Report No. UMTA-CA-06-0088-81-1

AUTOMATED MIXED TRAFFIC TRANSIT VEHICLE MICROPROCESSOR CONTROLLER

R. A. Marks
P. Cassell
A. R. Johnston

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California
(JPL Publication 81-24)

FEBRUARY 10, 1981

FINAL REPORT

Prepared for

U. S. DEPARTMENT OF TRANSPORTATION

(NASA-CR-164538) AUTOMATED MIXED TRAFFIC TRANSIT VEHICLE MICROPROCESSOR CONTROLLER Final Report, Mar. 1979 - Feb. 1981 (Jet Propulsion Lab.) 112 p HC A06/MF A01 CSCL 13F G3/85
N81-27979
Unclas
26792

Office of Technology Development and Deployment
Urban Mass Transportation Administration
Washington, D. C. 20590

NOTICES

This document was prepared by the Jet Propulsion Laboratory, California Institute of Technology, and was sponsored by the U.S. Department of Transportation through an agreement with NASA.

Technical Report Documentation Page

1. Report No. UMTA-CA-06-0088-81-1	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle Automated Mixed Traffic Transit Vehicle Microprocessor Controller		5. Report Date February 10, 1981
		6. Performing Organization Code
7. Author(s) R.A. Marks, P. Cassell, A.R. Johnston		8. Performing Organization Report No. JPL Publication 81-24
9. Performing Organization Name and Address Jet Propulsion Laboratory 4800 Oak Grove Drive Pasadena, CA 91109		10. Work Unit No. (TRAIS)
		11. Contract or Grant No. DOT-AT-60008T
12. Sponsoring Agency Name and Address U.S. Dept. of Transportation Urban Mass Transportation Administration Washington, D.C. 20590		13. Type of Report and Period Covered Final Report Mar 79 - Feb 81
		14. Sponsoring Agency Code UTD-42
15. Supplementary Notes		

16. Abstract

An improved Automated Mixed Traffic Vehicle (AMTV) speed control system employing a microprocessor and transistor chopper motor current controller is described and its performance is presented in terms of velocity versus time curves. The on-board computer hardware and software systems are fully described as is the software development system. All of the programming used in this controller was implemented using FORTRAN.

This new microprocessor controller has made possible a number of new safety features and has improved the comfort associated with starting and stopping. In addition, most of the vehicle's performance characteristics can be altered by simple program parameter changes.

A failure analysis of the microprocessor controller has been generated and the results are included in the Appendices. Flow diagrams for the speed control algorithms and complete FORTRAN code listings are also included in the Appendices.

17. Key Words Automation, Transit, Microprocessor, Electric Vehicle		18. Distribution Statement Document is available to the public through the National Technical Information Service, Springfield, VA, 22161	
19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 125	22. Price

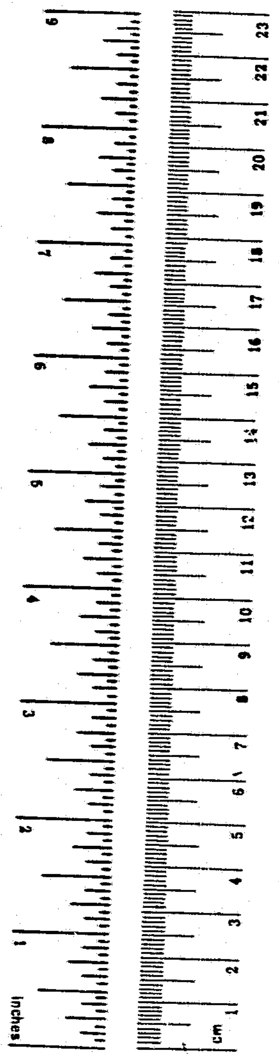
Form DOT F 1700.7 (8-72)

METRIC CONVERSION FACTORS

Approximate Conversions to Metric Measures

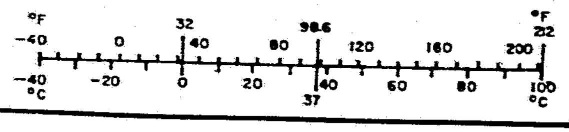
Symbol	When You Know	Multiply by	To Find	Symbol
	LENGTH			
in	inches	*2.5	centimeters	cm
ft	feet	30	centimeters	cm
yd	yards	0.9	meters	m
mi	miles	1.6	kilometers	km
	AREA			
in^2	square inches	6.5	square centimeters	cm^2
ft^2	square feet	0.09	square meters	m^2
yd^2	square yards	0.8	square meters	m^2
mi^2	square miles	2.6	square kilometers	km^2
	acres	0.4	hectares	ha
	MASS (weight)			
oz	ounces	28	grams	g
lb	pounds	0.45	kilograms	kg
	short tons (2000 lb)	0.9	tonnes	t
	VOLUME			
tsp	teaspoons	5	milliliters	ml
Tbsp	tablespoons	15	milliliters	ml
fl oz	fluid ounces	30	milliliters	ml
c	cups	0.24	liters	l
pt	pints	0.47	liters	l
qt	quarts	0.95	liters	l
gal	gallons	3.8	liters	l
ft^3	cubic feet	0.03	cubic meters	m^3
yd^3	cubic yards	0.76	cubic meters	m^3
	TEMPERATURE (exact)			
°F	Fahrenheit temperature	5/9 (after subtracting 32)	Celsius temperature	°C

*1 in = 2.54 (exactly). For other exact conversions and more detailed tables, see NBS Misc. Publ. 286, Units of Weights and Measures, Price $2.25, SD Catalog No. C13.10:286.



Approximate Conversions from Metric Measures

Symbol	When You Know	Multiply by	To Find	Symbol
	LENGTH			
mm	millimeters	0.04	inches	in
cm	centimeters	0.4	inches	in
m	meters	3.3	feet	ft
m	meters	1.1	yards	yd
km	kilometers	0.6	miles	mi
	AREA			
cm^2	square centimeters	0.16	square inches	in^2
m^2	square meters	1.2	square yards	yd^2
km^2	square kilometers	0.4	square miles	mi^2
ha	hectares (10,000 m^2)	2.5	acres	
	MASS (weight)			
g	grams	0.035	ounces	oz
kg	kilograms	2.2	pounds	lb
t	tonnes (1000 kg)	1.1	short tons	
	VOLUME			
ml	milliliters	0.03	fluid ounces	fl oz
l	liters	2.1	pints	pt
l	liters	1.06	quarts	qt
l	liters	0.26	gallons	gal
m^3	cubic meters	35	cubic feet	ft^3
m^3	cubic meters	1.3	cubic yards	yd^3
	TEMPERATURE (exact)			
°C	Celsius temperature	9/5 (then add 32)	Fahrenheit temperature	°F



ORIGINAL PAGE IS
OF POOR QUALITY

ACKNOWLEDGMENTS

The authors wish to acknowledge the assistance of Mark Nelson at the Jet Propulsion Laboratory (JPL) for his excellent job in designing and testing the digital tachometer interface circuits used in the controller. We would also like to thank Bruce Parham, also at JPL for the use of his software and hardware designs, which were used extensively in programming our non-volatile memory chips (EPROM).

We are also grateful to the many individuals who took the time to ride in the AMTV, attend reviews, and provide helpful comments and suggestions. Although this group is too large to list everyone, two in this group are owed a special debt of gratitude for their very special efforts and help. These persons are Dr. Garrett Paine (JPL), and Dr. Donald Davies (JPL).

We also thank G. Meisenholder (JPL) for his advice and assistance.

Finally, we thank R. Hoyler and D. MacKinnon of the Advanced Development Division, U.S. Department of Transportation, Urban Mass Transportation Administration (DOT/UMTA), for their continuing support and their many helpful suggestions and comments.

CONTENTS

Figures

Tables

1. SUMMARY AND CONCLUSIONS

1.1 PURPOSE, SCOPE, BRIEF SUMMARY

The long-range objective of the Automated Mixed Traffic Vehicle (AMTV) program is to develop the sensing and logic required to operate a low-speed, driverless shuttle bus or tram in mixed traffic, and to demonstrate operation of such a system in a user environment.

The objective of the task included in this report is to develop a microprocessor controller for the AMTV and evaluate its performance on the experimental JPL AMTV.

The use of a microprocessor-based speed controller has provided the AMTV with a very flexible means of controlling speed, acceleration, and jerk. In addition, a number of safety-related system checks are provided and back-up systems are used when needed.

1.2 MICROPROCESSOR CONTROLLER HARDWARE

The system chosen for this task was based on the popular 8-bit Z-80 microprocessor. The bus system chosen to support this processor was the "Standard Bus," also known as the "STD Bus."

Basically, the microprocessor controller consists of a single microprocessor card with on-board RAM and ROM memory, two specially designed and fabricated digital tachometer interface cards, one 8-port Transistor to Transistor Logic (TTL) input/output card, one card capable of 16 channels of analog-to-digital conversion at its input and 2 channels of digital-to-analog conversion at its output, and one specially designed card (named the "keep-alive" card) to monitor the health of the microprocessor system itself. All six cards were housed within a single 16 card STD Bus card-cage. The card-cage (including bus) and all cards except the two tachometer cards and the keep-alive card are standard commercial products.

1.3 MICROPROCESSOR CONTROLLER SOFTWARE

Based on a set of vehicle response criteria a set of control algorithms (described in Section 3.2) were generated. From these algorithms, a computer program was written using FORTRAN as the programming language. This program consists of a main program which calls upon 11 subroutine programs for specialized functions and calculations.

1.4 SOFTWARE DEVELOPMENT SYSTEM

All microprocessors (indeed, all digital computers) execute a series of instructions (termed a "program") each of which at its basic level is nothing more than a series of zeros and ones, i.e., binary numbers. Instructions stored in this form are termed "machine language" instructions. For applications like the AMTV controller, these instructions are stored in permanent memory modules called programmable read only memories (PROMs).

Although inexpensive instruments exist which allow one to program a PROM by hand by entering the binary numbers one-by-one, this process is very tedious, time consuming, and prone to errors. A far better solution is to allow a computer to interpret high level instructions into machine language instructions. A computer along with the necessary software to accomplish this is termed a Microprocessor Development System.

The system used for this task is based upon the same Z-80 microprocessor and STD Bus as the vehicle controller. The software development system consisted of the the following hardware: one microprocessor card, four RAM boards with 16k bytes each, one dual serial interface card (for video terminal and hard-copy printer), and one floppy disk controller card. In addition to these cards, all of the specialized input/output cards described in Section 1.2 above were used for vehicle interface testing.

This system stores and reads its files from "floppy" disk drive units. Access to the computer is accomplished using a combination video terminal and keyboard. Hardcopy output is via a dot matrix printer. Erasable Programmable Read Only Memories (EPROMs) are programmed via a specially constructed programmer connected to a standard I/O card.

The software development system uses the popular Control Program for Microprocessor (CP/M) disk operating system, FORTRAN compiler and linker, specialized programs for EPROM programming and checking, and a powerful editor/word processor.

1.5 VEHICLE PERFORMANCE

The present performance of the AMTV meets all of the generated specifications, as listed in Reference 1, pertaining to acceleration and jerk limits. Acceleration and deceleration have been subjectively described by most passengers as smooth. Speed control is accurate within approximately .12 km/h (.08 mph).

As presently configured, the vehicle features three fixed deceleration rates and one variable rate. The fixed rates are adjusted for non-obstacle-caused stops (approximately 0.09 g), obstacle caused stops (about 0.16 g), and emergency stop using hydraulic brakes only (unmeasured g level). The stop which features a variable deceleration is brought about when the vehicle detects an obstacle in its secondary sensors and is traveling at a velocity greater than 4.7 km/h (2.9 mi/h). In this case the necessary deceleration tostop within 2 m is calculated and the motor is commanded to plug at an ever increasing current until this deceleration is reached.

Accurate stops at passenger pick-up points have been attained regardless of vehicle speed prior to obtaining the command to stop. This is done by first commanding the vehicle to attain 3.2 km/h and then commanding a stop when the vehicle has passed 2.6 m beyond the magnetic street marker.

1.6 VEHICLE SAFETY

The employment of a microprocessor-based controller has afforded the opportunity to provide a number of safety features not previously used. First, tachometer (speedometer) integrity is easily tested by providing two tachometers and comparing their values. An emergency stop occurs if either or both tachometers malfunction. Second, more flexible stopping algorithms have been created which allow for safe stops in the case of an intrusion into the path of the vehicle too close for a normal stop, or if the primary sensor misses an obstacle. Third, hydraulic braking system malfunctions which result in the vehicle not remaining in a stationary position at passenger stops are detected and the motor is used as a backup system to hold the vehicle stationary. Fourth, overspeed or rolling backwards runaway conditions are detected and the vehicle is brought to an emergency stop. Fifth, failure of the microprocessor controller to properly cycle through its program is detected by a special circuit and the vehicle is brought to an emergency stop. Last, an automotive horn has been interfaced to the microprocessor and is presently programmed to honk briefly just before the vehicle begins to move.

1.7 SOLID STATE MOTOR CONTROLLER

A new motor controller has been successfully used on this vehicle to control motor current. It features a transistor chopper circuit which limits current to the motor to some maximum value, and has the advantage over the SCR type chopper controller of not needing a by-pass relay for the last increment of current. This allows for far smoother, quieter speed control due to the lack of time delay and hysteresis. Additionally, this motor controller features SCR-controlled motor field polarity change. This feature allows for extremely rapid and quiet change from forward to reverse motor torque (plugging) for both accurate speed control and stopping.

2. INTRODUCTION

2.1 PURPOSE

The purpose of this report is to document the design of a microprocessor speed controller for the AMTV vehicle and the vehicle performance which resulted.

2.2 BACKGROUND

The AMTV concept has been described in earlier papers (References 1, 2, and 3). Basically, it offers a cost-effective alternative to conventional busses or other transportation modes for many applications requiring frequent, low- speed service, since both the cost for a driver on each vehicle and the large investment required for exclusive guideway construction can be avoided.

A feasibility demonstration of an AMTV, using what was basically a breadboard vehicle was conducted at JPL during eary 1976. Results from this experiment are reported in Reference 3. The experiment used a wire-following steering system patterned after work of Fenton and Olson at Ohio State University (References 4 and 5). Similar techniques have been used by a number of investigators, and reliable speeds of 50 mph have been demonstrated (References 6 and 7). The sensors used for collision avoidance were developed at the JPL.

Although the JPL vehicle is a small electric tram, any type of vehicle could be incorporated into a future AMTV system, with the size and type of vehicle being determined by the service requirements of the specific application. A number of guideway systems employing automated vehicles have recently been put into service. Requirements of these vehicles are quite similar to those of an AMTV, except for the provision for collision-avoidance (headway) sensing, and the steering mechanization.

2.3 SCOPE

The scope of this task included developing a microprocessor controller for the AMTV and evaluating its performance on the experimental JPL AMTV. The task provides for the microprocessor controller hardware and software design for the speed control function only. The task also included replacing the SCR chopper motor controller with a new all-solid-state transistor chopper motor controller.

ORIGINAL PAGE IS
OF POOR QUALITY

2.4 ORGANIZATION OF REPORT

Section 1 provides a brief background and summary of the entire report. Section 2, the present section, more sharply defines the purpose, background, scope, and organization of the report. Section 3 offers a detailed specification of the on-vehicle microprocessor controller. Section 4 not only gives detailed specifications of the microprocessor development system, but also supplies information concerning the need for and use of the development system software. Section 5 describes the hardware and performance of the new all-solid-state motor controller. Section 6 describes the vehicle performance under microprocessor control. This is largely done using speed versus time curves obtained from the tachometers via the microprocessor during street testing. Section 7 details the safety aspects of the new microprocessor controller. Section 8 contains the findings and recommendations which have resulted from this task. Appendix A contains the speed control algorithms in flow-chart form. Appendix B contains the complete FORTRAN program listings while Appendix C contains new failure analysis material, which supplements the earlier failure mode and hazard analysis (Reference 1).

3. MICROPROCESSOR SYSTEM DESCRIPTION

3.1 ON-VEHICLE MICROPROCESSOR

The on-vehicle microprocessor consists of a card rack with built-in bus system and six plug-in cards. Additionally, there are ancillary pieces of equipment such as power supply and LED display panel drivers. These components are described below, and where noted, in more detail, in the appendices. Figure 3-1 shows the above described processor, while Figure 3-2 shows the same processor in its housing. Figure 3-3 shows a diagram of this system.

A Standard Bus (STD Bus) is used because of its size, flexibility, the variety of different types of commercially available cards, and the number of different manufacturers to choose from. This bus system is based on a 56 pin edge connector card of 11.4 cm by 16.5 cm (4.5 by 6.5 in.). It is designed for use with 8 bit microprocessors of the 6800, 8085, and Z-80 family. The card cage used on the AMTV (Prolog No. CR16) has spaces for 16 cards.

The microprocessor, Central Processing Unit, (CPU), chosen was the Zilog Z-80 because of its ability to use a more sophisticated set of software instructions. The card containing the Z-80 (Prolog Corp. No. 7803) also has sockets for 8192 bytes of programmable read only memory (PROM) or erasable, programmable, read only memory (EPROM). In addition, this card has sockets for 4096 bytes of random access memory. The card has its own crystal-controlled 400 ns clock and provision for pushbutton reset. All integrated circuits are socketed, and therefore easily replaced.

As installed with the current software this card is populated with three EPROMs (type 2716, 2048 bytes each) and 8 RAM chips (type 2114, 4096 bytes) of which only about 300 bytes are used.

To interface the central processing card to TTL compatible input-output (I/O) signals on the vehicle, a (Prolog No. 7604) universal TTL I/O card is used. This card has eight 8-bit ports, any number of which can function as either input or output ports, i.e., 7 in and 1 out or 4 in and 4 out, etc. We have modified this card for noise suppression by placing .01 microfarad capacitors across each input line. The present configuration of this card uses the first four ports (0 through 3) for input signals and the last four ports (4 through 7) for output signals. Each port is directly software addressable. Although several output signals from this card directly control vehicle function through relays or valves, others are used strictly to light LED indicators mounted on a panel above the computer. For the later purpose LED, driver cards were specially designed and built since the

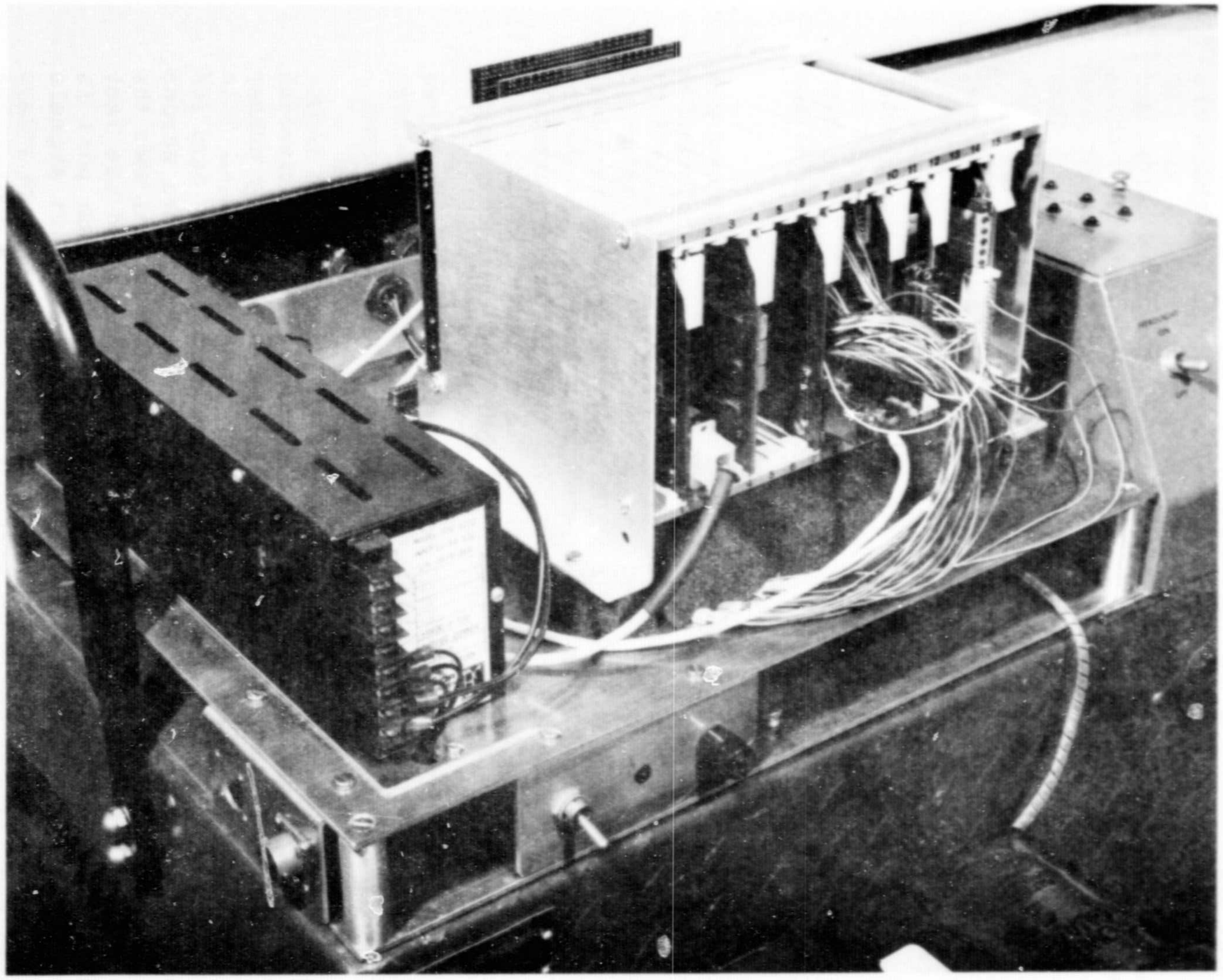


FIGURE 3-1. MICROPROCESSOR CONTROLLER SYSTEM. Dark rectangular object on left is voltage inverter. To the right of this is the standard bus card-cage complete with all controller cards. Standing upright behind standard bus card cage are LED driver cards. The top of the case housing the controller battery is visible in lower right portion of photograph.

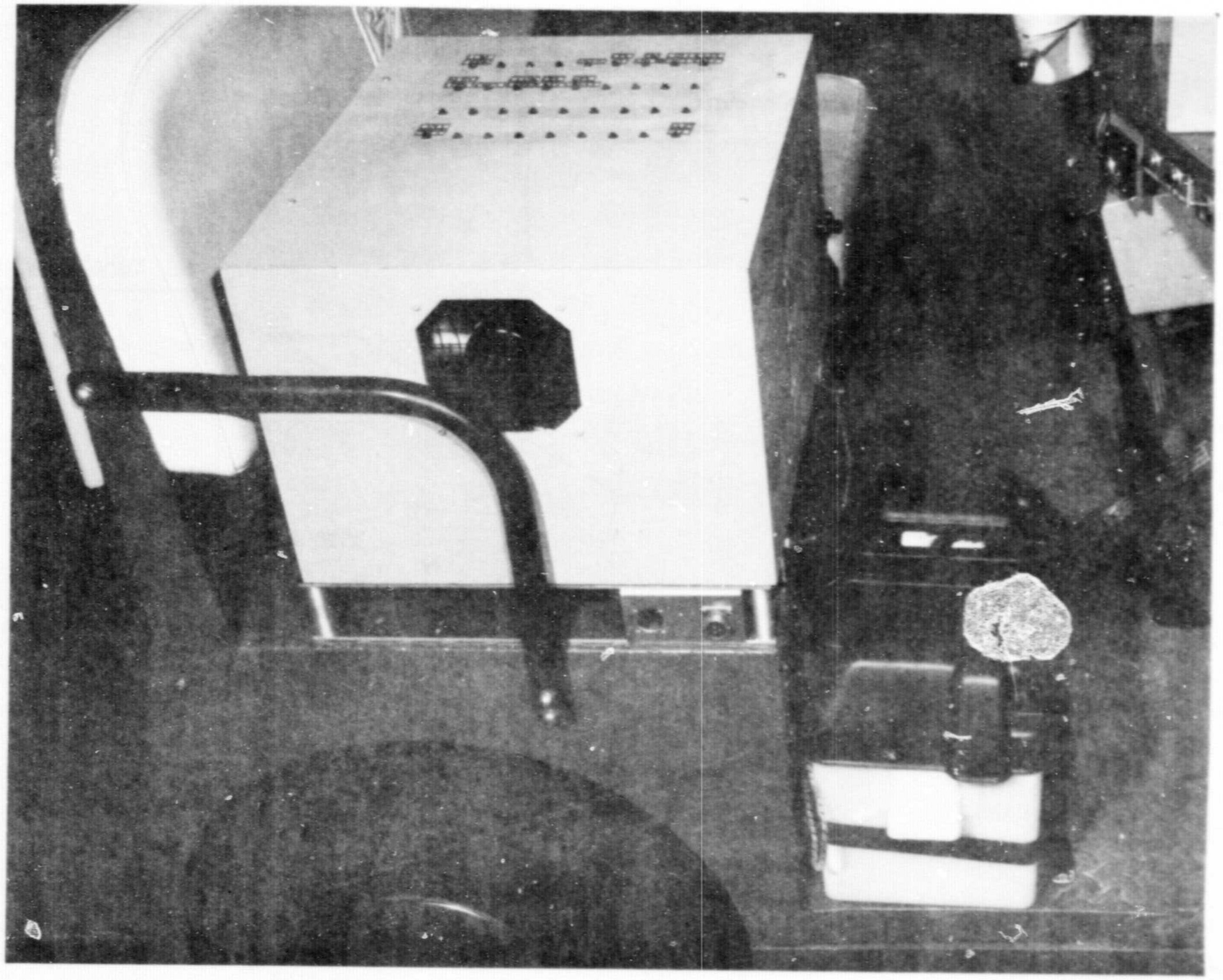


FIGURE 3-2. MICROPROCESSOR CONTROLLER HOUSING AND BATTERY. The LED display panel which forms the top of the housing is used to indicate state of TTL inputs as seen by the controller. Note "muffin" fan in side of housing used to cool controller circuits. Controller 12 V battery is located on floor boards in front of controller.

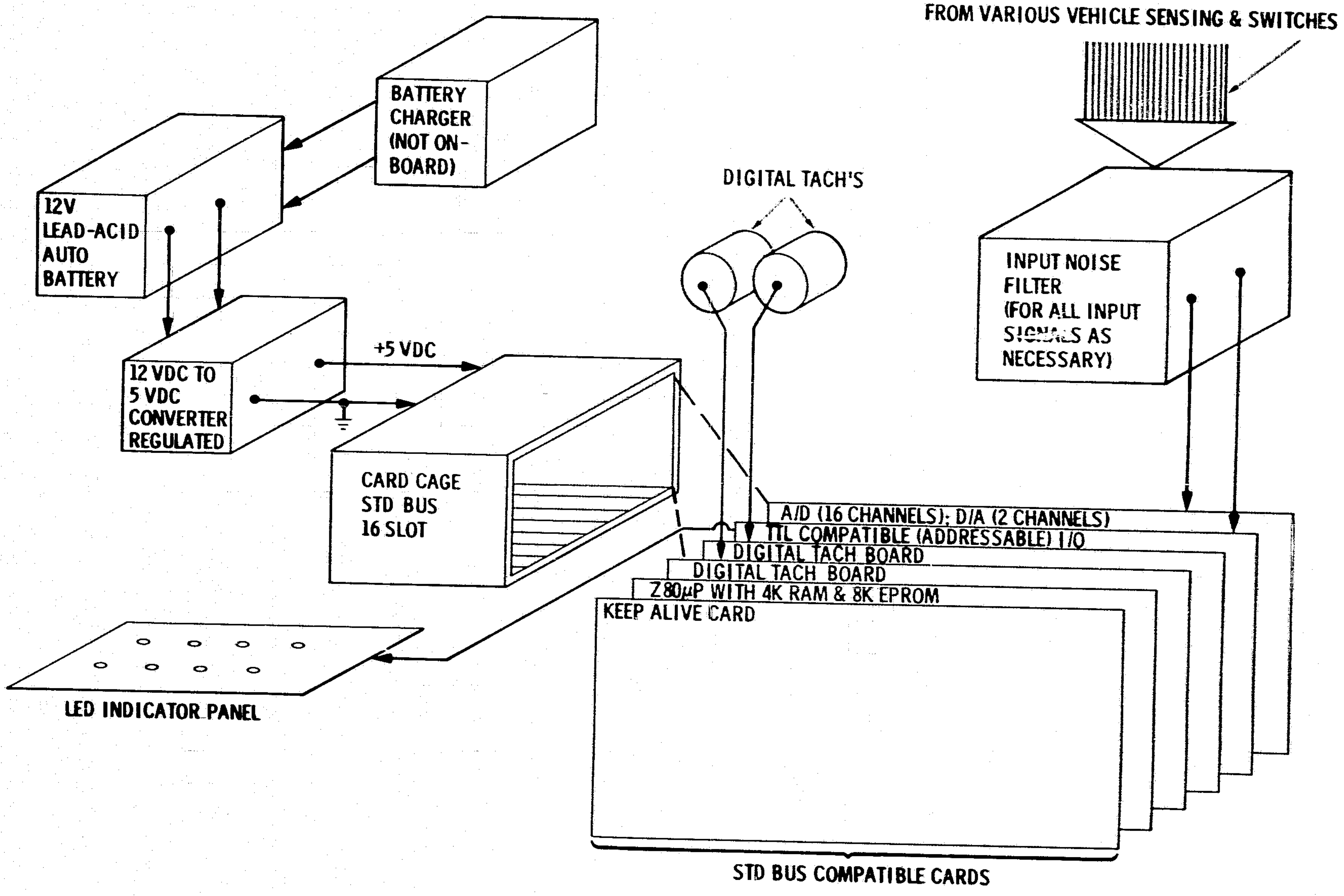

FROM VARIOUS VEHICLE SENSING & SWITCHES
BATTERY CHARGER (NOT ON-BOARD)
12V LEAD-ACID AUTO BATTERY
DIGITAL TACH'S
INPUT NOISE FILTER (FOR ALL INPUT SIGNALS AS NECESSARY)
+5 VDC
12 VDC TO 5 VDC CONVERTER REGULATED
CARD CAGE STD BUS 16 SLOT
A/D (16 CHANNELS); D/A (2 CHANNELS)
TTL COMPATIBLE (ADDRESSABLE) I/O
DIGITAL TACH BOARD
DIGITAL TACH BOARD
Z80µP WITH 4K RAM & 8K EPROM
KEEP ALIVE CARD
LED INDICATOR PANEL
STD BUS COMPATIBLE CARDS

FIGURE 3-3. AMTV ON-BOARD MICROPROCESSOR CONTROLLER SYSTEM

Prolog 7604 does not have the current output capacity to drive a LED. TTL I/O cards are now on the market which are able to drive LEDs directly. For reliability and compactness this will be the card of choice for future units.

To input or output analog signals to the CPU, an analog I/O card (Analog Devices No. RTI-1225) is used. This card has the capacity to input up to 16 analog input signals using its multiplexer through a 10 bit analog-to-digital converter (A/D). The RTI-1225 can also convert an 8 bit integer to an analog voltage available at its output terminals. There are two such channels for digital-to-analog (D/A) conversion (DAC). This card uses memory mapping instead of direct addressing. At the present time no A/D channels are being used. One D/A channel is used to output an analog motor control voltage to control motor current. It is anticipated that the A/D converter on this card may be used for future monitoring of motor current and other analog sensor outputs.

Two other cards (identical) within the microprocessor controller not only interface the two digital tachometers to the STD Bus but actually provide the basic timing cycle for initiating repetitive execution of the software. These cards were designed and built at the JPL and are referred to as tach cards. Each card works independently of the other and contains two 16 bit accumulators and one output register. Each 16 bit accumulator counts for 20 ms then holds its value for 20 ms. This is an alternating process; one accumulator holds while the other counts up for 20 ms, then the registers reverse roles. At the end of each 20 ms period the accumulator which has just finished counting has its data transferred to the output register. In this manner no data is lost while a register is being read. These cards each read the clock line on the STD Bus (2.54 mhz) and divide it down in order to create an accurate 20 ms flag. It is this flag which triggers the program to cycle through again. Counter-timer cards are now commercially available which may supply the same function as our specially designed cards. Because these new cards use printed circuits instead of the wire-wrap type circuits on our cards they are more reliable, economical, and far more compact. We hope to adapt them to our purpose in future controllers.

Another card, our "keep-alive" circuit, uses the standard bus as a mounting rack and power supply but uses no other bus lines. This circuit reads a line carrying a short pulse with a 20 ms period which is generated by the computer program and output from the TTL I/O card. This circuit card contains logic and a relay to bring the AMTV to an emergency stop should the keep-alive pulse disappear for more than 4 cycles or stay permanently on for 4 cycles. This disappearance would result from a central processor failure, a tach. card failure, or even a previously undetected software error which would send the program counter to an improper part of program memory.

To control an automotive type electric (12V) horn, a special interface card was designed which uses an optical coupler and amplifier circuit to energize a horn relay which in turn energizes the horn. This card is not presently mounted in the STD Bus rack.

The power supply for the STD bus (and all of the cards therein) consists of a 12 V to +5 V chopper type converter (Bikor model 1202) fed from a 12 V lead acid battery. The converter draws 4.2 A from the battery to supply the STD Bus. The cooling fan described below draws an additional .4 A from the battery. At this energy consumption rate the battery could be expected to supply energy for up to 20 hours.

All of the above components except for the battery are housed in an aluminum based and roofed, plywood walled enclosure. Ventillation of this enclosure is forced by a 12 V "muffin-type" fan .

3.2 VEHICLE RESPONSE CRITERIA

Before control algorithms can be constructed one must have a table of desired vehicle responses based on various inputs. The table used for this task is shown in Table 3-1.

3.3 ON-VEHICLE MICROPROCESSOR: CONTROL ALGORITHMS

We will define an algorithm as a method used to calculate an answer or product given a set of inputs. For the AMTV the control algorithm is broken down into one main algorithm and 11 specialized sub-algorithms. Each of the sub-algorithms has a specialized control function, and as each is needed it is called upon by the main algorithm to perform its function. This scheme makes it easier to understand the overall control and provides a logical and convenient framework for software development. Figure 3-4 shows the flow or block diagram for the main algorithm. A similar diagram is shown in Appendix A for each of the sub-algorithms. In all of these diagrams the somewhat elliptically-shaped outlines signify entrance and exit points, the rectangular outlines indicate a computational function, and the diamond shaped outlines are rhetorical. To fully understand the logic of these algorithms would require some understanding of the characteristics of all of the vehicle's sensors and its motor controller. Section 5 of this report describes the motor controller characteristics while References 1 and 2 describe the vehicle's sensor characteristics. Additionally, the FORTRAN software listings contained in Appendix B contain many comment statements which explain the logic of the control scheme used.

TABLE 3-1. VEHICLE PERFORMANCE DESIGN

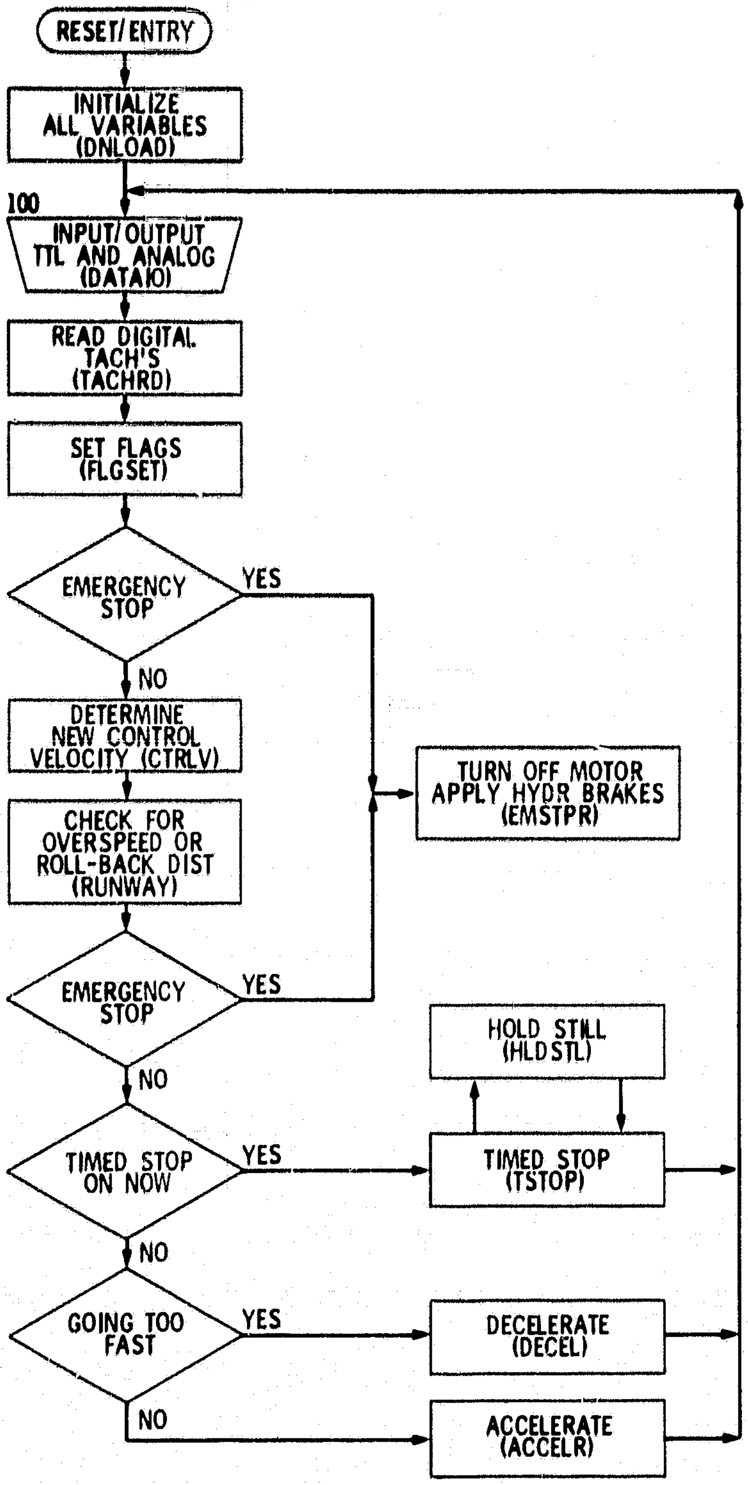
Input-Calculated or Sensed	Response Type	Desired Vehicle Response
Reset button pushed	Initialization	Initialize all variables; Begin 2-stage acceleration; First command speed = 3.2 km/hr; after this speed has been maintained for 3 s, command speed increases to 11 km/hr; Reset button may be pushed while in motion without abrupt changes in motion
Emergency stop push button	Emergency stop	Open motor controller main relay, apply hydraulic brakes full on; Mush push "RESET" button to resume automatic operation
Whisker pole switch closes	Energency stop	Same as above
Vehicle rolls back > .5 M	Emergency stop	Same as above
Vehicle speed > 11.5 km/hr (overspeed condition)	Emergency stop	Same as above
Tachometers don't agree	Emergency stop	Same as above
Both tachometers read 0 and motor controller > 5 V	Emergency stop	Same as above
Secondary sensor on and speed > 4.5 km/hr	Fast stop	Calculate deceleration to bring vehicle to stop in less than 2 m, use plugging to attain this deceleration rate; Disregard "G" level of stop! Resume control speed if sensor signal clears; Toot horn before resuming vehicle has come to stop

TABLE 3-1. VEHICLE PERFORMANCE DESIGN (Cont'd)

Input-Calculated or Sensed	Response Type	Desired Vehicle Response
Secondary sensor on and speed 4.5 km/hr	Normal stop	Make normal stop (0.16 G maximum); Resume control speed if sensor clears; Toot horn before resuming after stop
Loss of guide wire indication	Normal stop	Make normal stop (0.16 G maximum); Do not resume unless condition clears
Boarding tape switch closed	Slow stop	Make slow stop (0.90 G maximum); Remain stopped for 4 s or until 4 s after tape switch opens; Toot horn before resuming
Road stop (magnet) sensed	Slow, measured stop	Begin distance count at magnet; Decelerate at .12 G (or accelerate) to 3.2 km/hr; Initiate stop when vehicle has traveled 2.5 m beyond road magnet
Primary sensor on	Slow down	Slow to control speed of 3.2 km/hr; Maintain while sensor is on; Resume control speed if sensor clears
Steering angle indicator on	Slow down	Same as above except delay speed resumption for one second after steering angle signal clears

3.4 ON-VEHICLE MICROPROCESSOR - - SOFTWARE

The actual on-vehicle software consists of Z-80 machine language recorded on EPROMs. This code was generated from FORTRAN source code by use of a FORTRAN compiler and linker resident in the development system described in Section 4. Although the program itself resides on EPROM chips the memory variables are placed into RAM and modified there by the program.



RESET/ENTRY
INITIALIZE ALL VARIABLES (DNLOAD)
100
INPUT/OUTPUT TTL AND ANALOG (DATAIO)
READ DIGITAL TACH'S (TACHRD)
SET FLAGS (FLGSET)
EMERGENCY STOP
YES
NO
DETERMINE NEW CONTROL VELOCITY (CTRLV)
TURN OFF MOTOR APPLY HYDR BRAKES (EMSTPR)
CHECK FOR OVERSPEED OR ROLL-BACK DIST (RUNWAY)
EMERGENCY STOP
YES
NO
HOLD STILL (HLDSTL)
TIMED STOP ON NOW
YES
TIMED STOP (TSTOP)
NO
GOING TOO FAST
YES
DECELERATE (DECEL)
NO
ACCELERATE (ACCELR)

FIGURE 3-4. AMTV μP CONTROLLER PROGRAM MAIN PROGRAM

ORIGINAL PAGE IS OF POOR QUALITY

4. SOFTWARE DEVELOPMENT SYSTEM

4.1 HARDWARE DESCRIPTION

The on-vehicle computer programs exist on EPROMs in machine language form. Although relatively inexpensive instruments exist which allow one to program an EPROM by hand, entering the binary numbers one-by-one, this process is very tedious, time consuming, and prone to errors. A far better solution is to allow a computer to interpret high level instructions into machine language instructions, then using appropriate software and hardware automatically load these instructions onto an EPROM. An appropriate computer, along with the necessary software to accomplish this, is termed a Microprocessor Development System.

The development system used for this task is based upon the same Z-80 microprocessor and STD Bus used for the on-vehicle controller. This system is shown diagrammatically in Figure 4-1 and a photograph of the system is shown in Figure 4-2. The development system consists of the following hardware: one Z-80 based microprocessor card (Prolog #7803), four RAM boards with approximately 16k bytes of static RAM each (each board is a Prolog #7701), one dual serial interface card (Micro/Sys #SB8420) for communications between microprocessor and video terminal and hard-copy printer, and one floppy disk controller card (Micro/Sys #SB8500) for communication between the microprocessor and the floppy disk drives. In addition to these cards, for vehicle interface purposes during our initial testing phase, all of the specialized input/output cards described in Section 1.2 above were used.

This system stores and reads its files from three floppy disk drive units (Shugart #800-2). These drives use the popular 8 in. diameter, flexible plastic, iron oxide coated discs. These discs are produced by several manufacturers and for our system are IBM 3740 compatible, single-density, single-sided, and soft sectored. Each disc has a storage capacity of 256 k bytes.

The development system described above, along with a good deal of software described in Section 4.2, was supplied as a "turn-key" package by MICRO/SYS in La Canada, California.

Programmer access to the computer is accomplished using a combination video terminal and keyboard (Soroc Model IQ 120) operated at a rate of 19.2 k BAUD (or about 1920 characters per second maximum). Hardcopy output is via dot matrix printer (Teletype Model 43) operated at a rate of 300 BAUD (or about 30 characters per second). Erasable, programmable, Read Only Memories (EPROM's) are programmed via a specially constructed programmer circuit connected to a standard (Prolog #7604) I/O card.

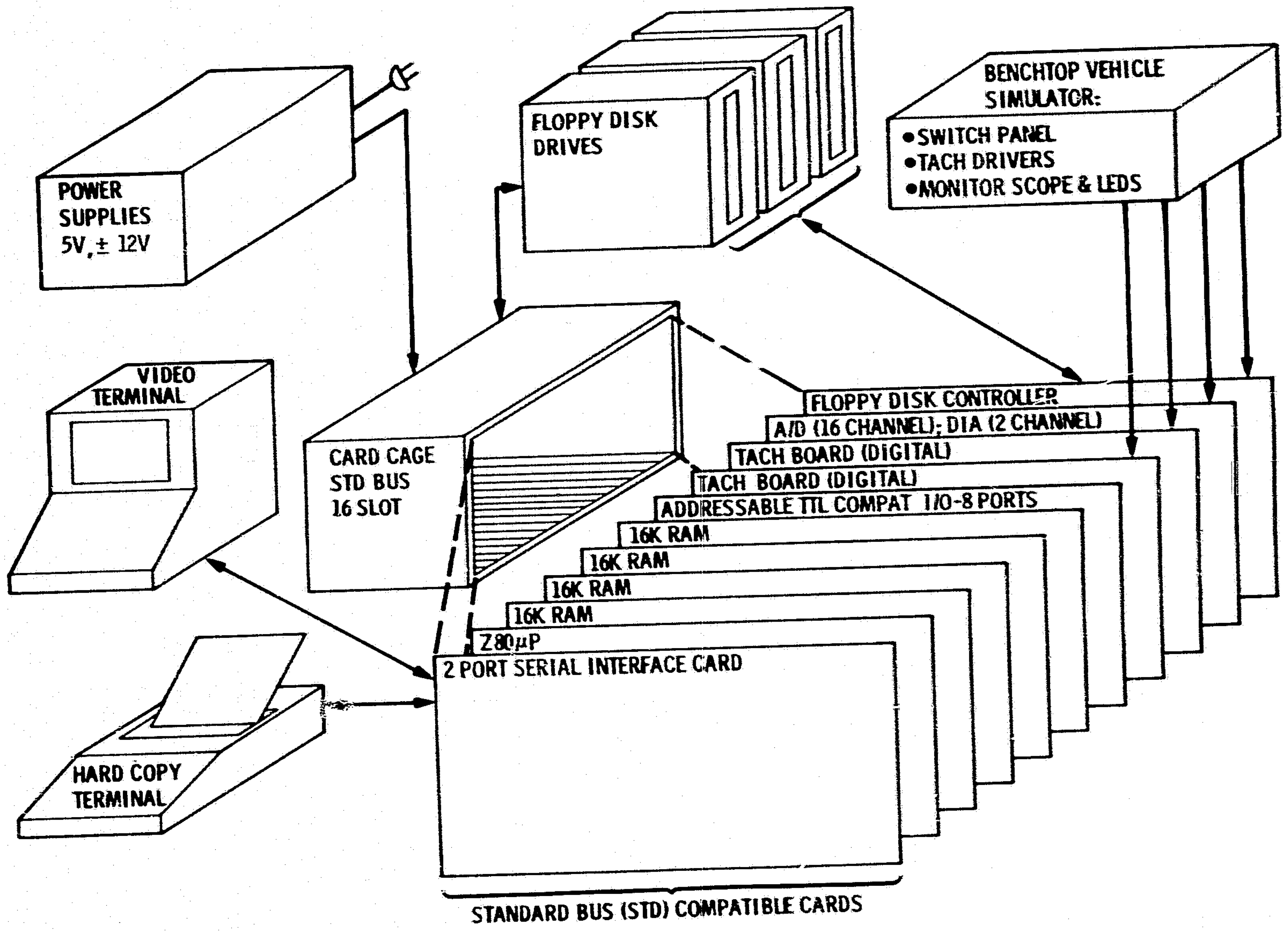


FIGURE 4-1. AMTV MICROPROCESSOR SOFTWARE DEVELOPMENT SYSTEM

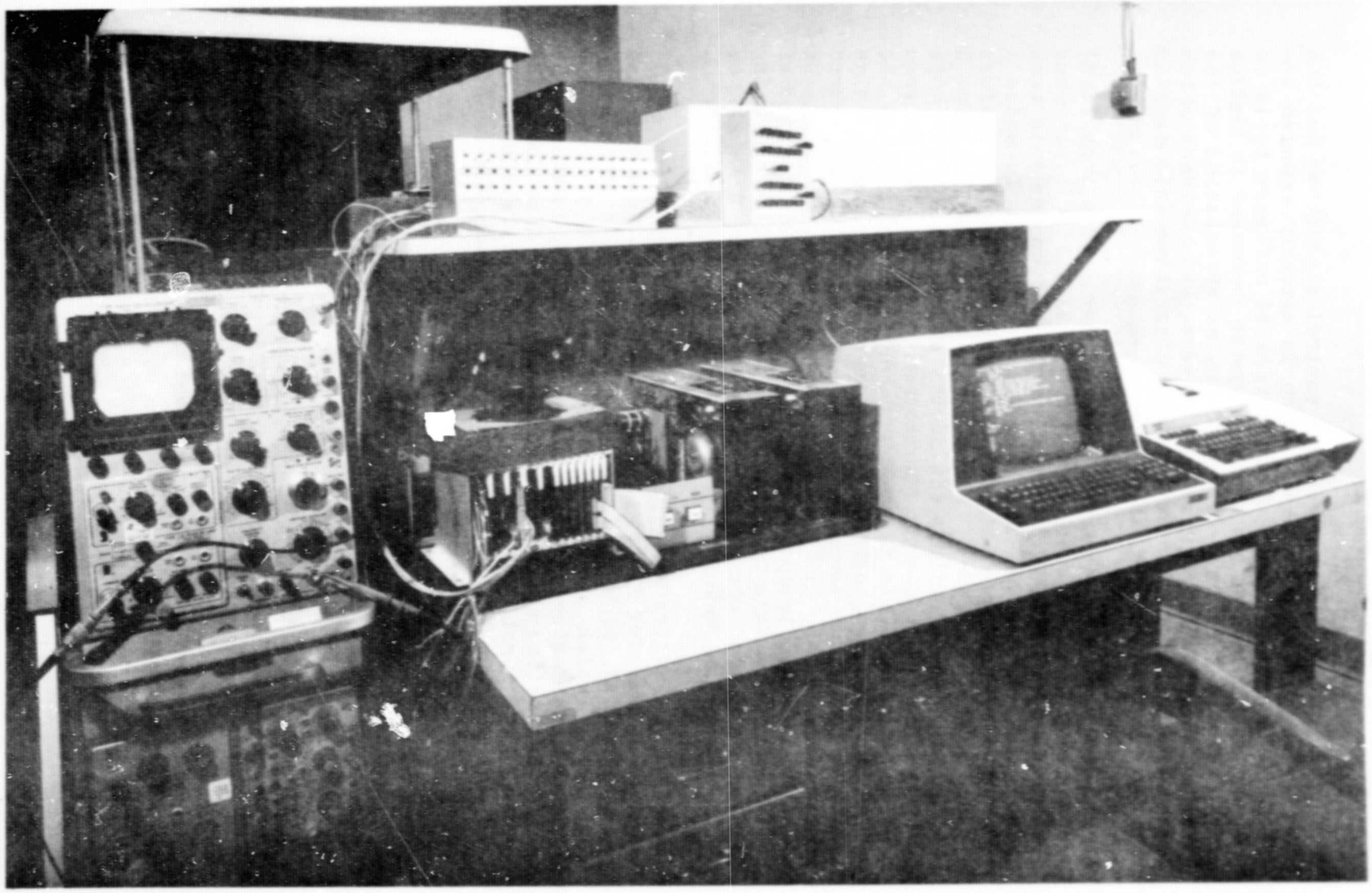


FIGURE 4-2. MICROPROCESSOR DEVELOPMENT SYSTEM. From left to right on work bench are: Micro/sys NDS development system complete with standard bus card cage and cards, power supplies, and two floppy disk drives. (See text for a more detailed description); Soroc video terminal; Teletype model 43 terminal. Box on top left of bench contains switch and LED panels for vehicle simulation during early program development. Terminal box on right side of bench top is connected to standard bus card used for analog-to-digital and digital-to-analog transformations. Oscilloscope on left used to monitor various input/output signals.

4.2 SOFTWARE DEVELOPMENT TOOLS

To bridge the gap between vehicle control algorithms and the machine language binary program which is actually executed on-board the AMTV, a number of sophisticated software tools or programs must reside on the hardware system described in Section 4.1. Figure 4-3 shows a diagrammatic representation of the interaction between programmer, software, and hardware which allows the transformation of control algorithms into machine language programs residing in EPROMS.

First, the programmer must be knowledgeable in a high level programming language such as FORTRAN, BASIC, PASCAL, or some other language. FORTRAN was the language chosen for this task based mainly on its availability and popularity. To write a program or communicate with any of the software which resides either in the computer's disc storage or its RAM storage a program must reside within the memory at all times which acts much the same as a competent housekeeper. That is, this program monitors at all times to the operator's commands which are entered via keyboard, correctly interprets these commands, then acts upon them. This housekeeper knows where to find all stored items and where to store newly created items and keeps an elaborate, carefully updated record of size and/or location of all stored items. This housekeeper is termed a "DISC Operating System." The disc operating system used is the CP/M system written by Digital Research located in Pacific Grove, California.

Several other utility programs are also employed: PIP to enable the user to transfer files, programs, data, etc., from one disc to another; STAT, to tell the user how much space is used on each disc and by which programs; DIR, to list all of the file (program) names stored on any disc; and, SYSGEN, to enable the user to initialize a new blank disc with a new disc operating system on its first two tracks. All of these utility programs were also purchased from Digital Research via Micro/Sys as part of the CP/M disk operating system package.

A number of other small but useful utility programs were supplied by Micro/Sys. These included programs which would: erase all contents on a disc, check a disc for errors, check the computer's RAM memory, list disc contents, etc.

To write and edit the FORTRAN source code a text editor or word processor must be used. The software chosen for this purpose was "Wordstar" written by Micropro International Corporation located in San Rafael, California. This is an outstanding example of "canned software" which lent a great deal of efficiency to program development.

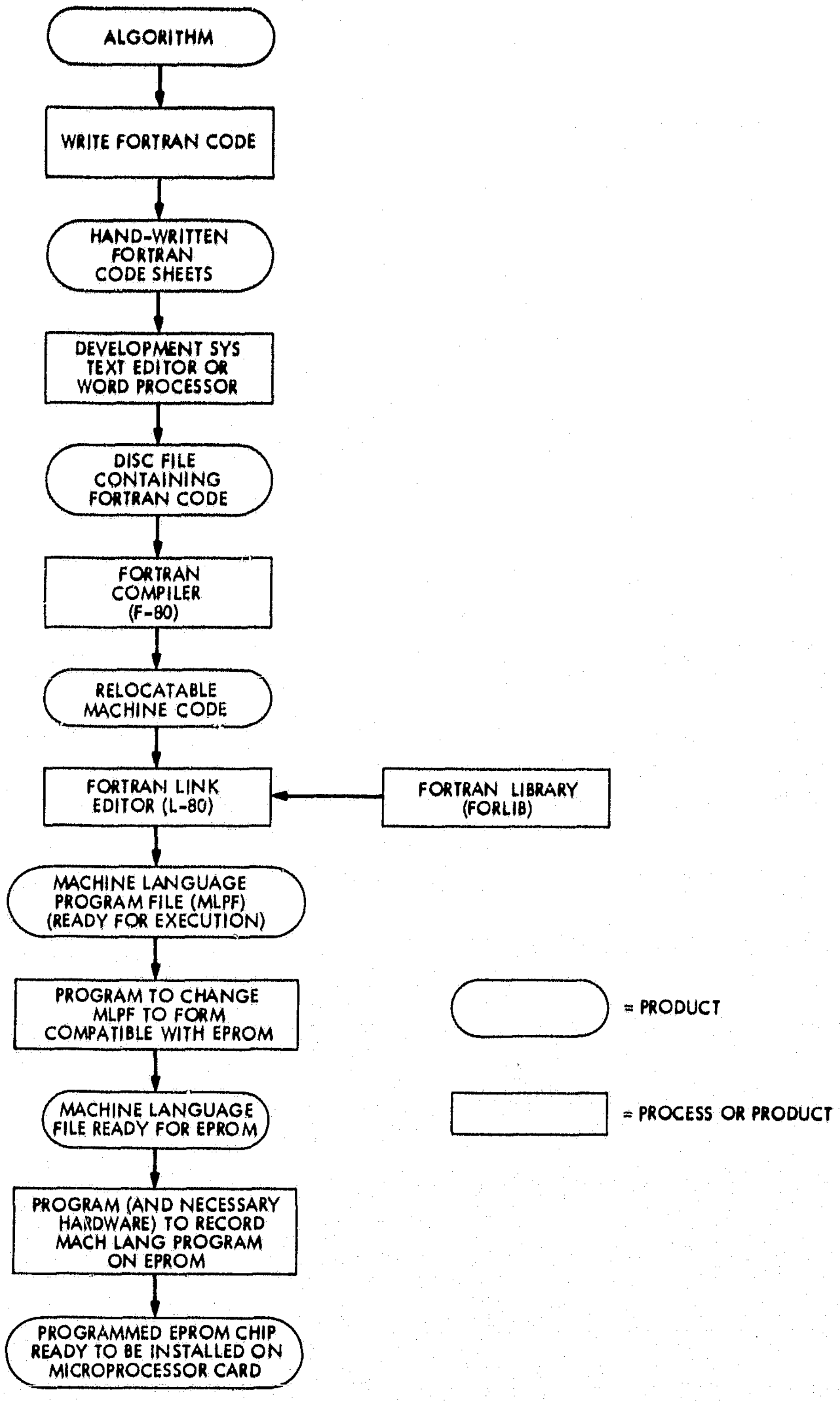

ALGORITHM
WRITE FORTRAN CODE
HAND-WRITTEN FORTRAN CODE SHEETS
DEVELOPMENT SYS TEXT EDITOR OR WORD PROCESSOR
DISC FILE CONTAINING FORTRAN CODE
FORTRAN COMPILER (F-80)
RELOCATABLE MACHINE CODE
FORTRAN LINK EDITOR (L-80)
FORTRAN LIBRARY (FORLIB)
MACHINE LANGUAGE PROGRAM FILE (MLPF) (READY FOR EXECUTION)
PROGRAM TO CHANGE MLPF TO FORM COMPATIBLE WITH EPROM
MACHINE LANGUAGE FILE READY FOR EPROM
PROGRAM (AND NECESSARY HARDWARE) TO RECORD MACH LANG PROGRAM ON EPROM
PROGRAMMED EPROM CHIP READY TO BE INSTALLED ON MICROPROCESSOR CARD
= PRODUCT
= PROCESS OR PRODUCT

FIGURE 4-3. STEPS FOR GENERATING EPROM. Diagram showing steps and intermediate products formed to produce a programmed EPROM starting with control algorithms.

Once the FORTRAN code is written (and stored) on-disc, another program must be used to transform it to machine language code (more accurately termed relocatable machine language code). The software used is the F80 FORTRAN compiler written by Microsoft of Bellevue, Washington. All FORTRAN mathematical symbols (/,*,+,-,[,],**,etc.) and functions (sin, cos, log, etc.) use already created subroutines stored in relocatable machine language in a FORTRAN subroutine library (FORLIB). To combine these subroutines with the relocatable machine code output by the (F-80) compiler another program termed the L-80 linking loader is used. Both FORLIB and L-80 are also supplied by Microsoft.

After "linking," the resulting machine language program can be loaded into RAM memory and executed, however, it is not quite in the correct form for use on an EPROM. Another program termed "UNLOAD" makes this final transformation. UNLOAD was supplied by another user at JPL.

Once a program is in the correct form for transfer to an EPROM, another program must exist which reads the program from a disc, and applies the correct and corresponding voltages to an EPROM to permanently record the bits on the EPROM at the correct address. This program (termed PROM) was also supplied at the JPL.

A typical sequence for transforming an algorithm into machine code on an EPROM follows:

1. Use CP/M to verify that disc has enough space left for programs. (Time required: about 5 sec.)

2. Use CP/M to call in WORDSTAR. Enter FORTRAN code lines. Store completed FORTRAN source code on disc. Print a copy of program for checking and debugging purposes. (This step may take from minutes to hours depending upon how much code needs to be entered and the typing speed and accuracy of the user).

3. Use CP/M to call in the F-80 compiler. Supply the compiler the file name for the source program created in the previous step. Compiler outputs list of syntax errors for user to correct. (This step requires from 15 to 30 seconds).

4. Use CP/M to recall WORDSTAR. Make corrections to FORTRAN source code to correct any errors found in compilation in Step 3. Now store new corrected version of FORTRAN SOURCE code. Repeat Steps 3 and 4 until no further syntax errors are found by the compiler. (This step usually takes only about 2 minutes unless the syntax error is very subtle).

5. Use CP/M to call in the L-80 Linking Loader. Supply the "linker" with the name of the compiled file. The product is a single program in machine language which incorporates all of the FORTRAN source programs converted to machine language, and all of the necessary FORTRAN library routines also in machine language.

   Note: In the case of the AMTV controller programs this was one main program and 11 subroutine programs. The linker put together all of these along with all of the necessary FORLIB subroutines. (For the controller programs this process takes about 2 minutes).

6. Use CP/M to call in UNLOAD. Supply the name of the file which is to be transformed into EPROM format. (This process takes less than 30 seconds).

7. Use CP/M to call in PROM. Mount erased EPROM in socket and use PROM to verify that EPROM is erased. Use PROM to transfer program to EPROM. This program actually checks its work by rereading the EPROM and checking that what was recorded was correct. (This process takes about 2 minutes per EPROM. There are three EPROMS, each with a capacity of 2048 bytes, used in the AMTV controller at this time).

8. Plug EPROM(s) into Z-80 CPU card. Then plug this card into the STD Bus on board the vehicle and check for correct operation with the vehicle on jacks. (This process takes about 3 minutes). If operation is not satisfactory analyze problem and begin back at Step 2.

5. SOLID-STATE MOTOR CONTROLLER

5.1 HARDWARE DESCRIPTION

As a part of this task a new transistor chopper motor current controller was purchased from EVC Inc. in Inglewood, California. This motor controller consists of two separate units (or circuits) mounted within the same case. One unit is a transistor chopper motor current controller, model EVC-400-36-12H, which is a standard EVC product. The other unit within the case consists of a solid-state circuit, model EVC-SSW400, which is used to reverse the field winding polarity of the motor for plugging (motor torque reversal). This later unit consists mainly of four large silicon controlled rectifiers (SCRs) and was specially designed and built at the request of the JPL. The total package is EVC model number PMC-100-36.

The chopper uses high current switching transistors to smoothly and efficiently control the speed of the motor. Both portions of this motor controller are rated for 400 A. Some of the features of this controller are:

(1) Controller turns off if battery is too low.

(2) Output current (to motor) cuts back if controller is overheated.

(3) Output stops if motor is shorted.

(4) Controls maximum battery current (current limiting).

(5) Full current output can be obtained without using bypass relay.

5.2 PERFORMANCE

Upon delivery this motor controller did not perform within specifications. However, close consultation and cooperation between the JPL and EVC resulted in a motor controller which operates smoothly over its entire range of motor currents. This controller uses only two electromechanical relays: one to turn on and off the main power and another (magnetic reed relay) which reacts to plugging current. The reed relay is used to reduce the motor control analog command voltage magnitude during plugging.

The SCR field reversing circuit allows transition between forward and plugging modes in less than 20 ms. This is far faster and quieter than the old relay type reversing circuit. The advantage is in eliminating dead time when using relatively tight feedback control. Similar advantages are obtained because the bypass relay (used during high current operation) is eliminated.

This motor controller has no provision for regenerative braking, however, both EVC and General Electric are known to be developing motor controllers which will incorporate this feature.

No curves or hard data were taken on the new EVC controller after delivery, but curves showing overall vehicle performance using this controller are contained in Section 6.

6. VEHICLE PERFORMANCE UNDER MICROPROCESSOR CONTROL

6.1 TESTING METHODS

The performance of the vehicle's sensors has been documented in Reference 2. With this in mind it was felt that stopping distances and Velocity versus time curves recorded during acceleration and deceleration testing on the road would be sufficient to characterize the performance of the microprocessor controlled AMTV.

Stopping distances were measured between obstacle and vehicle front using a standard grey target, a black target (3M Company Nextel Velvet Coating 101-C10 black), 1% reflectivity, and project personnel.

The speed-versus-time figures in the remainder of this section were generated in the following manner: the microprocessor was programmed (without interfering with the normal control program) to output two analog voltages: one proportional to speed and the other corresponding to various sensor (e.g., primary, secondary, etc.) and switch (e.g., passenger tape switches, bumper switch, etc.) flags via separate digital-to-analog converters (DACs). These voltages were monitored in the road using a battery powered dual-trace storage-type oscilloscope. Next, these oscilloscope images were photographed using a Polaroid oscilloscope camera. These Polaroid images were enlarged, then traced onto mylar plates. Because of the graphical steps involved and the inherent accuracy limits of an oscilloscope, accuracy is estimated to be within ± 5%.

6.2 DECELERATION AND STOPPING PERFORMANCE

Figures 6-1 through 6-7 show speed-versus-time for various initiated stops. Arrows pointing up the abscissa indicate the time at which a sensor or switch was first recognized by the computer. These arrows are coded to indicate which sensor or switch they represent. The numeral "1" represents a primary sensor, the numeral "2" represents a secondary sensor, the letter "R" indicates a <u>road</u> magnet was "read," and the letter "T" indicates that a passenger <u>tape</u> switch was squeezed. Arrows pointing down in Figure 6-4 indicate the time at which the indicated flag was cleared.

These tests were all made during downhill runs on the same day within about 1 h. Exact conditions and run descriptions are given in each figure.

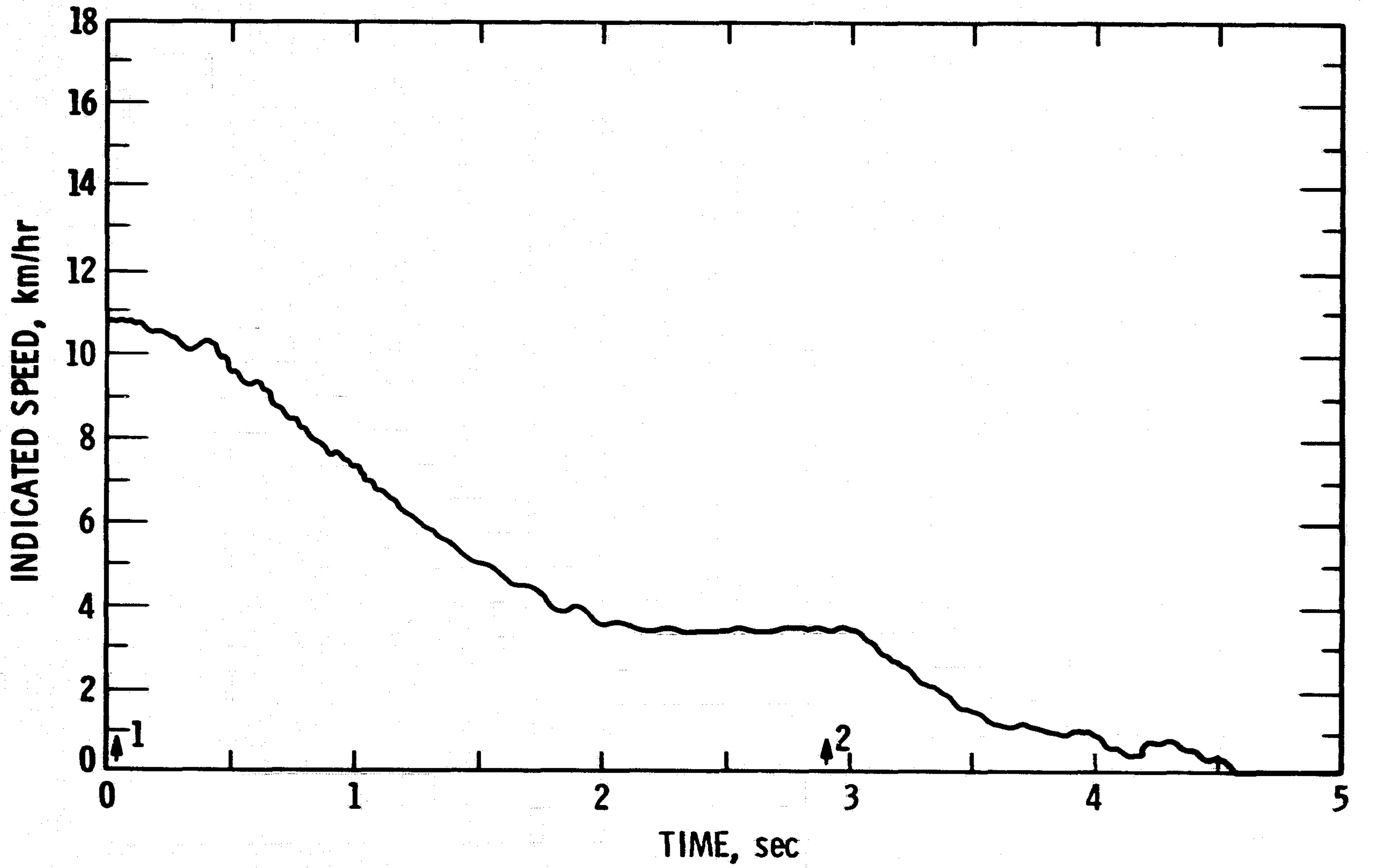


FIGURE 6-1. DECELERATION CURVE. Sensor-induced stop. Note vehicle deceleration to 3.2 km/hr while primary sensor on, and from 3.2 km/hr to rest with secondary sensor on. Obstacle is adult pedestrian. Vehicle controlling its speed using plugging at initiation of stop, was travelling down approximately 4% grade. Pedestrian is adult wearing clothing of medium reflectivity, however, similar results are obtained using our "standard" 20 X 100 cm gray target.

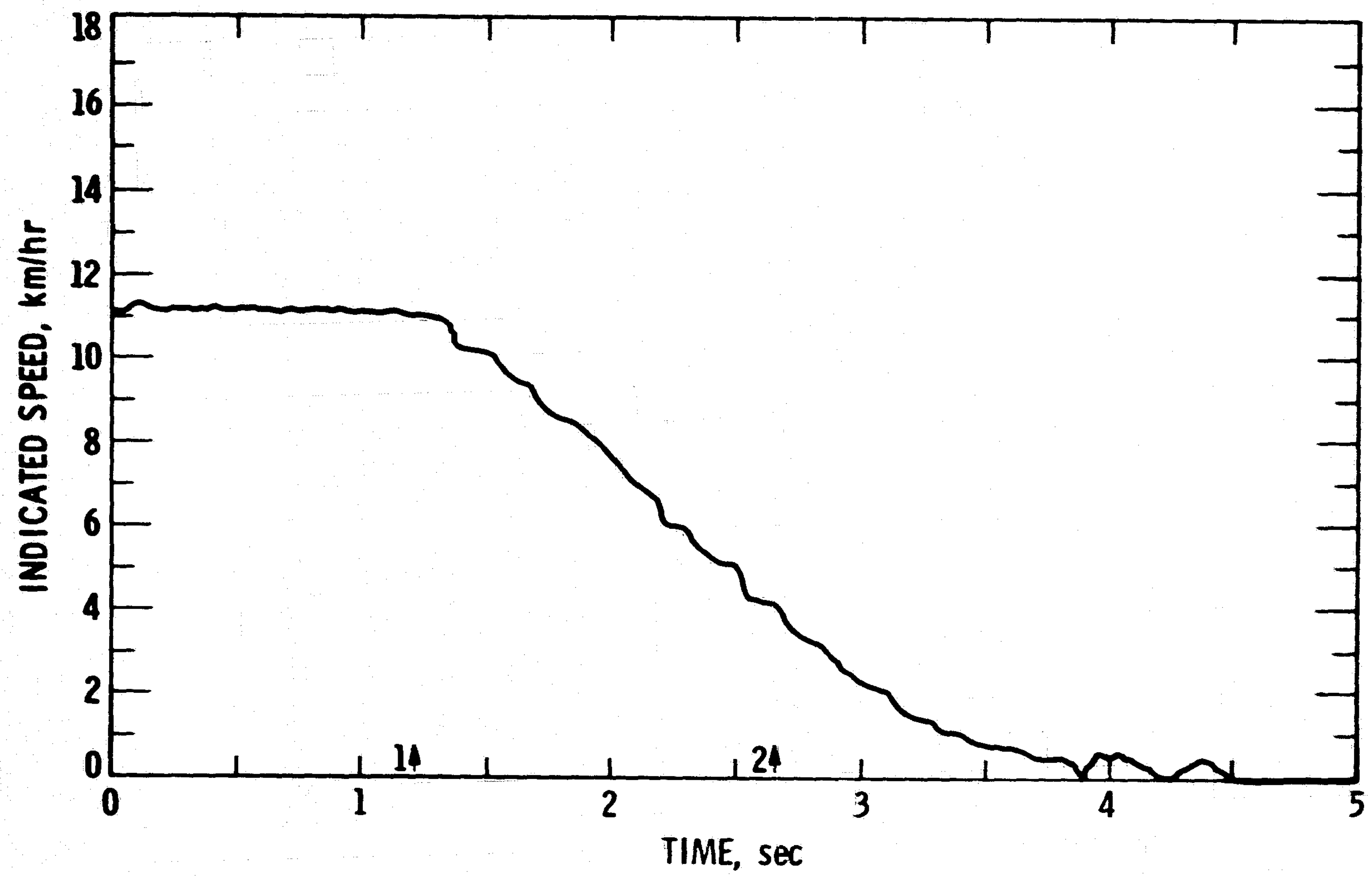


FIGURE 6-2. DECELERATION CURVE. Sensor-induced stop using black (3M Co. Nextel Velvet Coating 101-C10 Black) target. Note the lack of 3.2 km/hr plateau. This is due to a later than normal detection of target by primary sensors, thus leaving less time tc dwell at the 3.2 km plateau.

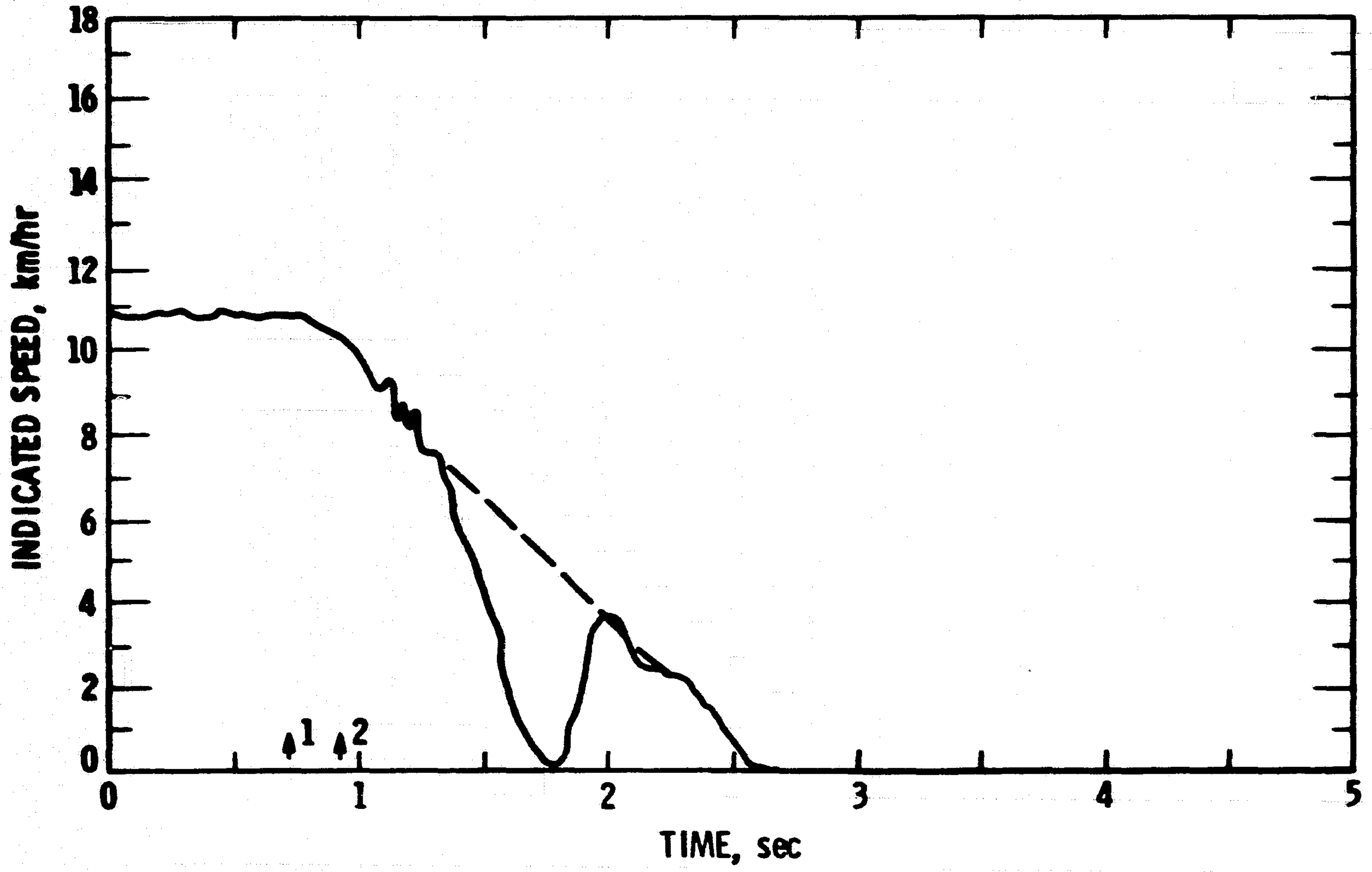


FIGURE 6-3. DECELERATION CURVE. Secondary sensor-induced fast stop. This stop caused by pedestrian jumping into field of view at about 3.3 m in front of the vehicle. First dip in curve to 0 velocity represents rear wheel skidding rather than true vehicle speed. Vehicle travelling down approximately 4% grade. Pedestrian is adult wearing clothing of medium reflectivity.

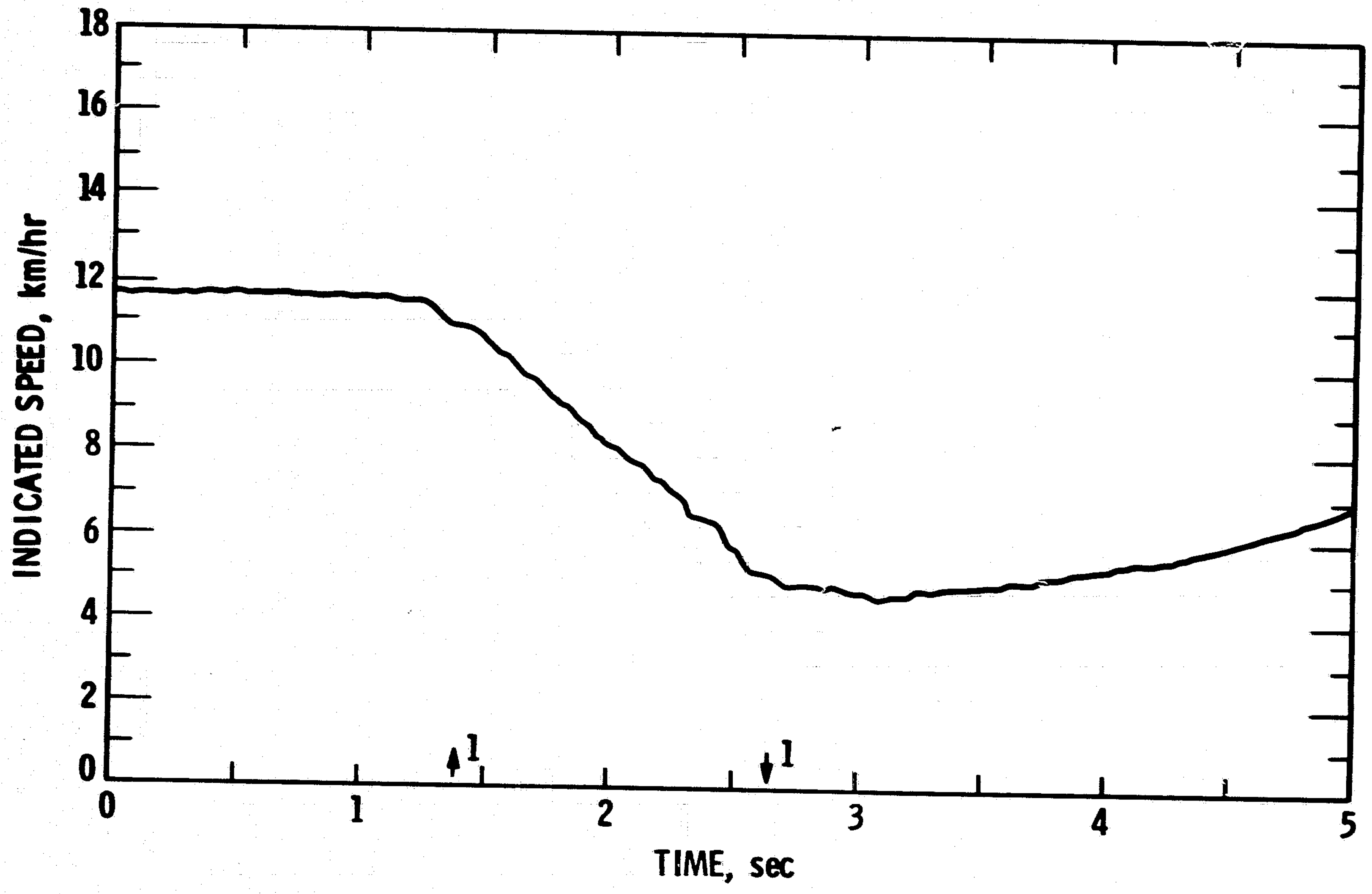


FIGURE 6-4. DECELERATION CURVE. Temporary slowdown caused by pedestrian intersecting primary sensors for a little over 1 s. Vehicle travelling down approximately 4% grade. Pedestrian is adult wearing clothing of medium reflectivity.

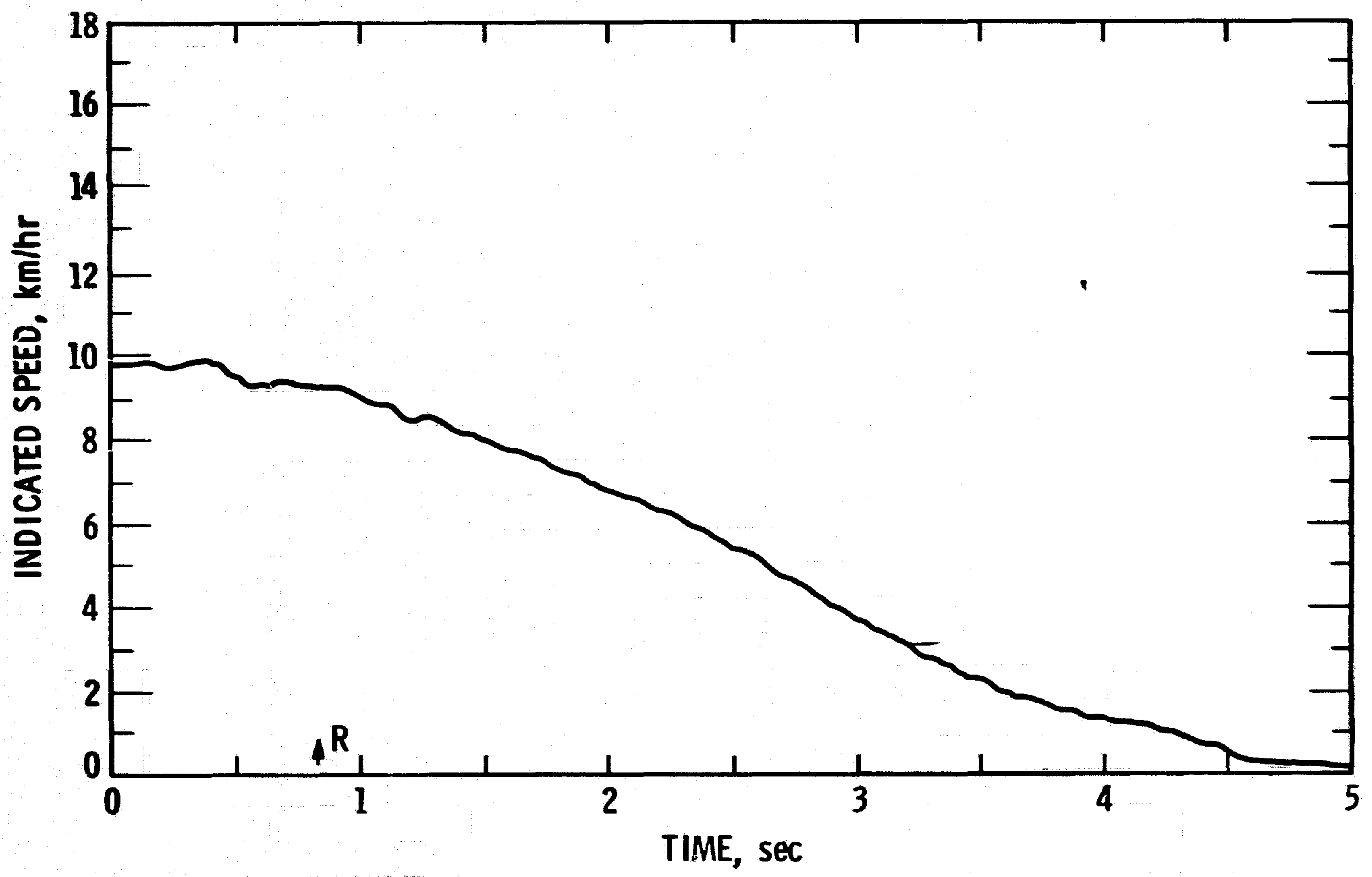


FIGURE 6-5. DECELERATION CURVE. Road stop (induced by sensed road magnet). Vehicle travelling uphill just before sensing magnet. Note that the speed at beginning of stop is below cruising speed. Motor controller turned fully on just prior to stop.

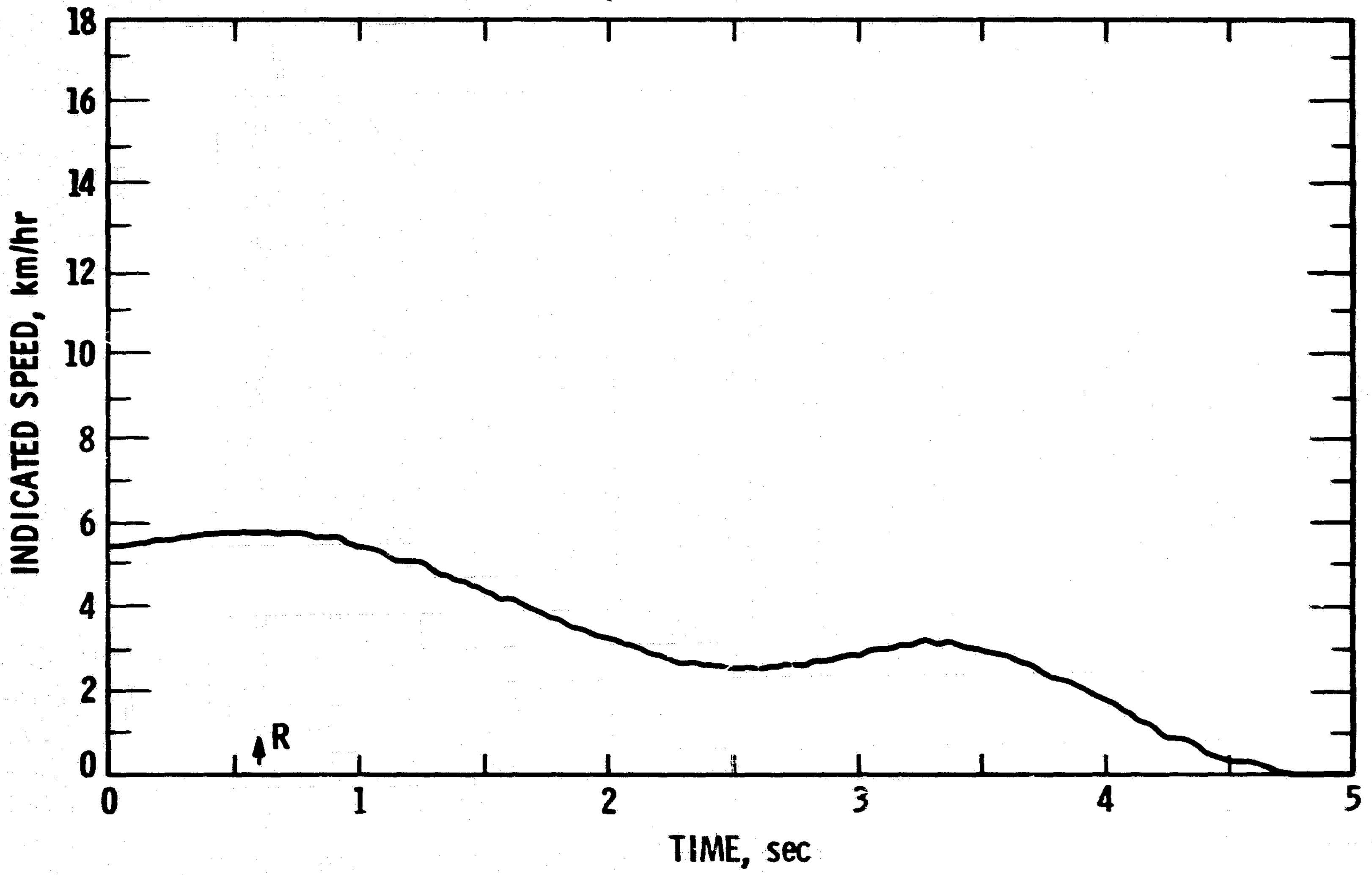


FIGURE 6-6. DECELERATION CURVE. Road stop (induced by sensed road magnet). Vehicle travelling slowly up steep incline just before sensing magnet. Motor and motor controller fully turned on just prior to stop. Also note the two stage stop with the intermediate velocity at nearly 3.2 km/hr.

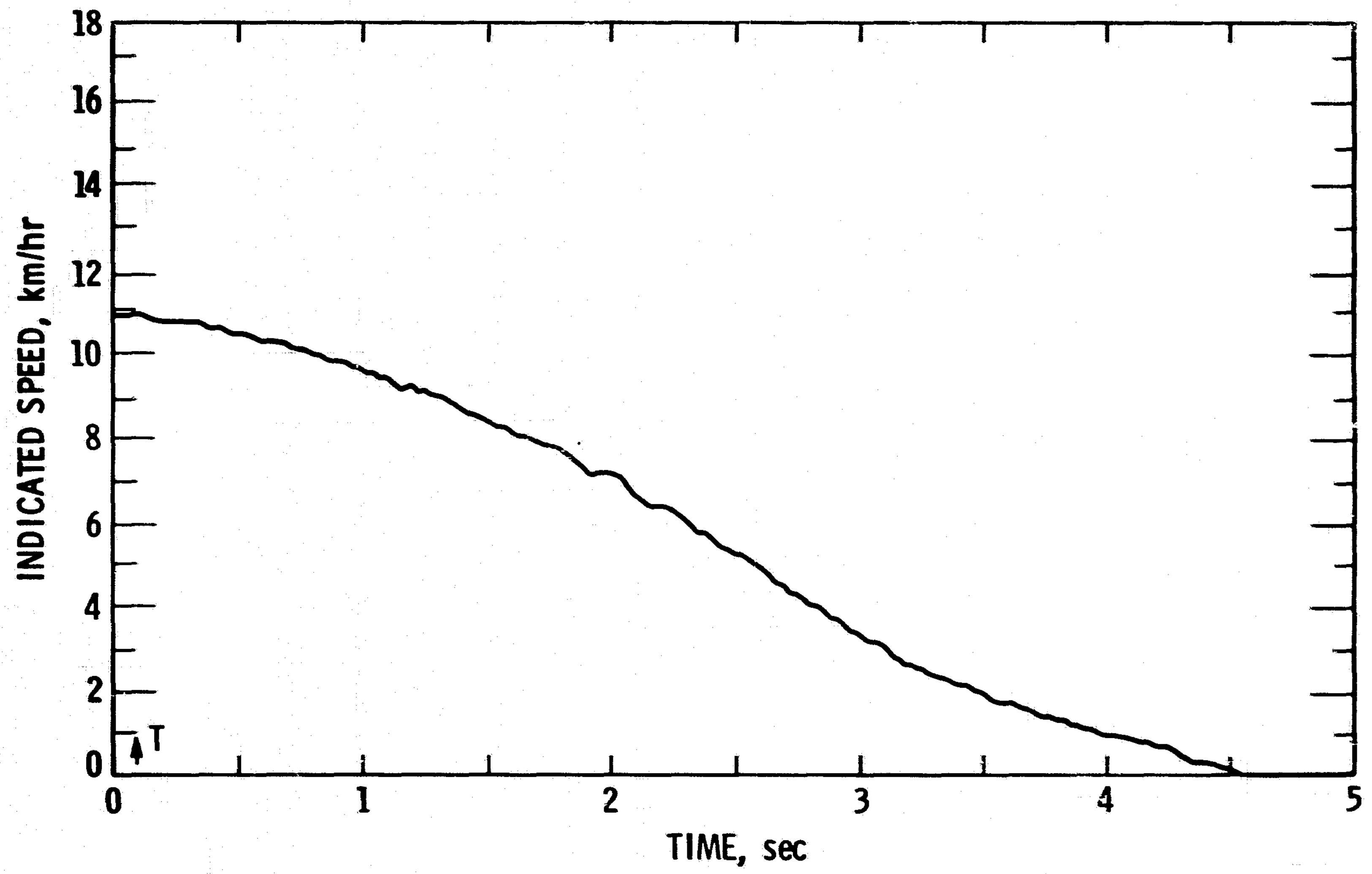


FIGURE 6-7. DECELERATION CURVE. Passenger (tape switch)-induced stop. Vehicle controlling speed at cruise using plugging just prior to stop. Stop made on 4% downhill grade.

Although no table of values for stopping distance will be presented, numerous trials using both black and grey targets and human subjects at various initial speeds and on different grades have proven that the AMTV will stop with approximately .7 m to spare if the sensed obstacle does not suddenly intersect the field of view from the side. Obstacles which suddenly intersect the field of view may be struck only if they do so when the vehicle is going too fast to stop with locked rear wheels. Section 7 will review the algorithms or deceleration philosophy used for sensed obstacle stops.

6.3 ACCELERATION PERFORMANCE

Acceleration from rest is carried out in two stages or plateaus. The vehicle is first accelerated to 3.2 km/h and upon obtaining this speed holds it for approximately 3 s. This initial slow velocity is intended to allow pedestrians and other vehicle drivers to observe and react to AMTV motion. The vehicle is then accelerated to its terminal speed of 11 km/h. On uphill runs this AMTV will not attain this top speed because of lack of motor size (torque). On level or downhill runs the vehicle will limit its speed to approximately 11.2 km/h by reducing forward motor torque or by plugging. Acceleration is limited to .16 g by feedback control using speed sensing and microprocessor calculations. Additionally, jerk, the time rate of change of acceleration, is held to comfortable limits by limiting the rate of change of motor current.

Figure 6-8 illustrates a typical acceleration to cruising speed from rest. This run was recorded on a slight downhill grade.

6.4 SPEED CONTROL

Vehicle speed is controlled at approximately 11 km/h for cruise speed. A speed of 3.2 km/h is used for: turning (such as "U-turns,") for maximum speed when the primary sensor has observed an obstacle, or for an initial control speed after sensing a road stop magnet. The vehicle attempts to reach either speed by either accelerating to it if traveling too slow or by reducing or reversing (plugging) motor torque if travelling too fast. Our measurements indicate that the vehicle speed is controlled to within approximately .3 km/h when not in transition between command speeds.

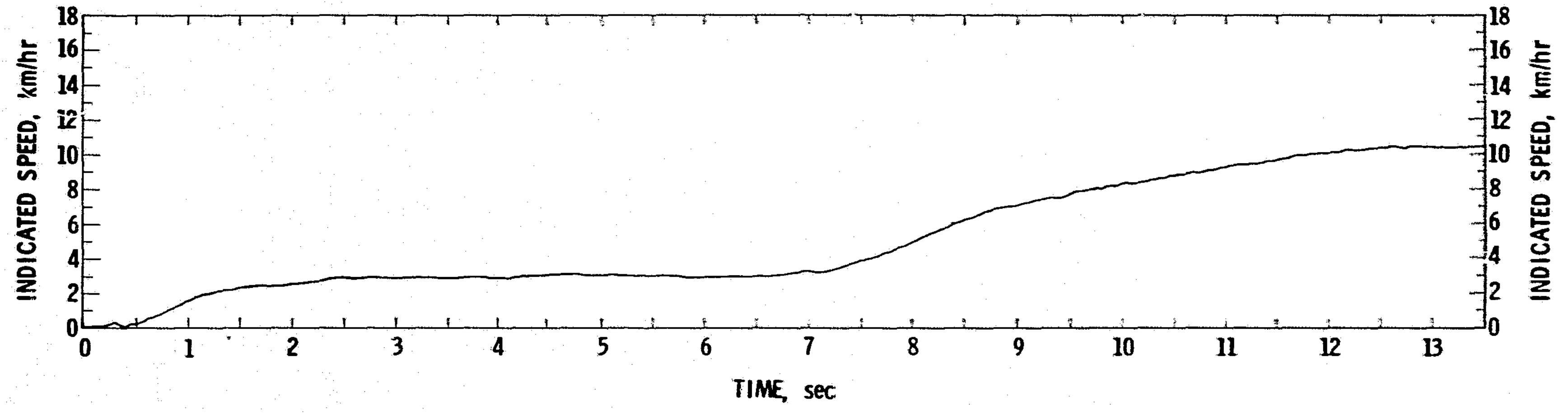


FIGURE 6-8. ACCELERATION CURVE. Two-stage acceleration down 4% grade.

7. SAFETY ASPECTS OF CONTROLLER DESIGN

7.1 KEEP ALIVE FUNCTION

A number of microprocessor hardware failure modes exist. Many of these failure types result in a disruption of the basic timing cycle inherent to the software; that is, the software is designed to cycle through the entire program every 20 ms under all but "emergency stop" conditions. As the program runs, it outputs a "keep-alive" (pulse) signal (analogous to a heartbeat) once every 20 ms on one of its numerous TTL output lines. A special card was designed and fabricated to monitor this signal. Any disruption in this signal's timing will cause this card to initiate a full emergency stop; that is, motor controller turned off and hydraulic brakes turned full on.

The keep-alive card presently has a weakness, in that it uses the same power source as the STD Bus, therefore, a bus power failure may not result in an emergency stop. This weakness will be remedied on the next vehicle.

The circuit has been successfully tested by disconnecting the Z-80 (CPU) card. This card has a built-in self-test button which has also proven to work effectively.

7.2 DUAL DIGITAL TACHOMETERS

The AMTV uses two identical digital tachometers. Each tachometer is driven by an "O"-ring belt attached to a pulley driven by the pinion shaft on the differential. For additional reliability, the two tachometers should be separated and each mounted on a different wheel. Each tachometer is connected by cable to a special tachometer card interfaced to the microprocessor via the STD Bus. Each card yields a 16-bit number which is proportional to speed every 20 ms.

The controller software checks each tachometer card for readiness to deliver its data number. If either or both tachometers fail to be ready to deliver their data within approximately 20 ms the vehicle will make an emergency stop. Assuming that both tachometers can be read, if they do not agree within a predetermined tolerance for a predetermined number of cycles, then a tachometer malfunction is assumed and the vehicle is brought to an emergency stop. This leaves the case that neither tachometer registers a speed greater than zero (such as if both drive belts break simultaneously) but the vehicle is nevertheless moving. This condition is checked by noting the magnitude of the motor control voltage. If this voltage is above a predetermined small quantity, it is assumed that the vehicle must be moving and a reading of zero by either tachometer is considered an error serious enough to warrant an emergency stop.

In summary, a malfunction of either or both tachometers is detected and results in an emergency stop. This safety feature has also been tested and proper operation has been verified under all possible fault conditions.

7.3 BACK-UP SYSTEMS

At the present time the controller does not determine if the motor or motor controller are doing an effective job of stopping the vehicle. In the case where a stop has been commanded (by optical sensor, tape switch, or road stop magnet) and the motor controller or motor fails to obey commands issued by the microprocessor, the following mechanisms are available to bring the vehicle to a stop via the hydraulic brake system: first, the emergency stop push buttons (if depressed by a passenger); second, the front (whisker type) bumper upon contacting some object; and third, if the vehicle exceeds 12 km/h in the forward direction or rolls backwards more than .5 M. This type of stop is termed an "emergency stop" and the vehicle will not resume operation unless the controller reset button is depressed. All other types of stops rely on the ability of the motor to plug.

When executing normal stops, the vehicle uses its hydraulic brake system to bring the vehicle to a complete halt (from a speed below about .3 km/h) and hold the vehicle stopped for the desired period of time. If the hydraulic brake system should fail to operate, the microprocessor controller will complete the stopping process and hold the vehicle stationary using only the motor. Note that this backup system only works from a very low speed when stopping. This system has also been tested and shown to work satisfactorily.

7.4 RUNAWAY OR ERROR DETECTION

As mentioned above, if the vehicle speed exceeds 12 km/h or if the vehicle proceeds backwards by more than .5 M, it is assumed by the microprocessor controller that normal motor speed control is inoperative, and an emergency stop is instituted. An emergency stop consists of opening the main relay (removing all power) of the motor controller and turning the hydraulic brakes full on. Once the vehicle is at rest it can then only be restarted by depressing the controller reset button. Both runaway conditions have been created on the road and the vehicle has responded with emergency stops as programmed.

7.5 VEHICLE ELECTRIC HORN

The AMTV has been fitted with an automotive type (12 V) horn. This horn is presently programmed to sound two short honks just before the vehicle accelerates from rest. It is anticipated that this horn will be used for other purposes as well in the future (see Section 8). This warning sound function has been tested and found to be fairly effective in alerting pedestrians and other vehicles of impending AMTV motion.

7.6 SPECIAL FAST STOP ALGORITHM

The controller now includes a special stopping algorithm designed to bring the vehicle to a fast stop in response to an anomalous optical sensor indication. If the vehicle speed is above 4.7 km/h, the controller calculates the deceleration necessary to stop the vehicle within 1.8 m and then uses plugging to attempt to attain this deceleration level. This mode is intended to function when an obstacle suddenly appears in front of the vehicle, not allowing the vehicle time to make a normal (.16 g) stop. The fast stop can be as severe as an emergency stop in that the rear wheels will slide. This type of stop may not be a comfortable stop for passengers, however, its main function is to protect pedestrians who suddenly move in front of the vehicle at a distance such that the vehicle cannot make a normal stop.

This fast stop algorithm has been extensively tested and found to be very effective.

7.7 FAIL-SAFE ANALYSIS

A failure analysis of the microprocessor controller and of certain critical sensor failures was carried out and the results are tabulated in Appendix C. The analysis was done following the same rationale as the failure modes and effects analysis presented in Reference 1. The present results can be considered an extension of the earlier analysis. A block diagram of the microprocessor controller, shown in Figure 3-3, was used as a guide, with the effect of a failure of each block being determined by analyzing the consequenses of worst-case false inputs from that block. The format of the tables is similar to Reference 1, except that a column indicating when implementation of the preventive action could occur is added.

In summary, a little less than half of the identified items are already implemented or planned for in the initial development phase of the follow-on vehicle (AMTV II). The most effective single test for future development is felt to be that of comparing the actual vehicle speed with a speed predicted by an an approximate algorithm based on existing sensor or passenger inputs. The model could be in a separate Z-80 card, which cross checks with the present card for correct function. Table 7-1 is a condensed summary of the most important results of the analysis.

TABLE 7-1. EXAMPLE OF MICROPROCESSOR CONTROLLER FAILURE MODE ANALYSIS

Element	Type of Failure	Risk	Prevention	Notes
Processor	Processor fails to execute program	4	(a) "Keep-alive" circuit (b) Second processor, see item below	Circuit must detect if program cycle time remains at 20 ms
Speed Control Loop	Failure of speed control function	4	Provide an independent processor with separate power, bus and sensor inputs; One contains software model to predict vehicle speed; Difference >x between predicted and measured speed commands emergency stop health of the first	Detects any failure which results in speed errors, such as tach errors, rollback, etc.; Note that detection of speed errors by predictive model may not require second processor, but second processor can check
TTL I-O	Critical error is one resulting in overspeed	4	(a) Read critical variable redundantly with analog I-O port and compare in software	TTL IO and analog I-O are separate and dissimilar; Need analysis to rule out single point failure that would not activate "keep-alive"
Steering command	Sudden hard-over command	4	(a) Fast detection of abnormal transient in command or in steering angle. Remove actuating pressure so steering will self-center; Command stop	Identified previously in AMTV System & Safety Study; Also a hazard in manual mode; Detection difficult because steering servo requires fast response ($\tau > 0.1$ sec)
Road Marker Sensor	Failure to stop or slow; False high-speed command; Result is overspeed condition	4	(a) Provide redundant sensors (b) Design magnet pattern so that critical commands can be checked for consistency (c) Require high-speed command to be refreshed at intervals	(a) and (b) both required Dissimilar sensor types would be a desirable type of redundancy

8. FINDINGS AND RECOMMENDATIONS

8.1 VEHICLE ALGORITHMS

All algorithms function as written and intended. Fairly extensive road testing and the failure analysis described previously have led us to a number of suggested improvements and modifications which would enhance AMTV performance and safety. These modifications are listed below:

(1) Front and rear turn signals which turn on prior to an actual turn using a coded road-based signal.

(2) Motor and/or motor controller feedback to the microprocessor controller indicating the effectiveness of microprocessor controller commands. This could be supplemented by or even replaced with some type of model, as discussed in the previous section.

(3) Additional horn warnings should be issued for appropriate conditions such as secondary sensor remaining on indicating that the AMTV path is blocked. This algorithm (or modification) would have to use some "sense" in not honking at inappropriate times.

(4) Tachometer signals have turned out to appear somewhat noisy, thus leading to very noisy acceleration data. It is believed that the tachometers are accurate; i.e., there is some randomness in the sensed speed at the tachometers due to gear backlash, tire and belt slippage, and/or eccentricity, etc. This noise is relatively well tolerated by the system. However, more reliable and repeatable performance could be attained if tachometer filtering could be introduced. Additionally, smoother stops would likely result. A tachometer filtering method is therefore recommended.

8.2 VEHICLE SOFTWARE

The Microsoft F-80 FORTRAN used for this task proved to work very satisfactorily. The tachometers are sampled every 20 ms (50 samples/second), thus the time needed to cycle through the program must remain less than 20 ms. Actual measurements indicate that the program executes in about 4 ms and "idles" for about 16 ms waiting for the 20 ms flag to come up. This indicates that only about 20% of the available processing time is actually used to perform calculations, tests, and to input or output data.

ORIGINAL PAGE IS
OF POOR QUALITY

The ability to change program parameters on the street was never developed due to the lack of standard hardware when the task was undertaken and the additional programming necessary. We now have what is probably the correct hardware but have determined from much testing and modification experience that a large percentage of changes involve actual program modification (rather than just modifying a parameter) and small changes take less than 15 min to make using the development system located in the laboratory.

8.3 SOFTWARE DEVELOPMENT SYSTEM

The Micro/Sys development system as now configured (see Section 4) is a self-contained and efficient means for providing software for microprocessor controller use. Future enhancements, upkeep, and replacements are expected, but will not change the system in any fundamental way.

8.4 MOTOR CONTROLLER PERFORMANCE

The new EVC motor controller is now performing adequately under repeated rapid stop conditions but seems to get quite warm, which may not be the case under normal use. Thermostatically-controlled forced cooling would rectify any potential problem. Very low current plugging may not be as smooth and predictable as required for bringing the vehicle to a complete stop without using the hydraulic brakes. At this time, indicated vehicle speed fluctuations overshadow or preclude this use. A method is available to improve the low current plugging ability of the motor current controller. This method consists of direct computer control of the motor current controller input voltage without the use of the provided plugging relay. This relay is presently used to rescale motor control input voltage when plugging current is some small magnitude.

8.5 VEHICLE PERFORMANCE

Vehicle performance is generally smooth and predictable. Stopping and starting characteristics and other performance characteristics can be modified easily by simply changing software.

It is probable that (using the current software) minor software changes would have to be made to adapt an AMTV to a new environment. It is believed that with some field experience the software could be made to function properly in virtually any setting or environment.

The completion of this task has provided a microprocessor-controlled AMTV whose performance is smoother, more flexible, and safer than was presently possible without the microprocessor. In addition, many safety-related improvements are incorporated, or are feasible for future development.

8.6 VEHICLE PERFORMANCE -- SAFETY RELATED

Recommendations are listed below:

(1) Sensors which allow the vehicle to sense obstructions while turning are still needed (References 1 and 2).

(2) Four-wheel hydraulic brakes for emergency stops are desirable. Preferably, this system would incorproate proportionally- controlled pressure or stopping force.

(3) Some sort of motor controller feedback, such as motor current, is important for monitoring stopping performance.

(4) A spring-applied electrically-released brake is required to bring the vehicle to a stop and hold it when a system power failure occurs.

8.7 LONG TERM RESEARCH AND DEVELOPMENT

Having a microprocessor on-board an AMTV opens up the possibility for future development of many improvements in vehicle function and safety. Some examples include the following:

1. Continuous monitoring of sensor and other major subsystem function or "health." This could be accomplished for all vehicle sensors (not just the optical headway sensors) and also for components such as motors, motor controllers, hydraulic system components, lights, etc., in an analogous manner to that which is routinely done on spacecraft. For example, motor current and temperature could be monitored continuosly and compared to the normal range. This monitoring could be used for early warning of impending failure, automatic switchover to back-up systems, and as an aid to system maintenance.

2. Use of an on-board, computer-controlled voice synthesizer. Recent advances in this technology have led to the production of single-chip inexpensive voice synthesizers. These chips use binary-coded information supplied from a microprocessor's memory and output audio frequency analog electrical signals which, when amplified and fed to a speaker, produce clearly understandable human speech. The entire system would be very small, inexpensive, flexible, and reliable. A multitude of specialized

messages could be retained in microprocessor memory, any one, or combination of which, could be announced as a result of a particular state or history of sensor inputs. These messages could instruct or warn passengers or even pedestrians. Such a message might tell a passenger how to restart an AMTV that has come to an emergency stop or request that the passenger check for open or jammed doors or summon help. These messages could be easily modified, updated or added to by simply inserting a single programmmed EPROM on a microprocessor board. Imagine a vehicle that could yell "someone help me" when its sensors indicate that it is in trouble!

3. Self-checking controller software. Additional work needs to be done on microprocessor controller self-checking routines. In particular, subroutines can be used to check the health of the RAM memory chips. With some additional hardware design, perhaps every card in the controller could be tested periodically with the results available via voice synthesizer as mentioned previously, LED character display, or in typewritten form using a plugged in printer. This last form of vehicle health documentation may be particularly useful in terms of long-term maintenance and in cases of liability where maintenence is questioned. The vehicle could literally keep its own maintenance records.

REFERENCES

1. Automated Mixed Traffic Vehicle, AMTV, Technology and Safety Study UMTA-CA-06-0088-78-1.

2. AMTV Headway Sensor and Safety Design Report No. UMTA-CA-06-0088-80-1.

3. Meisenholder, G.W. and Johnston, A.R., "Control Techniques for an Automated Mixed Traffic Vehicle," Proceedings, Joing Automated Control Conference, San Francisco, June 1977, p. 421.

4. Olsen, F.W., "Wire Reference Configurations in Vehicle Lateral Control," IEEE Trans, Vehicular Technology VT-26, 161 (1977).

5. Fenton, R.E., Melocik, G.C., and Olson, K.W., "On the Steering of Automated Vehicles, Theory and Experiment," IEEE Trans, on Automatic Control AC-21, 306 (1976).

6. Domann, H., AFE - An Automatic Guidance System for Road Vehicles," Proceedings, IEEE Workshop on Applied Magnetics, Washington, D.C., p. 140, May 1972.

7. Rodwell, J., "Driverless Bus Nears Reality," Commercial Motor, June 14, 1974; S. Penoyre, Private Communication, Transport and Road Research Laboratory, Crowthorne, Berks, U.K. Feb. 1975.

APPENDIX A

CONTROL ALGORITHMS FLOW CHARTS

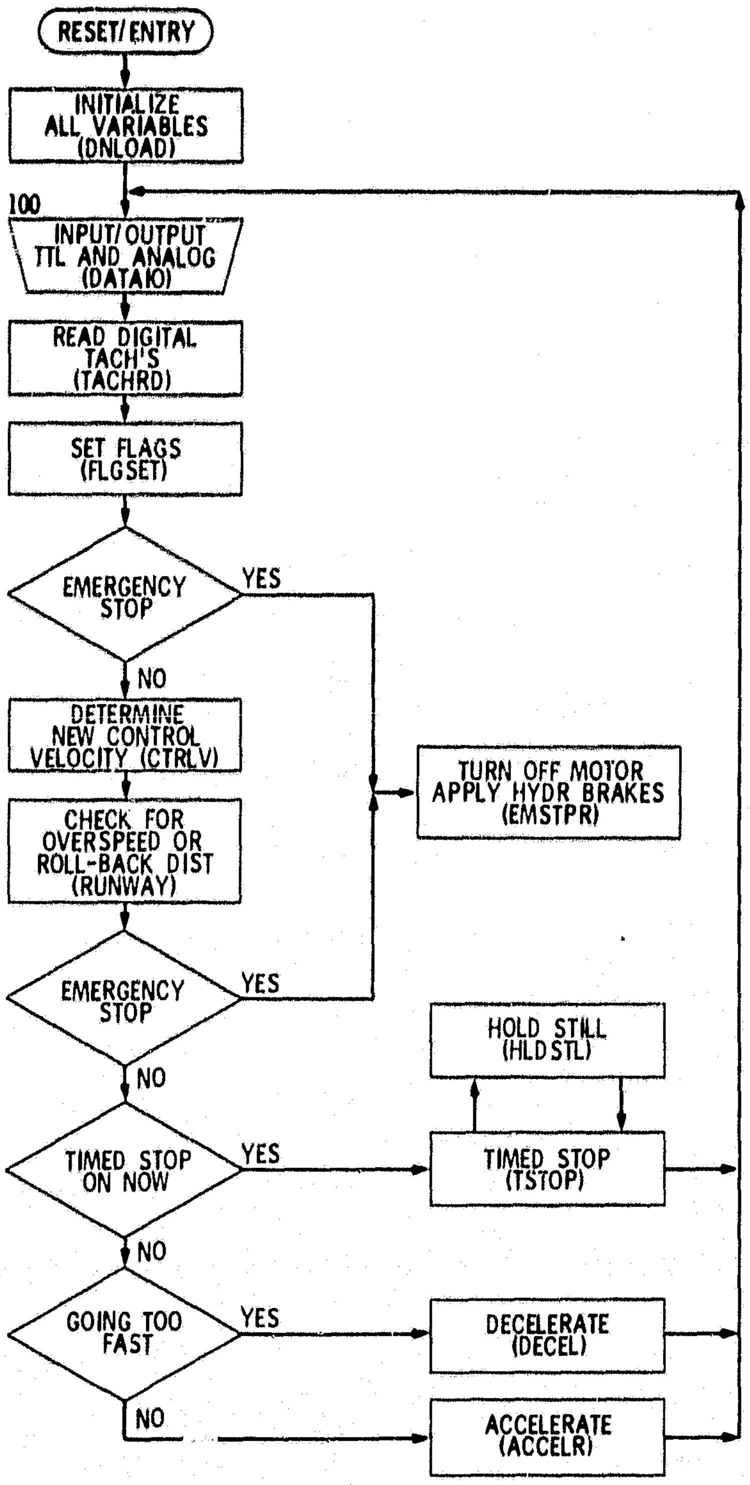

RESET/ENTRY
INITIALIZE ALL VARIABLES (DNLOAD)
100
INPUT/OUTPUT TTL AND ANALOG (DATAIO)
READ DIGITAL TACH'S (TACHRD)
SET FLAGS (FLGSET)
EMERGENCY STOP
YES
NO
DETERMINE NEW CONTROL VELOCITY (CTRLV)
CHECK FOR OVERSPEED OR ROLL-BACK DIST (RUNWAY)
TURN OFF MOTOR APPLY HYDR BRAKES (EMSTPR)
EMERGENCY STOP
YES
NO
HOLD STILL (HLDSTL)
TIMED STOP ON NOW
YES
TIMED STOP (TSTOP)
NO
GOING TOO FAST
YES
DECELERATE (DECEL)
NO
ACCELERATE (ACCELR)

FIGURE A-1. AMTV μP CONTROLLER PROGRAM MAIN PROGRAM

ORIGINAL PAGE IS OF POOR QUALITY

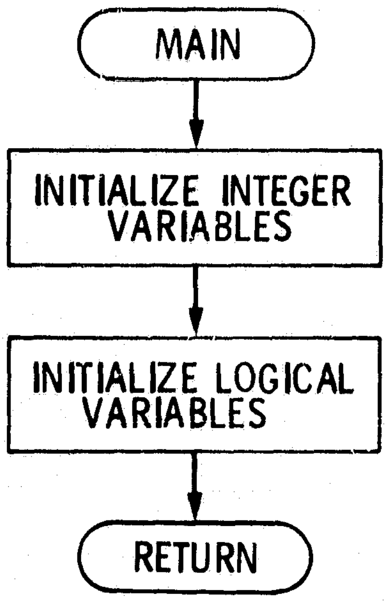


FIGURE A-2. AMTV μP CONTROLLER PROGRAM SUBROUTINE "DNLOAD"

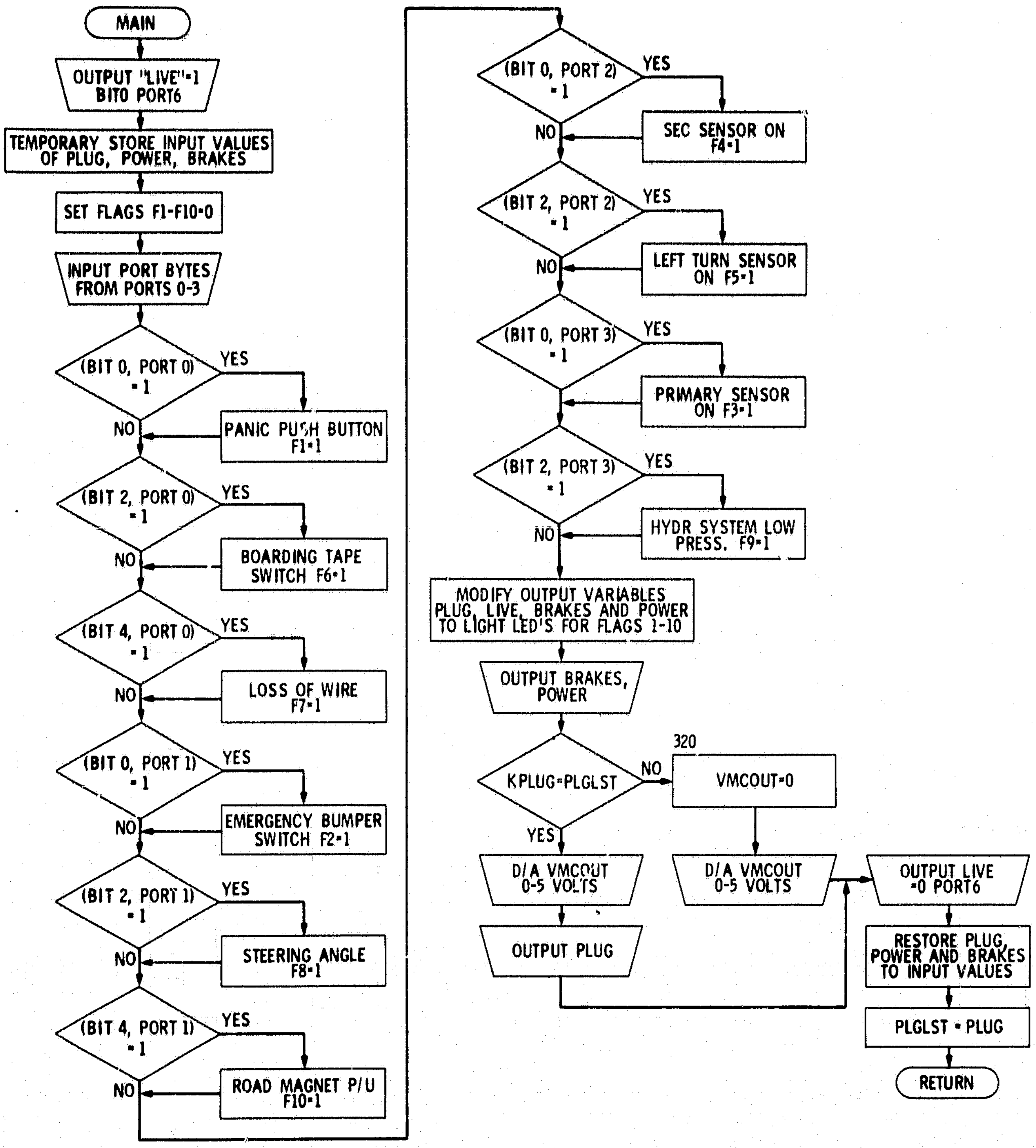


FIGURE A-3. AMTV μP CONTROLLER PROGRAM SUBROUTINE "DATAIO"

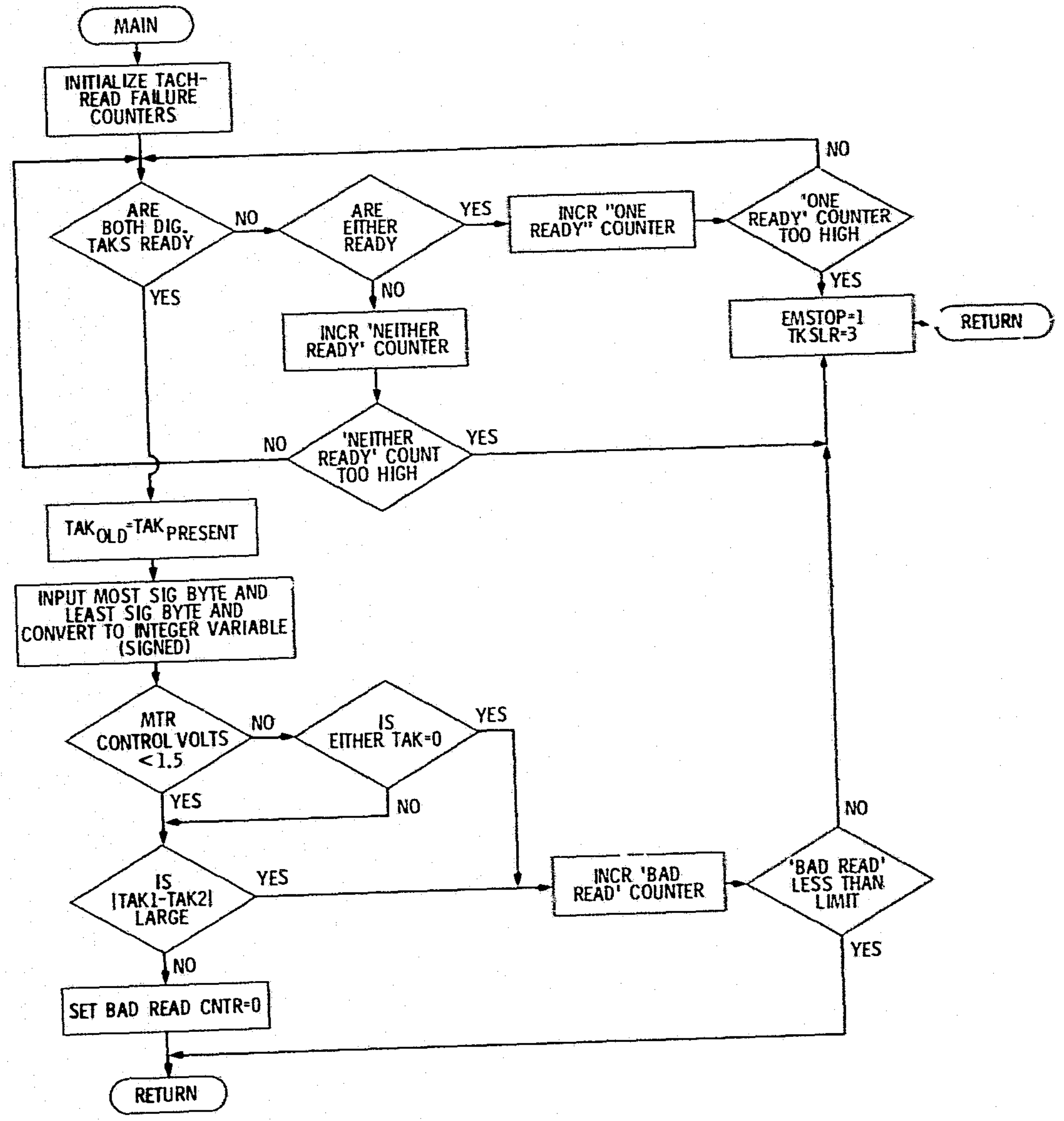

MAIN
INITIALIZE TACH-READ FAILURE COUNTERS
ARE BOTH DIG. TAKS READY
NO
YES
ARE EITHER READY
YES
NO
INCR "ONE READY" COUNTER
'ONE READY' COUNTER TOO HIGH
NO
YES
EMSTOP=1 TKSLR=3
RETURN
INCR 'NEITHER READY' COUNTER
'NEITHER READY' COUNT TOO HIGH
NO
YES
TAK$_{OLD}$=TAK$_{PRESENT}$
INPUT MOST SIG BYTE AND LEAST SIG BYTE AND CONVERT TO INTEGER VARIABLE (SIGNED)
MTR CONTROL VOLTS <1.5
NO
YES
IS EITHER TAK=0
YES
NO
IS \|TAK1-TAK2\| LARGE
YES
NO
INCR 'BAD READ' COUNTER
'BAD READ' LESS THAN LIMIT
NO
YES
SET BAD READ CNTR=0
RETURN

FIGURE A-4. AMTV μP CONTROLLER PROGRAM SUBROUTINE "TACHRD"

ORIGINAL PAGE IS
OF POOR QUALITY

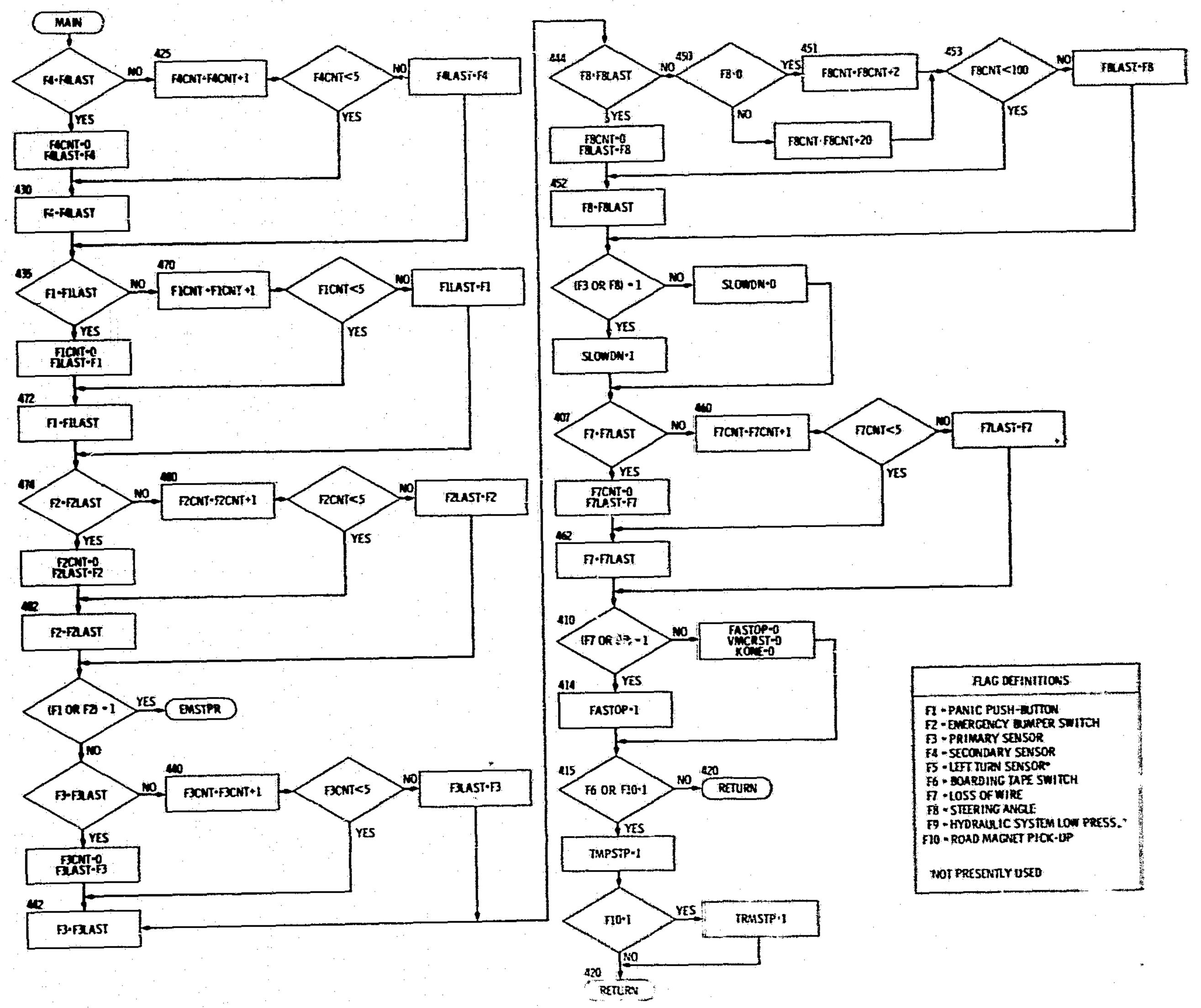


FIGURE A-5. AMTV μP CONTROLLER PROGRAM SUBROUTINE "FLGSET"

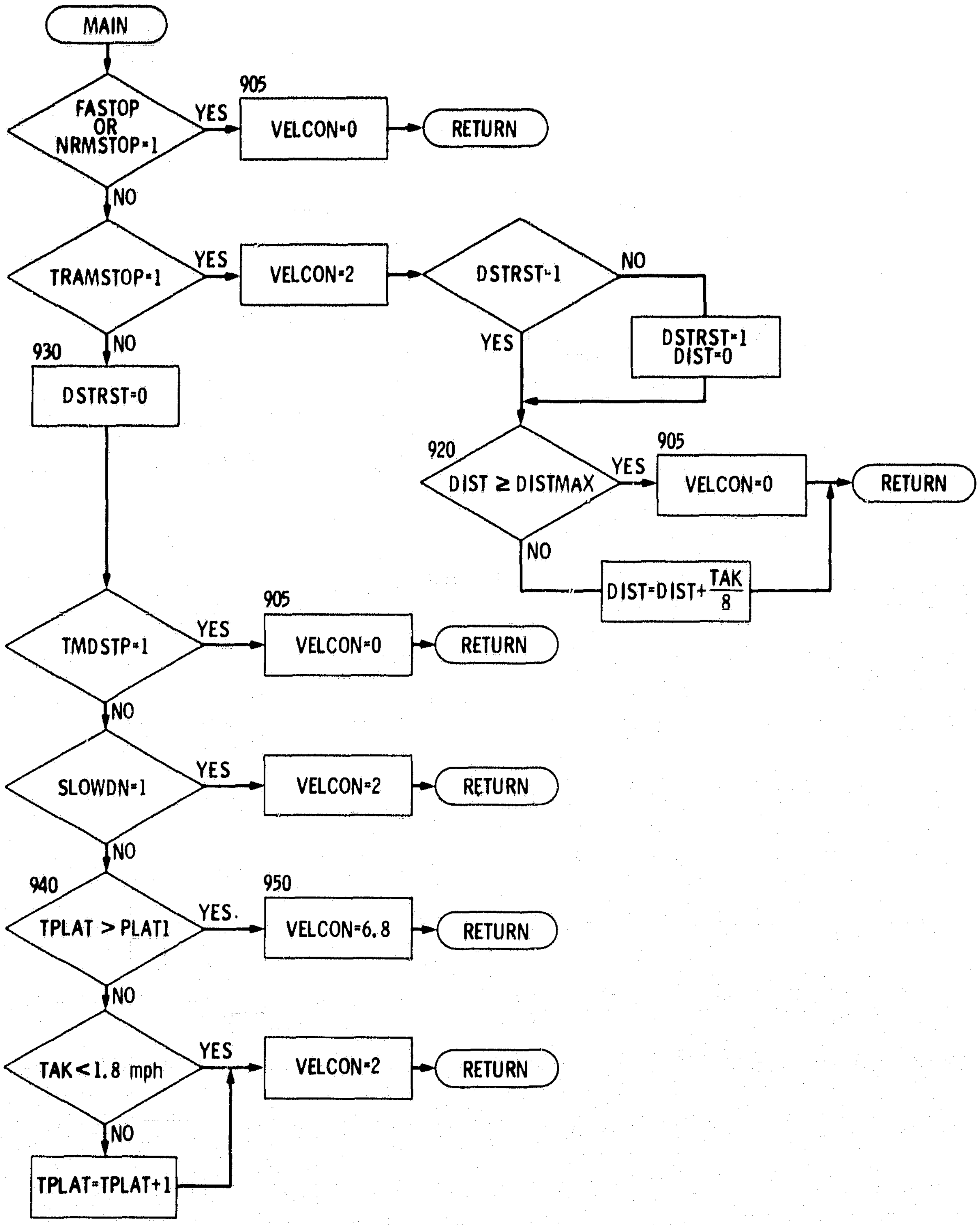


FIGURE A-6. AMTV μP CONTROLLER PROGRAM SUBROUTINE "CTRLV"

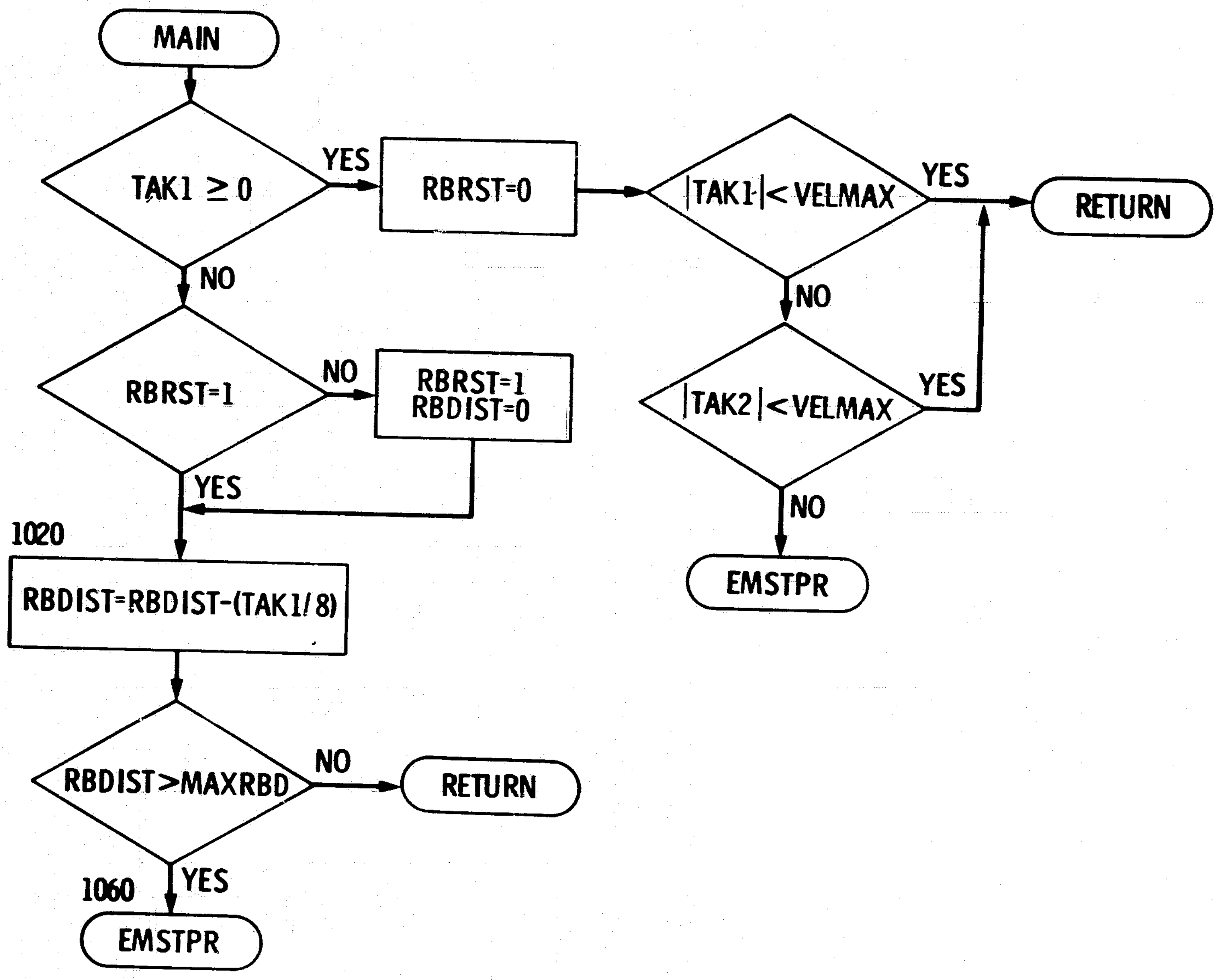


FIGURE A-7. AMTV μP CONTROLLER PROGRAM SUBROUTINE "RUNWAY"

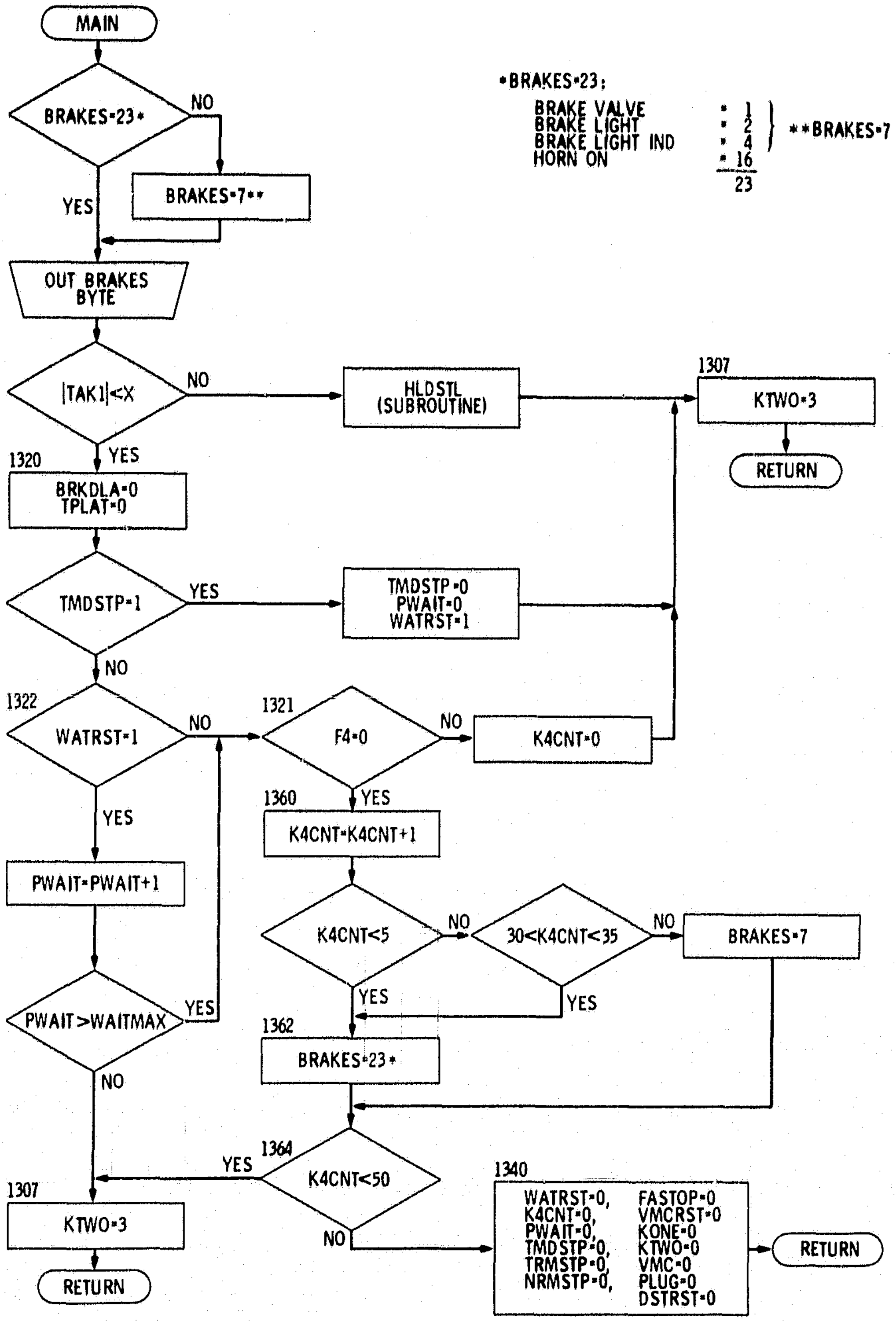


FIGURE A-8. AMTV μP CONTROLLER PROGRAM SUBROUTINE "TSTOP"

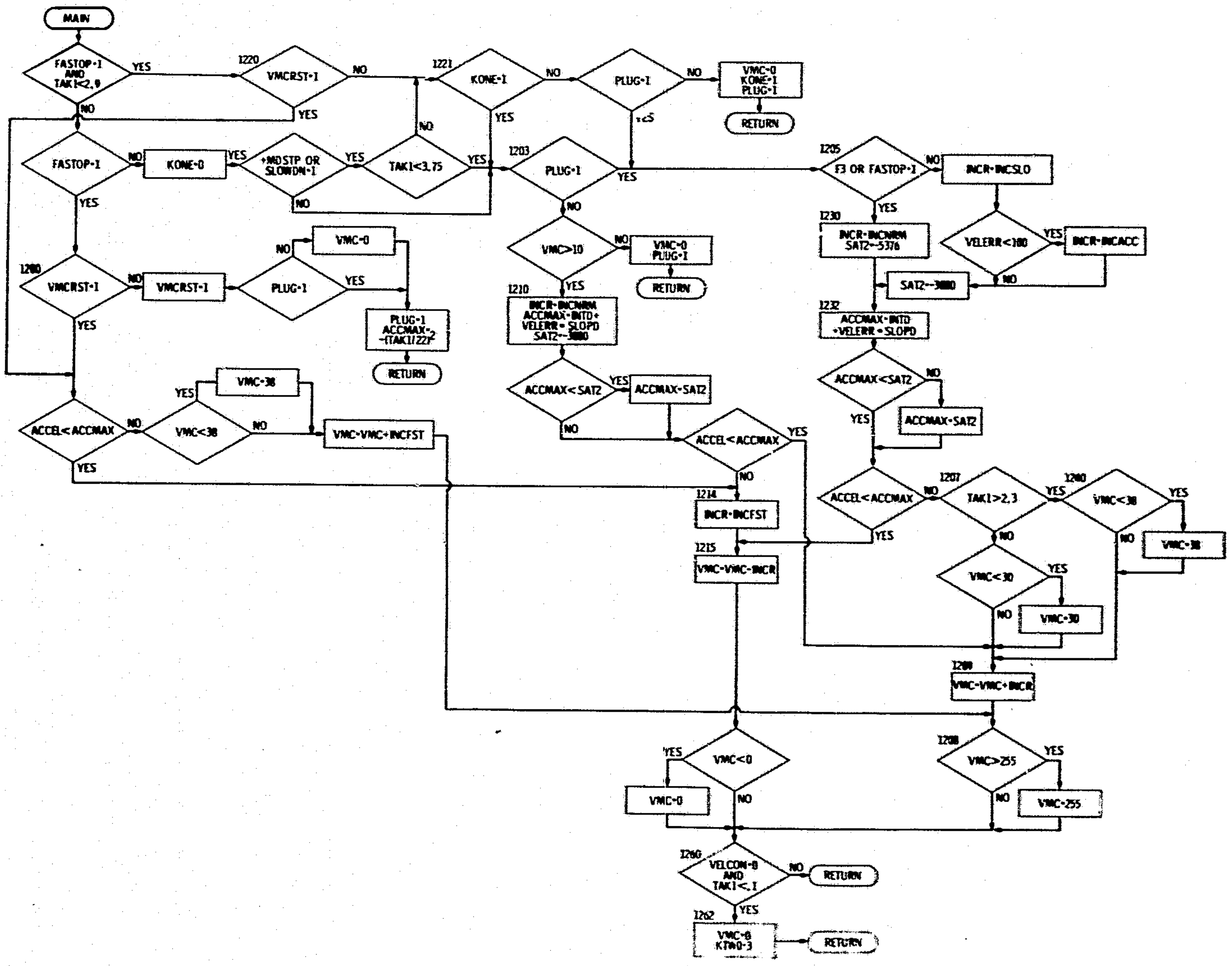


FIGURE A-9. AMTV μP CONTROLLER PROGRAM SUBROUTINE "DECEL"

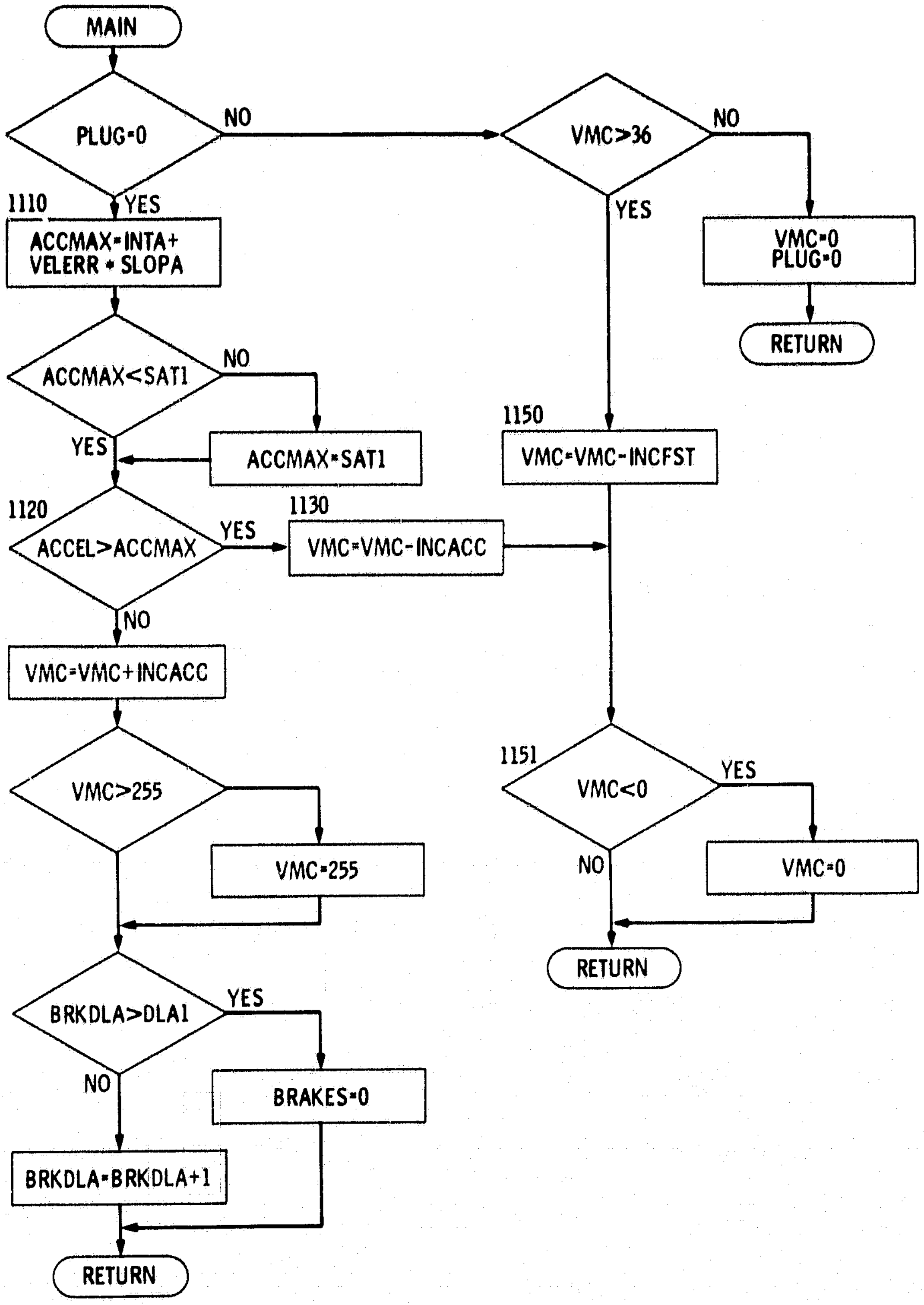


FIGURE A-10. AMTV μP CONTROLLER PROGRAM SUBROUTINE "ACCELR"

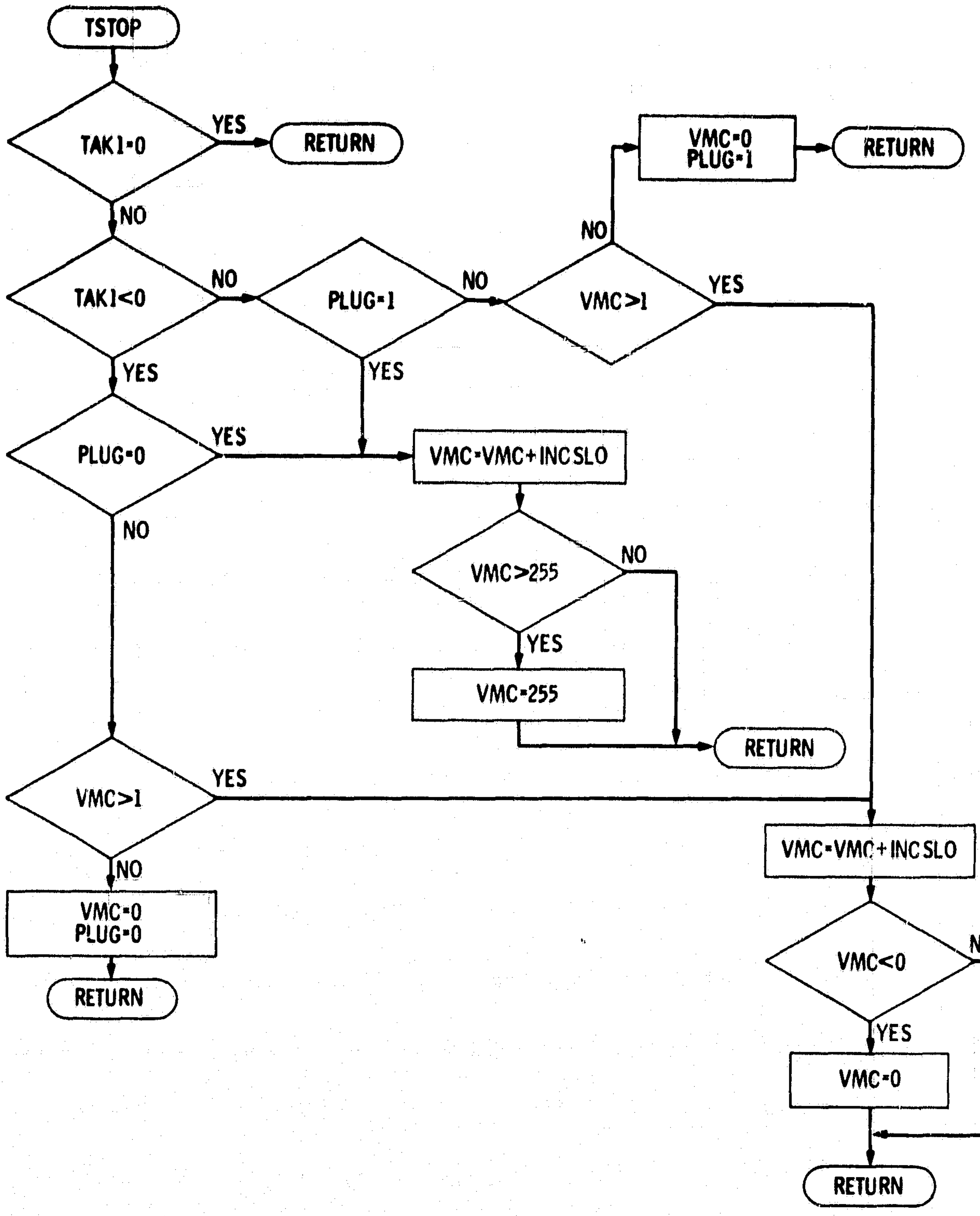
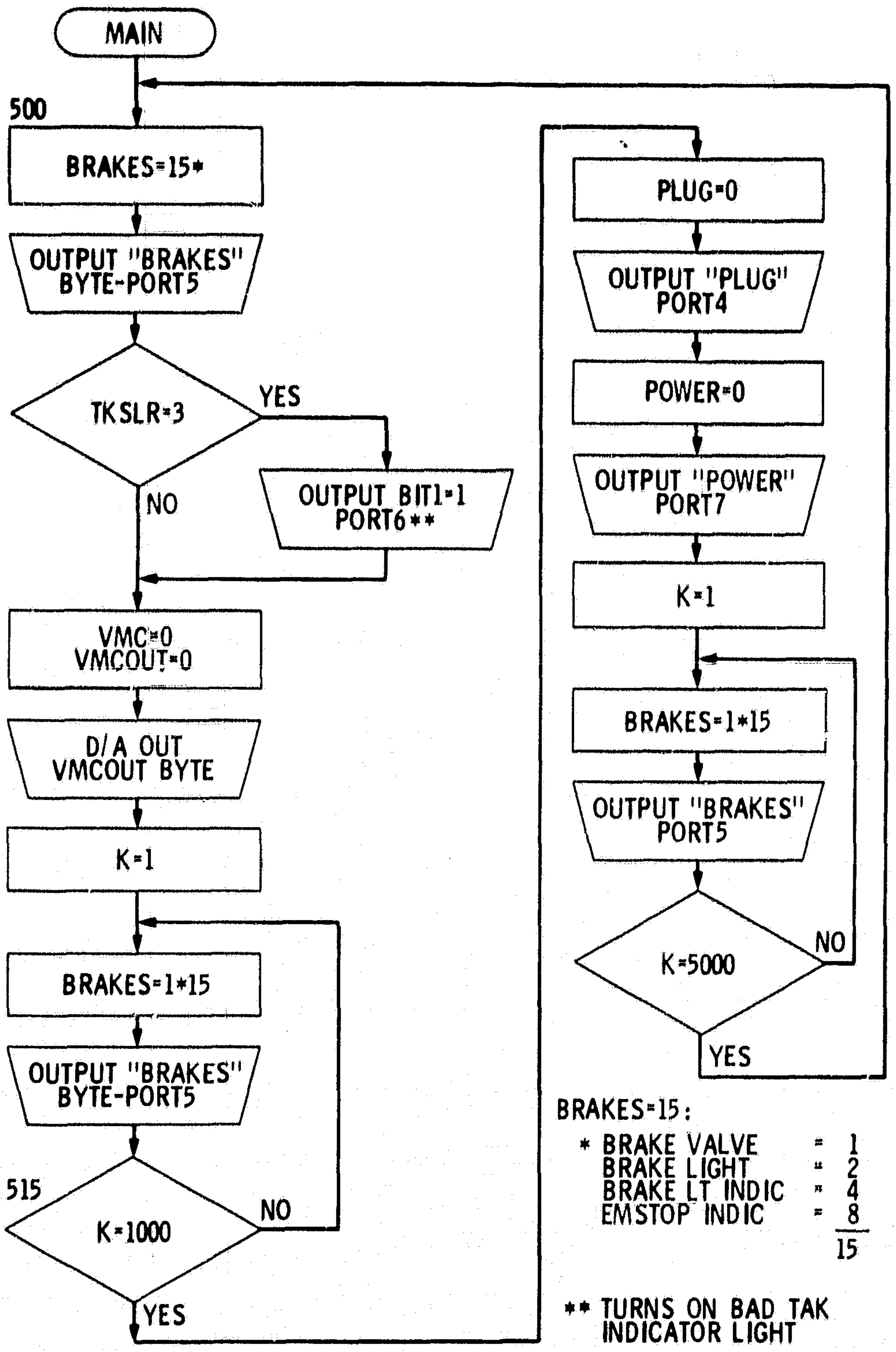


FIGURE A-11. AMTV μP CONTROLLER PROGRAM SUBROUTINE "HLDSTL"



BRAKES=15:

* BRAKE VALVE	=	1
BRAKE LIGHT	=	2
BRAKE LT INDIC	=	4
EMSTOP INDIC	=	8
		15

** TURNS ON BAD TAK INDICATOR LIGHT

FIGURE A-12. AMTV μP CONTROLLER PROGRAM SUBROUTINE "EMSTPR"

ORIGINAL PAGE IS OF POOR QUALITY

APPENDIX B

SOFTWARE LISTINGS

```
C**********************************************************************
C
C   MAIN [1] AMTV CONTROL PROGRAM
C
C   (REVISED 8/21/80) (COMMENTS ADDED 9/16/80)
C
C**********************************************************************
        PROGRAM MAIN
       LOGICAL BADRD,BRAKES,COUNT,COUNT2,DSTRST,EMSTOP,FASTOP,
     *F1,F1CNT,F1LAST,F2,F2CNT,F2LAST,F3,F3CNT,F3LAST,
     *F4,F4CNT,F4LAST,F5,F5CNT,F5LAST,F6,F6CNT,F6LAST,
     *F7,F7CNT,F7LAST,F8,F8CNT,F8LAST,F9,F9CNT,F9LAST,
     *F10,F10CNT,F10LST,
     *KCNT2,KCOUNT,KONE,KTHREE,KTWO,L,L2,MALTAK,
     *NRMSTP,PLUG,POWER,RBRST,RDY1,RDY2,SLOWDN,
     *TAKRDY,TKSLR,TMDSTP,TRMSTP,VMCRST,WATRST,XCOUNT,YCOUNT
       LOGICAL KBRAKE,KLIVE,KPLUG,KPOWER,LIVE,PLGLST
C*****************************************************************
       INTEGER ABSTAK,ABSVEL,ACCEL,ACCMAX,ACCOLD,BRKDLA,DIST,
     *INCACC,INCEM,INCFST,INCNRM,INCR,INCSLO,
     *DISTM,DLA1,INP0,INP1,INP17,INP2,INP3,INP4,
     *INP9,INTA,INTD,K,L2HIGH,L2LOW,LDIFF,LHIGH,
     *LLOW,LSB,MAXRBD,MSB,MSB0,K4CNT,NUMOUT,PLAT1,
     *PWAIT,RBDIST,SAT1,SAT2,SLOPA,SLOPD,TAK1,TAK2,
     *TAK3,TAKOLD,TAKTOL,TPLAT,VELCON,VELERR,VELMAX,VELOC,
     *VMC,VMCOUT,WAITMX,X,Y
C*****************************************************************
       COMMON /L/BADRD,BRAKES,COUNT,COUNT2,DSTRST,EMSTOP,FASTOP,
     *F1,F1CNT,F1LAST,F2,F2CNT,F2LAST,F3,F3CNT,F3LAST,
     *F4,F4CNT,F4LAST,F5,F5CNT,F5LAST,F6,F6CNT,F6LAST,
     *F7,F7CNT,F7LAST,F8,F8CNT,F8LAST,F9,F9CNT,F9LAST,
     *F10,F10CNT,F10LST,
     *KCNT2,KCOUNT,KONE,KTHREE,KTWO,L,L2,MALTAK,
     *NRMSTP,PLUG,POWER,RBRST,RDY1,RDY2,SLOWDN,
     *TAKRDY,TKSLR,TMDSTP,TRMSTP,VMCRST,WATRST,XCOUNT,YCOUNT
       COMMON /L1/KBRAKE,KLIVE,KPLUG,KPOWER,LIVE,PLGLST
C*****************************************************************
       COMMON /I/ABSTAK,ABSVEL,ACCEL,ACCMAX,ACCOLD,BRKDLA,DIST,
     *INCACC,INCEM,INCFST,INCNRM,INCR,INCSLO,
     *DISTM,DLA1,INP0,INP1,INP17,INP2,INP3,INP4,
     *INP9,INTA,INTD,K,L2HIGH,L2LOW,LDIFF,LHIGH,
     *LLOW,LSB,MAXRBD,MSB,MSB0,K4CNT,NUMOUT,PLAT1,
     *PWAIT,RBDIST,SAT1,SAT2,SLOPA,SLOPD,TAK1,TAK2,
     *TAK3,TAKOLD,TAKTOL,TPLAT,VELCON,VELERR,VELMAX,VELOC,
     *VMC,VMCOUT,WAITMX,X,Y
C*****************************************************************
C*********  THIS IS THE BEGINNING OR RESET POINT*****************
C NEXT SUBR DOWNLOADS ALL VARIABLES AND CONSTANTS INTO RAM AND GIVES
C INITIAL CONDITIONS TO ALL OF THEM
        CALL DNLOAD
C NEXT SUBR PERFORMS I/O OF MOST VARIABLES AND SCALES SOME OF THEM
100     CALL DATAIO
```

```
C NEXT SUBR READS TWO DIGITAL TACHOMETERS
        CALL TACHRD
C NEXT SUBR SETS OR CLEARS APPROPRIATE FLAGS BASED ON LATEST TTL INPUTS
        CALL FLGSET
C NEXT STATEMENTS ALLOW IMMEDIATE BRANCH TO EMERG STOP ROUTINE
C IF INDICATED BY ONE OF THE LATEST INPUTS
        IF(EMSTOP.EQ.1) CALL EMSTPR
C NEXT SUBR DETERMINES NEW CONTROL VELOCITY
        CALL CTRLV
C NEXT EXPRESSION CALCULATES VELOCITY ERROR
        VELERR=VELCON-TAK1
C       NEXT EXPRESSIONS USED TO OUTPUT FLAG AND SPEED DATA VIA DAC
C       KACCEL=F3*10+F4*20+F6*30+F10*40
C       KACCEL=TAK2/20
C       CALL POKE(-17,KACCEL)
C       KTAK=TAK1/20
C       CALL POKE(-18,KTAK)
C       NEXT EXPRESSION RESCALES ACCEL FOR LATER USE
        ACCEL=(TAK1-TAKOLD)*256
C NEXT SUBR DETERMINES IF THEIR IS A RUNAWAY CONDITION
        CALL RUNWAY
        IF(EMSTOP.EQ.1) CALL EMSTPR
C NEXT WE DETERMINE IF ACCELERATION OR DECELERATION IS CALLED FOR
C BY COMPARING CONTROL AND ACTUAL VELOCITY; BUT FIRST, ARE WE IN THE
C MIDDLE OF A TIMED STOP? VARIABLE KTWO=3 DURING TIMED STOP PERIOD
        IF(KTWO.EQ.3) GO TO 160
        IF((VELCON-TAK1).LE.0) GO TO 150
140     CALL ACCELR
        GO TO 100
150     CALL DECEL
        GO TO 100
160     CALL TSTOP
        GO TO 100
        END
```

```
C*********************************************************************
C
C  SUBROUTINE DNLOAD [2]...DOWNLOADS AND INITIALIZES VARIABLES INTO RAM
C  (REV. 7/15/80)  (COMMENTS ADDED 9/16/80)
C
C**********************************************************************
       SUBROUTINE DNLOAD
      LOGICAL BADRD,BRAKES,COUNT,COUNT2,DSTRST,EMSTOP,FASTOP,
     *F1,F1CNT,F1LAST,F2,F2CNT,F2LAST,F3,F3CNT,F3LAST,
     *F4,F4CNT,F4LAST,F5,F5CNT,F5LAST,F6,F6CNT,F6LAST,
     *F7,F7CNT,F7LAST,F8,F8CNT,F8LAST,F9,F9CNT,F9LAST,
     *F10,F10CNT,F10LST,
     *KCNT2,KCOUNT,KONE,KTHREE,KTWO,L,L2,MALTAK,
     *NRMSTP,PLUG,POWER,RBRST,RDY1,RDY2,SLOWDN,
     *TAKRDY,TKSLR,TMDSTP,TRMSTP,VMCRST,WATRST,XCOUNT,YCOUNT
      LOGICAL KBRAKE,KLIVE,KPLUG,KPOWER,LIVE,PLGLST
C*****************************************************************
      INTEGER ABSTAK,ABSVEL,ACCEL,ACCMAX,ACCOLD,BRKDLA,DIST,
     *INCACC,INCEM,INCFST,INCNRM,INCR,INCSLO,
     *DISTM,DLA1,INP0,INP1,INP17,INP2,INP3,INP4,
     *INP9,INTA,INTD,K,L2HIGH,L2LOW,LDIFF,LHIGH,
     *LLOW,LSB,MAXRBD,MSB,MSB0,K4CNT,NUMOUT,PLAT1,
     *PWAIT,RBDIST,SAT1,SAT2,SLOPA,SLOPD,TAK1,TAK2,
     *TAK3,TAKOLD,TAKTOL,TPLAT,VELCON,VELERR,VELMAX,VELOC,
     *VMC,VMCOUT,WAITMX,X,Y
C*****************************************************************
      COMMON /L/BADRD,BRAKES,COUNT,COUNT2,DSTRST,EMSTOP,FASTOP,
     *F1,F1CNT,F1LAST,F2,F2CNT,F2LAST,F3,F3CNT,F3LAST,
     *F4,F4CNT,F4LAST,F5,F5CNT,F5LAST,F6,F6CNT,F6LAST,
     *F7,F7CNT,F7LAST,F8,F8CNT,F8LAST,F9,F9CNT,F9LAST,
     *F10,F10CNT,F10LST,
     *KCNT2,KCOUNT,KONE,KTHREE,KTWO,L,L2,MALTAK,
     *NRMSTP,PLUG,POWER,RBRST,RDY1,RDY2,SLOWDN,
     *TAKRDY,TKSLR,TMDSTP,TRMSTP,VMCRST,WATRST,XCOUNT,YCOUNT
      COMMON /L1/KBRAKE,KLIVE,KPLUG,KPOWER,LIVE,PLGLST
C*****************************************************************
      COMMON /I/ABSTAK,ABSVEL,ACCEL,ACCMAX,ACCOLD,BRKDLA,DIST,
     *INCACC,INCEM,INCFST,INCNRM,INCR,INCSLO,
     *DISTM,DLA1,INP0,INP1,INP17,INP2,INP3,INP4,
     *INP9,INTA,INTD,K,L2HIGH,L2LOW,LDIFF,LHIGH,
     *LLOW,LSB,MAXRBD,MSB,MSB0,K4CNT,NUMOUT,PLAT1,
     *PWAIT,RBDIST,SAT1,SAT2,SLOPA,SLOPD,TAK1,TAK2,
     *TAK3,TAKOLD,TAKTOL,TPLAT,VELCON,VELERR,VELMAX,VELOC,
     *VMC,VMCOUT,WAITMX,X,Y
C*****************************************************************
C*******************************************************************
C***************** DOWNLOAD INTEGER VARIABLES **********************
C*******************************************************************
       ABSTAK=0
C SPARE VARIABLE
      ABSVEL=0
C ABSOLUTE VALUE OF VELOCITY FROM DIG TACH #1
```

```
      ACCEL=0
C ACCELERATION  (TAK1-TAKOLD)
      ACCMAX=5376
C MAXIMUM ALLOWED ACCELERATION
      BRKDLA=0
C BRAKE HOLD DELAY TIME, PREVENTS ROLLBACK
      DIST=0
C SCALED DISTANCE REG. CONTAINING DISTANCE PAST PASSENGER STOP MAGNET
C TRUE DISTANCE = 10,500 CTS/FT.  SCALED DIST IS TRUE/8
      DISTM=11150
C SCALED DIST FRM TRAM STP BUTTON @ WHICH CNTRL VEL DROPS TO 0
C SAME SCALING AS VARIABLE DIST ABOVE.
      DLA1=5
C MAXIMUM BRAKE HOLD DELAY TIME (50/SEC)
      INCEM=1
C SUBSTITUTED FOR "INCR" DURING CERTAIN CASES
      INCFST=7
C SUBSTITUTED FOR "INCR" DURING CERTAIN CASES
      INCNRM=4
C SUBSTITUTED FOR "INCR" DURING CERTAIN CASES
      INCSLO=3
C SUBSTITUTED FOR "INCR" DURING CERTAIN CASES
      INCACC=2
C SUBSTITUTED FOR "INCR" DURING ACCELERATION
      INCR=0
C INCREMENT USED IN M/C INCREMENTING EQUATIONS
      INP0=0
C INTEGER VALUE OF BYTE READ FROM PORT #0
      INP1=0
C INTEGER VALUE OF BYTE READ IN FROM PORT #1
      INP17=0
C INTEGER VALUE READ IN FROM PORT #17
      INP2=0
C INTEGER VALUE OF BYTE READ IN FROM PORT #2
      INP3=0
C INTEGER VALUE OF BYTE READ IN FROM PORT #3
      INP4=0
C INTEGER VALUE OF BYTE READ IN FROM PORT #4
      INP9=0
C INTEGER VALUE OF BYTE READ IN FROM PORT #9
      INTA=0
C INTERCPT:VEL ERROR VS ACCMAX CURVE FOR ACCELERATION
      INTD=0
C INTERCEPT:VEL ERROR VS ACCMAX CURVE FOR DECELERATION
       K=0
C SPARE VARIABLE
       L2HIGH=0
C HIGH ORDER BYTE FOR TAK2
       L2LOW=0
C LOW ORDER BYTE FOR TAK2
       LDIFF=0
C DIFFERENCE BETWEEN TAK1 AND TAK2
```

```
         LHIGH=0
C HIGH ORDER BYTE FOR TAK1
         LLOW=0
C LOW ORDER BYTE FOR TAK1
         LSB=0
C SPARE VARIABLE
         MAXRBD=2600
C MAXIMUM ALLOWED ROLL BACK DISTANCE DIVIDED BY 8 TRUE.
C TRUE SCALE FACTOR IS (10,500 CTS/FT)
         MSB=0
C SPARE VARIABLE
         MSBO=0
C SPARE VARIABLE
        K4CNT=0
C COUNTER USED FOR HONKING HORN
        NUMOUT=0
C SPARE VARIABLE
        PLAT1=200
C PLATEAU TIME PERIOD MAXIMUM VALUE (50/SEC)
        PWAIT=0
C PASSENGER STOP WAIT PERIOD REG. (50/SEC)
         RBDIST=0
C ROLLBACK DISTANCE REGISTER
        SAT1=5376
C MAXIMUM (SATURATION LEVEL) FOR ACCMAX CURVE
         SAT2=-5376
C MAXIMUM (SATURATION LEVEL) FOR DECEL MAX CURVE
        SLOPA=10
C SLOPE:VEL ERROR VS ACCMAX CURVE FOR ACCEL
        SLOPD=10
C SLOPE:VEL ERROR VS ACCMAX CURVE DURING ACCEL
        TAK1=0
C PRESENT VALUE OF DIG. TACH. #1
         TAK2=0
C PRESENT VALUE OF DIG. TACH. #2
         TAK3=0
C SPARE VARIABLE
         TAKOLD=0
C LAST VALUE OF DIG TAK1
         TAKTOL=200
C LARGEST ALLOWABLE DIFFERENCE BETWEEN TAK1 AND TAK2
        TPLAT=0
C PLAT. TIME PRD REG FOR 2 STAGE ACCEL, (50/S)
        VELCON=0
C CONTROL VELOCITY
        VELERR=0
C VELOCITY ERROR (CONTROL-ACTUAL)
        VELMAX=2300
C MAXIMUM ALLOWED VEHICLE VELOCITY ( APPROXIMATELY 310/1 MPH)
        VMC=0
C MOTOR CONTROL VOLTAGE [0-255]=[0-10VOLTS]
        VMCOUT=0
```

```
C SCALED MTR CNTRL VOLTS FOR DAC
      WAITMX=150
C WAIT PERIOD DURING TIMED STOP (50/SEC)
      X=50
C VEL TOL ABOVE WHICH MTR USED TO HLD VELOC=0,DIG TAK (310/MPH)
      Y=20
C SPARE VARIABLE
C*********************************************************************
C ****************** NOW DOWNLOAD LOGICAL VARIABLES ****************
C*********************************************************************
       BADRD=0
C REGISTER FOR # OF TIMES LDIFF > TAKTOL
      BRAKES=0
C HYDRAULIC BRAKES VALVE CNTRL SIG. ON=1
      COUNT=0
C REG. # OF TIMES ONLY ONE TACH READY TO READ
      COUNT2=0
C REG. # OF TIMES THAT NO DIG TACH READY TO READ
      DSTRST=0
C DISTANCE REGISTER RESET FLAG PROGRMD STOPS,1=RESET
      EMSTOP=0
C EMERGENCY STOP FLAG , STOP=1
      F1=0
C EMERG STOP PB SWITCH
      F1CNT=0
C COUNTER USED TO COUNT # OF CONSEQUTIVE TIMES F1=1
      F1LAST=0
C HOLDS LAST VALUE OF F1 UNTIL F1 HAS REMAINED FLIPPED FOR N TIMES
      F2=0
C BUMPER SWITCH
       F2CNT=0
C COUNTER USED TO COUNT # OF CONSEQUTIVE TIMES F2=1
      F2LAST=0
C HOLDS LAST VALUE OF F2 UNTIL F2 HAS REMAINED FLIPPED FOR N TIMES
      F3=0
C PRIMARY SENSOR, DETECTS SOMETHING=1
       F3CNT=0
C COUNTER USED TO COUNT # OF CONSEQUTIVE TIMES F3=1
      F3LAST=0
C HOLDS LAST VALUE OF F3 UNTIL F3 HAS REMAINED FLIPPED FOR N TIMES
      F4=0
C SECONDARY SENSOR, DETECTS SOMETHING=1
      F4CNT=0
C COUNTER USED TO COUNT NUMBER OF CONSECUTIVE TIMES F4=1
      F4LAST=0
C KEEPS LAST VALUE OF F4 UNTIL F4 HAS REMAINED FLIPPED FOR 5 CONSECUTIVE TIMES
      F5=0
C LEFT TURN SENSOR, DETECTS SOMETHING=1
       F5CNT=0
C COUNTER USED TO COUNT # OF CONSEQUTIVE TIMES F5=1
      F5LAST=0
C HOLDS LAST VALUE OF F5 UNTIL F5 HAS REMAINED FLIPPED FOR N TIMES
```

```
      F6=0
C BOARDING TAPE SWITCH, CLOSED=1
       F6CNT=0
C COUNTER USED TO COUNT # OF CONSEQUTIVE TIMES F6=1
      F6LAST=0
C HOLDS LAST VALUE OF F6 UNTIL F6 HAS REMAINED FLIPPED FOR N TIMES
      F7=0
C LOSS OF WIRE SIGNAL, LOST=1
       F7CNT=0
C COUNTER USED TO COUNT # OF CONSEQUTIVE TIMES F7=1
      F7LAST=0
C HOLDS LAST VALUE OF F7 UNTIL F7 HAS REMAINED FLIPPED FOR N TIMES
      F8=0
C STEERING ANGLE INDICATOR, TURNING=1
       F8CNT=0
C COUNTER USED TO COUNT # OF CONSEQUTIVE TIMES F8=1
      F8LAST=0
C HOLDS LAST VALUE OF F8 UNTIL F8 HAS REMAINED FLIPPED FOR N TIMES
      F9=0
C HYDRAULIC SYSTEM LOW PRESSURE, LOW=1
       F9CNT=0
C COUNTER USED TO COUNT # OF CONSEQUTIVE TIMES F9=1
      F9LAST=0
C HOLDS LAST VALUE OF F9 UNTIL F9 HAS REMAINED FLIPPED FOR N TIMES
       F10=0
C ROAD MARKER DETECTION SIGNAL, DETECT=1
       F10CNT=0
C COUNTER USED TO COUNT # OF CONSEQUTIVE TIMES F10=1
      F10LST=0
C HOLDS LAST VALUE OF F10 UNTIL F10 HAS REMAINED FLIPPED FOR N TIMES
      FASTOP=0
C FAST STOP FLAG, STOP=1
       KBRAKE=0
C USED IN SUBR DATAIO, RETAINS VALUE OF BRAKES
       KCNT2=0
C REGISTER FOR # OF TIMES NEITHER DIG TAK READY TO READ
       KCOUNT=0
C REGISTER FOR # OF TIMES ONLY ONE DIG TAK READY TO READ
       KLIVE=0
C USED IN SUBR DATAIO,RETAINS VALUE OF LIVE
       KPLUG=0
C USED IN SUBR DATAIO, RETAINS VALUE OF PLUG
       KPOWER=0
C USED IN SUBR DATAIO, RETAINS VALUE OF POWER
      KONE=0
C SPARE VARIABLE
      KTWO=1
C USED AS FLAG TO INDICATE TIMED STOP UNDERWAY (KTWO=3 DURING TIMED STOP)
       L=0
C STATUS REGISTER FOR TAK1
       L2=0
C STATUS REGISTER FOR TAK2
```

```
        LIVE=0
C INITIAL VALUE FOR OUTPUT PORT 6
      MALTAK=0
C USED TO LIGHT TACH MALFUNCTION INDIC. LT. 1=LT ON
      NRMSTP=0
C NORMAL STOP FLAG, STOP=1
      PLUG=0
C MOTOR CONTROL PLUG/NO PLUG CONTROL:PLUG=0>FORWARD MOTION
        PLGLST=0
C LAST VALUE OF PLUG USED IN SUBR DATAIO
      POWER=1
C USED TO HOLD MAIN PWR RELAY CLOSED, 1=CLOSED
      RBRST=0
C ROLLBACK DIST. REG. RESET FLAG, 1=RESET
      RDY1=0
C FLAG FROM TACH 1 INDIC READY TO BE READ
      RDY2=0
C FLAG FROM TACH 2 INDIC READY TO BE READ
      SLOWDN=0
C SET=1 IN SUBR. FLGSET IF PRIMARY SENSOR OR STEERING ANGLE ARE ON
      TAKRDY=0
C TACH. OUTPUTS READY FOR READING
      TKSLR=1
C USED AS FLAG INDICATING BAD TACHOMETER
      TMDSTP=0
C TIMED STOP FLAG, 1=SET
      TRMSTP=0
C SET AFTER ROAD MRKR DETECTED. USED IN TIMED STOP
      VMCRST=0
C RESET FLAG USED IN DECEL WHEN FAST STOP MUST BE MADE
      WATRST=0
C PASSENGER WAIT REG RESET FLG, 1=RESET
      XCOUNT=5
C MAX # "COUNT" CAN REACH, THEN EMERGENCY ACTION. USED IN SUBR TACHRD
      YCOUNT=5
C MAX # "COUNT2" CAN REACH,THEN EMERGENCY ACTION. USED IN SUBR TACHRD
        RETURN
        END
```

```
C*************************************************************************
C
C      TACHRD [4]....TACHOMETER READING SUBROUTINE
C
C      (REVISED 8/11/80)  (COMMENTS ADDED 9/16/80)
C
C*************************************************************************
C THIS SUBR READS AND COMPARES DIGITAL TACH'S.
C IF THE DIGITAL TACH'S DONT AGREE (W/IN TOLERANCE) WITH ONE ANOTHER
C AN EMERGENCY STOP FLAG IS SET.
        SUBROUTINE TACHRD
      LOGICAL BADRD,BRAKES,COUNT,COUNT2,DSTRST,EMSTOP,FASTOP,
     *F1,F1CNT,F1LAST,F2,F2CNT,F2LAST,F3,F3CNT,F3LAST,
     *F4,F4CNT,F4LAST,F5,F5CNT,F5LAST,F6,F6CNT,F6LAST,
     *F7,F7CNT,F7LAST,F8,F8CNT,F8LAST,F9,F9CNT,F9LAST,
     *F10,F10CNT,F10LST,
     *KCNT2,KCOUNT,KONE,KTHREE,KTWO,L,L2,MALTAK,
     *NRMSTP,PLUG,POWER,RBRST,RDY1,RDY2,SLOWDN,
     *TAKRDY,TKSLR,TMDSTP,TRMSTP,VMCRST,WATRST,XCOUNT,YCOUNT
      LOGICAL KBRAKE,KLIVE,KPLUG,KPOWER,LIVE,PLGLST
C*****************************************************************
      INTEGER ABSTAK,ABSVEL,ACCEL,ACCMAX,ACCOLD,BRKDLA,DIST,
     *INCACC,INCEM,INCFST,INCNRM,INCR,INCSLO,
     *DISTM,DLA1,INP0,INP1,INP17,INP2,INP3,INP4,
     *INP9,INTA,INTD,K,L2HIGH,L2LOW,LDIFF,LHIGH,
     *LLOW,LSB,MAXRBD,MSB,MSBO,K4CNT,NUMOUT,PLAT1,
     *PWAIT,RBDIST,SAT1,SAT2,SLOPA,SLOPD,TAK1,TAK2,
     *TAK3,TAKOLD,TAKTOL,TPLAT,VELCON,VELERR,VELMAX,VELOC,
     *VMC,VMCOUT,WAITMX,X,Y
C*****************************************************************
      COMMON /L/BADRD,BRAKES,COUNT,COUNT2,DSTRST,EMSTOP,FASTOP,
     *F1,F1CNT,F1LAST,F2,F2CNT,F2LAST,F3,F3CNT,F3LAST,
     *F4,F4CNT,F4LAST,F5,F5CNT,F5LAST,F6,F6CNT,F6LAST,
     *F7,F7CNT,F7LAST,F8,F8CNT,F8LAST,F9,F9CNT,F9LAST,
     *F10,F10CNT,F10LST,
     *KCNT2,KCOUNT,KONE,KTHREE,KTWO,L,L2,MALTAK,
     *NRMSTP,PLUG,POWER,RBRST,RDY1,RDY2,SLOWDN,
     *TAKRDY,TKSLR,TMDSTP,TRMSTP,VMCRST,WATRST,XCOUNT,YCOUNT
      COMMON /L1/KBRAKE,KLIVE,KPLUG,KPOWER,LIVE,PLGLST
C*****************************************************************
      COMMON /I/ABSTAK,ABSVEL,ACCEL,ACCMAX,ACCOLD,BRKDLA,DIST,
     *INCACC,INCEM,INCFST,INCNRM,INCR,INCSLO,
     *DISTM,DLA1,INP0,INP1,INP17,INP2,INP3,INP4,
     *INP9,INTA,INTD,K,L2HIGH,L2LOW,LDIFF,LHIGH,
     *LLOW,LSB,MAXRBD,MSB,MSBO,K4CNT,NUMOUT,PLAT1,
     *PWAIT,RBDIST,SAT1,SAT2,SLOPA,SLOPD,TAK1,TAK2,
     *TAK3,TAKOLD,TAKTOL,TPLAT,VELCON,VELERR,VELMAX,VELOC,
     *VMC,VMCOUT,WAITMX,X,Y
C*****************************************************************
C*****************************************************************
C FIRST RESET APPROPRIATE COUNTERS
        LREG=0
```

```
C "LREG"= COUNT UP REGISTER, NUMBER OF TIMES ONLY ONE DIG TAK READY TO READ
        LLREG=0
C "LLREG"= COUNT UP REG FOR NUMBER OF TIMES BOTH DIG TAKS NOT READY TO READ
C*****************************************************************
C CHECK IF TACH READY FLAGS ARE SET ON BOTH DIGITAL TAKS
C FIRST CLEAR RDY STATUS FLAGS
600     RDY1=0
        RDY2=0
C NOW INPUT WORDS CONTAINING STATUS
        L=INP(16)
        L2=INP(32)
C READY STATUS APPEARS IN BIT 1; NEXT STATEMENTS DETECT STATUS OF READY
        IF((L.AND.2).EQ.2) RDY1=1
        IF((L2.AND.2).EQ.2) RDY2=1
C ARE BOTH DIG TACHS READY TO BE READ?
        IF((RDY1.AND.RDY2).EQ.1) GO TO 640
C NO? THEN ARE EITHER READY?
        IF((RDY1.OR.RDY2).EQ.1) GO TO 610
        GO TO 630
C NO? THEN GO COUNT HOW MANY TIMES THAT THIS HAS OCCURED THEN TRY AGAIN
C NEXT COUNT HOW MANY TIMES ONLY ONE TACH WAS READY TO BE READ
610     LREG=LREG+1
        IF(LREG.LT.5000) GO TO 600
C IF 5000 IS EXCEEDED THEN ONLY ONE DIG TACH IS WORKING
C SET BAD TAK FLAG FOR PANEL INDICATOR
        TKSLR=3
C SET EMERGENCY STOP FLAG
        EMSTOP=1
        RETURN
C COUNT HOW MANY TIMES NEITHER DIG TACH WAS READY
630     LLREG=LLREG+1
C IF THIS COUNT EXCEEDS 5000 BETTER CALL FOR EMERGENCY STOP
        IF(LLREG.LT.5000) GO TO 600
C GUESS WE CAN'T READ EITHER DIG TACH! BETTER STOP
C SET BAD TAK INDIC FLAG AND EMERGENCY STOP FLAG
        TKSLR=3
        EMSTOP=1
        RETURN
C BOTH DIG TACHS READY TO READ, LETS READ AND COMPARE!
C  BUT FIRST STORE OLD TAK1 VALUE FOR USE IN DETERMINING ACCELERATION
640     TAKOLD=TAK1
C FIRST INPUT LOW AND HIGH ORDER WORDS FROM TAK1 BOARD
        LLOW=INP(17)
        LHIGH=INP(18)
C NEXT, INPUT LOW AND HIGH ORDER WORDS FROM TAK2 BOARD
        L2LOW=INP(33)
        L2HIGH=INP(34)
C TAK BOARDS DON'T KNOW FROM 2'S COMPLEMENT!! RESCALE TO INTEGER
        IF(LLOW.LT.0) LLOW=256+LLOW
        IF(L2LOW.LT.0) L2LOW=256+L2LOW
C NOW COMBINE LOW AND HIGH ORDER WORDS INTO ONE 16 BIT INTEGER
        TAK1=LLOW+256*LHIGH
```

```
      TAK2=L2LOW+256*L2HIGH
C NEXT CHECK STATUS WORD FOR TACH (VEHICLE) DIRECTION
C THEN APPLY PROPER SIGN TO TAK VARIABLES
      IF((L.AND.1).EQ.1) TAK1=-TAK1
      IF((L2.AND.1).EQ.1) TAK2=-TAK2
C NEXT TWO LINES CHECK FOR BOTH DIG TAKS OFF-LINE AND MTR CNTRLR ON
      IF(VMC.LT.50) GO TO 642
      IF((TAK1.OR.TAK2).EQ.0) GO TO 644
C THE FOLLOWING 2 LINES CHECK FOR OUT-OF-LIMITS DIFF BETWEEN DIG TAKS
642   LDIFF=IABS(TAK1-TAK2)
      IF(LDIFF.LT.TAKTOL) GO TO 649
      GO TO 643
644   BADRD=BADRD+2
      GO TO 645
643   BADRD=BADRD+2
C NEXT STATEMENT ALLOWS TAKS TO READ DIFFERENT VALUES FOR 50 TIMES
645   IF(BADRD.LT.50) GO TO 650
C IF BADRD>50 SET BAD TAK AND EMERG STOP FLAGS
641   TKSLR=3
      EMSTOP=1
      RETURN
C CLEAR BADRD COUNTER
649   BADRD=0
650   RETURN
C ANY ANALOG TACH READ ROUTINE CAN BE PUT BELOW. USE ANTAK.FOR
      END
```

```
C****************************************************************
C
C       CTRLV [8].......ESTABLISHES CONTROL VELOCITY
C
C       (REVISED 7/14/80)  (COMMENTS ADDED 9/16/80)
C
C****************************************************************
        SUBROUTINE CTRLV
      LOGICAL BADRD,BRAKES,COUNT,COUNT2,DSTRST,EMSTOP,FASTOP,
     *F1,F1CNT,F1LAST,F2,F2CNT,F2LAST,F3,F3CNT,F3LAST,
     *F4,F4CNT,F4LAST,F5,F5CNT,F5LAST,F6,F6CNT,F6LAST,
     *F7,F7CNT,F7LAST,F8,F8CNT,F8LAST,F9,F9CNT,F9LAST,
     *F10,F10CNT,F10LST,
     *KCNT2,KCOUNT,KONE,KTHREE,KTWO,L,L2,MALTAK,
     *NRMSTP,PLUG,POWER,RBRST,RDY1,RDY2,SLOWDN,
     *TAKRDY,TKSLR,TMDSTP,TRMSTP,VMCRST,WATRST,XCOUNT,YCOUNT
      LOGICAL KBRAKE,KLIVE,KPLUG,KPOWER,LIVE,PLGLST
C****************************************************************
      INTEGER ABSTAK,ABSVEL,ACCEL,ACCMAX,ACCOLD,BRKDLA,DIST,
     *INCACC,INCEM,INCFST,INCNRM,INCR,INCSLO,
     *DISTM,DLA1,INPO,INP1,INP17,INP2,INP3,INP4,
     *INP9,INTA,INTD,K,L2HIGH,L2LOW,LDIFF,LHIGH,
     *LLOW,LSB,MAXRBD,MSB,MSBO,K4CNT,NUMOUT,PLAT1,
     *PWAIT,RBDIST,SAT1,SAT2,SLOPA,SLOPD,TAK1,TAK2,
     *TAK3,TAKOLD,TAKTOL,TPLAT,VELCON,VELERR,VELMAX,VELOC,
     *VMC,VMCOUT,WAITMX,X,Y
C****************************************************************
      COMMON /L/BADRD,BRAKES,COUNT,COUNT2,DSTRST,EMSTOP,FASTOP,
     *F1,F1CNT,F1LAST,F2,F2CNT,F2LAST,F3,F3CNT,F3LAST,
     *F4,F4CNT,F4LAST,F5,F5CNT,F5LAST,F6,F6CNT,F6LAST,
     *F7,F7CNT,F7LAST,F8,F8CNT,F8LAST,F9,F9CNT,F9LAST,
     *F10,F10CNT,F10LST,
     *KCNT2,KCOUNT,KONE,KTHREE,KTWO,L,L2,MALTAK,
     *NRMSTP,PLUG,POWER,RBRST,RDY1,RDY2,SLOWDN,
     *TAKRDY,TKSLR,TMDSTP,TRMSTP,VMCRST,WATRST,XCOUNT,YCOUNT
      COMMON /L1/KBRAKE,KLIVE,KPLUG,KPOWER,LIVE,PLGLST
C****************************************************************
      COMMON /I/ABSTAK,ABSVEL,ACCEL,ACCMAX,ACCOLD,BRKDLA,DIST,
     *INCACC,INCEM,INCFST,INCNRM,INCR,INCSLO,
     *DISTM,DLA1,INPO,INP1,INP17,INP2,INP3,INP4,
     *INP9,INTA,INTD,K,L2HIGH,L2LOW,LDIFF,LHIGH,
     *LLOW,LSB,MAXRBD,MSB,MSBO,K4CNT,NUMOUT,PLAT1,
     *PWAIT,RBDIST,SAT1,SAT2,SLOPA,SLOPD,TAK1,TAK2,
     *TAK3,TAKOLD,TAKTOL,TPLAT,VELCON,VELERR,VELMAX,VELOC,
     *VMC,VMCOUT,WAITMX,X,Y
C****************************************************************
C****************************************************************
C EITHER FASTOP OR NRMSTP MUST BE SET TO COMMAND VELOCITY =0
        IF((FASTOP.OR.NRMSTP).NE.1) GO TO 910
905     VELCON=0
        RETURN
C TRMSTP INDICATES ROADSTOP (MAGNET) MUST MAKE TWO-STAGE MEASURED TYPE
```

```
C STOP
910     IF(TRMSTP.NE.1) GO TO 930
C CONTROL VELOCITY OF 620 = 2 MPH
        VELCON=620
C MAKE SURE STOPPING DISTANCE COUNTER IS ZERO'D AT BEGINNING OF STOP
        IF(DSTRST.EQ.1) GO TO 920
        DSTRST=1
        DIST=0
C MUST SCALE DOWN DIST TO REMAIN BELOW 32,768
C TAK1 UNSCALED = 10,500 CTS/FT
C DISTM IS MAXIMUM DIST ALLOWED PAST MAGNET THEN STOP IS COMMANDED
920     IF(DIST.GE.DISTM) GO TO 905
        DIST=DIST+(TAK1/8)
        RETURN
930     DSTRST=0
        IF(TMDSTP.EQ.1) GO TO 905
        IF(SLOWDN.NE.1) GO TO 940
C SLOWDN=1 FOR PRIM SENS AND TURNING INDIC; 2 MPH= 620
        VELCON=620
        RETURN
C STATEMENTS BELOW USED FOR TWO STAGE ACCEL. FROM STOP
C VEHICLE MUST ATTAIN AT LEAST 1.75 MPH BEFORE PLATEAU TIME PERIOD STARTS
940     IF(TPLAT.GT.PLAT1) GO TO 950
        IF(TAK1.LT.550) GO TO 942
        TPLAT=TPLAT+1
942     VELCON=620
        RETURN
C CONTROL VELOCITY OF 2000 = 6.8 MPH
950     VELCON=2000
        RETURN
        END
```

ORIGINAL PAGE IS
QUALITY

```
C*********************************************************************
C
C      SUBROUTINE FLGSET [5]..SETS FLAGS BASED ON TTL INPUTS
C
C      (REVISED 7/31/80)   (COMMENTS ADDED 9/18/80)
C
C*********************************************************************
        SUBROUTINE FLGSET
      LOGICAL BADRD,BRAKES,COUNT,COUNT2,DSTRST,EMSTOP,FASTOP,
     *F1,F1CNT,F1LAST,F2,F2CNT,F2LAST,F3,F3CNT,F3LAST,
     *F4,F4CNT,F4LAST,F5,F5CNT,F5LAST,F6,F6CNT,F6LAST,
     *F7,F7CNT,F7LAST,F8,F8CNT,F8LAST,F9,F9CNT,F9LAST,
     *F10,F10CNT,F10LST,
     *KCNT2,KCOUNT,KONE,KTHREE,KTWO,L,L2,MALTAK,
     *NRMSTP,PLUG,POWER,RBRST,RDY1,RDY2,SLOWDN,
     *TAKRDY,TKSLR,TMDSTP,TRMSTP,VMCRST,WATRST,XCOUNT,YCOUNT
      LOGICAL KBRAKE,KLIVE,KPLUG,KPOWER,LIVE,PLGLST
C*****************************************************************
      INTEGER ABSTAK,ABSVEL,ACCEL,ACCMAX,ACCOLD,BRKDLA,DIST,
     *INCACC,INCEM,INCFST,INCNRM,INCR,INCSLO,
     *DISTM,DLA1,INPO,INP1,INP17,INP2,INP3,INP4,
     *INP9,INTA,INTD,K,L2HIGH,L2LOW,LDIFF,LHIGH,
     *LLOW,LSB,MAXRBD,MSB,MSBO,K4CNT,NUMOUT,PLAT1,
     *PWAIT,RBDIST,SAT1,SAT2,SLOPA,SLOPD,TAK1,TAK2,
     *TAK3,TAKOLD,TAKTOL,TPLAT,VELCON,VELERR,VELMAX,VELOC,
     *VMC,VMCOUT,WAITMX,X,Y
C*****************************************************************
      COMMON /L/BADRD,BRAKES,COUNT,COUNT2,DSTRST,EMSTOP,FASTOP,
     *F1,F1CNT,F1LAST,F2,F2CNT,F2LAST,F3,F3CNT,F3LAST,
     *F4,F4CNT,F4LAST,F5,F5CNT,F5LAST,F6,F6CNT,F6LAST,
     *F7,F7CNT,F7LAST,F8,F8CNT,F8LAST,F9,F9CNT,F9LAST,
     *F10,F10CNT,F10LST,
     *KCNT2,KCOUNT,KONE,KTHREE,KTWO,L,L2,MALTAK,
     *NRMSTP,PLUG,POWER,RBRST,RDY1,RDY2,SLOWDN,
     *TAKRDY,TKSLR,TMDSTP,TRMSTP,VMCRST,WATRST,XCOUNT,YCOUNT
      COMMON /L1/KBRAKE,KLIVE,KPLUG,KPOWER,LIVE,PLGLST
C*****************************************************************
      COMMON /I/ABSTAK,ABSVEL,ACCEL,ACCMAX,ACCOLD,BRKDLA,DIST,
     *INCACC,INCEM,INCFST,INCNRM,INCR,INCSLO,
     *DISTM,DLA1,INPO,INP1,INP17,INP2,INP3,INP4,
     *INP9,INTA,INTD,K,L2HIGH,L2LOW,LDIFF,LHIGH,
     *LLOW,LSB,MAXRBD,MSB,MSBO,K4CNT,NUMOUT,PLAT1,
     *PWAIT,RBDIST,SAT1,SAT2,SLOPA,SLOPD,TAK1,TAK2,
     *TAK3,TAKOLD,TAKTOL,TPLAT,VELCON,VELERR,VELMAX,VELOC,
     *VMC,VMCOUT,WAITMX,X,Y
C*****************************************************************
C*****************************************************************
C THE FOLLOWING 8 LINES OF CODE TOGGLE ON AND OFF THE SECONDARY
C SENSOR FLAG ONLY AFTER 5 CONSECUTIVE O'S OR 1'S HAVE BEEN DETECTED.
        IF(F4.NE.F4LAST) GO TO 425
        F4CNT=0
        F4LAST=F4
```

```
430     F4=F4LAST
        GO TO 435
425     F4CNT=F4CNT+1
        IF(F4CNT.LT.5) GO TO 430
        F4LAST=F4
C THE FOLLOWING 8 LINES OF CODE TOGGLE ON AND OFF THE EMERG. STOP
C PUSH BUTTON FLAG ONLY AFTER 5 CONSECUTIVE 0'S OR 1'S HAVE BEEN DETECTED.
435     IF(F1.NE.F1LAST) GO TO 470
        F1CNT=0
        F1LAST=F1
472     F1=F1LAST
        GO TO 474
470     F1CNT=F1CNT+1
        IF(F1CNT.LT.5) GO TO 472
        F1LAST=F1
C THE FOLLOWING 8 LINES OF CODE TOGGLE ON AND OFF THE BUMPER SWITCH
C FLAG ONLY AFTER 5 CONSECUTIVE 0'S OR 1'S HAVE BEEN DETECTED.
474     IF(F2.NE.F2LAST) GO TO 480
        F2CNT=0
        F2LAST=F2
482     F2=F2LAST
        GO TO 484
480     F2CNT=F2CNT+1
        IF(F2CNT.LT.5) GO TO 482
        F2LAST=F2
C CALL FOR EMERG. STOP IF PANIC BUTTON OR BUMPER SW. ARE HIT
484     IF((F1.OR.F2).EQ.1) CALL EMSTPR
C THE FOLLOWING 8 LINES OF CODE TOGGLE ON AND OFF THE PRIMARY
C SENSOR FLAG ONLY AFTER 5 CONSECUTIVE 0'S OR 1'S HAVE BEEN DETECTED.
        IF(F3.NE.F3LAST) GO TO 440
        F3CNT=0
        F3LAST=F3
442     F3=F3LAST
        GO TO 444
440     F3CNT=F3CNT+1
        IF(F3CNT.LT.5) GO TO 442
        F3LAST=F3
C THE FOLLOWING 11 LINES OF CODE TOGGLE ON  THE STRG. ANGLE INDIC.
C FLAG ONLY AFTER 5 CONSECUTIVE  1'S HAVE BEEN DETECTED.
C THIS FLAG CAN BE TURNED OFF ONLY AFTER 50 CONSEQUTIVE 0'S ARE DETECTED.
444     IF(F8.NE.F8LAST) GO TO 450
        F8CNT=0
        F8LAST=F8
452     F8=F8LAST
        GO TO 408
450     IF(F8.EQ.0) GO TO 451
        F8CNT=F8CNT+20
        GO TO 453
451     F8CNT=F8CNT+2
453     IF(F8CNT.LT.100) GO TO 452
        F8LAST=F8
C IF PRIM. OR STRG. ANGLE FLAG IS ON GO SET SLOWDN FLAG=1
```

```
408     IF((F3.OR.F8).EQ.1) GO TO 405
        SLOWDN=0
        GO TO 407
405     SLOWDN=1
407     CONTINUE
C THE FOLLOWING 8 LINES OF CODE TOGGLE ON AND OFF THE LOSS OF WIRE
C FLAG ONLY AFTER 5 CONSECUTIVE 0'S OR 1'S HAVE BEEN DETECTED.
        IF(F7.NE.F7LAST) GO TO 460
        F7CNT=0
        F7LAST=F7
462     F7=F7LAST
        GO TO 410
460     F7CNT=F7CNT+1
        IF(F7CNT.LT.5) GO TO 462
        F7LAST=F7
C IF LOSS OF WIRE OR SEC. ARE ON GO SET FASTOP FLAG=1, OTHERWISE CLEAR
C THIS FLAG
410     IF((F7.OR.F4).EQ.1) GO TO 414
        FASTOP=0
        VMCRST=0
        KONE=0
        GO TO 415
C THE FOLLOWING TMDSTP FLAG CAN ONLY BE CLEARED BY COMPLETE STOP
C AND COMPLETION OF WAIT PERIOD.  BOTH TMDSTP &TRMSTP FLAGS
C CLEARED IN SUBR.TSTOP
414     FASTOP=1
C IF BOARDING TAPE SW OR ROAD MARKER ARE ON SET TMDSTP=1
415     IF((F6.OR.F10).NE.1) GO TO 420
C TMDSTP=1 CAUSES VEHICLE TO REMAIN STOPPED FOR PERIOD OF TIME AFTER
C IT HAS COME TO A STOP.
        TMDSTP=1
C TRMSTP=1 CAUSES ACCURATE TWO STAGE MEASURED DECELERATION
C TO CORRECT STREET LOCATION
        IF(F10.EQ.1) TRMSTP=1
420     RETURN
C FLAGS F5 AND F9 HAVE BEEN REMOVED UNTIL CONNECTED ON VEHICLE
        END
```

```
C*********************************************************************
C
C  SUBROUTINE HLDSTL [14]....USED TO HOLD VEHICLE STATIONARY USING MTR
C                         AND TO BRING TO STOP USING MTR DURING EMERG.
C
C  (REVISED 7/31/80) (COMMENTS ADDED 9/18/80)
C
C*********************************************************************
        SUBROUTINE HLDSTL
       LOGICAL BADRD,BRAKES,COUNT,COUNT2,DSTRST,EMSTOP,FASTOP,
      *F1,F1CNT,F1LAST,F2,F2CNT,F2LAST,F3,F3CNT,F3LAST,
      *F4,F4CNT,F4LAST,F5,F5CNT,F5LAST,F6,F6CNT,F6LAST,
      *F7,F7CNT,F7LAST,F8,F8CNT,F8LAST,F9,F9CNT,F9LAST,
      *F10,F10CNT,F10LST,
      *KCNT2,KCOUNT,KONE,KTHREE,KTWO,L,L2,MALTAK,
      *NRMSTP,PLUG,POWER,RBRST,RDY1,RDY2,SLOWDN,
      *TAKRDY,TKSLR,TMDSTP,TRMSTP,VMCRST,WATRST,XCOUNT,YCOUNT
       LOGICAL KBRAKE,KLIVE,KPLUG,KPOWER,LIVE,PLGLST
C*****************************************************************
       INTEGER ABSTAK,ABSVEL,ACCEL,ACCMAX,ACCOLD,BRKDLA,DIST,
      *INCACC,INCEM,INCFST,INCNRM,INCR,INCSLO,
      *DISTM,DLA1,INP0,INP1,INP17,INP2,INP3,INP4,
      *INP9,INTA,INTD,K,L2HIGH,L2LOW,LDIFF,LHIGH,
      *LLOW,LSB,MAXRBD,MSB,MSBO,K4CNT,NUMOUT,PLAT1,
      *PWAIT,RBDIST,SAT1,SAT2,SLOPA,SLOPD,TAK1,TAK2,
      *TAK3,TAKOLD,TAKTOL,TPLAT,VELCON,VELERR,VELMAX,VELOC,
      *VMC,VMCOUT,WAITMX,X,Y
C*****************************************************************
       COMMON /L/BADRD,BRAKES,COUNT,COUNT2,DSTRST,EMSTOP,FASTOP,
      *F1,F1CNT,F1LAST,F2,F2CNT,F2LAST,F3,F3CNT,F3LAST,
      *F4,F4CNT,F4LAST,F5,F5CNT,F5LAST,F6,F6CNT,F6LAST,
      *F7,F7CNT,F7LAST,F8,F8CNT,F8LAST,F9,F9CNT,F9LAST,
      *F10,F10CNT,F10LST,
      *KCNT2,KCOUNT,KONE,KTHREE,KTWO,L,L2,MALTAK,
      *NRMSTP,PLUG,POWER,RBRST,RDY1,RDY2,SLOWDN,
      *TAKRDY,TKSLR,TMDSTP,TRMSTP,VMCRST,WATRST,XCOUNT,YCOUNT
       COMMON /L1/KBRAKE,KLIVE,KPLUG,KPOWER,LIVE,PLGLST
C*****************************************************************
       COMMON /I/ABSTAK,ABSVEL,ACCEL,ACCMAX,ACCOLD,BRKDLA,DIST,
      *INCACC,INCEM,INCFST,INCNRM,INCR,INCSLO,
      *DISTM,DLA1,INP0,INP1,INP17,INP2,INP3,INP4,
      *INP9,INTA,INTD,K,L2HIGH,L2LOW,LDIFF,LHIGH,
      *LLOW,LSB,MAXRBD,MSB,MSBO,K4CNT,NUMOUT,PLAT1,
      *PWAIT,RBDIST,SAT1,SAT2,SLOPA,SLOPD,TAK1,TAK2,
      *TAK3,TAKOLD,TAKTOL,TPLAT,VELCON,VELERR,VELMAX,VELOC,
      *VMC,VMCOUT,WAITMX,X,Y
C*****************************************************************
C*****************************************************************
C THIS SUBR. DOES NOT TRY TO CONTROL ACCEL. OR JERK
C IF STOPPED RETURN
        IF(TAK1.EQ.0) RETURN
C CHECK FOR FORWARD OR REVERSE MOTION
```

```
      IF(TAK1.LT.0) GO TO 710
C IF DIDN'T BRANCH THEN MUST BE FORWARD MOTION
C IF ALREADY PLUGGING, JUST INCREMENT UP M/C VOLTS
      IF(PLUG.EQ.1) GO TO 705
C IF NOT PLUGGING, AND M/C VOLTS ARE HIGH, THEN INCR DOWN
      IF(VMC.GT.1) GO TO 703
C NOTE: PLUG=1 MEANS TO PLUG!!
C VMC IS SMALL ENOUGH TO TURN OFF AND CALL FOR PLUGGING
      VMC=0
      PLUG=1
      RETURN
C REDUCE FORWARD VELOCITY IN NON-PLUG MODE BY REDUCING MTR CNTRL VOLTS
C NO ACCEL. OR JERK FEEDBACK USED.
703   VMC=VMC-INCSLO
C M/C CONTROL VOLTS RANGE IS 0 TO +5 VOLTS
      IF(VMC.LT.0) VMC=0
      RETURN
C REDUCE FORWARD VELOCITY IN PLUGGING MODE BY INCREASE MTR CNTRL VOLTS
C OR..REDUCE REARWARD VELOCITY IN NON PLUGGING MODE.
705   VMC=VMC+INCSLO
C VMC=255 (EQUIVALENT TO 5.0 VOLTS DAC OUTPUT), MAX ALLOWED M/C INPUT
      IF(VMC.GT.255) VMC=255
      RETURN
C REDUCE VEHICLE REARWARD VELOCITY
C REMEMBER PLUG=0 MEANS NOT PLUGGING
C BRANCH TO 710 IMPLIES VEHICLE ROLLING BACKWARDS
C IF NOT PLUGGING JUST INCREASE M/C VOLTS
710   IF(PLUG.EQ.0) GO TO 705
C IF PLUGGING MUST FIRST GET M/C VOLTS BACK TO ~ 0 THEN TURN OFF PLUGGING
      IF(VMC.GT.1) GO TO 703
      VMC=0
      PLUG=0
      RETURN
      END
```

```
C*********************************************************************
C
C  SUBROUTINE RUNWAY [9]....USED TO DETECT "RUNAWAY CONDITIONS
C
C  (REVISED 7/15/80) (COMMENTS ADDED 9/18/80)
C
C*********************************************************************
C THIS SUBROUTINE DETECTS OVERSPEED OR EXCESS REARWARD MOVEMENT
C AND CALLS FOR APPROPRIATE ACTION.
C*********************************************************************
        SUBROUTINE RUNWAY
      LOGICAL BADRD,BRAKES,COUNT,COUNT2,DSTRST,EMSTOP,FASTOP,
     *F1,F1CNT,F1LAST,F2,F2CNT,F2LAST,F3,F3CNT,F3LAST,
     *F4,F4CNT,F4LAST,F5,F5CNT,F5LAST,F6,F6CNT,F6LAST,
     *F7,F7CNT,F7LAST,F8,F8CNT,F8LAST,F9,F9CNT,F9LAST,
     *F10,F10CNT,F10LST,
     *KCNT2,KCOUNT,KONE,KTHREE,KTWO,L,L2,MALTAK,
     *NRMSTP,PLUG,POWER,RBRST,RDY1,RDY2,SLOWDN,
     *TAKRDY,TKSLR,TMDSTP,TRMSTP,VMCRST,WATRST,XCOUNT,YCOUNT
      LOGICAL KBRAKE,KLIVE,KPLUG,KPOWER,LIVE,PLGLST
C****************************************************************
      INTEGER ABSTAK,ABSVEL,ACCEL,ACCMAX,ACCOLD,BRKDLA,DIST,
     *INCACC,INCEM,INCFST,INCNRM,INCR,INCSLO,
     *DISTM,DLA1,INP0,INP1,INP17,INP2,INP3,INP4,
     *INP9,INTA,INTD,K,L2HIGH,L2LOW,LDIFF,LHIGH,
     *LLOW,LSB,MAXRBD,MSB,MSBO,K4CNT,NUMOUT,PLAT1,
     *PWAIT,RBDIST,SAT1,SAT2,SLOPA,SLOPD,TAK1,TAK2,
     *TAK3,TAKOLD,TAKTOL,TPLAT,VELCON,VELERR,VELMAX,VELOC,
     *VMC,VMCOUT,WAITMX,X,Y
C****************************************************************
      COMMON /L/BADRD,BRAKES,COUNT,COUNT2,DSTRST,EMSTOP,FASTOP,
     *F1,F1CNT,F1LAST,F2,F2CNT,F2LAST,F3,F3CNT,F3LAST,
     *F4,F4CNT,F4LAST,F5,F5CNT,F5LAST,F6,F6CNT,F6LAST,
     *F7,F7CNT,F7LAST,F8,F8CNT,F8LAST,F9,F9CNT,F9LAST,
     *F10,F10CNT,F10LST,
     *KCNT2,XCOUNT,KONE,KTHREE,KTWO,L,L2,MALTAK,
     *NRMSTP,PLUG,POWER,RBRST,RDY1,RDY2,SLOWDN,
     *TAKRDY,TKSLR,TMDSTP,TRMSTP,VMCRST,WATRST,XCOUNT,YCOUNT
      COMMON /L1/KBRAKE,KLIVE,KPLUG,KPOWER,LIVE,PLGLST
C****************************************************************
      COMMON /I/ABSTAK,ABSVEL,ACCEL,ACCMAX,ACCOLD,BRKDLA,DIST,
     *INCACC,INCEM,INCFST,INCNRM,INCR,INCSLO,
     *DISTM,DLA1,INP0,INP1,INP17,INP2,INP3,INP4,
     *INP9,INTA,INTD,K,L2HIGH,L2LOW,LDIFF,LHIGH,
     *LLOW,LSB,MAXRBD,MSB,MSBO,K4CNT,NUMOUT,PLAT1,
     *PWAIT,RBDIST,SAT1,SAT2,SLOPA,SLOPD,TAK1,TAK2,
     *TAK3,TAKOLD,TAKTOL,TPLAT,VELCON,VELERR,VELMAX,VELOC,
     *VMC,VMCOUT,WAITMX,X,Y
C****************************************************************
C****************************************************************
C FIRST CHECK FOR BACKWARDS VELOCITY
C THEN, CHECK ROLLBACK DISTANCE
```

```
        IF(TAK1.GE.0) GO TO 1050
C RBRST IS A REGISTER RESET FLAG FOR RBDIST REGISTER
        IF(RBRST.EQ.1) GO TO 1020
        RBRST=1
        RBDIST=0
C TAK1 DIVIDED BY 8 BECAUSE OF ACTUAL 10,500 CTS/FT. MAXRBD IS ALSO SCALED BY 8
C SUM SCALED TAK COUNTS FOR TOTAL ROLLED BACK DIST.
1020    RBDIST=RBDIST-(TAK1/8)
        IF(RBDIST.GT.MAXRBD) GO TO 1060
        RETURN
1050    RBRST=0
C BOTH DIGITAL TAKS MUST BE OUT OF SAFE SPEED RANGE FOR EMERG. STOP
        IF((TAK1.LT.VELMAX).AND.(TAK1.GT.-VELMAX)) GO TO 1065
C NEXT STATEMENT TEMPORARILY DISCARDED UNTIL BIG TACHRD ROUTINE IS UP
        IF((TAK2.LT.VELMAX).AND.(TAK2.GT.-VELMAX)) GO TO 1065
1060    CALL EMSTPR
        EMSTOP=1
1065    RETURN
        END
```

```
C*****************************************************************************
C
C   SUBROUTINE TSTOP [12].....TIMED STOP ROUTINE USED TO TIME STOP PERIOD,
C                        APPLY HYDRAULIC BRAKES, AND USE MTR TO HOLD
C                        VEHICLE STATIONARY IN CASE OF BRAKE FAILURE
C
C   (REVISED 7/31/80) (COMMENTS ADDED 9/18/80)
C
C*****************************************************************************
        SUBROUTINE TSTOP
      LOGICAL BADRD,BRAKES,COUNT,COUNT2,DSTRST,EMSTOP,FASTOP,
     *F1,F1CNT,F1LAST,F2,F2CNT,F2LAST,F3,F3CNT,F3LAST,
     *F4,F4CNT,F4LAST,F5,F5CNT,F5LAST,F6,F6CNT,F6LAST,
     *F7,F7CNT,F7LAST,F8,F8CNT,F8LAST,F9,F9CNT,F9LAST,
     *F10,F10CNT,F10LST,
     *KCNT2,KCOUNT,KONE,KTHREE,KTWO,L,L2,MALTAK,
     *NRMSTP,PLUG,POWER,RBRST,RDY1,RDY2,SLOWDN,
     *TAKRDY,TKSLR,TMDSTP,TRMSTP,VMCRST,WATRST,XCOUNT,YCOUNT
      LOGICAL KBRAKE,KLIVE,KPLUG,KPOWER,LIVE,PLGLST
C******************************************************************
      INTEGER ABSTAK,ABSVEL,ACCEL,ACCMAX,ACCOLD,BRKDLA,DIST,
     *INCACC,INCEM,INCFST,INCNRM,INCR,INCSLO,
     *DISTM,DLA1,INP0,INP1,INP17,INP2,INP3,INP4,
     *INP9,INTA,INTD,K,L2HIGH,L2LOW,LDIFF,LHIGH,
     *LLOW,LSB,MAXRBD,MSB,MSBO,K4CNT,NUMOUT,PLAT1,
     *PWAIT,RBDIST,SAT1,SAT2,SLOPA,SLOPD,TAK1,TAK2,
     *TAK3,TAKOLD,TAKTOL,TPLAT,VELCON,VELERR,VELMAX,VELOC,
     *VMC,VMCOUT,WAITMX,X,Y
C******************************************************************
      COMMON /L/BADRD,BRAKES,COUNT,COUNT2,DSTRST,EMSTOP,FASTOP,
     *F1,F1CNT,F1LAST,F2,F2CNT,F2LAST,F3,F3CNT,F3LAST,
     *F4,F4CNT,F4LAST,F5,F5CNT,F5LAST,F6,F6CNT,F6LAST,
     *F7,F7CNT,F7LAST,F8,F8CNT,F8LAST,F9,F9CNT,F9LAST,
     *F10,F10CNT,F10LST,
     *KCNT2,KCOUNT,KONE,KTHREE,KTWO,L,L2,MALTAK,
     *NRMSTP,PLUG,POWER,RBRST,RDY1,RDY2,SLOWDN,
     *TAKRDY,TKSLR,TMDSTP,TRMSTP,VMCRST,WATRST,XCOUNT,YCOUNT
      COMMON /L1/KBRAKE,KLIVE,KPLUG,KPOWER,LIVE,PLGLST
C******************************************************************
      COMMON /I/ABSTAK,ABSVEL,ACCEL,ACCMAX,ACCOLD,BRKDLA,DIST,
     *INCACC,INCEM,INCFST,INCNRM,INCR,INCSLO,
     *DISTM,DLA1,INP0,INP1,INP17,INP2,INP3,INP4,
     *INP9,INTA,INTD,K,L2HIGH,L2LOW,LDIFF,LHIGH,
     *LLOW,LSB,MAXRBD,MSB,MSBO,K4CNT,NUMOUT,PLAT1,
     *PWAIT,RBDIST,SAT1,SAT2,SLOPA,SLOPD,TAK1,TAK2,
     *TAK3,TAKOLD,TAKTOL,TPLAT,VELCON,VELERR,VELMAX,VELOC,
     *VMC,VMCOUT,WAITMX,X,Y
C******************************************************************
C******************************************************************
C VEHICLE SHOULD BE STOPPED BEFORE ENTERING THIS ROUTINE
C FIRST, APPLY HYDRAULIC BRAKES (KEEP A STOPPED VEHICLE STOPPED!)
C BRAKES=1>> OPEN HYDRAULIC VALVE; BRAKES=2>> TURN ON BRAKE LT;
```

```
C BRAKES=4>> BRAKE LT INDIC LIGHT; BRAKES=8>> EM STOP INDIC LT;
C BRAKES=16>> HORN ON;:::: NUMBERS ADD FOR VARIOUS COMBINATIONS
C OF FUNCTIONS
        IF(BRAKES.EQ.23) GO TO 1305
        BRAKES=7
1305    CALL OUT(5,BRAKES)
C MAKE SURE VEHICLE IS NOT MOVING APPRECIABLY +/- X IS VELOCITY TOLERANCE;
C IF IT'S MOVING THEN USE MOTOR TO STOP
        IF((TAK1.LT.X).AND.(TAK1.GT.-X)) GO TO 1320
        CALL HLDSTL
1307    KTWO=3
C KTWO=3 INSURES RETURN TO THIS SUBR FROM MAIN
        RETURN
C REMAINDER OF SUBR. USED FOR PASSENGER STOP TIMER ETC.
C CAN NOT PASS THIS POINT UNLESS VEHICLE IS STOPPED!
C RESET BRAKE DELAY AND PLATEAU TIME COUNTERS FOR NEXT ACCEL.
1320    BRKDLA=0
        TPLAT=0
        IF(TMDSTP.NE.1) GO TO 1322
C NEXT STATEMENTS INSURE TIMED STOP COUNTER REMAINS SET TO 0 SO
C LONG AS SOMEONE IS HOLDING TAPE SW CLOSED
        TMDSTP=0
        PWAIT=0
        WATRST=1
        GO TO 1307
1322    IF(WATRST.EQ.1) GO TO 1330
C F4= SECOND SENSOR; DONT TAKE OFF UNTIL THIS IS CLEARED FOR A WHILE
1321    IF(F4.EQ.0) GO TO 1360
C K4CNT VARIABLE IS USED FOR HORN TOOT TIMING
        K4CNT=0
        GO TO 1307
1360    K4CNT=K4CNT+1
        IF(K4CNT.LT.5) GO TO 1362
C STATEMENT BELOW USED TO CONTROL HORN TOOTS
        IF((K4CNT.GT.30).AND.(K4CNT.LT.35)) GO TO 1362
        BRAKES=7
        GO TO 1364
1362    BRAKES=23
1364    IF(K4CNT.LT.50) GO TO 1307
1350    GO TO 1340
C PWAIT IS BASIC PASSENGER STOP TIMER
1330    PWAIT=PWAIT+1
        IF(PWAIT.GT.WAITMX) GO TO 1321
        GO TO 1307
C RESET ALL NECESSARY FLAGS AND REGISTERS BELOW
1340    WATRST=0
        K4CNT=0
        PWAIT=0
        TMDSTP=0
        TRMSTP=0
        NRMSTP=0
        FASTOP=0
```

```
VMCRST=0
KONE=0
KTWO=0
VMC=0
PLUG=0
DSTRST=0
RETURN
END
```

```
C**********************************************************************
C
C  SUBROUTINE ACCELR [10]....USED TO CONTROL ACCELERATION
C
C  (REVISED 7/31/80) (ADDEL COMMENTS 9/18/80)
C
C**********************************************************************
        SUBROUTINE ACCELR
       LOGICAL BADRD,BRAKES,COUNT,COUNT2,DSTRST,EMSTOP,FASTOP,
     *F1,F1CNT,F1LAST,F2,F2CNT,F2LAST,F3,F3CNT,F3LAST,
     *F4,F4CNT,F4LAST,F5,F5CNT,F5LAST,F6,F6CNT,F6LAST,
     *F7,F7CNT,F7LAST,F8,F8CNT,F8LAST,F9,F9CNT,F9LAST,
     *F10,F10CNT,F10LST,
     *KCNT2,KCOUNT,KONE,KTHREE,KTWO,L,L2,MALTAK,
     *NRMSTP,PLUG,POWER,RBRST,RDY1,RDY2,SLOWDN,
     *TAKRDY,TKSLR,TMDSTP,TRMSTP,VMCRST,WATRST,XCOUNT,YCOUNT
       LOGICAL KBRAKE,KLIVE,KPLUG,KPOWER,LIVE,PLGLST
C*****************************************************************
       INTEGER ABSTAK,ABSVEL,ACCEL,ACCMAX,ACCOLD,BRKDLA,DIST,
     *INCACC,INCEM,INCFST,INCNRM,INCR,INCSLO,
     *DISTM,DLA1,INP0,INP1,INP17,INP2,INP3,INP4,
     *INP9,INTA,INTD,K,L2HIGH,L2LOW,LDIFF,LHIGH,
     *LLOW,LSB,MAXRBD,MSB,MSBO,K4CNT,NUMOUT,PLAT1,
     *PWAIT,RBDIST,SAT1,SAT2,SLOPA,SLOPD,TAK1,TAK2,
     *TAK3,TAKOLD,TAKTOL,TPLAT,VELCON,VELERR,VELMAX,VELOC,
     *VMC,VMCOUT,WAITMX,X,Y
C*****************************************************************
       COMMON /L/BADRD,BRAKES,COUNT,COUNT2,DSTRST,EMSTOP,FASTOP,
     *F1,F1CNT,F1LAST,F2,F2CNT,F2LAST,F3,F3CNT,F3LAST,
     *F4,F4CNT,F4LAST,F5,F5CNT,F5LAST,F6,F6CNT,F6LAST,
     *F7,F7CNT,F7LAST,F8,F8CNT,F8LAST,F9,F9CNT,F9LAST,
     *F10,F10CNT,F10LST,
     *KCNT2,KCOUNT,KONE,KTHREE,KTWO,L,L2,MALTAK,
     *NRMSTP,PLUG,POWER,RBRST,RDY1,RDY2,SLOWDN,
     *TAKRDY,TKSLR,TMDSTP,TRMSTP,VMCRST,WATRST,XCOUNT,YCOUNT
       COMMON /L1/KBRAKE,KLIVE,KPLUG,KPOWER,LIVE,PLGLST
C*****************************************************************
       COMMON /I/ABSTAK,ABSVEL,ACCEL,ACCMAX,ACCOLD,BRKDLA,DIST,
     *INCACC,INCEM,INCFST,INCNRM,INCR,INCSLO,
     *DISTM,DLA1,INP0,INP1,INP17,INP2,INP3,INP4,
     *INP9,INTA,INTD,K,L2HIGH,L2LOW,LDIFF,LHIGH,
     *LLOW,LSB,MAXRBD,MSB,MSBO,K4CNT,NUMOUT,PLAT1,
     *PWAIT,RBDIST,SAT1,SAT2,SLOPA,SLOPD,TAK1,TAK2,
     *TAK3,TAKOLD,TAKTOL,TPLAT,VELCON,VELERR,VELMAX,VELOC,
     *VMC,VMCOUT,WAITMX,X,Y
C*****************************************************************
C*****************************************************************
C NOTE: PLUG=0 MEANS NOT PLUGGING!!
        IF(PLUG.EQ.0) GO TO 1110
C IF PLUGGING AT GREATER THAN (36/255)*5.0 VOLTS THEN MUST FIRST DECREMENT
C OTHERWISE JUST BACK OFF M/C VOLTS TO 0 AND STOP PLUGGING
        IF(VMC.GT.36) GO TO 1150
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
        VMC=0
        PLUG=0
        RETURN
C FIND ACCMAX ON LINEAR CURVE CORRESPONDING TO VELOC ERROR
1110    ACCMAX=INTA+VELERR*SLOPA
C MAKE SURE ACCMAX IS NOT ABOVE MAX. ALLOWED  LEVEL
        IF(ACCMAX.LE.SAT1) GO TO 1120
        ACCMAX=SAT1
C CHECK THAT ACTUAL ACCELERATION IS LESS THAN ACCMAX
1120    IF(ACCEL.GT.ACCMAX) GO TO 1130
C IF ACCEL HAS NOT EXCEEDED ACCMAX THEN CONTINUE TO INCREASE M/C VOLTS
C OTHERWISE GO TO 1130 AND DECREASE M/C VOLTS
        VMC=VMC+INCACC
C VMC RANGE MUST REMAIN 0-255 (0 TO 5.0 VOLTS)
        IF(VMC.GT.255) VMC=255
C STATEMENTS BELOW USED TO HOLD HYDR. BRAKES ON FOR A LITTLE WHILE
C WHEN STARTING UP TO PREVENT ROLLBACK ON HILLS
        IF(BRKDLA.GT.DLA1) GO TO 1140
        BRKDLA=BRKDLA+1
        RETURN
C BRAKES=0 > RELEASE BRAKES
1140    BRAKES=0
        RETURN
1130    VMC=VMC-INCACC
        GO TO 1151
C NOTE: MUST RETURN VMC TO 0 IF YOU ARE CURRENTLY PLUGGING BUT WANT
C       TO ACCELERATE.
C 1150 USED TO BACK OFF M/C VOLTS FAST IN ORDER TO STOP PLUGGING AND
C BEGIN ACCELERATION
1150    VMC=VMC-INCFST
C VMC RANGE MUST REMAIN 0-255 (0 TO 5.0 VOLTS)
1151    IF(VMC.LT.0) VMC=0
        RETURN
        END
```

```
C*****************************************************************************
C
C   SUBROUTINE DECEL [11].....USED TO CONTROL ACTIVE (PLUGING) AND PASSIVE
C                           DECELERATION.
C
C   (REVISED 8/20/80) (ADDED COMMENTS 9/18/80)
C
C*****************************************************************************
        SUBROUTINE DECEL
       LOGICAL BADRD,BRAKES,COUNT,COUNT2,DSTRST,EMSTOP,FASTOP,
      *F1,F1CNT,F1LAST,F2,F2CNT,F2LAST,F3,F3CNT,F3LAST,
      *F4,F4CNT,F4LAST,F5,F5CNT,F5LAST,F6,F6CNT,F6LAST,
      *F7,F7CNT,F7LAST,F8,F8CNT,F8LAST,F9,F9CNT,F9LAST,
      *F10,F10CNT,F10LST,
      *KCNT2,KCOUNT,KONE,KTHREE,KTWO,L,L2,MALTAK,
      *NRMSTP,PLUG,POWER,RBRST,RDY1,RDY2,SLOWDN,
      *TAKRDY,TKSLR,TMDSTP,TRMSTP,VMCRST,WATRST,XCOUNT,YCOUNT
       LOGICAL KBRAKE,KLIVE,KPLUG,KPOWER,LIVE,PLGLST
C***********************************************************************
       INTEGER ABSTAK,ABSVEL,ACCEL,ACCMAX,ACCOLD,BRKDLA,DIST,
      *INCACC,INCEM,INCFST,INCNRM,INCR,INCSLO,
      *DISTM,DLA1,INPO,INP1,INP17,INP2,INP3,INP4,
      *INP9,INTA,INTD,K,L2HIGH,L2LOW,LDIFF,LHIGH,
      *LLOW,LSB,MAXRBD,MSB,MSBO,K4CNT,NUMOUT,PLAT1,
      *PWAIT,RBDIST,SAT1,SAT2,SLOPA,SLOPD,TAK1,TAK2,
      *TAK3,TAKOLD,TAKTOL,TPLAT,VELCON,VELERR,VELMAX,VELOC,
      *VMC,VMCOUT,WAITMX,X,Y
C***********************************************************************
       COMMON /L/BADRD,BRAKES,COUNT,COUNT2,DSTRST,EMSTOP,FASTOP,
      *F1,F1CNT,F1LAST,F2,F2CNT,F2LAST,F3,F3CNT,F3LAST,
      *F4,F4CNT,F4LAST,F5,F5CNT,F5LAST,F6,F6CNT,F6LAST,
      *F7,F7CNT,F7LAST,F8,F8CNT,F8LAST,F9,F9CNT,F9LAST,
      *F10,F10CNT,F10LST,
      *KCNT2,KCOUNT,KONE,KTHREE,KTWO,L,L2,MALTAK,
      *NRMSTP,PLUG,POWER,RBRST,RDY1,RDY2,SLOWDN,
      *TAKRDY,TKSLR,TMDSTP,TRMSTP,VMCRST,WATRST,XCOUNT,YCOUNT
       COMMON /L1/KBRAKE,KLIVE,KPLUG,KPOWER,LIVE,PLGLST
C***********************************************************************
       COMMON /I/ABSTAK,ABSVEL,ACCEL,ACCMAX,ACCOLD,BRKDLA,DIST,
      *INCACC,INCEM,INCFST,INCNRM,INCR,INCSLO,
      *DISTM,DLA1,INPO,INP1,INP17,INP2,INP3,INP4,
      *INP9,INTA,INTD,K,L2HIGH,L2LOW,LDIFF,LHIGH,
      *LLOW,LSB,MAXRBD,MSB,MSBO,K4CNT,NUMOUT,PLAT1,
      *PWAIT,RBDIST,SAT1,SAT2,SLOPA,SLOPD,TAK1,TAK2,
      *TAK3,TAKOLD,TAKTOL,TPLAT,VELCON,VELERR,VELMAX,VELOC,
      *VMC,VMCOUT,WAITMX,X,Y
C***********************************************************************
C***********************************************************************
C NOTE THAT FASTOPS AT LOW SPEEDS ARE HANDLED DIFFERENTLY THAN AT HIGH SPEEDS
C IF FASTOP CALLED FOR AND VEHICLE IS GOING "FAST" GO TO 1220
        IF((TAK1.LT.900).AND.(FASTOP.EQ.1)) GO TO 1220
C IF FASTOP CALLED FOR AND VEHICLE IS TRAVELLING "SLOW GO TO 1200
```

```
      IF(FASTOP.EQ.1) GO TO 1200
C VMCRST AND KONE INSURE THAT ONCE SPECIAL FASTOP DECEL ROUTINE HAS
C BEEN INVOKED IT IS FOLLOWED THROUGH AS LONG AS FASTOP FLAG=1
      KONE=0
      IF((TMDSTP.OR.SLOWDN).NE.1) GO TO 1203
C IF NOT MAKING TMDSTOP OR SLOWDN THEN MUST BEGIN PLUGGING ASAP IF
C VEHICLE SPEED>3.75 MPH
      IF(TAK1.LT.1160) GO TO 1203
      GO TO 1221
C NEXT TWO STATEMENTS MAKE SURE THAT IN THE CASE OF FASTOP UNDERWAY
C WE DONT RESET VMC=0 AGAIN
1220  IF(VMCRST.EQ.1) GO TO 1201
1221  IF(KONE.EQ.1) GO TO 1203
C NEXT FOUR STATEMENTS GET M/C INTO PLUG MODE ASAP IF NOT ALREADY THERE
      IF(PLUG.EQ.1) GO TO 1205
      VMC=0
      KONE=1
      PLUG=1
      RETURN
C LINES OF CODE BELOW SET UP DECEL FOR FAST STOP FROM HIGH SPEEDS
C FOR DECEL TO 0 SPEED WITHIN 6 FT. DECEL=(TAK1/22)**2
1200  IF(VMCRST.EQ.1) GO TO 1201
      VMCRST=1
      IF(PLUG.NE.1) VMC=0
      PLUG=1
      ACCMAX=TAK1/22
      ACCMAX=-ACCMAX*ACCMAX
      RETURN
C NEXT THREE LINES USED FOR FAST STOP FROM HIGH SPEED ONLY
C DONT LET DECEL EXCEED LIMIT
1201  IF(ACCEL.LT.ACCMAX) GO TO 1214
C NEXT LINE REMOVES  MOTOR CONTROLLER DEAD-BAND. IN PLUGGING MODE
      IF(VMC.LT.50) VMC=50
      VMC=VMC+INCFST
      GO TO 1208
1203  IF(PLUG.EQ.1) GO TO 1205
C NEXT THREE LINES TRANSITION FROM NON-PLUG TO PLUG FOR LOW VMC
      IF(VMC.GT.10) GO TO 1210
C NOTE: PLUG=1 MEANS PLUGGING!!
      VMC=0
      PLUG=1
      RETURN
C LINES BETWEEN 1205 AND 1232 SET UP INCR AND MAX DECEL DIFFERENTLY
C DEPENDING UPON SENSOR INITIATED STOP OR OTHER. ALSO, IF SMALL VELOC
C ERROR THEN SMALLER INCR IS USED.  SAT2= MAXIMUM ALLOWED DECEL VALUE
C (5376=.16 G)
1205  IF((F3.OR.FASTOP).EQ.1) GO TO 1230
      INCR=INCACC
      IF(VELERR.LT.100) INCR=1
      SAT2=-3000
      GO TO 1232
1230  INCR=INCNRM
```

```
        SAT2=-5376
C 1232 CALCULATES MAX ALLOWED DECEL CORRESPONDING TO VELOCITY ERROR
1232    ACCMAX=INTD+VELERR*SLOPD
C NEXT STATEMENT MAKES SURE THAT THIS DECEL DOES NOT EXCEED SOME MAX FIXED #
        IF(ACCMAX.LT.SAT2) ACCMAX=SAT2
C KEEP ACCMAX (MAGNITUDE) BELOW A SATURATION LEVEL
C NOTE: ACCMAX AND JRKMAX ARE NEGATIVE NUMBERS
C ACCEL<ACCMAX IMPLIES DECEL IS TOO GREAT
        IF(ACCEL.LT.ACCMAX) GO TO 1215
C STATEMENTS BETWEEN 1207 AND 1240 ADJUST PLUGING DEAD-BAND ACCORDING TO VELOC
1207    IF(TAK1.GT.700) GO TO 1240
C ELIMINATES MTR CNTRLR DEAD BAND DURING PLUGGING AT LOW SPEEDS
        IF(VMC.LT.32) VMC=32
        GO TO 1209
C NEXT STATEMENT ELIMINATES DEAD BAND DURING PLUGGING AT HIGH SPEEDS
1240    IF(VMC.LT.40) VMC=40
1209    VMC=VMC+INCR
C VMC MUST REMAIN IN RANGE 0-255 (0 TO 5.0 VOLTS)
1208    IF(VMC.GT.255) VMC=255
        GO TO 1260
C NOTE: ACCMAX IS A  NEGATIVE NUMBER
1210    INCR=INCNRM
C CALCULATE MAX ALLOWED DECEL ASSOCIATED WITH PRESENT VELOCITY ERROR
        ACCMAX=INTD+VELERR*SLOPD
        SAT2=-3000
        IF(ACCMAX.LT.SAT2) ACCMAX=SAT2
        IF(ACCEL.LT.ACCMAX) GO TO 1209
1214    INCR=INCFST
1215    VMC=VMC-INCR
        IF(VMC.LT.0) VMC=0
C 1260 CHECKS IF STOPPED ENOUGH TO GO TO TSTOP SUBR AND USE HYDR BRAKES
1260    IF((VELCON.EQ.0).AND.(TAK1.LT.30)) GO TO 1262
        RETURN
C IF GOING SLOW ENOUGH, TURN OFF M/C VOLTS AND DONT RETURN HERE
1262    VMC=0
        KTWO=3
C KTWO=3 > VEHICLE HAS STOPPED, BRANCH TO SUBR TSTOP VIA MAIN
        RETURN
        END
```

```
C*****************************************************************
C*
C*  SUBROUTINE DATAIO [3]..USED TO INPUT/OUTPUT TTL, A/D, D/A DATA
C*
C*  (REVISED 7/14/80) (COMMENTS ADDED 9/18/80)
C*
C*****************************************************************
        SUBROUTINE DATAIO
      LOGICAL BADRD,BRAKES,COUNT,COUNT2,DSTRST,EMSTOP,FASTOP,
     *F1,F1CNT,F1LAST,F2,F2CNT,F2LAST,F3,F3CNT,F3LAST,
     *F4,F4CNT,F4LAST,F5,F5CNT,F5LAST,F6,F6CNT,F6LAST,
     *F7,F7CNT,F7LAST,F8,F8CNT,F8LAST,F9,F9CNT,F9LAST,
     *F10,F10CNT,F10LST,
     *KCNT2,KCOUNT,KONE,KTHREE,KTWO,L,L2,MALTAK,
     *NRMSTP,PLUG,POWER,RBRST,RDY1,RDY2,SLOWDN,
     *TAKRDY,TKSLR,TMDSTP,TRMSTP,VMCRST,WATRST,XCOUNT,YCOUNT
      LOGICAL KBRAKE,KLIVE,KPLUG,KPOWER,LIVE,PLGLST
C*****************************************************************
      INTEGER ABSTAK,ABSVEL,ACCEL,ACCMAX,ACCOLD,BRKDLA,DIST,
     *INCACC,INCEM,INCFST,INCNRM,INCR,INCSLO,
     *DISTM,DLA1,INP0,INP1,INP17,INP2,INP3,INP4,
     *INP9,INTA,INTD,K,L2HIGH,L2LOW,LDIFF,LHIGH,
     *LLOW,LSB,MAXRBD,MSB,MSBO,K4CNT,NUMOUT,PLAT1,
     *PWAIT,RBDIST,SAT1,SAT2,SLOPA,SLOPD,TAK1,TAK2,
     *TAK3,TAKOLD,TAKTOL,TPLAT,VELCON,VELERR,VELMAX,VELOC,
     *VMC,VMCOUT,WAITMX,X,Y
C*****************************************************************
      COMMON /L/BADRD,BRAKES,COUNT,COUNT2,DSTRST,EMSTOP,FASTOP,
     *F1,F1CNT,F1LAST,F2,F2CNT,F2LAST,F3,F3CNT,F3LAST,
     *F4,F4CNT,F4LAST,F5,F5CNT,F5LAST,F6,F6CNT,F6LAST,
     *F7,F7CNT,F7LAST,F8,F8CNT,F8LAST,F9,F9CNT,F9LAST,
     *F10,F10CNT,F10LST,
     *KCNT2,KCOUNT,KONE,KTHREE,KTWO,L,L2,MALTAK,
     *NRMSTP,PLUG,POWER,RBRST,RDY1,RDY2,SLOWDN,
     *TAKRDY,TKSLR,TMDSTP,TRMSTP,VMCRST,WATRST,XCOUNT,YCOUNT
      COMMON /L1/KBRAKE,KLIVE,KPLUG,KPOWER,LIVE,PLGLST
C*****************************************************************
      COMMON /I/ABSTAK,ABSVEL,ACCEL,ACCMAX,ACCOLDCR,INCSLO,
     *DISTM,DLA1,INP0,INP1,INP17,INP2,INP3,INP4,
     *INP9,INTA,INTD,K,L2HIGH,L2LOW,LDIFF,LHIGH,
     *LLOW,LSB,MINTD,K,L2HIGH,L2LOW,LDIFF,LHIGH,
     *LLOW,LSB,MAXRBD,MSB,MSBO,K4CNT,NUMOUT,PLAT1,
     *PWAIT,RBDIST,SAT1,SAT2,SLOPA,SLOPD,TAK1,TAK2,
     *TAK3,TAKOLD,TAKTOL,TPLAT,VELCON,VELERR,VELMAX,VELOC,
     *VMC,VMCOUT,WAITMX,X,Y
C*****************************************************************
C*****************************************************************
C NEXT CALL BEGINS "LIVE" PULSE OUT TO PORT 6
        LIVE=1
        CALL OUT(6 ,LIVE)
C TEMPORARILY STORE EXISTING VALUES OF PLUG, POWER, AND BRAKES
        KPLUG=PLUG
      KPOWER=POWER
      KBRAKE=BRAKES
C NOW TOGGLE ALL "F" FLAGS OFF
      F1=0
      F2=0
      F3=0
      F4=0
      F5=0
      F6=0
      F7=0
      F8=0
      F9=0
      F10=0
C NEXT INPUT PORT  BYTES
      INP0=INP(0)
      INP1=INP(1)
      INP2=INP(2)
      INP3=INP(3)
C IF BIT 0 IN PORT 0 =1, EMERG STOP PB IS ON, SET FLAG F1
      IF((INP0.AND.1).EQ.1) F1=1
C IF BIT 2 IN PORT 0 =1 BRDG TAPE SW IS ON, SET FLAG F6
      IF((INP0.AND.4).EQ.4) F6=1
C IF BIT 4 IN PORT 0 =1, LOSS OF WIRE INDIC.,SET FLAG F7
      IF((INP0.AND.16).EQ.16) F7=1
C IF BIT 0 IN PORT 1 =1, EM ST BMPR SW IS ON, SET FLAG F2
      IF((INP1.AND.1).EQ.1) F2=1
C IF BIT 2 IN PORT 1 =1, STRG ANGLE INDIC IS ON, SET FLAG F8
      IF((INP1.AND.4).EQ.4) F8=1
C IF BIT 4 IN PORT 1 =1, ROAD MRKR INDIC. IS ON, SET FLAG F10
      IF((INP1.AND.16).EQ.16) F10=1
C IF BIT 0 IN PORT 2 =1, SEC SENSOR IS ON, SET FLAG F4
      IF((INP2.AND.1).EQ.1) F4=1
C IF BIT 2 IN PORT 2 =1, LFT TURN SENSOR IS ON, SET FLAG F5
      IF((INP2.AND.4).EQ.4) F5=1
C IF BIT 0 IN PORT 3 =1, PRIMARY SENSOR IS ON, SET FLAG F3
      IF((INP3.AND.1).EQ.1) F3=1
C IF BIT 2 IN PORT 3 =1, HYDR. SYS. LOW PRESS SW IS ON, SET FLAG F9
      IF((INP3.AND.4).EQ.4) F9=1
C PLUG IS OUTPUT BYTE USED TO LIGHT UP PANEL INDIC LTS; BELOW
C IF STATEMENTS TURNJON APPROPRIATE LIGHTS
      IF(F1.EQ.1) PLUG=PLUG+2
      IF(F6.EQ.1) PLUG=PLUG+4
      IF(F7.EQ.1) PLUG=PLUG+8
      IF(F2.EQ.1) PLUG=PLUG+16
      IF(F8.EQ.1) PLUG=PLUG+32
      IF(F10.EQ.1) PLUG=PLUG+64
      IF(F4.EQ.1) PLUG=PLUG+128
C HER LIVE OUTPUT BYTE IS USED TO TURN ON INDIC LT IF SEC SENS IS ON
      IF(F4.EQ.1) LIVE=3
C STATEMENT FROM HERE TO 310 TURN ON OR OFF: BRAKE VALVE, STOP LT, STOP LT
C INDIC., OR HORN DEPENDING UPON NEED. (SEE COMMENTS AT BEGINNIGN OF SUBR
C TSTOP FOR "BRAKES" BREAKDOWN)
```

```
         IF((KBRAKE.EQ.7).OR.(KBRAKE.EQ.23)) GO TO 310
         IF((NRMSTP.OR.EMSTOP.OR.FASTOP.OR.SLOWDN).EQ.1) BRAKES=6
         IF(TMDSTP.EQ.1) BRAKES=6
C NEXT CALL USED TO CONTROL HYDRAULIC BRAKES  (VALVE)
310      CALL OUT(5,BRAKES)
C NEXT CALL CONTROLS MAIN POWER RELAY AND TURNS ON PANEL INDIC LT FOR PRI SENS
         IF(F3.EQ.1) POWER=POWER+2
         CALL OUT(7,POWER)
C NEXT EXPRESSION CAN BE USED TO SCALE MTR CNTRL VOLTAGE PRIOR TO OUTPUT
         VMCOUT=VMC
C BELOW, ROUTINE MAKES SURE THAT VMCOUT=0 BEFORE PLUG CHANGES VALUE!!!
         IF(KPLUG.NE.PLGLST) GO TO 320
C NEXT CALL USED TO PLACE MTR CNTRL VOLT IN RAM FOR DAC
         CALL POKE(-1,VMCOUT)
C NEXT CALL TO PORT 5 USED TO CONTROL MTR CNTRLR FOR/REV
         CALL OUT(4,PLUG)
         GO TO 330
320      VMCOUT=0
         CALL POKE(-1,VMCOUT)
C NEXT CALL ENDS "LIVE" PULSE AT OUTPUT PORT 7
330      LIVE=0
         CALL OUT(6,LIVE)
C NOW THAT PLUG,POWER,BRAKE HAVE BEEN PROSTITUTED TO OUTPUT INDIC LT
C INFORMATION, CHANGE THEM BACK TO VALUES STORED AT ENTRY TO THIS SUBR
         PLUG=KPLUG
         POWER=KPOWER
         BRAKES=KBRAKE
         PLGLST=PLUG
         RETURN
         END
```

C-2

```
C***********************************************************************
C
C        SUBROUTINE EMSTPR [15]..USED FOR EMERGENCY STOPS
C           IMPORTANT:THIS SUBR DOES NOT RETURN, IT IS A DEAD END!!
C        (REVISED 7/14/80) (COMMENTS ADDED 9/18/80)
C
C**********************************************************************
C THIS SUBROUTINE AS SPECIFIED BY AL JOHNSTON
       SUBROUTINE EMSTPR
      LOGICAL BADRD,BRAKES,COUNT,COUNT2,DSTRST,EMSTOP,FASTOP,
     *F1,F1CNT,F1LAST,F2,F2CNT,F2LAST,F3,F3CNT,F3LAST,
     *F4,F4CNT,F4LAST,F5,F5CNT,F5LAST,F6,F6CNT,F6LAST,
     *F7,F7CNT,F7LAST,F8,F8CNT,F8LAST,F9,F9CNT,F9LAST,
     *F10,F10CNT,F10LST,
     *KCNT2,KCOUNT,KONE,KTHREE,KTWO,L,L2,MALTAK,
     *NRMSTP,PLUG,POWER,RBRST,RDY1,RDY2,SLOWDN,
     *TAKRDY,TKSLR,TMDSTP,TRMSTP,VMCRST,WATRST,XCOUNT,YCOUNT
      LOGICAL KBRAKE,KLIVE,KPLUG,KPOWER,LIVE,PLGLST
C*****************************************************************
      INTEGER ABSTAK,ABSVEL,ACCEL,ACCMAX,ACCOLD,BRKDLA,DIST,
     *INCACC,INCEM,INCFST,INCNRM,INCR,INCSLO,
     *DISTM,DLA1,INPO,INP1,INP17,INP2,INP3,INP4,
     *INP9,INTA,INTD,K,L2HIGH,L2LOW,LDIFF,LHIGH,
     *LLOW,LSB,MAXRBD,MSB,MSBO,K4CNT,NUMOUT,PLAT1,
     *PWAIT,RBDIST,SAT1,SAT2,SLOPA,SLOPD,TAK1,TAK2,
     *TAK3,TAKOLD,TAKTOL,TPLAT,VELCON,VELERR,VELMAX,VELOC,
     *VMC,VMCOUT,WAITMX,X,Y
C*****************************************************************
      COMMON /L/BADRD,BRAKES,COUNT,COUNT2,DSTRST,EMSTOP,FASTOP,
     *F1,F1CNT,F1LAST,F2,F2CNT,F2LAST,F3,F3CNT,F3LAST,
     *F4,F4CNT,F4LAST,F5,F5CNT,F5LAST,F6,F6CNT,F6LAST,
     *F7,F7CNT,F7LAST,F8,F8CNT,F8LAST,F9,F9CNT,F9LAST,
     *F10,F10CNT,F10LST,
     *KCNT2,KCOUNT,KONE,KTHREE,KTWO,L,L2,MALTAK,
     *NRMSTP,PLUG,POWER,RBRST,RDY1,RDY2,SLOWDN,
     *TAKRDY,TKSLR,TMDSTP,TRMSTP,VMCRST,WATRST,XCOUNT,YCOUNT
      COMMON /L1/KBRAKE,KLIVE,KPLUG,KPOWER,LIVE,PLGLST
C*****************************************************************
      COMMON /I/ABSTAK,ABSVEL,ACCEL,ACCMAX,ACCOLD,BRKDLA,DIST,
     *INCACC,INCEM,INCFST,INCNRM,INCR,INCSLO,
     *DISTM,DLA1,INPO,INP1,INP17,INP2,INP3,INP4,
     *INP9,INTA,INTD,K,L2HIGH,L2LOW,LDIFF,LHIGH,
     *LLOW,LSB,MAXRBD,MSB,MSBO,K4CNT,NUMOUT,PLAT1,
     *PWAIT,RBDIST,SAT1,SAT2,SLOPA,SLOPD,TAK1,TAK2,
     *TAK3,TAKOLD,TAKTOL,TPLAT,VELCON,VELERR,VELMAX,VELOC,
     *VMC,VMCOUT,WAITMX,X,Y
C*****************************************************************
C*****************************************************************
C THIS SUBROUTINE CYCLES IN AN ENDLESS LOOP DOING THE FOLLOWING:
C (1) APPLY HYDR. BRAKES, (2) SET VMC=0, (2A) WAIT 30 MSEC,
C (3) SET PLUG=0,  (4) TURN OFF MTR CNTRLR MAIN POWER RELAY,
C (5) APPLY HYDRAULIC BRAKES, START AT (1) AGAIN
```

```
C*****************************************************************
C FIRST, APPLY HYDRAULIC BRAKES
500     BRAKES=15
        CALL OUT(5,BRAKES)
C NEXT LIGHT UP BAD TAK INDICATOR IF EMSTOP CAUSED BY TAK
        IF(TKSLR.EQ.3) CALL OUT(6,2)
C NEXT, SET MOTOR CONTROL VOLTS TO 0
        VMC=0
        VMCOUT=VMC
C NEXT CALL UOTPUTS VMCOUT FROM DAC
        CALL POKE(-1,VMCOUT)
C THE FOLLOWING DO LOOP JUST INSERTS TIME DELAY TO ALLOW VMC TO REACH ZERO !!
        DO 515 K=1,1000
        BRAKES=1*15
        CALL OUT(5,BRAKES)
515     CONTINUE
C NOTE: PLUG=0 MEANS DONT PLUG!!!
        PLUG=0
        CALL OUT(4,PLUG)
C NEXT TURN OFF MTR CNTRLR POWER COMPLETELY
        POWER=0
        CALL OUT(7,POWER)
C NEXT TURN ON HYDRAULIC BRAKES AND HOLD IN AN ENDLESS LOOP!!!!!
502     DO 507 K=1,5000
        BRAKES=1*15
        CALL OUT(5,BRAKES)
507     CONTINUE
        GO TO 500
        END
```

APPENDIX C

FAILURE ANALYSIS TABLE

AMTV FAILURE MODE ANALYSIS DATA SHEET

Subsystem 1. Electronic Control Unit Power

Element	Failure Type	Effect	Risk	Prevention	When	Notes
+5 V Power Supply	Short to 12 V 0 V	Dead processor; speed control inoperative or processor bit errors	4	(a) Keep-alive circuit to test if program cycle remains at 20 msec period; Anomaly results in emergency stop	C	Must function properly independently of bus power failures
	Out of limits			(b) Read power voltage and stop if out of predetermined range	D	See also Microprocessor subsystem
$\pm$15 V Power Supply	Out of design limits	(a) Headway sensors inoperative	4	(a) Read power voltage and stop if out limits	D	
				(b) Hardware test circuit in sensor design of sensor so that it gives slow or stop output if supply voltage is out of tolerance	D	
				(c) Provide two outputs: Normal and complemented	B	
		(b) Steering angle readout/ erroneous		(a) Same as (a) above	D	See also steering subsystem
				(b) Software check for agreement between command and response		
		(c) Manual control faulty	2	(a) Rate limit on steering response so operator can compensate during safe stop period	D	

Risk:		When:	
4	Castastrophic	B	Controller Task
3	Severe	C	AMTV II
2	Small	D	Future
1	Negligible	F	Response to be Determined

AMTV FAILURE MODE ANALYSIS DATA SHEET (Cont'd)

Subsystem 2. Tachometer

Element	Failure Type	Effect	Risk	Prevention	When	Notes
Tach (#1 or 2)	error	Low reading results in overspeed condition	4	(a) Software check for difference in tach output greater than x; Command emergency	B	
		Speed control loop does not function		(b) Software check for condition where both tachs output 0 and commanded motor control signal is larger than x; Command emergency stop	B	Possible non-independent failure mode if both tach drives fail from same cause

Risk:	4	Castastrophic	When:	B	Controller Task
	3	Severe		C	AMTV II
	2	Small		D	Future
	1	Negligible		F	Response to be Determined

AMTV FAILURE MODE ANALYSIS DATA SHEET (Cont'd)

Subsystem 3. Microprocessor

Element	Failure Type	Effect	Risk	Prevention	When	Notes
Program Execution	Processor dead	No speed control	4	Keep-alive circuit to test for correct program cycle	B	
	Processor caught in loop	No speed control	4	(a) Keep-alive circuit as above	B	Keep-alive circuit must distinguish between random looping through subroutine supplying keep-alive pulse and normal looping through same
				(b) Check against model in a second processor; Anomaly commands emergency stop	D	
				(c) Software subroutine sequence counter; Missed subroutine commands stop	D	Item (b) in prevention above can substitute for Item (a) in prevention
				(d) For critical calculations, software counter to check that vital steps are taken	D	
	False variable value	Control outputs are improper	4	(a) Store critical variables in two locations and compare each cycle	D	Example is false secondary sensor flag Becomes critical if error repeats each cycle

Risk:	4	Castastrophic	When: B	Controller Task
	3	Severe	C	AMTV II
	2	Small	D	Future
	1	Negligible	F	Response to be Determined

AMTV FAILURE MODE ANALYSIS DATA SHEET (Cont'd)

Subsystem 4. TTL I-O (Assumes microprocessor is alive)

Element	Failure Type	Effect	Risk	Prevention	When	Notes
Sensor input from any critical sensor	Fails to detect input which should command a slow down or stop	AMTV fails to slow or stop from sensor input	4	(a) Read critical sensor inputs redundantly with A-D input	D	Need analysis to ensure there is no single point failure of microprocessor systems that would also not actuate keep-alive circuit
				(b) Periodic check daily before committing ATMV to service	D	Possibility of single point failure must be guarded against
LT, RT, U Turn Signal to turn on sensor	Fails to turn on sensor	No sensor function when required for turn	3	(a) Read sensor turn-on output with A-D input channel	D	Prevention (a) & (b) with sensors turned on/off by TTL output
				(b) Design sensor so lack of actuation results in slow output	D	Prevention (c) is for sensors which operate at all times
				(c) Operate turn sensors at all times, read only when enabled	D	Prevention (d) applies to both of above
				(d) Periodic check	D	
Brake signal output	Fails off, no brakes	No ability to stop	4	(a) Read hydraulic pressure with A-D input; Emergency stop command results if no agreement	D	Assume spring applied brake available
				(b) Compare vehicle speed to model	C	
				(c) Algorithm to hold vehicle stopped using motor only	B	In present vehicle software
				(d) Read TTL I/O output with analog A/D and compare	D	
	On	Stop	1	—		Not a hazard

Risk:		When:	
4	Castastrophic	B	Controller Task
3	Severe	C	AMTV II
2	Small	D	Future
1	Negligible	F	Response to be Determined

AMTV FAILURE MODE ANALYSIS DATA SHEET (Cont'd)

Subsystem 4. TTL I-0 (Cont'd) (Assumes microprocessor is alive)

Element	Failure Type	Effect	Risk	Prevention	When	Notes
Forward-Reverse Line	Fails in F state	Speed control inoperative; Overspeed condition when slowing is commanded	4	(a) Read forward-reverse line at motor controller with A-D input	C	
				(b) Read motor current with A-D input	D	
				(c) Compare predicted and measured speed	D	
	Fails in R state	Rollback occurs; Speed control loop inoperative	4	(a) See (a) above	C	
				(b) Rollback detection algorithm emergency stop is commanded after x rollback	B	
				(c) Hardware rollback detector or preventor (e.g., a oneway clutch)	D	Desirable for independent protection
TTL output to alive circuit	Noise	False shut down	1	Design of keep alive circuit	B	
	False output	False shut down	1		B	
Relay-motor power enable	Locked on	No disconnect in event of another failure calling for E-stop	1	(a) Read state of realy with A-D input; Provide indication of failure	D	No hazard: Emergency stop applies hydraulic brakes
	Locked off	No plugging speed control	4	See analog I/O subsystem, "Main motor control output"	D	

Risk:	4	Castastrophic	When:	B	Controller Task
	3	Severe		C	AMTV II
	2	Small		D	Future
	1	Negligible		F	Response to be Determined

AMTV FAILURE MODE ANALYSIS DATA SHEET (Cont'd)

Subsystem 5. Analog I-O (Assumes processor alive)

Element	Failure Type	Effect	Risk	Prevention	When	Notes
Main motor control output	Off	No plugging; Could be rollback cause; Failure to make sensor stop	4	(a) Compare measured speed to predicted model; Command emergency stop for difference >x	D	
				(b) Roll-back detection algorithm	B	
				(c) Read back A-D control voltage	D	
	Full-on	Runway: Possible loss of speed control	4	(a) Read back applied Motor control volt and command emergency stop if difference >x	D	
				(b) Compare measured and predicted speed (overspeed check)	D	
	Noisy-poorly cor-related to command value	Jerky—could result in overspeed	4	(a) Compare measured speed to model	D	
				(b) Test based on measured motor current	D	
				(c) Compare outputted value with control signal; Monitor using A-D channel; Stop if difference >x	D	Valid for all items above

Risk:	4	Castastrophic	When:	B	Controller Task
	3	Severe		C	AMTV II
	2	Small		D	Future
	1	Negligible		F	Response to be Determined

ORIGINAL PAGE IS OF POOR QUALITY

AMTV FAILURE MODE ANALYSIS DATA SHEET (Cont'd)

Subsystem 5. Analog I-O (Cont'd) (Assumes processor alive)

Element	Failure Type	Effect	Risk	Prevention	When	Notes
Motor current input	Any	Checks based on this signal are lost	1		D	Failure has no direct consequence; Causes false emergency stop
Steering angle input	Does not read angle correctly	Fails to provide redundant speed control on curves; Loss of consistency check of steering servo	2	(a) Provide indication of failure	C	Present vehicle uses TTL I/O for this function; Failure has no direct consequence; Causes false emergency stop
				(b) Checks based on this signal will be activated		
Analog brake output for service brake	Low output or no output	Loss of hydraulic brake control	4	(a) Compare vehicle speed with model	D	
				(b) Use two A-D boards and split hydraulic system	D	
				(c) Measure hydraulic brake pressure and compare to command in software	D	
Hydraulic brake pressure	Does not read pressure correctly	Loss of hydraulic brake system check	1	(a) Provide indication of failure	D	Failure has no direct consequence; Causes false emergency stop

Risk:			When:	
4	Castastrophic		B	Controller Task
3	Severe		C	AMTV II
2	Small		D	Future
1	Negligible		F	Response to be Determined

AMTV FAILURE MODE ANALYSIS DATA SHEET (Cont'd)

Subsystem 6. Road Marker Sensor

Element	Failure Type	Effect	Risk	Prevention	When	Notes
Road Marker Sensor	Fails to sense magnet	Failure to stop, overspeed on curve, or fails to attain or slow from 30 km/hr	4	(a) Redundant sensors	D	Duplicate sensors, together with appropriate design of the coded road magnet patterns are required to ensure detection of sensor errors
				(b) Require repeat of signal at intervals for fast mode to continue	D	
				(c) Redundant magnets	D	Use of two dissimilar sensors (e.g., optical and magnetic) is desirable; Desirable to investigate mode of primary headway sensor to include 30 km/hr enable function
	Fails to pass over magnet(s)	Same as above	4	Redesign sensor for extended lateral range; Range to be greater than guide-wire acquisition tolerance	D	

Risk:	4	Castastrophic	When:	B	Controller Task
	3	Severe		C	AMTV II
	2	Small		D	Future
	1	Negligible		F	Response to be Determined

AMTV FAILURE MODE ANALYSIS DATA SHEET (Cont'd)

Subsystem 7. Steering System

Element	Failure Type	Effect	Risk	Prevention	When	Notes
Electronic-guide wire sensor	Sudden hard-over response	Vehicle upset possible; Sudden large deviation from path; Large lateral acceleration	4	(a) Monitor supply voltages; Center steering and stop vehicle if anomaly detected	D	Fast-response in detecting anomaly and removing power from servo is desirable for all hard-over type failures; Stopping can follow
				(b) Monitor servo input (sensor signal) for transient; Center steering and stop forstep input >x	D	Does not test for servo failure
				(c) Monitor servo output (steering angle); Test for:	D	
				(1) Transient-slew rate >x; Limit slew rate and stop if limit imposed		Acts if slew rate exceeds limit; Functionally equivalent to (e) below;
				(2) Lateral acceleration using steering angle and velocity; Limit acceleration and stop if limit exceeded		Equivalent to (f) below
				(d) Test for agreement between command (sensor signal) and response (steering angle)	D	Does not detect problem in sensor
				(e) Limit hydraulic flow rate with special valve	D	Works in manual mode; Effect on servo stability must be determined; Prevents lurch, but relies on loss of acquisition to stop

Risk:	4	Castastrophic	When:	B	Controller Task
	3	Severe		C	AMTV II
	2	Small		D	Future
	1	Negligible		F	Response to be Determined

AMTV FAILURE MODE ANALYSIS DATA SHEET (Cont'd)

Subsystem 7. Steering System (Cont'd)

Element	Failure Type	Effect	Risk	Prevention	When	Notes
				(f) Limit hydraulic force using relief valves between two ends of actuating cylinder	D	Works in manual mode; Direct limitation on lateral acceleration; Relies on loss of acquisition on stop; Effect on servo stability must be determined
				(g) Install hydraulic steering snubber with a self-centering mechanical deadband	D	An alternative method for limiting slew rate as in (c-1) and (e) above
				(h) Mechanical steering angle limit switches; Electrical limit to travel before switch actuation; Actuation commands emergency stop	D	Does not act until after steering angle goes hard-over

Risk:			When:		
	4	Castastrophic		B	Controller Task
	3	Severe		C	AMTV II
	2	Small		D	Future
	1	Negligible		F	Response to be Determined

ORIGINAL PAGE IS
OF POOR QUALITY

AMTV FAILURE MODE ANALYSIS DATA SHEET (Cont'd)

Subsystem 8. Keep Alive

Element	Failure Type	Effect	Risk	Prevention	When	Notes
Keep-alive circuit - Output to vehicle	Locked in OK state	Failure to shut down if second failure occurs	4	(a) Periodic check before committing AMTV to service variant of (c) below	D	No hazard until a second failure occurs
				(b) Check vehicle speed against independent model in a second processor with independent I/O	D	A substitute for the keep-alive circuit; However, same failure mode could exist
				(c) Periodic check of function when AMTV stopped		

Risk:		When:	
4	Castastrophic	B	Controller Task
3	Severe	C	AMTV II
2	Small	D	Future
1	Negligible	F	Response to be Determined