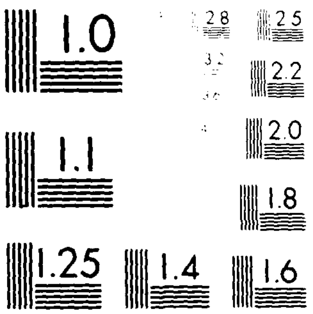
AD-A113 103 INSTITUTE FOR SOFTWARE ENGINEERING PALO ALTO CA F/G 9/2
AN/UYK-20 CONFIGURATION CAPACITY, NELC-UDI-A-106.(U)
JAN 78 L M TRAISTER N66001-77-C-0252
UNCLASSIFIED NL

1 OF 2
AD A
113103


1.0
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

PHOTOGRAPH THIS SHEET

AD A11 3103

DTIC ACCESSION NUMBER

I

LEVEL

INVENTORY

AN/UYK-20 Configuration Capacity NELC-UDI-A-106

Final Rept.

DOCUMENT IDENTIFICATION

Contract N66001-77-C-0252

30 Jan. '78

DISTRIBUTION STATEMENT

ACCESSION FOR

NTIS GRA&I ☒
DTIC TAB □
UNANNOUNCED □
JUSTIFICATION

BY Per Ltr. on file

DISTRIBUTION /

AVAILABILITY CODES

DIST	AVAIL AND/OR SPECIAL	
F		

DISTRIBUTION STAMP

DATE ACCESSIONED

81 7 23 71

DATE RECEIVED IN DTIC

PHOTOGRAPH THIS SHEET AND RETURN TO DTIC-DDA-2

File 375-016

AD-A113103

AN/UYK-20

CONFIGURATION CAPACITY

NELC-UDI-A-106

FINAL REPORT

Prepared for:

NAVAL OCEAN SYSTEMS CENTER

Contract No. N66001-77-C-0252BW

Date: January 30, 1978

Prepared By: Leon M. Traister

INSTITUTE FOR SOFTWARE ENGINEERING

P.O. Box 637, Palo Alto, CA 94303

CONTENTS

FIGURES

TABLES

Table No. Page

OVERVIEW

This report presents the development of equations and curves that describe the capacity characteristics of the AN/UYK-20 and its peripherals. These will form the basis for a future Capacity Calculations handbook and will contribute to the subsequent capability of making capacity assessments under conditions of actual utilization of the AN/UYK-20 by a large variety of both systems and applications software.

The means employed to assess capacity in this report are those of Software Physics. This discipline, developed in Kolence's *An Introduction to Software Physics*, mathematically derives capacity characteristics of computing equipment from fundamental equipment descriptions and specifications. However, capacity available is not necessarily power actually used by a workload (software) executing on the equipment. Software Physics theory identifies the parameters which govern the utilization of capacity and predicts the quantity of power actually used. This is essential to the improvement of performance. Further discussion of capacity, power and performance, as well as other Software Physics terminology can be found in the Glossary which follows the body of this report.

A predictive theory must be tested; and so the companion report, "AN/UYK-20 Capacity Equipment Specification" proposes experiments intended to verify the power equations and curves of this study. It is anticipated that a future report will contain the full design of such experiments and that the comprehensive handbook of AN/UYK-20 Capacity Characteristics will be developed concurrent with the experiment performance.

SECTION 1

INTRODUCTION

1.0 GENERAL

This is the final report for the study of AN/UYK-20 configuration capacity performed for the Naval Ocean Systems Center, San Diego, under Contract N6601-77-C-0252BW. Included in this report are the exposition of methodology for, and presentation of, theoretical capacity (power) equations and curves for typical AN/UYK-20 configurations, devices and processors.

1.1 OBJECTIVES AND GOALS

The objectives of this study are to produce theoretically derived capacity (power) characteristics for the AN/UYK-20 computer CP and peripherals. These will form the basis for a future capacity calculation handbook and will contribute to the subsequent capability of making capacity assessments under conditons of actual utilization by a large variety of both systems and applications software units. As a consequence, methods of predicting performance (e.g., throughput, response times, etc.) become capable of realization. Additionally, were characterization techniques thoroughly developed for AN/UYK-20 introduction mixes and IOC command sequences, the software units could then become subject to control of performance and capable of optimization at the design stage.

It is a particular feature of this study that, for the goal of describing actual performance, it considers the effects of contention for main memory by the CP, IOC and DMA facility on CP capacity. Thus CP execution power will be described as functions of concurrent IOC or DMA power used.

1.2 SCOPE AND APPROACH

The following sections present a series of equipment capacity equation and curve developments leading to the higher level IOC/ channel configuration power characteristics. Each development of capacity equations and curves for peripheral devices is preceded by a brief list of pertinent device specifications. More complete information is to be found in specifications provided by the manufacturers.

Three peripheral devices were selected for analysis. These are the AN/USH-26 Cartridge Magnetic Tape Unit (CMTU), the AN/USH-23 Disk Controller/Storage System and the AN/USQ-69 Keyboard/Display unit. For the first two, capacity curves are also developed for multiple units operating with a single controller; that is, for a control unit configuration. These curves are shown tabulated and plotted for select cases of parameterization. The presentation of more extensive tabulation will be the function of an anticipated software physics handbook for these devices and configurations.

The subsequent sections of this report present a development of a CP power methodology for the AN/UYK-20. This includes a vector formulation of CP forces with components determined by the various containers operated on and by the nature of the action. From this we will present a methodology developing CP power for classes of instructions and the workloads of which they are constituents. The impact of IOC or DMA contention with the CP for main memory is then analyzed and the effects on CP execution power are developed for some specific instruction classes and a typical instruction mix.

These methods and results should be considered the predecessors of a comprehensive handbook of AN-UYK-20 capacity methodology and tabulations, offering the potential of the use of software physics theory and results in effective hardware configuration design and in the design and implementation of system or applications software.

1.3 REFERENCE PUBLICATIONS

This report assumes familiarity with the fundamentals and terminology of Software Physics and some familiarity with the AN/UYK-20 and its peripherals. The following two publications are offered as containing material adequate to satisfy these prerequisites.

(a) Kolence, Kenneth, "*An Introduction to Software Physics*", Institute for Software Engineering, Inc., Palo Alto, 1977.

(b) Sperry Univac Defense Systems, "*AN/UYK-20 Technical Description*", Publication number PX10431C, Sperry Univac Corporation, St. Paul, Minn.

1.4 ACKNOWLEDGEMENT

The author gratefully acknowledges the contributions to the development of this report provided by:

William J. Dejka (NOSC)
Robert B. Holland (System Industries, Inc.)
Kenneth W. Kolence (Institute for Software Engineering)
John Westergren (Sperry-Univac)
William Yonkers (Qantex, Inc.)

SECTION 2

AN/USH-26 CMTU POWER

2.1 GENERAL

The AN/USH-26 Cartridge Magnetic Tape Unit (CMTU) like many other tape devices operates in the Software Physics Type 1 mode. This means that the control unit and channel are in execution whenever an individual drive is performing actions consequent on the issuance of a read or write command.

Thus, when only these read/write related actions are considered, the α or β configuration power is:

(a) Independent of the number of drives in the α or β configuration.

(b) Identical to the power curve for a single drive using the average blocksize for the entire α or β configuration.

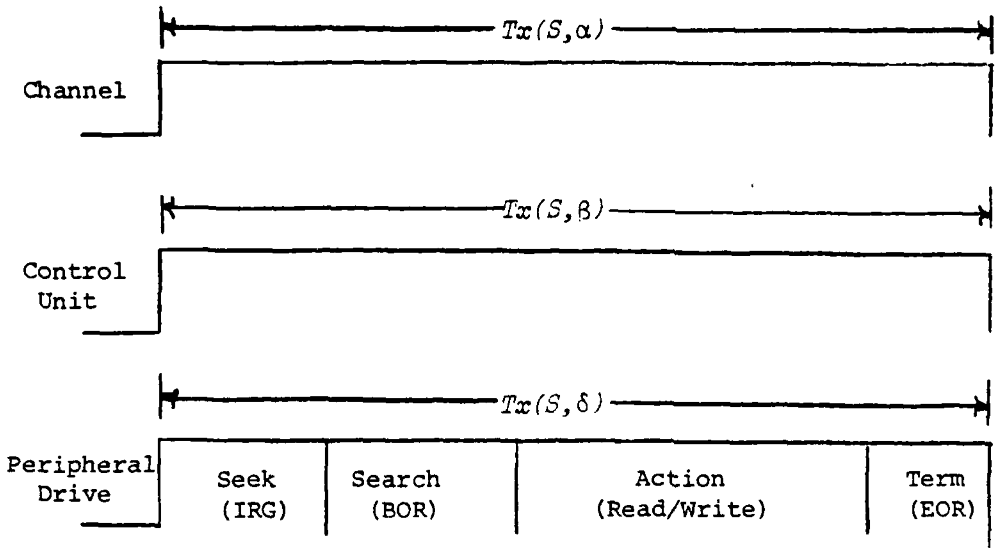
2.2 CMTU CHARACTERISTICS

(a)	Tape speed	-	30 ips
(b)	Start-up time	-	30 ± 1 msec
(c)	Interrecord gap (IRG)	-	1.2 - 1.8 inches. (1.6 - 1.8 inches for successive read/write operations)
(d)	Recording density	-	1600 bits/inch (serial)
(e)	Data rates	-	320 msec between 16 bit words (nominal)
(f)	Overhead bytes	-	16 bit preamble 16 bit CRC 16 bit postamble
(g)	Record lengths	-	2048 bytes max. recommended
(h)	Rewind time	-	42 sec. for 300 ft. of tape
(i)	Configuration(s)	-	1-4 drives/controller

2.3 DEVICE AND CONFIGURATION STATE CHARTS

The STANDARD CYCLE STATE CHART, Figure 2.1, shows the drive/control unit/channel busy states for read/write actions.

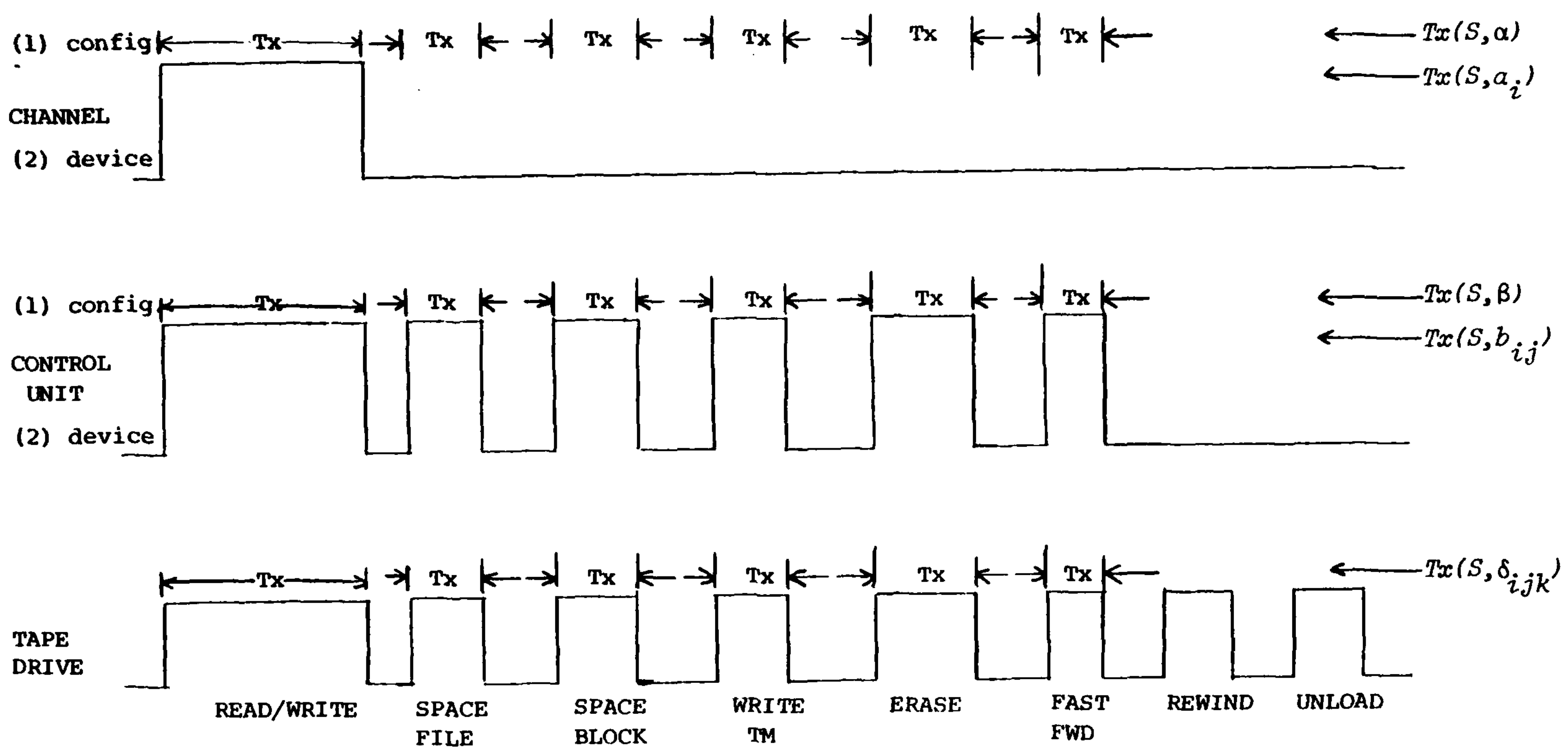
The FULL STATE CHART, Figure 2.2, shows the busy states for all drive activating commands.



STANDARD CYCLE STATE CHART (Type 1)

AN/USH-26 CMTU

Figure 2.1



AN/USH-26 CMTU TAPE FULL STATE CHART

Figure 2.2

2.4 CMTU DRIVE POWER

As an instance of type 1 (tape) power, we have that:

$$P(tp,CMTU) = \frac{W}{t_1 + t_2 + t_4 + W \div MBTR}$$

where: $MBTR = \text{tape speed} \times \text{byte density} = 30 \times 200 = 6 \times 10^3 \text{ bytes/sec}$

$$t_1 = \text{IRG time} = \ell(IRG)/\text{tape speed}$$

$$= 1.7/30 = 56.7 \text{ msec}$$

$$t_2 = \text{Record preamble time} = \#\text{ preamble bytes} \div MBTR$$

$$= 2/6 \times 10^3 = .333 \text{ msec}$$

$$t_4 = \text{Record CRC + Postamble time}$$

$$= (\#CRC \text{ bytes} + \#\text{postamble bytes}) \div MBTR$$

$$= (2+2) \div 6 \times 10^3 = 0.667 \text{ msec.}$$

So $$P(tp,CMTU) = \frac{W}{56.7 + 0.333 + 0.667 + W/6 \times 10^3 \times 10^{-3}} \quad \text{kw/s}$$

$$= \frac{W}{57.7 + 0.167\,W} \quad \text{kw/s} \qquad (2.1)$$

and the block size efficiency

$$= \frac{P(tape,CMTU,W_i)}{P_{asymptotic}} = \frac{P(tape,CMTU,W_i)}{6 \times 10^3} \qquad (2.2)$$

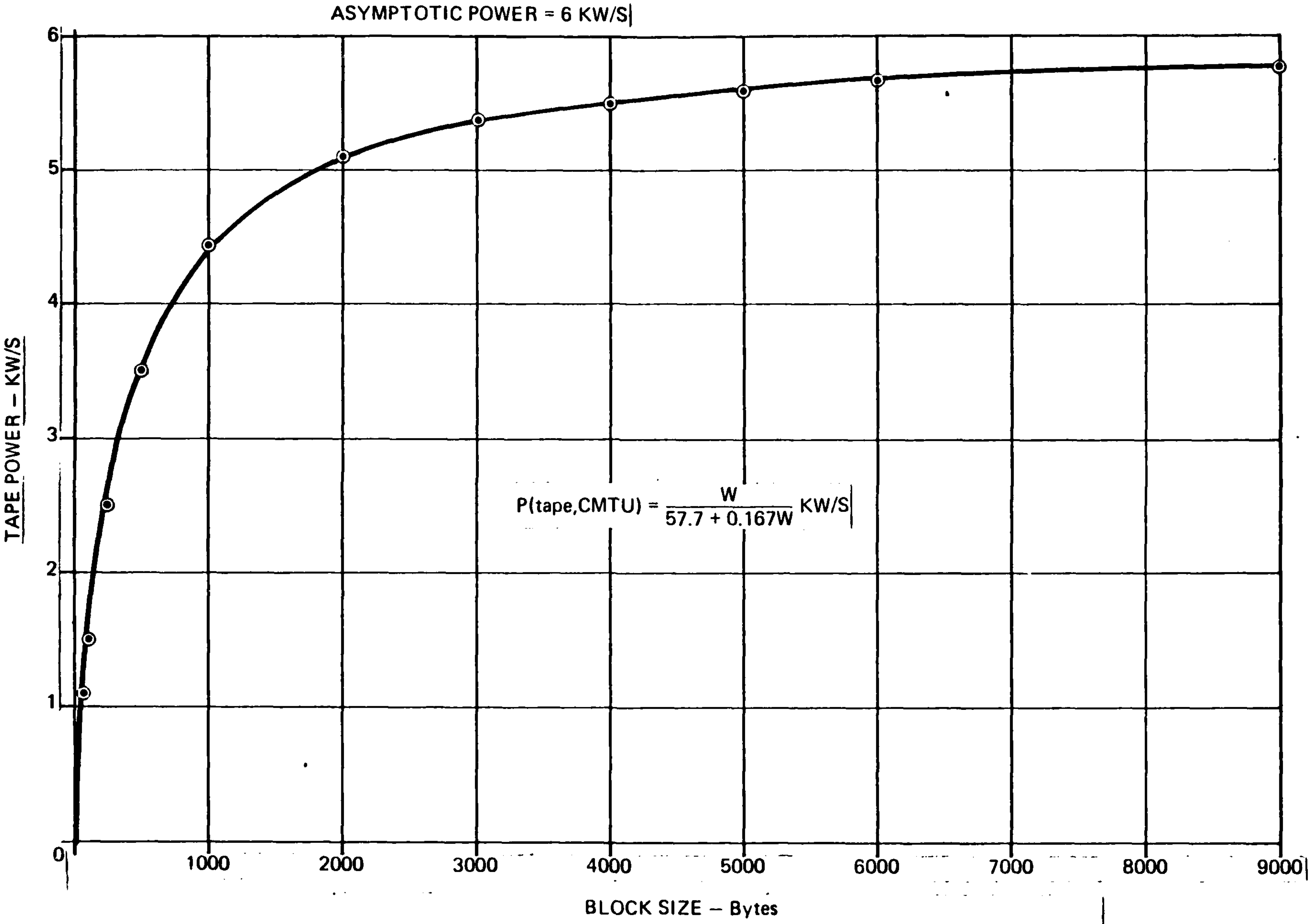
Equations (2.1) and (2.2) are tabulated and plotted in Table 2.1 and Figure 2.3, respectively for a wide range of data blocksizes.

As a consequence of the fact that the channel and control unit are busy throughout the device standard cycle, the defining characteristic of type 1 power, and thus as no power-producing overlap is possible between multiple drives on a control unit or channel, the power characteristics of the α (channel) or β (control unit) configurations are identical to that of a single drive.

BLOCKSIZE (Bytes)	EXECUTION TIME (10^{-3} sec)	TAPE POWER (KW/sec)	%BLOCKSIZE EFFICIENCY
80	71.06	1.126	18.77
120	77.74	1.544	25.73
250	99.45	2.514	41.90
500	141.2	3.541	59.02
1,000	224.7	4.450	74.17
2,000	391.7	5.105	85.08
3,000	558.7	5.370	89.50
4,000	725.7	5.512	91.87
5,000	892.7	5.600	93.33
6,000	1060.	5.662	94.39
10,000	1728.	5.788	96.47
20,000	3398.	5.886	98.10
100,000	16760.	5.967	99.45
1,000,000	167057.	5.986	99.77

CONTINUOUS READ/WRITE POWER (Type 1)
AN/USH-26 CARTRIDGE MAGNETIC TAPE UNIT

Table 2.1



CONTINUOUS READ/WRITE POWER – AN/USH-26 CARTRIDGE MAGNETIC TAPE UNIT
Channel/Control Unit/Drive

Figure 2.3

SECTION 3

AN/USH-23 DISK SYSTEM POWER

3.0 GENERAL

The AN/USH-23 disk system (currently the System Industries Model 3500) consists of a control unit and from one to eight drives with either a fixed or removable disk platter.

The control unit and drives operate in the Software Physics type 2 mode; that is, the initial positioning seek on one drive may be overlapped with actions on others. Data records are formatted into fixed length sectors on the disk and the system is capable of reading or writing records that span sector, track or cylinder boundaries as effectively one operation.

3.1 AN/USH-23 DISK SYSTEM CHARACTERISTICS

3.1.1 System

(a) System Capacity - 19488 k Bytes

(b) Up to 8 drives may be attached to a single controller. Drives 5-8 are daisy-chained from 1-4 and share controller registers.

At least one drive in a daisy-chained pair must be a removable cartridge type.

(c) Addressing by disk sector (type 2).

(d) Overlap seek permits concurrent seeking on up to 8 drives. Subsequent data transfer may occur after seek on one drive while others are still seeking.

(e) Spanning of sectors, tracks and cylinders continues without IOC action required until the word count is satisfied.

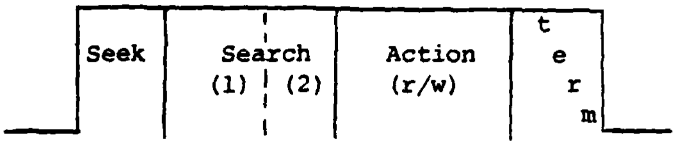
3.1.2 Drives (DIABLO models 31/33F)

(a)	Tracks per surface	- 203
(b)	Tracks per cylinder	- 2
(c)	Words/sector	- 256 or 128
(d)	Sectors/track	- 12 or 24
(e)	Disk capacity	- 2436 k Bytes
(f)	Byte transfer rate (MBTR)	- 195.2 k Bytes/sec
(g)	Average latency	- 20 ms
(h)	Head movement times:	
	Cylinder to cylinder	- 15 ms
	Average	- 70 ms
	200 cylinders	- 135 ms
(i)	Sector format:	
	i) First preamble	- 20 bytes
	ii) Sector address word	- 2 bytes
	iii) Sector status word	- 2 bytes
	iv) Second preamble	- 20 μsec
	v) Data	- 512/256 bytes
	vi) CRC	- 2 bytes
(j)	Interrecord gap:	
	12 sector format	- 524 μsec
	24 sector format	- 168 μsec
(k)	Maximum transfer	- 128 k Bytes

3.2 AN/USH-23 DRIVE POWER

3.2.1 Single Sector Standard Cycle

Figure 3.1 presents a standard cycle for the input/output of a single sector on the series 30 disk drive. There are certain irregularities for this drive as compared to the ordinary type 2 drive power. These are:



initial action	t_{10}	t_{20}, t_{22}	t_3	t_{41}, t_{42}
track boundary	0	$0, t_{22}$	t_3	t_{41}, t_{42}
cyl boundary	t_{11}	t_{21}, t_{22}	t_3	t_{41}, t_{42}

where: t_{10} = stand alone seek time

t_{11} = cyl to cyl positioning time - 15 ms

t_{20} = average rotational delay - 20 ms

t_{22} = sector preamble time - 0.14 ms

t_{21} = rot delay after cylinder to cylinder repositioning - 25 ms

t_3 = read/write action time

12 sector format - 2.611 ms

24 sector format - 1.306 ms

t_{41} = CRC read/write time - .01 ms

t_{42} = overhead (interrecord) time

12 sector - .524 ms

24 sector - .128 ms

Notes: (a) Stand alone seek time, t_{10}, is for an overlap seek operation.

(b) Initial action search time ($t_{20}+t_{22}$) includes an unexecuted seek.

MODEL 31/33F TYPE 2 STANDARD CYCLE

(SINGLE SECTOR)

Figure 3.1

(a) There are <u>two</u> search substates due to the sector format and the fact that sector, track and cylinder spanning is possible for input/output of a single logical record. The first substate represents the time of expected latency after an initial seek (t_{20}) or of rotational delay after cylinder repositioning (t_{21}). The second substate represents the time to detect and pass over the first record preamble (t_{22}). Note that the rotational delay after cylinder to cylinder repositioning is a fixed value, not a statistical average as for initial seek expected latency.

We have therefore that:

i) After the initial seek:

t_2 = *search time* = $t_{20} + t_{22}$

where t_{20} = *average rotational delay = 20 ms.*

t_{22} = *sector preamble time = 0.14 ms.*

ii) After a cylinder to cylinder repositioning:

$t_2 = t_{21} + t_{22}$

where t_{21} = *rotation time - seek time*
= *40 - 15 = 25 ms.*

t_{22} = *sector preamble time = 0.14 ms.*

iii) Within a cylinder:

$t_2 = t_{22} = 0.14$ *ms.*

(b) There are <u>two</u> termination substates, t_{41} and t_{42} due to the fact that interrecord overhead must be accounted for in all but the last sector read or written for a logical record.

Thus t_4 = *termination time* = $t_{41} + t_{42}$

for all but the last sector

$t_4 = t_{41}$ for the last sector.

3.2.2 Formulation of $Tx(S,\delta)$

As multiple sectors, tracks and cylinders can be spanned during a single input/output operation on a Model 31/33F drive, and we will need to formulate an expression for the device execution time that accounts for the boundary events noted in Figure 3.1.

We let N_{ij} equal the number of occurrences of the events whose time is t_{ij}; e.g., N_{21} is the number of search (1) events.

Let b = *bytes/block (sector)*

s = *sectors/track*

c = *tracks/cylinder*

Let σ_0 = *the sector offset of the first block on the commencing track (0-s)*

Let τ_0 = *the track offset of the first track on the commencing cylinder (0-c)*

Now for a given byte count W,

$$B = \textit{block occupancy} = \left[\frac{W}{B}\right]_{\uparrow} \qquad (3.1)$$

where $[a]_{\uparrow}$ = *the first integer* $\geq a$

$$T = \textit{track occupancy} = \left[\frac{B}{s}\right] + \left[\frac{B \bmod s + \sigma_0}{s}\right]_{\uparrow} \qquad (3.2)$$

where $[a]$ = *integer portion of* a

$$C = \textit{cylinder occupancy} = \left[\frac{T}{C}\right] + \left[\frac{T \bmod c + \tau_0}{c}\right]_{\uparrow} \qquad (3.3)$$

So we have:

$N_{11} = C - 1$ (first cylinder seek time is t_{10})

$N_{21} = C - 1$ (we have cylinder to cylinder rotational delay on all but the first cylinder)

$N_{22} = B$ (one preamble for each block)

$N_{3} = B$ (data portion each block)

$N_{41} = B$ (CRC termination each block)

$N_{42} = B - 1$ (overhead IRG for all blocks but the last)

So for a logical record of W bytes:

$$Tx(S,\delta) = t_{10} + N_{11}t_{11} + t_{20} + N_{21}t_{21} + N_{22}t_{22} + N_{3}t_{3} + N_{41}t_{41} + N_{42}t_{42}$$

Substituting in terms of blocks and cylinders we have:

$$Tx(S,\delta) = t_{10} + (C-1)(t_{11}+t_{21}) + t_{20} + B(t_{22}+t_{3}+t_{41}) + (B-1)t_{42} \tag{3.4}$$

3.2.3 Evaluation of Series 30 Drive Power

The power of a single Series 30 drive, denoted $P(disk,31/33F)$, is given by:

$$P(disk,31/33F) = \frac{W}{Tx(S,\delta)} \tag{3.5}$$

where: W is the work done in a standard cycle (possibly multi-sector)

$Tx(S,\delta)$ is the execution time of the drive for the standard cycle as given by Equation (3.4).

Table 3.1 presents *P(disk,31/33F)* tabulated for values of *W* that are equivalent to the start, midpoint and endpoint of sectors to beyond the capacity of a cylinder. It thus shows the discontinuous power drops at the ends of sectors and at the end of boundary. The asymptotic power is, as usual, equivalent to the maximum byte transfer rate (MBTR), which for this drive is 195.2 KW/sec. Figure 3.1a is a plot of the initial part of the power curve showing in detail the sector endpoint discontinuities which give the function a sawtooth shape. The subsequent Figure 3.1b additionally shows the larger discontinuity in the power function at the first cylinder boundary. In both figures, curves are drawn through the sector maximum or minimum power points defining smooth power envelopes within the domain of a single cylinder.

3.2.4 Discussion

The discontinuities in the Model 31/33F power function indicate that the optimization of device power can depend on the sector and track offsets of the commencing record. Thus the designer should note that in general these offsets should be chosen so as to minimize the number of cylinder to cylinder repositions required to satisfy the request. Record sizes that utilize only a small fraction of a sector are inefficient as well. It should be emphasized that the time intervals caused by the spanning of sectors, tracks or cylinders are not available for other actions (except previously initiated seeks) by other devices on the same Input/Output channel.

DASD DRIVE POWER - KW/SEC

BLOCK(SECTOR) OFFSET: 0 TRACK OFFSET: 0

INPUT/OUTPUT WORK	FOR INITIAL SEEK TIMES (MSEC):				
(BYTES)	0.00000E+00	1.50000E+01	4.50000E+01	7.00000E+01	1.00000E+02
1.00000E+00	4.39344E-02	2.64822E-02	1.47577E-02	1.07804E-02	8.14589E-03
2.56000E+02	1.12472E+01	6.77945E+00	3.77797E+00	2.75977E+00	2.08535E+00
5.12000E+02	2.24944E+01	1.35589E+01	7.55595E+00	5.51955E+00	4.17070E+00
5.13000E+02	1.96956E+01	1.24981E+01	7.22063E+00	5.34117E+00	4.06993E+00
7.68000E+02	2.94858E+01	1.87105E+01	1.08098E+01	7.99613E+00	6.09299E+00
1.02400E+03	3.93145E+01	2.49474E+01	1.44131E+01	1.06615E+01	8.12399E+00
1.02500E+03	3.49453E+01	2.31212E+01	1.37896E+01	1.03190E+01	7.92536E+00
1.28000E+03	4.36389E+01	2.88733E+01	1.72201E+01	1.28861E+01	9.89704E+00
1.53600E+03	5.23667E+01	3.46480E+01	2.06642E+01	1.54634E+01	1.18764E+01
1.53700E+03	4.71230E+01	3.22785E+01	1.98024E+01	1.49781E+01	1.15898E+01
1.79200E+03	5.49410E+01	3.76338E+01	2.30878E+01	1.74630E+01	1.35126E+01
2.04800E+03	6.27897E+01	4.30100E+01	2.63860E+01	1.99577E+01	1.54430E+01
2.04900E+03	5.70721E+01	4.02538E+01	2.53269E+01	1.93481E+01	1.50770E+01
2.30400E+03	6.41747E+01	4.52635E+01	2.84789E+01	2.17560E+01	1.69534E+01
2.56000E+03	7.13052E+01	5.02927E+01	3.16432E+01	2.41733E+01	1.88371E+01
2.56100E+03	6.53530E+01	4.72621E+01	3.04203E+01	2.34551E+01	1.83997E+01
2.81600E+03	7.18602E+01	5.19680E+01	3.34493E+01	2.57906E+01	2.02317E+01
3.07200E+03	7.83930E+01	5.66924E+01	3.64901E+01	2.81352E+01	2.20710E+01
3.07300E+03	7.23529E+01	5.34692E+01	3.51311E+01	2.73223E+01	2.15691E+01
3.32800E+03	7.83568E+01	5.79061E+01	3.80463E+01	2.95895E+01	2.33589E+01
3.58400E+03	8.43842E+01	6.23604E+01	4.09729E+01	3.18656E+01	2.51557E+01
3.58500E+03	7.83477E+01	5.90050E+01	3.95008E+01	3.09699E+01	2.45956E+01
3.84000E+03	8.39205E+01	6.32020E+01	4.23105E+01	3.31728E+01	2.63451E+01
4.09600E+03	8.95152E+01	6.74155E+01	4.51312E+01	3.53843E+01	2.81014E+01
4.09700E+03	8.35393E+01	6.39729E+01	4.35653E+01	3.44162E+01	2.74887E+01
4.35200E+03	8.87389E+01	6.79546E+01	4.62768E+01	3.65583E+01	2.91997E+01
4.60800E+03	9.39588E+01	7.19519E+01	4.89990E+01	3.87088E+01	3.09173E+01
4.60900E+03	8.80791E+01	6.84559E+01	4.73553E+01	3.76774E+01	3.02571E+01
4.86400E+03	9.29522E+01	7.22434E+01	4.99753E+01	3.97619E+01	3.19311E+01
5.12000E+03	9.78444E+01	7.60457E+01	5.26056E+01	4.18547E+01	3.36117E+01
5.12100E+03	9.20825E+01	7.25219E+01	5.08979E+01	4.07680E+01	3.29085E+01
5.37600E+03	9.66677E+01	7.61331E+01	5.34324E+01	4.27980E+01	3.45472E+01
5.63200E+03	1.01271E+02	7.97585E+01	5.59768E+01	4.48361E+01	3.61923E+01
5.63300E+03	9.56393E+01	7.62263E+01	5.42164E+01	4.37011E+01	3.54503E+01
5.88800E+03	9.99688E+01	7.96770E+01	5.66708E+01	4.56794E+01	3.70551E+01

DIABLO SERIES 30 TYPE 2 DRIVE POWER - PART I

Table 3.1a

INPUT/OUTPUT WORK	DASD DRIVE POWER - KW/SEC				
	BLOCK(SECTOR) OFFSET: 0	TRACK OFFSET: 0			
(BYTES)	FOR INITIAL SEEK TIMES (MSEC):				
	0.00000E+00	1.50000E+01	4.50000E+01	7.00000E+01	1.00000E+02
6.14400E+03	1.04315E+02	8.31412E+01	5.91347E+01	4.76655E+01	3.86662E+01
6.14500E+03	9.88203E+01	7.96154E+01	5.73315E+01	4.64884E+01	3.78892E+01
6.40000E+03	1.02921E+02	8.29192E+01	5.97106E+01	4.84175E+01	3.94615E+01
6.65600E+03	1.07038E+02	8.62360E+01	6.20991E+01	5.03542E+01	4.10399E+01
6.65700E+03	1.01682E+02	8.27278E+01	6.02614E+01	4.91405E+01	4.02312E+01
6.91200E+03	1.05577E+02	8.58967E+01	6.25697E+01	5.10228E+01	4.17722E+01
7.16800E+03	1.09487E+02	8.90780E+01	6.48871E+01	5.29126E+01	4.33194E+01
7.16900E+03	1.04270E+02	8.55959E+01	6.30220E+01	5.16670E+01	4.24820E+01
7.42400E+03	1.07979E+02	8.86406E+01	6.52637E+01	5.35048E+01	4.39930E+01
7.68000E+03	1.11703E+02	9.16972E+01	6.75141E+01	5.53498E+01	4.55100E+01
1.12650E+04	1.18535E+02	1.02376E+02	8.04439E+01	6.82580E+01	5.77587E+01
1.15200E+04	1.21218E+02	1.04693E+02	8.22648E+01	6.98031E+01	5.90661E+01
1.17760E+04	1.23912E+02	1.07020E+02	8.40929E+01	7.13543E+01	6.03787E+01
1.17770E+04	1.19781E+02	1.03926E+02	8.21723E+01	6.99676E+01	5.93836E+01
1.20320E+04	1.22375E+02	1.06176E+02	8.39516E+01	7.14826E+01	6.06694E+01
1.22880E+04	1.24979E+02	1.08436E+02	8.57378E+01	7.30035E+01	6.19602E+01
1.22890E+04	8.67831E+01	7.84708E+01	6.58553E+01	5.80749E+01	5.08638E+01
1.25440E+04	8.85839E+01	8.00991E+01	6.72219E+01	5.92800E+01	5.19193E+01
1.28000E+04	9.03917E+01	8.17338E+01	6.85937E+01	6.04898E+01	5.29788E+01
1.28010E+04	8.83491E+01	8.00607E+01	6.74123E+01	5.95697E+01	5.22722E+01
1.30560E+04	9.01090E+01	8.16556E+01	6.87552E+01	6.07563E+01	5.33135E+01
1.33120E+04	9.18759E+01	8.32567E+01	7.01033E+01	6.19476E+01	5.43588E+01
1.33130E+04	8.98457E+01	8.15866E+01	6.89163E+01	6.10194E+01	5.36433E+01
1.35680E+04	9.15666E+01	8.31493E+01	7.02363E+01	6.21882E+01	5.46708E+01
1.38240E+04	9.32943E+01	8.47182E+01	7.15616E+01	6.33616E+01	5.57023E+01
1.74090E+04	9.97891E+01	9.18885E+01	7.93273E+01	7.12147E+01	6.34305E+01
1.76640E+04	1.01251E+02	9.32344E+01	8.04892E+01	7.22578E+01	6.43596E+01
1.79200E+04	1.02718E+02	9.45856E+01	8.16557E+01	7.33051E+01	6.52923E+01
1.79210E+04	1.00825E+02	9.29787E+01	8.04559E+01	7.23370E+01	6.45237E+01
1.81760E+04	1.02260E+02	9.43017E+01	8.16007E+01	7.33663E+01	6.54418E+01
1.84320E+04	1.03700E+02	9.56299E+01	8.27500E+01	7.43996E+01	6.63635E+01
1.84330E+04	1.01824E+02	9.40323E+01	8.15517E+01	7.34300E+01	6.55912E+01
1.86880E+04	1.03232E+02	9.53332E+01	8.26799E+01	7.44458E+01	6.64986E+01
1.89440E+04	1.04647E+02	9.66391E+01	8.38125E+01	7.54656E+01	6.74096E+01
1.89450E+04	1.02787E+02	9.50513E+01	8.26161E+01	7.44947E+01	6.66342E+01
1.92000E+04	1.04170E+02	9.63307E+01	8.37282E+01	7.54974E+01	6.75311E+01

DIABLO SERIES 30 TYPE 2 DRIVE POWER - PART II

Table 3.1b

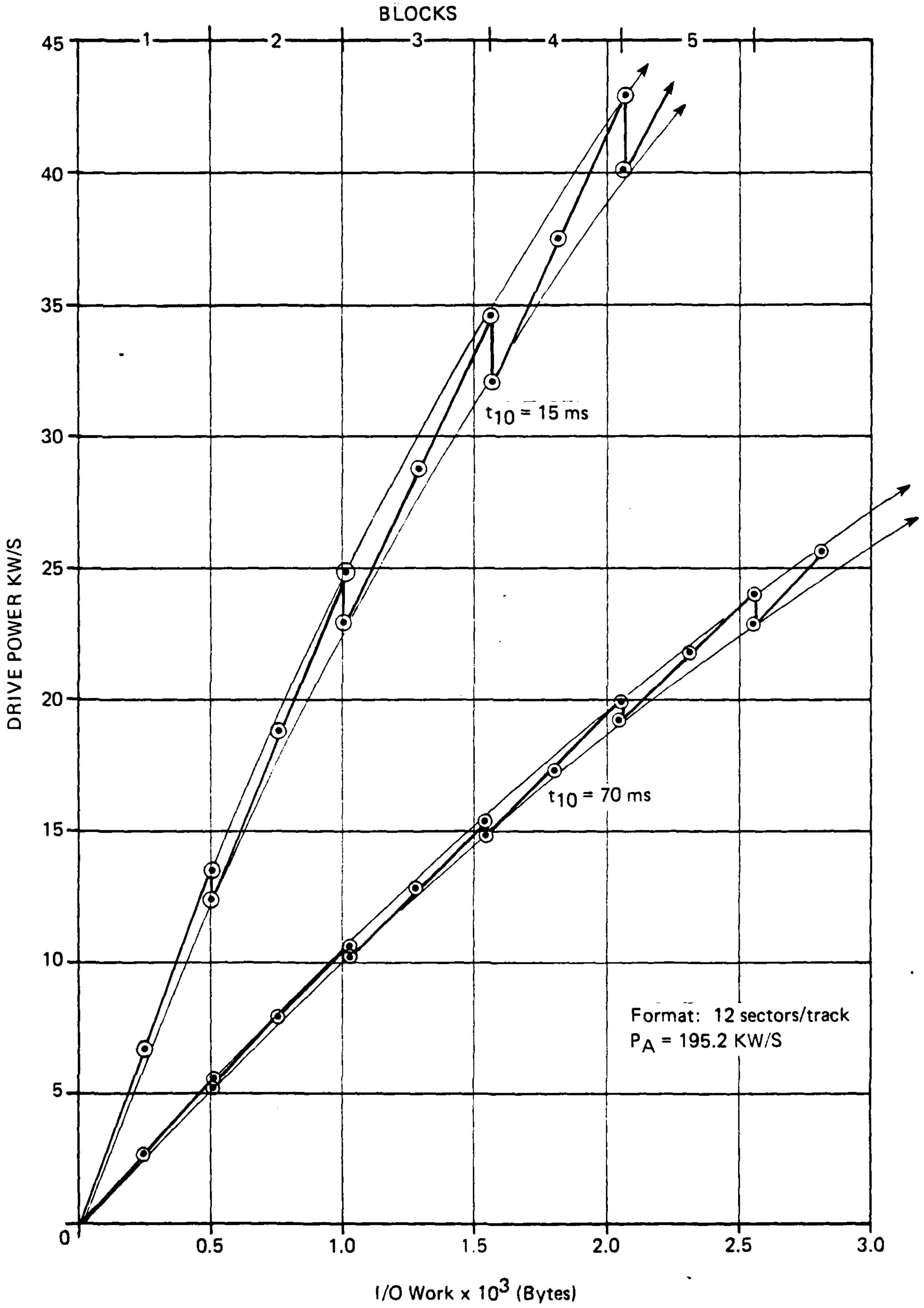
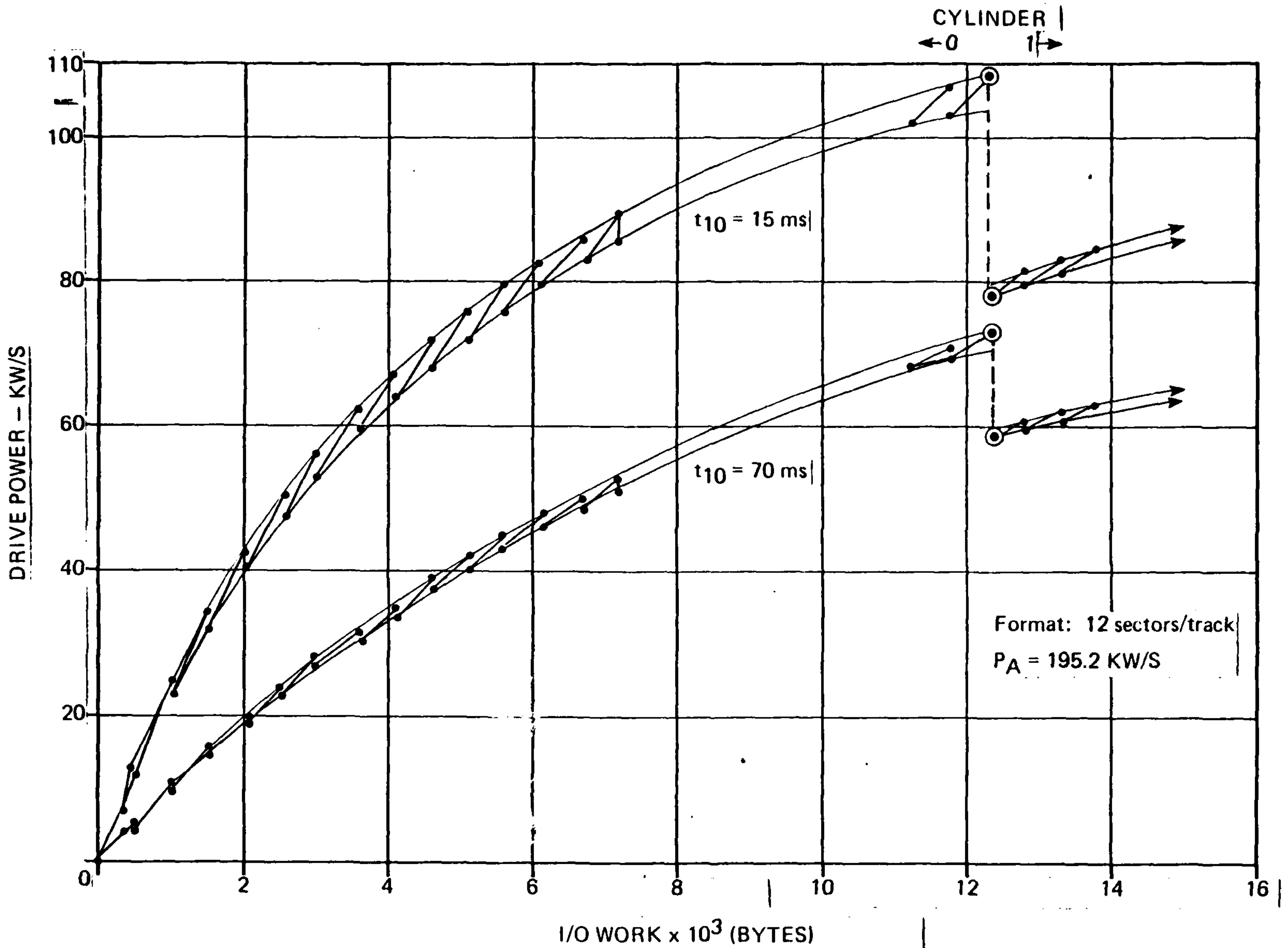


Figure 3.1a DIABLO SERIES 30 TYPE 2 DRIVE POWER (1)



DIABLO SERIES 30 TYPE 2 DRIVE POWER (2)

Figure 3.1b

3.3 AN/USH-23 CONTROL UNIT AN/UYK-20 CHANNEL POWER

We will develop the maximum theoretical power equations and curves for an AN/UYK-20 IOC/channel with multiple AN/USH-23 (Model 3500) controller disk β-configurations each with 1 to 8 Series 30 movable head disk drives.

3.3.1 Channel/Controller/Device States

Althoug'ı the model 3500 disk subsystem is at <u>any time</u> capable of accepting and initiating seeks to any drive not active, AN/UYK-20 IOC logic considers the controller unavailable when any drive on that controller is engaged in any of the following actions:

i) search (t_{20}, t_{21}, t_{22})

ii) read/write (t_3)

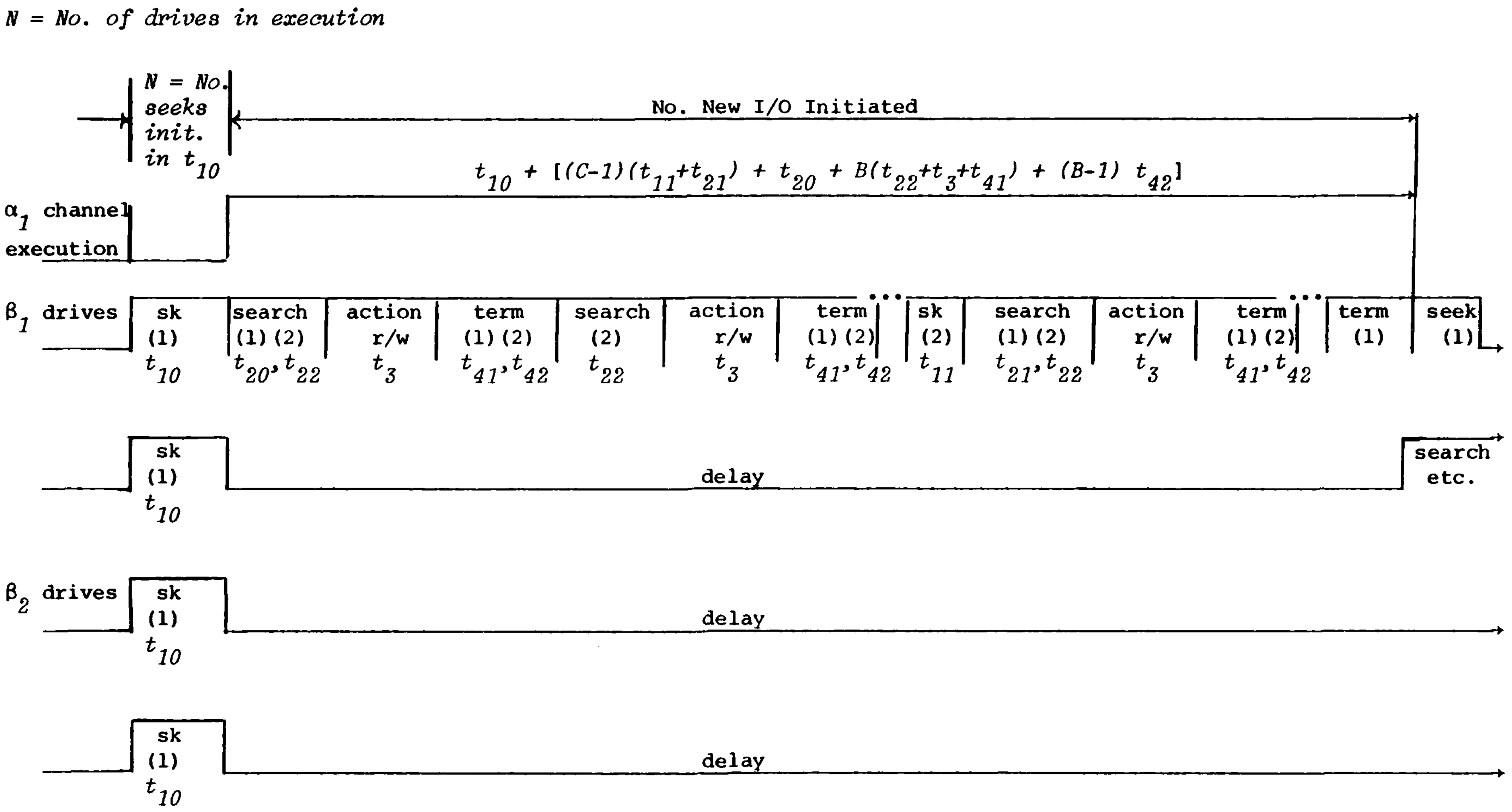
iii) termination (t_{41}, t_{42}) <u>except</u> last block t_{42} time.

iv) cylinder to cylinder reposition during multi-block read/write (t_{11}).

Figure 3.2 shows the channel, control unit and device states during the first composite standard cycle (i.e., the standard cycle that includes sector spanning).

The channel device is busy whenever the controller is, so the Software Physics Type 2 mode characterizes the identical control unit and channel configuration powers.

Note that when more than 4 drives are attached to a controller, daisy chaining is implied on one or more ports. This introduces the possibility of increased latency time when positioning to the starting sector on a daisy-chained drive, if it is not the first sector occuring on the track (i.e., sector offset $\sigma_0 \neq 0$). This is because in this mode, the platter index marker must be sensed prior to any search operation.


N = No. of drives in execution
N = No. seeks init. in t_{10}
No. New I/O Initiated
$t_{10} + [(C-1)(t_{11}+t_{21}) + t_{20} + B(t_{22}+t_3+t_{41}) + (B-1)\ t_{42}]$
α_1 channel execution
β_1 drives
sk (1) t_{10}
search (1)(2) t_{20}, t_{22}
action r/w t_3
term (1)(2) t_{41}, t_{42}
search (2) t_{22}
action r/w t_3
term (1)(2) t_{41}, t_{42}
sk (2) t_{11}
search (1)(2) t_{21}, t_{22}
action r/w t_3
term (1)(2) t_{41}, t_{42}
term (1)
seek (1)
sk (1) t_{10}
delay
search etc.
β_2 drives
sk (1) t_{10}
delay
sk (1) t_{10}
delay

AN/USH 23 (Model 3500) MOVABLE HEAD DISK

Channel State Chart - Type 2, Case 1 or Case 2 Startup

Figure 3.2

3.3.2 Power Equation Development

As a notational convenience, let

t_ϕ = *the composite standard cycle search, action and termination time*

$$= (C-1)(t_{11}+t_{21}) + t_{20} + B(t_{22}+t_3+t_{41}) + B(B-1)t_{42}$$

where C = *cylinder occupancy*

B = *block occupancy*

for a typical record of length $\overline{W}$

To obtain the maximum theoretical power with N drives active, assume that drive orders are always waiting and that (stand alone) seeks are issued whenever possible. We then have the following two cases:

3.3.2.1 Case 1: $t_{10} \leq (N-1)t_\phi$

In this case all steady state stand-alone seeks are concurrent with input/output actions on the other drives. (See Figure 3.3). We then have the steady state power

$$\overline{P}(\alpha, 3500, case\ 1) = \frac{\overline{W}}{t_\phi} \qquad (3.6)$$

since in a standard cycle $N\overline{W}$ bytes are transferred in time Nt_ϕ. Note that this power is as for a single drive with initial seek time $t_{10} = 0$.

3.3.2.2 Case 2: $t_{10} \geq (N-1)t_\phi$

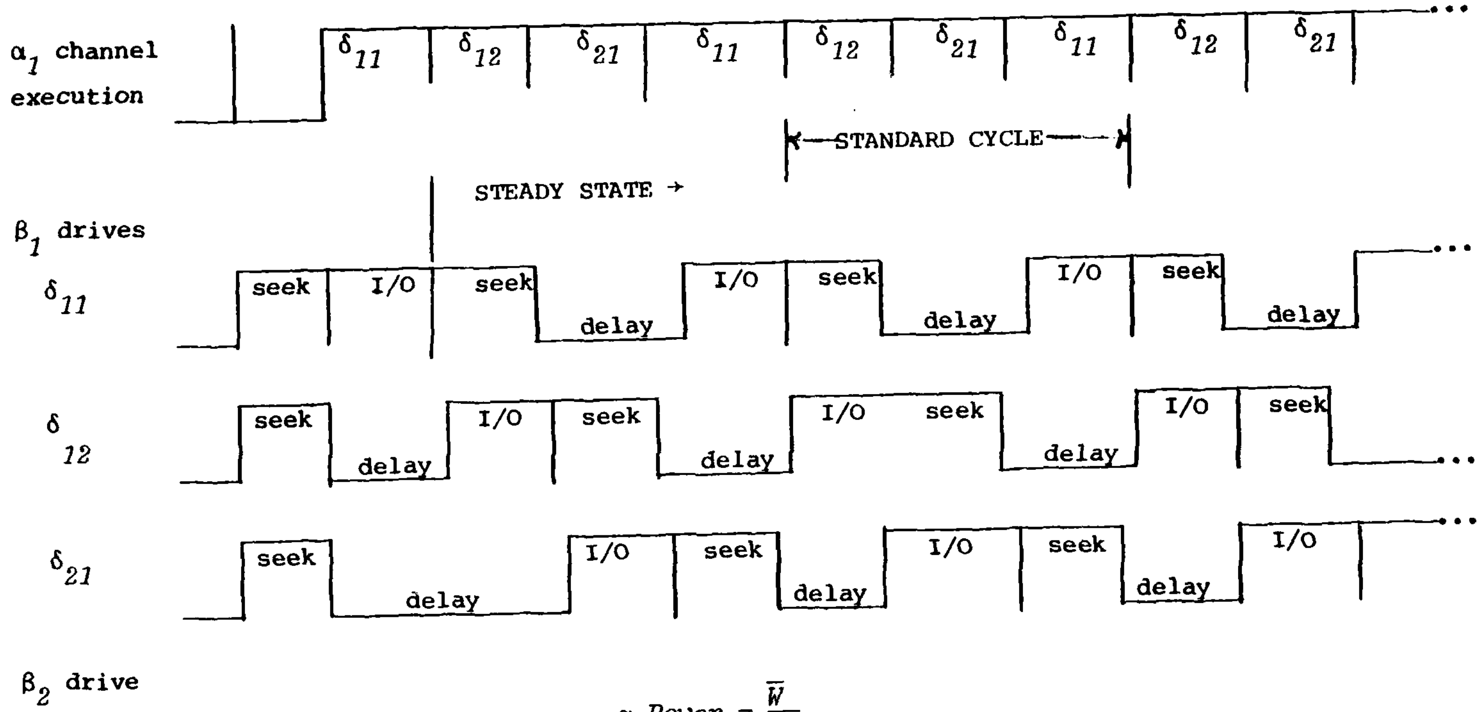
In this case $(N-1)$ composite input/output actions can be overlapped with the stand-alone seek. (See Figure 3.4).

We then have the steady state power:

$$P(\alpha, 3500, case\ 2) = \frac{N\overline{W}}{t_{10} + t_\phi} \qquad (3.7)$$

$t_{10} \leq (N-1)\ t_{\phi}$

N = Number of drives in execution



$$\alpha\text{-}Power = \frac{\overline{W}}{t_{\phi}}$$

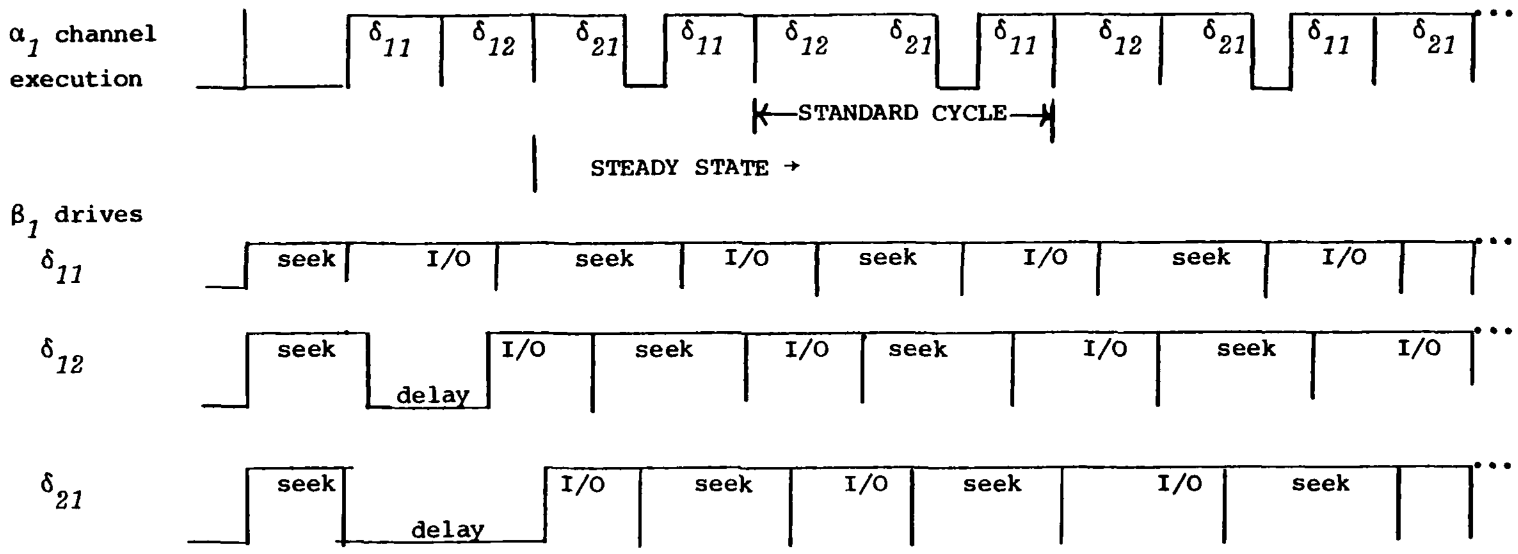
AN/UYK 20 IOC/TYPE 3500 DISK SUBSYSTEM (MOVABLE HEAD)

Channel Configuration State Chart (Type 2, Case 1)

Figure 3.3

$t_{10} \leq (N-1)\ t_{\phi}$

N = Number of drives in execution



β_2

$$\alpha\text{-}Power = \frac{N\overline{W}}{t_{10} + t_{\phi}}$$

AN/UYK 20 IOC/TYPE 3500 DISK SUBSYSTEM (MOVABLE HEAD)

Channel Configuration State Chart (Type 2, Case 2)

Figure 3.4

Note that when $t_{10} = (N-1)t_{\phi}$

$$P(\alpha, 31/33F, case\ 2) = \frac{N\overline{W}}{(N-1)t_{\phi} + t_{\phi}}$$

$$= \frac{N\overline{W}}{Nt_{\phi}} = \frac{\overline{W}}{t_{\phi}} = P(\alpha, 3500, case\ 1)$$

as expected.

3.3.3 α or β Configuration Power Data (AN/USH-23)

Table 3.2 and Figure 3.5 show the multiple drive powers obtained from Equations (3.6) and (3.7) for an average byte count $\overline{W} = 2048$. Since these powers are derived from average logical record lengths, uniform seek times across drives and an inexhaustible queue of disk orders, these values are to be considered as the average theoretical maximum powers.

We have for this logical record length:

$$B = 4,\ C = 1$$

$$t_{\phi} = (C-1)(t_{11}+t_{21}) + t_{20} + B(T_{22}+t_{3}+t_{41}) + (B-1)(t_{42})$$

$$= 0 + 20 + 4(0.14+2.611+0.01) + 3(0.524)$$

$$= 32.62\ msec.$$

The case 1 equation becomes:

$$t_{10} \geq (N-1)t_{\phi} \quad \text{i.e.,}\ t_{10} \leq (N-1)(32.62)$$

$$P(\alpha, 3500, case\ 1) = \frac{2048}{32.62} = 62.79\ kw/s$$

And for case 2:

$$t_{10} \geq (N-1)t_{\phi} \quad \text{i.e.,}\ t_{10} \leq (N-1)(32.62)$$

$$P(\alpha, 3500, case\ 2) = \frac{N \times 2048}{t_{10} + 32.62}$$

N	$t_{10} = 0$		$t_{10} = 15ms$		$t_{10} = 45ms$		$t_{10} = 70ms$		$t_{10} = 100ms$		$t_{10} = 135ms$	
	(Case)	$P(\alpha/\beta)$	(Case)	$P(\alpha/\beta)$	(Case)	$P(\alpha/\beta)$	(Case)	$P(\alpha/\beta)$	(Case)	$P(\alpha/\beta)$	(Case)	$P(\alpha/\beta)$
1	(1)	62.79	(2)	43.01	(2)	26.38	(2)	19.96	(2)	15.44	(2)	12.22
2	(1)	62.79	(1)	62.79	(2)	52.77	(2)	39.91	(2)	30.88	(2)	24.44
3	(1)	62.79	(1)	62.79	(1)	62.79	(2)	59.88	(2)	46.32	(2)	36.66
4	(1)	62.79	(1)	62.79	(1)	62.79	(1)	62.79	(2)	61.76	(2)	48.87
*5	(1)	62.79	(1)	62.79	(1)	62.79	(1)	62.79	(1)	62.79	(2)	61.09
*6	(1)	62.79	(1)	62.79	(1)	62.79	(1)	62.79	(1)	62.79	(1)	62.79
*7	(1)	62.79	(1)	62.79	(1)	62.79	(1)	62.79	(1)	62.79	(1)	62.79
*8	(1)	62.79	(1)	62.79	(1)	62.79	(1)	62.79	(1)	62.79	(1)	62.79

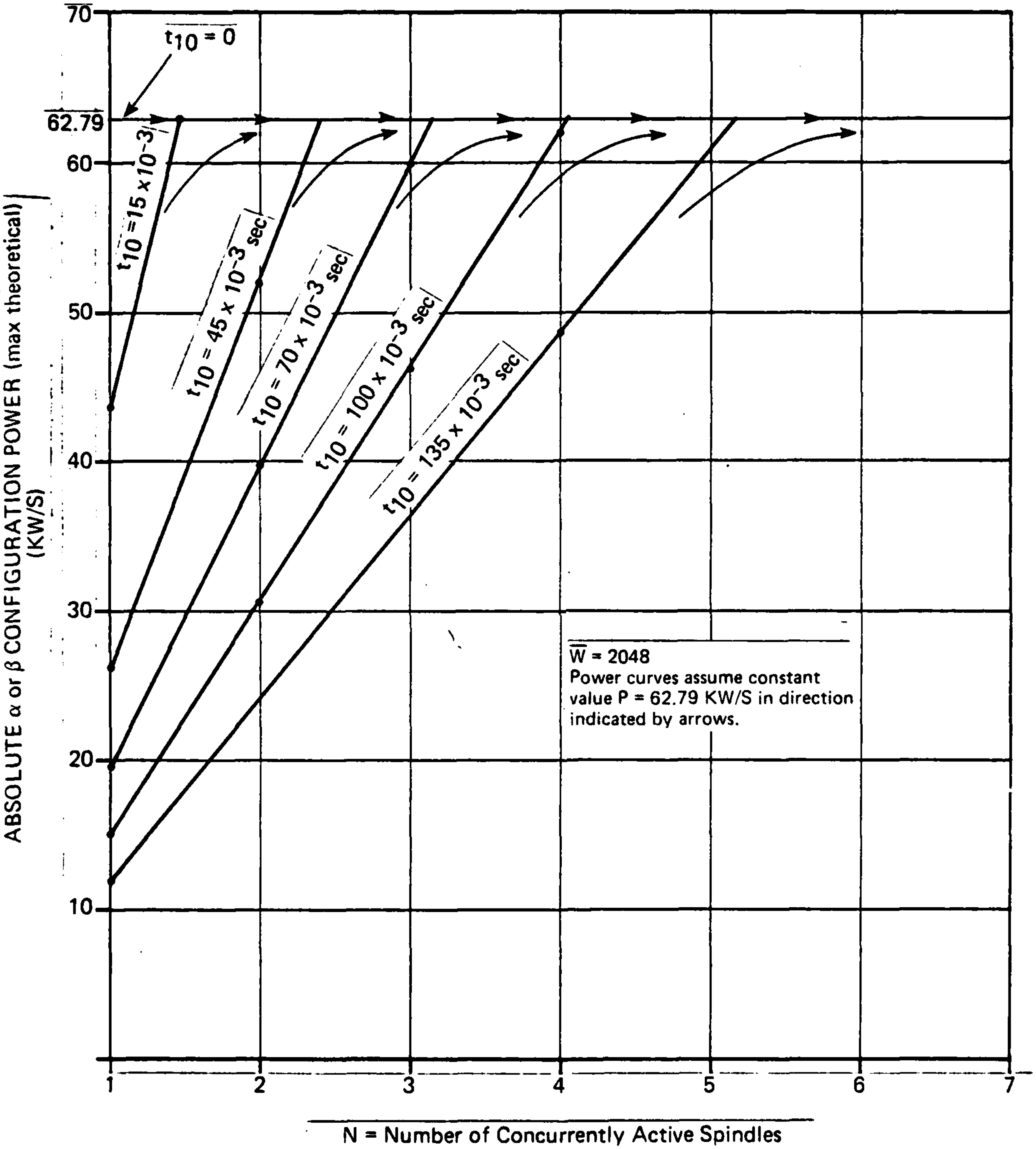
*Powers for $N > 4$ on a single controller are valid only when the block offset β_0 is zero - See text.

AN/USH-23 (Model 3500) DISK SUBSYSTEM

(with Series 31/33F Drives)

α or β Configuration POWER ($\overline{W}=2048$)

TABLE 3.2



$\overline{W} = 2048$
Power curves assume constant value P = 62.79 KW/S in direction indicated by arrows.

Figure 3.5 AN/USH-23 – MODEL 3500 DISK SUBSYSTEMS
α or β CONFIGURATION POWER (MAX)
AVERAGE BLOCKSIZE = 2048 BYTES

3.3.4 α,β Configuration Powers as a Function of Seek Time (AN/USH-23)

For one or more Model 3500 β configurations we wish to show absolute α or β configuration power as a function of the initial seek time, t_{10} We do this by noting that for any number of spindles we have:

$$\overline{P}(\alpha,3500) = \overline{P}(\alpha,3500,case\ 1) = constant$$

$$\text{when } t_{10} \leq (N-1)\ t_{\phi}$$

and we determine the points on the line

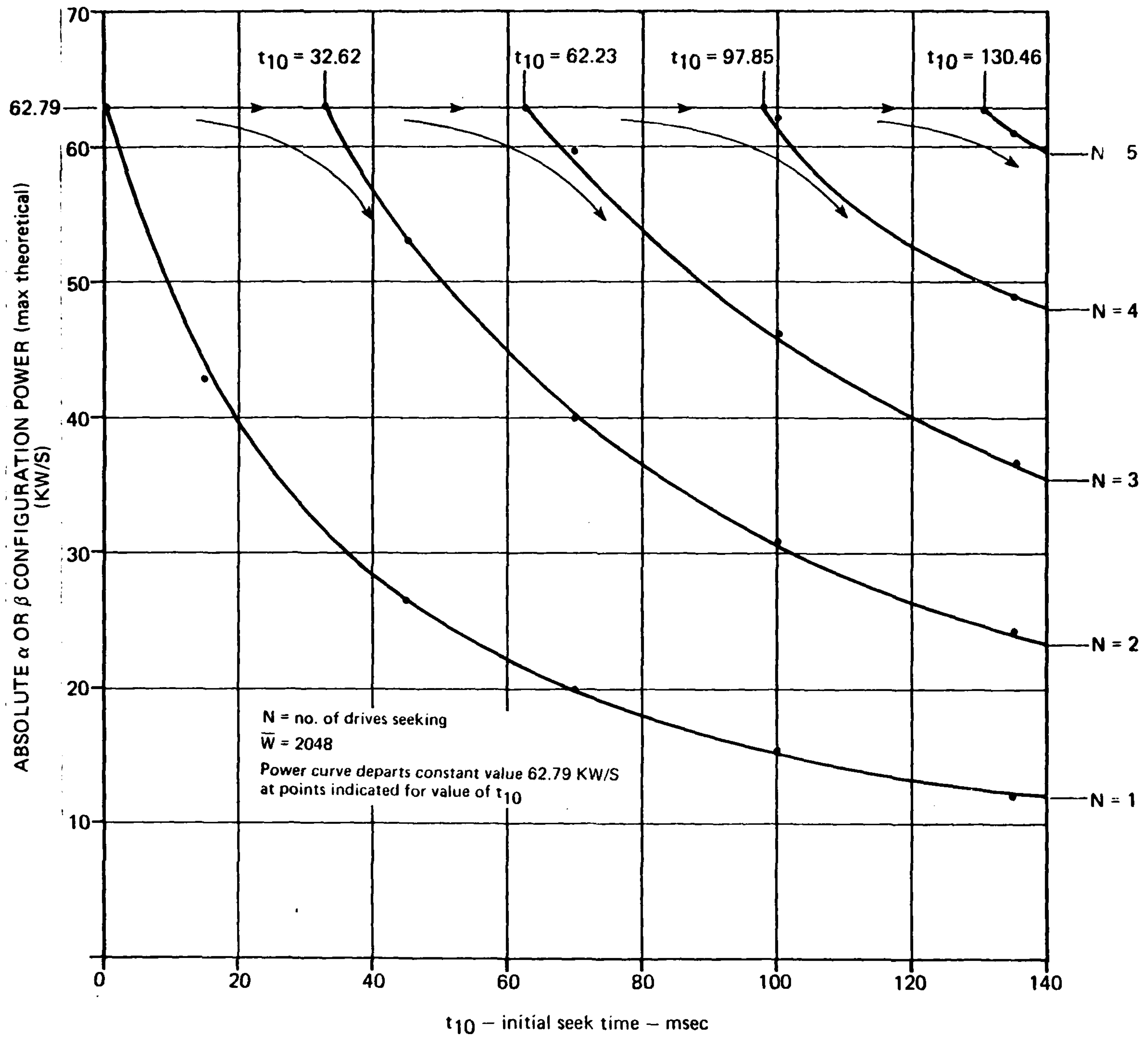
$$\overline{P}(\alpha,3500) = \overline{P}(\alpha,3500,case\ 1)$$

intercepted by the functions

$$\overline{P}(\alpha,3500) = \overline{P}(\alpha,3500,case\ 2) = \frac{N\overline{W}}{t_{10} + t_{\phi}}$$

by setting $\frac{N\overline{W}}{t_{10} + t_{\phi}} = \overline{P}(\alpha,3500,case\ 1)$

These functions are shown plotted as a function of the seek time in Figure 3.6 for a value of $\overline{W} = 2048$. The value of $P(\alpha,3500)$ maintains the case 1 constant value (62.79 kw/s) until the indicated intercepts for each N. Note that for $N > 5$, the intercept value of t_{10} exceed the device maximum of 135 msec, so that the device powers remain at the case 1 value for all values of t_{10}.



MODEL 3500 α or β CONFIGURATION POWER – FUNCTION OF INITIAL SEEK TIME AT AVERAGE BLOCKSIZE = 2048

Figure 3.6

SECTION 4

AN/USQ-69 (ADD DISPLAY) POWER

4.1 GENERAL

The AN/USQ-69 Alphanumeric Digital Data (ADD) Display is a keyboard/ CRT device capable of data entry to and data display from the AN/UYK-20 computer. Modes of operation include a block burst mode for input or display and an input only character mode. It is the first of these modes which will provide us the basis for a richer analysis.

4.2 AN/USQ-69 ADD DEVICE CHARACTERISTICS

(a) Input/Output Modes:

i) Burst mode (Input or Display):

Block transfers to or from internal memory
up to 2000 characters (standard)
up to 6000 characters (optional)

ii) *Character mode (Input only)*

(b) Display:

i) Capacity: 2000 characters (25 lines @ 80 characters)

ii) Refresh: Period - 16.7 msec. Time - 1.36 msec.

(c) Interfaces:

i) Parallel

MIL-STD 1397 (A),(B) or (C) parallel channels in 8 bit mode.

ii) Serial

MIL-STD-188 or EIA-STD-RS232C
Serial asynchronous channels @ 2400 baud
MIL-STD-188 Serial synchronous channels @ 9600 baud

(d) Configuration:

i) Each ADD input/display includes a dedicated controller.

ii) Up to 8 displays may be daisy-chained on one asynchronous serial channel.

4.3 ADD Input (keyboard) Power

4.3.1 Burst Mode

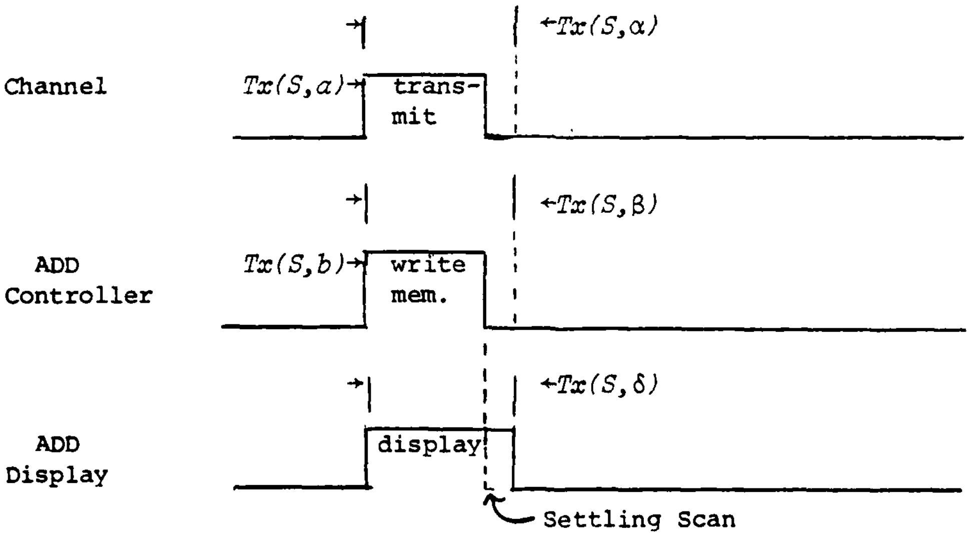
For any of the specified interfaces, transfer times from the ADD device buffer to the AN/UYK-20 are small compared to execution times at the ADD device and controller level; that is, the time required to key a block message. This latter quantity is, of course, extremely variable and will not be assessed here.

4.3.2 Character Mode

The slowest interface constrains the character transmission rate to 2400/8 = 300 characters per second so, here too, it is operator key-in rates and functions that effectively determine device power.

4.3.3 ADD Output (Display) Power

(a) The output state chart of Figure 4.1 shows a single transmit-display cycle:



AN/USQ-69 DISPLAY STATE CHART

Figure 4.1

During a transmit-display cycle, the controller becomes busy for the time that data is transmitted to ADD memory. The display execution time, $Tx(S,\delta)$, is given as the time required for full settling of the display from start of memory rewrite. This is estimated to be the memory rewrite time plus 1.36 msec, this latter quantity being the time required to scan a single CRT frame. Subsequent screen refreshes are not considered as part of device execution.

We thus have the following equation for the theoretical maximum ADD display output power:

$$\overline{P}(ADD\ display) = \frac{\overline{W}}{W \div MBTR_a + 1.36}\ KW/S \qquad (4.1)$$

where $\overline{W}$ is the average block length transmitted

and $MBTR_a$ is the maximum byte transfer rate of the channel-interface device.

4.3.3.1 Parallel Channels (8 bit mode)

(a) For the MIL-STD-1397 type A interface:

$$P(ADD\ Display[1397(A)]) = \frac{\overline{W}}{W \div 41.6 + 1.36}\ KW/S \qquad (4.2)$$

Table 4.1 and the graph of Figure 4.2 show ADD device power for block lengths of up to a full screen (2000 bytes.)

Block Length (Bytes)	Execution Time (10^{-3} sec)	ADD Display Power KW/S	% Block Length Efficiency
80	3.28	24.4	58.7
160	5.21	30.7	73.8
240	7.13	33.7	81.0
400	11.0	36.4	87.5
800	20.6	38.9	93.5
1200	30.2	39.7	95.4
1600	39.8	40.2	96.6
2000	49.4	40.5	97.4

AN/USQ-69 DISPLAY POWER
MIL-STD-1397 (A) INTERFACE

Table 4.1

(b) For the MIL-STD-1397 B and C interfaces:

$$P(ADD\ display)[1397(B,C)]) =$$

$$\frac{\overline{W}}{W \div 190 + 1.36}\ KW/sec \qquad (4.3)$$

This equation is tabulated in Table 4.2 and plotted in Figure 4.3.

Block Length (Bytes)	Execution Time (10^{-3} sec)	ADD Display Power KW/S	% Block Length Efficiency
80	1.78	44.9	23.6
120	1.99	60.3	31.7
240	2.62	91.5	48.2
400	3.47	115.4	60.7
800	5.57	143.6	75.6
1200	7.68	156.3	82.3
1600	9.78	163.6	86.1
2000	11.9	168.3	88.6

AN/USQ-69 DISPLAY POWER
MIL-STD-1397 (B), (C) INTERFACES

Table 4.2

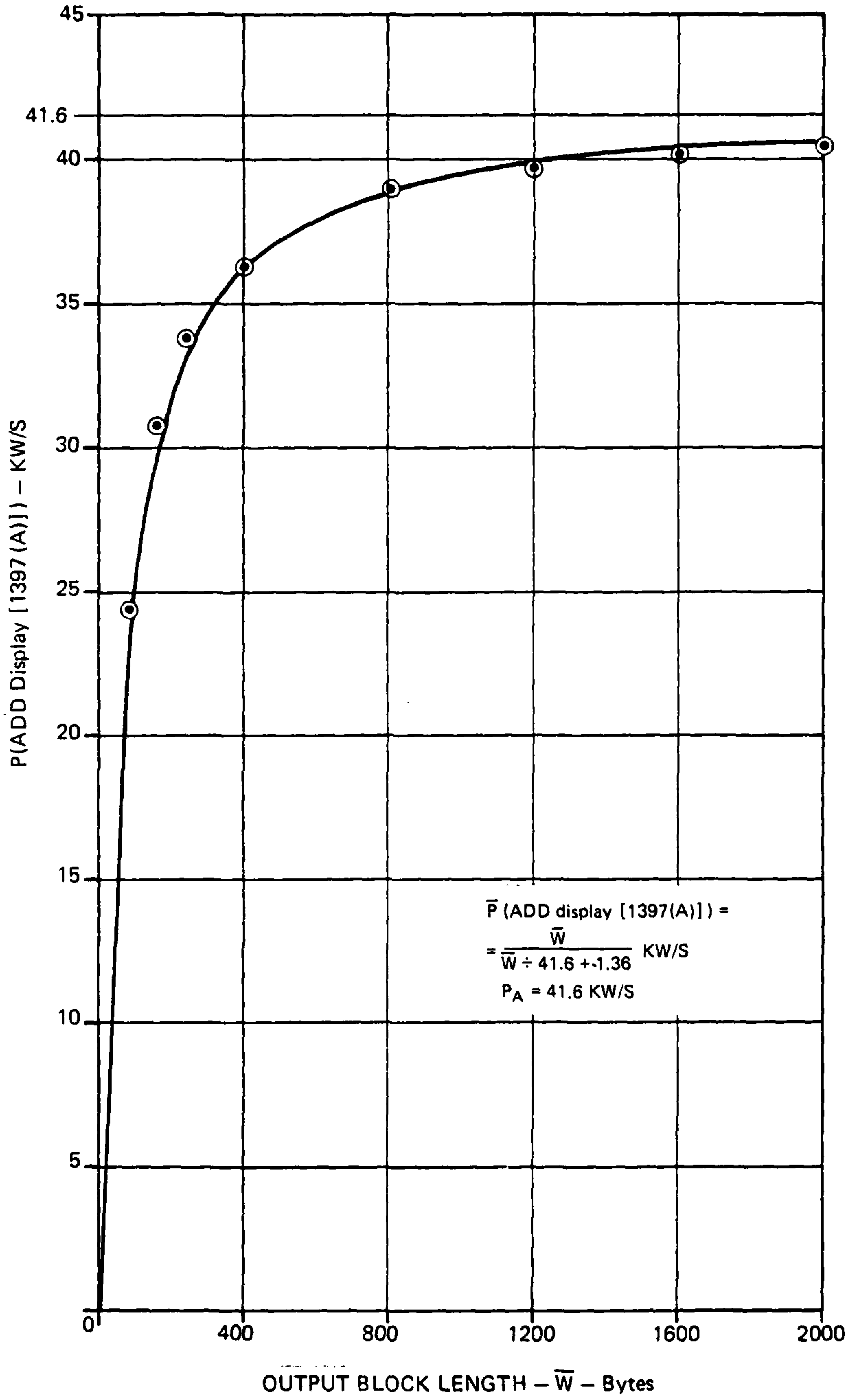
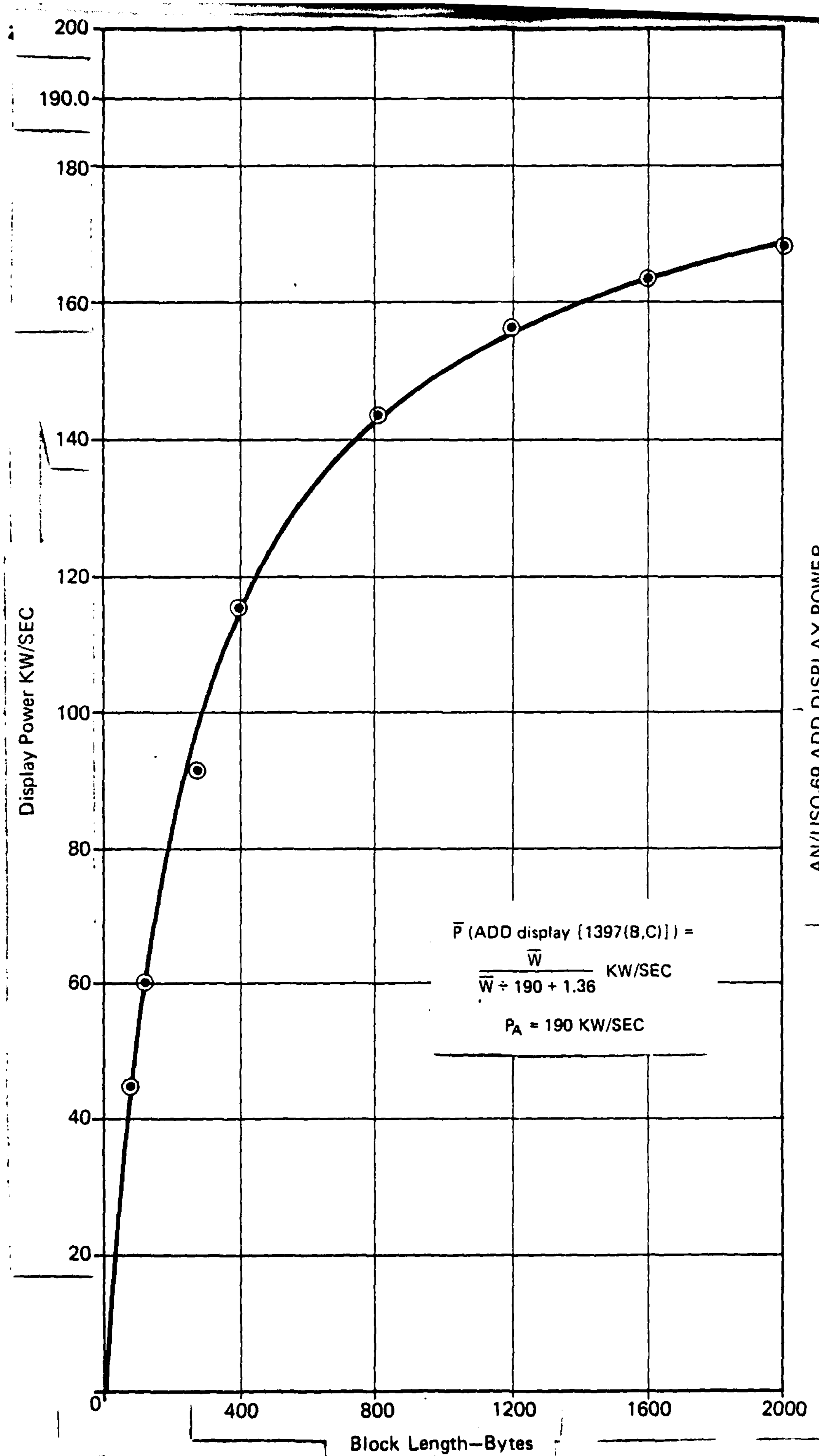


Figure 4.2 AN/USQ-69 ADD DISPLAY POWER
MIL-STD-1397 TYPE A INTERFACE

ADD Display Power—MIL-STD-1397 Types B & C Interfaces



AN/USQ-69 ADD DISPLAY POWER
MIL-STD-1397 TYPES B & C INTERFACES

Figure 4.3

(c) Serial Channels (8 bit mode)

For the EIA-STD-RS232C & MIL-STD 188C Serial Interfaces in asynchronous mode at their maximum rate (2400 bits/sec)

$$P(ADD\ display\ [RS232,asynch]) =$$

$$\frac{\overline{W}}{W \div 0.3 + 1.67} \simeq 0.3\ kw/sec \text{ (for range 80-2000 bytes)}$$

and in synchronous mode at their maximum rate of 9600 baud:

$$P(ADD\ display\ [RS232C,synch]) =$$

$$\frac{\overline{W}}{W \div 1.2 + 1.67} \simeq 1.2\ kw/sec \text{ (for range 80-2000 bytes)}$$

4.3.4 α,β Configuration Powers (AN/USQ-69 ADD DISPLAY)

4.3.4.1 ADD Control Unit Power

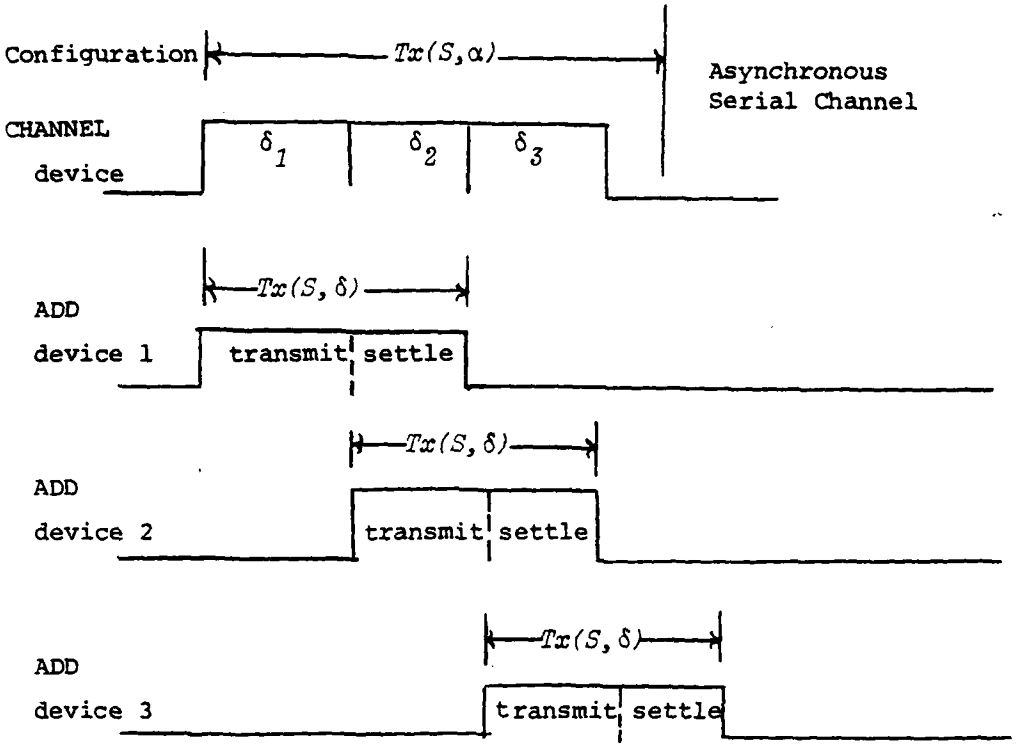
Since the ADD control unit is a device dedicated to a single ADD display it thus exhibits power characteristics identical to the ADD display itself.

4.3.4.2 ADD Channel Configuration Power

Here we have a single special case to be considered; that is, the daisy-chaining of two to eight ADD displays from a single asynchronous channel. We examine α power for a single set of transmissions to each device. The state chart of Figure 4.4 shows the settling scan of the first N-1 devices coincide with data transmission to subsequent devices. So, for N ADD devices daisy-chained on an asynchronous serial channel (MIL-STD-188/EIA-STD-RS232C), and with each unit to receive a message of average length $\overline{W}$ characters, we have for maximum theoretical channel configuration power:

$$\overline{P}(\alpha,ADD\ display[188/RS232C]) = \frac{N\overline{W}}{NW \div MBTR + 1.36}$$

$$= \frac{N\overline{W}}{(N\overline{W}\div 300) \times 10^{-3} + 1.36} \qquad (4.4)$$



AN/USQ-69 CHANNEL STATE CHART

Figure 4.4

Note that for blocksizes $\geq$ 80 (one line) we have that $(NW \div 300) \times 10^3 >> 1.36$ for all N. We may thus conclude that in block mode, it is this channel rate (300 characters per second) that determines the maximum power (0.300 kw/s.)

SECTION 5

AN/UYK-20 CHANNEL DEVICE AND CONFIGURATION POWERS

5.1 GENERAL

The AN/UYK-20 performs input/output activity through an incorporated input/output controller (IOC) which operates substantially independent of the CP.

Each IOC-peripheral parallel mode interface consists of an output channel to transmit data and control functions to the peripheral device. Input channels are used to receive data or interrupt codes from the external device. All parallel mode input/output activity is asynchronous, with the timing (and hence power) dependent on the speed of the peripheral device.

Serial I/O channels are also available for communications circuits which operate in either synchronous or asynchronous modes. The IOC performs all necessary serial-to-word and word-to-serial conversions.

5.2 CHANNEL DEVICE CHARACTERISTICS AND POWERS

5.2.1 Parallel I/O Channels

These are supplied in groups of 4 input and 4 output channels operating in the full duplex mode, permitting concurrent input and output. Furthermore, these channels may be operated in byte (8 bit), single (16 bit), or dual (32 bit) modes, the last requiring the use of a channel n and $n + 4$, i.e., the use of 2 groups. Maximum transfer rates and powers are given in Table 5.1 for the 16 and 32 bit word modes for input or output. Rates and powers are also given for concurrent input/output computed as 1.75 times the unidirectional rate* subject to a maximum of 1,000,000 16-bit words/sec; i.e., 2000 kw/s.

* ref. SPERRY-UNIVAC PX 11772

Interface & Voltage (mode)		Max. Word Rate (k words/sec)/Power (KW/sec.)			
		NUMBER OF CHANNELS ACTIVE:			
		1-4	5-8	9-12	13-16
MIL-STD-1397 TYPE A					
NTDS Single - 15V	In or out:	41.6/83.3	83.3/166.6	125/250	166.6/333.3
(16 bit) *	In & out:	72.8/145.8	145.8/291.7	218.8/437.5	291.7/583.3
MIL-STD-1397 TYPE A					
NTDS Dual - 15V	In or out:	40.5/162	81/324	-	-
(32 bit)	In & out:	70.9/283.5	141.8/567		
			In: 444/888	In: 570/111	In: 889/1778
MIL-STD-1397 TYPES B & C ANEW + 3.5 V &	In or out:	190/380	Out: 333/666	Out: 570/1140	Out: 666/1332
NTDS - 3.0 V Single (16 bit)	In & out:	332.5/665	665/1330	997.5/1995	1000/2000
MIL-STD-1397 TYPES B & C					
ANEW + 3.5 V &	In or out:	167/334	380/760	-	-
NTDS - 3.0 V Dual (32 bit)	In & out:	292.3/584.5	665/1330		

* For 8 bit mode the power in KW/S is the same as the word rate in K words/sec

AN/UYK-20 PARALLEL CHANNEL INTERFACE MAXIMUM I/O RATES AND POWERS

Table 5-1

5.2.2 Serial I/O Channels

These are provided in 2 channel groups. Serial-to-word and word-to-serial conversions are performed by the AN/UYK-20 IOC.

Maximum rates and powers for the various interfaces are as follows:

(a) NTDS SERIAL CHANNEL:

125,000 32 bit words/sec equivalent to power <u>500 kw/sec</u>.

(b) EIA-STD-RS232C and MIL-STD-188C

SERIAL CHANNELS:

Asynchronous:	2400, 1200, 600, 300, 150 or 75 bits/sec equivalent to powers 300, 150, 75, 37.5, 18.75 works/sec.
Synchronous:	Up to 9600 baud; i.e., equivalent to 1200 works/sec.

5.3 CHANNEL CONFIGURATION POWER

5.3.1 Discussion

Maximum theoretical channel configuration powers depend on the kinds of devices attached and for this reason these powers were developed along with the powers for the devices and their controllers.

In general, however, note that:

(a) Type 1 (e.g., tape) devices produce channel powers equivalent to those of a single drive.

(b) Type 2 (e.g., disk) drives produce channel powers which depend on the number of drives concurrently active up to a limit value beyond which the addition of drives adds *no more* to the maximum theoretical power.

(c) IOC configuration (that is, the total input/output configuration power) is obtained by simple algebraic addition of the individual channel powers developed in those sections pertaining to attached devices. This sum is, however, limited to the constraint that total IOC power cannot exceed 2000 KW/sec.

Finally, it must again be noted that the computed IOC configuration power being the sum of maximum theoretical powers is itself a theoretical maximum and will thus not be achieved in practice except instantaneously.

SECTION 6

AN/UYK-20 CP POWER

6.1 INTRODUCTION

In this section, we will develop a methodology and notation for expressing AN/UYK CP power in terms of the power of individual instructions or classes of instructions. Work performed for instruction setup will be considered as well as work done on operands (data).

The final subsections will consider the effects of concurrent IOC and DMA facility operation on CP power; that is, we will express CP capacity when executing certain instructions and instruction classes in terms of concurrent IOC or DMA power. The resulting equations for I/O activity degraded CP power will be shown tabulated and graphed for an instruction mix considered typical by the manufacturer.

6.2 AN/UYK-20 CP ARCHITECTURE AND CHARACTERISTICS

The AN/UYK-20 CP is, in actuality, emulated by a microprogrammed controller (MPC), a set of registers, and a two-bus data exchange structure. Thus the execution of AN/UYK-20 instructions res·.lts in the execution of microprogrammed code with data and control bits shuttled to and from the data and program registers and main memory via the source and destination buses.

Some pertinent CP and memory access characteristics are:

(a) Instruction formats - lengths

RR (Register/Register) - 16 bit
RI (Register/Indirect Memory) - 16 bit
RK (Register/Literal Constant) - 32 bit
RX (Register/Indexed Address) - 32 bit

(b) 16 general purpose registers provided @ 16 bits.

(c) MPC cycle time - 155 ± 5 nsec.

(d) Direct addressing to 65K words.

(d) Cascaded indirect addressing:

Each level of cascade requires double word fetch.

(f) Overlapped fetch on certain instructions.

(g) Memory access cycle - without DMA - 750 ± 10 nsec.
- with DMA - 790 ± 10 nsec. max.

(h) DMA access through additional ports on each 32K work bank of memory.

(i) CP has priority over DMA for main memory access.

(j) IOC has priority over CP for main memory access.

6.3 AN/UYK-20 CP POWER DERIVATION

6.3.1 CP Software Work - Force Vectors

For our purposes, CP work for a given workload, L, may be categorized in terms of the domains and ranges of action by:

(a) $W(L,\gamma)_M$ - CP/memory work

(1 unit for every byte transferred between the cpu [or more specifically a cpu register] and memory.)

(b) $W(L,\gamma)_R$ - register/register work

(1 unit for every byte transferred between cpu registers.)

Another distinction of importance is that between the kind of CP work done when the CP is in states performing:

(a) Setup/termination work - effectively, the work of instruction fetch.

Denoted: $W_I(L,\gamma)_M$, $W_I(L,\gamma)_R$

(b) Control function work - setting of control registers that can be tested by the running code. This is a type of register/register work denoted:

$$W_C(L,\gamma)_R$$

(c) Data transfer work - operand (data) fetches and stores, operand actions as per instruction definition:

Denoted: $W_D(L,\gamma)_M$, $W_D(L,\gamma)_R$.

We will be concerned, at this time, mainly with CP/memory work, as register to register work can be thought of as <u>internal work</u> and generally represents a constant fraction of the CP/memory work. For simplicity we will denote CP/memory work by $W(L,\gamma)$, or by $W_I(L,\gamma)$ or $W_D(L,\gamma)$ when the instruction or data states are to be distinguished. Note that because of the extensive property of software work:

$$W(L,\gamma) = W_I(L,\gamma) + W_D(L,\gamma)$$

6.3.1.1 Software Containers

The CP instructions and operands are represented and manipulated in portions of storage or registers called containers. For example, an instruction that alters a 16-bit word as data is said to perform work on that container. The quantity of work performed on the container is 2 works, consistent with the software physics definition of 1 work for each 8 bit byte with changed symbol state.

Tables 6.1a and 6.1b show an assignment of codes to the various AN/UYK-20 containers. Note that these are grouped by classes derived from container functions and location. The codes have been formulated so that the last digit is the container length in bytes (8 bit units). Digits to the right of the decimal are read as eighths of a byte, that is, bits.

These container codes will provide a notational convenience when we speak of instructions which map operands from a domain to a range, each being defined by a container type. Note that instructions fetches as well are interpreted as work done on a type of container.

CLASS	CONTAINER	CODE
Storage	Bit	0.1
	Literal	0.4
	Byte	1
	Single Word	12
	Double Word	14
	Float Double	24
	Triple Word	16
	Interrupt Area	38
	IOC Command Cells	42
	IOC External Interrupt Area	49
Data Register	General	102
	General - odd	112
	General - even	122
	General - pair even-odd	134

NOTE: Container length indicated by final digit value. Fractions are eighths of a byte (i.e. bits)

AN/UYK-20 DATA CONTAINERS & CODES

PART I

Table 6.1a

CLASS	CONTAINER	CODE
Control Register	Program Addr	202
	Status 1:	318G
	DMA	310.1
	Interrupt I,II,III	311.1
	FP Round	312.1
	FP Interrupt	313.1
	Condition Code	314.1
	Overflow	315.1
	Carry	316.1
	NDRO	317.1
	Stack	318.1
	Status 2:	328G
	Interrupt Code	321
	Indirect Control	320.2
	Memory Address	902
Instruction Register	Instruction	500G
	m-field	501.4
	a-field	502.4
	y-field	512

NOTE: Container length indicated by final digit value. "G" suffix indicates group. Fractions are eighths of a byte (i.e. bits)

AN/UYK-20 DATA CONTAINERS & CODES
PART II

Table 6.1b

6.3.1.2 CP Software Force Vector Diagrams

The vector nature of CP software work can be illustrated for individual CP instructions or potentially for sequences of instructions by a graphical device which we will call a Software Force Vector Diagram. The basic form for these diagrams is shown in Figure 6.2. The container types listed on the left (or bottom) of the diagram are for the domain of the mapping action of an instruction, while those on the right (or top) are for the range of the mapping. Directed line segments of types to be listed below are drawn from domain containers to range containers. These indicate a directed software force acting on the domain containers. The couplet (c_1,c_2) composed of the domain and range container codes indicate the direction of the force denoted by:

$$f_{t,(c_1,c_2)}$$

where $t = I, D, C$ depending on the type of work performed:

I - Instruction work (fetches, indirect addressing)

D - Data transfer work (operands)

C - Control function work (program-accessed control registers)

The software force of an instruction can thus be represented by the vector:

$$\vec{F} = [f_{t,(c_i c_j)}]$$

where t varies over work types and c_i, c_j vary over all container codes.

Work is done when the Software Force acts through a distance $\vec{h}(c_i, c_j)$ whose magnitude is the length of c_j in bits ÷ 8, denoted h_j. For CP/memory work, we have the scalar quantity, work, defined by:

$$W = \vec{F}\cdot\vec{h} = f_{I,(c_{11},c_{12})}\cdot h_{12} + f_{I,(c_{21},c_{22})}\cdot h_{22} +$$

$$\cdots + f_{D,(c_{11},c_{12})}\cdot h_{12} + f_{D,(c_{21},c_{22})}\cdot h_{22} + \cdots$$

$$= \sum_i [(f_{I,(c_{i1},c_{i2})} + f_{D,(c_{i1},c_{i2})})\cdot h_{i2}]$$

$$= W_I(L,\gamma) + W_D(L,\gamma)$$

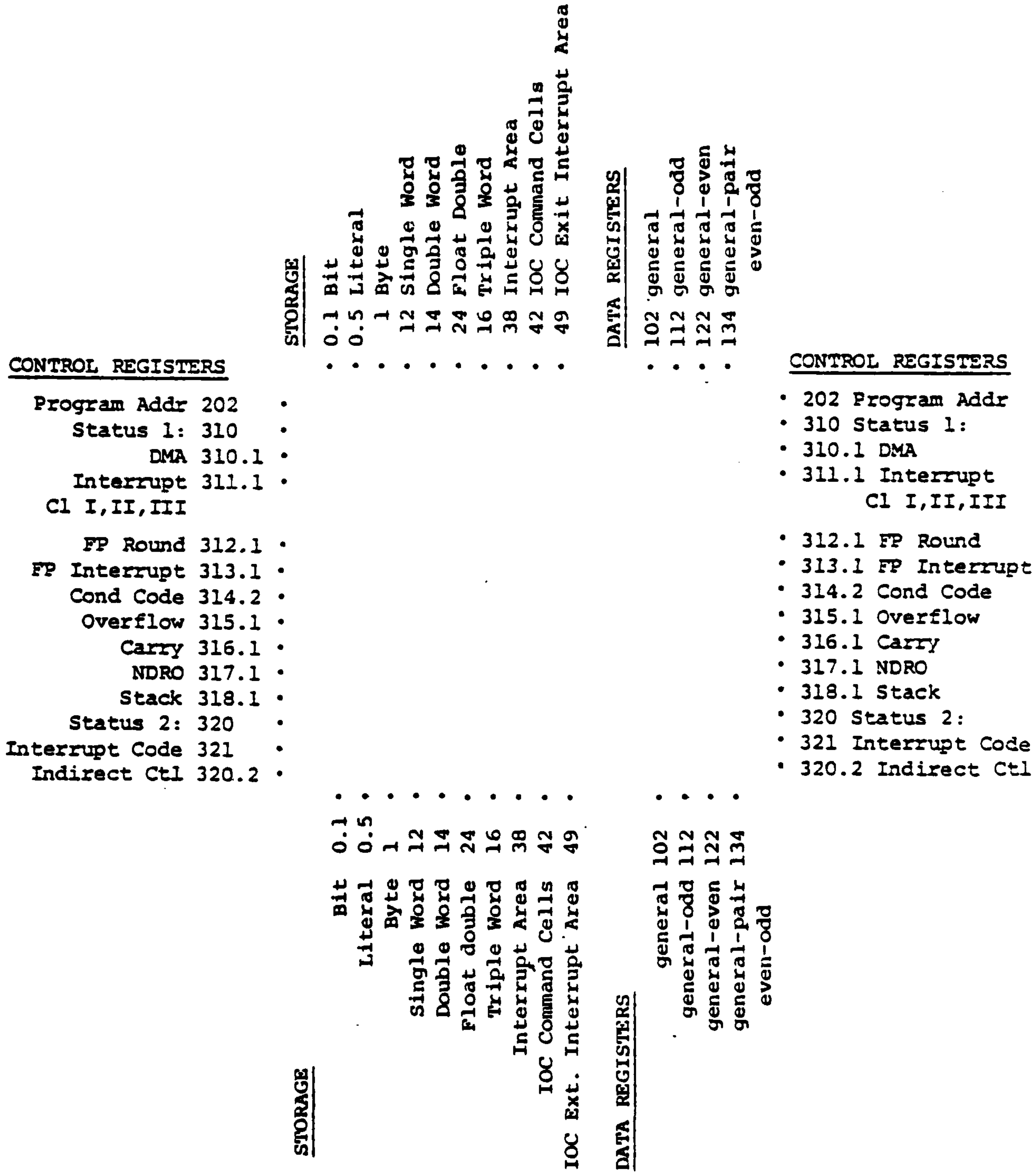
Table 6.2 summarizes the types of $W(L,\gamma)$ corresponding to the states I, D and C, their notation and graphic symbols.

CPU SOFTWARE FORCE VECTORS

AN/UYK-20 Repertoire

Instruction:
OP:
Mnemonic:
Format:
Tx(μsec):

Description:



STORAGE
- 0.1 Bit
- 0.5 Literal
- 1 Byte
- 12 Single Word
- 14 Double Word
- 24 Float Double
- 16 Triple Word
- 38 Interrupt Area
- 42 IOC Command Cells
- 49 IOC Exit Interrupt Area

DATA REGISTERS
- 102 general
- 112 general-odd
- 122 general-even
- 134 general-pair even-odd

CONTROL REGISTERS
- Program Addr 202
- Status 1: 310
- DMA 310.1
- Interrupt 311.1 Cl I,II,III
- FP Round 312.1
- FP Interrupt 313.1
- Cond Code 314.2
- Overflow 315.1
- Carry 316.1
- NDRO 317.1
- Stack 318.1
- Status 2: 320
- Interrupt Code 321
- Indirect Ctl 320.2

CONTROL REGISTERS
- 202 Program Addr
- 310 Status 1:
- 310.1 DMA
- 311.1 Interrupt Cl I,II,III
- 312.1 FP Round
- 313.1 FP Interrupt
- 314.2 Cond Code
- 315.1 Overflow
- 316.1 Carry
- 317.1 NDRO
- 318.1 Stack
- 320 Status 2:
- 321 Interrupt Code
- 320.2 Indirect Ctl

STORAGE
- Bit 0.1
- Literal 0.5
- Byte 1
- Single Word 12
- Double Word 14
- Float double 24
- Triple Word 16
- Interrupt Area 38
- IOC Command Cells 42
- IOC Ext. Interrupt Area 49

DATA REGISTERS
- general 102
- general-odd 112
- general-even 122
- general-pair 134 even-odd

CP SOFTWARE FORCES - BASIC DIAGRAM

Figure 6.2

DOMAIN / RANGE	SYMBOL	FUNCTION	GRAPHIC
cpu/memory	$W_I(L,\gamma)$	instruction - fetch	→
cpu/memory	$W_D(,\gamma)$	operands - indirect address generation	******→
cpu/memory	$W_D(L,\gamma)$	data transfer (operands)	————→
register/register	$W_D(L,\gamma)_R$	operands - address generation	•——•——•→
register/register	$W_D(L,\gamma)_R$	data transfer (operands)	x——x——x→
register/register	$W_C(L,\gamma)_R$	control function	••••••••••→

CPU SOFTWARE WORK QUANTITIES

Decomposition of $W(L,\gamma)$

Table 6.2

The series of Figures 6.3 show examples of completed CP Software Work Vector diagrams with directed line segments indicated as follows:

(a) cpu/memory work - instruction (fetch)
(dashed line)

(b) cpu/memory work - instruction (indirect address generation - one doubleword [4 bytes] fetched for each level)
(starred line)

(c) cpu/memory work - data transfer
(solid line)

(d) register/register work - instruction and data transfer
(alternating dot/dash line)

(e) register/register work - control function
(dotted line)

6.3.2 Instruction Class CP Power

6.3.2.1 Decomposition of Power by Class

We can partition the AN/UYK-20 instruction repertoire into disjoint classes by considering sets of instructions of like format or like function or by another characteristic useful for a specific purpose. The occurrence of only one instruction per class is the degenerate case. Let $\mathcal{S}_i$ denote the $i^{\underline{th}}$ instruction class.

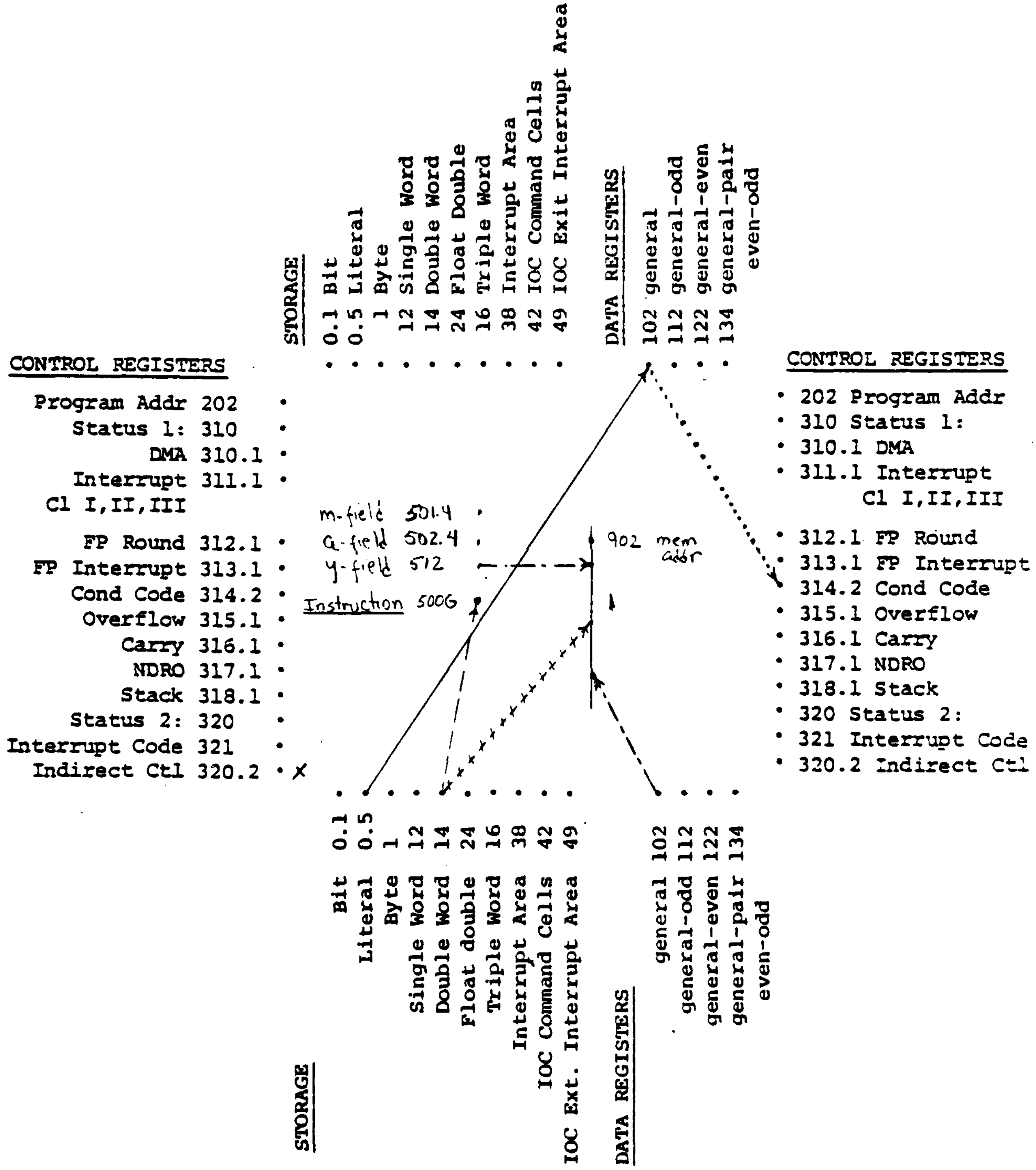
Let L represent some workload in which instructions $s \in \bigcup \mathcal{S}_i$ are executed. Let S_i be the subworkload consisting of all the executions of $s \in \mathcal{S}_i$.

CPU SOFTWARE FORCE VECTORS

AN/UYK-20 Repertoire

Instruction: Byte Load
OP: ØØ
Mnemonic: BL, y, Rm
Format: RX
Tx(μsec): 2.25

Description:
$Byte_m[y] \rightarrow R_a$;
$Y = y + [R_m]/2$
Set CC



CP SOFTWARE FORCES - RX BYTE LOAD

Figure 6.3a

CPU SOFTWARE FORCE VECTORS

AN/UYK-20 Repertoire

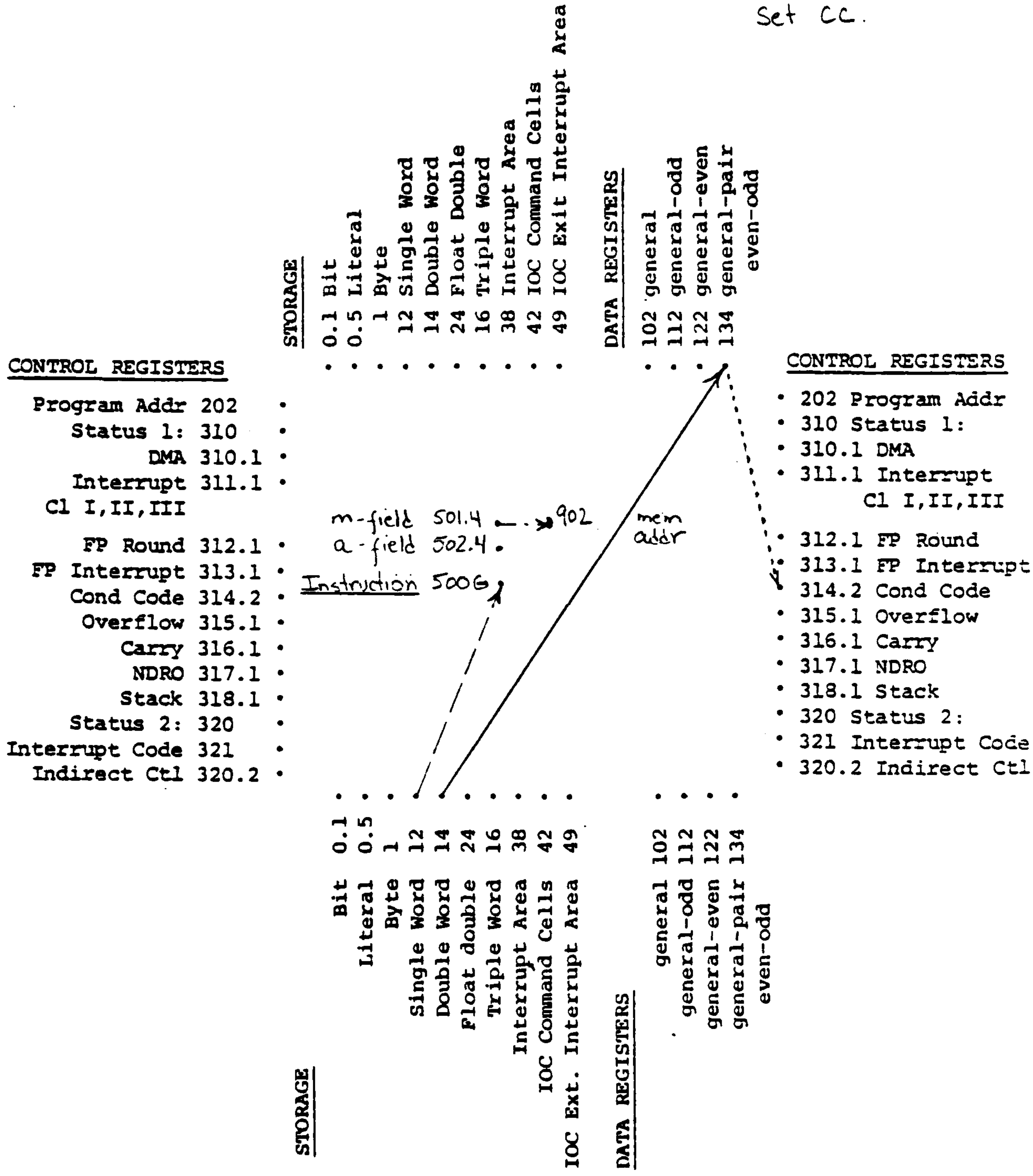
Instruction:	LOAD DOUBLE
OP:	Ø2
Mnemonic:	LDI Ra, Rm
Format:	RI-2
Tx(μsec):	2.25

Description:

$[[R_m]] \rightarrow R_a$,

$[[R_m]+1] \rightarrow R_{a+1}$;

Set CC.



CP SOFTWARE FORCES - RI-2 LOAD DOUBLE

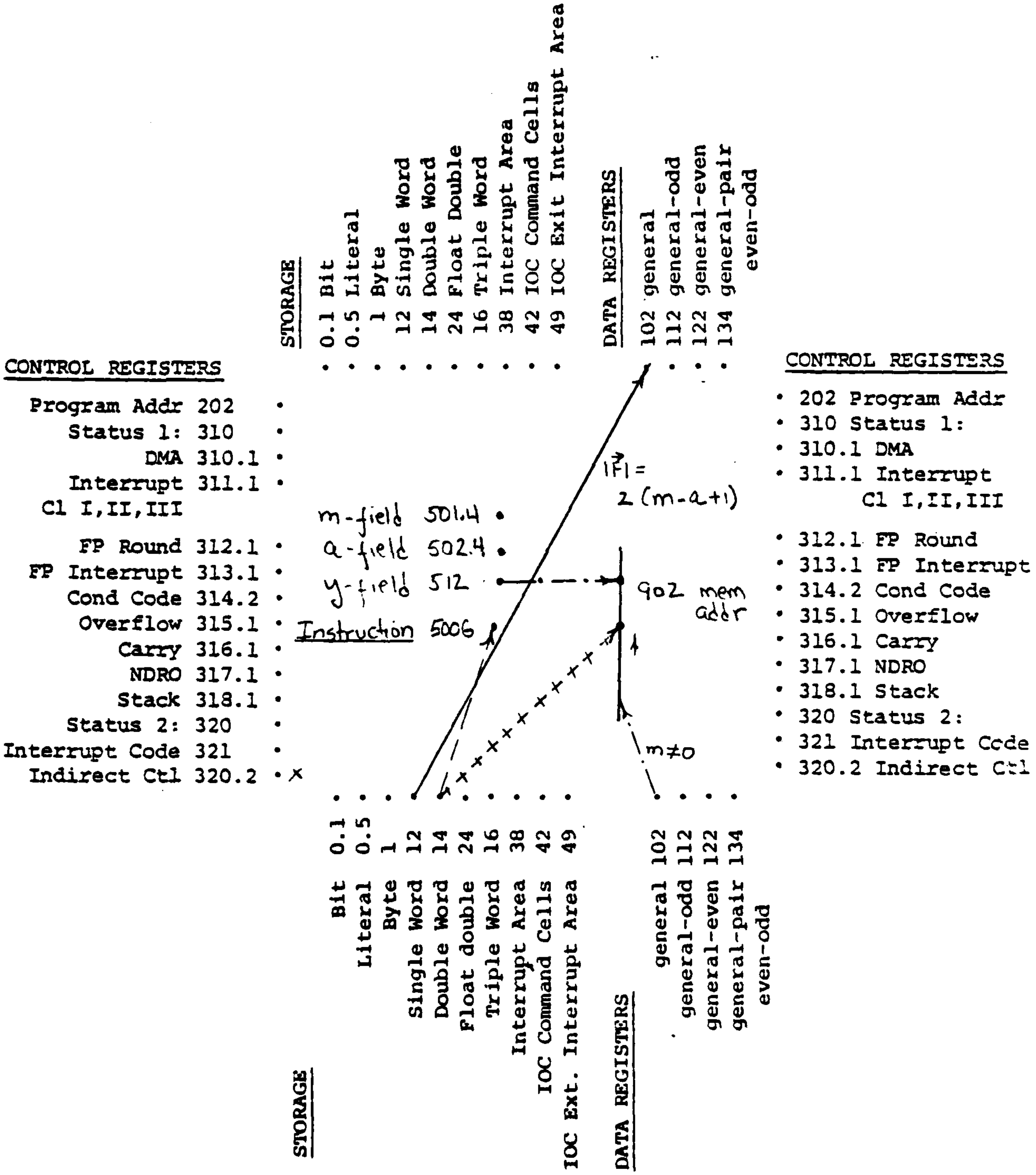
Figure 6.3b

CPU SOFTWARE FORCE VECTORS

AN/UYK-20 Repertoire

Instruction: LOAD MULTIPLE
OP: Ø3
Mnemonic: LM R_a, y, R_m
Format: RX
Tx(μsec): $1.5 + 1.1(m-a+1)$

Description:
$[Y] \rightarrow R_a,\ [Y+1] \rightarrow R_{a+1},\ \cdots [Y+m-a] \rightarrow R_m$



CP SOFTWARE FORCES - RX LOAD MULTIPLE

Figure 6.3c

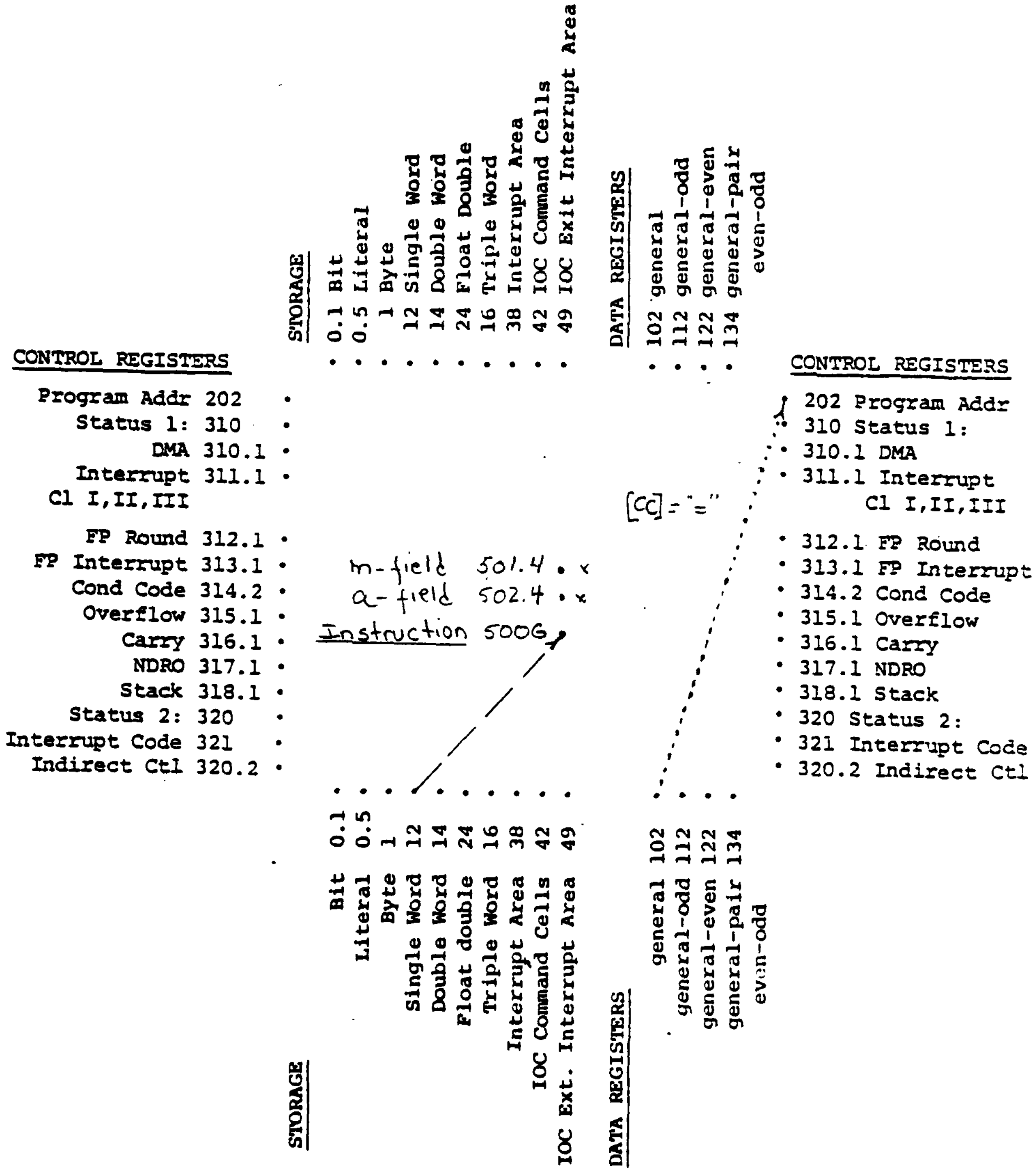
CPU SOFTWARE FORCE VECTORS

AN/UYK-20 Repertoire

Instruction: JUMP EQUAL
OP: 4Ø
Mnemonic: JER R_m
Format: RR
Tx(μsec): 1.1

Description:

$[CC] = "=" \Rightarrow [R_m] \rightarrow P$



CP SOFTWARE FORCES - RR JUMP EQUAL

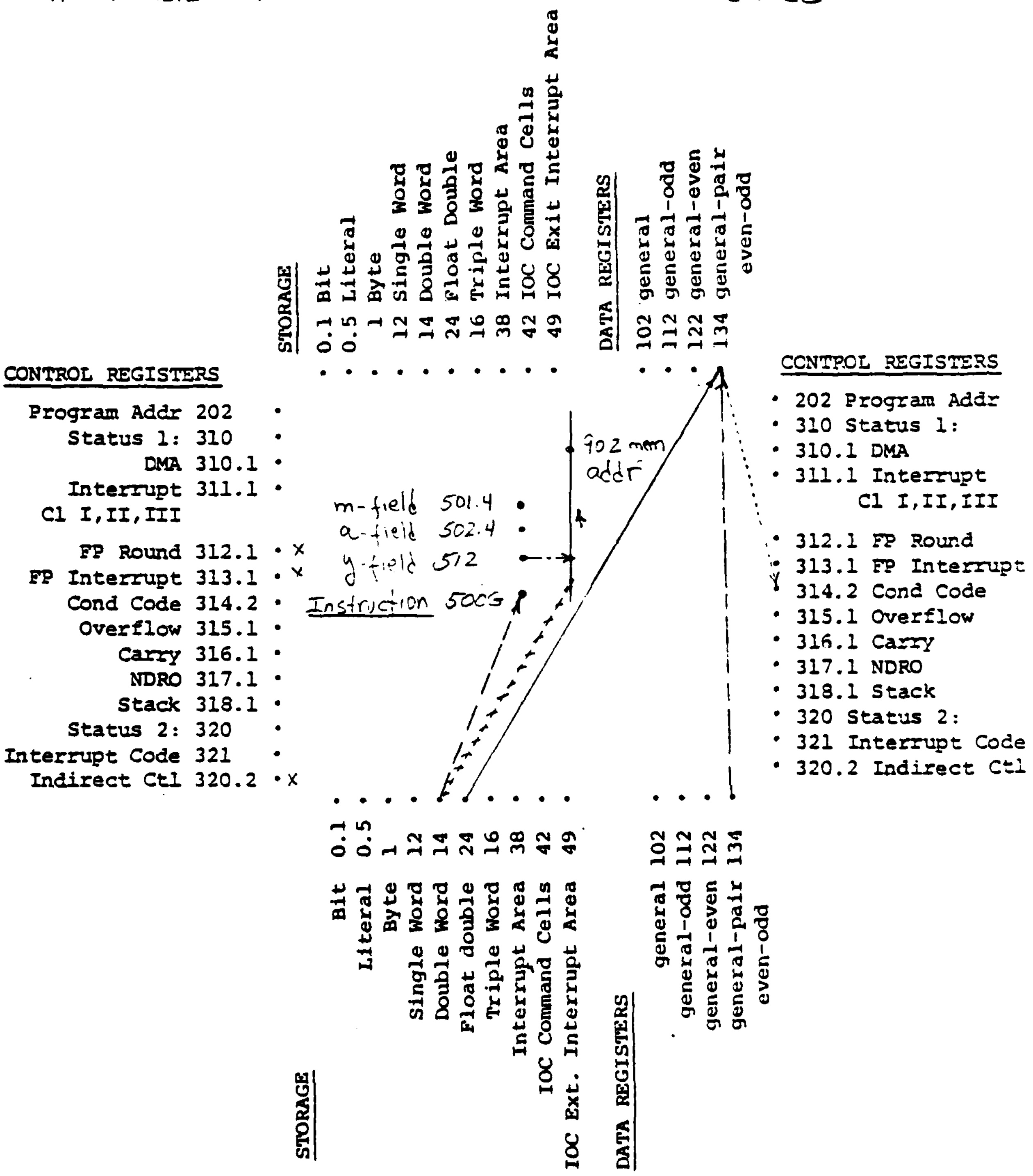
Figure 6.3d

CPU SOFTWARE FORCE VECTORS

AN/UYK-20 Repertoire

Instruction: FLOAT MULTIPLY
OP: 52
Mnemonic: FM a,y,m
Format: RX
Tx(μsec): 15.2 - 18.9

Description:
(Y, Y+1) × (Ra, Ra+1)
Normilize, Round (if spec,
Set CC



CP SOFTWARE FORCES - RX FLOAT MULTIPLY

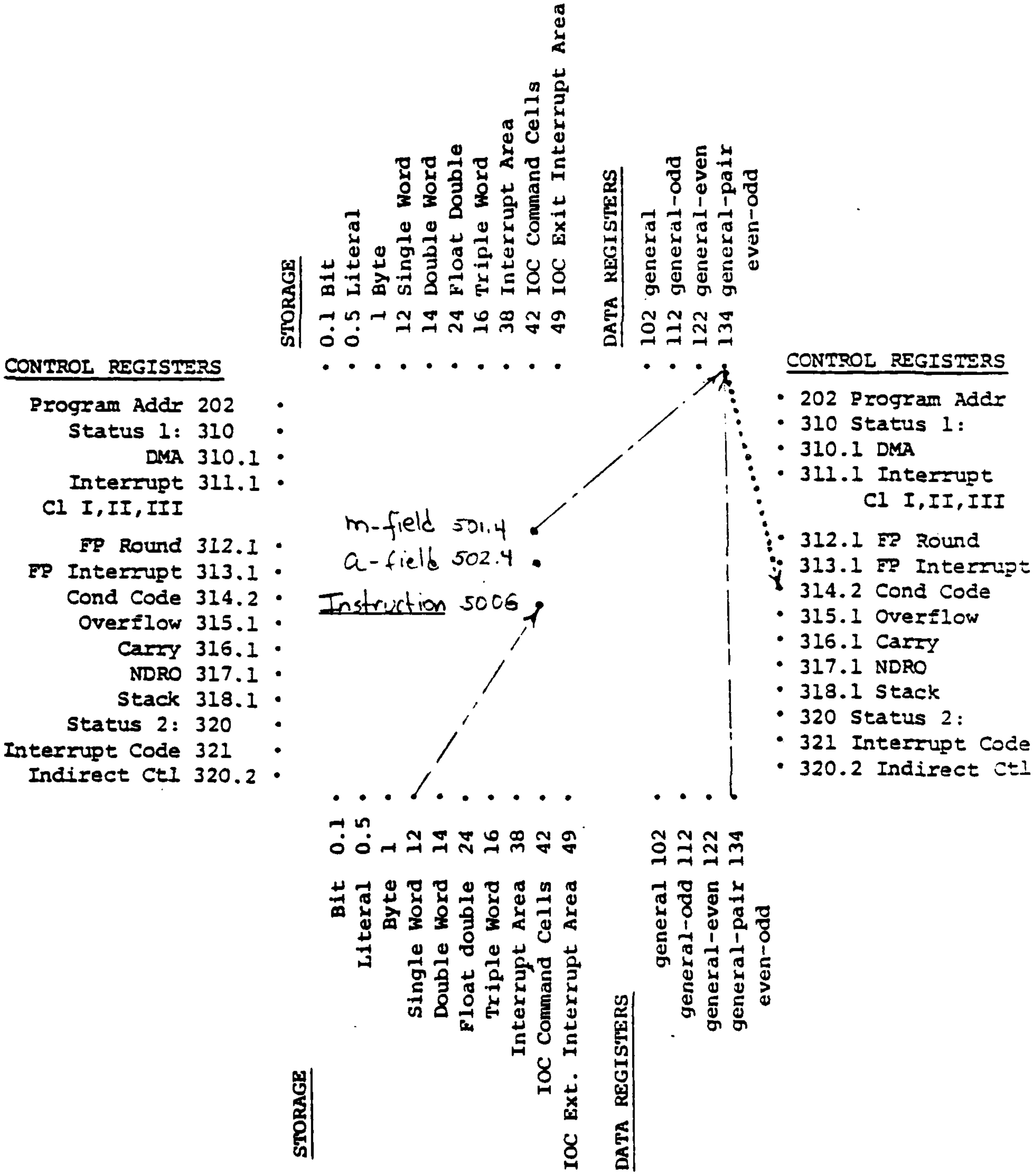
Figure 6.3e

CPU SOFTWARE FORCE VECTORS

AN/UYK-20 Repertoire

Instruction: SUBTRACT DOUBLE
OP: 62
Mnemonic: LSUD Ra, Rm
Format: RL-2
Tx(μsec): 2.35

Description:
$[R_a, R_{a+1}] - m \rightarrow R_a, R_{a+1}$;
Set CC



CP SOFTWARE FORCES - RL-2 SUBTRACT DOUBLE

Figure 6.3f

Then:

$$P(L,\gamma) = \frac{W(S_1,\gamma) + W(S_2,\gamma) + \cdots}{Tx(L,\gamma)}$$

$$= \frac{W(S_1,\gamma)}{Tx(L,\gamma)} + \frac{W(S_2,\gamma)}{Tx(L,\gamma)} + \cdots \qquad (6.1)$$

We now use a representative instruction, s_i, chosen or imagined so that:

$$n_i \cdot W(s_i,\gamma) = W(S_i,\gamma)$$

and $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ (6.2)

$$n_i \cdot Tx(s_i,\gamma) = Tx(S_i,\gamma)$$

where n_i is the number of instruction executions of $s \in \mathscr{S}_i$ in L.

For each term $\frac{W(S_i,\gamma)}{Tx(L,\gamma)}$ in (6.1)

write

$$\frac{W(S_i,\gamma)}{Tx(L,\gamma)} = \frac{Tx(S_i,\gamma)}{Tx(L,\gamma)} \cdot \frac{W(S_i,\gamma)}{Tx(S_i,\gamma)}$$

i.e.,

$$\frac{W(S_i,\gamma)}{Tx(L,\gamma)} = \frac{Tx(S_i,\gamma)}{Tx(L,\gamma)} \cdot P(S_i,\gamma) \qquad (6.3)$$

where $P(S_i,\gamma)$ denotes the absolute power of the subworkload, S_i.

Noting that from the defining relations, (6.2)

$$P(S_i,\gamma) = \frac{n_i W(s_i,\gamma)}{n_i Tx(s_i,\gamma)} = \frac{W(s_i,\gamma)}{Tx(s_i,\gamma)} = P(s_i,\gamma)$$

and since,

$$Tx(L,\gamma) = \sum_i n_i Tx(s_i,\gamma)$$

we have from (3):

$$\frac{W(S_i,\gamma)}{Tx(L,\gamma)} = \frac{Tx(S_i,\gamma)}{Tx(L,\gamma)} \cdot P(S_i,\delta)$$

$$= \frac{n_i Tx(s_i,\gamma)}{\sum\limits_i n_i Tx(s_i,\gamma)} \cdot P(s_i\gamma)$$

and so

$$P(L,\gamma) = \sum \frac{n_i Tx(s_i\gamma)}{\sum\limits_i n_i Tx(s_i,\gamma)} \cdot P(s_i,\gamma)$$

$$= \frac{\sum \left[n_i Tx(s_i\gamma) \cdot P(s_i,\gamma)\right]}{\sum n_i Tx(s_i,\gamma)} \tag{6.5}$$

We have thus derived the power of the cpu in execution on the workload L in terms of class representative instruction counts, times and absolute powers.

6.3.2.2 Choice of Instruction Classes

What bears further investigation are the way of partitioning the AN/UYK-20 instruction repertoire into classes which will be useful in guiding the software design process.

Among the possibilities for defining the classes, $\mathcal{S}_i$, are:

(a) $s \in \mathcal{S}_i$ *iff* $P(s,\gamma) = P_i \pm \varepsilon_i$

i.e., instructions of approximately like powers.

(b) By instruction format RI, RX, RK, RL or more generally:

(c) Supposing that we have chosen an index set of source containers, $\{c_{1j}\}$ and an index set of target containers, $\{c_{2j}\}$ where the c_{ij} are container codes, and that we represent the components of software force in the direction of c_{ij} to c_{2j} by $f_{t,(c_{1j},c_{2j})}$ as previously defined. Then we define

$$s \in \mathcal{S} \text{ iff } f_{t,(c_{1j},c_{2j})} \neq 0 \text{ for some } i,j$$

That is if the instruction s maps a container listed in $\{c_{1j}\}$ to one listed in $\{c_{2j}\}$ it belongs to the class $\mathcal{S}$.

This type of partitioning would prove useful in choosing instructions for specific types of arithmetic or logical functions.

6.3.2.3 The Definition of $Tx(s,\gamma)$

For an individual instruction $s \in \mathcal{S}_i$, the time of instruction $Tx(s,\gamma)$ is the sum of the times:

i) $T_I(s,\gamma)$ - The instruction setup/termination time, here effectively the instruction fetch time. This is a function of instruction length. Fetches may be overlapped with execution. In general, however, the fetch times by format are:

RR RI Type 2} RL	1 memory cycle
RK RX }	2 memory cycles

(where an AN/UYK-20 memory access cycle requires 750 ± 10 nanoseconds.)

ii) $T_D(s,\gamma)$ - The time period commencing with the instruction access in the CP register by the macro-instruction-emulating microprogrammed controller (MPC) to the next instruction fetch. It includes operand fetches, if they are required.

When indirect addressing is in effect, the time for additional accesses should be added to the time $T_D(s,\gamma)$.

The instruction execution times quoted in the AN/UYK-20 Technical Description - SPERRY-UNIVAC PX 10431C are based on actual execution and are composed of $T_D(s,\gamma)$ for the instruction plus $T_I(s,\gamma)$ for the following instruction in the sequence. We will assume that these times represent a fair value of $Tx(s,\gamma)$ for all instructions.

As an example, consider the

(RI) 02 LOAD DOUBLE

instruction.

$W_I(s,\gamma)$ (instruction fetch) $= 2\ W$

$W_D(s,\gamma)$ (data transfer) $= 4\ W$

(no indirect addressing)

$W(s,\gamma)$ (total) $= \overline{6\ W}$

$Tx(s,\gamma) = 2.25$ msec.

So absolute instruction power:

$$P(s,\gamma) = (6/2/25) \times 10^{-6} = 2.67\ KW/S$$

6.4 AN/UYK-20 CP POWER - IOC ACTIVITY INTERACTION

6.4.1 Introduction

We will extend the previously derived instruction class power equation:

$$P(L,\gamma) = \frac{\sum[n_i Tx(s_i,\gamma)\cdot(P(s_i,\gamma)]}{\sum n_i Tx(s_i,\gamma)} \tag{6.5}$$

to include the effects of delays in CP execution due to memory access demands from IOC input/output activity.

It will be convenient to rewrite (6.5) in terms of instruction class work thus:

$$P(L,\gamma) = \frac{\sum[n_i Tx(s_i,\gamma)\cdot\frac{W(s_i,\gamma)}{Tx(s_i,\gamma)}]}{\sum n_i Tx(s_i,\gamma)} \tag{6.6}$$

This is done as the contention for memory access will affect all the terms $Tx(s_i,\gamma)$ by replacing them with the execution time of a higher level processor, Γ, which is execution whenever either the CP or IOC are.

The resultant relative power will be referred to as IOC-Degraded Workload Power and will be denoted $P^*(L,\gamma,\Gamma)$.

We shall see that the degradation will depend on the instruction formats in execution and on the composition of IOC power by memory access bandwidth.

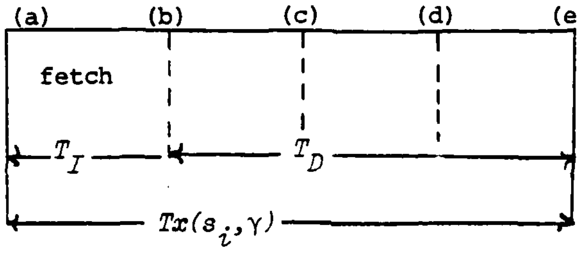
We will first develop the execution time of the processor Γ for the duration of the instruction s_i, $Tx(s_i,\Gamma)$, as a function of the CP instruction execution time $Tx(s_i,\gamma)$and the concurrent IOC input output power $P(\phi)$.

6.4.2 MPC Emulation of CP and IOC activity

AN/UYK-20 execution is driven by a microprogrammed controller (MPC) and master clock running at 155 ± 5 nsec (denoted t_c) per clock cycle. This is the processor Γ. The micro instruction code emulates the CP program macro instructions and services IOC memory access requests. The following model will be used to describe the augmentation of (macro) instruction execution time, $Tx(s_i,\gamma)$, due to memory access requests of the IOC.

i) The microprogram, through the use of the "emulate" instruction, allows an IOC main memory request before each CP instruction fetch. The "emulate" executed before CP macro instruction execution begins will be referred to as an "emulate start."

ii) The sequence of events from start CP instruction fetch to the fetch of the next instruction is charted as follows:



(a) Start fetch: T_I instruction fetch time (cpu/memory work). Depends on instruction format:
RR, RI, RL - 1 memory cycle
RK, RX - 2 memory cycles

(b) Begin macroinstruction: T_D is the data (operand) action time.

(c) Possible operand: memory references — Included in T_D; not in RR instructions.

(d) Resume MPC execution

(e) Start next instruction fetch

The software physics $Tx(s_i,\gamma)$ is the total of instruction fetch time, T_I, and the nominal published execution time, T_D. The effects of indirect addressing or overlap will not be considered here but can be accounted for by adding increments to the operand work performed and the time T_D for processing and additional fetches.

iii) Any CP macroinstruction main memory reference is preceded by an "emulate" macroinstruction to permit IOC access first.

iv) The microcode services all outstanding IOC memory access requests at any IOC-caused suspension of CP emulation.

v) Let n_c be the number of microcode instructions required for a single word access. Then the number of microcode instructions for a byte mode access is also n_c, while the number of microcode instructions for a double word is $2n_c$.

vi) Additional "emulate" instructions are inserted into CP emulation sequences to permit IOC memory access service. Let n_e be the total number of "emulates" (of any type) that occur in a single macroinstruction microcode execution over time $Tx(s_i,\gamma)$.

vii) A return microcode sequence is required whenever the IOC channel suspension of CP emulation occurs at an "emulate" which is not an "emulate start" [see i)]. Let n_r be the number of microinstructions required in the return sequence.

viii) A memory cycle wait of t_w = 800 nsec is required when the break-in is not on an emulate start.

ix) A memory hold time, t_h, is required for each access. The values of t_h are

input: $t_h = 360\ nsec$
output: $t_h = 40\ nsec$

6.4.3 Assumptions for a Worst-Case Degradation

We will develop a worst case degradation of CP instruction execution time under the assumption that the probability of a return sequence being required for each transfer is the same as the probability that the "emulate" which allowed it is not an "emulate start." In actuality, more than one transfer can occur per "emulate" because if more than one buffer is active, then there is a non-zero probability that there are concurrent requests outstanding. We will also not consider the effects of chaining or of instruction fetch overlaps.

We thus have that the probability of a return sequence and of a memory cycle wait are:

$$P_r = P_{\overline{es}} = 1 - P_{es} = 1 - \frac{1}{n_e}$$

where P_{es} is the probability that an "emulate" is an "emulate start."

6.4.4 IOC - Augmented MPC Execution Time - Tx^*

Let the number of IOC memory access demands/second be denoted $D_{\phi 8}$, $D_{\phi 16}$ and $D_{\phi 32}$ depending on whether the request is for a byte (8 bit), single word (16 bit), or double word (32 bit) access, respectively.

Letting Tx stand for $Tx(s_i, \gamma)$ we set:

$$\begin{aligned} Tx^* = Tx(s_i, \Gamma) = {} & Tx \\ & + Tx(D_{\phi 8} + D_{\phi 16} + 2D_{\phi 32})\ n_c t_c \times 10^{-9} \\ & + Tx(D_{\phi 8} + D_{\phi 16} + D_{\phi 32})(1 - \frac{1}{n_e})\ n_r t_c \times 10^{-9} \\ & + Tx(D_{\phi 8} + D_{\phi 16} + D_{\phi 32})(1 - \frac{1}{n_e})\ t_w \times 10^{-9} \\ & + Tx(D_{\phi 8} + D_{\phi 16} + 2D_{\phi 32})\ t_h \times 10^{-9} \end{aligned} \tag{6.7}$$

where: $\overset{*}{Tx}$ represents the augmented time of macroinstruction execution due to the $n_c t_c$ nsec microcode execution times per single word (or byte) accesses, the $n_r t_c$ nsec return sequence time and cycle wait time, t_w, each with probability $(1-\frac{1}{n_e})$ for any access and the memory hold time, t_h.

Note that equation (6.7) is valid only when

$$D_{\phi 8}, D_{\phi 16} \text{ and } D_{\phi 32} \leq \frac{n_e}{Tx} \text{ emulates/sec.}$$

since accesses are allowed only by virture of the recurring "emulate" microinstructions.

Denoting the IOC powers for byte, single word and double word access by $P(\phi_8)$, $P(\phi_{16})$ and $P(\phi_{32})$, respectively, (each in KW/sec) we have:

$$D_{\phi 8} = P(\phi_8) \times 10^3 \text{ , } D_{\phi 16} = \frac{P(\phi_{16}) \times 10^3}{2}$$

$$\text{and } D_{\phi 32} = \frac{P(\phi_{32}) \times 10^3}{4}$$

Substitution in (6.7) gives:

$$\begin{aligned}\overset{*}{Tx} = Tx + Tx\{1 &+ [P(\phi_8) + \frac{P(\phi_{16})}{2} + \frac{P(\phi_{32})}{2}] n_c t_c \times 10^{-6}\\ &+ [P(\phi_8) + \frac{P(\phi_{16})}{2} + \frac{P(\phi_{32})}{4}] [1-\frac{1}{n_e}] n_r t_c \times 10^{-6}\\ &+ [P(\phi_8) + \frac{P(\phi_{16})}{2} + \frac{P(\phi_{32})}{4}] [1-\frac{1}{n_e}] t_w \times 10^{-6}\\ &+ [P(\phi_8) + \frac{P(\phi_{16})}{2} + \frac{P(\phi_{32})}{2}] t_h \times 10^{-6}\}\end{aligned}$$

Collecting terms in $P(\phi_8)$, $P(\phi_{16})$ and $P(\phi_{32})$ we have:

$$\overset{*}{Tx} = Tx\{1 + P(\phi_{32})[\frac{2n_c t_c + (1-\frac{1}{n_e})(n_r t_c + t_w)}{4} + \frac{t_h}{2}] \times 10^{-6}$$

$$+ [2P(\phi_8) + P(\phi_{16})][\frac{n_c t_c + (1-\frac{1}{n_e})(n_r t_c + t_w)}{2} + \frac{t_h}{2}] \times 10^{-6}\}$$

(6.8)

and this is valid only when:

$$P(\phi_8) \leq \frac{n_e}{Tx} \times 10^{-3} \text{ , } P(\phi_{16}) \leq \frac{2n_e}{Tx} \times 10^{-3} \text{ and } P(\phi_{32}) \leq \frac{4n_e}{Tx} \times 10^{-3}$$

or more concisely when:

$$P(\phi_k) \leq \frac{kn_e}{8Tx} \times 10^{-3} \qquad (k = 8,\ 16,\ 32)$$

The term in braces in equation (6.8) is equal to:

$$\overset{*}{Tx}/Tx \equiv \eta(\phi) \equiv 1/\xi(\phi)$$

where:

$\eta(\phi)$ is called the I/O-Degraded CP Execution Time Factor, and $\xi(\phi)$ is called the I/O-Degraded CP Execution Power Factor for the reason that it will appear as a multiplier of the instruction power $P(s_i,\gamma)$ in the expression for CP execution power when the IOC is concurrently active.

6.4.5 IOC-Degraded Instruction Power

We now have a relative I/O-degraded CP execution power:

$$P^*(s_i,\gamma,\Gamma) = \frac{W(s_i,\gamma)}{\overset{*}{Tx}}$$

$$= \frac{W(s_i,\gamma)}{Tx(s_i\gamma)} \cdot \frac{Tx(s_i,\gamma)}{\overset{*}{Tx}} = P(s_i,\gamma) \cdot \xi_i(\phi) \qquad (6.9)$$

where $\xi_i(\phi)$ is the function $\xi(\phi)$ evaluated with the quantity n_e valid for s_i.

We will formulate $\xi_i(\phi)$ for input and output (denoted $\xi_i(\phi_I)$ and $\xi_i(\phi_O)$, respectively) for two restricted instruction mixes and a general mix considered typical by the manufacturer. n_c and n_r are given as 5 microinstructions each for the access and return microcode sequences. t_w, the memory cycle wait time, is 800 nsec and the memory hold time is 360 nsec for input, 40 nsec for output.

i) Restricted Mix 1 - RI Add and Logical.

Assume that the instructions executed are limited to 22 RI Add and 31 RI Logical instructions:

We have from the manufacturer's data:

$Tx(s_i,\gamma) = 1.6\ \mu sec\ at\ t_c = 155\ nsec$

n_e, the number of "emulates" is 2 (during T_D)
+ 1 (during fetch) = 3

Assuming that the double word power $P(\phi_{32})$ dominates, we have $P(\phi) \simeq P(\phi_{32})$. Equation (6.8) then becomes:

$$Tx^* = Tx[1 + (650+\frac{t_h}{2}) \times 10^{-6}]\quad P(\phi)$$

so for input,

$$\xi_i(\phi_I)_{mix\ 1} = \frac{1}{1 + (650+\frac{360}{2}) \times 10^{-6}\ P(\phi)}$$

$$= \frac{1}{1 + 830 \times 10^{-6}\ P(\phi)} \qquad (6.10a)$$

and for output,

$$\xi_i(\phi_O)_{mix\ 1} = \frac{1}{1 + (650+\frac{40}{2}) \times 10^{-6}\ P(\phi)}$$

$$= \frac{1}{1 + 670 \times 10^{-6}\ P(\phi)} \qquad (6.10b)$$

ii) Restricted Mix 2 - RX Add and Logical.

Assume that the instructions executed are limited to 22 RX Adds and 31 RX Logical instructions.

From the manufacturer's data:

$Tx(s_i,\gamma) = 2.3\ \mu sec$

n_e is 3 (during T_D) + 2 (during fetch) = 5

Again assuming that , we have:

$$T\overset{*}{x} = Tx[1 + (702.5+\frac{t_h}{2}) \times 10^{-6}\ P(\phi)]$$

so for input,

$$\xi_i(\phi_I)_{mix\ 2} = \frac{1}{1 + 722.5 \times 10^{-6}\ P(\phi)}$$

$$= \frac{1}{1 + 882.5 \times 10^{-6}\ P(\phi)} \qquad (6.11a)$$

and for output,

$$\xi_i(\phi_O)_{mix\ 2} = \frac{1}{1 + 722.5 \times 10^{-6}\ P(\phi)} \qquad (6.11b)$$

iii) General Mix

The manufacturer has provided the following instruction mix in document PX 11901 and considers it typical:

17% 22 RI Adds	*(2 emulates in T_D; $Tx = 1.6$ μsec)*
17% 22 RX Add	*(3 emulates in T_D; $Tx = 2.3$ μsec)*
17% 31 RI Logical	*(2 emulates in T_D; $Tx = 1.6$ μsec)*
17% 31 RX Logical	*(3 emulates in T_D; $Tx = 2.3$ μsec)*
12% 44 RI Jumps	*(2 emulates in T_D; $Tx = 1.3$ μsec)*
8% Miscellaneous	*(1 emulate in T_D; $Tx = 0.84$ μsec)*
6% 44 RX Jumps	*(3 emulates in T_D; $Tx = 2.4$ μsec)*
4% 26 RX Multiplies	*(3 emulates in T_D; $Tx = 4.5$ μsec)*
1% 26 RI Multiplies	*(2 emulates in T_D; $Tx = 4.3$ μsec)*
1% 27 RK divides	*(3 emulates in T_D; $Tx = 7.6$ μsec)*

We add: 1 emulate in T_I for RI and miscellaneous instructions

or

2 emulates in T_I for RX and RK instructions.

We then have that there are 3.8 emulates in $Tx(s_i,\gamma)$, the execution time for the mix representative (average) instruction.

So for this average s_i we have $Tx(s_i,\gamma) = 2.00 \times 10^{-6}$ *sec* and $\bar{n}_e = 3.8$.

Now when $P(\phi) \simeq P(\phi_{32})$ we have from equation (6.8):

$$\overset{*}{Tx} = Tx[1 + (677.7+\frac{t_h}{2}) \times 10^{-6}\ P(\phi)]$$

From which:

$$\xi_i(\phi_I)_{mix\ G} = \frac{1}{1 + 857.7 \times 10^{-6}\ P(\phi)} \qquad (6.12a)$$

$$\xi_i(\phi_O)_{mix\ G} = \frac{1}{1 + 697.7 \times 10^{-6}\ P(\phi)} \qquad (6.12b)$$

If $P(\phi) \simeq P(\phi_{16})$, we have:

$$Tx^{*} = Tx[1 + (967.9+\frac{t_h}{2}) \times 10^{-6}\ P(\phi)]$$

From which:

$$\xi_i(\phi_I)_{mix\ G} = \frac{1}{1 + 1148 \times 10^{-6}\ P(\phi)} \qquad (6.12c)$$

$$\xi_i(\phi_O)_{mix\ G} = \frac{1}{1 + 987.9 \times 10^{-6}\ P(\phi)} \qquad (6.12d)$$

and finally if $P(\phi_{16}) = P(\phi_{32}) = \frac{P(\phi)}{2}$, we have:

$$Tx^{*} = Tx[1 + (967.9+\frac{t_h}{2}) \times 10^{-6}\ P(\phi_{16})$$

$$+ (677.7+\frac{t_h}{2}) \times 10^{-6}\ P(\phi_{32})]$$

$$= Tx\ 1[+ (484.0+338.9+\frac{t_h}{2}) \times 10^{-6}\ P(\phi)]$$

$$= Tx\ 1[+ (822.9+\frac{t_h}{2}) \times 10^{-6}\ P(\phi)]$$

From which:

$$\xi_i(\phi_I)_{mix\ G} = \frac{1}{1 + 1003 \times 10^{-6}\ P(\phi)} \qquad (6.13a)$$

$$\xi_i(\phi_O)_{mix\ G} = \frac{1}{1 + 842.9 \times 10^{-6}\ P(\phi)} \qquad (6.13b)$$

We tabulate and plot the equations (6.13) in Table 6.3 and Figure 6.4, respectively, giving $\xi_i(\phi_I)$ and $\xi_i(\phi_O)$ the I/O-Degraded Execution Power Factors for the SPERRY-UNIVAC PX 11901 General Instruction Mix with the IOC power composition $P(\phi_{16}) = P(\phi_{32}) = \frac{P(\phi)}{2}$. We additionally show in the table and graph, the curve for the anticipated maximum degradation arising from the case of equation (6.12c) for input power when $P(\phi) = P(\phi_{16})$.

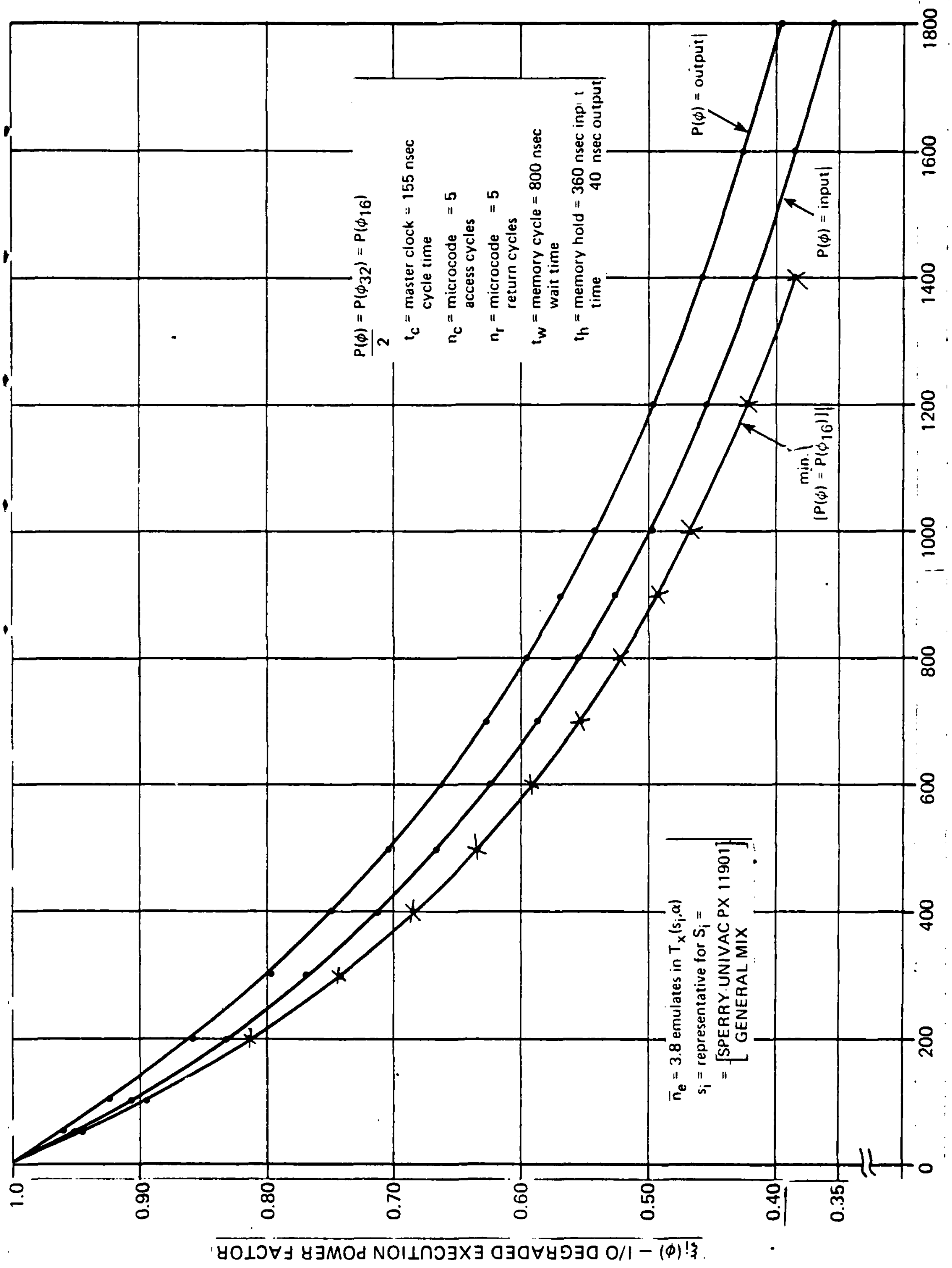
IOC I/O POWER	I/O-Degraded Execution Power Factor - $\xi_i(\phi)$		
KW/SEC	Input	Output	Min.
	$P(\phi_{16}) = P(\phi_{32}) = \frac{P(\phi)}{2}$	$P(\phi_{16}) = P(\phi_{32}) = \frac{P(\phi)}{2}$	$P(\phi) = P(\phi_{16})$ (maximum degradation)
50	0.952	0.960	0.946
100	0.909	0.922	0.897
200	0.832	0.856	0.813
300	0.769	0.798	0.744
400	0.713	0.750	0.685
500	0.666	0.704	0.735
600	0.624	0.664	0.592
700	0.588	0.629	0.554
800	0.555	0.597	0.521
900	0.526	0.569	0.492
1000	0.499	0.543	0.466
1200	0.454	0.497	0.421
1400	0.416	0.459	0.384
1600	0.384	0.426	0.353
1800	0.356	0.397	0.326
2000	0.333	0.372	0.303

RELATIVE CP POWER DEGRADATION - IOC ACTIVITY

Execution Power Factor - $\xi_i(\phi)$

SPERRY-UNIVAC PX 11901 General Mix

Table 6.3


ξi(φ) — I/O DEGRADED EXECUTION POWER FACTOR
1.0
0.90
0.80
0.70
0.60
0.50
0.40
0.35
0
200
400
600
800
1000
1200
1400
1600
1800
P(φ) = P(φ32) = P(φ16)
2
tc = master clock = 155 nsec cycle time
nc = microcode = 5 access cycles
nr = microcode = 5 return cycles
tw = memory cycle = 800 nsec wait time
th = memory hold = 360 nsec input time
40 nsec output
P(φ) = output|
P(φ) = input|
min.|P(φ) = P(φ16)||
n̄e = 3.8 emulates in Tx(si,α)
si = representative for Si =
= [SPERRY-UNIVAC PX 11901 GENERAL MIX]

6.4.6 Relative IOC-Degraded Power for the Full Workload - $P^*(L,\gamma,\Gamma)$

As a consequence of the above and equation (6.5) we have for the full workload:

$$P^*(L,\gamma,\Gamma) = \frac{\sum(n_i Tx^* \cdot P(s_i,\gamma) \cdot \xi_i(\phi))}{\sum n_i Tx^*}$$

$$= \frac{\sum[\frac{n_i Tx(s_i,\gamma)}{\xi_i(\phi)} \cdot P(s_i,\gamma) \cdot \xi_i(\phi)]}{\sum(n_i Tx(s_i,\gamma)/\xi_i(\phi))}$$

$$= \frac{\sum(n_i Tx(s_i\gamma) \cdot P(s_i,\gamma))}{\sum(n_i Tx(s_i,\gamma) \cdot \eta_i(\phi))} \qquad (6.14)$$

valid when

$$P(\phi_k) \leq \frac{n_e}{8Tx(s_i,\gamma)} \times 10^{-3} \qquad (k = 8,\ 16,\ 32)$$

where:

$Tx(s_i,\gamma)$ is the instruction time including fetch expressed in seconds.

n_e is the number of microcode "emulate" instructions in the microcode sequence for a single CP macroinstruction execution.

and

$P(\phi_k)$ is the k-bit partial power expressed in KW/sec $(k = 8,\ 16,\ 32)$.

Note that in equation (6.14), the effect of IOC activity is expressed purely in the denominator as augmentations of the instruction execution times by the multiplicative factors $\eta_i(\phi)$. This corresponds to the notion that the CP work done is the same with or without IOC activity but the effective execution time has increased.

The factor $\eta_i(\phi)$, the reciprocal of $\xi_i(\phi)$, is tabulated and plotted in Table 6.4 and Figure 6.5 for the manufacturer's PX 11901 instruction mix for input and output powers when $P(\phi_{32}) = P(\phi_{16}) = \frac{P(\phi)}{2}$. In addition, values for the theoretical maximum degradation are shown occurring for input where $P(\phi) = P(\phi_{16})$.

IOC I/O POWER	I/O-Degraded CP Execution Time Factor		
	Input	Output	Max.
KW/S	$P(\phi_{16}) = P(\phi_{32}) = \frac{P(\phi)}{2}$	$P(\phi_{16}) = P(\phi_{32}) = \frac{P(\phi)}{2}$	$P(\phi) = P(\phi_{16})$
50	1.050	1.042	1.057
100	1.100	1.085	1.115
200	1.202	1.156	1.230
300	1.300	1.253	1.344
400	1.403	1.333	1.460
500	1.501	1.420	1.575
600	1.603	1.506	1.689
700	1.701	1.590	1.805
800	1.802	1.675	1.919
900	1.901	1.758	2.033
1000	2.004	1.842	2.146
1200	2.203	2.012	2.375
1400	2.404	2.179	2.604
1600	2.604	2.347	2.833
1800	2.809	2.519	3.067
2000	3.003	2.688	3.300

RELATIVE CP POWER DEGRADATION - IOC ACTIVITY

CP Execution Time Factor - $\eta_i(\phi)$

SPERRY-UNIVAC PX 11901 General Mix

Table 6.4

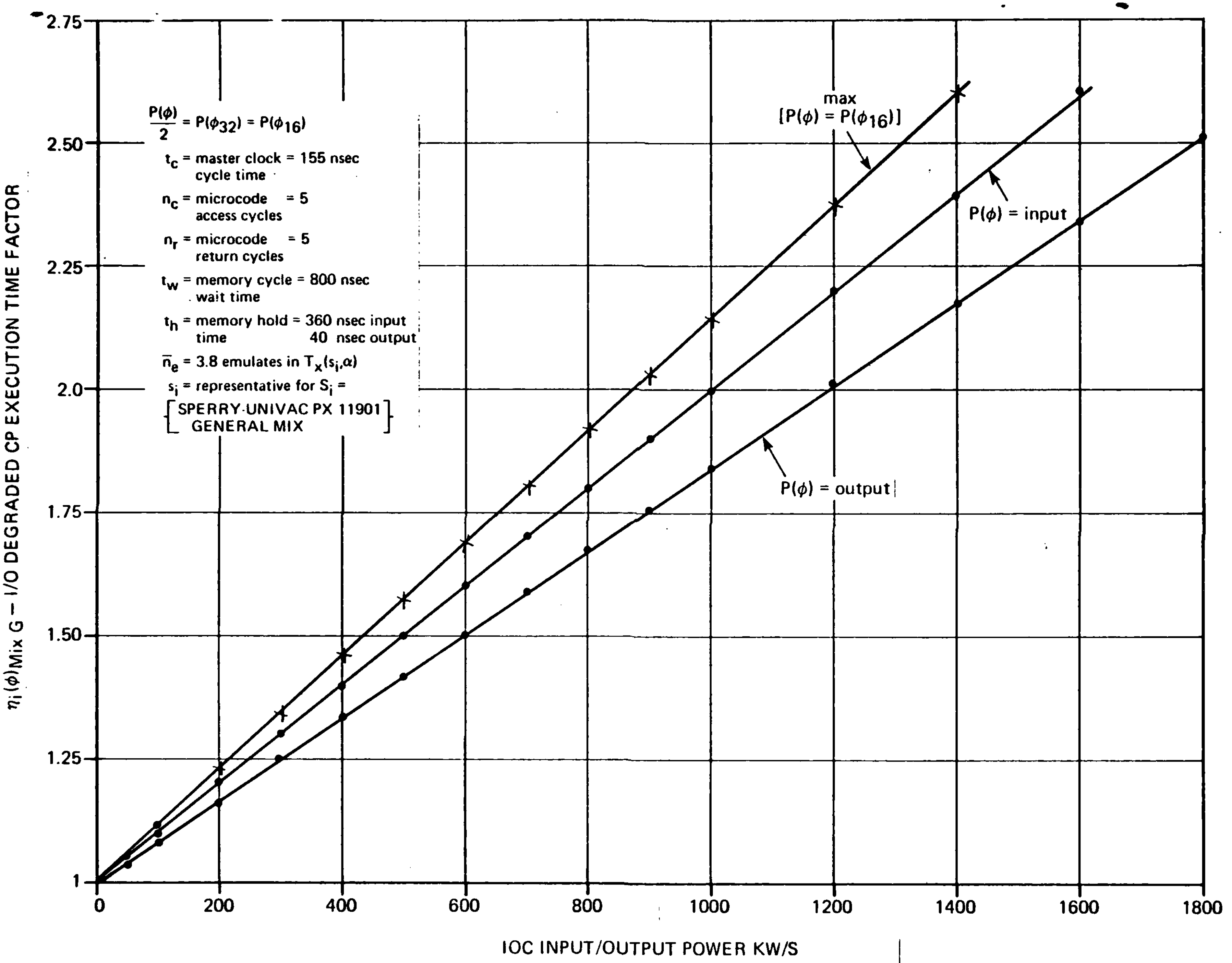


Figure 6.5

CP EXECUTION TIME DEGRADATION – IOC ACTIVITY

6.5 AN/UYK-20 CP POWER - DMA FACILITY EFFECTS

6.5.1 General DMA Characteristics

The AN/UYK-20 design incorporates a direct memory access (DMA) facility which allows an external device to read from and write into main memory via a second memory interface.

The incorporation of the DMA facility increases CP instruction execution times by a small amount, 65 nsec maximum. The actual increases in instruction execution time have been tabulated by the manufacturer for each instruction in the SPERRY-UNIVAC publication PX 11772 and will not be repeated here. We will, however, subscript the symbols γ or Γ in denoted execution times with the letter D to note the fact that the DMA facility is in the system and that the values of T_D *or* Tx for instructions are to include the appropriate increment. Thus $Tx(s_i, \gamma_D)$ is the instruction execution time when the DMA facility is in the system.

Another DMA feature provides for separate access ports on each of the 32K memory banks. This allows access by the DMA-attached device on one bank concurrent with accesses by the CP/IOC on the other. Should requests for memory access on the same bank be simultaneous from the DMA-attached device and the CP/IOC, priority of access is given to the latter units.

6.5.2 Worst-Case CP Degradation Effects - Assumptions

We will first develop a worst case execution time augmentation factor $\mu(\phi_D)$ for DMA facility activity in a memory bank with concurrent CP/IOC activity. Our assumptions are as follows:

i) Delays in instruction execution memory accesses caused by the DMA activity occur only when a DMA memory read or write is already in progress. However, for a worst case analysis, any derived coincidence of a DMA and CP access request will be as if the DMA request preceded that from the CP.

ii) For double word CP accesses, we assume that memory becomes available between each of the two single word accesses.

iii) The average DMA caused delay to the CP memory access will be taken to be one half the memory access cycle time (i.e., 750 ÷ 2 = 375 nsec). Effectively, each DMA access is for a 16 bit word.

iv) We will not consider the effects of indirect addressing here.

6.5.3 Development of the DMA-CP Degradation Factors

We first note that the probability of a DMA read/write access in progress is given by:

$$p_D = \frac{\text{memory access time}}{\text{DMA access period}} = \frac{t_m}{\left(\frac{P(\phi_D) \times 10^3}{2}\right)^{-1}}$$

$$= \frac{t_m \cdot P(\phi_D) \times 10^3}{2}$$

where: t_m is the memory access cycle time in seconds.

$P(\phi_D)$ is the DMA power in KW/sec.

Now considering the CP related memory fetches (instructions + operands) as independent attempts at access, we have that the most probable or expected number of times that the CP would encounter a DMA access in the course of a single instruction execution is:

$$E_{\gamma D} = n_{m\gamma} \cdot p_D$$

where: $n_{m\gamma}$ is the number of accesses required in the full (fetch included) execution of an instruction.

Letting $n_{o\gamma}$ be the number of effective single word instruction operands we have for:

RR, RL, RI - 1 instructions, $n_{m\gamma} = 1$ (fetch only)

RI - 2 instructions, $n_{m\gamma} = 1 + n_{o\gamma}$

RK instructions, $n_{m\gamma} = 2$ (fetches only)

RX instructions, $n_{m\gamma} = 2 + n_{o\gamma}$

We now can write for a degraded CP instruction execution time, letting $\overset{*}{Tx}$ stand for $Tx(s,\Gamma_D)$ and noting that the average delay is $t_m \div 2$:

$$\overset{*}{Tx} = Tx + E_{D\gamma} \cdot t_m \div 2$$

$$= Tx + \frac{n_{m\gamma} \cdot t_m^2 \cdot P(\phi_D) \times 10^3}{4} \tag{6.15}$$

The second term in (6.15) is the additive DMA Power-Degraded Instruction Time Augmentation Factor, $\mu_i(\phi_D)$ where the i subscript is used to indicate that class of instructions for which the value of $n_{m\gamma}$ is valid.

6.5.4 DMA Power-Degraded Instruction and Workload Power

We may now write for the degradation of CP instruction execution power, expressed relative to MPC/DMA execution time:

$$P_D^*(s_i,\gamma_D,\Gamma_D) = \frac{W(s_i,\gamma)}{\overset{*}{Tx}} = \frac{W(s_i,\gamma)}{Tx(s_i,\gamma_D) + \mu_i(\phi_D)}$$

$$= P(s_i,\gamma_D) \cdot [1 - \frac{\mu_i(\phi_D)}{\mu_i(\phi_D) + Tx(s_i,\gamma)}] \tag{6.16}$$

And for the full workload:

$$P_D^*(L,\gamma_D,\Gamma_D) = \frac{\sum(n_i \overset{*}{Tx} \cdot P(s_i,\gamma_D)}{\sum n_i \overset{*}{Tx}}$$

$$= \frac{\sum[n_i(Tx(s_i,\gamma_D) + \mu_i(\phi_D)) \cdot P(s_i,\gamma_D)]}{\sum[n_i(Tx(s_i,\gamma_D) + \mu_i(\phi_D))]} \tag{6.17}$$

Or in terms of previously derived terms:

$$P_D^*(L,\gamma_D,\Gamma_D) =$$

$$= \frac{\sum n_i Tx(s_i,\gamma_D)\cdot P(s_i,\gamma_D) + \sum n_i P(s_i,\gamma_D)\cdot \mu_i(\phi_D)}{\sum n_i Tx(s_i,\gamma_D) + \sum n_i \cdot \mu_i(\phi_D)} \qquad (6.18)$$

We will compute the additive time factor $\mu_i(\phi_D)$ for the previously described (Section 6.4.5) SPERRY-UNIVAC PX 11901 instruction mixes. We will use the nominal $t_m = 750 \times 10^{-6}$ *sec* for the memory access cycle time.

i) Restricted mix 1 - RI add and logical.

Assuming that the instructions executed are limited to 22 RI add and 31 RI logical instructions, we have:

For both instructions $n_{m\gamma} = 1$ *(fetc h) + 1 (operand) = 2 memory accesses/execution.*

From which:

$$\mu_i(\phi_D) = \frac{2\cdot(750\times 10^{-9})^2\cdot P(\phi_D) \times 10^3}{4}$$

$$= 0.2813\ P(\phi_D) \times 10^{-9} \text{ secs/execution}$$

ii) Restricted mix 2 - RX add and logical.

Assuming that the instructions executed are limited to 22 RX add and 31 RX logical instructions, we have:

For both instructions $n_{m\gamma} = 2$ *(fetch) + 1 (operand) = 3* per *memory access/execution.*

From which:

$$\mu_i(\phi_D) = \frac{3\cdot(750\times 10^{-9})^2\cdot P(\phi_D) \times 10^3}{4}$$

$$= 0.4219\ P(\phi_D) \times 10^{-9} \text{ secs/execution}$$

iii) PX 11901 General mix.

We have the following numbers of instruction main memory accesses (single word equivalent) per execution:

17% 22 RI Adds	*(2 accesses/execution)*
17% 22 RX Adds	*(3 accesses/execution)*
17% 31 RI Logical	*(2 accesses/execution)*
17% 31 RX Logical	*(3 accesses/execution)*
12% 44 RI Jumps	*(2 accesses/execution)*
8% Miscellaneous	*(est. 2 accesses/execution)*
6% 44 RX Jumps	*(3 accesses/execution)*
4% 26 RX Multiplies	*(3 accesses/execution)*
1% 26 RI Multiplies	*(2 accesses/execution)*
1% 27 RK Divides	*(2 accesses/execution)*

From the above values we obtain the *weighted average number* of accesses $\bar{n}_{m\gamma} = 2.44$

From which:

$$\mu_i(\phi_D) = \frac{2.44\cdot(750\times10^{-9})^2\cdot P(\phi_D) \times 10^3}{4}$$

$$= 0.3431\ P(\phi_D) \times 10^{-9} \text{ secs/execution}$$

We now show $\mu_i(\phi_D)$ tabulated in Table 6.5 and plotted in Figure 6.6 for $n_{m\gamma} = 1, 2, 2.44, 3, 4, 5$. These values, it will be recalled, are increments to be added to the times $Tx(s_i,\gamma_D)$, the DMA-installed instruction or class representative execution times, for the computation of augmented DMA Power-degraded instruction execution times and degraded relative powers.

DMA POWER $P(\phi_D)$ KW/S	COMMON BANK CP INSTRUCTION TIME AUGMENTATION FACTOR - $\mu_i(\phi_D)$ $(\times 10^{-9}$ *seconds*$)$ Instruction Main Memory Accesses:					
	1	2	2.44*	3	4	5
50	7.03	14.06	17.16	21.09	28.13	35.16
100	14.06	28.13	34.31	42.19	56.25	70.31
200	28.13	56.25	68.63	84.38	112.5	140.6
300	56.25	84.38	102.9	126.6	168.8	210.9
400	84.38	112.5	137.3	168.8	225.0	281.3
500	112.5	140.6	171.6	210.9	281.3	351.6
600	140.6	168.8	205.9	253.1	337.5	421.9
700	168.8	196.9	240.2	295.3	393.8	492.2
800	196.9	225.0	274.5	337.5	450.0	562.5
900	225.0	253.1	308.8	379.7	506.3	632.8
1000	253.1	281.3	343.1	421.9	562.5	703.1
1200	281.3	337.5	411.8	506.3	675.0	843.8
1400	337.5	393.8	480.4	590.6	787.5	984.4
1600	393.8	450.0	549.0	675.0	900.0	1125.
1800	450.0	506.3	617.6	759.4	1012.	1266.
2000	506.3	562.5	686.3	843.8	1125.	1406.

* Average for SPERRY-UNIVAC PX 11901 General Mix

AN/UYK-20 RELATIVE CP POWER DEGRADATION - DMA ACTIVITY

Table 6.5

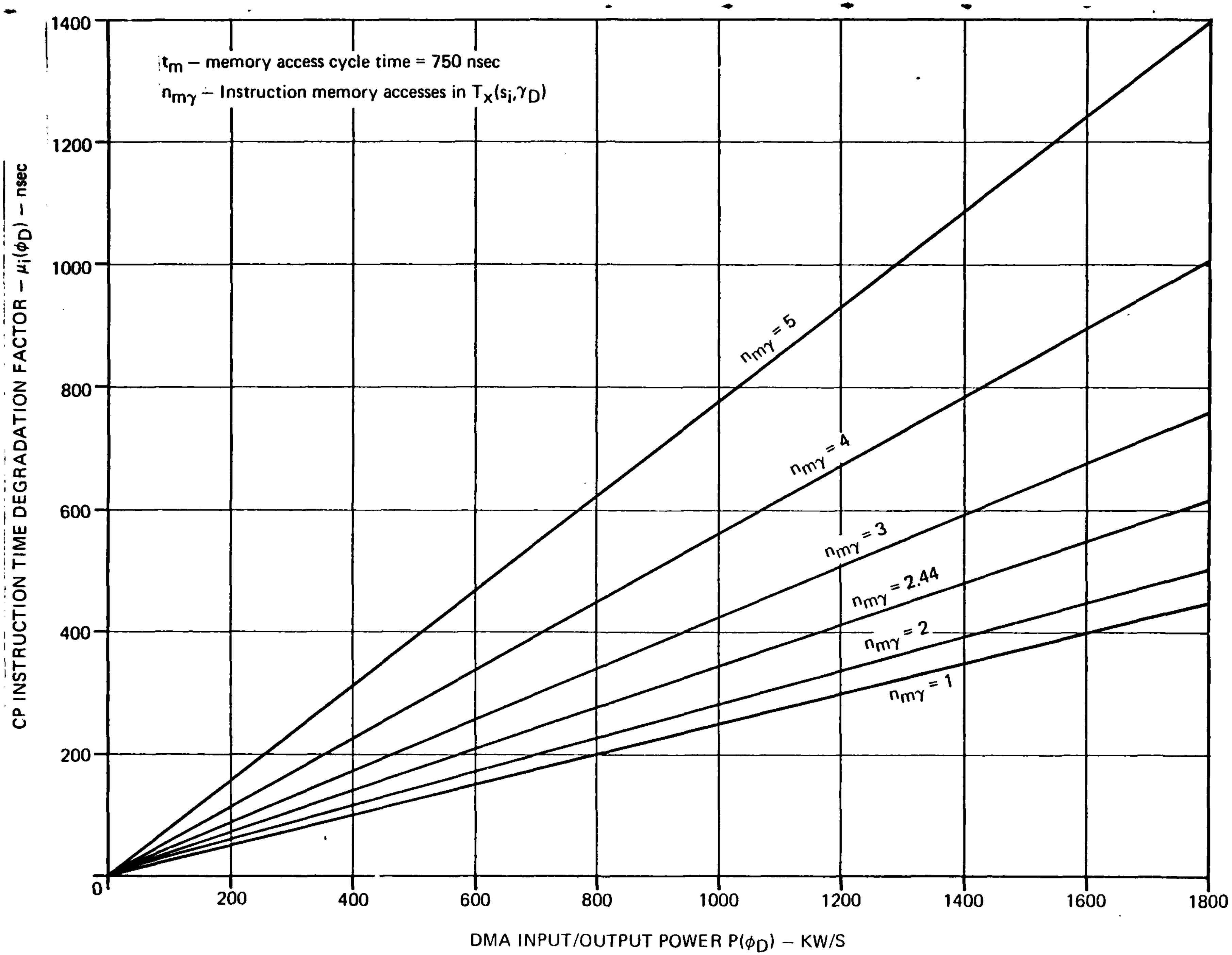


Figure 6.6

AN/UYK-20 CP POWER DEGRADATION – DMA ACTIVITY

As an illustration of the DMA activity CP degradation effect in terms of instruction power, let us consider the impact on the RI and RX Add instructions of our previously employed restricted mixes.

i) 22 RD Add

We have, as before, $n_{m\gamma} = 2$.

From the instruction specifications in SPERRY-UNIVAC PX 11772,

$$Tx(s_i,\gamma_D) = 1.64 \times 10^{-6}$$

So we obtain:

$$P(s_i,\gamma_D) = \frac{W(s_i,\gamma_D)}{Tx(s_i,\gamma_D)}$$

$$= \frac{W_I(s_i,\gamma_D) + W_D(s_i,\gamma_D)}{Tx(s_i,\gamma_D)}$$

$$= \frac{2 + 2}{1.64 \times 10^{-6}} = 2439\ KW/S$$

Now, when there is DMA activity:

$$P_D^*(s_i,\gamma_D,\Gamma_D) = \frac{W(s_i,\gamma_D)}{Tx(s_i,\gamma_D) + \mu_i(\phi_D)}$$

$$= \frac{4}{1.64 \times 10^{-6} + \mu_i(\phi_D)}\ W/S$$

ii) 22 RX Add

For this instruction,

$n_{m\gamma} = 3$

$Tx(s_i, \gamma_D) = 2.40 \times 10^{-6}$ *sec.*

$W(s_i, \gamma_D) = 4 + 2 = 6$ *works*

From which,

$$P(s_i, \gamma_D) = \frac{6}{2.40 \times 10^{-6}} = 2500 \text{ KW/S}$$

and

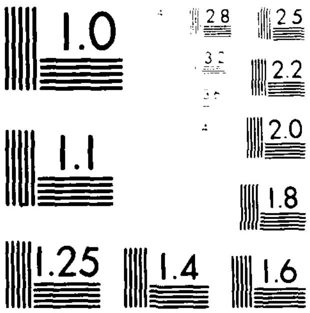
$$P_D^*(s_i, \gamma_D, \Gamma_D) = \frac{6}{2.40 \times 10^{-6} + \mu_i(\phi_D)} \text{ W/S}$$

For both of these instructions we tabulate the relative DMA-Degraded Execution Power $P_D^*(s_i, \gamma_D, \Gamma_D)$, shortened to P_D^* for convenience, in the Table 6.6. We also define a multiplicative DMA-Degraded Instruction Execution Power Factor, $\xi_i(\phi_D) \equiv P_D^*/P$, where P is the instruction power when $P(\phi_D) = 0$. This factor, analogous to the one developed for IOC activity in Section 6.4.4, is also shown in the table and is plotted in Figure 6.7 for the RI ADD instruction.

AD-A113 103 INSTITUTE FOR SOFTWARE ENGINEERING PALO ALTO CA F/G 9/2
AN/UYK-20 CONFIGURATION CAPACITY, NELC-UDI-A-106.(U)
JAN 78 L M TRAISTER N66001-77-C-0252
UNCLASSIFIED NL

2 of 2
AD A
113103

END
DATE
FILMED
04-82
DTIC



MICROCOPY RESOLUTION TEST CHART

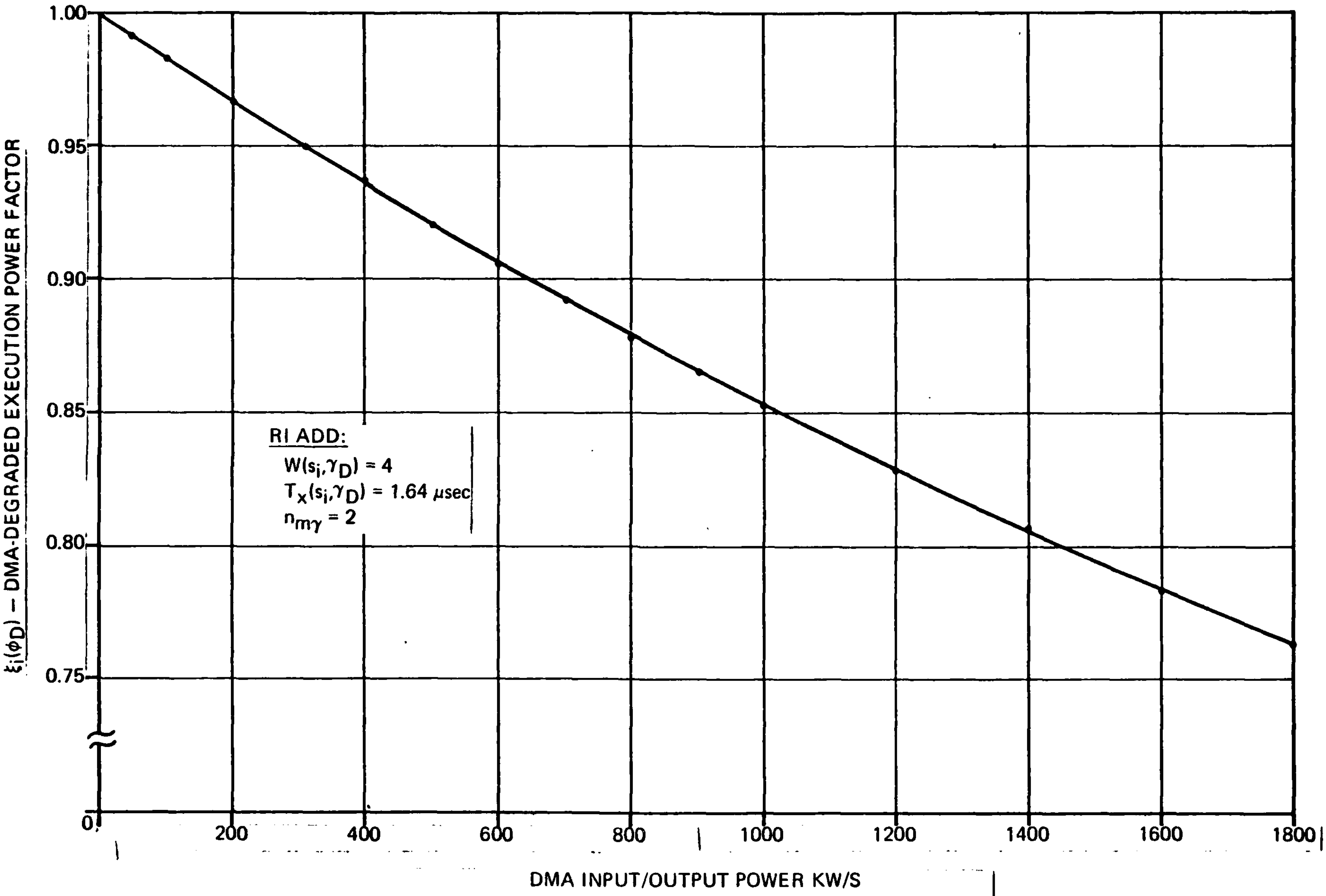
NATIONAL BUREAU OF STANDARDS-1963-A

DMA POWER	COMMON BANK DMA-CP EXECUTION			
$P(\phi_{\sim})$	22 RI ADD	$P = 2439$ KW/S	22 RX ADD	$P = 2500$ KW/S
KW/S	P_D^*	$\xi_i(\phi_D)$	P_D^*	$\xi_i(\phi_D)$
	(KW/S)		(KW/S)	
50	2418.	0.9914	2478.	0.9913
100	2398.	0.9832	2457.	0.9827
200	2358.	0.9668	2415.	0.9960
300	2320.	0.9512	2375.	0.9499
400	2282.	0.9356	2336.	0.9343
500	2246.	0.9209	2298.	0.9192
600	2211.	0.9065	2262.	0.9046
700	2178.	0.8930	2226.	0.8904
800	2145.	0.8795	2192.	0.8767
900	2113.	0.8663	2159.	0.8634
1000	2082.	0.8536	2126.	0.8505
1200	2023.	0.8294	2064.	0.8258
1400	1967.	0.8065	2006.	0.8025
1600	1914.	0.7847	1951.	0.7805
1800	1864.	0.7642	1899.	0.7596
2000	1816.	0.7446	1850.	0.7399

CP POWER DEGRADATION-DMA ACTIVITY

22 RI, RX ADD INSTRUCTIONS

Table 6.6



CP POWER DEGRADATION – DMA COMMON BANK ACTIVITY
22 RI ADD INSTRUCTION

Figure 6.7

Fig 6.7

6.6 DISCUSSION - CP / I/O INTERACTIONS

6.6.1 Introduction

From models of interaction betewen the IOC and the CP and the DMA facility and the CP, we have developed execution time augmentation factors $\eta(\phi)$ and $\mu(\phi_D)$ which lead to the multiplicative power factors $\xi(\phi)$ and $\xi(\phi_D)$. All of these are expressed as functions of the instruction execution time $Tx(s_i,\gamma)$ and the Input or Output powers $P(\phi_I)$ or $P(\phi_O)$ for the IOC and $P(\phi_D)$ for the DMA. We will present a brief amplification of the meaning of these factors and a discussion of the significance of the power factor values derived.

6.6.2 The Time Augmentation Factors

The factors $\eta(\phi)$ and $\mu(\phi_O)$ augment the execution time of a processor, Γ, which emulates CP and IOC memory access activity. This processor includes those facilities, which normally emulating the CP, must suspend that function and service I/O memory access requests.

We have defined the execution time of the processor Γ to be identical with that of the CP when there is no IOC or DMA activity, i.e.,

$$\begin{aligned} Tx(L,\Gamma) &= Tx(L,\gamma) \\ Tx(L,\Gamma_D) &= Tx(L,\gamma_D) \\ &\text{when} \quad P(\phi) = P(\phi_D) = 0 \end{aligned} \tag{6.19}$$

The time augmentation factors $\eta(\phi)$ and $\mu(\phi_D)$ increase $Tx(L,\Gamma)$ or $Tx(L,\Gamma_D)$ over their γ-processor (CP) based execution times; η operates by multiplication and μ by addition because of the differences in the models from which each factor is derived.

When there is input/output activity from either the IOC or DMA we do not increase $Tx(L,\gamma)$ because the γ-processor is defined to be stopped when:

(a) The MPC is servicing I/O requests for memory access.

(b) CP memory access is blocked because an IOC or DMA access is in progress.

Thus the absolute power, $P(L,\gamma)$ of the CP has not changed; it is the CP power relative to the Γ-processor, $P^*(L,\gamma,\Gamma)$, which decreases with I/O activity. This distinction is emphasized because the instruction or workload power equations (6.9), (6.14), (6.16), (6.17) and (6.18) express time in terms of $Tx(s_i,\gamma)$, instruction execution time for the CP.

6.6.3 The Power Degradation Factors

The factors $\xi(\phi)$ and $\xi(\phi_D)$ each are the ratio of a relative power of an instruction when there is no IOC or DMA activity to that when there is IOC or DMA power in use. Because of the relationships (6.19) we have:

$$\begin{aligned} P(s_i,\gamma,\Gamma) &= P(s_i,\gamma) \\ P(s_i,\gamma_D,\Gamma_D) &= P(s_i,\gamma_D) \end{aligned} \qquad (6.20)$$

$$\text{when} \quad P(\phi_D) = P(\phi) = 0$$

Because of the relationships (6.20) we were able to develop the relative power ratios ξ by considering only the absolute CP instruction execution powers and are able to express the full workload relative powers degraded by I/O activity (equations (6.14), (6.18) in terms of the absolute CP instruction execution powers.

6.6.4 Significance of the Power Factor Values

Inasmuch as the power factors ξ_i show degradation of relative instruction and workload power for the CP when there is I/O activity, they may be thought of as reductions to the instruction throughput caused by the interactions described by the models.

The IOC activity, in particular, was theoretically shown to reduce the instruction relative power $P(s_i,\gamma,\Gamma)$ to as little as 30% of its non-I/O active value for maximum IOC power. By contrast, access through the DMA facility on a common memory bank with the CP degrades $P(s_i,\gamma_D,\Gamma_D)$ to no less than 75% of its non-DMA active level. These differences can be explained by the demands that the IOC makes on the microprogrammed controller (MPC) for servicing its memory access requests. The DMA facility, on the other hand, requires that devices using it must provide their own memory interface logic through additional low priority ports to the 32K memory banks. Since these ports are of lower priority than those for the CP/IOC, the likelihood of overruns on DMA connected devices is substantial. In fact, it is doubtful that one could achieve in practice DMA input/output power levels comparable to those for the IOC channels without experiencing frequent overruns.

System and program designers must be aware of the consequences of the CP-IOC, CP-DMA instructions in regard to the effects they can have on workload throughput and system response times. These considerations are a substantial portion of the factors that would determine the performance characteristics of a workload distributed over AN/UYK-20 configurations. In turn we could expect that performance requirements, the knowledge of acceptable trade-offs and the availability of processor power characteristics would provide the basis for satisfactorily performing AN/UYK-20 configurations.

GLOSSARY

GLOSSARY

ASYMPTOTIC POWER

See the discussion under the entry $\vec{P}(A)$

CAPACITY

Two forms of computing system capacity are identified in software physics: 1) processor capacity, expressed in units of software power (works/second), and 2) storage capacity, expressed in units of byte-seconds or their equivalent on non-byte computer systems. In either case, the quantity determined to be the capacity of the system must be calculated from theoretical considerations, and cannot be obtained directly by measurement. Measured values represent the quantity used, not the available quantity of power or byte-seconds.

Both processor capacity and storage capacity can be determined as appropriate for individual devices, subconfigurations, or the full system configuration. In general, processor capacity is primarily a function of equipment speeds and configuration connections, and secondarily a function of workload characteristics. Storage capacity is simply the total storage available by equipment class or subconfiguration over time.

The amount of power actually used is the quantity normally called *performance*. Thus, processor capacity and performance are directly relatable quantities: one is the power available, the other is the power used. The ratio of performance to capacity is called the efficiency of the workload.

See the *power* entry for a more detailed discussion of capacity.

CONFIGURATIONS AND SUBCONFIGURATIONS

A configuration is an arbitrary collection of processors and storage devices, normally connected so that processors cause data to flow to and from storage devices. In certain applications of software physics, however, one is not limited to fully connected configurations.

A subconfiguration is a configuration within a configuration. Often, the prefix "sub" is not used when dealing with parts of a full configuration. For example, "channel configuration" is the collection of a channel, control units, and the drives (printers, terminals, etc.) which can be addressed through the channel. This configuration is part of the I/O configuration, which is the set of all such channel subconfigurations. The "full configuration" is a special term which includes all processors (cpu and I/O) and all storage devices under consideration.

Greek letters are used in software physics to represent configurations and subconfigurations. See the *software physics notation* entry for the symbols used for the standard configurations and subconfigurations.

CONTAINER (STORAGE)

A container is a portion of a storage medium which can be separately addressed. Its size is measured in the number of bytes which are contained in the container, for byte-oriented machines. An 8-bit container has been arbitrarily designated as the standard size container in software physics. As a result, non-byte oriented machine container sizes are determined by dividing the number of bits by eight. Containers such as cards and print positions on paper are counted as having a size equal to the number of bytes (or bits ÷ 8) required to either read or write a character.

DEVICES

Devices are considered as either processors or storage devices. Generally speaking, a device is the lowest level component of a configuration or subconfiguration. For example, a tape drive is considered as a device, as is a tape drive control unit. Together, these devices would make up a tape control unit subconfiguration. Extending this, a channel is considered as a device, but the collection of channel, control unit, and tape drives would be a channel subconfiguration.

The lowest level of a configuration or subconfiguration is still a "configuration." In software physics, configurations and subconfigurations are generally denoted by a Greek symbol. In particular, because devices are the lowest level of a configuration they are normally symbolized by Greek letters; e.g., δ is a drive, γ is a cpu. However, to avoid confusion between channel devices and configurations, a lower case *a* is used to denote the device, Greek α is used to denote the channel configuration with attached control unit subconfigurations. Similarly, *b* is used to denote the control unit as a device, β is used to symbolize the configuration with attached drives.

FORCE, SOFTWARE

In general, a force is the agent, "mechanism," or method by which energy is converted to work. So closely are the concepts of force and energy linked that nearly two centuries elapsed after Newton before a distinction was commonly made between them in the classical physics. In software physics, the means by which software energy results in software force is through the agency of an instruction, either cpu or I/O. As a result, software physics considers an instruction as formally representing a force. A unit of software is a collection of instructions and associated operands, where operands can be considered as being the software physics analogies of inertial mass. Together, the result is that a software unit is considered equivalent to the classical physics system of forces acting on point masses.

Software force is measured in units of *work/byte*. The direction of action of a software force is from the container accessed to the container receiving the symbols transferred. Software force is a vector quantity when a given instruction transfers symbols from more than one set of source-target containers.

MBTR (MAXIMUM BYTE TRANSFER RATE)

This is the rate at which I/O data is read or written, excluding all time required to position or otherwise locate data. It is normally given by equipment manufacturers as either the rated speed (e.g., 2000 lines per minute) or data transfer rate (e.g., 806,000 bytes per second for a 3330 disk drive). In practice, this data transfer rate is never achievable except for burst rate conditions due to a variety of set-up and positioning time requirements. When thought of in terms of software work rather than bytes, this value is also called the asymptotic power of the device.

$\vec{P}(A)$ (ASYMPTOTIC POWER)

The asymptotic theoretical power or capacity of a configuration is symbolized as the vector quantity $\vec{P}(A)$. Its elements are the asymptotic powers of the ideal configuration, by equipment class, denoted P_A. Asymptotic power is equivalent to the maximum byte transfer rate, converted to units of software work and power, available from the equipment being considered. For subconfigurations *including disks and/or tapes, the asymptotic power is* calculated using the first level of "bottlenecking."

Asymptotic power calculations for disk, tape, and other variable block length devices assume an infinite block length. For fixed block length devices, such as printers, the maximum byte transfer rate or its equivalent as specified by the manufacturer and converted to units of power is used; e.g., 2000 lines per minute. For central processors, the asymptotic power is calculated by dividing the bytes accessed from buffer or main storage by the smallest corresponding cycle time.

PERFORMANCE

In common usage this term is not associated with any specific quantitative value. In software physics, the word is fully equivalent to the word software power, and is normally associated with the power usage of the workload, $\vec{P}(L,\psi)$. This is called throughput power. If the batch workload alone is considered, then throughput power is completely equivalent to the common measure "throughput"; i.e., "jobs" per hour. More generally, however, performance may be defined for any level of configuration and/or subworkload by taking the appropriate power usage measure.

The quantitative definition of performance as the level of software power usage is fully in accord with the intuitive meaning of the word, but its use in this sense requires one clarification. When a portion of the workload is removed, such as may be done by changes to the operating system or using TSA runs, capacity is recovered. But capacity is the power available for use by the workload. As such, the power usage level (performance) may decrease quantitatively. Generally, a portion of the recovered power will go into the workload, and a decrease in elapsed time will occur. However, the reduction in time may not be proportional to the reduction in software work. Thus, recovering capacity by reducing the quantity of software work to be performed may result in a decrease in the level of power usage even though a reduction in elapsed time is also observed.

Performance is directly related to workload efficiency, as the latter is equal to $\vec{P}(L,\psi) \div \vec{P}(C)$. Thus, for a given value of $\vec{P}(C)$, one may quote performance either in units of power or in percent efficiency.

POWER, SOFTWARE

Software power is the link between the quantity of software work to be performed and the time required to accomplish it. The term may be used in either of two senses: 1) the power used, formally defined as the work performed divided by the time to accomplish it, or 2) the power available from a device, configuration, or equipment class, calculated from theoretical considerations. When used in the first sense, power is equivalent among other things to the concept of performance. In the second sense, power is equivalent to the concept of capacity.

Power usage is defined as the ratio of work performed to the time required to perform it. Time, however, can be measured in a variety of ways. If the execution time of a device, subconfiguration, or equipment class is used, then the power value calculated is the work performed by these divided by the execution time of the equipment. The execution time is often less than the externally observed elapsed time of the full configuration processing the full quantity of work. This results in two possible ways of calculating power usage level:

1) The power used by a device, subconfiguration, or equipment class relative to the full configuration elapsed time. This is called the relative power.

   An example would be the relative cpu power, $P(L,\gamma,\psi) = W(L,\gamma) \div Tx(L,\psi)$. It represents the work performed by the cpu during the entire period of time required to process the workload L, which would normally include some time when the cpu was not executing any instructions.

2) The power used by a device, a subconfiguration, or equipment class when and only when the corresponding processors are in execution. That is, the overall elapsed time of higher level systems is of no concern in this calculation, only the absolute time of execution of the processors being considered. This is called the absolute power. Using the example of the cpu again, the absolute power of the cpu is the work performed divided by the seconds of cpu execution time required to do so. Symbolically, one has $P(L,\gamma) = W(L,\gamma) \div Tx(L,\gamma)$.

Relative power is related to absolute power by the corresponding percent utilization factor. In the cpu example, the relative cpu power equals the product of the quantity percent cpu utilization and the absolute cpu power. Since cpu utilization equals $Tx(L,\gamma) \div Tx(L,\psi)$, one has $P(L,\gamma,\psi) = (\%\text{ cpu utilization}) \times P(L,\gamma)$.

The absolute power usage arises from the equipment speeds and basic workload parameters such as instruction mix and block sizes. Relative power usage levels reflect absolute power parameters, the proportions of power between subconfigurations and equipment classes, and the ratios of work to be performed in these subsystems. That is, relative power usage levels reflect both basic workload parameters and the "fit" between the workload and the computing system.

Since absolute power calculations do not require knowledge of the overall workload characteristics, the theoretical absolute power available from a device, subconfiguration or equipment class can be calculated. The theoretical considerations include the factors which in general can affect the absolute power levels attainable from the equipment. As such, they identify the effect of changes and provide a means of determining the power loss due to the way the equipment is being used. For a given quantity of work, it is then possible to calculate the changes in execution time which will result from a new level of absolute power usage obtained by altering the manner of equipment use.

Relative power involves the use of elapsed time, and overall elapsed time can be predicted from a knowledge of the absolute power usage levels attainable and the quantities of work to be performed by equipment class and/or subconfiguration. This is accomplished using a full workload characterization with "offset" information. Offsets represent the quantity of cpu work to be performed before a quantity of I/O work can be performed. They are established empirically for a given workload from execution time profiles and their equivalent form, work concurrency charts. For additional information, see the corresponding entries in this glossary.

The comparison of actual power levels to theoretical power levels, and the determination of the relative importance of the factors degrading actual power from the maximum attainable power, provides the knowledge necessary to formulate an installation performance improvement plan. Inherent in such a plan would be a recognition of the cost-effectiveness of various possible performance improvements, and trade-offs between these activities and additional equipment plans. The same knowledge needed for this plan is necessary to correctly evaluate and understand the effects of possible new equipment on performance.

PROCESSORS

A processor is any collection of digital circuitry which is capable of accepting an instruction and executing it; i.e., generating the set of logical state changes represented symbolically by the instruction. Typical processors are cpu's, disk drives, tape drives, printers, terminals, various control units, channels, etc.

It is not true that processors of interest need to be separately packaged as a distinct physical entity. For example, disk drives often come two or more to a package. For this reason, a subconfiguration composed of several different device-level processors can also be considered as a single processor. For example, a channel device with attached control units and disk drive devices can be treated as if it were a single processor. Such processors are called equivalent processors when necessary to distinguish between processors packaged in a single box and processors whose circuitry is distributed among several boxes.

Additionally, the ability of a configuration to handle a forecasted workload needs to be determined. It is much more convenient for these purposes to use an equivalent form of an execution time profile expressed in terms of software work rather than time. Such a chart is called a work concurrency chart. It is constructed by multiplying the execution time components of the profile by the corresponding actual absolute power levels. The overall Pert structure of the profile is reflected into the work concurrency chart by offsetting I/O work by equipment class by an amount equal to the cpu work corresponding to the time the cpu is in execution but not the equipment class. These quantities of cpu work are called "cpu offsets", and one per equipment class is calculated.

Given a work concurrency chart representing the typical offsets found in an installation workload, changes in the quantitites of work by equipment class and/or observed absolute power levels are easily translated to an execution time profile. More importantly, workload forecasts can be translated to execution time profiles. The elapsed time of the new system with the new or forecasted workload is also easily calculated. The new percent utilizations are also predicted by the same calculations.

RESPONSE TIME

Response time is normally associated with the elapsed time between inputing a command or inquiry to an on-line system and receiving a response. Since the work to be performed is a function of the nature of the input, two major techniques for determining response time are used: 1) the time for a standard input or set of inputs is measured, or 2) a percentile of actual response times is chosen, e.g., "85% of all response times are equal to or less than 5 seconds."

In software physics, response time is clearly a function of the work vector corresponding to the input, and the vector power delivered into the on-line system on behalf of the input. Conceptually, given these two quantities, determination of response time is a straightforward calculation. In practice, these quantities are often difficult if not impossible to obtain due to lack of proper instrumentation. However, it is interesting to note that queuing theory parameters needed for response time prediction are generally adequate for software physics purposes as well.

SOFTWARE PHYSICS

Software physics is the study of the quantitative and measurable properties of executable instructions and their operands, and their interactions with computing systems equipment and configurations.

SOFTWARE PHYSICS NOTATION

Software physics uses a special form of notation designed to identify three items of interest:

1) the property to be measured. The properties of general interest are software work *(W)*, execution time *(Tx)*, elapsed time *(Te)*, storage occupancy *(R)*, available store *(Z)*, Power *(P)*, storage capacity usage *(Cσ)*, and Intensity *(I)*.

2) the unit of software whose executable code and/or operand properties are to be measured. The symbols used are *S* to represent a general software unit and *L* to represent that software unit representing the full workload. Subscripts are used to denote constituent software units and/or subworkloads. For an actual software unit, called say "Job XYZ", the actual name "XYZ" would be used instead of *S*. Similarly, the word "Batch" might be used for the batch subworkload.

3) the set of equipment over which the value of a desired property is to be obtained. Either a configuration or subconfiguration, or an equipment class (but not both unless they are identical) may be specified. Configurations are identified by lower case Greek letters, equipment classes by abbreviations. Typical configuration symbols used are ψ for the full configuration, γ for the cpu, φ for I/O, α for a channel subconfiguration, β for a control unit subconfiguration, and δ for a drive. Equipment class abbreviations are *cpu* for control processor, *disks* for disk drives, *tapes* for tape drives, *ptr* for high speed printers, etc.

The structure of the notation permits a desired measurement to be symbolized precisely. The property to be measured is given first, followed in parentheses by the software unit(s) to be measured and then by the configuration(s) or equipment classes to be measured. If the property is to be measured for more than one software unit, both are given, in the order of their occurrence in the corresponding equation. Similarly for configurations and equipment classes.

Vector representations are denoted by an arrow over the property symbol. Examples:

$Tx(L,\psi)$: the equation time of the full workload on the full configuration.

$Tx(L,disks)$: the execution time of the disk drive equipment class for the full workload.

$W(L,\psi)$: the software work of the full workload on the full configuration (a single number)

$\vec{W}(L,\psi)$: a vector representation of the quantity $W(L,\psi)$.

$P(L,\gamma,\psi) = W(L,\gamma) \div Tx(L,\psi)$: the relative cpu power.

SOFTWARE PHYSICS PROPERTIES

The term "properties" denotes a measurable, quantitative characteristic of software units and/or computing configurations or devices. There are three fundamental properties; software work, execution time, and storage occupancy. These and only these properties (or their equivalents, energy, time, and available storage) are used in software physics. Certain important other properties are derived using the fundamental properties. These are called derived properties, and include power, storage capacity usage, intensity, force, and distance.

To obtain an actual measurement of some property, both a unit of software and the computing equipment must be specified. These are called software physics systems, and the property is a characteristic of the systems being measured.

See the corresponding glossary entries for more discussion.

SOFTWARE UNITS AND WORKLOADS

A software unit is a basic system of interest in software physics. It is formally defined as an arbitrary collection of executable (object) code and its associated operands. A software unit therefore, corresponds to a program with its associated data, an application and data, the full workload and data, and even a single instruction and its operands. Since it is defined in such a general way, software physics theory requires that any statement made about a general software unit be true for all software units. Also, since a software unit can be a single executable instruction and its operands, this requires that only those quantitative properties displayed by such a software unit can be associated with all software units.

A workload is a special software unit only in the sense that it represents the total collection of executable code and data over some period of time. Because of this however, certain statements and equations true for the total workload may not be true for all software units. The reverse of course is true; i.e., any statement about a software unit holds for workloads as well.

TIME

Time is a basic property of software physics. Its unit of measure is seconds, as measured by a standard clock. Time is a measure of state change processes, and a standard wall clock is assumed to be measuring universal state changes. In software physics, the basic time quantity of interest is called execution time, symbolized as Tx. For a given software physics system, Tx is increased if and only if an instruction is being executed by some processor. If this condition is not true, then even though the wall clock time may increase, the corresponding execution time increase will be equal to zero time.

For example, a unit of software S may be in execution on a configuration ψ from t_1 to t_2. Its execution time during this period is $Tx_1(S,\psi) = t_2 - t_1$. At time t_2, the software unit is capable of being executed (is dispatchable), but is "involuntarily" caused to wait until time t_3. The execution time during this period $t_3 - t_2$ is equal to zero; i.e., $Tx_2(S,\psi) = 0$. From t_3 to t_4, S is again in execution; $Tx_3(S,\psi) = t_4 - t_3$. If S is now completed, the total execution time would be $Tx_1(S,\psi) + Tx_2(S,\psi) + Tx_3(S,\psi) = (t_2-t_1) + 0 + (t_4-t_3)$.

A second form of time, called elapsed time and symbolized as *Te* is also of interest. Elapsed time is not an independent quantity, as it is defined in terms of execution time. Formally, the basic definition is that $Te(L,\psi) \equiv Tx(L,\psi)$. That is the software physics elapsed time *excludes* any pure idle time; i.e., *Te* is a measure of state change processes which occur within the entire configuration. If no instruction (cpu or I/O) is occurring, then the change in *Te* is zero even though the wall clock time is being incremented. The difference between wall clock time and elapsed time is called idle time.

Elapsed time measures occupancy of storage, execution time measures instruction execution time within the configuration, subconfiguration, or equipment class of interest. The elapsed time of a given unit of software *S* is measured by the changes in the quantity $Te(L,\psi)$ from the point in wall clock time that *S* occupies storage and is capable of being executed until it has been completely executed and no longer occupies storage. This quantity would be symbolized as $Te(S,\psi)$. By definition, it will always be true that $Te(S,\psi) \geq Tx(S,\psi)$.

Pure idle time is not used directly in software physics equations, except that it represents power that could have been delivered and was not. As such, when determining the capacity remaining on a configuration, pure idle must be considered. Since pure idle time is equal to wall clock time minus execution time, the remaining capacity is always a function of "scheduled-on time" for computing system.

WORK, SOFTWARE

Software work is one of the basic properties of software physics. In general, work is performed when a change in state occurs. In software physics, a processor executing an instruction will perform work on a storage device when the processor causes a symbol state change to occur. The standard symbol size is defined as an eight bit byte, resulting in the following formal definition:

> A processor performs one unit of software work (called a "work", (symbol w) on a storage device when it changes the symbol state of one byte of storage.

The instrumentation problem of observing if a transfer of one byte to storage actually causes a symbol state change results in the following operational definition of software work:

> A processor performs one unit of work on a storage device when it transfers one byte to that storage device.

Software work is measured in units of "work" or "works", symbolized by a lower case w. Normal metric prefixes are used for larger quantities; i.e.,

1,000 works = 1 kilowork = 1 kw
1,000,000 works = 1 megawork = 1 mw
1,000,000,000 works = 1 gigawork = 1 gw
1,000 kw = 1 mw
1,000 mw = 1 gw

Software work has the property that the whole is simply equal to the sum of its parts. For example, if the cpu work of some software unit S_1 is $W(S_1,cpu)$ and of some software unit S_2 is $W(S_2,cpu)$, then the cpu work performed by both is simply $W(S_1,cpu) + W(S_2,cpu)$. This is true whether S_1 and S_2 are executed concurrently or sequentially.

It should be explicitly noted that a software unit is a collection of executable code and data: the same executable code over different data is formally a different unit of software. Therefore, software physics does not imply that two different runs of the same program over different data will result in the same quantity of software work. In fact, since the term "data" in software physics includes the sequence in which operands are presented, different sequences of the same operands need not result in the same quantities of software work.

Software work may be measured directly with a hardware monitor or software monitor in many instances. However, two standard approximation equations are generally used. These are:

1) work = (number of instructions executed) × (av. work/instruction)

2) work = (average power) × (seconds of execution time)

The first approximation equation is most often used when the number of I/O read/write actions or instructions are known, and also the average block size read or written. The equation thus becomes:

I/O work = (#I/O reads/writes)(average block size)

The quantity "#EXCP's" is given by the IBM instrumentation software known as SMF. Using this as an approximation to the number of I/O reads and writes, one has

I/O work = (#EXCP's)(average block size)

The second equation is used when cpu seconds (of execution time) is known. A hardware monitor is used to establish the average cpu power, and the approximation equation then becomes:

cpu work = (av. cpu power)(no. of cpu seconds)

LMED

-8