MS-2000 Controller Optimal Alignment Procedures

> ⚠️ This page is largely obsolete, especially details about setting AA which is set at the factory and does not normally need to be changed. For obtaining fastest possible move times see the more up-to-date page here.

MS-2000 controllers with firmware Version 8.0 and later may use the procedures outlined below to obtain optimal performance of the MS-2000 stage and/or Z-drive. Users with earlier firmware versions, or older hardware, should contact ASI for further information regarding upgrade options, or other service issues.
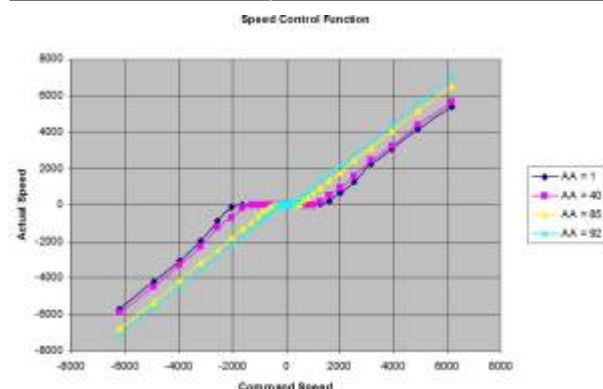
Alignment Overview

The MS-2000 controllers with Version 8.0+ firmware are highly configurable. Settings for the stage running speed, ramp up/down time, error thresholds, servo parameters, driver alignment settings, etc., may all be changed and saved in non-volatile memory. At ASI, we choose an optimal set of parameters suitable for most users before the stage leaves the factory, but our configuration may not necessarily the best for your application, or it is possible that conditions have changed and realignment is in order. Complete optimization consists of four parts. 1) Aligning the drive card to the stage and/or Z-drive motors, 2) Selecting speed and configuration parameters that you wish to use for the controller, 3) Saving the new settings to non-volatile memory so they are available upon power-up, and 4) Testing the new settings for suitability.

Proper calibration of the analog driver is required to get smooth, very slow speed operation of the stage with the digital closed-loop control. Proper drive card alignment also makes the joystick operation as smooth as possible under high magnification.

Drive Card Alignment

The purpose of aligning the analog feed back for the drivers is to reduce the dead zone near zero speed when low voltage is applied to the motor. The chart below shows typical a typical transfer function for various feed back adjustment values.

Click to Enlarge

As you can see, without any feedback, (blue diamonds, AA=1 case) there is a substantial region where supplying a commanded speed to the motor results in no motion whatsoever. The PID loop can overcome this non-linearity and still provide accurate positioning, but the dead zone will give rise to "wind-up" and overshoot in a feedback loop that is well-tuned for the constant slope section of the transfer curve. Increasing the feedback (larger AA number) reduces the size of the dead region and significantly "linearizes" the transfer function near zero. Not surprisingly, problems associated with poor drive card alignment show up when the motors need to move slowly, e.g. when completing a move.

Adjusting the Feedback Alignment (AA)

All of the steps below require a connection to a computer terminal program. Refer to RS-232 Communication

⚠️ WARNING: When doing the alignment of the motors, be sure there is room for the stage to travel a few centimeters and that there is no objective lens in place.

The following example will show alignment of the X axis. Alignment of the Y axis and Z drive channels is analogous.

Before we change anything we need to know the current feedback parameter setting. To find out, issue the command:

```
User's typing shown in this typeface
MS-2000 responses are shown in red
```

```
AA X?
X= 82
:A
```

If you think the feedback is a little low (stage sluggish on landing) you can just increase the present value by a few counts using:

```
AA X=85
:A
```

Always issue the auto-zero command after changing the AA value to re-balance the output amplifiers.

```
AZ X
Zero C:0
C C:0 H:0 L:101
D C:1 H:0 L:109
Bracket H:125 L:109
E C:1 H:125 L:109
E C:1 H:117 L:109
Zerod at: 110
```

The procedure re-balances the amplifier now with the new Feedback value.

If you don't get good results the first time, you can try higher or lower feedback values and re-test the motion.

Too much feedback can result in unstable jerky motion, the motor may tend to buzz, and the AZ command will often produce inconsistent results when repeated several times. Reduce the AA value is you see these symptoms. Too little feedback can result in no motion. With too little feedback, the motors will be trying to get to target but will be unable to do so, resulting in the status letter **M** (*Motor*) or **B** (*Busy*) on the LCD screen staying on for longer periods of time after a move.

Checking the Feedback Alignment (AA)

> ⚠️ This built-in alignment routine hasn't worked since ~2015 or maybe even earlier

The alignment routine contains a built-in self-test that generates the data used in the chart above. To start the test use the command:

```
AA X=500
:A
```

The number 500 triggers this test.

The data from this test are saved in the controller and must be dumped to the screen for you to evaluate. To do this, type the dump command:

```
DU
idmp = 52
pos  err   spd
   -25        0        0
   -31        0        0
   -38        0        0
```

-47	0	0
-58	0	0
-72	0	0
-90	0	0
-112	0	0
-140	0	0
-175	0	0
-218	-3	0
-272	-11	0
-340	-32	0
-425	-46	0
-531	-290	0
-663	-488	0
-828	-686	0
-1035	-990	0
-1293	-1326	0
-1616	-1689	0
-2020	-2172	0
-2525	-2792	0
-3156	-3520	0
-3945	-4430	0
-4931	-5673	0
-6163	-7114	0
25	0	0
31	0	0
38	0	0
47	0	0
58	0	0
72	0	0
90	0	0
112	0	0
140	11	0
175	1	0
218	21	0
272	46	0
340	77	0
425	156	0
531	284	0
663	431	0
828	578	0
1035	932	0
1293	1254	0
1616	1636	0
2020	2063	0
2525	2723	0
3156	3438	0
3945	4353	0
4931	5560	0
6163	6983	0

The numbers show the results for moves in both directions. The first number in each triplet it the speed 'drive' value, the independent variable. The second number, the relative speed the stage is moving. You want to be sure that there is motion at the low drive levels (below +/-500), and that the speed is at least quasi-monotonic. The example shows a typical, well-adjusted stage axis.

If you see very slow, usually zero, movement for drive values < 500, you could increase the AA value.

Setting Speed and Configuration Parameters

The discrete nature of encoders and timing cycles place a few limits on acceptable speed and up/down ramp settings. We do not try to go slower than one encoder count per servo-cycle, so this imposes certain discrete values for minimum slow speeds. It also makes no sense for a ramp to have a change in speed of less than one encoder count per servo-cycle. These constraints are built into the range-checking that is performed whenever either the speed (S) or the ramp time (AC) setting is changed. For instance, if we want to move at about 100 microns/second, we issue the command:

S X=.1 Y=.1

Then, ask for those values to be read back:

```
S X? Y?
: X=0.088110 Y=0.088110 A
```

We see that the controller has adjusted our choices somewhat. You can quickly see many of the controller parameter settings by issuing the info command:

```
I X
Axis Name      : X                 Error Status   :
Input Device   : JS_X      [J]     Motor Signal   :      128
Max Lim        : 110.947 [SU]      Min Lim        : -109.053 [SL]
Ramp Time      :       36 (ms)[AC] Ramp Steps     :        6
Run Speed      :  0.08811 (mm/s)[S] vmax_enc      :        6
dv_enc         :        1          enc_bl_crossovr:       55
Drift Error    : 0.000500 (mm)[E]  enc_drift_err  :        5
Finish Error   : 0.000097 (mm)[PC] enc_finish_err :        1
Backlash       : 0.040000 (mm)[B]  enc_backlash   :      453
Kp             :       20 [KP]     Ki             :        1 [KI]
Kv             :       25 [KV]
Axis Enable    :        1 [MC]     Motor Enable   :        0
CMD_stat       :  NO_MOVE          Move_stat      :   FINISH
Current pos    :  0.00000          enc position   :  8388608
Target pos     :  0.00000          enc target     :  8388608
enc pos error  :       -1          EEsum          :       -5
Lst Settle Time:       54 (ms)     Ave Settle Time:       84 (ms)
```

The parameters that are user-changeable are indicated by the presence of their command shortcut in the listing above. For instance, if you wished to reduce the Drift Error parameter you could issue the command:

```
E X=.0002 Y=.0002
```

Subsequent info requests would show the change to the **Drift Error** as well as to its corresponding parameter **enc_drift_err**, which is the Drift Error expressed in units of encoder counts.

Setting the appropriate error tolerances can have a dramatic effect on stage performance. Landing "on the count" is often possible, but will occasionally cause the controller to hang because mechanical "stiction" just doesn't allow the motor to move such small amounts. Hence, we usually ship controllers with the **Finish Error** set to a distance equal to one encoder count. Once the stage moves to within the **Finish Error** distance of the target, it declares the move finished, clears the busy flag (the **B** on the LCD display), and enters the **NO_MOVE** state. Should the stage subsequently drift away from the target more than an amount specified by the **Drift Error**, the controller will re-energize the motors and bring the stage back to within the Finish Error distance once again.

In some instances, final positioning is very important so you may wish to reduce the **Drift Error** to one or two encoder counts. Doing so will cause the controller to attempt to reposition the stage more frequently. If you don't care so much about accuracy, but are very concerned about the time it takes for the stage to settle to the target, you could increase the **Finish Error** parameter. You should notice a substantial decrease in the settling time.

On stages without linear encoders, we recommend using the built-in anti-backlash routine to improve the repeatability of the stage movement. The Backlash parameter specifies an offset distance from the final target position. A commanded move first moves to a target position offset by the Backlash parameter, and then subsequently moves on to the target. Although this helps to improve the repeatability of commanded moves, for some applications, it will cause more problems than it cures. To turn the Backlash off, just set its value to zero. **(B X=0)**

Setting Servo Parameters

Changing the servo parameters should be done only if drive card alignment still doesn't get the performance that is required from the system. The servo parameters that you may want to adjust are the KP and KI settings that determine the response of the servo loop to trajectory errors during a move. The KP term sets the motor drive proportional to the error, and the KI term sets the motor drive proportional to the time integral of the error. Before changing anything always query the controller to find out the present values of the parameters.

```
KP X?
A: X=50
KI X?
A: X=5
```

In general, increasing the KP parameter increases the stiffness of the motion; the stage more closely follows the desired command trajectory. Increasing the KI parameter fixes persistent errors more quickly, especially finding target at the end of travel. Too much of either parameter can cause instability and overshoots. Here are a few rules of thumb for setting these parameters.

- Start with KI X=0. Increase KP value until unstable motion is detected on long moves. Back off KP value until motion is smooth.
- Start with KI approximately 10% of the KP value obtained above. Make several short

commanded moves and be sure the Busy clears quickly. If not, increase KI.

Once you have made changes, be sure to save the settings as discussed below.

Saving Settings to Non-Volatile Memory

Once you are happy with your settings, you can save them to flash EPROM so they will be used on subsequent start-ups. To save all of the current settings to Flash memory, use the SAVESET command

Testing and Error Checking Servo Motion

Testing the Motion

The MS-2000 controller has several built-in diagnostic capabilities that are useful for troubleshooting difficulties and for tuning the servo motion parameters. It is often useful to see how well the servo motion tracks the theoretical trajectory programmed by the controller for the move. The controller has a built-in buffer that can hold 200 move steps. For best results, restrict testing to a single axis at a time; otherwise, information from multiple axes will be interleaved in the dump buffer. Any motion from any axis will write information into the dump buffer until it is full. To begin a test, first reset the buffer with the command:

DU X [reset the dump buffer]

Then make a short move, e.g.:

M X=12345 [Moves about 1.2 mm] After the move is complete, you can dump the buffer to the screen:

```
DU
idmp = 121
pos   err    spd
 0 , 45 , 2250
 45 , 135 , 6310
 114 , 270 , 11348
 169 , 450 , 15840
 196 , 675 , 19218
 184 , 945 , 21034
 169 , 1215 , 20476
 118 , 1485 , 18464
 59 , 1755 , 16118
 18 , 2025 , 14482
 -1 , 2295 , 13722
 2 , 2565 , 13842
 22 , 2835 , 14646
 27 , 3105 , 14852
 50 , 3375 , 15784
 70 , 3645 , 16600
```
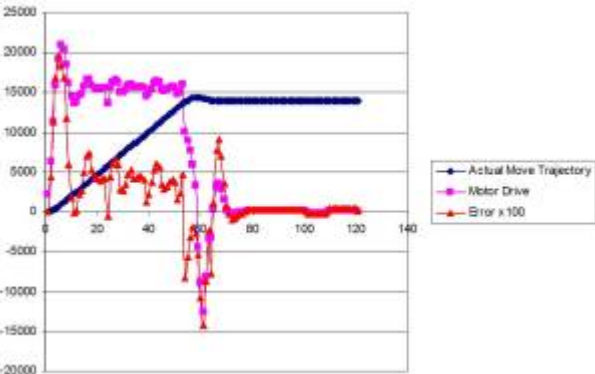
```
 74 , 3915 , 16778
 53 , 4185 , 15950
 45 , 4455 , 15640
 41 , 4725 , 15490
 39 , 4995 , 15418
 42 , 5265 , 15548
 42 , 5535 , 15558
 -5 , 5805 , 13678
 44 , 6075 , 15648
 61 , 6345 , 16342
 66 , 6615 , 16558
 59 , 6885 , 16292
 28 , 7155 , 15058
 27 , 7425 , 15024
 35 , 7695 , 15352
 47 , 7965 , 15842
 53 , 8235 , 16094
 41 , 8505 , 15624
 42 , 8775 , 15674
 44 , 9045 , 15764
 45 , 9315 , 15814
 40 , 9585 , 15624
 12 , 9855 , 14506
 22 , 10125 , 14910
 37 , 10395 , 15518
 52 , 10665 , 16130
 61 , 10935 , 16504
 57 , 11205 , 16358
 34 , 11475 , 15446
 29 , 11745 , 15252
 32 , 12015 , 15380
 38 , 12285 , 15628
 40 , 12555 , 15718
 38 , 12825 , 15646
 16 , 13095 , 14770
 23 , 13365 , 15054
 47 , 13635 , 16024
 -83 , 13790 , 10160
 -57 , 14015 , 8936
 -31 , 14195 , 7720
 -19 , 14330 , 5946
 -27 , 14420 , 3370
 -54 , 14433 , -4422
 -107 , 14343 , -8818
 -142 , 14208 , -12502
 -87 , 14190 , -8072
 -24 , 14145 , -3308
 -77 , 14010 , -3098
 11 , 14010 , 424
 78 , 14010 , 3122
```

```
91 , 14010 , 3664
71 , 14010 , 2880
36 , 14010 , 1488
7 , 14010 , 328
-2 , 14010 , -32
-9 , 14010 , -314
-8 , 14010 , -276
-5 , 14010 , -156
-2 , 14010 , -36
-2 , 14010 , -36
0 , 14010 , 44
3 , 14010 , 120
3 , 14010 , 120
3 , 14010 , 120
3 , 14010 , 120
3 , 14010 , 120
3 , 14010 , 120
3 , 14010 , 120
3 , 14010 , 120
3 , 14010 , 120
3 , 14010 , 120
3 , 14010 , 120
3 , 14010 , 120
3 , 14010 , 120
3 , 14010 , 120
3 , 14010 , 120
3 , 14010 , 120
3 , 14010 , 120
3 , 14010 , 120
3 , 14010 , 120
0 , 14010 , 0
-2 , 14010 , -80
-2 , 14010 , -80
-2 , 14010 , -80
-2 , 14010 , -80
-2 , 14010 , -80
-2 , 14010 , -80
-2 , 14010 , -80
0 , 14010 , 0
4 , 14010 , 160
4 , 14010 , 160
4 , 14010 , 160
4 , 14010 , 160
4 , 14010 , 160
4 , 14010 , 160
4 , 14010 , 160
4 , 14010 , 160
```

```
 4 , 14010 , 160
 4 , 14010 , 160
 4 , 14010 , 160
 1 , 14010 , 40
```

All those numbers may not look particularly useful, but they have a lot of information about the system dynamics in them. Using *HyperTerminal*'s file capture capability, we can save the dumped information to a file and plot the results in a charting program like Microsoft's Excel.

Click to Enlarge

The data shows the move trajectory, the error from desired path, as well as the drive signal given to the motor mover. For this move, there were six ramp-up and down points on the way to full velocity. Backlash of ~450 counts was enabled, the finish error was 1 encoder count, and the drift error was 2 encoder counts. During the time of maximum acceleration, there is also the most error. The short ramp (six cycles) contributes to the error (about 17 microns peak error from the theoretical path). Once the move is under way, the average error settles down to about 50 counts (4 microns). You can see the retrograde motion caused by the anti-backlash routine starting at about the 59th time step. There is a fair amount of overshoot and then the move settles in to find its final position. The move would clear the busy status at about step 77, but then it drifts outside the drift error range and cannot settle for another 45 time steps.

The user is advised that servo tuning can be a painstaking process, so only try new settings if you are unhappy with the default ones shipped from the factory.

Checking Internal Errors

The controller is designed to work with a wide variety of parameter settings, but there certainly could be conditions where the choice of parameters causes motion errors of one sort or another. To aid in diagnosing these problems, the MS-2000 controller has an error buffer of the last 256 error conditions it has encountered since last power up. The error codes can be accessed by the command:

DU Y [dump error codes] The buffer is cleared with:

DU X [clear buffers] Error codes are dumped to the screen with the last error code shown first. The table below lists the meanings of the error codes, as of the date of this publication.

Error Codes for MS2000 and TG1000 Diagnostics

Error codes are dumped to the serial terminal with the last error code shown first using the DU Y command.

If you're looking for command syntax error codes such as N-1, N-2, etc you can find a table at the bottom of the page.

If you're looking for filter wheel error codes, they are listed at the bottom of the page.

DU Y Example Output

```
2DU Y

Adr:2:ZF
        0       0       0       0       0       0       0       0
0       0       0       0
        0       0       0       0       0       0       0       0
0       0       0       0
        0       0       0       0       0       0       0       0
0       0       0       0
        0       0       0       0       0       0       0       0
0       0       0       0
        0       0       0       0       0       0       0       0
0       0       0       0
        0       0       0       0
```

An empty error buffer on card #2.

The table below lists the meanings of the error codes.

Error Code Table	
Error Number	**Error Description**
0	No Error
1-9	OVERTIME – RECOVERABLE. Error caused by competing tasks using the microprocessor. Should be treated as warning.
10-14 †	OVERSHOT – Move overshot the target; happens frequently, not really an error.
15	NEGATIVE LOG – Negative number for Log conversion.
20-23 †	AXIS DEAD – FATAL. No movement for 100 cycles; axis halted.
30-33 †	RUN AWAY – FATAL. Getting further from the target; axis halted. Commonly preceded by 90-93.
34	UPPER LIMIT – Upper Limit reached. (axis unspecific)
35	LOWER LIMIT – Lower Limit reached. (axis unspecific)
36	MOVE INTO UPPER (axis unspecific)
37	MOVE INTO LOWER (axis unspecific)
38	BACK VOLTAGE LIMIT (axis unspecific)
42	CRISP Error
43	CRISP Halted

44	Finish Speed Clamp
45	ADC_LOCK_OOR – Out-of-range error for locked servo - causes unlock.
46	ADC_FOLLOW_ERR – Error attempting to follow an analog ADC input.
47	Servo Locked
48	Task Loop Overtime
49	Low Light
50-53 †	ENCODER ERROR OVERFLOW – FATAL. Error term so large that move intent is indiscernible; axis halted.
54	I2C Poll Error
55	EPROM NO LOAD – Saved-settings on EPROM not loaded, compile date mismatch.
56	I2C Busy Error
57	I2C Write Error 1
58	I2C Read Error 1
59	I2C No Acknowledgement Error , followed by I2C Chip Address
60-64 †	ADJUST-MOVE ERROR – Failed to clear 'M' more than 18 times in a row. FATAL
65-69 †	ADJUST-MOVE WARNING - Failed to clear 'M' less than 19 times in a row, but eventually cleared. RECOVERABLE.
85	SCAN LOST PULSES – During a scan, missing pulses were detected.
86	SCAN INCOMPLETE – During a scan, terminated before completing the row.
87	TTL Report Buffer Overrun
90-94 †	ERROR_LARGE – RECOVERABLE. Motor set to FULL SPEED; hope to catch up (e.g. speed set faster than possible).
100-104 †	INDEX NOT FOUND
105	Buffer Overrun
106	Buffer Underrun
110	SPIM Loop Time, followed by the SPIM State and the Task ID
120-124 †	Encoder E Flag
140	ADEPT High Voltage low
141	ADEPT I2C Dead
142	PIEZO READ POS
143	PIEZO WRITE POS
144	PIEZO MOVE ERR
145	PIEZO READ POS1
146	PIEZO INIT
147	PIEZO POS ERROR
148	Autofocus 200um safety limit Encountered
149	I2C_BAD_BUSY ERROR
150	READ_I2C_ZERO_POT_ERR1
151	READ_I2C_ZERO_POT_ERR2
152	READ_I2C_FEEDBACK_POT_ERR1
153	READ_I2C_FEEDBACK_POT_ERR2
154	READ_I2C_ALIGNSET_ERR1
155	READ_I2C_ALIGNSET_ERR2
156	WRITE_I2C_ALIGNSET_ERR1
157	WRITE_I2C_ALIGNSET_ERR2
158	READ_BYTE_I2C_U15_ERR1

159	READ_BYTE_I2C_U15_ERR2
160	READ_BYTES_I2C_U15_ERR1
161	READ_BYTES_I2C_U15_ERR2
162	WRITE_BYTE_I2C_U15_ERR1
163	WRITE_BYTE_I2C_U15_ERR2
164	WRITE_BYTES_I2C_U15_ERR
165	WRITE_I2C_ZERO_POT_ERR1
166	WRITE_I2C_ZERO_POT_ERR2
167	WRITE_I2C_FEEDBACK_POT_ERR1
168	WRITE_I2C_FEEDBACK_POT_ERR2
169	DC_PORT_SETUP1_ERR
170	DC_PORT_SETUP2_ERR
171	DC_PORT_SETUP3_ERR
172	I2C_CALIBRATION_ERR
173	I2C_AXIS_ENABLE_ERR1
174	I2C_AXIS_ENABLE_ERR2
175	I2C_AXIS_MUTE1_ERR
176	I2C_AXIS_MUTE2_ERR
177	I2C_READ_TTL_ERR1
178	I2C_READ_PIEZO_DAC_ERR1
179	I2C_READ_PIEZO_DAC_ERR2
180	I2C_WRITE_PIEZO_DAC_ERR
181	I2C_READ_ERR2
182	MS_I2C_IDLE_ERR
183	MS_I2C_STOP_ERR
184	I2C_WRITE_ERR2
185	I2C_WRITE_ERR3
186	I2C_WRITE_ERR4
187	I2C_WRITE_ERR5
188	I2C_WRITE_ERR6
189	I2C_WRITE_ERR7
190	I2C_WRITE_ERR8
191	I2C_WRITE_ERR9
192	I2C_WRITE_ERRA
193	I2C_WRITE_ERRB
194	I2C_WRITE_ERRC
195	I2C_NACK_ERR3
196	I2C_NACK_ERR4
197	I2C_READ_ERR3
198	I2C_READ_ERR4
199	I2C_READ_ERR5
200	I2C_READ_ERR6
201	I2C_READ_ERR7 Tunable lens temp sensor I2C_NACK error
202	I2C_READ_TTL_ERR2
203	I2C_NACK_ERROR
204	ERR_TTL_READ_TIMEOUT

205	ERR_TTL_MISMATCH I2C bus error.
206	I2C_WRITE_ERRD
207	I2C_WRITE_ERRE
208	I2C_READ_ERR8
209	I2C_READ_ERR9
210	I2C_WRITE_ERRF
211	I2C_WRITE_ERR10
212	I2C_WRITE_ERR11
213	I2C_WRITE_ERR12
214	I2C_WRITE_ERR13
215	I2C_WRITE_ERR14
216	I2C_WRITE_ERR15
217	READ_BYTE_I2C_U15_ERR3
218	READ_BYTE_I2C_U15_ERR4
219	READ_BYTE_I2C_U15_ERR5
220	READ_BYTE_I2C_U15_ERR6
221	I2C_BUS_ERROR_RD
222	I2C_BUS_ERROR_WR
223	I2C_WRITE_ERR16
224	I2C_WRITE_ERR17
225	RDBYTE_0
226	RDBYTE_1
227	RDBYTE_2
228	RDBYTE_3
229	RDBYTE_4
230	RDBYTE_5
231	RDBYTE_6
233	RDBYTE_7
234	RDBYTE_8
235	RDBYTE_9
236	READ_I2C_ALIGNSET_ERR3
243	I2C_NACK_WRITING
244	LIMIT_NOT_FOUND
254	REPORT_PSD
255	WRITE_DAC_ERROR0
256	WRITE_DAC_ERROR1
257	WRITE_DAC_ERROR2
258	I2C_DIP_SWITCH_ERR0
259	I2C_DIP_SWITCH_ERR1
260	I2C_DIP_SWITCH_ERR2
261	WRITE_DAC_ERROR3
262	I2C_DIP_SWITCH_ERR3
263	WRITE_I2C_ALIGNSET_ERR3
264	LCD_STATE_ERROR
300	Autofocus Scan failed due to insufficient contrast
301	Autofocus Calibration Failed

302	Clutch Disengaged, Engage clutch to do Autofocus
305-311	Source of last Reset , Very common there will always be one preset on controller start. 305(External VDD Mon),309(Software cmd or reset button),307(Missing Clk), 306(Onboard VDD Mon)
500	TX1_OVERRUN
501	TST_ERROR0
502	TST_ERROR1
503	TST_ERROR2
504	TST_ERROR3
505	TST_ERROR4
600-604	FEEDBACK_POT0_TEST
610-614	ZERO_POT0_TEST
620-624	ALIGNSET0_TEST
630-634	ENCODER_TEST
635	DIP_SWITCH_SELF_TEST
636	PIEZO_DAC_TEST
640-641	FW_DEAD_ERROR
650-651	FW_ABSENT_ERROR
665	I2C_RECOVER_SUCCESS
666	I2C_RECOVER_FAILED
670	TTL1_LIMIT_X - TTL1 went "HIGH", halted X axis
671	TTL1_LIMIT_Y - TTL1 went "HIGH", halted Y axis
672	TTL1_LIMIT_Z - TTL1 went "HIGH", halted Z axis
673	TTL1_LIMIT_F - TTL1 went "HIGH", halted F axis
65535	10 MINUTE ELAPSED TIME MARK

† The last digit indicates the axis number that is in error in controller ordering starting from 0. E.g. on three-axis units X=0, Y=1, and Z=2; on single-axis MFC units, Z=0.

FATAL errors cause the controller to halt motion on the axis that has the error. A commanded move will not be completed to the desired precision if a FATAL error occurs.

RECOVERABLE errors do not stop the controller from attempting to complete a commanded move. Large numbers of recoverable errors should be taken as a warning. Frequent servo errors (numbers 90-92) often mean that the speed is near or exceeding the stage maximum. Frequent overtime errors (numbers 1-9) often mean that competing processes, such as over-frequent serial status requests, are using too much CPU time.

Legacy CRIFF Error Codes

These error codes are for legacy CRIFF devices.

Error Number †	Error Description
237	I2C_WRITE_INT_ERR1
238	I2C_WRITE_INT_ERR2
239	I2C_WRITE_OP_CODE_ERR1

Error Number †	Error Description
240	I2C_WRITE_OP_CODE_ERR2
241	I2C_READ_INT_ERR1
242	I2C_READ_INT_ERR2
248	CRIFF_I2C_ERR1
249	CRIFF_I2C_ERR2
250	I2C_READ_FAIL

Other Error Types

Command Syntax Error Codes

When a command is received that the controller cannot interpret, for one reason or another, an error is returned in the following format:

```
:N-<error code>
```

The error codes are as follows:

Command Syntax Error Codes	
:N-1	Unknown Command *(not issued in TG-1000)*
:N-2	Unrecognized Axis Parameter *(valid axes are dependent on the controller)*
:N-3	Missing Parameters *(command received requires an axis parameter such as x=1234)*
:N-4	Parameter Out of Range
:N-5	Operation Failed
:N-6	Undefined Error *(command is incorrect, but for none of the above reasons)*
:N-7	Invalid Card Address
:N-8 ... :N-10	Reserved
:N-11 ... :N-20	Reserved For Filterwheel
:N-21	Serial Command Halted *(by the HALT command)*
:N-30 ... :N-39	Reserved

Filter Wheel Error Codes

A list of recent errors are available through the Dump Errors (DE) command.

Error Codes		
Name	Number	Notes
OVERSHOT	10	Plus axis number.
DRIVER_HOT	25	
RUN_AWAY	30	Plus axis number, followed by mode. See mode table below.
LARGE_OVERFLOW	50	Plus axis number.
SLIP_ERROR	70	Plus axis number.
PAST_ERROR	80	Plus axis number.
ERROR_OVERFLOW	90	Plus axis number.

KP_ERR_OVERFLOW	100	Plus axis number.
KV_SPD_OVERFLOW	110	Plus axis number.
KI_SUM_OVERFLOW	120	Plus axis number.
ERR_INTEGRAL_OF	125	Plus axis number.
KD_TRM_OVERFLOW	130	Plus axis number.
KA_TRM_OVERFLOW	140	Plus axis number.
ENCODER0_ERROR	160	Plus error number.
ENCODER1_ERROR	170	Plus error number.
MOVE_OVERTIME	180	Plus axis number.

2016/03/14 20:45 · vik

serial, tech note, ms2000, tiger

ms2000, tiger, tech note

From:

http://www.asiimaging.com/docs/ - **Applied Scientific Instrumentation**

Permanent link:

http://www.asiimaging.com/docs/ms2000_optimal_alignment_procedures

Last update: **2024/08/30 17:24**