NASA CR - 132626

A STUDY OF DIGITAL GYRO COMPENSATION LOOPS

Final Report

April 1975

Prepared under Contract No. NAS1-13088 by
TELEDYNE SYSTEMS COMPANY
Northridge, California

for
NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

ABSTRACT

The primary objective of this program was to demonstrate the feasibility of replacing existing state-of-the-art analog gyro compensation loops with digital computations. This objective was realized during the course of the program.

A breadboard design was established in which one axis of a Teledyne tuned-gimbal TDF gyro was caged digitally while the other was caged using conventional analog electronics. The digital loop was designed analytically to closely resemble the analog loop in performance. The breadboard was subjected to various static and dynamic tests in order to establish the relative stability characteristics and frequency responses of the digital and analog loops. Several variations of the digital loop configuration were evaluated. The results were very favorable - it appears that digital caging is indeed a practical approach.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (Continued)

I. INTRODUCTION

Background

In January 1974 Teledyne Systems Company completed under contract to NASA Langley Research Center an "Investigation of the Application of Two-Degree-of-Freedom Dry Tuned-Gimbal Gyroscopes to Strapdown Navigation Systems"*. During the course of that study an entirely new approach to the control and compensation of the TDF gyro, as well as accelerometers, was suggested and studied. This technique was based upon the use of "all-digital" sensor compensation loops and control functions.

An analysis of the hardware content and costs of present day inertial navigation systems showed that a significant portion, on the order of 30%, of the system, consists of analog and computer interface electronics. The advent of newly available, versatile, digital microcomputers and high-speed analog-to-digital and digital-to-analog converters now makes it practical to replace essentially all of the analog electronics by functionally equivalent digital computations.

Figure 1 shows a functional block diagram of a conventional strapdown inertial navigation system mechanization. Both the sensor compensation functions and the spin motor and pickoff excitations are generated by analog electronics which operate independently of the digital processor. The sensor torquing currents are fed through precision resistances to develop voltages which are proportional to angular rates and accelerations. These voltages are then converted into digital numbers for use in the navigation and attitude equations which must be solved by the computer.

Figure 2 shows the corresponding block diagram for a strapdown system employing digital sensor control and compensation. Here all of the compensation loop servo functions are mechanized as digital computations in the processor. The basic inputs to the A/D converter are the sensor pickoff signals rather than the restoring signals. The torquing signals are computed as digital numbers, converted to analog signals and, after power amplification, used to restore the sensors. These same digital torquing quantities may be used directly in the strapdown attitude and navigation computations as well. (In an optional configuration the actual torquing signals are also passed through the A/D converter for use in the attitude and navigation equations.)

*The final report of this study was published as NASA CR-132419

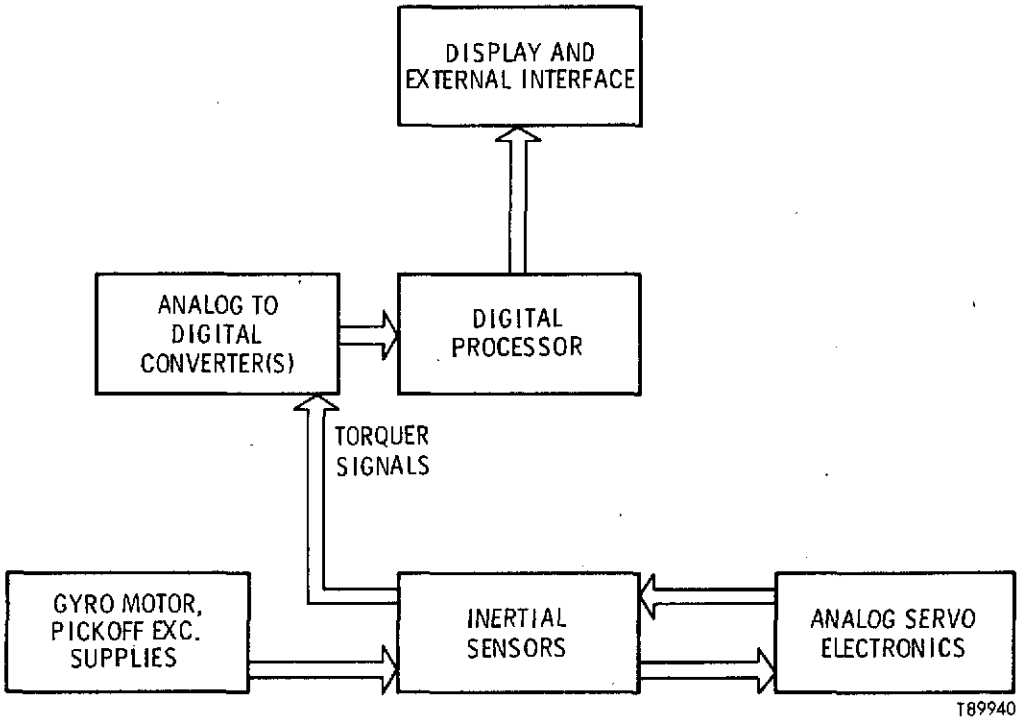


Figure 1. Functional Block Diagram of Conventional Strapdown Mechanization

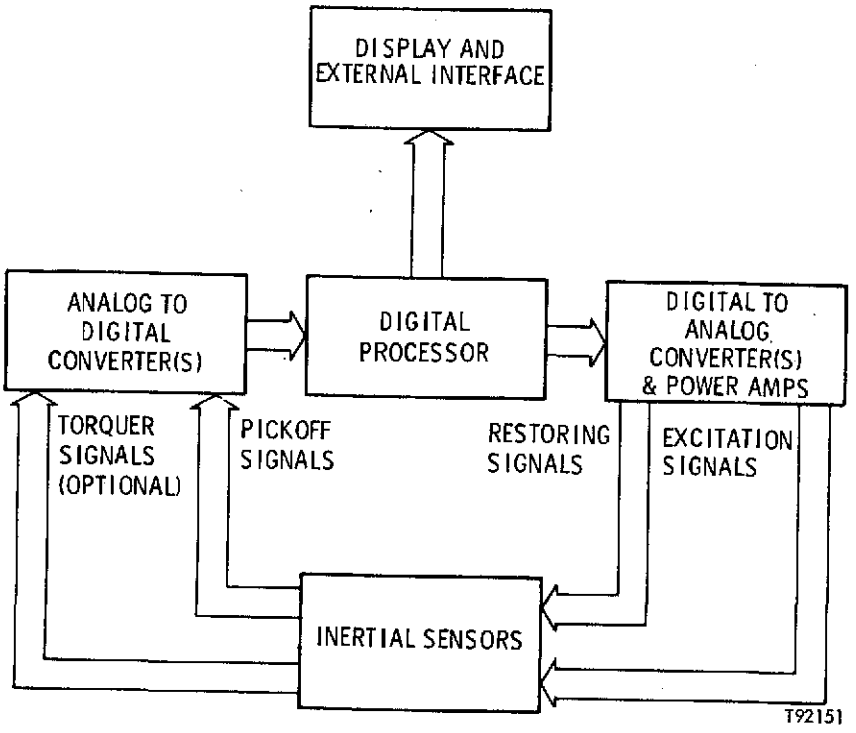


Figure 2. Functional Block Diagram of "All-Digital" Strapdown Mechanization

What is gained by such an "all-digital" approach is that the presently cumbersome and necessarily specialized analog and digital interface electronics can be virtually eliminated, being replaced by digital computations (software) performed in the computer hardware which is, in any case, required for the attitude and navigation computations. This results in substantial savings in cost, size, weight, and power consumption while simultaneously increasing the reliability of the system by reducing the component count.

Objective

The objective of this program was to determine the feasibility of replacing present state-of-the-art analog circuits in strapdown tuned-gimbal gyro compensation circuits with their digital counterparts. This was accomplished by designing appropriate compensation loops for the dry tuned TDF gyro, selecting appropriate data conversion and processing techniques and algorithms, and, using existing laboratory equipment, breadboarding the design for laboratory evaluation.

The principal area of engineering design involved in the program was the determination of the specific software requirements for closing the instrument loops and performing the required compensation. Specific requirements relative to processing rate, word length, computer time and memory utilization were established in order that corresponding requirements for a full complement of strapdown sensors may be readily extrapolated. Additionally, appropriate analog-to-digital and digital-to-analog conversion designs were established and techniques selected which were capable of meeting these requirements.

An available dry tuned-gimbal two-degree-of-freedom gyroscope was utilized together with an electronic circuit breadboard and existing laboratory hardware to implement the digital control loop. Selection of hardware and techniques was based on the anticipated availability of proven hardware in the late 1970's. The breadboard design was evaluated using standard test and evaluation procedures.

This final report summarizes the test configuration and the program results which were obtained.

II. ANALYSIS

The breadboard design which was established for this study, consisted of a dry-tuned two-degree-of-freedom (TDF) gyroscope with one axis being caged in a conventional manner using simplified existing "state-of-the-art" analog compensation. The second axis was caged using a digital compensation loop designed to emulate the performance of the analog compensation. Figure 3 shows a simplified block diagram of the breadboard design. An analysis of the two caging techniques employed in the design is presented in the following paragraphs.

Analog Compensation Design

The general Teledyne gyro caging loop configuration for conventional analog compensation is shown in the simplified block diagram of Figure 4. Observe that both direct and cross axis compensation is employed in this design in order to increase the loop bandwidth and reduce the rotor hangoff during acceleration inputs. In this block diagram θ_X and θ_Y represent the gyro pickoff angles, T_X and T_Y the gyro torques, P_X and P_Y the precessional torques resulting from rate inputs ω_X and ω_Y, and M_X and M_Y the rebalance torques provided by the compensation loops.

A more detailed block diagram of the compensation portions of the loop for the current Teledyne design is shown in Figure 5. This block diagram shows the actual transfer functions which are implemented in the analog electronics as well as the pickoff and torquer transfer characteristics. A derivation and analysis of this compensation design is included in NASA CR-132419. The pole-zero locations for the compensation electronics using this design are shown in Figure 6.

Concurrently with the digital loop demonstration program Teledyne has been involved in an extensive redesign effort on its conventional analog compensation electronics. This effort has as its goals improved loop performance and simpler loop design. Although work is continuing in this area, it has been found that excellent performance can be obtained with relatively simple analog transfer functions. The pole-zero configuration for one such simple loop is shown in Figure 7. This configuration provides a baseline for much of the digital loop design work which was performed during the program.

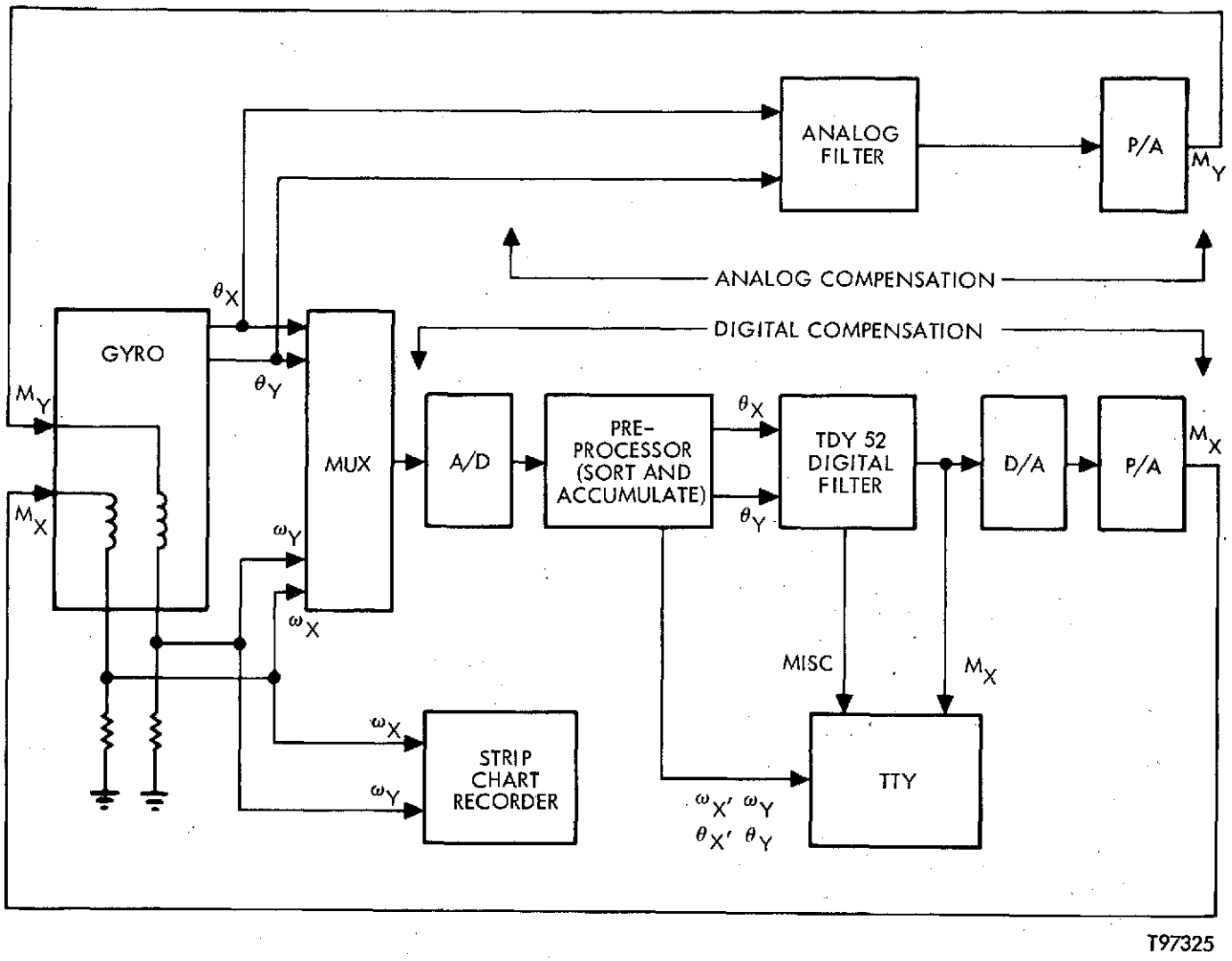


Figure 3. Block Diagram of the System Used in the Digital Gyro-Caging Study

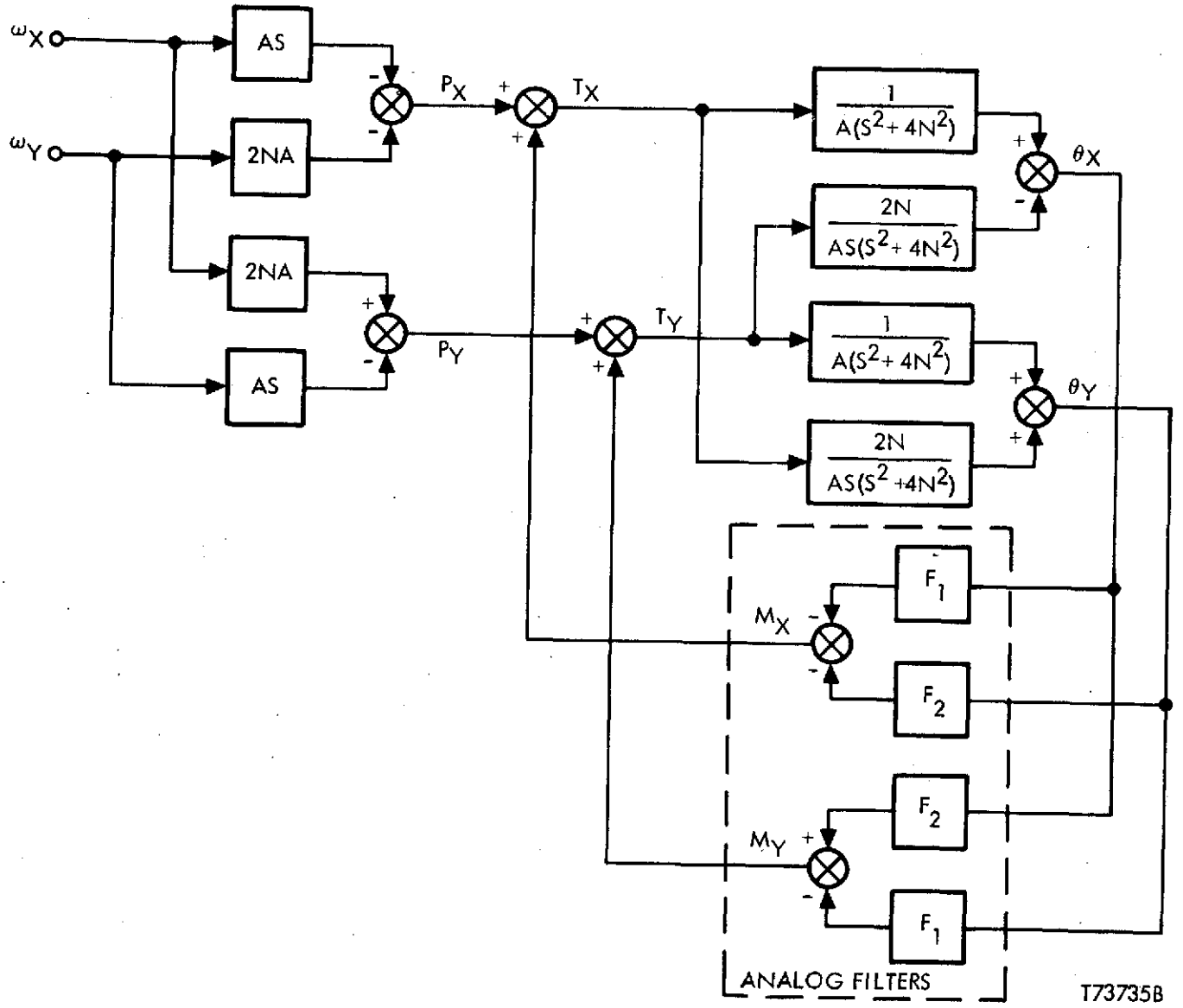


Figure 4. Block Diagram of Gyro and Caging Electronics

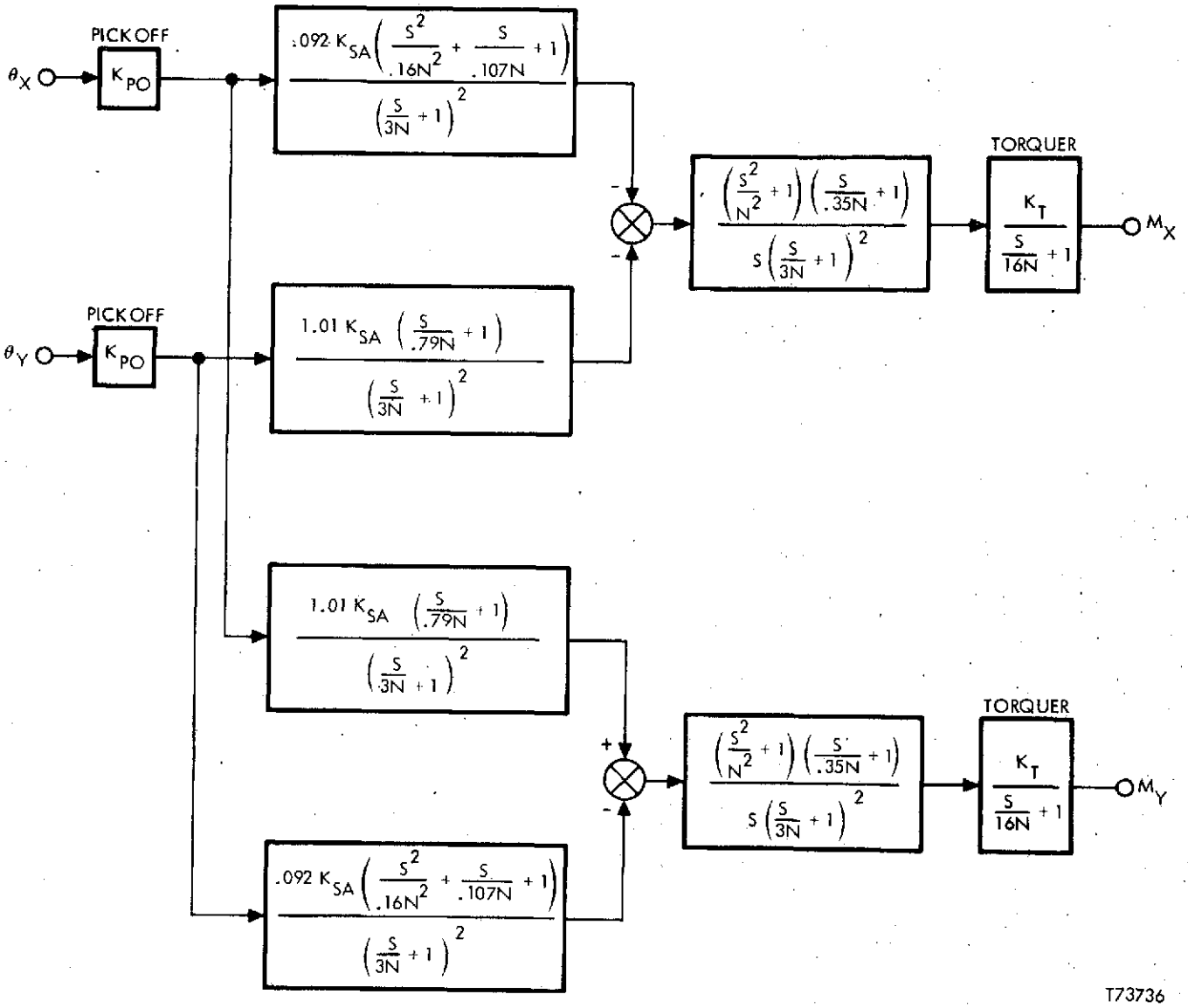


Figure 5. Caging Loop Mechanization

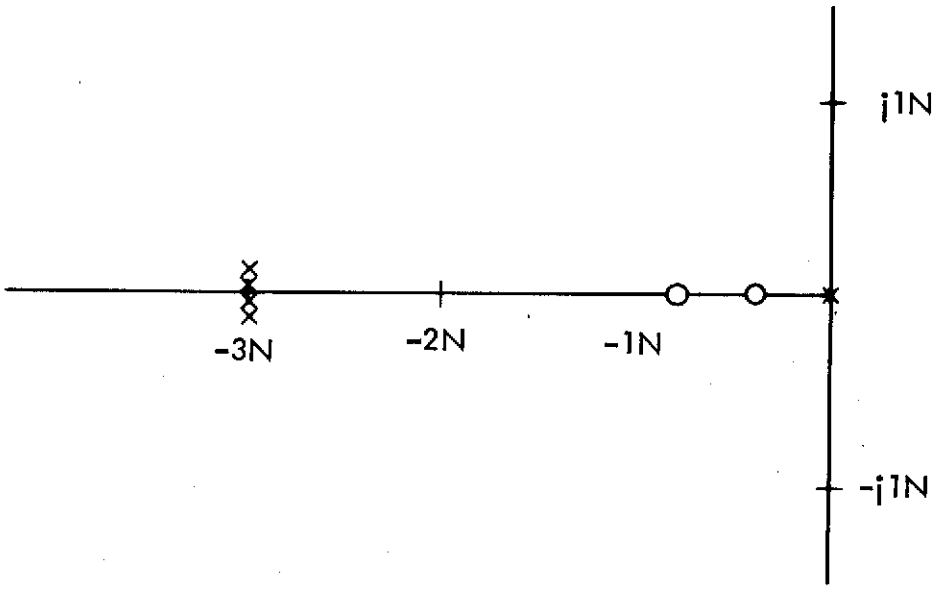
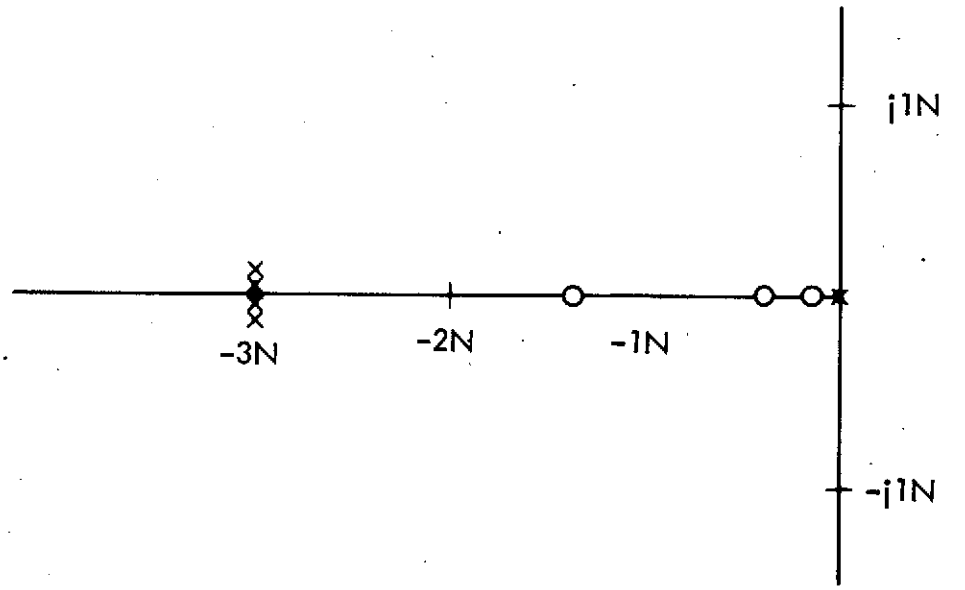


Figure 6a. Pole-Zero Locations for Cross Axis-Current Design



T97323

Figure 6b. Pole-Zero Locations for Direct Axis-Current Design

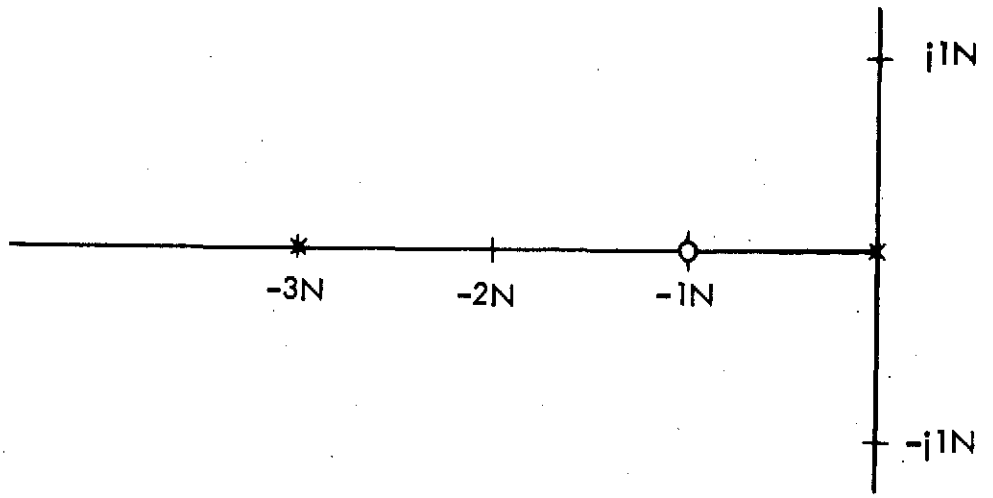


Figure 7a. Pole-Zero Locations for Cross Axis - New Design

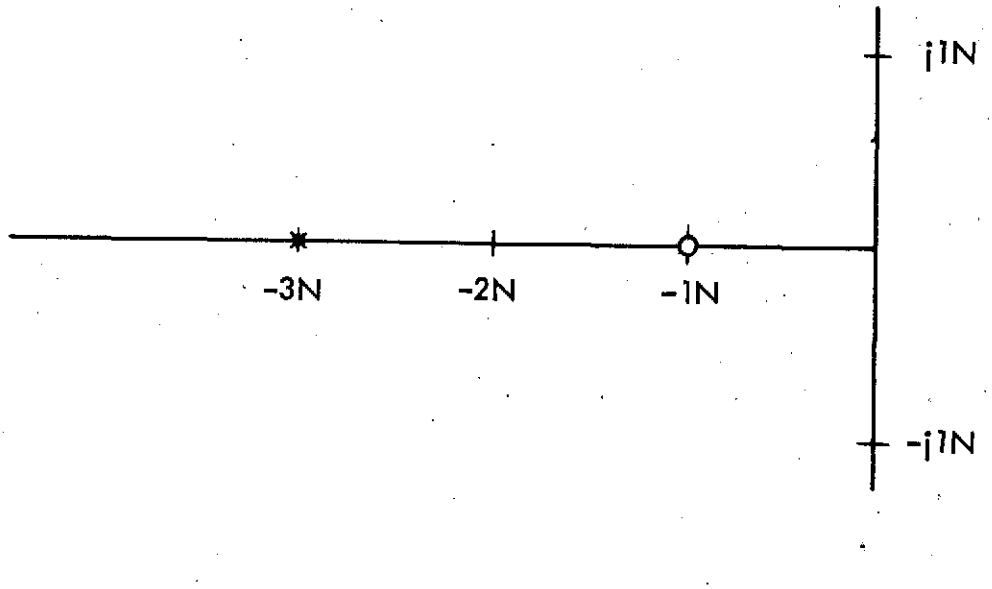


Figure 7b. Pole-Zero Locations for Direct Axis - New Design

Digital Compensation Analysis

Figure 8 shows a simplified block diagram of the digital gyro caging loop. This figure should be compared with Figure 4. One of the primary objectives of the program was to design digital filters which would emulate the performance of the existing analog filter design.

The study of digital filters and digital servos has progressed rapidly in the past decade and an extensive literature now exists with regard to these subjects. Rabiner and Rader*, for example, provides both theoretical and practical coverage of digital filtering as well as extensive bibliographies. The large variety of digital filters and the subtleties (e.g., the aliasing effect) which arise in their implementation, however, serve to maintain digital filtering as more of an art than a science.

Some "cookbook" approaches exist for obtaining digital equivalents to analog filters. Some of these have been analyzed, aided where required by simulation, in order to determine their applicability to the digital compensation loop design. In selecting a practical approach, particular attention was paid to the impact on the computer with regard to memory and time utilization. As an example, it was desirable to synthesize digital filters whose coefficients are integer powers of two, since this allows the replacement of multiply instructions with less time consuming shift instructions.

As one simple example of using the cookbook approaches, consider the bilinear transformation approach using the transformation

$$s \longrightarrow \frac{1-z^{-1}}{1+z^{-1}}$$

The transfer function

$$H(s) = \frac{\frac{s}{.79N}+1}{\left(\frac{s}{3N}+1\right)^2}$$

appearing in Figure 5 is seen to have a zero at $\omega_1 = .79N = 496$ rad/sec and a double pole at $\omega_2 = 3N = 1884$ rad/sec. In order to synthesize a digital filter with these characteristics we first derive an analog filter with a zero at

*Rabiner, L.R. and C.M. Rader, Digital Signal Processing, IEEE Press, New York, 1972.

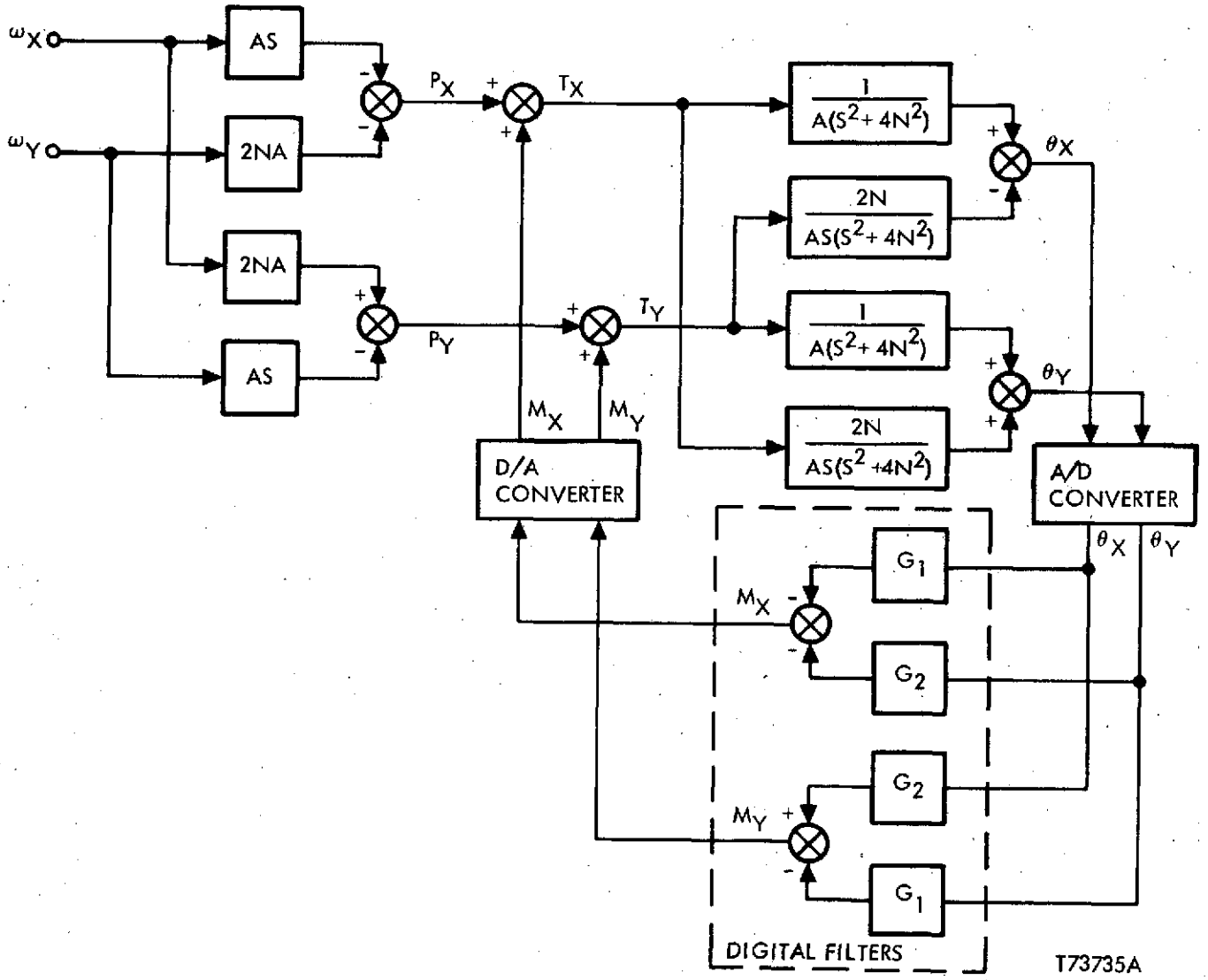


Figure 8. Block Diagram of Digital Compensation Loop

$$\omega'_1 = \tan \frac{\omega_1 \tau}{2}$$

and a double pole at

$$\omega'_2 = \tan \frac{\omega_2 \tau}{2}$$

where τ is the computer iteration interval. Assuming, for example, an iteration rate of 200 Hz, τ is 5 milliseconds and

$$\omega'_1 = \tan \frac{(496)(.005)}{2} = 2.91$$

$$\omega'_2 = \tan \frac{(1884)(.005)}{2} = 418.59$$

so that the "dummy" analog transfer function is

$$H'(s) = \frac{\frac{s}{2.91}+1}{\left(\frac{s}{418.59}+1\right)^2}$$

Finally, the digital transfer function is found by introducing the transformation

$$s \longrightarrow \frac{1-z^{-1}}{1+z^{-1}} \quad \text{so that}$$

$$H(z) = \frac{\frac{1}{2.91}\frac{(1-z^{-1})}{(1+z^{-1})}+1}{\left(\frac{1}{418.59}\frac{(1-z^{-1})}{(1+z^{-1})}+1\right)^2}$$

$$= \frac{\left[\frac{1}{2.91}(1-z^{-1}) + (1+z^{-1})\right]\left[1+z^{-1}\right]}{\left[\frac{1}{418.59}(1-z^{-1}) + (1+z^{-1})\right]^2}$$

$$= \frac{\left(1+\frac{1}{2.91}\right)+2z^{-1}+\left(1-\frac{1}{2.91}\right)z^{-2}}{\left(1+\frac{2}{418.59}+\frac{1}{(418.59)^2}\right)+2\left(1-\frac{1}{(418.59)^2}\right)z^{-1}}$$

$$\frac{}{+\left(1-\frac{2}{(418.59)}+\frac{1}{(418.59)^2}\right)z^{-2}}$$

The bracketed groups in this expression are fixed constants. Thus an exact mechnaization of this filter requires 6 multiplies and 4 adds per iteration of the computer. Observe, however, that one of these multiplications is by the number two. Thus for this operation a shift may replace a multiply instruction resulting in a substantial saving in computer time since a simple shift requires only a small fraction of the time needed for execution of a multiply instruction.

Further simplifications may be possible in this transfer function. For example, consider approximating the factor $(1 + 1/2.91) = 1.34$ by

$$1.375 = 1 + \frac{1}{4} + \frac{1}{8} \quad .$$

Then the multiplication (of the current input) by this factor may be accomplished by a double shift, a single shift, and two additions, again effecting a substantial saving in computer time. Such approximations must be carefully analyzed, however. Working "backward" through the preceding derivation it is seen that this approximation has the effect of shifting the zero of the transfer function from $\omega_1 = 496$ rad/sec to

$$\omega_1 = \frac{2}{\tau} \tan^{-1}(2.67)$$

$$= \frac{2}{.005}(1.21) = 485 \frac{\text{rad}}{\text{sec}}$$

In this case the effect on the filter response is probably negligible. This is not the case for all approximations which appear on the surface to be reasonable, however.

Initial efforts in designing the digital compensation filters concentrated on emulating existing "old design" analog filters as used in previous strapdown systems. The transfer functions of these analog filters are shown in the block diagram of Figure 4 and their pole-zero locations in Figure 5.

Consider first the cross axis transfer function relating M_Y to θ_X, i.e. the y-axis torquer output to the x-axis pickoff angle. The analog transfer function is

$$\frac{M_Y(s)}{\theta_X(s)} = \frac{1.01\, K_{sa}\, K_{po}\, K_T \left(\frac{s}{.79N}+1\right)\left(\frac{s}{.35N}+1\right)\left(\frac{s^2}{N^2}+1\right)}{s\left(\frac{s}{16N}+1\right)\left(\frac{s}{3N}+1\right)^4}$$

Since the pickoff and torquer transfer functions will exist in the digital loop in any case, they do not have to be implemented digitaly. The gain factor ($1.01\, K_{sa}$) may also be neglected in the digital design as the gain may be provided by the power amplifier. Thus the digital transfer function to be mechanized should be equivalent to

$$H(s) \doteq \frac{\left(\frac{s}{.35N}+1\right)\left(\frac{s}{.79N}+1\right)\left(\frac{s^2}{N^2}+1\right)}{s\left(\frac{s}{3N}+1\right)^4}$$

Using the bilinear transformation technique and assuming now an 800 Hz update rate, the dummy analog breakpoints are computed

$$\omega'_1 = \tan\frac{(2\pi)(35)}{(2)(800)} = .13832$$

$$\omega'_2 = \tan\frac{(2\pi)(79)}{(2)(800)} = .32058$$

$$\omega'_3 = \tan\frac{(2\pi)(100)}{(2)(800)} = .41421$$

$$\omega'_4 = \tan\frac{(2\pi)(300)}{(2)(800)} = 2.41421$$

Making the substitution

$$s \longrightarrow \frac{1 - z^{-1}}{1 + z^{-1}}$$

the digital transfer function is found (after much arithmetic manipulation) to be

$$H(z) = 57.8787 \frac{1 - 1.68566z^{-1} + .50190z^{-2} + 1.36534z^{-3} - 1.43278z^{-4} + 3.8945z^{-5}}{1 + .65685z^{-1} - .62742z^{-2} - .74517z^{-3} - .25483z^{-1} - .02944z^{-5}}$$

In order to obtain a rough estimate of computer timing requirements for implementation of the digital loop, the numerator of H(z) was considered. The numerator was first approximated in order to eliminate the need for multiplications. Thus

$$1.68566 \simeq 1 + 2^{-1} + 2^{-3} + 2^{-4} = 1.6875$$

$$.50190 \simeq 2^{-1} = .500$$

$$1.36534 \simeq 1 + 2^{-2} + 2^{-3} = 1.375$$

$$1.43278 \simeq 1 + 2^{-2} + 2^{-3} + 2^{-4} = 1.4375$$

$$.38945 \simeq 2^{-2} + 2^{-3} + 2^{-6} = .390625$$

so that the numerator is implemented as

$$\theta_x(k) - (1+2^{-1}+2^{-3}+2^{-4})\,\theta_x(k-1) + (2^{-1})\theta_x(k-2) + (1+2^{-2}+2^{-3})\,\theta_x(k-3)$$

$$- (1+2^{-2}+2^{-3}+2^{-4})\,\theta_x(k-4) + (2^{-2}+2^{-3}+2^{-6})\,\theta_x(k-5)$$

(Working "backward" as before, these approximations resulted in analog breakpoints of .519N, .749N and .711N $\pm$j.706N, a reasonable agreement with the original transfer function.) Using the TDY-52B/IMP-16 instructions, a simple program was written to implement the equation above. The instruction breakdown was as follows

Instruction	Number	Time Per Instruction	Time (μ sec)
Load	6	7.7	46.20
Store	5	9.1	45.50
Reg. Copy	1	8.75	8.75
Reg. Add	10	4.55	45.50
Add/Sub	5	7.7	38.50
Shift Rt. 1	3	10.1	30.30
Shift Rt. 2	5	14.3	71.50
Shift Rt. 3	1	17.5	17.50
			303.75 μsec

Assuming that each transfer function numerator and denominator is of roughly the same complexity, the total time required to impement the four transfer functions is

$$(303.75) \times 2 \times 4 = 2.43 \text{ msec}$$

which is approximately twice the available time for an 800 Hz update rate. Since this estimate does not include I/O processing, scaling, executive, control, etc. it was clear that much simpler processing would be required if the digital compensation loop were to be successfully implemented with the TDY-52B computer.

Fortunately, as described in Section II, efforts were underway concurrent to the digital loop program to simplify and improve analog compensation techniques. This work resulted in the designs described in the preceeding section, and showed that very simple analog transfer functions could be used to obtain the desired servo loop response characteristics. Furthermore, it was decided to mechanize one axis of the gyro in an analog loop in order to more accurately compare the analog loop and digital loop responses, thus reducing the digital computations which were required. These two factors resulted in a digital loop design which could be implemented with a TDY-52B even though the update rate was increased to 1200 Hz.

The "first cut" at implementing the new transfer functions started with the analog functions whose pole-zero locations are shown in Figure 7.

$$F_1(s) = \frac{s + N}{s + 3N}$$

$$F_2(s) = \frac{s + N}{s(s + 3N)}$$

where $F_1(s)$ is the direct axis transfer function and $F_2(s)$ is the cross axis transfer function. Using a 1200 Hz update, the dummy breakpoint frequencies were computed to be

$$\omega'_1 = \tan\frac{(2\pi)\,(100)}{(2)\,(1200)} = .26795$$

$$\omega'_2 = \tan\frac{(2\pi)\,(300)}{(2)\,(1200)} = 1.0000$$

so that

$$F_1(z) = \frac{\frac{(z-1)}{(z+1)} + .26795}{\frac{(z-1)}{(z-1)} + 1.0000} = \frac{1.26795z - .73205}{2z} = .63398(1-.57735z^{-1})$$

$$F_2(z) = \frac{\frac{(z-1)}{(z+1)} + .26795}{\frac{(z-1)}{(z+1)}\left[\frac{z-1}{z+1} + 1.000\right]} = .63398\,\frac{(1 + .42265z^{-1} - .57735z^{-2})}{(1-z^{-1})}$$

Since the constant gain term can be readily implemented in the analog portion of the loop, it may be neglected. The transfer functions were further simplified to

$$G_1(z) = (1 - .5z^{-1})$$

$$G_2(z) = \frac{(1 + .5z^{-1} - .5z^{-2})}{(1-z^{-1})}$$

for easier digital implementation. This has the effect of changing ω'_1 to .333 and ω_1 (the analog loop zero) to 1.23N instead of N.

The comparison of analog and digital frequency responses is shown in Figures 9a and 9b for this configuration. It is seen that the breakpoints and low - frequency asymptotics are identical, with variations occurring at intermediate points. The high frequency performance of the digital loop is, of course, different from the analog response due to the "fold-over" effect. For this reason, the filter update rate must be sufficiently high that the deleterious effects of this phenomenon are countered by attenuation in the analog portions of the loop. (Test results indicate that 1200 Hz is the minimum update frequency required to obtain the desired response and that higher frequencies are desirable.)

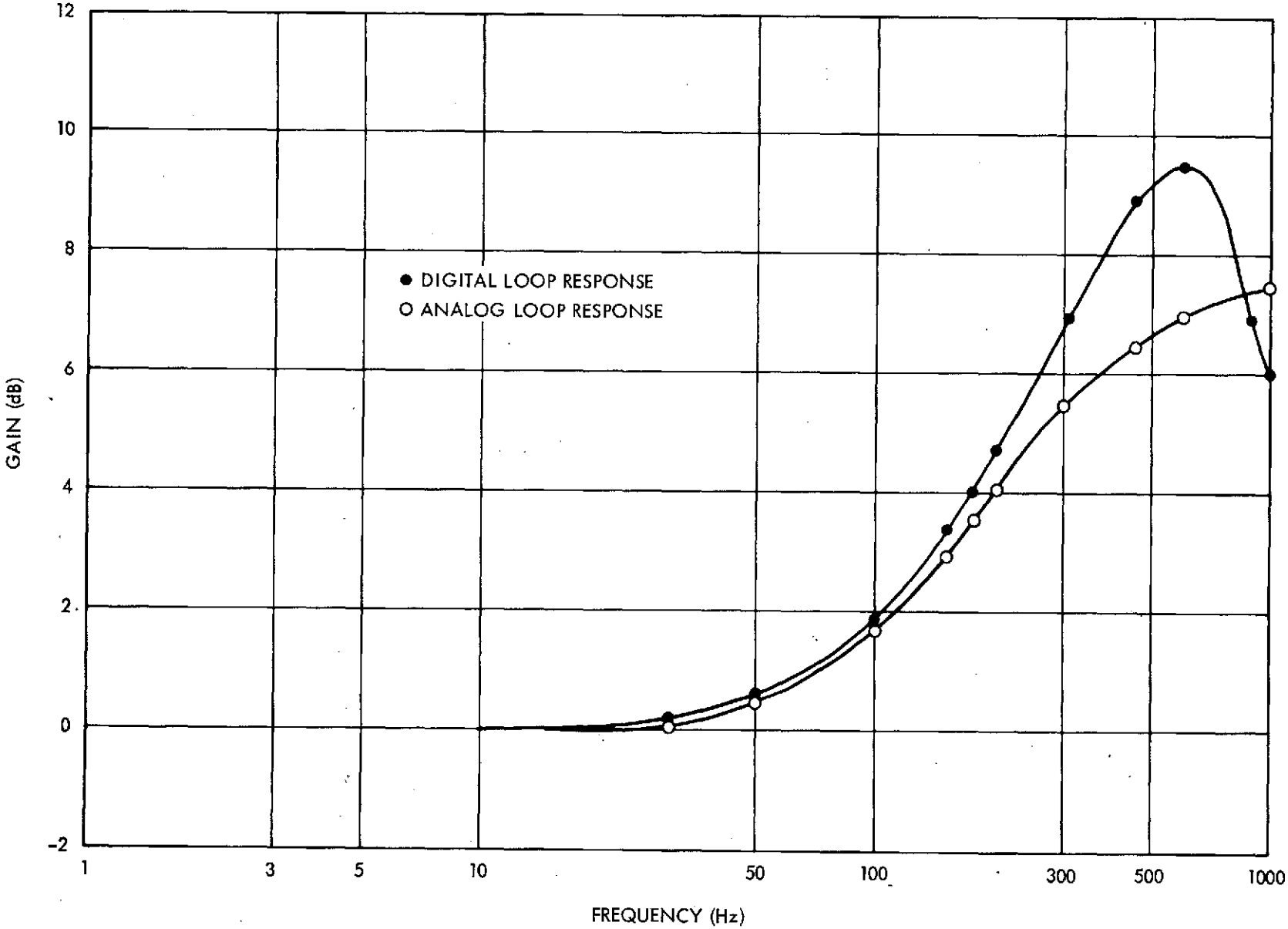


Figure 9a. Direct Axis Frequency Response

T97321

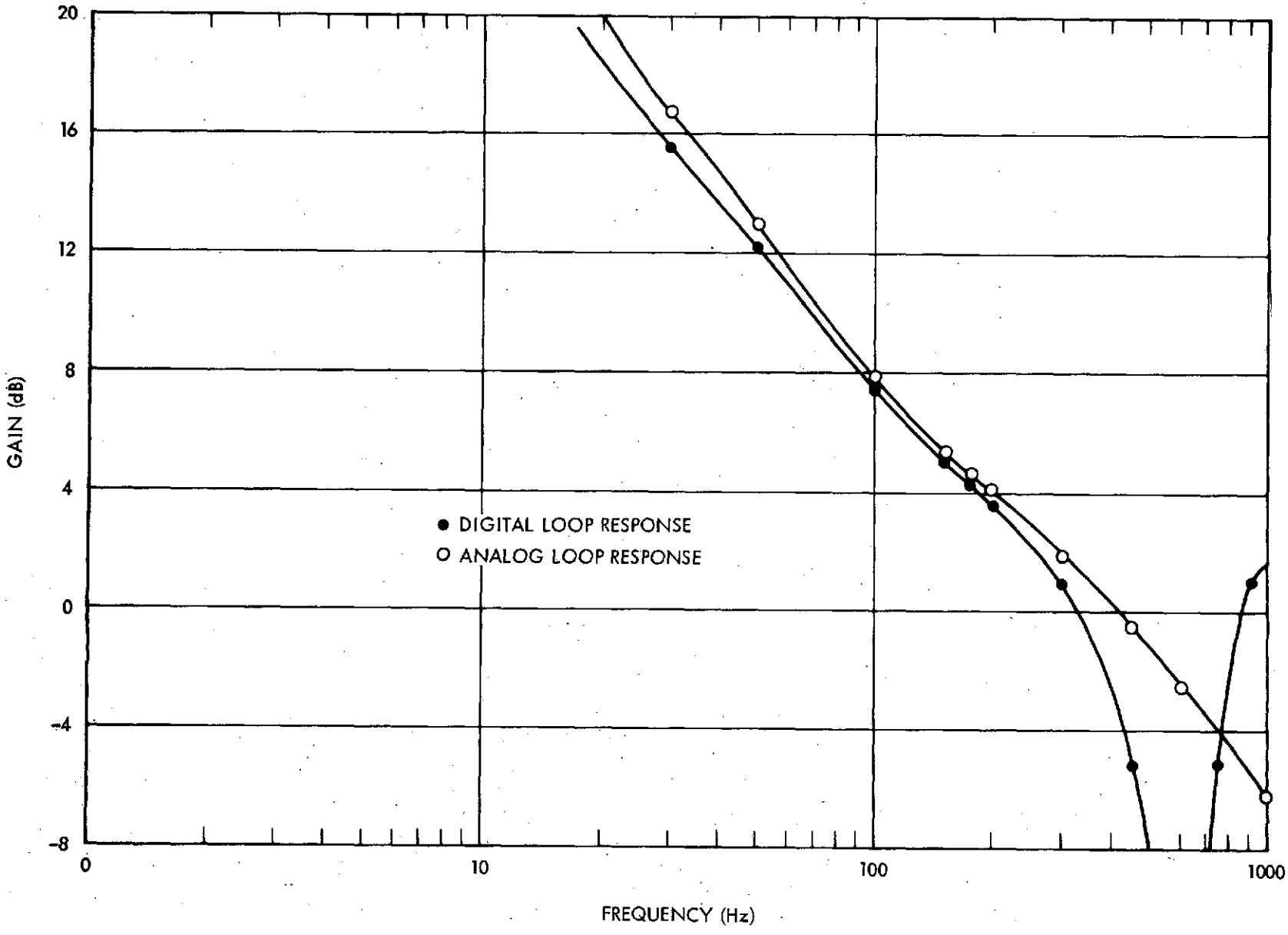


Figure 9b. Cross Axis Frequency Response

Several variations of parameters were tested within the constraints of the basic filter configurations described by $G_1(z)$ and $G_2(z)$ above. It was found that the transfer functions

$$G_1(z) = \left(1 - \frac{15}{16} z^{-1}\right)$$

$$G_2(z) = \frac{1 + \frac{1}{16} z^{-1} - \frac{15}{16} z^{-2}}{1 - z^{-1}}$$

provided reasonably optimal performance for the digital loop*. Most of the results described in the report reflect this configuration. Thus the equations which were mechanized in the computer were

$$M_x(k) = M_x(k-1) + \theta_x(k) + (2^{-4})\,\theta_x(k-1) - (1-2^{-4})\,\theta_x(k-2)$$

$$M_y(k) = \theta_y(k) - (1-2^{-4})\,\theta_y(k-1)$$

$$M(k) = M_x(k) - M_y(k)$$

where $M(k)$ is the computed gyro torque at the k^{th} iteration.

The digital number $M(k)$ is converted by the digital-to-analog converter and, after power amplification, used to torque the gyro. In the absense of errors, $M(k)$ represents the true torque input and is, therefore, proportional to the angular rate of the gyro case. This is precisely the information which is desired for attitude computation in a strapdown navigator. Any errors introduced by the D/A converter therefore propagate directly into attitude errors. For this reason it is necessary to use an accurate D/A converter in the digital loop.

*This corresponds to moving the zero locations in Figure 6 from 100 Hz to approximately 12.7 Hz.

The analog-to-digital conversion requirements are much less stringent. Due to the closed-loop nature of the digital compensator, errors in the A/D converter are of relatively minor importance to the accuracy of rate measurement (just as pickoff measurement errors are unimportant in a conventional analog loop mechanization).

The test results indicate that a 16 bit computer word length is sufficient for mechanization of the digital loop. The D/A converter should be an accurate device with excellent (or, at least, well compensatable) bias and scale factor characteristics. The A/D converter can be less accurate without seriously compromising torque measurement accuracy. The digital loop was mechanized using a 12 bit A/D converter and a 13 bit D/A converter. These resolutions are not critical, however, for the reasons outlined above. As long as the analog torque applied accurately represents the (quantized) digital torque, no net error will result in the computation of attitude due to the D/A converter resolution.

III. COMPUTER

The computer used in the breadboard is an IMP-16P (National Semiconductor), which is electrically and functionally interchangeable with the TSC hybrid packaged TDY-52B. Although these two machines are functionally the same, they are vastly different physically. The CPU of each consists of four 4-bit NS (National Semiconductor) GPC/P RALU slices. The microprogram of each is contained in a set of two Control Read Only Memories (CROMs) that implement the NS IMP-16 computer repertoire. Input/Output and other hardware details are also similar. The IMP-16P and TDY-52B are thus, basically, the same machine.

The most obvious difference between the IMP-16P and TDY-52B is one of size. Teledyne has packaged the heart of the IMP-16P into a small hybrid unit that is only two inches on a side and .2" thick. The IMP-16P computer, including a convenient control panel and 4K of 16-bit RAM memory, occupies a box with a frontal dimension of about 10 1/2" x 17" and a depth of about 24".

The IMP-16P was selected for the digital loop demonstration primarily because its control panel feature facilitated the operation and modification of the breadboard. The results which were obtained are, however, equally applicable to the TDY-52B.

Table 1. TDY-52B/IMP-16 Instruction Repertoire (Sheet 1 of 2)

MNEMONIC	INSTRUCTION NAME	FUNCTION	FORMAT	EXECUTION TIME IN MICROSECONDS
LOAD AND STORE				
LD	LOAD	(EA) → (ACr), IF INDIRECT ((EA)) → (ACr)	2	7.0, 9.8 IF INDIRECT
LDB	LOAD BYTE	(1/2 EA) → (AC0 LESS SIGNIFICANT BYTE)	5	16.8 TO 28.0
LI	LOAD IMMEDIATE	D → (ACr)	4B	4.2
ST	STORE	(ACN) → (EA), IF INDIRECT (ACr) → ((EA))	2	8.4, 11.2 IF INDIRECT
STB	STORE BYTE	(AC0 LESS SIGNIFICANT BYTE) → (1/2 EA) 0 → (SEL)	5	23.8 TO 32.2
RXCH	REGISTER EXCHANGE	(SR) → (DR), (DR) → (SR)	6	11.2
RCPY	REGISTER COPY	(SR) → (DR)	6	8.4
LOGICAL				
RXOR	REGISTER EXCLUSIVE OR	(SR) ⊕ (DR) → (DR)	6	8.4
RAND	REGISTER AND	(SR) "AND" (DR) → (DR)	6	8.4
AND	AND	(R01) "AND" (EA) → (R01)	3	7.0
OR	OR	(R01) "OR" (EA) → (R01)	3	7.0
ARITHMETIC				
RADD	REGISTER ADD	(SR) + (DR) → (DR) OV, CY	6	4.2
ADD	ADD	(ACr) + (EA) → (ACr) OV, CY	2	7.0
SUB	SUBTRACT	(ACr) − (EA) → (ACr) OV, CY	2	7.0
MPY	MULTIPLY	(EA) * (AC1) → {(AC0, AC1)} L 0 → (SEL)	5	148.4 TO 170.8
DIV	DIVIDE	{(AC0), (AC1)} → (EA) → (AC0) QUOTIENT 0 → (SEL) OV, L (AC1) REMAINDER	5	177.8 TO 222.6
DADD	DOUBLE PRECISION ADD	{(AC0), (AC1)} + {(EA), (EA+1)} → {(AC0), (AC1)} 0 → (SEL) OV, CY	5	16.8
DSUB	DOUBLE PRECISION SUBTRACT	{(AC0), (AC1)} − {(EA), (EA + 1)} → {(AC0), (AC1)} 0 → (SEL) OV, CY	5	16.8
CAI	COMPLEMENT AND ADD IMMEDIATE	~ (ACr) + D → (ACr)	4B	4.2
SHIFT				
ROL	ROTATE LEFT	2 (ACr) → (ACr) IF SEL = 0, (BIT 15) → (BIT 0) IF SEL = 1, (BIT 15) → (L), (L) → (BIT 0) } D TIMES	4B	5.6 + 4.2D
ROR	ROTATE RIGHT	1/2 (ACr) → (ACr) IF SEL = 0, (BIT 0) → (BIT 15) IF SEL = 1, (BIT 0) → (L), (L) → (BIT 15) } D TIMES	4B	5.6 + 4.2D
SHL	SHIFT LEFT	2 (ACr) → (ACr) 0 → (BIT 0) IF SEL = 1, (BIT 15) → (L) } D TIMES	4B	5.6 + 4.2D
SHR	SHIFT RIGHT	1/2 (ACr) → (ACr) IF SEL = 0, 0 → (BIT 15) IF SEL = 1, (L) → (BIT 15), 0 → (L) } D TIMES	4B	5.6 + 4.2D
SINGLE BIT				
SETST	SET STATUS BIT	1 → (STATUS FLAG N)	9	18.2 TO 44.8
CLRST	CLEAR STATUS BIT	0 → (STATUS FLAG N)	9	18.2 TO 44.8
SETBIT	SET BIT	1 → (AC0 BIT N)	9	18.2 TO 44.8
CLRBIT	CLEAR BIT	0 → (AC0 BIT N)	9	18.2 TO 44.8
CMPBIT	COMPLEMENT BIT	$\overline{(AC0\ BIT\ N)}$ → (AC0 BIT N)	9	18.2 TO 44.8

T92170-1A

Table 1. TDY-52B/IMP-16 Instruction Repertoire (Sheet 2 of 2)

MNEMONIC	INSTRUCTION NAME	FUNCTION	FORMAT	EXECUTION TIME IN MICROSECONDS
JUMP				
JMP	JUMP	EA→(PC), IF INDIRECT (EA)→(ACr)	4A	4.2,7.0 IF INDIRECT
JMPP	JUMP THROUGH POINTER	$(100_{16}$ +N)→(PC)	9	9.8
JINT	JUMP INDIRECT TO LEVEL 0	(PC)→(STK), 0→(IEF)	9	9.8
		$(120_{16}$ +N)→PC		
BOC	BRANCH ON CONDITION	IF CONDITION CC IS TRUE,	1	5.6,7.0 IF BRANCH
		(PC) + D →(PC)		
JSR	JUMP TO SUBROUTINE	(PC)→(STK)	4A	5.6,8.4 IF INDIRECT
		EA→(PC), IF INDIRECT (EA)→(PC)		
JSRI	JUMP TO SUBROUTINE IMPLIED	(PC)→(STK)	8	5.6
		$FF80_{16}$ + C→(PC)		
JSRP	JUMP TO SUBROUTINE THROUGH	(PC)→(STK)	8	11.2
	POINTER	$(100_{16}$ + C)→(PC)		
RTS	RETURN FROM SUBROUTINE	(STK) + C→(PC)	8	5.6
RTI	RETURN FROM INTERRUPT	(STK) + C→(PC)	8	7.0
		1→(IEF)		
SKIP				
AISZ	ADD IMMEDIATE AND SKIP IF ZERO	(ACr) + D→(ACr) OV, CY IF (ACr) = 0, (PC) + 1→(PC)	4B	5.6,7.0 IF SKIP
SKAZ	SKIP IF "AND" IS ZERO	IF (R01) "AND" (EA) = 0, (PC) + 1→(PC)	3	8.4,9.8 IF SKIP
ISZ	INCREMENT AND SKIP IF ZERO	(EA) + 1→(EA) IF (EA) = 0, (PC) + 1→(PC)	4B	9.8, 11.2 IF SKIP
DSZ	DECREMENT AND SKIP IF ZERO	(EA) - 1→(EA) IF (EA) = 0, (PC) + 1→(PC)	4B	11.2,12.6 IF SKIP
SKG	SKIP IF GREATER THAN	IF (ACr) >(EA), (PC) + 1→(PC)	2	11.2 TO 14.0
SKNE	SKIP IF NOT EQUAL	IF (ACr) ≠ (EA), (PC) + 1→(PC)	2	8.4
SKSTF	SKIP IF STATUS FLAG TRUE	IF (STATUS FLAG N) = 1, (PC) + 1→(PC) 0 (SEL)	9	18.2 TO 44.8
SKBIT	SKIP IF BIT TRUE	IF (AC0 BIT N) = 1, (PC) + 1→(PC) 0 (SEL)	9	18.2 TO 44.8
STACK				
PUSHF	PUSH STATUS FLAGS ONTO STACK	(SF)→(STK)	8	5.6
PULLF	PULL STATUS FLAGS FROM STACK INTO FLAG REGISTER	(STK)→(ACr)	8	7.0
PUSH	PUSH ONTO STACK	(ACr)→(STK)	4B	4.2
PULL	PULL FROM STACK	(STK)→(ACr)	4B	4.2
XCHRS	EXCHANGE REGISTER AND STACK	(ACr) →(STK) (STK)→(ACr)	4B	7.0
INPUT/OUTPUT				
RIN	REGISTER INPUT	(AC3) + C→(IO ADDR) (IO DATA)→(AC0)	8	9.8
ROUT	REGISTER OUTPUT	(AC3) + C →(IOADDR) (AC0)→(IO DATA)	8	9.8
SFLG	SET FLAG	C→(IOADDR), 1→(CONTROL FLAG FC)	7	5.6
PFLG	PULSE FLAG	C→(IOADDR), 1→(CONTROL FLAG FC)	7	5.6
ISCAN	INTERRUPT SCAN	1/2 (AC1)→(AC1) UNTIL 1 SHIFTED OUT (AC2) + NUMBER OF SHIFTS→(AC2)	9	8.4 TO 100.8
HALT	HALT	PROCESSOR HALTS	8	—

T92170-2A

Notation Used in Instruction Descriptions

Notation	Meaning
ACr	Denotes a specific working register (AC0, AC1, AC2, or AC3), where *r* is the number of the accumulator referenced in the instruction.
AR	Denotes the address register used for addressing memory or peripheral devices.
cc	Denotes the 4-bit condition code value for conditional branch instructions.
ctl	Denotes the 7-bit control-field value for flag, input/output, and miscellaneous instructions.
CY	Indicates that the Carry flag is set if there is a carry due to the instruction (either an addition or a subtraction).
disp	Stands for displacement value and it represents an operand in a nonmemory reference instruction or an address field in a memory reference instruction. It is an 8-bit, signed twos-complement number except when base page is referenced; in the latter case, it is unsigned.
dr	Denotes the number of a destination working register that is specified in the instruction-word field. The working register is limited to one of four: AC0, AC1, AC2, or AC3.
EA	Denotes the effective address specified by the instruction directly, indirectly, or by indexing. The contents of the effective address are used during execution of an instruction. See table 3–1.
fc	Denotes the number of the referenced flag (see table 3–20 under 3.6.10, Input/Output, Halt, and Flag Instructions).
INTEN	Denotes the Interrupt Enable control flag.
IOREG	Denotes an input/output register in a peripheral device.
L	Denotes 1-bit link (L) flag.
OV	Indicates that the overflow flag is set if there is an overflow due to the instruction (either an addition or a subtraction).
PC	Denotes the program counter. During address formation, it is incremented by 1 to contain an address 1 greater than that of the instruction being executed.
r	Denotes the number of a working register that is specified in the instruction-word field. The working register is limited to one of four: AC0, AC1, AC2, or AC3.
SEL	Denotes the Select control flag. It is used to select the carry or overflow for output on the carry and overflow (CYOV) line of the CPU, and to include the link bit (L) in shift operations.
sr	Denotes the number of a source working register that is specified in the instruction-word field. The working register is limited to one of four: AC0, AC1, AC2, or AC3.
xr	When not zero, this value designates the number of the register to be used in the indexed and relative memory-addressing modes.

Notation Used in Instruction Descriptions (Continued)

Notation	Meaning
( )	Denotes the contents of the item within the parentheses. (ACr) is read as "the contents of ACr." (EA) is read as "the contents of EA."
[ ]	Denotes "the result of."
~	Indicates the logical complement (ones complement) of the value on the right-hand side of ~.
→	Means "replaces."
←	Means "is replaced by."
@	Appearing in the operand field of an instruction, denotes indirect addressing.
∧	Denotes an AND operation.
∨	Denotes an OR operation.
∇	Denotes an exclusive OR operation.

IV. SOFTWARE

Introduction

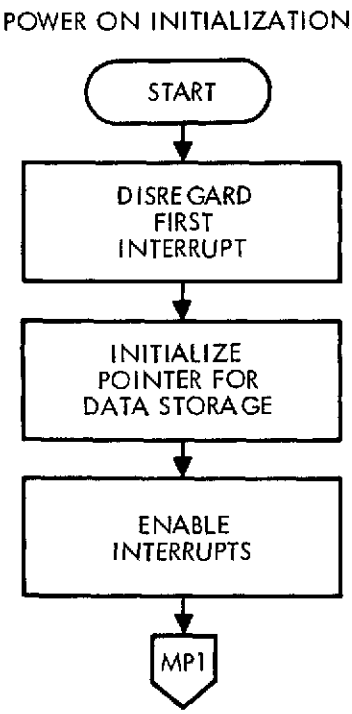
The intent of this study was not to demonstrate the particular processor which was used but rather to demonstrate how the program was mechanized for this particular computer in such a manner that the software could be mechanized for any given computer with equivalent characteristics. The reader should have some knowledge, however, of the computer in which the equation set was mechanized in order to fully understand the work performed. An example of how the flow charts reflect the instruction repertoire of the processor is in the handling of negative quantities. The TDY-52B/IMP-16 has a logical shift only, so that when a multiplication by 1/16 is to be performed it is necessary to handle negative quantities in a different manner than positive quantities where the multiplication can be performed by a shift of 4 bit positions.

In order to give a better understanding of the digital computer used in this study a summary of instructions was included in Section III. Should the reader have further interest in this processor including the interrupt system, push-pull stack operation, logic mechanization,etc., complete descriptive manuals can be obtained from Teledyne Systems Company.

Power On Processing

Upon recognition of Power being applied to the unit, coding is performed to disregard the initial multiplexer interrupt. This is done to allow the program to become synchronized with the timing base of the multiplexer so that after the initial interrupt the timing base of approximately 104 microseconds between interrupts is established for the remaining processing. The next task is to initialize a pointer within the processor so that whenever data is generated for storage the starting point for this storage is established.

The interrupts are then enabled so that the processor is ready to recognize interrupts from the multiplexer. Finally a transfer is made to the Main Processing Loop. As the processor used in this study turns on with Random Access Memory (RAM) set to an all one's condition, the initial iterations of the Main Processing do not have the proper data for use as a priori iteration data so that the data generated for the first two iterations of the Main Processing Loop is not precise.



Main Programming Loop

The Main Programming Loop is where the majority of calculations are performed. This routine is coded to run continuously, i. e. from Power On the calculation process is repeatedly performed with the only interruptions being the X peak and Y peak Interrupts where data from the analog to digital converter is input. Once the data has been input within either of these interrupt routines, program control is transfered back to the Main Programing Loop at the point the program was interrupted.

Essentially the processing within this software module consists of solving the equations below:

$$M_x(K) = M_x(K-1) + U_x(K) + 1/16\ U_x(K-1) - 15/16\ U_x(K-2)$$

$$M_y(K) = U_y(K) - 15/16\ U_y(K-1)$$

$$M_{\ }(K) = U_x(K) - U_y(K)$$

where:

M_x = filtered digital quantity from X axis pickoff

M_y = filtered digital quantity from Y axis pickoff

M = total digital torque output to D/A

Index (k) designates quantity input or calculated on current program iterations.

Index (k-1) designates quantity input or calculated on previous iteration

Index (k-2) designates quantity input or calculated two iterations previous

The other calculations performed in this loop and shown on the flow diagrams for the Main Processing Loop is the interrogation of two software flags, X peak flag and Y peak flag, to properly correlate the data from the storage buffers and to transform the data from its 2's complement form for the sign magnitude digital to analog convertor.

Following is the flow chart for the Main Processing Loop. (Figure 10.)

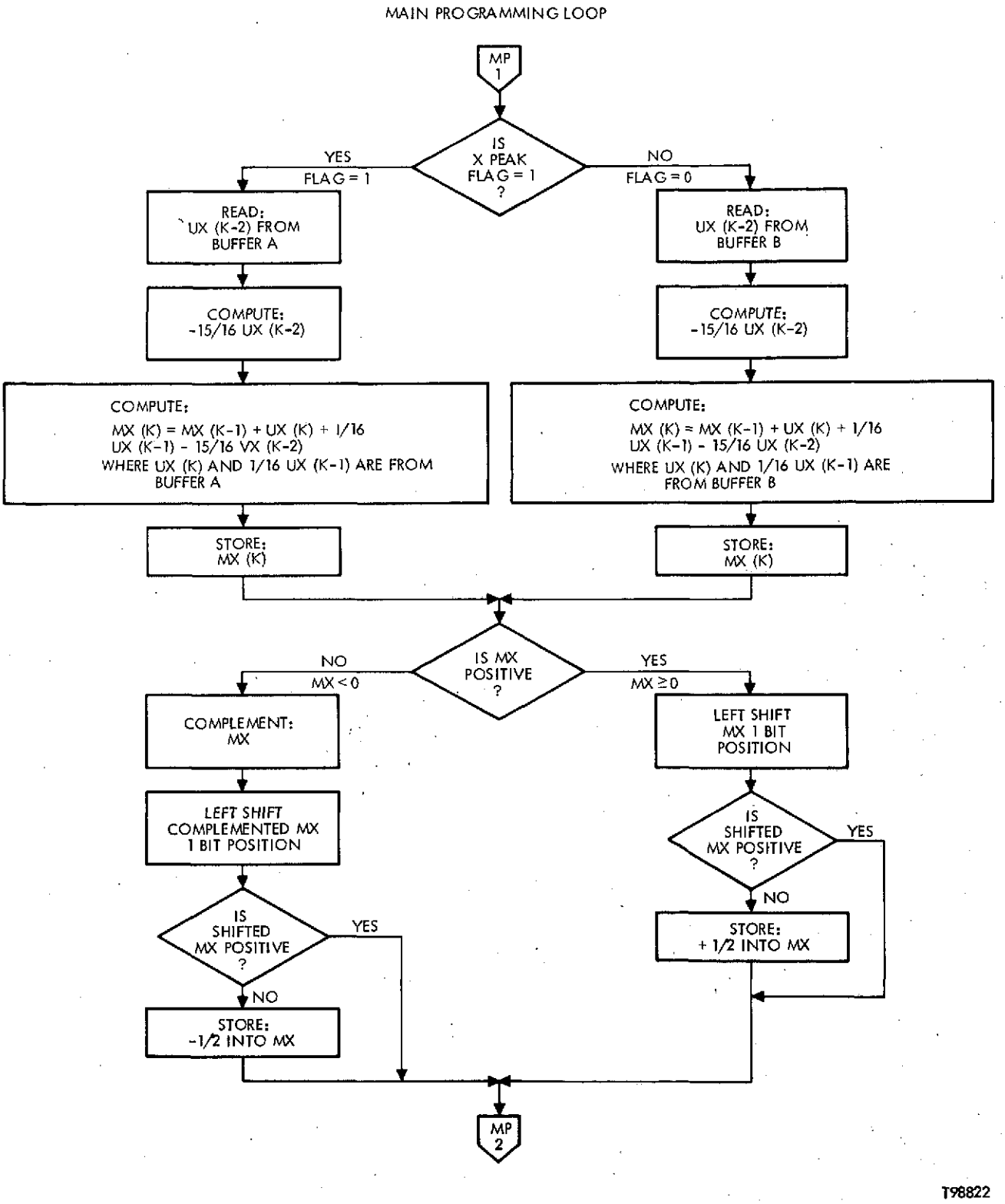


Figure 10. Main Programming Loop

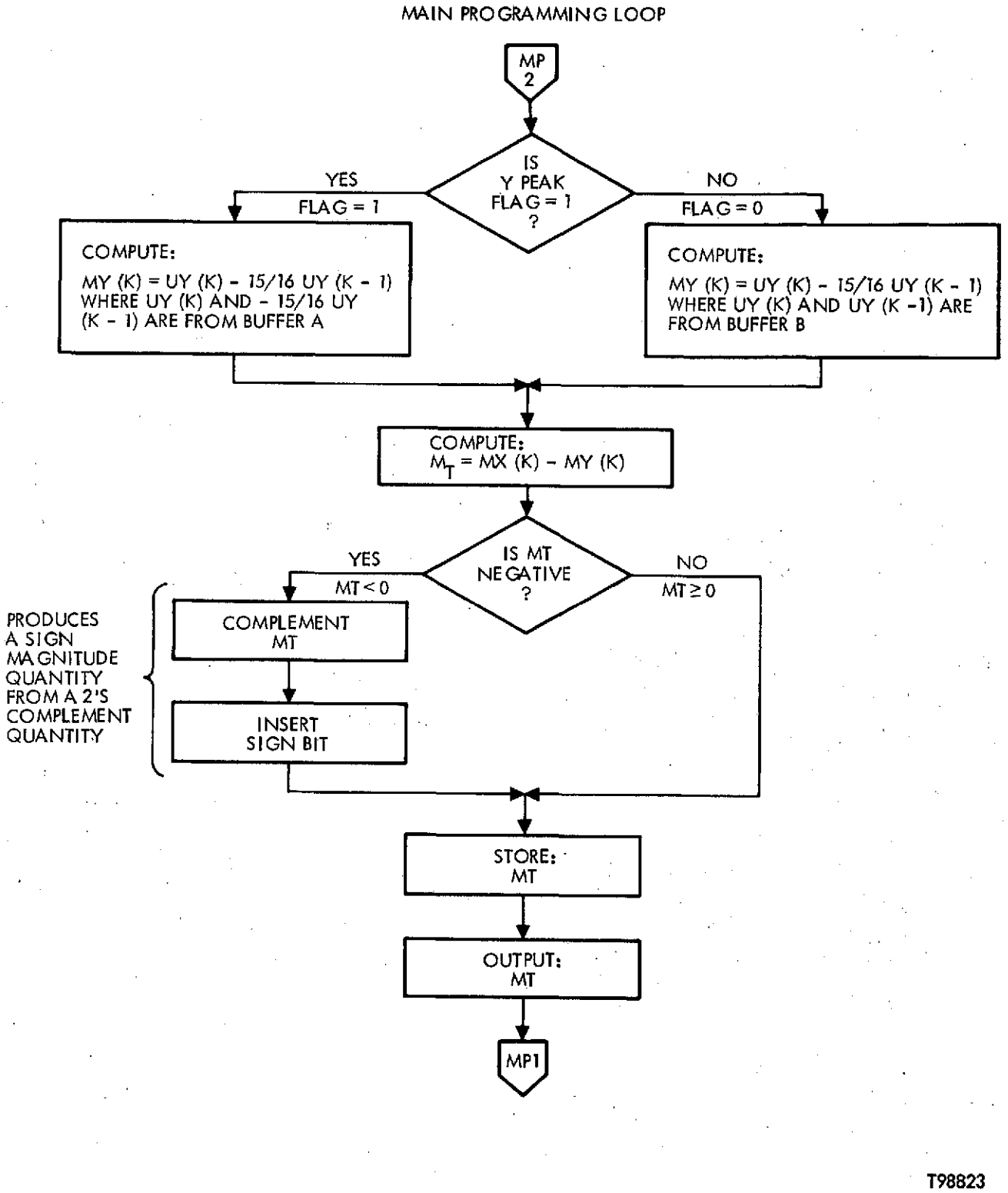


Figure 10. Main Programming Loop (Continued)

X Peak Interrupt Servicing

Upon recognition by the processor that sample data from the X axis pickoff is available, the Main Processing Loop is interrupted at the end of the instruction it is performing and the program is transferred to the X Peak Interrupt Servicing routine. The purpose of this routine is to accept the digitized X axis pickoff data for the computation to be performed by the Main Processing Loop. In order to keep the data being computed in the Main Programming Loop correlated it was necessary to mechanize two buffer areas which are merely memory cells set aside for temporary data storage and retrieval. At any given point of time one of the buffer areas is set for reading by the Main Processing Loop and the other buffer area is set for writing or storage by the X Peak Interrupt Servicing routine. The purpose of these buffers is to ensure that the data being computed is indeed from successive iterations. A memory cell is used as a flag (X Peak Flag in flow charts) to indicate how the buffers are currently being utilized. This same flag is again interrogated in the Main Processing Loop for the same purpose. A graphic illustration of the storage areas referred to as 'swinging buffers' will perhaps clarify this coding concept. At the time the X Peak Interrupt is entered by the program assume that the two buffers are as shown below

Buffer A	$U_X(K)$	$U_X(K-1)$	$U_X(K-2)$
Buffer B	$U_X(K)$	$U_X(K-1)$	$U_X(K-2)$

and further assume that the flag (X Peak Flag) indicates that Buffer A is the area that should be used to store data. The cell from Buffer B containing $U_X(K-1)$ must first be moved to Buffer A as it now becomes two iterations old, i.e. $U_X(K-2)$. In the same manner the Buffer B cell containing $U_X(K)$ is moved to Buffer A and is now one iteration old, i.e. $U_X(K-1)$. The digitized X axis pickoff data input during the current iteration is then stored in Buffer A as $U_X(K)$. Buffer A then contains data from three successive inputs. The process is reversed the next iteration such that Buffer B contains current data. The last task of the X Peak Interrupt processing is to divide the data from the previous iteration by 16 (accomplished by a right shift of 4 places) before storing to relieve the Main Processing Loop of this computational burden. Once these tasks are performed the program is returned to the Main Processing Loop to the point at which the interrupt occurred.

Following is the flow charts for the X Peak Interrupt Processing. (Figure 11.)

X PEAK INTERRUPT SERVICING FLOWS

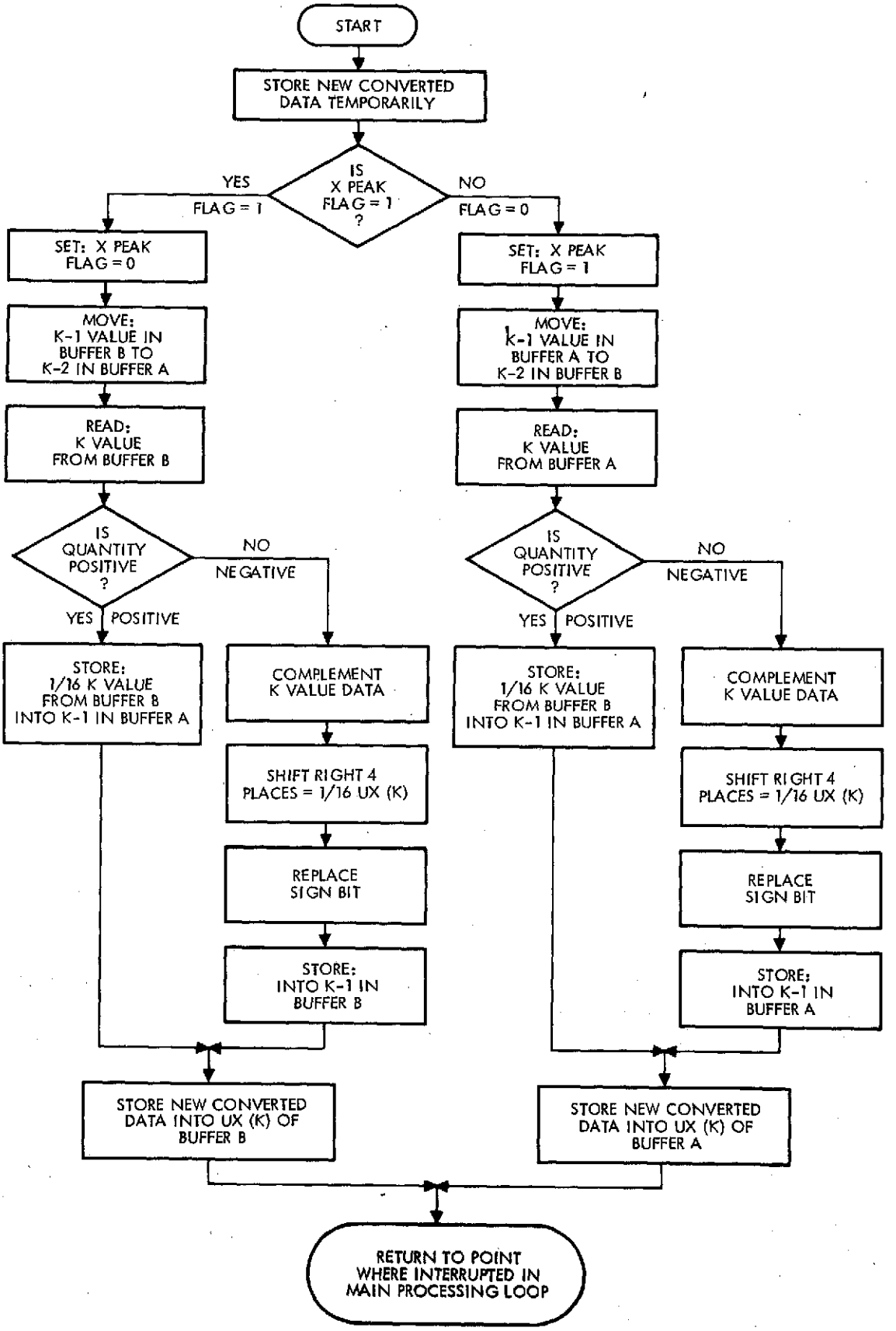
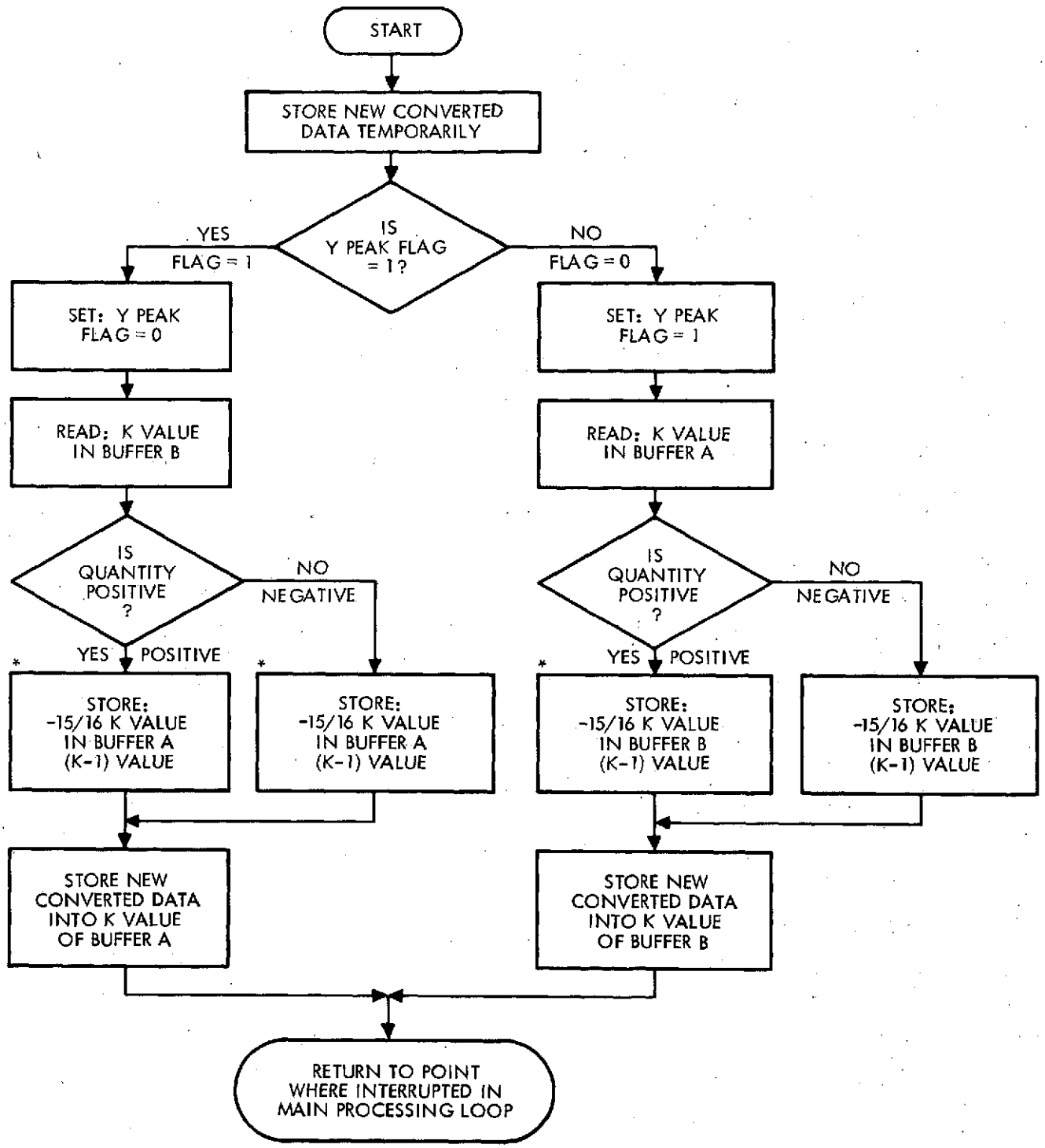


Figure 11. X Peak Interrupt Servicing Flow

Y Peak Interrupt Servicing

Upon recognition by the processor that sample data from the Y axis pickoff is available the Main Processing Loop is interrupted at the end of the instruction it is performing and the program is transferred to the Y Peak Interrupt Servicing routine. The purpose of this routine is to accept the digitized Y axis pickoff data for the computation to be performed by the Main Processing Loop. This data is correlated in much the same manner as previously explained in the X Peak Interrupt Servicing in that a flag (Y Peak Flag) is used to determine which data is time correlated and buffers are used to determine read and write areas. In this process the last iteration Y axis pickoff data $U_Y(K-1)$ is multiplied by 15/16 as a time saving step to the Main Processing Loop. Again, as in the X Peak Interrupt Processing, a return is made to the point in the Main Processing Loop where processing was interrupted.

Following is the flow charts for the Y Peak Interrupt Processing. (Figure 12.)



* -15/16 UY (K-1) = -1 + 1/16 UY (K-1)

T98820

Figure 12. Y Peak Interrupt Servicing Flow

V. ANALOG INTERFACE

The analog interface for the IMP 16 computer consists of an A/D converter, a D/A converter, and the control logic for the two converters. A block diagram of the interface is shown in Figure 13.

The A/D converter section, shown in Figure 14, consists of an input multiplex switch, a sample and hold, a 12-bit converter, and a tri-state buffer register. The input multiplex selects an input channel which may be a peak, zero, or ground input from one of the gyros. This input depends on which channel is selected by the computer and logic decode. The input is applied to the sample and hold circuit, which will hold the input signal value until the A/D performs a conversion. The A/D start command and clock are supplied by the control logic. The tri-state buffer holds the A/D conversion and outputs it to the CPU until the next input enable allows the data to update.

The D/A converter accepts a 13-bit sign-magnitude word from the CPU and converts it to a + or - DC output to drive the gyro torquer amplifier. This converter consists of a $\pm$ DC reference, a 12-bit ladder network, plus $\pm$ output buffer amplifiers. The control logic decodes the input data word and outputs a sign bit so that the proper polarity is applied to the ladder network. The control logic also applies a 12-bit word to the ladder network to determine amplitude. The signal from the ladder network is applied to output buffer amplifiers. The control logic enables either a + or - sample switch which applies the respective output signal to the hold capacitor located on the torquer amplifier on the platform.

The control logic of the analog interface receives the 5.72 MHz CPU clock and divides it by 10 and 12 successively to provide basic square waves of 572 KHz and approximately 48 KHz. The 48 KHz clock is used to generate periodically four sequential analog select signals, each spanning an interval of 32 clock periods. Two successive analog-to-digital conversions are made during an analog select interval, one in the first half-interval and one in the second half-interval. The 48 KHz bit times of each half-interval are designated T_0 through T_{15}. Analog sampling signals are enabled during T_1 of each half-interval such that an analog peak signal is sampled in the first half-interval and an anlaog zero crossing is sampled in the second half-interval. Analog to digital conversion is accomplished during T_2 and T_3 of each half cycle. An interrupt request is then sent to the CPU at T_4. The converted 12-bit analog data may be input to the computer by the command, $R_{IN}\ (0070)_{16}$. The analog data is identified by a three-bit code which is inputted to the computer by the command, $R_{IN}\ (0078)_{16}$.

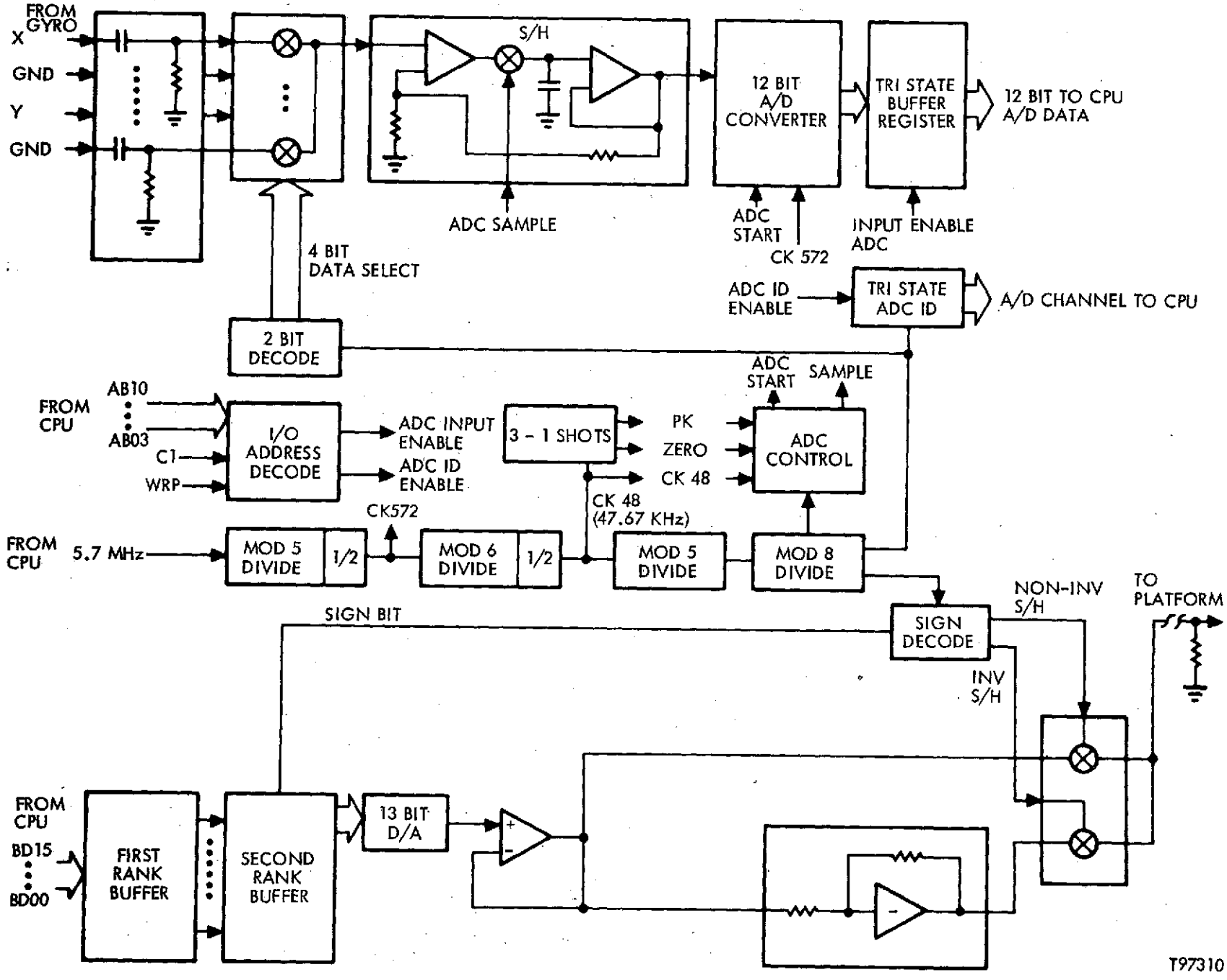


Figure 13. A/D - D/A Control Block Diagram

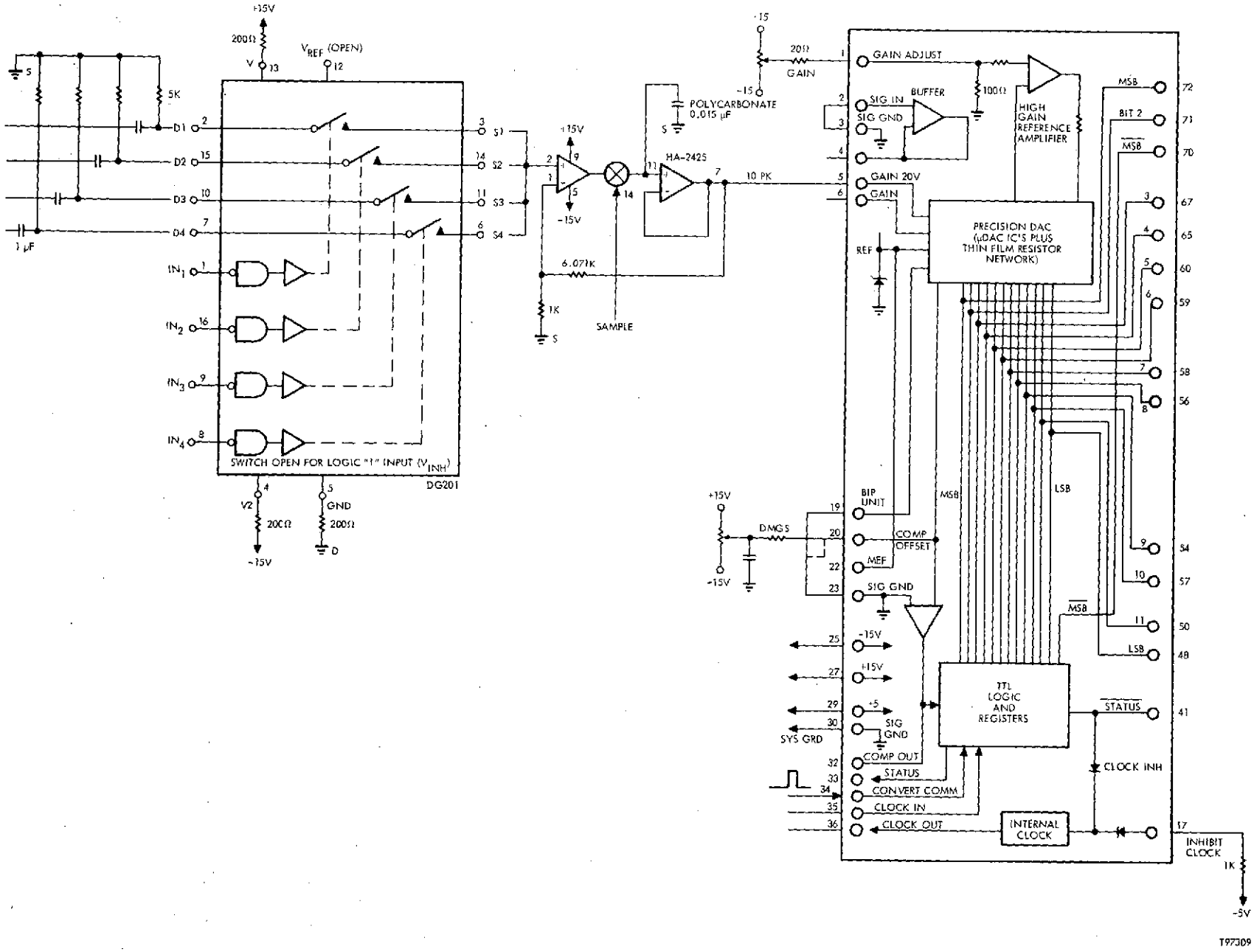


Figure 14. A/D Converter

The CPU outputs a 13-bit word in sign-magnitude format for conversion to analog by the command, $R_{OUT}\ (0078)_{16}$. The data word is latched in a 1st rank buffer from which it is transferred to a 2nd rank buffer at bit time T_0 of a continuous 8-bit cycle of the 48 KHz square wave. The conversion is accomplished during each T_0. Also, sampling of the converted output is disabled during T_0.

VI. TEST RESULTS

Test Description

Figures 15 through 17 illustrate the general test setup. In this test configuration, the gyro was mounted on the rate table such that the input rates would be about the gyro x axis.

The x and y gyro pickoff signals were fed to both the digital servo and the analog servo. The digital servo controlled only the y torquer, and the analog servo controlled only the x torquer.

The rate table was driven by the oscillator portion of the wave analyzer with a constant voltage applied to the rate table.

The wave analyzer was used to measure the torquer signals and pickoff signals. The x axis (direct) pickoff signal was also measured in the open loop condition. The open loop and closed loop measurements were used to calculate the gains that are plotted in Figures 18 and 19.

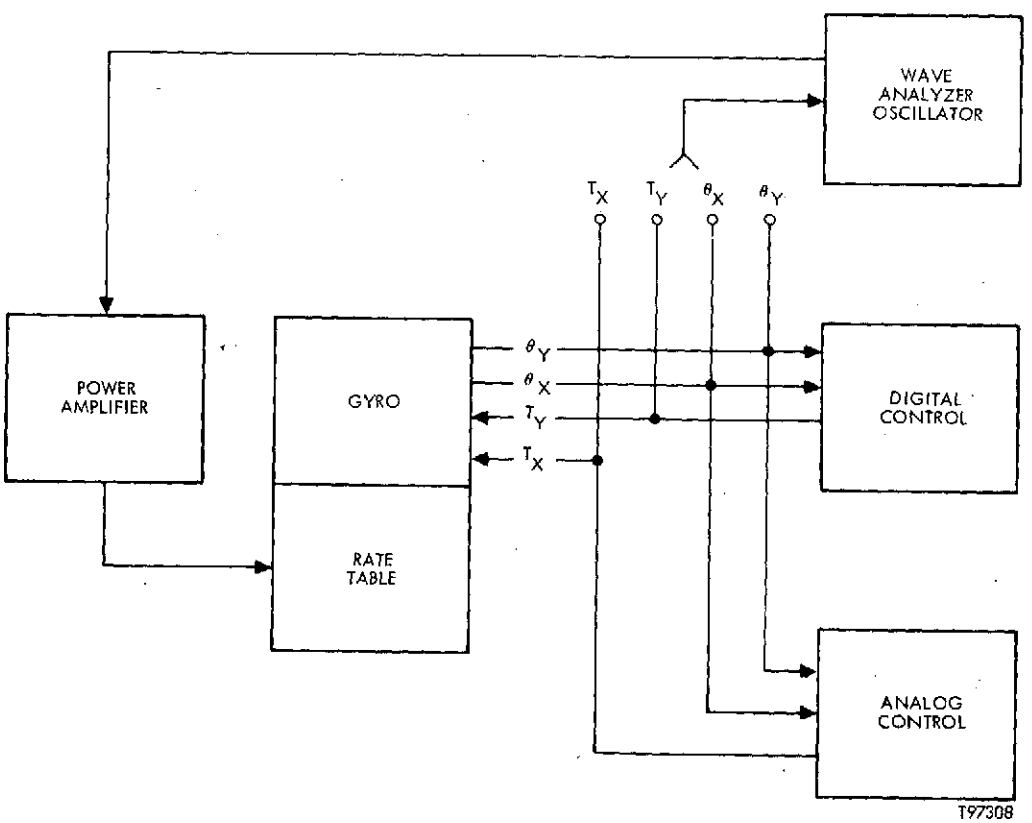


Figure 15. Block Diagram, Test Set Up

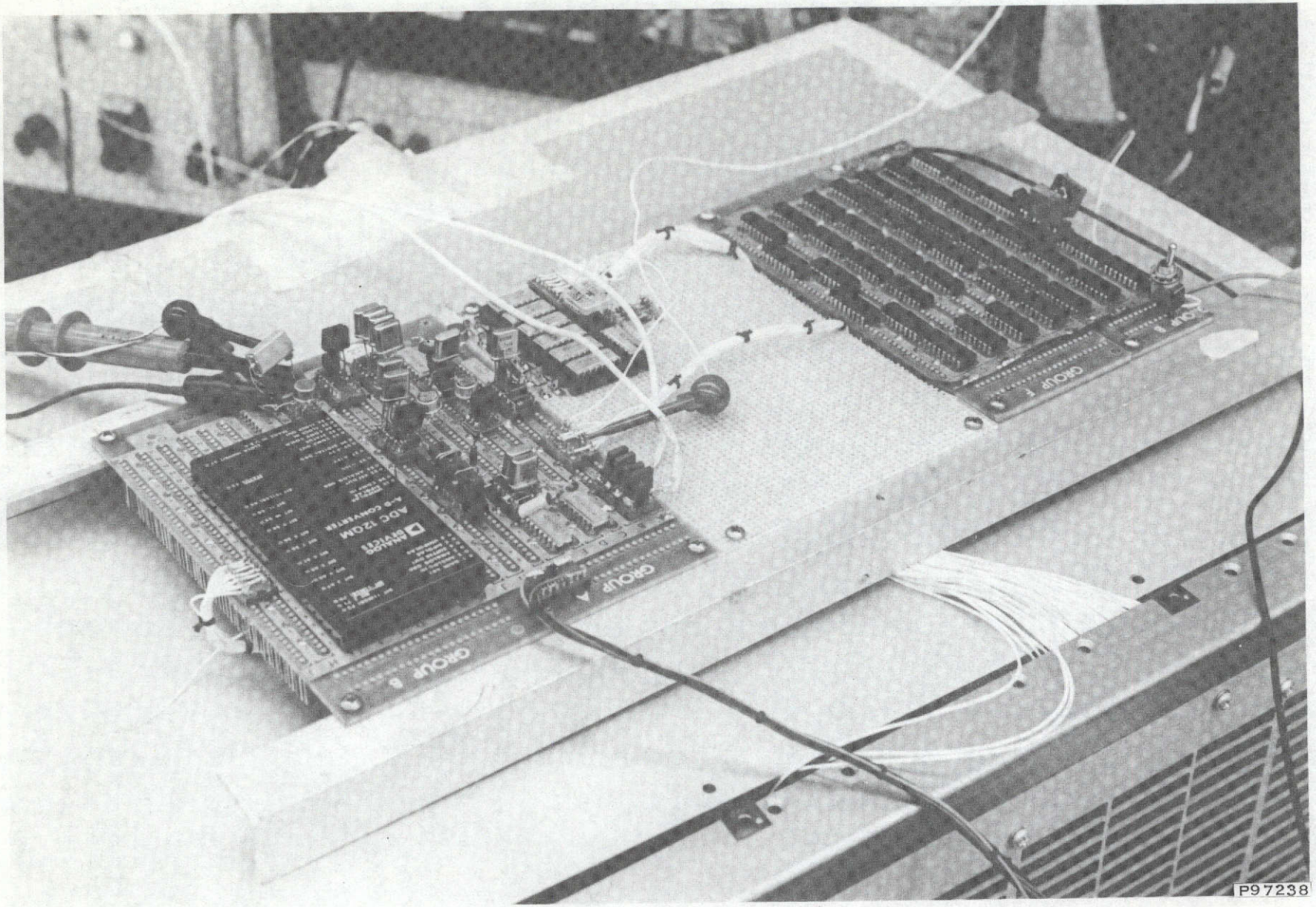
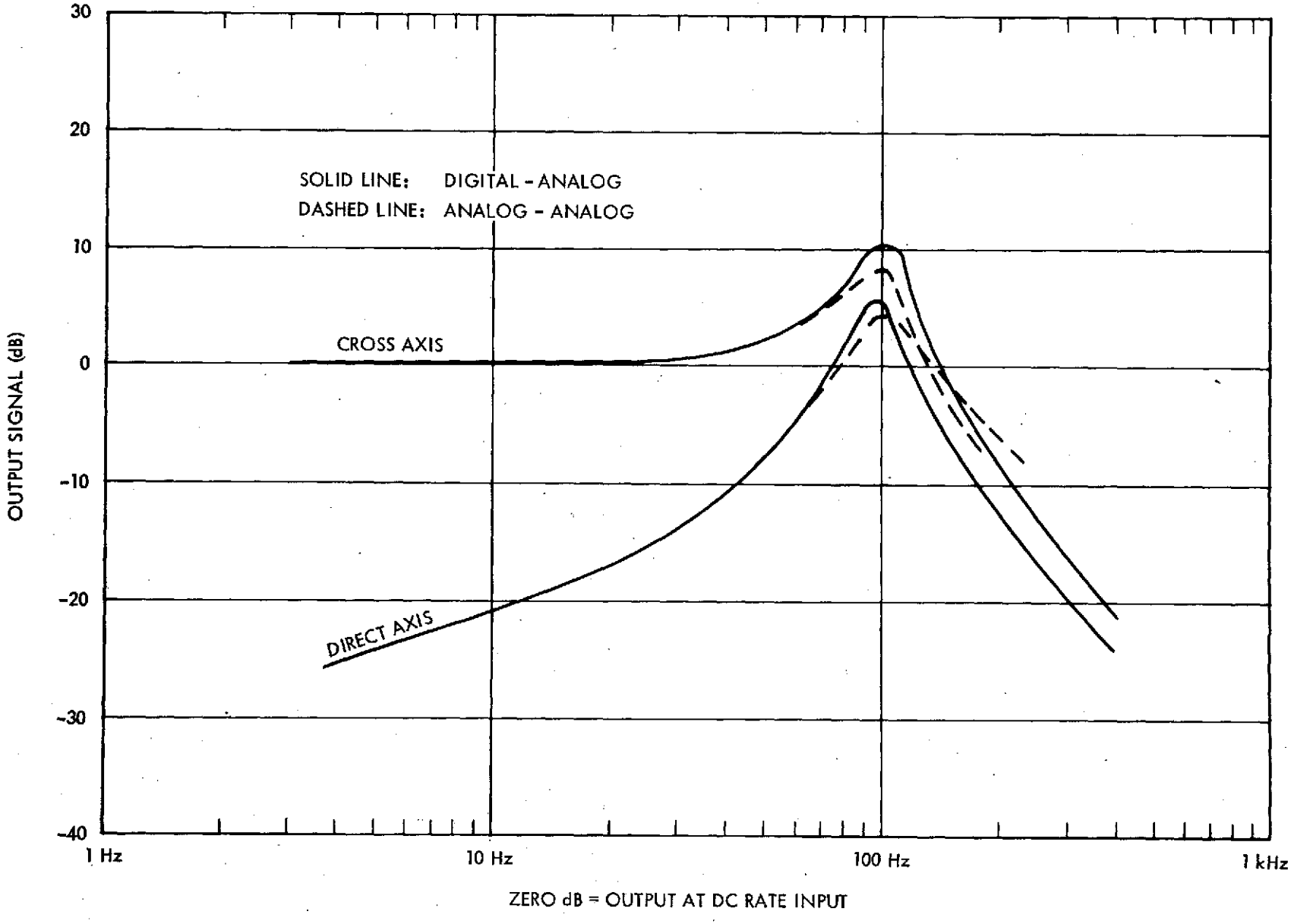


Figure 16. Test Set-Up for Digital Compensation Loop Feasibility Study

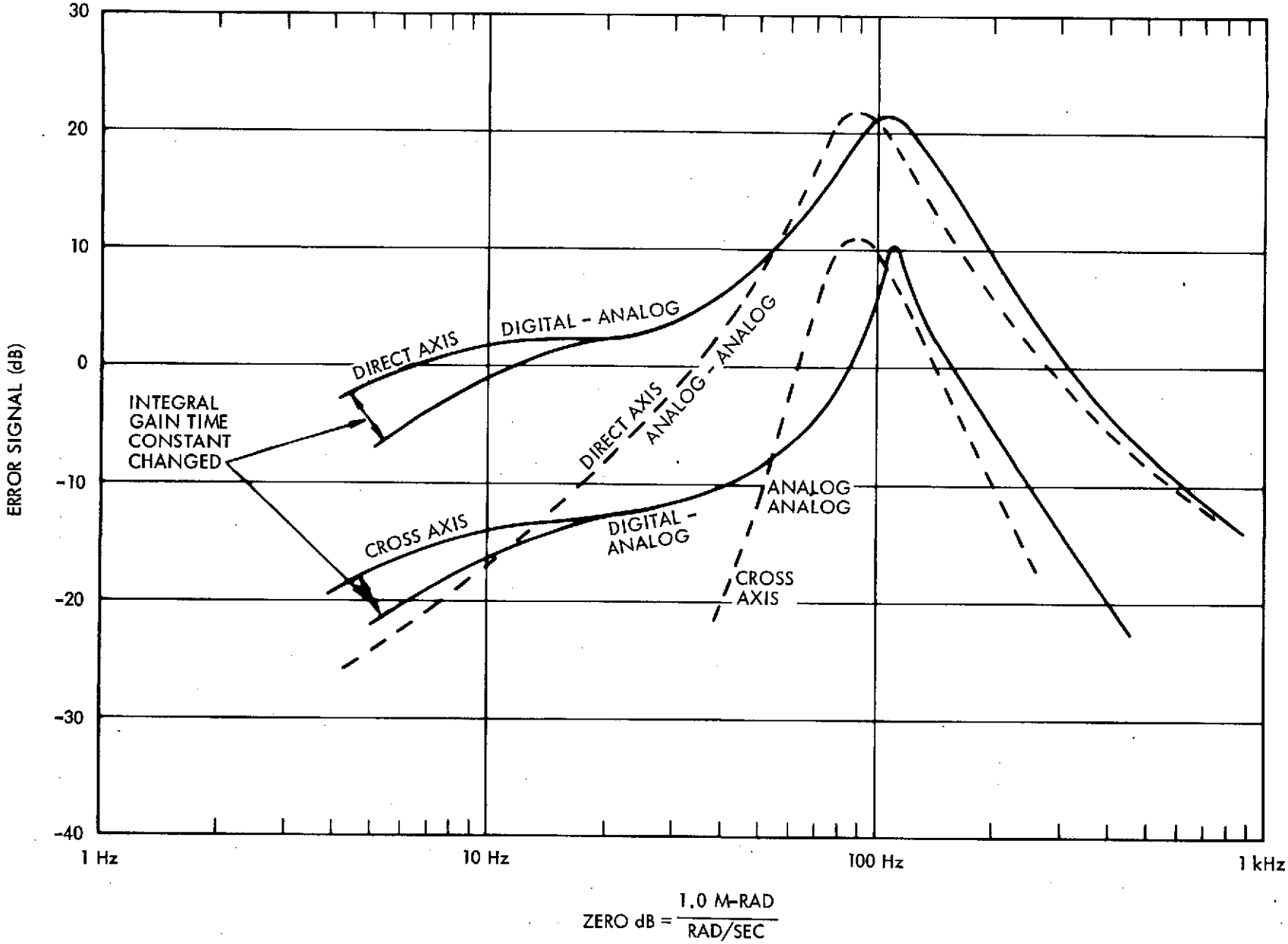
Figure 17. Test Set-up of Converters and Interface for Digital Compensation Loop System



DATA TAKEN 8.2.74 TMW

T97307

Figure 18. Output Signal (Torquer Voltage) vs Frequency
Digital-Analog Loop ~ 100 Hz Bandwidth



DATA TAKEN 8.2.74 TMW

T97306

Figure 19. Error Signal (Pickoff Angle) vs Frequency
Digital-Analog Loop ~ 100 Hz Bandwidth

The rate table drive frequency was varied from 10 Hz to 1 KHz with 3 to 5 measurements taken per decade depending on how rapidly the data was changing with frequency.

The integral gain time constant in the digital servo was modified (software modification) and the test repeated. There were no modifications to the analog servo.

Discussion of Test Results

The test setup described above provided a means for one-to-one comparison of the performance of the digital and analog caging loops. In order to obtain such a comparison, however, it was necessary to match the gains of the two loops since the digital computations, as described in Section 2, assumed that the gain would be adjusted on the analog level, i.e., in the power amplifier. The criterion used to match the gain (applied torque vs pickoff angle) of the digital loop to that of the analog loop was to operate each servo 6 db below its marginal stability level. Using this criterion it was found that the digital loop gain was approximately 10 db below that of the analog loop.

Frequency response data was taken for both the digital and analog loops. Figure 20 shows the output signal (i.e., the torquer voltage) response versus frequency. Figure 21 shows the error signal (i.e., pickoff angle) versus frequency. These figures are actually families of responses showing variations resulting from adjusting the "integral gain time constant" in the digital loop. This time constant is directly related to the location of the zero of the digital transfer functions described in Section 2. The range or variation corresponds to varying this zero location between 123 Hz and 12.7 Hz.

It is apparent from Figure 18 that the closed loop response of the digital loop compares very favorably with that of the analog loop in terms of the torquer output versus frequency. This quantity is, of course, of primary interest as it is the ultimate measure of gyro rate.

Figure 19 indicates that the gyro pickoff angles are substantially larger for the digital loop than for the analog loop for low frequency inputs. This is attributable to the lower gain in the digital loop which was established by the gain matching criterion described above.

The primary reason that the marginal stability gain for the digital loop is lower than for the analog loop is the effective phase shift introducted by the finite computer iteration rate. Thus as the computational frequency is increased it is expected that the digital loop gain established by the gain matching procedure would be increased and the pickoff angle versus frequency response of the digital loop would more closely approximate the corresponding analog response. Unfortunately this conclusion could not readily be verified due to computer timing limitations.

The primary ramifications of the lower loop gain in the digital loop is a lowering of the bandwidth of the caging loop which would occur in a gyro with both axes mechanized digitally. Subsequent analysis indicates that the effective bandwidth of such an all-digital gyro would be approximately 25 Hz for the configuration which was tested (i.e., 1200 Hz digital processing). This corresponds closely with the 30 Hz bandwidth used in Teledyne's current analog caging loops. This bandwidth can be further increased by increasing the computational frequency and can be expected to approach the 100 Hz of the new analog design with very high speed processing.

In summary, then, the test results were in general concert with expectations and the feasibility of using digital filtering in place of conventional analog caging loops for the TDF tuned-gimbal gyroscope was demonstrated.

VII. SUMMARY AND CONCLUSIONS

The primary objective of this program was to demonstrate the feasibility of replacing existing state-of-the-art analog gyro compensation loops with digital computations. This objective was realized during the course of the program.

A breadboard design was established in which one axis of a Teledyne tuned-gimbal TDF gyro was caged digitally while the other was caged using conventional analog electronics. The digital loop was designed analytically to closely resemble the analog loop in performance. The breadboard was subjected to various static and dynamic tests in order to establish the relative stability characteristics and frequency responses of the digital and analog loops. Several variations of the digital loop configuration were evaluated. The results were very favorable - it appears that digital caging is indeed a practical approach.

The primary limitation to the digital compensation loop approach appears to be the requirement for high processing rates. The tests which were performed indicate that the minimum processing rate for acceptable loop performance is approximately 1200 Hz. Using the general purpose IMP-16 on TDY-52B computers essentially all of the available computer time was required to perform the digital caging loop functions for a single gyro axis at this rate. (These functions include all timing, control, sampling, etc., as well as the implementation of the actual digital filters.) For this reason it appears desirable to consider a special purpose digital processor for the digital caging loop task. Such a processor could be quite simple in concept since the only arithmetic functions required to implement the digital filters which were derived during the program are adds, subtracts, and shifts.

The primary advantages of using digital caging loops in place of conventional analog loops are reduction of cost, size, weight, and power, increased reliability, and simplified maintenance, resulting from the elimination of a substantial amount of hardware from the IMU Electronics. The electronics which are eliminated using the digital technique include pickoff demodulators, caging electronics, spin supplies, clock and timing functions, and pickoff excitation generation. New functions which must be performed with the digital loop are digital-to-analog conversion and, optionally, the special purpose digital processing. Since both of these functions may be multiplexed to handle all sensor axes, the hardware required is minimal. A preliminary design analysis has indicated the hardware requirement to perform these two

functions is of the order of 5 to 20% of that required for the analog functions which they replace depending upon the level of redundancy which is employed in the system. The resulting savings in direct acquisition cost for a fail-op/fail-op redundant strapdown navigator is projected to be approximately $4000 per system in quantities of 150 systems.

The performance of a system using digital caging loops is expected to be virtually identical to that using conventional analog technology. This is particularly true if a special purpose processor is employed, since the processing rate can be increased beyond the 1200 Hz which was used in the demonstration. Higher rates will provide digital loop performance which matches the analog loop performance even more closely than described in this report.

Other than the limitations imposed by computational speed, the digital loop which was breadboarded during the program appears to be suitable for use in a practical system design without any significant modifications. The general design philosophy, filtering algorithms, computer word length, A/D and D/A converter accuracies and resolutions, etc., which were employed in the breadboard design are all suitable for use in a production design.

APPENDIX A

PROGRAM LISTING

This is the assembly language gyro program as assembled on Teledyne unspecialized assembler TUMPA. Assembled code was punched out in special loading format on mylar teletype tape for loading via teletype into the IMP-16P. This program will also run on the TDY-52B.

***** TELEDYNE UNSPECIALIZED MACRO ASSEMBLER

B BAD LITERAL INITIAL CHARACTER
O OTHER THAN OCTAL DIGIT
O OCTAL CHARACTER IN OCTAL FIELD
K TOO MANY CHARACTERS IN FIELD
P PROGRAM COUNTER PROBLEM
S MEANINGLESS $PAU LINE
Q MACRO OUT IN TABLE
O MACRO TOO DEEPLY NESTED

```
C001 $.C/ GYRO PROGRAM VERSION 1.7 TAPE 987
C002 $.PC,20,1/
C003 $.PCIF,2,LDB,2/
C004 $.PCIF,2,STB,2/
C005 $.PCIF,2,DADD,2/
C006 $.PCIF,2,DSUB,2/
C007 $.PCIF,2,MPY,2/
C008 $.PCIF,2,DIV,2/
C009 $.TRAN,3,2/
C010 $.TRAN,4,2/
C011 $.TRAN,3,1/
C012 $.TRAN,4,1/
C013 $.TRAN,5,1/
C014 $.BOOL,2,R0,0/
C015 $.BOOL,2,R1,1/
C016 $.BOOL,2,R2,2/
C017 $.BOOL,2,R3,3/
C018 $.BOOL,2,AC0,0/
C019 $.BOOL,2,AC1,1/
C020 $.BOOL,2,AC2,2/
C021 $.BOOL,2,AC3,3/
C022 $.BOOL,2,BASE,0/
C023 $.BOOL,2,PC,1/
C024 $.BOOL,2,XR2,2/
C025 $.BOOL,2,XR3,3/
C026 $.BOOL,2,XMIT,4/
C027 $.BOOL,2,LNK,17/
C028 $.BOOL,2,OV,16/
C029 $.BOOL,2,CY,15/
C030 $.BOOL,2,ION,1/
C031 $.BOOL,2,SEL,2/
C032 $.BOOL,2,READR,3/
C033 $.BOOL,2,XMIT,4/
C034 $.BOOL,2,JC14,16/
C035 $.BOOL,2,JC1,1/
C036 $.BOOL,2,JC2,2/
C037 $.BOOL,2,JC3,3/
```

```
C038 $.BOOL,2,JC4,4/
C039 $.BOOL,2,JC5,5/
C040 $.BOOL,2,JC8,10/
C041 $.BOOL,2,JC9,11/
C042 $.BOOL,2,JC10,12/
C043 $.BOOL,2,JC11,13/
C044 $.FEED/
```

```
                C045 $.ORG,1/
                C046 $.C/ INTERRUPT EXECUTIVE
          00001 C047   ,PUSH,AC0/
          00002 C048   ,PUSH,AC3/
          00003 C049   ,LI,AC3,0/
          00004 C050   ,RIN,170/
          00005 C051 XX,RCPY,AC0,AC3/
          00006 C052   ,JMP,AC3,0/
                C053 $.ORG,10/
                C054 $.C/ X PEAK BUFFERS
          00010 C055 XP1,DCT,0/ FLAG1
          00011 C056 XP2,DCT,0/ CELL2
          00012 C057 XP3,DCT,0/ CELL3
          00013 C058 XP4,DCT,0/ CELL4
          00014 C059 XP5,DCT,0/ FLAG2
          00015 C060 XP6,DCT,0/ CELL5
          00016 C061 XP7,DCT,0/ CELL6
          00017 C062 XP8,DCT,0/ CELL7
          00020 C063 C10,DCT,40000/
          00021 C064 C14,DCT,140000/
          00022 C065 MT,DCT,0/
                C066 $.ORG,30/
                C067 $.C/ Y PEAK BUFFERS
          00030 C068 YP1,DCT,0/ FLAG1
          00031 C069 YP2,DCT,0/ CELL1
          00032 C070 YP3,DCT,0/ CELL2
          00033 C071 YP4,DCT,0/ FLAG2
          00034 C072 YP5,DCT,0/ CELL3
          00035 C073 YP6,DCT,0/ CELL4
                C074 $.ORG,40/
          00040 C075 XX1,RCPY,AC0,AC3/
          00041 C076 XX2,JMP,BASE,FGD/
          00042 C077 FGD,LD,AC0,BASE,XX1/
          00043 C078   ,ST,AC0,BASE,XX/
          00044 C079   ,PULL,AC3/
          00045 C080   ,PLLL,AC0/
          00046 C081   ,RTI,0/
                C082 $.ORG,100/
                C083 $.C/ DATA STORAGE POINTER
          00100 C084 DSPC,DCT,4000/
          00101 C085 DSP,DCT,0/
          00102 C086 SIGN,DCT,100000/
                C087 $.C/ GYRO MAIN
          00103 C088 MX,DCT,0/
          00104 C089 M,DCT,0/
                C090 $.ORG,1000/
                C091 $.C/ X PEAK INTERRUPT SERVICE
          01000 C092   ,LI,AC3,0/
          01001 C093   ,RIN,160/
          01002 C094   ,PUSH,AC0/
          01003 C095   ,LD,AC0,BASE,XP1/
```

```
01004 C096    ,BCC*,1,XPEAK2/
01005 C097    ,LI,ACC,0/
01006 C098    ,ST,ACC,BASE,XP1/
01007 C099    ,LD,ACC,BASE,XP7/
01010 C1C0    ,ST,ACC,BASE,XP4/
01011 C101 XPEAK4,LD,ACC,BASE,XP6/
01012 C102    ,BCC*,2,XPEK4/
01013 C103    ,CAI,ACC,1/
01014 C104    ,SHR,ACC,4/
01015 C105    ,CAI,ACC,1/
01016 C106    ,ST,ACC,BASE,XP3/
01017 C107    ,JMP*,XPEK5/
01020 C108 XPEK4,SHR,ACC,4/
01021 C109    ,ST,ACC,BASE,XP3/
01022 C110 XPEK5,PULL,ACC/
01023 C111    ,ST,ACC,BASE,XP2/
01024 C112    ,PULL,AC3/
01025 C113    ,PULL,ACC/
01026 C114    ,RTI,0/
01027 C115 XPEAK2,LI,ACC,1/
01030 C116    ,ST,ACC,BASE,XP1/
01031 C117    ,LD,ACC,BASE,XP3/
01032 C118    ,ST,ACC,BASE,XP8/
01033 C119 XPEAK6,LD,ACC,BASE,XP2/
01034 C120    ,BCC*,2,XPEAK7/
01035 C121    ,CAI,ACC,1/
01036 C122    ,SHR,ACC,4/
01037 C123    ,CAI,ACC,1/
01040 C124    ,ST,ACC,BASE,XP7/
01041 C125    ,JMP*,XPEAK8/
01042 C126 XPEAK7,SHR,ACC,4/
01043 C127    ,ST,ACC,BASE,XP7/
01044 C128 XPEAK8,PULL,ACC/
01045 C129    ,ST,ACC,BASE,XP6/
01046 C130    ,PULL,AC3/
01047 C131    ,PULL,ACC/
01050 C132    ,RTI,0/
      C133 $,CRG,1200/
      C134 $,C/ Y PEAK INTERRUPT SERVICE
01200 C135 YPEAK,LI,AC3,0/
01201 C136    ,RIN,160/
01202 C137    ,PUSH,ACC/
01203 C138    ,LD,ACC,BASE,YP1/
01204 C139    ,BCC*,1,YPEAK2/
01205 C140    ,LI,ACC,0/
01206 C141    ,ST,ACC,BASE,YP1/
01207 C142    ,LD,ACC,BASE,YP5/
01210 C143    ,BCC*,2,YPEAK3/
01211 C144    ,CAI,ACC,1/
01212 C145    ,SHR,ACC,4/
01213 C146    ,CAI,ACC,4/
```

```
01214 C147    ,SUB,ACO,BASE,YP5/
01215 C148    ,ST,ACC,BASE,YP3/
01216 C149    ,JMP*,YPEAK4/
01217 C150 YPEAK3,SHR,ACO,4/
01220 C151    ,SUB,ACO,BASE,YP5/
01221 C152    ,ST,ACC,BASE,YP3/
01222 C153 YPEAK4,PLLL,ACO/
01223 C154    ,ST,ACC,BASE,YP2/
01224 C155    ,PLLL,AC3/
01225 C156    ,PLLL,ACO/
01226 C157    ,RTI,0/
01227 C158 YPEAK2,LI,ACO,1/
01230 C159    ,ST,ACC,BASE,YP1/
01231 C160    ,LD,ACC,BASE,YP2/
01232 C161    ,BCC*,2,YPEAK5/
01233 C162    ,CAI,ACO,1/
01234 C163    ,SHR,ACO,4/
01235 C164    ,CAI,ACO,1/
01236 C165    ~~,SUB,ACO,YP2/~~                    ,SUB,ACφ, BASE, YP2/
01237 C166    ,ST,ACO,BASE,YP6/
01240 C167    ,JMP*,YPEAK6/
01241 C168 YPEAK5,SHR,ACO,4/
01242 C169    ,SUB,ACO,BASE,YP2/
01243 C170    ,ST,ACC,BASE,YP6/
01244 C171 YPEAK6,PLLL,ACO/
01245 C172    ,ST,ACC,BASE,YP5/
01246 C173    ,PLLL,AC3/
01247 C174    ,PLLL,ACO/
01250 C175    ,RTI,0/
      C176 $.ORG,1500/
      C177 $.C/ DATA STORAGE INTERRUPT SERVICE
01500 C178 DS,LD,ACC,BASE,DSP/
01501 C179    ,SHR,ACC,14/
01502 C180    ,BCC*,3,DS6/
01503 C181    ,LD,ACC,BASE,XP1/
01504 C182    ,BCC*,1,DS2/
01505 C183    ,LD,ACC,BASE,Xp6/
01506 C184    ,JMP*,DS3/
01507 C185 DS2,LD,ACO,BASE,XP2/
01510 C186 DS3,LD,AC3,BASE,DSP/
01511 C187    ,ST,ACC,AC3,0/
01512 C188    ,LI,ACO,1/
01513 C189    ,RADD,ACO,AC3/
01514 C190    ,LD,ACC,BASE,YP1/
01515 C191    ,BCC*,1,DS4/
01516 C192    ,LD,ACO,BASE,YP5/
01517 C193    ,JMP*,DS5/
01520 C194 DS4,LD,ACO,BASE,YP2/
01521 C195 DS5,ST,ACO,AC3,0/
01522 C196    ,LI,ACC,1/
01523 C197    ,RADD,ACO,AC3/
```

```
01524 C198    ,LD,ACO,BASE,MT/
01525 C199    ,ST,ACO,AC3,0/
01526 C200    ,LI,ACO,1/
01527 C201    ,RADD,ACO,AC3/
01530 C202    ,ST,AC3,BASE,DSP/
01531 C203 D56,PULL,AC3/
01532 C204    ,PULL,ACO/
01533 C205    ,RTI,0/
      C206 $,ORG,2000/
      C207 $,C/ GYPO MAIN PROGRAM
02000 C208 MAIN,LD,ACO,BASE,XX2/
02001 C209    ,ST,ACC,BASE,XX/
02002 C210    ,LI,AC3,0/
02003 C211    ,LD,ACC,BASE,DSPC/
02004 C212    ,ST,ACC,BASE,DSP/
02005 C213    ,LI,ACO,0/
02006 C214    ,ST,ACC,BASE,MX/
02007 C215    ,SFLG,IEN,0/
02010 C216 RET,LD,ACO,BASE,XP1/
02011 C217    ,BCC*,1,MAIN2/
02012 C218    ,LD,ACC,BASE,XP8/
02013 C219    ,SHL,ACO,4/
02014 C220    ,CAI,ACO,1/
02015 C221    ,ADD,ACO,BASE,XP8/
02016 C222    ,ADD,ACO,BASE,XP6/
02017 C223    ,ADD,ACO,BASE,XP7/
02020 C224    ,ADD,ACO,BASE,MX/
02021 C225    ,ST,ACO,BASE,MX/
02022 C226    ,JMP*,MAIN3/
02023 C227 MAIN2,LD,ACO,BASE,XP4/
02024 C228    ,SHL,ACO,4/
02025 C229    ,CAI,ACO,1/
02026 C230    ,ADD,ACO,BASE,XP4/
02027 C231    ,ADD,ACO,BASE,XP2/
02030 C232    ,ADD,ACO,BASE,XP3/
02031 C233    ,ADD,ACO,BASE,MX/
02032 C234    ,ST,ACC,BASE,MX/
02033 C235 MAIN3,BCC*,2,MAIN4/
02034 C236 ,CAI,ACO,1/
02035 C237 ,SHL,ACO,1/
02036 C238 ,BCC*,2,MAIN5/
02037 C239 ,LD,ACO,BASE,C14/
02040 C240 ,ST,ACO,BASE,MX/
02041 C241 ,JMP*,MAIN5/
02042 C242 MAIN4,SHL,ACO,1/
02043 C243 ,BCC*,2,MAIN5/
02044 C244 ,LD,ACO,BASE,C10/
02045 C245 ,ST,ACO,BASE,MX/
02046 C246 MAIN5,LD,ACC,BASE,YP1/
02047 C247 ,BCC*,1,MAIN6/
02050 C248    ,LI,ACC,0/
```

```
02051 C249    ,ADD,ACO,BASE,YP5/
02052 C250    ,ADD,ACO,BASE,YP6/
02053 C251    ,ST,ACO,BASE,M/
02054 C252    ,JMP*,MAIN7/
02055 C253 MAIN6,LI,ACO,0/
02056 C254    ,ADD,ACO,BASE,YP2/
02057 C255    ,ADD,ACO,BASE,YP3/
02060 C256    ,ST,ACO,BASE,M/
02061 C257 MAIN7,LD,ACO,BASE,MX/
02062 C258    ,SUB,ACO,BASE,M/
02063 C259    ,ST,ACO,BASE,M/
02064 C260 MAIN10,BEC*,2,MAIN11/
02065 C261 ,CAI,ACO,1/
02066 C262 ,OR,ACO,BASE,SIGN/
02067 C263 MAIN11,LI,AC3,0/
02070 C264 ,RCUT,17C/
02071 C265    ,ST,ACO,BASE,MT/
02072 C266 ,JMP*,RET/
      C267 $.C/ PROGRAM SECTION DUMP
02073 C268 DUMP,LD,AC3,BASE,DSPC/
02074 C269 DP2,RCPY,AC3,ACO/
02075 C270    ,SHR,ACO,14/
02076 C271    ,BEC*,3,DP2/
02077 C272    ,LD,ACO,AC3,0/
02100 C273    ,SHR,ACO,10/
02101 C274    ,LD*,AC2,PUTC./
02102 C275    ,JSR,AC2,0/
02103 C276    ,LD,ACO,AC3,0/
02104 C277    ,AND*,ACO,MASK/
02105 C278    ,LD*,AC2,PUTC./
02106 C279    ,JSR,AC2,0/
02107 C280    ,LI,ACO,1/
02110 C281    ,RADD,ACO,AC3/
02111 C282    ,JMP*,DP2/
02112 C283 PUTC.,OCT,771131/
02113 C284 MASK,OCT,377/
      C285 $.END/
      C286 $.FEED/
```

CROSS REFFRENCE TABLE LISTING

```
C10     00020 C2044
C14     00021 C2037
DP2     02074 C2076 02111
DS      01500
DS2     01507 C1504
DS3     01510 C1506
DS4     01520 C1515
DS5     01521 C1517
DS6     01531 C1502
DSP     00101 C1500 01510 01530 02004
DSPC    00100 C2003 02073
DUMP    02073
FGU     00042 C0041
M       00104 C2053 02060 02062 02063
MAIN    02000
MAIN10  02064
MAIN11  02067 C2064
MAIN2   02023 C2011
MAIN3   02033 C2022
MAIN4   02042 C2033
MAIN5   02046 C2036 02041 02043
MAIN6   02055 C2047
MAIN7   02061 C2054
MASK    02112 C2104
MT      00022 C1524 02071
MX      00103 C2006 02020 02021 02031 02032 02040 02045 02061
PLTC.   02112 C2101 02105
RET     02010 C2072
SIGN    00102 C2066
XP1     00010 C1003 01006 01030 C1503 02010
XP2     00011 C1023 01033 01507 02027
XP3     00012 C1016 C1021 01031 02030
XP4     00013 C1010 02023 02026
XP5     00014
XP6     00015 C1011 01045 01505 02016
XP7     00016 C1007 01040 01043 02017
XP8     00017 C1032 02012 02015
XPEAK2  01027 C1004
XPEAK4  01011
XPEAK6  01033
XPEAK7  01042 C1034
XPEAK8  01044 C1041
XPEK4   01020 C1012
XPEK5   01022 C1017
XX      00001 C0043 02001
XX1     00040 C0042
XX2     00041 C2000
YP1     00030 C1203 01206 01230 C1514 02046
YP2     00031 C1223 01231 01236 01242 01520 02056
YP3     00032 C1215 01221 02057
```

```
YP4    00033
YP5    00034 C1207 01214 01220 01245 01516 02051
YP6    00035 C1237 01243 02052
YPEAK  01200
YPEAK2 01227 C1204
YPEAK3 01217 C1210
YPEAK4 01222 C1216
YPEAK5 01241 01232
YPEAK6 01244 C1240
             C298 $.FEED/
```

```
C289 $.C/ THESE CARDS SPECIFY WORD ASSEMBLY
C290 $.IF,2,LD/
C291 $.SETC,6,10/
C292 $.JUMP,A/
C293 $.IF,2,LI/
C294 $.SETC,5,23/
C295 $.JUMP,B/
C296  $.IF,2,LIN/
C297 $.SETC,5,23/
C298 $.SETC,6,0/
C299 $.SUB,6,4,4/
C300 $.JUMP,B/
C301 $.IF,2,LDI/
C302 $.SETC,6,11/
C303 $.JUMP,A/
C304 $.IF,2,LDB/
C305 $.SETC,5,2300/
C306 $.JUMP,K/
C307 $.IF,2,ST/
C308 $.SETC,6,12/
C309 $.JUMP,A/
C310 $.IF,2,STI/
C311 $.SETC,6,13/
C312 $.JUMP,A/
C313 $.IF,2,STB/
C314 $.SETC,5,2320/
C315 $.JUMP,K/
C316 $.IF,2,RXCH/
C317 $.SETC,5,34/
C318 $.JUMP,D/
C319 $.IF,2,RCPY/
C320 $.SETC,5,35/
C321 $.JUMP,D/
C322 $.IF,2,PUSH/
C323 $.SETC,4,20/
C324 $.JUMP,C/
C325 $.IF,2,PULL/
C326 $.SETC,4,21/
C327 $.JUMP,C/
C328 $.IF,2,XCHRS/
C329 $.SETC,4,25/
C330 $.JUMP,C/
C331 $.IF,2,PUSHF/
C332 $.SETC,3,1/
C333 $.JUMP,L/
C334 $.IF,2,PULLF/
C335 $.SETC,3,5/
C336 $.JUMP,L/
C337 $.IF,2,ADD/
C338 $.SETC,6,14/
C339 $.JUMP,A/
```

```
0340 $.IF,2,DADD/
0341 $.SETC,5,2240/
0342 $.JUMP,K/
0343 $.IF,2,CAI/
0344 $.SETC,5,24/
0345 $.JUMP,B/
0346 $.IF,2,RADD/
0347 $.SETC,5,30/
0348 $.JUMP,D/
0349 $.IF,2,SUB/
0350 $.SETC,6,15/
0351 $.JUMP,A/
0352 $.IF,2,DSUB/
0353 $.SETC,5,2260/
0354 $.JUMP,K/
0355 $.IF,2,MPY/
0356 $.SETC,5,2200/
0357 $.JUMP,K/
0358 $.IF,2,DIV/
0359 $.SETC,5,2220/
0360 $.JUMP,K/
0361 $.IF,2,AND/
0362 $.SETC,6,14/
0363 $.JUMP,G/
0364 $.IF,2,RAND/
0365 $.SETC,5,37/
0366 $.JUMP,D/
0367 $.IF,2,OR/
0368 $.SETC,6,13/
0369 $.JUMP,G/
0370 $.IF,2,RXOR/
0371 $.SETC,5,36/
0372 $.JUMP,D/
0373 $.IF,2,SETBIT,3/
0374 $.SETC,4,162/
0375 $.JUMP,J/
0376 $.IF,2,CLRBIT,4/
0377 $.SETC,4,163/
0378 $.JUMP,J/
0379 $.IF,2,CMPBIT,13/
0380 $.SETC,4,166/
0381 $.JUMP,J/
0382 $.IF,2,SETST/
0383 $.SETC,4,160/
0384 $.JUMP,J/
0385 $.IF,2,CLRST/
0386 $.SETC,4,161/
0387 $.JUMP,J/
0388 $.IF,2,AISZ/
0389 $.SETC,5,22/
0390 $.JUMP,R/
```

```
C391 $.IF,2,ISZ/
C392 $.SETC,5,36/
C393 $.JUMP,B/
C394 $.IF,2,DSZ/
C395 $.SETC,5,37/
C396 $.JUMP,B/
C397 $.IF,2,SKG/
C398 $.SETC,6,16/
C399 $.JUMP,A/
C400 $.IF,2,SKNE/
C401 $.SETC,6,17/
C402 $.JUMP,A/
C403 $.IF,2,SKAZ/
C404 $.SETC,6,16/
C405 $.JUMP,G/
C406 $.IF,2,SKBIT/
C407 $.SETC,4,165/
C408 $.JUMP,J/
C409 $.IF,2,SKSTF/
C410 $.SETC,4,164/
C411 $.JUMP,J/
C412 $.IF,2,ISCAN/
C413 $.SETC,3,2420/
C414 $.JUMP,OCT/
C415 $.IF,2,JNP/
C416 $.SETC,5,10/
C417 $.JUMP,B/
C418 $.IF,2,JNPI/
C419 $.SETC,5,11/
C420 $.JUMP,B/
C421 $.IF,2,JNPP/
C422 $.SETC,4,120/
C423 $.JUMP,J/
C424 $.IF,2,BCC/
C425 $.SETC,5,1/
C426 $.JUMP,E/
C427 $.IF,2,HALT/
C428 $.SETC,3,0/
C429 $.JUMP,L/
C430 $.IF,2,JSR/
C431 $.SETC,5,12/
C432 $.JUMP,B/
C433 $.IF,2,JSRP/
C434 $.SETC,4,6/
C435 $.JUMP,H/
C436 $.IF,2,JSRI/
C437 $.SETC,5,13/
C438 $.JUMP,B/
C439 $.IF,2,JSRIN/
C440 $.SETC,4,7/
C441 $.JUMP,H/
```

```
C442 $.IF,2,JINTC/
C443 $.SETC,4,122/
C444 $.JUMP,J/
C445 $.IF,2,JINT/
C446 $.SETC,4,122/
C447 $.JUMP,J/
C448 $.IF,2,RTS/
C449 $.SETC,4,4/
C450 $.JUMP,H/
C451 $.IF,2,RTI/
C452 $.SETC,4,2/
C453 $.JUMP,H/
C454 $.IF,2,RCL/
C455 $.SETC,5,26/
C456 $.JUMP,B/
C457 $.IF,2,RCR/
C458 $.SETC,5,26/
C459 $.JUMP,BR/
C460 $.IF,2,SHL/
C461 $.SETC,5,27/
C462 $.JUMP,R/
C463 $.IF,2,SHR/
C464 $.SETC,5,27/
C465 $.JUMP,BR/
C466 $.IF,2,RIN/
C467 $.SETC,4,10/
C468 $.JUMP,H/
C469 $.IF,2,ROUT/
C470 $.SETC,4,14/
C471 $.JUMP,H/
C472 $.IF,2,SFLG/
C473 $.SETC,5,2/
C474 $.JUMP,E/
C475 $.IF,2,PFLG/
C476 $.SETC,5,3/
C477 $.JUMP,E/
C478 $.IF,2,LC*/
C479 $.SETC,5,10/
C480 $.JUMP,APC/
C481 $.IF,2,LDI*/
C482 $.SETC,5,11/
C483 $.JUMP,APC/
C484 $.IF,2,ST*/
C485 $.SETC,5,12/
C486 $.JUMP,APC/
C487 $.IF,2,STI*/
C488 $.SETC,5,13/
C489 $.JUMP,APC/
C490 $.IF,2,ADD*/
C491 $.SETC,5,14/
C492 $.JUMP,APC/
```

```
0493 $.IF,2,SUB*/
0494 $.SETC,5,15/
0495 $.JUMP,APC/
0496 $.IF,2,AND*/
0497 $.SETC,5,14/
0498 $.JUMP,GPC/
0499 $.IF,2,OR*/
0500 $.SETC,5,15/
0501 $.JUMP,GPC/
0502 $.IF,2,SKG*/
0503 $.SETC,5,16/
0504 $.JUMP,APC/
0505 $.IF,2,SKGE*/
0506 $.SETC,5,17/
0507 $.JUMP,APC/
0508 $.IF,2,SKAZ*/
0509 $.SETC,5,16/
0510 $.JUMP,GPC/
0511 $.IF,2,JMP*/
0512 $.SETC,4,10/
0513 $.JUMP,BPC/
0514 $.IF,2,JMPI*/
0515 $.SETC,4,11/
0516 $.JUMP,BPC/
0517 $.IF,2,BCC*/
0518 $.SETC,5,1/
0519 $.JUMP,FPC/
0520 $.IF,2,JSR*/
0521 $.SETC,4,12/
0522 $.JUMP,BPC/
0523 $.IF,2,JSRI*/
0524 $.SETC,4,13/
0525 $.JUMP,BPC/
0526 $.IF,2,RTS*/
0527 $.SETC,4,4/
0528 $.JUMP,HPC/
0529 $.IF,2,RTI*/
0530 $.SETC,4,2/
0531 $.JUMP,HPC/
0532 $.IF,2,RIN*/
0533 $.SETC,4,10/
0534 $.JUMP,HPC/
0535 $.IF,2,ROUT*/
0536 $.SETC,4,14/
0537 $.JUMP,HPC/
0538 $.IF,2,ISZ*/
0539 $.SETC,4,36/
0540 $.JUMP,BPC/
0541 $.IF,2,DSZ*/
0542 $.SETC,4,37/
0543 $.JUMP,BPC/
```

```
0544 $.IF,2,OCT/
0545 $.SPCT,OCT/
0546  $.MOVE,3,0,20,0/
0547 $.WORD,1/
0548 $.RET/
0549 $.IF,2,JMPP~/
0550 $.SETC,4,120/
0551 $.SETC,5,400/
0552 $.ADD,3,5,3/
0553 $.JUMP,J/
0554 $.IF,2,JSRP~/
0555 $.SETC,4,6/
0556 $.SETC,5,400/
0557 $.ADD,3,5,3/
0558 $.JUMP,H/
0559 $.IF,2,JINT~/
0560 $.SETC,4,122/
0561 $.SETC,5,440/
0562 $.ADD,3,5,3/
0563 $.JUMP,J/
0564 $.IF/  **DEFAULT**
0565 $.MOVE,1,0,20,0/
0566 $.WORD,1/
0567 $.RET/
0568 $.SPCT,A/
0569 $.MOVE,6,0,4,14/
0570 $.MOVE,3,0,2,12/
0571 $.MOVE,4,0,2,10/
0572 $.MOVE,5,0,10,0/
0573 $.WORD,1/
0574 $.RET/
0575 $.SPCT,BR/
0576 $.SETC,6,0/
0577 $.SUB,6,4,4/
0578 $.SPCT,B/
0579 $.MOVE,5,0,6,12/
0580 $.MOVE,3,0,2,10/
0581 $.MOVE,4,0,10,0/
0582 $.WORD,1/
0583 $.RET/
0584 $.SPCT,C/
0585 $.MOVE,4,0,6,12/
0586 $.MOVE,3,0,2,10/
0587 $.WORD,1/
0588 $.RET/
0589 $.SPCT,D/
0590 $.MOVE,5,3,4,14/
0591 $.MOVE,5,2,1,7/
0592 $.MOVE,3,0,2,0/
0593 $.MOVE,3,0,2,12/
0594 $.MOVE,4,0,2,10/
```

```
C595 $.WORD,1/
C596 $.RET/
C597 $.SPOT,E/
C598 $.MOVE,5,1,1,13/
C599 $.MOVE,5,0,1,7/
C600 $.MOVE,3,0,3,10/
C601 $.MOVE,4,0,7,0/
C602 $.WORD,1/
C603 $.RET/
C604 $.SPOT,F/
C605 $.MOVE,5,0,4,14/
C606 $.MOVE,3,0,4,10/
C607 $.MOVE,4,0,10,0/
C608 $.WORD,1/
C609 $.RET/
C610 $.SPOT,G/
C611 $.MOVE,6,0,5,13/
C612 $.MOVE,3,0,1,12/
C613 $.MOVE,4,0,2,10/
C614 $.MOVE,5,0,10,0/
C615 $.WORD,1/
C616 $.RET/
C617 $.SPOT,H/
C618 $.MOVE,4,0,4,7/
C619 $.MOVE,3,0,7,0/
C620 $.WORD,1/
C621 $.RET/
C622 $.SPOT,J/
C623 $.MOVE,4,0,14,4/
C624 $.MOVE,3,0,4,0/
C625 $.WORD,1/
C626 $.RET/
C627 $.SPOT,K/
C628 $.MOVE,5,0,20,0/
C629 $.MOVE,3,0,2,10/
C630 $.WORD,1/
C631 $.MOVE,4,0,20,0/
C632 $.WORD,1/
C633 $.RET/
C634 $.SPOT,L/
C635 $.MOVE,3,0,4,7/
C636 $.WORD,1/
C637 $.RET/
C638 $.SPOT,APC/
C639 $.CPC,6/
C640 $.SETC,7,1/
C641 $.MOVE,5,0,4,14/
C642 $.MOVE,3,0,2,12/
C643 $.MOVE,7,0,1,10/
C644 $.ADD,6,7,7/
C645 $.SUB,4,7,4/
```

```
0646 $,MOVE,4,0,10,0/
0647 $,WORD,1/
0648 $,RET/
0649 $,SPCT,APC/
0650 $,CPC,5/
0651 $,SETC,6,1/
0652 $,MOVE,4,0,6,12/
0653 $,MOVE,6,0,2,10/
0654 $,ADD,5,6,6/
0655 $,SUB,3,6,3/
0656 $,MOVE,3,0,10,0/
0657 $,WORD,1/
0658 $,RET/
0659 $,SPCT,FPC/
0660 $,CPC,6/
0661 $,SETC,7,1/
0662 $,MOVE,5,0,4,14/
0663 $,MOVE,3,0,4,10/
0664 $,ADD,6,7,7/
0665 $,SUB,4,7,4/
0666 $,MOVE,4,0,10,0/
0667 $,WORD,1/
0668 $,RET/
0669 $,SPCT,GPC/
0670 $,CPC,6/
0671 $,SETC,7,1/
0672 $,MOVE,5,0,5,13/
0673 $,MOVE,3,0,1,12/
0674 $,MOVE,7,0,2,10/
0675 $,ADD,6,7,7/
0676 $,SUB,4,7,4/
0677 $,MOVE,4,0,10,0/
0678 $,WORD,1/
0679 $,RET/
0680 $,SPCT,HPC/
0681 $,CPC,5/
0682 $,SETC,6,1/
0683 $,MOVE,4,0,11,7/
0684 $,ADD,5,6,6/
0685 $,SUB,3,6,3/
0686 $,MOVE,3,0,7,0/
0687 $,WORD,1/
0688 $,RET/
0689 $,ASM/
```

ASSEMBLED MACHINE LANGUAGE LISTING

```
000001 000000040000  000002 000000041400  000003 000000047400  000004 000000002170  000005 000000031601
000006 000000021400  000010 000000000000  000011 000000000000  000012 000000000000  000013 000000000000
000014 000000000000  000015 000000000000  000016 000000000000  000017 000000000000  000020 000000040000
000021 000000140000  000022 000000000000  000030 000000000000  000031 000000000000  000032 000000000000
000033 000000000000  000034 000000000000  000035 000000000000  000040 000000031601  000041 000000020042
000042 000000100040  000043 000000120005  000044 000000043400  000045 000000042000  000046 000000000400
000100 000000004000  000101 000000000000  000102 000000100000  000103 000000000000  000104 000000000000
001000 000000047400  001001 000000002160  001002 000000040000  001003 000000010010  001004 000000010422
001005 000000046000  001006 000000120010  001007 000000100016  001010 000000120013  001011 000000100015
001012 000000011005  001013 000000050001  001014 000000056374  001015 000000050001  001016 000000120012
001017 000000020402  001020 000000056374  001021 000000120012  001022 000000042000  001023 000000120011
001024 000000043400  001025 000000042000  001026 000000000400  001027 000000046001  001030 000000120010
001031 000000100012  001032 000000120017  001033 000000100011  001034 000000011005  001035 000000050001
001036 000000056374  001037 000000050001  001040 000000120016  001041 000000020402  001042 000000056374
001043 000000120016  001044 000000042000  001045 000000120015  001046 000000043400  001047 000000042000
001050 000000000400  001200 000000047400  001201 000000002160  001202 000000040000  001203 000000100030
001204 000000010422  001205 000000046000  001206 000000120030  001207 000000100034  001210 000000011006
001211 000000050001  001212 000000056374  001213 000000050004  001214 000000150034  001215 000000120032
001216 000000020403  001217 000000056374  001220 000000150034  001221 000000120032  001222 000000042000
001223 000000120031  001224 000000043400  001225 000000042000  001226 000000000400  001227 000000046001
001230 000000120030  001231 000000100031  001232 000000011006  001233 000000050001  001234 000000056374
001235 000000050001  001236 X 000000150400  001237 000000120035  001240 000000020403  001241 000000056374
001242 000000150031  001243 000000120035  001244 000000042000  001245 000000120034  001246 000000043400
001247 000000042000  001250 000000000400  001500 000000100101  001501 000000056364  001502 000000011426
001503 000000100010  001504 000000010402  001505 000000100015  001506 000000020401  001507 000000100011
001510 000000106101  001511 000000121400  001512 000000046001  001513 000000031400  001514 000000100030
001515 000000010402  001516 000000100034  001517 000000020401  001520 000000100031  001521 000000121400
001522 000000046001  001523 000000031400  001524 000000100022  001525 000000121400  001526 000000046001
001527 000000031400  001530 000000126101  001531 000000043400  001532 000000042000  001533 000000000400
002000 000000100041  002001 000000120005  002002 000000047400  002003 000000100100  002004 000000126101
002005 000000046000  002006 000000120103  002007 000000004400  002010 000000100010  002011 000000010411
002012 000000100017  002013 000000056004  002014 000000050001  002015 000000140017  002016 000000140015
002017 000000140016  002020 000000140103  002021 000000120103  002022 000000020410  002023 000000100013
002024 000000056004  002025 000000050001  002026 000000140013  002027 000000140011  002030 000000140012
002031 000000140103  002032 000000120103  002033 000000011006  002034 000000050001  002035 000000056001
002036 000000011007  002037 000000100021  002040 000000120103  002041 000000020404  002042 000000056001
002043 000000011002  002044 000000100020  002045 000000120103  002046 000000100030  002047 000000010405
002050 000000046000  002051 000000140034  002052 000000140035  002053 000000120104  002054 000000020404
002055 000000046000  002056 000000140031  002057 000000140032  002060 000000120104  002061 000000100103
002062 000000015104  002063 000000120104  002064 000000011002  002065 000000050001  002066 000000006402
002067 000000047400  002070 000000003170  002071 000000120022  002072 000000020715  002073 000000106100
002074 000000036201  002075 000000056364  002076 000000011775  002077 000000101400  002100 000000056370
002101 000000104410  002102 000000025000  002103 000000101400  002104 000000060406  002105 000000104404
002106 000000025000  002107 000000046001  002110 000000031400  002111 000000020762  002112 000000077131
002113 000000000377
```

APPENDIX B

LIST OF SYMBOLS

A	Gyro rotor moment of inertia normal to spin axis
$F_1(s)$, $F_2(s)$	Direct-axis and cross-axis analog transfer functions
$F_1(z)$, $F_2(z)$	Direct-axis and cross-axis digital transfer functions
$G_1(z)$, $G_2(z)$	Mechanized direct-axis and cross-axis digital transfer functions
H(s)	General analog transfer function
H(z)	General digital transfer function
H'(s)	Dummy analog transfer function
k, K	Iteration index
K_{PO}	Pickoff gain factor
K_{SA}	Compensation gain factor
K_T	Torquer gain factor
M(k)	Total digital torque
M_x, M_y	Components of computer digital torque resulting from X- and Y- pickoff signals
M_X, M_Y	Analog rebalance torques
N	Gyro spin speed
P_X, P_Y	Precessional torques
s, S	Laplace Operator
T_X, T_Y	Total torques
U_X, U_Y	Digitized gyro pickoff angles
X_{peak}	Sampled X-axis pickoff angle
Y_{peak}	Sampled Y-axis pickoff angle

z^{-1}	Delay operator
θ_x, θ_y	Digitized gyro pickoff angles
θ_X, θ_Y	Gyro pickoff angles
τ	Iteration period
ω_X, ω_Y	Angular rate inputs
ω_i, i = 1, 2....	Analog filter break frequencies
ω'_i, i = 1, 2....	Dummy analog filter break frequencies