical. The database converts this angle into a linear position along an axis that is one meter perpendicular to the center of rotation.

The following database implements the system described above. The database consist of four records:

- rotary - a *soft* motor record where the user issues angular position commands in *degrees*.
- convertDriveValue - a *calcout* record that converts the commanded position from *degrees* to *mm*.
- linear - a *hard* motor record using Oms58 device support that accepts linear position commands in *mm.*
- convertReadbackValue - *calcout* record that converts the feedback position from *mm* to *degrees*.

Note the following concerning the example database and Soft Channel device support:

- Jogging a soft channel motor record is **not** supported.
- In the *convertReadbackValue* record, CP is required in the INPA field to get continuous readback updates to the *rotary* record. Otherwise, the *rotary* record will only update at the end of a move.
- The *convertDriveValue* record converts *rotary's* DVAL field from degrees to radians, takes the tangent, converts from meters to millimeters and finally, sends this value to *linear's* DVAL field.
- The *convertReadbackValue* record converts *linear's* DRBV field from millimeters to meters, takes the inverse tangent and finally, converts from radians to degrees.

```
grecord(motor,"$(user):rotary")
{
   field(DTYP,"Soft Channel")
   field(OUT,"$(user):convertDriveValue.A  PP MS")
   field(RDBL,"$(user):convertReadbackValue.VAL  NPP MS")
   field(URIP,"Yes")
   field(STOO,"$(user):linear.STOP  PP MS")
   field(DINP,"$(user):linear.DMOV  NPP MS")
   field(MRES,"0.001")
   field(RRES,"1.000")
   field(PREC,"3")
   field(DHLM,"45")
   field(DLLM,"-45")
   field(TWV,"5")
   field(RTRY,"0")
   field(EGU,"deg.")
}
```

```
grecord(motor,"$(user):linear")
{
   field(DTYP,"OMS VME58")
   field(VBAS,"1.0")
   field(VELO,"25.0")
   field(OUT,"#C0 S0 @")
   field(MRES,"0.001")
   field(PREC,"3")
   field(DHLM,"1000")
   field(DLLM,"-1000")
   field(RTRY,"0")
   field(TWV,"1")
   field(EGU,"mm.")
}
```

```
grecord(calcout,"$(user):convertDriveValue")
{
   field(DESC,"Convert rotary to linear")
   field(CALC,"TAN(A / 57.296) * 1000")
   field(OUT,"$(user):linear.DVAL  PP MS")
}
```

```
grecord(calcout,"$(user):convertReadbackValue")
{
   field(DESC,"Convert linear to rotary")
   field(CALC,"ATAN(A / 1000) * 57.296")
   field(INPA,"$(user):linear.DRBV  CP MS")
}
```

# Design Decisions

This section of the document is an attempt to record the reasoning behind recent motor record design decisions.

### Motor Controller Travel Limits

For those motor controllers that provide their own internal software travel limits (e.g., MM4000), the motor record will only allow the user to set dial travel limits (i.e., D[H/L]LM) that are within the controller's travel limit range, inclusively. In making this design decision it was assumed that an *expert* user would enter the controller travel limits (possibly, through a front panel user interface) that would protect other users from reaching the hard travel limits. Controllers with their own travel limits that are commanded to move outside their valid range respond with an error. Hence, there is no utility in allowing the dial travel limits outside the controller's range.

### Separate EGU's and MRES's between Controller and Motor Record

Device driver support for controllers that have characteristics different from the OMS controllers, such as the Newport MM4000, has raised new issues. One of these issues is whether or not the motor record should set it's EGU and MRES fields from the controller's values. We think it should not.

Background: The MM4000 is a stand-alone controller with it's own front panel, non-volatile memory and power supply. The user can configure the MM4000 entirely from the front panel; including which engineering units will be displayed on the front panel. Once engineering units are selected, these same units will be used by the MM4000 when communicating with a host system (i.e., EPICS). Since a host can query the MM4000 for both the controller's engineering units and it's positioning resolution, it is possible to have the motor record's EGU and MRES fields automatically set based on values stored in the MM4000 controller, but we decided not to do this.

To maximize flexibility, we decided to keep these characteristics separate between the motor record and the controller. This decision allows the motor record EGU field (and consequently, MRES) to be set to an application specific value (e.g., kV) that is not supported by the motor controller.

---

*Suggestions and comments to:*

Ron Sluiter : (sluiter@aps.anl.gov)
Tim Mooney : (mooney@aps.anl.gov)
Joe Sullivan : (sullivan@aps.anl.gov)
Last modified: June 25, 2003