WORKING PAPER 68

X-Y TABLE USER'S MANUAL

by

NOBLE LARSON

Massachusetts Institute of Technology

Artificial Intelligence Laboratory

May, 1974

Abstract

This working paper describes the mini-robot group's X-Y table and associated hardware.

Work reported herein was conducted at the Artificial Intelligence Laboratory, a Massachusetts Institute of Technology research program supported in part by the Advanced Research Projects Agency of the Department of Defence and monitored by the Office of Naval Research under Contract number N00014-70-A-0362-0005.

Working Papers are informal papers intended for internal use.

X-Y TABLE

I. General Description

A numerically controlled "X-Y" positioning table has been interfaced to the 11-40 processor for use in the Micro-Automation lab. The table consists of a moveable cast aluminum platform on a heavy base. The platform can be moved throughout a 6" interval in either of two horizontal directions, referred to subsequently as the "X-direction" and the "Y-direction". The motion is accomplished by two Fujitsu 109 stepping motors, the unit of rectilinear motion, or step being 1/1000th of an inch.

The interfacing hardware allows the platform to be moved forward or backward in either direction by a programmable amount and at a programmable rate. The platform, however, cannot be moved outside of the 6" by 6" area mentioned above, on account of limit switches, which when triggered, prevent any further motion in a particular direction. Motion is initiated by the program providing a count, which is interpreted as a number of steps that the table is to be moved in a particular direction, and a rate. The program is then free to do something else. Upon completion of the motion, either by exhausting the count or by triggering of a limit switch, the hardware will reset a status bit and attempt to interrupt the processor. Programming-wise and hardware-wise the mechanisms for effecting motion in the two directions represent completely independent channels.

II. Programming Information

Software communicates with the interface hardware through 4 memory locations.

164000 X Count register
164002 X Rate/Status register
164004 Y Count register
164006 Y Rate/Status register

The count register is a buffer for one's complement representations of numbers of steps the platform is to be moved in the corresponding direction. As the table moves this count gets continuously decremented [one's complement] by hardware until the count is exhausted. The terminal value is 777777 [one's complement zero]. The rate/status register has several sections with the following intepretation:

bits 0-11 constitute an encoded rate value [See Table I];
bit 12 is the busy bit;
bit 13 is the forward/backward bit;
bit 14 the limit/reset bit; and
bit 15 is the interrupt enable bit.

The rate value bits can be loaded and read by software, and are never cleared or otherwise altered by hardware. The busy bit is set by software to initiate motion of the platform in the corresponding direction. Depending on whether the forward/backward bit is set or reset, the motion initiated will be forward or backward respectively. Upon either completion of the motion or triggering of a limit switch, the busy bit is reset by hardware, and an interrupt condition will occur in the channel. If the interrupt enable bit is set, an interrupt request will be made on the bus at level 4.

The interrupt vector for the X-channel is at 340.
The interrupt vector for the Y-channel is at 344.

After having been interrupted or having tested the busy bit and found it reset, software can determine the reason for the interrupt by testing bit 14. If it is on, a limit switch was hit, otherwise, the reason was normal completion of motion. In the event of the former, the state of the forward/backward bit indicates which way the platform had been moving, and therefore determines which limit switch was triggered.

Once a limit switch has been hit, it is necessary to issue a software reset to the channel involved, so as to clear certain conditions in the hardware. Unless this is done it will not be possible to back the platform out of the limit switch. Software reset is accomplished by writing a one in bit 14. It must be understood that status bit 14 refers to two completely different signals depending on whether it is read or written. Reading it, as stated above, gives the value of a signal which tells if a limit switch is depressed. Writing a one in it, however, causes a reset pulse to be issued to the channel. Moreover, due to the way bit-set instructions are implemented in hardware, it is not possible to use them to alter the rate/status register when the platform is depressing a limit switch. Instead, the full word move instruction must be used [Byte moves have not been implemented in the table hardware].

The standard procedure for moving the platform in a given channel is as follows: The one's complement count is loaded into the count register. Then the rate/status register is loaded with the proper value for the desired rate, direction, and interrupt enabling. The busy bit can be set along with the other bits [With, say, the same move instruction], or it can be set subsequently with a bit-set instruction [Unless the platform is depressing a limit switch]. As long as one avoids running the platform into one of the limits, it is possible to set up a rate and an interrupt enabling in a channel and then move the platform back and forth in that channel using only move instructions to reload the count register, and bit-sets to control the busy bit and forward/backward bit.

Table I

Rate Counter [Octal]	PPS [Decimal]
1000	67.
2000	78.
3000	94.
4000	117.
5000	156.
6000	234.
7000	469.
7100	536.
7200	625.
7300	750.
7400	938.
7500	1250.
7550	1579.
7600	1875.
7650	3000.
7700	3750.
7704	4000.

Rate Counter = -(240000./PPS)

III. Mechanical Information

Design Components DC-66 X-Y Positioning Table:

6" by 6"	Motion
10" by 10"	Work Surface
5"	Height
50 lb	Weight
15 arc-sec	Perpendicularity

.00015"	Repeatablility
.0004"	Linear Accuracy
.001"	Step Size [For 2.25 degree shaft rotation]

[Icon Motor Translators and Buffer Amplifiers 601-TR's are used to drive the motors].

Fujitsu Pulse Motor 109 [Specifications]:

Angular increment:	2.25 degrees
Steps per revolution:	160.
Maximum stepping rate:	8000 PPS

[However, the motors driving the table should not be driven faster than 4000 PPS.]

PPS	Torque [lb-in]
0.	2.6
1000.	3.3
2000.	2.8
4000.	1.9
8000.	1.2

Power: .05 hp [at 8000 PPS]

Weight: 3.3 lbs.

Electrical:	R = .4 ohms [One winding]
	L = 1.5 mH [One winding]
	I = 3.5 amp [Per active phase]

Switching frequency of coils = 1/10 pulse rate

Inertia [Calculated]:

```
Rotor:                          From motor specs
Lead-screw       ( .8 lbs ): J = m*r*r/2   ( r = .3" )
Reflected table  ( 20 lbs ): J = m*p*p     ( p = .16"/2π )

Rotor:             .000030 lb-in/sec/sec
Lead-screw:        .000085 lb-in/sec/sec
Reflected table:   .000035 lb-in/sec/sec

Total Inertia:     .000150 lb-in/sec/sec
```

[The actual total inertia may be a bit higher than this]

Natural Oscillations [Estimated]

Let Je be the inertia in addition to that of the rotor.

t[2ph] = .95*SQRT[Je+.000045]   t[3ph] = .70*SQRT[Je+.000045]

Expected in our case: Je+.000045 = .00015

So: t[2ph] = 11.5 ms   t[3ph] = 8.6 ms

[Damping is much stronger with 3 phases on than with 2.]

Stiffness [Estimated]:

$\omega = 2\pi/t$
$\omega = \mathrm{SQRT}[\, k/J]$
$k = J\ [2\pi/t]^2$

k[2ph] = 45 lb-in/radian
k[3ph] = 80 lb-in/radian

k[2ph] = 1.8 lb-in/step
k[3ph] = 3.0 lb-in/step

[A 2:3 ratio is to be expected]
[These figures are consistent with torque figures.]

Single Step Time [Estimated]:

Angular acceleration: $\dot{\omega} = T/J$

The angular motion of the shaft in time t: $\theta = \dot{\omega}[t/2]^2$

So: t = 2*SQRT[θJ/T]

T = 3 lb-in
J = .00015 lb-in/sec/sec
θ = $2\pi/160$ = .0392 radians

So: t = 2.8 ms

[This agrees with 1/4 to 1/2 cycles of oscillation.]

Multiple Step Time [experimental]:

n	t [ms]	t/n	distance
1	2.6	2.6	.001 "
3	6.5	2.2	.003
7	15	2.1	.007
15	30	2.0	.015
31	60	1.9	.031
63	100	1.6	.063
127	180	1.4	.127

These measurements are dependent on gain adjustments in the pulse ramping [buffer] modules. The present settings are conservative: to achieve reliability at the expense of speed.

When the busy bit goes off the platform is within +2 or -2 steps.

Maximum Start-Stop Rate [Torque = .5 in-lb]:

PPSO = 16./SQRT[Je + .00045] = 1300. PPS for our case.

Acceleration Time Constant [Torque= .87 in-lb] [To 8000 PPS]:

t[accel] = 500[Je+.00003]
= 500[.00015] = 75 ms [for our case]

Deceleration Time Constant [Torque= 0 in-lb] [From 8000 PPS]:

t[decel] = 500[Je+.00003]
= 500[.00015] = 75 ms [for our case]

[For lower top speeds, time constant can be less.]

The above figures are theoretical maximum values. It is doubtful that one can use such low time constants and such high start-stop pulse rates in actual practice.

IV. Software

The assembler directive

.MCALL .TABLE

will define a macro called .TABLE which, when called, expands into a set of subroutines for moving the x-y table. These routines are called using the convention JSR PC,SUBR. The table subroutines are:

CALTBL calibrates the x-y table and leaves it in position (0,0)

VELTBL sets up the velocity for the next table movement. R0 should contain the velocity for x and R1 should contain the velocity for y

ABSTBL moves the table to the absolute location (x,y), where x is contained in R0 and y is contained in R1

RELTBL causes the table to move relative to its current location. The x and y in R0 and R1 respectively are taken as offsets for the relative motion and are preserved so that successive calls of RELTBL will reference them.

Note: neither ABSTBL nor RELTBL wait for the table to finish moving. Neither should be called if there is a chance that the table is in motion without first calling WTTBL.

WTTBL waits for the table's motion to finish. WTTBL will take a skip return if the table motion completes normally (without running into a limit stop). If a limit stop is encountered, WTTBL will take a non-skip return. Thus:

```
JSR PC,WTTBL
(error return)
(normal return)
```

WHRTBL returns the table's x position in R0 and its y position in R1.

NOTE: These macros will protect the user from moving the table to a negative position; motion will stop at zero, and the table will not have been decalibrated. Similarly, attempting to move the table too far forward in either x or y will result in a cessation of the table's motion without running into the physical limit stops or decalibrating the table. WTTBL will take the error return whenever such a premature stoppage occurs. Note also that all coordinates kept by the .TABLE routines are relative to the calibration point, and thus CALTBL always should be the first routine called.