

data entry newsletter

Product Marketing

data entry

Date: June, 1981

No. : 323

Type: Marketing & Technical

NIXDORF 600 "SERIES"

-VS-

1900/10

This newsletter may look too long and too technical, but the marketing explanations attached to the technical ones should make reading worthwhile. After reading the "naked truth" about Nixdorf's 600 you will never need to use the excuse that you lost because they were 40% less money than a 1900/10. That excuse just died.

Nixdorf is so intent on dumping the Entrex name that they're gradually replacing nameplates on their installed equipment. However, changing the name hasn't improved the product. It is still an Entrex with all its inherited inefficiencies. The 600 Series operating system (DPEX) is not a truly new design, but it has a few capabilities (such as 80 Series (ADEX) operating system. In an effort to keep 80 Series software upward migratable, the DPEX operating system was forced to retain many of the years old, keypunch replacement, design and programming concepts inherent in the older systems.

Nixdorf has been successful in selling as benefits "Features" we believe are shortcomings. For example, in some selling situations against Univac they took virtual memory with its "thrashing" problems and said "you never have to buy more core like a 1900/10". They told people that a system save (a complete snapshot of disk & memory) should be taken 3-4 times a day and it only takes a "few minutes" (approx. 1 min. per MB). They compared that to our job, compile or batch saves which take longer and claimed an advantage. They neglected to mention that you have to shut down all operators & 3270 comm since system saves are offline and that retrieval of one batch from a system save is a real trauma (we discuss that technique later).

The best way to beat Nixdorf is:

1. Be there first! Being the 2nd or 3rd vendor into any deal always puts you on the defensive. Their prospect is already sold on their strong points (everybody has at least 1 or 2), and you have a big un-selling job to do. That's never desirable.
2. Know our strengths and their weaknesses. By positively selling your strengths, it is up to them to do the thankless and often negative task of unselling your prospect.

This newsletter will point out many ways in which a 1900/10 will do a better job for your prospect than any member of the 600 series.

GENERAL MARKETING INFORMATION

Prices on individual pieces of the 600 Series are included here. (They are not available in Datapro). Prices were effective as of Feb. 1981. List prices for Nixdorf are quite close to the 1900/10 pricing structure. It appears that considerable discounting flexibility must be available at a branch level since we experience such erratic pricing levels in competitive situations. In one city a bid for 10 systems may be bid at 0-5% discount, and in the next a single system may be bid with a 35% or greater discount. Recently the incidence of deep discounting by Nixdorf has increased. This nearly always is a sign that bookings are off dramatically.

The 600/25 system is not sold as a stand-alone device, it is marketed as a remote terminal system for the 600/35, 600/45 and 600/55 systems. The differences between the 35, 45 and 55's are apparently artificially created. The change of processor cycle time from 1100 NS to 750 NS to 400 NS on these models is (according to what we are told) accomplished on site with a tunable processor chip. The amount of real memory installed by Nixdorf is based on their pre-sales analysis of the customers throughput requirements. If the system throughput deteriorates, they may add memory, at no charge, up to the max. for that model. If that isn't sufficient memory, the customer can be asked to upgrade to the next model, and they actually can end up paying for more memory, contrary to the presales talk. We are told that the limits on device counts are a function of the sysgen. Therefore, when the Nixdorf user upgrades to a new member of the 600 Series, he appears to basically receive: another memory board, a retuned processor chip and a fresh sysgen. The 600 "Series" appears to be a "Series" in name only.

As is always true of any competitive comparison, time changes things. Caution should be exercised when comparing our product to any other product. Every attempt is made to make any competitive analysis accurate and up to date. It is always possible that our understanding of a competitor's product is incomplete or outdated. Enhancements do occur that eliminate a knock-off point.

The best way to use an analysis of this type is to read it, understand it and sell those features that we know are our strengths. Make sure that those features relate to benefits that the customer agrees are beneficial to him. Do as little direct comparison to a specific competitor as is possible. If you sell your 1900/10's strengths and it turns out later that competition has repaired a deficiency, you are not hurt since you never said he could not do it, you only stated that you could. Most customers assume that your selling points are strengths or exclusive features. If you have convinced them that they need the feature they will check with competition, and they will force competition to justify their shortcoming.

It is often a temptation to hand prospects a comparison chart or a knock-off list on your competitor. Don't do it! Hope instead that competition hands your prospect a knock-off list. In all probability it contains 2-3 mistakes. Once you point out their "falsehoods" the entire list is discredited, and their credibility usually suffers, not yours.

This newsletter is written in a very candid style and, as a result, should not be shown to any user. Using this approach in a selling situation will only prove to be a liability, not an asset.

Competitive selling must be positive selling if it is to succeed.

600 "SERIES" - CONFIGURATIONS

	<u>600/35⁷</u>	<u>600/45⁷</u>	<u>600/55</u>
MAX REAL MEMORY ¹	64K	96K	128K
MAX VIRTUAL MEMORY	2Mb	3Mb	8Mb
MIN/MAX DISK ²	4.8/9.6 Mb (2 DRIVES)	33/132 Mb (2 DRIVES)	33/264 Mb (4 DRIVES)
MIN/MAX DISKETTES	0/2 (315K ea)	0/2 (315K ea)	0/2 (315K ea)
MIN/MAX TAPE (45IPS)	1/1	1/2	1/4
MAX TERMINALS ³	8	16	32
MAX TERMINAL PRINTERS ⁴	8	16	32
MAX SYSTEM PRINTERS ⁵	1	1	1
MAX CARD READER (300CPM)	1	1	1
MIN/MAX COMMUN. CONTROLLERS ⁶	1/2	1/2	1/2

- NOTES: 1. LESS REAL MEMORY MAY BE INSTALLED INITIALLY. THERE WILL BE NO CHARGE TO UPGRADE TO THE MAX MEMORY FOR THE CUSTOMERS MODEL IF SYSTEM THROUGHPUT HAS DETERIORATED & MORE MEMORY IS NEEDED. CHANGING TO THE NEXT HIGHER MODEL USUALLY MEANS ADDITIONAL CUSTOMER CHARGES.
2. THE 600/35 CPU HOUSES ITS 4.8Mb FIXED DISKS. THE 600/45 & 600/55 USE A LARGE ADDITIONAL CABINET TO HOUSE EACH 66 Mb DRIVE. THE 66 Mb DRIVE IS A SINGLE REMOVABLE PACK. THE 33 Mb DRIVE IS A STRAPPED DOWN 66 Mb.
3. BOTH 480 & 1920 CHAR CRTS ARE SUPPORTED.
4. USE OF A TERMINAL PRINTER LOCKS UP A TERMINAL WHILE PRINTING. THEORETICALLY 32 TERMINAL PRINTERS CAN BE ATTACHED TO A 600/55 BUT, IT ISN'T LIKELY THEY COULD BE DRIVEN SIMULTANEOUSLY AT AN ACCEPTABLE SPEED. TERM. PRINTERS ARE AVAILABLE AT 45 CPS, 165 CPS & 300 LPM. SOFTWARE-WISE, ANYTHING THAT CAN BE PRINTED OUT ON THE SYSTEM PRINTER CAN BE PRINTED ON A TERMINAL PRINTER.
5. THE SYSTEM PRINTER IS AVAILABLE IN 165 CPS, 300 LPM, 600 LPM & 900 LPM MODELS.
6. THE TWO COMMUNICATIONS CONTROLLERS ONLY ALLOW CERTAIN PROTOCOL COMBINATIONS SIMULTANEOUSLY; BYSYNC & 3270, OR HASP & 3270.
7. IF A 600/35 HAS BEEN BID AGAINST YOU, REMIND THE PROSPECT OF THESE FACTS. WHEN THE CUSTOMER ADDS HIS NINTH TERMINAL, THAT ONE CRT REQUIRES CONSIDERABLE COST ABOVE THE COST OF THE TERMINAL. 1. UPGRADE TO A 600/45 CONTROL GROUP (\$255/MONTH). 2. AT LEAST 1-33Mb WHICH IS MORE COSTLY THAN 2 - 4.8Mb DRIVES (\$203/MONTH) THATS \$458/MONTH PLUS MAINTENANCE FOR A RETUNED CPU CRYSTAL AND ADDITIONAL DISK HE MAY NOT NEED. THE UPGRADE FROM 16 TO 17 TUBES COSTS AN EXTRA \$200/MONTH PLUS MAINTENANCE FOR THE CONTROL GROUP UPGRADE.

NIXDORF 600 "SERIES" PRICING

PRICES EFFECTIVE FEB. 1981

<u>MODEL NO.</u>	<u>DESCRIPTION</u>	<u>ONE TIME INSTALLATION CHG</u>	<u>3 YEAR</u>	<u>5 YEAR</u>	<u>PURCHASE</u>	<u>MAINT</u>
600-35	CPU	255	166	153	7970	35
600-45	CPU	285	443	408	21260	81
600-55	CPU	285	643	592	30860	113
161A	480 CHAR CRT	30	36	33	1730	23
162A	1920 CHAR CRT	30	48	44	2300	28
	KEYBOARD	-	10	9	580	-
	DESK	-	9	8	250	-
416A	10½ 9TR-1600 BPI	60	416	382	19970	114
418A	10½ 9TR-800 BPI	60	346	318	16660	92
417A	10½ 7TR 556/800 BPI	60	346	318	16660	92
263A	4.8 Mb	115	176	162	8450	69
*224A	33 Mb 1ST DRIVE	230	572	527	27450	98
*224B	33 Mb ADDIT. DRIVES	230	496	457	23810	92
*225U	33 Mb UPGRADE TO 66 Mb	115	147	136	7060	24
225A	66 Mb 1ST DRIVE	345	719	663	34510	122
225B	66 Mb ADDIT. DRIVES	345	643	593	30810	116
**515C	45 CPS	30	119	109	5710	46
**510&510C	165 CPS	60	189	174	9070	70
**533&533C	300 LPM	115	396	365	19000	100
536	600 LPM	115	584	538	28030	119
539	900 LPM	115	753	693	36100	158
621A	COMMUNICATION CONT.	30	73	67	3500	35
621F	2ND COMM. CONT (3270 PROTOCOL ONLY)	30	73	67	3500	35
710	CARD READER (300 CPM)	115	195	180	9360	43
601	CTC (CNTRL TRNSMSSN CONTROLLER)	175	50	46	2400	11
616A	CTI (CNTRL TRNSMSSN INTERFACE & MODEM)	30	35	32	1680	11
616E	CTI (WITHOUT NIXD. MODEM)	115	21	19	1000	9
617A	TTA (TRMNL TRNSMSSN ADAPTER & MODEM)	60	35	32	1680	11
617E	TTA (WITHOUT NIXD. MODEM)	115	21	19	1000	9
605	DAA (DATA ACCESS ARRANGE- MENT)	30	10	10	150	-
624	RTC (REMOTE TERM CONCEN- TRATOR FOR 4/TRMNLS)	115	77	71	3690	31
628	RTC (REMOTE TERM CONCEN- TRATOR FOR 8/TRMNLS)	115	119	109	5710	47

NOTE: ALL PERIPHERAL PRICING INCLUDES CONTROLLERS
ALL SYSTEMS REQUIRE TAPE & COMMUNICATIONS

* 33 Mb DRIVES ARE STRAPPED DOWN 66 Mb DRIVES

** PRINTER NO'S WITHOUT "C" SUFFIX ARE SYSTEM PRINTERS
PRINTER NO'S WITH "C" SUFFIX ARE TERMINAL PRINTERS

NIXDORF 600 "SERIES" PRICING

EFFECTIVE: FEB, 1981

<u>SOFTWARE</u>	<u>MONTHLY RENTAL</u>	<u>NOTES</u>
470P DPEX O.S.	53	REQUIRED ON ALL SYSTEMS
472P DPEX SORT	17	REQUIRED ON ALL SYSTEMS
471P DPEX SECURITY	17	EXPANDS SECURITY BEYOND A SYSTEM PASSWORD.
474P DPEX EXTENDED 3270	17	REQUIRED IF 3270 IS TO BE USED ON SYSTEM.
475P DPEX LOG BOOK	17	THIS IS NIXDORF'S ATTEMPT TO EMU- LATE OUR LOGGING TO THE PRINTER. THEY ONLY SPOOL TO DISK IN THE SYSTEM AREA OF THE DISK. (THIS IS POOR USE OF A VITAL & LIMITED POR- TION OF THEIR DISK AND NOT MANY USERS PURCHASE IT.) AFTER SPOOL- ING THEY CAN OUTPUT TO THE PRINTER BUT, THEY CAN'T DO IT ON THE FLY AS WE DO. USE IT AS AN ADVANTAGE. IT IS!
478P DPEX HASP	27	MUST HAVE 1 TERMINAL PRINTER IF HASP IS ORDERED.

INTRODUCTION TO THE 600

In selling against Nixdorf or comparing our system to theirs, a few terms must be defined. Nixdorf uses words to describe their programming techniques and systems that have unique meaning to them. Prospects and Sperry Univac personnel can get rapidly confused when trying to compare the two systems. This writeup will attempt to define Nixdorf terminology in terms that allow clearer comparisons of the systems.

In an effort to shift the image of the 600 from a Data Entry to a more distributed processing image, Nixdorf has confused their terminology even more. The term "Batches" has disappeared, they are now called work files. Indexed (Data Base) files are called Master Files. Nothing actually changed but the name.

PROGRAMMING CONSIDERATIONS

Programming techniques on the 600 are very different from the 1900/10. The 1900/10 uses several levels linked together to create a format program. "Format" or "Format Program" is Nixdorf's name for a level. A 1900/10 has a formal header which defines record size, tape blocking, labeling, level selection, etc. These functions are not covered by the individual format programs (levels) on a 600. They create another module called a "Standard Job". A standard job is not an autoprompt batch, it simply replaces our level headers and defines the levels and linkages for this complete program. A record format (entry) program is the checkbox portion of the entry program. Field end edits (their editor language commands) are similar to our Cobol procedures but are not put on the check box record format form. They are a separate program (using different forms) and are linked logically to fields in the record format. Each field in the record format program has an "Enable Edit" check box. If "Y" is entered, the program links out to the separate field end edit program. Each procedure must be identified by its related format (level) number and field number in order to execute properly. If any field is added or deleted in the record format program, the relative field number changes and the field number references in the field end edit program must be altered also. They cannot link to procedures by field names.

If the customer desires extensive double checking, cross footing, subtotalling, totaling or some other function which is best performed at batch end, a file (batch) end edit program is required. File end edit programs execute only as foreground tasks. The operator sits and watches as each record is momentarily flashed on the screen during the file end edit function.

Any output of a batch requires an output program. If output is the same as input, it's a simple program with one procedure "output all". New output field positions, justification of output fields, field fills at output or conversion to packed decimal must be controlled in the output program (not in the checkbox like a 1900/10).

The 1900/10 can perform the non-checkbox output functions as part of our single, entry program or as a separate output program as is required on the 600.

Output programs, file edit programs, and field end edit programs, all link their procedures to the record format (entry) program by record format (level) number and relative field number. Debugging of new or changed programs is a real challenge to the user.

The 600 allows variable length records (levels) in a batch. (Since few mainframe applications allow variable length records, this Nixdorf "improvement" is corrected, by padding the records, at output time.) As a result dup information, in fact, no information can be carried forward to the next level unless the data is moved to a variable. One function of a variable, in the 600, is to act as a register. Variables have a maximum size of 14 numeric char. or 20 alphanumeric char. The 20 alpha character limit is a major one. To get around the 20 character limit the programmer must break a field down into 20 character pieces with subscripting. To our knowledge, this is not a customer requested enhancement, it's just the way a 600 works. Variables are also used to define a key with which to search a file using data base management. The variable record length also means that mid level selects are not allowed, another disadvantage.

Following is a comparison chart of 1900/10 to 600 Programming Features:

PROGRAMMING COMPARISON

1900/10	FEATURE	600
COBOL	PROGRAMMING LANGUAGE ¹	"EDITOR LANGUAGE"
1 OPERAND TO MULTIPLE DESTINATIONS	"MOVE" VERB ²	1 OPERAND TO 1 DESTINATION
YES	"CONNECT" VERB ³	NO
YES	"ALLOW"&"DISALLOW" VERBS	NO
MULTIPLE OPERANDS TO A DESTINATION	"ADD"&"SUBTRACT" VERBS ⁴	1 OPERAND TO 1 DESTINATION
ROUND WITH THE REMAINDER	"DIVIDE" VERB ⁵	TRUNCATE REMAINDER & SET OVERFLOW INDICATOR
OR		
PLACE REMAINDER IN A REGISTER		
OR		
TRUNCATE AND SET OVERFLOW		
YES	PROGRAMMERS COMMENTS ALLOWED	YES
YES	"DISPLAY", LITERALS, FIELDS & REGISTERS	YES
YES	SUBSCRIPTING	YES
READ OR WRITE	PACKED DECIMAL	WRITE-YES, READ-NO
CAN BE "CALL"ED BY MULTIPLE PROGRAMS	SUBPROGRAMS ⁶	EMBEDDED IN FORMAT PROGRAM
ALPHA OR NUMERIC	RANGE CHECK	NUMERIC ONLY
13 NUMERIC + SIGN	REGISTER (NIXDORF "VARIABLE") SIZE ⁷	14 NUMERIC (INCLUDES OVERSIGN) 20 ALPHA
542 ALPHA		
YES	OPERATOR DISPLAY OF REGISTERS ⁸	NO
32	LEVELS (FORMATS) PER PROGRAM ⁹	10
32,767	TAPE OUTPUT BLOCK MAX	4,096
99/PROGRAM	TABLES-MAX. NUMBER ¹⁰	20/SYSTEM
RESIDENT-MAIN MEMORY	TABLES-PHYSICAL LOCATION	ON DISK
YES	FIELD DESIGNATED TAB STOP & START	YES
ALPHA, NUMERIC, PUNCTUATION, BLANKS, OR USER DEFINED TABLES	DATA "TYPE" VALIDATION	ALPHA, NUMERIC
999	FIELD SIZE	99
323	KEY SIZE-INDEXED FILES ¹¹	36
INDEX TREE	KEY STRUCTURE	INDEX TREE

PROGRAMMING COMPARISON (CONTINUED)

1900/10	FEATURE	600
YES	FIELD END EDITING ¹²	YES
YES	PRINTER SHARING	YES
YES	FORMAT PROGRAMS USE TERMINAL PRINTER ¹³	YES
YES	AUTO PROMPT (NIXDORF COMMAND MACROS)	YES
99	BATCH BALANCE ACCUMULATORS ¹⁴	5
1,344 BYTES	MAX READ PER I/O FROM DISK	240 BYTES
YES	DATA GENERATED AT OUTPUT TIME ¹⁵	YES
NO	FIELD PROMPTS STORED ON DISK ¹⁶	NO
STANDARD	PROGRAM CONTROLLED SECURITY	YES - EXTRA COST ITEM
NONE	CHECK DIGIT LIMIT ¹⁷	15
YES	TAPE LABELS TO PREVENT OVERWRITE ¹⁸	YES
60 FILES	# FILES OR INDEXES OPEN PER PROGRAM	4 INDEXES
SIMPLE	PROGRAM MAINT. & DEBUGGING ¹⁹	VERY COMPLEX
YES	CONTINUATION FIELD	NO
YES	1 PROGRAM FOR INPUT & OUTPUT	NO
NO	VARIABLE RECORD LENGTH ²⁰	YES
YES	RECORD (NOT FILE) LOCK	YES
YES	SEQUENTIAL I/O	YES
AUTOMATIC	REPLAY ²¹	PROGRAMMERS RESPONSIBILITY

FOOTNOTES: PROGRAMMING COMPARISON

1. Although Nixdorf often refers to their Editor language as Cobol or Cobol-like, it actually bears little resemblance to Cobol, in verbs, verb function or language syntax. It isn't even English-like, which they often claim.
2. The 1900/10's - MOVE A to B, C, D is much less code (memory conservation) and executes more quickly than the multiple "MOVE" statements on a 600.
3. Nixdorf's lack of "CONNECT", means multiple "MOVES" using subscripting, on the 600, both time and code (memory usage) consuming.
4. ADD A, B, C, D, giving E or SUBTRACT A, B, C, D, from E saves time and code over multiple "ADD" & "SUBTRACT" statements as is required on the 600.
5. To always truncate remainders is not an acceptable practice.
6. One set of code for a subprogram can be used by multiple programs. The 600 requires that the subprogram be encoded in each program that uses it - wasteful coding.
7. A 20 alpha char max. for variables (registers) is a major programming difficulty. Although their indexed files allow 36 char. keys only 20 can be addressed. Data cannot be duped or carried forward from 1 level in a program to the next except within this extremely restrictive 20 character limit.
8. The lack of the 600's capability to display registers by operator keystrokes is a real drawback to a programmer debugging a program or to an operator trying to look at out of balance registers.
9. The 600's 10 levels (formats) per program often requires ingenious and user un-friendly programming to complete a complex application.
10. The 600's single entry tables are searched sequentially. The replacement (double entried) tables are very similar to their indexed files. The argument is established as its index and an index tree is built. The upper levels above level 0) are stored in "memory" (actually they are in the virtual memory area on disk not in main memory). Each level requires an I/O to the disk, this is in miliseconds (not microseconds like our memory resident tables). They can create indexed files as tables but this requires "OPENS" "READS" "CLOSES" etc. by the programmer. That increases program size and maintenance problems. Referencing a table number in the check box portion of their format program is easier. On a

600, tables are accessed by specifying the numbers 1-20 on the master table list. Tables must be loaded and unloaded by the supervisor to/from the master list in order to be accessed by format programs. On a 1900/10 the supervisor does nothing to make tables available to a program. Our programs simply access the table by name. The 20 table limit is a limit for the entire system so indexed tables are sometimes used to circumnavigate the problem. They also recommend mixing various types of data in a table. Updating a mixed table may be difficult if any data is duplicated between the lists that were merged.

11. Although the max, key size on the 600 is 36, remember their "VARIABLE(s)" limit the search to 20 characters.
12. By forcing field end edits into a separate program, debugging, time to link out to each procedure, & program maintenance create big problems for the 600.
13. The terminal printer locks up the terminal while printing'
14. The limit of 5 batch balance accumulators creates challenges to the programmers and chronic complaints from Nixdorf users. 40 additional balance registers can be used in a program but are established by using the "SET" verb in a field end edit. Problem: It is our understanding that if an operator changes a value in a field affecting balance registers, only the 5 system balance registers will be automatically corrected. The additional 40 registers in the field endedit won't be automatically updated. If the user put these additional registers in a batch end edit program as a "Workaround" the operator sits idly by at batch close as this foreground jpb is performed.
15. Nixdorf often tries to say that they can store dup or generated fields once on disk and perform this generation at output and that a 1900/10 can't. This is not true. We can perform those functions at output also.
16. This comment is only to combat another erroneous charge by Nixdorf against the 1900/10.
17. The 600's 15 check digit limit will be sufficient for most users. We have a number of financial institutions processing master charge slips and that requires over 60 different check digits. Know your prospects application.
18. Only four indexes (whether on separate files or all on the same file) can be open simultaneously in a program on the 600. This has created a programmer's nightmare of opens and closes in an application using multiple indexes.

19. By having separate modules (standard job, format (entry) programs, field edit program, file (batch) end edit program and output program) linking their statements together by format (level) number and relative field number in that level, the programmer is offered many frustrating hours of trying to tie it together and get it running. This frustration is a "No Charge" item from Nixdorf.
20. If customers (with big, cheap disks) need to save a few bytes, Nixdorf's variable record length can be of help in certain applications. The bytes saved however, mean that dup and other information that must be moved to the next levels must fit the 20 char (variable) limit. Midlevel selects, another operator oriented benefit, is out due to the variable record length. Is the space benefit truly worth the other major drawbacks?
21. Nixdorf's lack of full replay is another example of how they sometimes try to sell "Less as More". The 1900/10 offers full replay as a standard part of the operating system. Don't sell it short! It's an outstanding feature! Without replay you're still in the stone age of data entry.

In 1974 B.R. (Before Replay) everyone handled replay manually, and some still do. For example, in a payroll application, if the operator keyed 60 hours instead of 40, she used to back up to the field and first key a minus 60. It was her responsibility to back the erroneous hours out of the balance or other registers. She then rekeyed "40" the correct amount. Today the 1900/10's replay handles this for the operator. In the days when Cobol procedures began to be executed at field end (rather than record or batch end) new problems arrived. Now keying a 60 and field releasing it could calculate the persons pay and store it in subtotal and total registers. For the operator to make her own corrections without Cobol replay means she has to understand fully each facet of the format program. Field End Editing, using Cobol replay, can vastly simplify correction procedures for an operator or programmer. We introduced full replay with the introduction of the 1900 in 1974.

With replay the 1900/10 establishes checkpoints as records are keyed into a batch. Batch balance accumulators, registers, etc., are recorded as the checkpoint is passed. If an operator backs up and makes a change which affects subsequent results, "Replay" replays the program, including the Cobol procedures starting at the checkpoint prior to the change(s) just made by the operator. Without the auto-correction of replay, the user could have many errors in the batch due to one field being changed. In a 600, only the 5 balance registers are automatically corrected. The

field end edits are not re-executed. It is the programmers responsibility to program if correcting techniques into each program. In one of Nixdorf's writeups where they try to knockoff replay, they said that if an operator uses our search mode to locate and change a record in an existing batch, replay is invoked and the system executes all the Cobol in records preceding the desired record. In a large batch this may take a minute or two. They felt that waiting wasn't worth it. In fact, now all the registers are current and the operators change will fully correct the entire batch.

We have years of refinement built into our replay code. The Nixdorf approach gives the user the opportunity to "Re-invent the Wheel" in each program. It's up to the user to debug each program's replay instructions as they (or if they) discover that replay is causing it.

Nixdorf sells their "User Code Replay" as an advantage. They say we always invoke replay even if the user doesn't want it. In truth, if we save all batch balancing and Cobol procedures for a batch end routine, replay won't be executed in the body of the program. We only execute replay when its needed, but we always execute it completely. Nixdorf says the operator has to wait while replay occurs, so why not get rid of it. True, they have to wait, but the accuracy of the results is worth it. It's no wonder that Nixdorf "nixes" replay. They don't have a replay technique that does all the correcting for the user that it could.

ENTREX SYSTEM STANDARD JOB DEFINITION

STANDARD JOB NAME _____

BATCH NAME _____

DATE _____

PAGE _____ OF _____

INPUT FORMAT NAMES:

#	NAME	LINKS TO	#	NAME	LINKS TO
1	-----	---	2	-----	---
3	-----	---	4	-----	---
5	-----	---	6	-----	---
7	-----	---	8	-----	---
9	-----	---	0	-----	---

RCD EDIT _____ BATCH EDIT _____

- 1 — OUTPUT DEVICE ---
- 2 — RECORD SIZE -----
- 3 — BLOCK SIZE -----
- 4 — FIXED LENGTH (Y/N)? —
- 5 — BLOCKED RECORDS (Y/N)? —
- 6 — CHARACTER COUNT -----
- 7 — PADDING CHARACTER ---
- 8 — CODE SET -----
- 9 — OUTPUT PROGRAM -----

- 1 Output Device
Tn indicates a tape drive. n = 1 through 4,
PR = printer TC = terminal control
CM = communications RE = remote entry
- 2 Record Size
For fixed-length record, the record size in characters.
For variable-length records, the maximum record size in characters.
- 3 Block Size
For fixed-length records, the block size in characters.
For variable-length records, the maximum block size in characters.
- 4 Fixed Length?
Y = records are fixed length (default).
N = records are variable length. When N is specified, block size must be a multiple of record size.
- 5 Blocked Records?
Y = records are to be blocked.
N = records are not to be blocked (default).
- 6 Character Count
FIELD RELEASE indicates no character count.
If character count is desired at the beginning of each record, specify the format desired.
D = decimal count
B = binary count
S = spaces
D and B cannot be specified together. Imbedded spaces are not allowed. Up to 4 positions can be specified for the count.
- 7 Padding Character
Any keyboard character. Space is the default.
- 8 Code Set
Library name of the desired code set. FIELD RELEASE indicates standard system code set.
9. Output Program
Name of the output program to be used.

ENTREX 600 SERIES RECORD FORMAT LAYOUT

STANDARD JOB NAME SALES RECORD FORMAT SALES1 ORIGINATOR _____ DATE _____ PAGE _____ OF _____

FIELD NO	TAG LOC		TAG END TAG WITH #	FIELD LENGTH	FLD USE	VERIFY	REVERIFY	MUST ENTER	MUST COMPLETE	BOUNDARY	RIGHT JUST	FILL DATA	BATCH NO DATA	BATCH BAL	ACC NO	VALUE TABLE NO	TABL ENTRIES	ORDER	RANGE CHECKS	LIMITS		AUTO CHECK NO	CHECK DIGIT	METHOD	MULTI FLD	UPDATE	LAST CHR	TAB FLD	ADDRESS	FORM LNA	ENABLE EDIT	INSERT/DELETE			
	LINE	COL																		LOWER	UPPER														
001	001	05	ORDER NO: #	06	N	K	Y	Y	Y																										
002	001	07	DATE: #	02	N	R					Y										I														
003	001	35	L#	02	N						Y										I														
004	001	38	L#	02	N					Y																									
005	002	01	NAME: #	20	L			Y		Y																									
006	003	01	ADDR: #	20	L					Y																									
007	004	01	CITY: #	14	L			Y		Y																									
008	005	01	STATE: #	02	A			Y	Y																										
009	005	11	ZIP: #	05	N					Y			S																						
010	005	22	PHONE: #	03	N					Y			S																						
011	005	32	-#	03	U					Y			S																						
012	005	36	-#	04	N					Y			S																						
013	000	00		01	L																														

- | | | | |
|---|--|--|---|
| <p>6. Field Use
A = alpha only field
L = lower shift (default)
M = numeric shift for both keypunch and typewriter keyboards
N = numeric only field
U = upper shift</p> <p>7. Verify
C = conditional verification
E = enter
K = key verification
R = end of verification for the current record including this field
S = scan verification (default)
V = visual verification</p> <p>8. Reverify
Y = mandatory reverification if COR is used</p> <p>9. Must Enter
Y = At least 1 keystroke is required</p> <p>10. Must Complete
Y = If entered, must be completed</p> <p>11. Boundary
Y = field boundary check</p> <p>12. Right Justify
Y = right justification</p> | <p>13. Fill Character
S = spaces
Z = zeroes</p> <p>14. Specifies S/Z if field is released, no data entered</p> <p>15. Batch Balance
A = add field content to an accumulator
S = subtract field content from an accumulator</p> <p>16. Accumulator Number
Enter the number (1 - 5) of the accumulator</p> <p>17. Value Table Number
Enter the number (1 - 20) associated with the field</p> <p>18. Invalid Entries
Y = table is a list of illegal entries for the field</p> <p>19. Table Order
A = ascending order
D = descending order
R = random order (default)</p> <p>20. Range Check
I = data must be in the range
O = data must be out of the range</p> | <p>21. Enter lower limit</p> <p>22. Enter upper limit</p> <p>23. AUTO Functions
D = dup
E = emit
I = increment
S = skip</p> <p>24. Check Digit Number
Enter the number (1-15) of the desired algorithm</p> <p>25. Check Digit Method
G = system generated
V = system validated</p> <p>26. Multifield Check Digit
Y = the check digit is calculated using several data fields</p> <p>27. Update
C = conditional verification
E = enter
K = key verification
R = end of update for the current record including this field
S = scan verification (default)
V = visual verification</p> | <p>28. Ascendancy Check
Y = field content must be equal to or greater than that of corresponding field in previous record</p> <p>29. Tab Field
Y = set a tab stop</p> <p>30. Added Field
Y = add this field to data records during tape to disk operation</p> <p>31. Conditional Linkage
Y = the digit in the last position of this field automatically selects the next input format to be used</p> <p>32. Enable Edit
Y = Field End Edit executed</p> <p>33. Field Insert/Delete
Y = FLD INS and FLD DEL function keys can be used on field</p> |
|---|--|--|---|

NOTES

FIELD 13 -
NON-DISPLAYABLE,
USED IN FIELD EDIT

Figure 3-13. Record Format Layout SALES1

PROGRAM NAME S A L E S T O T

LIBRARY ASSIGNMENT FIELD EDIT

ORIGINATOR _____

DATE _____

PAGE _____ OF _____

PAGE 1

LINE #	
1	DECLARE TOTAL
2	
3	WHEN FMT 3 GO TO !COMPARE
4	WHEN FMT 2 ADD (2) TO TOTAL
5	RELEASE
6	
7	!COMPARE IF TOTAL = (1) MOVE ϕ TO TOTAL;
8	RELEASE
9	!BACKLOOP BACK
10	WHEN NOT FMT 1 GO TO !BACKLOOP

NOTES

Figure 6-4a. Page 1, Field Edit Program

PAGE 2

LINE #	
1	PAUSE (LOC 5) ORDER NO. (1)
2	'IS NOT IN BALANCE'
3	FLAG (13)
4	SHOW (LOC 5) 40'
5	MV ϕ TO TOTAL
6	RELEASE
7	
8	
9	
10	

NOTES

Figure 6-4b. Page 2, Field Edit Program

PAGE _____

LINE #	
1	DECLARE, CUSNAME, ADDRESS, CITY, STATE, ZIP.
2	WHEN, FIMIT, 3, GO TO, !TOTAL.
3	WHEN, INOT, FIMIT, 2, RELEASE.
4	OUTPUT, (Z), (Z), <DEFER>.
5	MOVE, (5), TO, CUSNAME.
6	MOVE, (6), TO, ADDRESS.
7	MOVE, (7), TO, CITY.
8	MOVE, (8), TO, STATE.
9	MOVE, (9), TO, ZIP.
10	RELEASE.

NOTES

PAGE _____

LINE #	
1	!TOTAL
2	OUTPUT, (1), '---', CUSNAME, ADDRESS,
3	CITY, STATE, ZIP, <SKIP, 80>.
4	RELEASE.
5	
6	
7	
8	
9	
10	

NOTES

Figure 7-5. Tape Output Program

When a Format 3 record (transaction total) is encountered, the program branches to !TOTAL. This routine outputs the following:

- Transaction total (Field 1 from Record Format 3) masked for fixed dollar sign, decimal point insertion and zero suppression;
- Customer name and address (Fields 5 through 9 from Record Format 1) which were stored in variables;
- Spaces to fill out the 80-character record (using the <SKIP> control function).

In summary, this Output program efficiently outputs to tape all Format 1 and Format 3 data needed to create a transaction summary for the sales order application. All Format 2 records (line items) are released, since they are not needed to create the summary record.

PROGRAM NAME SLSBLDN

LIBRARY ASSIGNMENT SOET

ORIGINATOR _____

DATE _____

PAGE _____ OF _____

PAGE _____

LINE #

1	WHEN FMT IS SORT (1) . . .
2	RELEASE . . .
3	
4	
5	
6	
7	
8	
9	
10	

NOTES

Figure I-2. Sort Program SLSBLDN

OPERATIONAL CONSIDERATIONS

The disk on a 600 is divided into two sections:

1. Data Area
2. System Area

The Data Area stores:

1. Work Files (Batches)
2. Master Files (Indexed Files) with all levels of the index except the highest level
3. Single entried value tables
4. Double entry value tables with the primary index level of their index trees.

The System Area stores:

1. All operating system modules
2. Operator statistics and ID tables
3. Job passwords file
4. Highest index level of indexed files & double entry value tables
5. Overflow area for data portion of every record added to an indexed file
6. LIBRARIES (Source & Object Code for Compiled Programs)
 - a. Standard Jobs
 - b. Record (Entry) Formats
 - c. Field End Edit Programs
 - d. File (Batch) End Edit Programs
 - e. Output Programs
 - f. Sort Programs
 - g. Code Conversion (during output) Programs
 - h. Check Digit Programs
 - i. Command Sequence (Autoprompt) Programs

In spite of the large number of items required for the system area of the disk it is limited in size based on the model and disk in use. If 33 or 66MB drives are used, the system area can be up to 8MB. This 8MB System Area is also the 600's virtual memory area. Therefore, expect Nixdorf to tell prospects that any of the previously mentioned items in the System Area are always "Resident in Memory". This type of memory (Disk), however, requires multiple I/O's to access data in the virtual (Disk) memory area (max. 240 bytes handled per I/O). Each I/O takes milliseconds, not micro seconds (as in the 1900/10's real memory). Becoming I/O bound becomes an increasing possibility as more terminals come on-line thereby reducing the area available for each task in real memory. Although more than 240 bytes of main memory will be made available for program storage of each task, as the total area assigned to that task is reduced, the frequency of the 240 byte I/O's can increase dramatically (600 programming manual recommends keeping

each field end edit below 240 bytes to eliminate extra I/O's). This problem is called "Thrashing" and it describes perfectly the excessive number of I/O's that begin to occur as the number of tasks increase.

Virtual memory was developed years ago due to the high cost of main memory. Mainframes often had only 64K of main memory, and swapping pages of code in and out allowed additional multi-tasking, albeit at the expense of throughput.

Mainframe systems offering virtual storage today, however, offer 2-3MB of main memory. Nixdorf's limit of 128K in their virtual system makes no practical sense since main memory prices today are approximately 1/1000 the price per bit of what they were when virtual memory was introduced. The 1900/10 also uses the virtual concept for rolling various, little used, operating system overlays in and out as they're needed. The difference is that we increased main memory (up to 512K) in order to minimize the need to roll non-resident code, in and out, thereby staying with our design concept of maximum throughput and ease of use.

The Nixdorf sales story on disk storage contradicts itself. In one breath they claim disk is far less costly than real memory, so they create virtual (Disk) memory systems. As soon as they design their programming and operational functions, their intent is to do everything to conserve disk to the detriment of operational ease of use and system throughput. They sacrifice in many areas to save disk, even after they said it was inexpensive. The 1900/10 design concepts recognize the low cost of both memory and disk and maximize ease of use to the programmer as well as system throughput.

As was previously mentioned, Nixdorf allows multiple record sizes in a batch in order to save inexpensive disk space, but this isolates data movement from one level to the next, adding to the programming burden (more code, costs more memory & decreases throughput due to increased number of I/O's and additional code execution). This theme of "save disk at any cost" is found in multiple places in the Entrex system, such as in handling sorted data and accessing records in indexed files. Both will be covered later in more detail.

Following is a chart comparing various operational features of the 1900/10 to the 600 System.

OPERATIONAL COMPARISONS

1900/10	FEATURE	600
NEVER	A.F.L (AUTO FILE LOAD) ¹	YES-OFTEN
DEFINITELY NOT	MULTIPLE SUPERVISORS ²	YES
YES	SYSGEN BY USER ³	NOT ALLOWED
YES	SUPERVISOR ON-LINE DIAGNOSTICS ⁴	NO
YES	MAINTENANCE PANEL ON SYSTEM ⁵	NO
YES	RECOVERS WITHOUT VTOC ⁶	VERY DIFFICULT
SYSTEM UTILITY	SUPERVISOR OUTPUT TO DEVICES ⁷	USER PROGRAMS
YES	SYSTEM SAVE ⁸	YES
YES	AUTO SAVE	NO
AT SUPV. COMMAND	COMPILES ⁹	AUTOMATIC
SUPV. UTILITY	SORTING ¹⁰	USER PROGRAMS
NO	UNIVERSAL SYSTEM PASSWORD ¹¹	YES
RECORD END	AUDIBLE SIGNAL ¹²	FIELD END
NO	CHARACTER INSERT/DELETE KEYS ¹³	YES
NO	LOCATION RETURN KEY ¹⁴	YES
NO	HELP KEY ¹⁵	YES
AUTO ON-AUTOMATICALLY	AUTO OFF KEY ¹⁶	AUTO ON-MANUALLY
YES	DUP KEY ¹⁷	YES?

OPERATIONAL COMPARISONS (Continued)

1900/10	FEATURE	600
YES	AUX DUP KEY ¹⁸	NO
FAST	KEYING NEGATIVE NUMBERS ¹⁹	AWKWARD-SLOW
YES	VERIFY CHECKPOINT	YES
YES	VERIFY TO BALANCE	YES
YES	SUPERVISOR MONITOR	YES
NO	DISK FULL - DATA LOSS ²⁰	YES
NO	USER - SYSTEM OPTIMIZATION ²¹	YES
YES	REMOTE KEY STATIONS ²²	YES
YES	CONCURRENT TAPE & COMMUN ²³	NO
YES	CONCURRENT TAPE & COMPILES ²³	LIMITED
YES	TAPE SEARCH ²⁴	LIMITED
BACKGROUND	COMMUNICATIONS ²⁵	FOREGROUND
YES	3270 ²⁶	YES
YES	3770-SDLC ²⁷	NO
YES	360/20 HASP ²⁸	BEING TESTED
DOCUMANGER	DOCUMENT PROCESSING ²⁹	DECOMMITTED W.P. SOFTWARE

FOOTNOTES: OPERATIONAL COMPARISONS

1. A.F.L. (AUTO FILE LOAD) on the 600 is the same thing as an A.B.L. (AUTO BATCH LOAD) on the 480 System. The basic purpose of an A.F.L. is to clean up disk linkages for records, since they dynamically allocate disk space as we do. Increased activity on a system means linkages get complicated even sooner and A.F.L.'s are required more often. The 600 user knows he needs an A.F.L. when system throughput is impacted. Some users perform A.F.L.'s once a week: The time required for an A.F.L. depends greatly on disk space in use, on the number (& size) of double entried tables, and the number & size of master (indexed) files. Users with small amounts of data on disk may only require 1-2 hours per A.F.L. One user (we are told) which rebuilt large, active master (indexed) files, took 48-52 hours to rebuild the system, every 2 weeks. All files written to tape lose the entire index (including indexes to double entried tables). After an A.F.L., the files must be sorted and new keys rebuilt. The 1900/10 saves level "0" of our index tree and rebuild of the tree after a restore from tape is very quick. The 1900/10 doesn't need an A.F.L. to clean up disk linkages, we do that automatically each time the "Power Off" command is used.

To perform an A.F.L., a bootstrap AFL program is loaded and all user data (not operating system modules) is saved to tape. All the record linkages are used and the records are then physically adjacent on tape. This tape is then restored to disk. Records are now adjacent on disk & linkages are clean. New keys for double entried tables & indexed files are now rebuilt.

A.F.L. can be a very big user objection.

2. We have had numerous requests for multiple supervisors. At one time we actually implemented multiple supervisors on the 1900. The user complaints mounted very quickly and we removed that "feature". Users found that the system became unmanageable. One operator deleted a batch before another person was ready to have it deleted. Job contention for printers & tape drives meant that low priority work often ran ahead of high priority work. Users didn't like everyone being able to recompile programs, thereby potentially defeating security measures. The 1900/10 allows a number of non-destructive functions to be executed by multiple operators but protects certain more vital functions by limiting access to them. The 1900/10 doesn't require the supervisor to perform the following functions (but a 600 does):

- a. Communications Receive or Transmit
- b. Build Directories for Spooling or Comm.
- c. Spooling Control
- d. Initiate Autoprompt Batches for Command Mgr. Functions
- e. Assign Input & Output Devices for Spooling, Autolog and Autosave
- f. Enter or Change Format Programs (No Compiling)
- g. Display Operator Stats at Keystation at Batch Close
- h. Make Tables Available to Format Programs (See Note #11 of Programming Comparison)

The 600 requires supervisor for these functions. Therefore, they attempt to sell multiple supervisors as an advantage. Some Nixdorf customers have purchased the 600's extra-cost security package and have written software to keep operators out of the supervisor functions due to the confusion, mistakes and lack of system control that can occur with open access to all functions.

- 3. Our customers can perform their own sysgens if they desire. Addition of 1 terminal means a new sysgen on a 600. They sysgen memory with only the number of terminal control blocks (256 Bytes each) needed for the existing number of terminals. When memory is limited you conserve any way you can, even if the user pays for the new sysgen. A 600 requires a special panel that is brought in for the Sysgen by a C.E. Therefore, Sysgen on the 600 can cost the user a C.E. visit. (A \$150 minimum, we understand)
- 4. The 1900/10 allows the supervisor to check the functional performance of the system via the "TEST" command. Keystations, station printers, magnetic tape units, the line printer and the communication controller can be easily tested without the aid of a C.E.
- 5. The 1900/10's resident maintenance panel allows the supervisor (sometimes after a phone call to our C.E.'s) to take corrective action and continue operations while waiting for the C.E. to arrive. The diagnostic (Sysgen) panel on the 600 that a C.E. brings with him means that when a 600 is down, it stays down until the C.E. arrives. Not installing a diagnostic (Sysgen) panel on each 600:
 - 1. Saves Nixdorf a little money
 - 2. Gives them more control over the user.

Most users don't see these as long term user benefits.

6. If the VTOC ("Directory File" on a 600) is blown, "RECOVERY" on the 1900/10 rebuilds it automatically at restart by simply reading the disk & rebuilding the VTOC. A 600 builds a new VTOC by performing an AFL (described in footnote #1). However, the AFL must reload data from a prior system save. (They can't use the AFL bootstrap program to offload all data, now on the system, because it uses the VTOC). The AFL will give them a new clean operating system, but all data keyed since the last system save is lost & must be rekeyed. As an Alternative they could simply reload an entire system save (which includes operating system modules and a snapshot of memory) which is far faster than an A.F.L. The full system save from 2-3 hours ago may contain, however, the software problem that caused the current failure. The 600 really creates user problems if it loses a V.T.O.C.'
7. Both systems allow the supervisor to output data to any acceptable device. The 1900/10 supervisor uses menu driven utilities to accomplish the task. The 600 requires that the user write output programs to accomplish this basic user task.
8. Nixdorf has attempted to sell their "System Save" as a primary backup technique, for years. (They recommend system saves 3-4 times a day). The system save does a bit for bit read of each track on disk, as well as reading main memory and then puts this system "snapshot" out on tape. It is quite fast (they say 1MB per minute) and Nixdorf S.R.'s try to compare this to our supervisor-save which takes longer. This is an apples & oranges comparison. Our counterpart of the 600's system-save is our software-diagnostic-save which also reads the disk bit by bit, to tape, at high speed. We only recommend this as an occasional save for "Catastrophic Loss" protection. Both our soft.-diag.-save & the Nixdorf system-save require that the entire save be restored to disk, thereby overwriting any resident data. It is an extremely poor way to retrieve one batch.

Many Nixdorf users do retrieve single batches from system-saves. The system-save is offline. All operators close their jobs and stop keying. A system-save is performed to save all current data. Last week's system-save is reloaded (overwriting the entire disk) and the problem batch is corrected and then written or "Rescued" to tape. The current system-save is then reloaded, and only then can operators start keying. Since this process can be lengthy, these users usually fix old batches during the lunch break or after hours. On a 600, programs, check digit libraries, etc. can be saved to tape by a supervisor utility that is concurrent with other operations. This is similar to our supervisor-saves and takes about the same time to perform as ours do, because they are not a bit-for-bit dump. These saves (on both systems) use the VTOC to collect records in sequence

and put them out in retrievable batches or files. Retrieval of single files or batches is concurrent with other functions. The 600 doesn't have a utility to save batches. They ask the user to write a "Rescue" program (it's a universal program for all batches) and use it to save batches and files in a retrievable format. Many users are unaware of it and continue to go through the lengthy process, just described, to retrieve single batches from old tapes. Since Nixdorf sells the system-save as being needed 3-4 times a day and as being far faster than our saves, they often fail to train users to use the "Rescue" save. To now recommend our slower save technique as being more practical in daily use would be a little embarrassing.

Note: Indexed files and double-entried tables saved to tape with rescued programs throw away all levels of the index tree requiring a complete index rebuild when they're restored.

9. The 600 compiles automatically when an enter or change session is terminated. You have no choice. If you want to terminate in the middle of your job, it compiles it unnecessarily. (Compiles affect tape operations and tie up the compiling terminal until completed). The 1900/10 allows entry or change of programs to be put out on the floor as a keying job for an operator but protecting the compile function for supervisor use only. The 600's design means that anyone who is allowed to enter or change programs also has compiling capability.
10. Sorting on the 1900/10 is a straightforward supervisor utility that builds a second batch with the data records resequenced in the desired order. The original batch is deleted, if not needed, to save disk storage. The original and sorted batches are accessible by the existing format programs. On a 600 it's a very different story. Work files (batches) and master (indexed) files are both sorted by the same utility. Data records are NOT resequenced. A key is designated and a sorted index is built. In the case of work files (batches) the new key is deleted and only the sorted record pointers are retained. The only thing you can do with this sorted batch is use an "output" program to write the batch to tape or comm. No other format programs can access this sorted batch. If changes must be made to data in the batch, the pointers are deleted, the batch is changed and the batch is "sorted" again. This is a severely limiting "Sort" technique to many users.
11. Both systems have a user changeable system password. The 600 has another free "Feature", a universal password that allows you to get into any users system. It is five keystrokes "Left Cursor, Right Cursor, Left Cursor, Right Cursor, Release". This "feature" can be programmed out by the user who buys Nixdorf's extra-cost security software package.

12. The 1900/10 sounds an audible signal at record end or as errors occur. The 600 audible signal sounds at each field end as well as for errors. This is annoying to most operators & they turn the beeper off. The 600 doesn't have a visual flash (as on a 1900/10) to then take over signalling duties. If an error occurs (invalid entry, etc.) the 600 operator may key for a while before realizing that the system isn't accepting keystrokes. This wastes time. (See the Auto off problem in footnote #16).
13. Nixdorf sells this "Feature" against the 1900/10's lack thereof. Our lack is intentional. Operator statistics prove time and again that rekeying the entire field is faster and impacts keystroke rates less than using an insert or delete key.
14. The location return key on the 600 is of nominal real value that is given big marketing play. They say that pressing one key returns you quickly to the point of penetration. On the 1900/10, use of the record forward or field forward key with the repeat key also returns you to the point of penetration rapidly.
15. The Help key on a 600 is sold as a "very important" feature since the 1900/10 doesn't have one. Whenever there is an operator error made, the Help key is used to recover and restart the sequence. Depressing the Help key also displays a limited amount of verbiage explaining some options to the operator. In truth, it takes very little time for a 1900/10 operator to remember the responses for most of the error messages. In 600 installations there appears to be an operator dependence on the Help key. Reading messages & then responding isn't as fast as responding from memory. The 600 system also misuses the Help key at times, in operational procedures. In setting up a job to run, both systems require several responses. On a 1900/10 an erroneous response can be corrected immediately and the set up stream continues. On the 600 an erroneous response forces you to use the "Help" key. The "Help" responses only take you back to the beginning of the sequence of questions. Not much "Help" there!
16. The Auto Off key on a 1900/10 only keeps Auto Off for the current field. Auto comes back on as the operator exits the field. On a 600, Auto Off turns it off permanently. It is the operator's responsibility to turn it back on. In many shops where operators have turned the audible signal off (see footnote #12) they aren't notified when the system hangs at record end, since auto is still off and key strokes are simply thrown away. If they notice it several fields (records) later, they turn it back on and rekey the missed data. Before the backspace key can be used, auto off must be depressed. (Even if the operator doesn't want it off) and the operator must then turn it back on!

17. The 600 Dup key has its own peculiarities. If Auto is ON it dups the field from the previous record. If Auto is OFF it only dups one character from the previous field. This must be another 600 "feature".
18. For those users who use the Aux dup key today, they may not enjoy having to write code into a format program to take its place.
19. On a 1900/10 you key the number and then the minus/skip key. On a 600 you:
 1. Key all but the units position of the number
 2. Key the oversign key
 3. Key the units position
 4. Field Release.

It's only 1 extra keystroke, but the sequence is very awkward. The CRT displays an alpha representation (A, G, etc.) of the units position since this really is an oversign technique, just like the good old keypunch. (Nixdorf must not have wanted to leave their heritage totally behind). This odd keying sequence impacts operator performance to such a degree that one Nixdorf user told us that their programs that key credits and debits have a batch end edit program written to insert the oversign for the operator. In this customer's program the operator keys the entire number and then one letter to indicate a debit or credit and then they field release. The batch edit uses this character to create the oversign. This program doesn't save any keystrokes, but the improved keying sequence pays for itself in throughput. Once again Nixdorf lets the user create his own solutions for commonly required features that the 600 doesn't have.

20. The 1900/10 continuously displays the disk full condition on the supervisors screen. The 600 requires the supervisor to request a disk full readout. The 1900/10 sounds an alarm at 95% full. The 600 waits until 98%. and finally it broadcasts the disk condition to all screens. In installations with higher keystation counts, the operators, at times, see the message but try to "just complete their batch" and all of a sudden everything stops because the disk is full. The supervisor must do a warm start and delete some batches before any keying can resume. All operators are idle during this time. The Nixdorf manual warns that data loss may occur at a 100% disk full condition, and this has been found to be true in Nixdorf installations.

21. The 1900/10 priorities for all functions on the system were optimized at the factory. Keystrokes are No. 1 on our priority CUE. The 600 has a utility screen which the user can access to change priorities. They can give higher or lower priority to batch balancing, communications, printing, output, etc. We have yet to interview a Nixdorf user who has noticed any effect on throughput from using this "Feature" of the 600.
22. Both systems offer remote keystations with all the capabilities of a local keystation. The Nixdorf technique uses 1 telephone line per terminal unless a remote terminal concentrator is added (it doesn't substitute for anything) to the network. More detail is available in the remote keystation section of this report.
23. It is our understanding that communications, tape & the compiler in a 600 use the same buffer in main memory (probably due to their limited 128K max. main memory). When comm. is running, tape functions all but cease. Tape is also noticeably impacted by compiles. Job mix on any system including a 1900/10 can affect throughput. However, we use separate buffers for each of these functions and these concurrent functions will not halt or nearly halt the other jobs being performed.
24. The 1900/10 can search a tape and modify data in place on the tape. The 600 can search and display data but can't change it.
25. With communications running as a foreground task, one terminal must be dedicated during communications regardless of the protocol in use. The 600 does have two communications ports but the second port uses only the 3270 protocol.
26. 3270 on both systems allows pass through of records during data entry jobs. The 600 also allows emulation of a 3270 terminal (it can accept host screen formatting commands). That may prove to be an advantage where the prospect truly needs full emulation. Be sure that your prospect actually needs it and not just wants it because Nixdorf told him he "needed" it.
27. We have 3770 SDLC (great for RJE) up and running. At this time, if a customer forces the issue Nixdorf may say it is "planned", but no one is quoting delivery dates, that we know of. Use it while you have it!!
28. 360/20 HASP has been running successfully for some time on the 1900/10. Nixdorf is just now beginning to field test their version.
29. For the user who needs to do a limited amount of document processing on his 1900/10, Documanager is the answer. Nixdorf recently decommitted their word processing package on the 600. They are now marketing a stand-alone word processing system that can be attached to the 600. (It is our information that the "New" word processing systems are reworked 600/30's, the predecessor of the 600/35).

Nixdorf Computer Corporation

600 Series Strengths and Weaknesses

Weaknesses:

- No Command Manager so no background tasks exist
- Very few on-line or off-line diagnostics available for user
- Only 10 format levels
- No autosave function
- No tape labeling/processing functions provided to user
- Very limited status line for keying operators
- Master files limited to 63 with only 4 open at any time
- No mid-level select
- Only 20 value tables available
- Programming occurs from a variety of modules (Max. of 11) and tends to lead to confusion
- Only regional education available with limited amounts allocated per installation
- No keyboard monitor facility
- No error message highlighting
- No repeat key function
- Need to disk re-organize periodically - AFL

This page was taken from a competitive writeup produced by NIXDORF. These are weaknesses Nixdorf lists when comparing the 600 Series to our product.

DATA BASE MANAGEMENT - COMPARISON

Indexed files can be created by keying batches under format program control. A key is designated and a sort of the key takes place. An index tree with multiple levels is created. This process is similar to that on the 1900/10.

Since a 600 operates on a 256 byte sector instead of a 1344 byte A.U., the number of levels in a tree can be greater than ours. For example: File Size = 40,000 Records, Key Length = 15 alpha numeric characters. Using the attached disk sizing charts for Nixdorf 600's, we find that this index tree will have 5 levels. On the 1900/10 the tree would be 3 levels. The 1900/10 carries the top level of each index in main memory. The 1900/10 will require only 2 disk accesses to reach the record. The 600 stores all but the highest index level in the data area of the disk and the highest level in the virtual "Memory" area of disk. To access a record on the 600 means 5 disk accesses since all levels are on disk. In an active environment this big increase in I/O's can have a very, very noticeable, negative effect on system throughput.

The 600 only allows 4 INDEXES to be open simultaneously per program. That can be 1 file with 4 indexes or 4 files, each with 1 index. 600 users can find this very constraining and very messy to program around. The 1900/10 allows 60 FILES (600 indexes) to be open at one time for each program. Our 60 files may be overkill, but Nixdorf's 4 indexes are very definitely underkill.

Any records added to the file affect the system area (operating system/virtual memory area) of the disk. (See the description of system & data areas of disk under "Operational Considerations" for more detail.) The data portion of the added record is always placed in the system area, 1 record per 256 byte sector. (In a 600/55 the max. system area is 8MB.) This means that indexed files must be reorganized to clean out the system area. To reorganize, the entire file is written to tape. All levels of the index tree are automatically stripped by a write to tape. The tape is then read to disk. An index sort and rebuild of the entire tree must be performed. This is a very time consuming task (see footnote #1 of the Operational Comparison). The 1900/10 "SAVES" Level "0" of the tree and after a "RESTORE", rebuild of the upper levels of the tree is fast and simple.

Nixdorf calls their indexed file technique "Data Base Management". In reality, the programmer has the responsibility for managing it. On a 1900/10 we add a record to the file (key & Data) in its proper location with a single "WRITE" command. On a 600 its a much

more complex process that if implemented improperly can destroy a file's integrity in minutes. To add a record takes a series of instructions:

1. "GET" a valid key
2. "INSERT" the new record before or after "the existing record" (insert allocates space in the systems area)
3. "MOVE" Data into the record
4. "INCLUDE" the new key in the index tree.

Following are the disk sizing formulas for indexed files on a 600. In a very competitive situation these may be useful to illustrate differences in disk usage of the two systems.

NIXDORF FILE SIZING

KEY SIZING

KEYS ARE DESIGNATED AS ALPHANUMERIC, ALPHA ONLY OR NUMERIC ONLY AS THE INDEX IS BEING CREATED BY THE SUPERVISOR.

ALPHANUMERIC KEY - $K = \text{KEYLENGTH} + 8$


ALPHA ONLY KEY - $K = \frac{\text{KEYLENGTH}}{2} + 8$

ALL RESULTS THAT ARE NOT EVEN INTEGERS MUST BE ROUNDED UP TO THE NEXT HIGHER EVEN INTEGER

NUMERIC ONLY KEY - $K = \frac{2}{3} \times \text{KEYLENGTH} + 8$

e.g. $K = 10.66$ ROUND UP TO 12

$K = 11.33$ ROUND UP TO 12

$K./S. \left(\begin{array}{c} \text{KEYS} \\ \text{PER} \\ \text{SECTOR} \end{array} \right) = \frac{242}{K}$  RESULTS ARE ROUNDED TO NEXT LOWEST INTEGER

e.g. $\frac{242}{23} = 10.52$ ROUND DOWN TO 10

P.I.L. = PRIMARY INDEX LEVEL

N.K. = NUMBER OF KEYS IN FILE

S.P.L. = NUMBER OF SECTORS IN PRIOR LEVEL

#SECTORS IN P.I.L. = $NK \quad K/S$ e.g. 40,000 10 = 4000

#SECT IN 2ND LEVEL = $SPL \quad K/S$ e.g. 4000 10 = 400

#SECT IN 3RD LEVEL = $SPL \quad K/S$ e.g. 400 10 = 40

#SECT IN 4TH LEVEL = $SPL \quad K/S$ e.g. 40 10 = 4

#SECT IN 5TH LEVEL - $SPL \quad K/S$ e.g. 4 10 = ~~0.4~~ ROUND UP TO 1

(THE CALCULATIONS FOR EACH LEVEL CONTINUE UNTIL THE HIGHEST LEVEL HAS 2 OR LESS SECTORS)

ADD TOGETHER SECTORS FOR EACH LEVEL = 4445

TOTAL SECTORS FOR KEYS X 256 (BYTES/SECTOR) = TOTAL DISK SPACE FOR KEYS

e.g. $4445 \times 256 = 1,137,920$ BYTES OF DISK

NOTE: THE PRIMARY LEVEL IS STORED IN THE "DATA AREA" OF THE DISK. ALL UPPER LEVELS ARE STORED IN THE "SYSTEM AREA" OF THE DISK, FOR A MARGINAL IMPROVEMENT IN ACCESS TIME. CAUTION: NIXDORF WILL SAY THAT ALL UPPER LEVELS ARE CARRIED IN "MEMORY" - THEY INTERPRET SYSTEM AREA ON DISK TO BE PART OF THE VIRTUAL MEMORY. ACCESSES TO THE "SYSTEM AREA" ARE IN MILLISECONDS NOT MICROSECONDS.

NIXDORF FILE SIZING

DATA SIZING

$$D.S.B = \left(\frac{R (LS + F + 6) + 263}{242} \right) 256$$

FIXED ADD ON → 263
 ↙ SECTOR SIZE → 256
 ↖ USABLE AREA IN SECTOR → 242

DSB = DATA SIZE IN BYTES
 R = NUMBER OF RECORDS IN FILE
 LS = BYTES IN THIS LEVEL
 F = FIELDS PER RECORD

e.g. R = 40,000
 LS = 320
 F = 35

$$DSB = \left(\frac{40,000 (320 + 35 + 6) + 263}{242} \right) 256$$

$$DSB = \left(\frac{14,440,263}{242} \right) 256$$

$$DSB = (59670.5) 256$$

YOU CAN'T PARTIALLY FILL SECTORS,
 SO ROUND UP TO NEXT HIGHER INTERGER

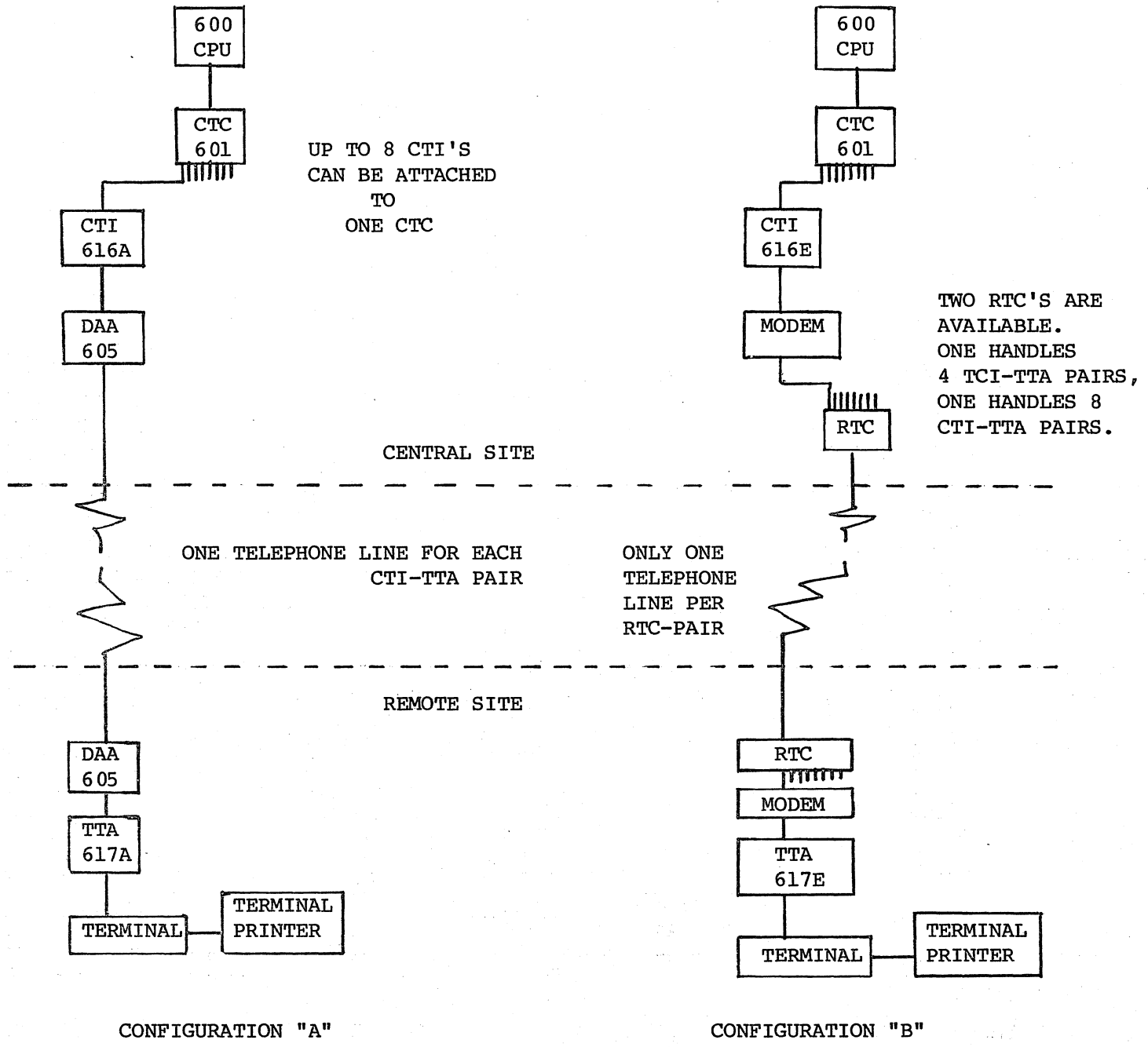
$$DSB = (59671) 256$$

DSB = 15,275,776 BYTES FOR DATA

NOTE: THIS EQUATION IS USED TO CALCULATE DISK SIZING FOR BOTH BATCHES & THE DATA PORTION OF INDEXED FILES. SINCE THE 600'S ACCEPT VARIABLE RECORD LENGTHS WITHIN A BATCH OR FILE THIS CALCULATION MUST BE CARRIED OUT FOR EACH LEVEL, AND THE RESULTS ARE ADDED TOGETHER. THE 263 CHARACTER FIXED ADD ON IS DONE ONLY ONCE FOR THE COMPLETE FILE, NOT FOR EACH LEVEL CALCULATION.

NIXDORF 600'S

REMOTE TERMINAL CAPABILITY



NOTE: IN R.T.C. CONFIGURATIONS EACH CTI OR TTA INTERNAL MODEM MUST BE REPLACED WITH AN EXTERNAL (BELL TYPE MODEM). REMOTE TERMINALS CAN USE ALL TERMINAL PRINTERS INCLUDING THE 300 LPM MODEL.

- CTC = CENTRAL TRANSMISSION CONTROLLER
- CTI = CENTRAL TRANSMISSION INTERFACE & MODEM
- DAA = DATA ACCESS ARRANGEMENT
- TTA = TERMINAL TRANSMISSION ADAPTER & MODEM
- RTC = REMOTE TERMINAL CONCENTRATOR

"600" REMOTE TERMINALS VS. 1900/10 WITH RDA

Example 1- 3 CRT's at the remote site

	<u>600</u>			<u>1900/10</u>	
	<u>PURCHASE</u>	<u>5 YR.W/MAINT.</u>		<u>PURCHASE</u>	<u>5YR.W/MAINT.</u>
1-601 CTC	\$2,400	\$ 57			
3-616A CTI	5,040	129	2-RDA's	\$4,800	\$168
3-617A TTA	5,040	129	2-MODEMS*	--	120
6-605 DAA	<u>450</u>	<u>60</u>			
	\$12,930	\$375		\$4,800	\$288
Plus 3 Telephone Lines					
Plus Installation \$535					

Example 2- 3 CRT's at a very distant location where line costs must be minimized by using a concentrator. The Nixdorf internal modem & DAA must be replaced by an industry modem when using an R.T.C.

	<u>600</u>			<u>1900/10</u>	
	<u>PURCHASE</u>	<u>5 YR. W/MAINT.</u>		<u>PURCHASE</u>	<u>5YR.W/MAINT.</u>
1-601 CTC	\$ 2,400	\$ 57	2-RDA's	\$4,800	\$168
3-616E CTI	3,000	84	2-MODEMS*		120
3-617E TTA	3,000	84			
6-MODEMS*	--	360			
2-624 RTC	<u>7,380</u>	<u>204</u>			
	\$15,780	\$789		\$4,800	\$288
Plus installation \$1095					

* Typical pricing for full duplex -4800 Baud dialup industry modem

EDITOR INSTRUCTION SET

Instruction	Function
ACCEPT	Allows input from keystation.
ADD	Adds numeric data.
AUDIT	Outputs selected data to tape.
BACK	Accesses the previous record in the file, or in an indexed master file.
BYPASS	Advances processing to the next file when multiple files are being processed.
CLEAR	Clears an error flag.
CLOSE	Closes an indexed master file.
DECLARE	Declares variable names to be used in the program.
DEFINE	Defines indexed master file(s) to be accessed from the program.
DELETE	Deletes a record from a master file.
DIVIDE	Divides numeric data.
EXECUTE	Specifies name of the Command Macro that is to be executed.
FLAG	Sets an error flag.
FORWARD	Accesses the next record in the file or in an indexed master file.
GET	Makes available a record in an indexed master file.
GOTO	Transfers processing to a different location in the program.
IF	Compares data and performs embedded instructions depending on the result of the comparison.
INCLUDE	Creates a new key in an index of a master file.
INSERT	Inserts a record into a master file.
LINK	Links to a specified Record Format.
LOAD*	Loads a variable with contents of a batch balance accumulator.
MOVE	Moves data to a field or variable.
MULTIPLY	Multiplies numeric data.
NOTE	Allows comments in a source program.
OPEN	Opens an indexed master file for access.
OUTPUT	Outputs and formats data.
PAUSE	Sounds the error tone and displays a message on the DATA/TERMINAL CRT screen.

*Field Edit Only

EDITOR INSTRUCTION SET (continued)

Instruction	Function
PERFORM	Performs a subroutine.
POSITION*	Returns processing to a specified field or character position within a field in the current data record.
PROTECT	Prevents access of a record in an indexed master file by more than one program.
RELEASE	Advances processing to the next record in the file.
REMOVE	Removes a key from an indexed master file.
SET*	Sets contents of variable into batch balance accumulators for subtotalling.
SHOW	Displays a message on the DATA/TERMINAL CRT screen.
SORT	Sorts data in a file in a user specified manner
STOP	Stops processing.
SUBTRACT	Subtracts numeric data.
TYPE	Writes data to DATA/PRINTER.
WHEN	Tests for the existence of a condition and performs embedded instructions depending on the result.

*Field Edit Only

TABLE B-2. EDITOR LANGUAGE INSTRUCTION SET

ACCEPT [*<LOC line* [, *column*]] *variable* [ELSE *instruction* [; *instruction*]...]. p. 4-32

ADD $\left\{ \begin{array}{l} \textit{literal} \\ \textit{field} \\ \textit{variable} . \\ \textit{expression} \end{array} \right\}$ TO *variable* . p. 4-13

AUDIT $\left\{ \begin{array}{l} \textit{literal} \\ \textit{field} \\ \textit{variable} \\ \textit{<control-function>} \end{array} \right\}$ p. 4-33

BACK [ELSE *instruction* [; *instruction*]...]. p. 4-17

BACK *index-name* [ELSE *instruction* [; *instruction*] . . .] . p. 4-40

BYPASS [ELSE *instruction* [; *instruction*]...]. p. 4-19

CLEAR *field* . p. 4-33

CLOSE *index-name* . p. 4-33

DECLARE *variable* p. 4-11

DEFINE *index-name* p. 4-36

DELETE *index-name* p. 4-42

DIVIDE $\left\{ \begin{array}{l} \textit{literal} \\ \textit{field} \\ \textit{variable} \\ \textit{expression} \end{array} \right\}$ INTO *variable* p. 4-15

FLAG *field* . p. 4-32

FORWARD [ELSE *instruction* [; *instruction*] ...] . p. 4-17

FORWARD *index-name* [ELSE *instruction* [; *instruction*] ...] . p. 4-39

GET *index-name* USING $\left\{ \begin{array}{l} \textit{alpha-literal} \\ \textit{field} \\ \textit{variable} \end{array} \right\}$ ELSE *instruction* . p. 4-38

GET FIRST *index-name* ELSE *instruction* [; *instruction* ...] . p. 4-38

GET LAST *index-name* ELSE *instruction* [; *instruction* ...] . p. 4-38

GET NEXT *index-name* ELSE *instruction* [; *instruction* ...] . p. 4-38

GET PRIOR *index-name* ELSE *instruction* [; *instruction* ...] . p. 4-38

GET CURRENT *index-name* ELSE *instruction* [; *instruction* ...] . p. 4-38

GOTO *label* . p. 4-26

IF $\left\{ \begin{array}{l} \textit{literal} \\ \textit{field} \\ \textit{variable} \\ \textit{expression} \end{array} \right\} \left\{ \begin{array}{l} = \\ \neq \\ > \\ < \end{array} \right\} \left\{ \begin{array}{l} \textit{literal} \\ \textit{field} \\ \textit{variable} \\ \textit{expression} \end{array} \right\}$ *instruction* [; *instruction*] p. 4-20

IF *comparison* OR IF *comparison* [OR IF *comparison*] ... *instruction* [; *instruction*] p. 4-22

IF *comparison* IF *comparison* [IF *comparison*] ... *instruction* [; *instruction*] p. 4-22

INCLUDE *record-pointer* IN *index-name* USING $\left\{ \begin{array}{l} \textit{literal} \\ \textit{field} \\ \textit{variable} \end{array} \right\}$. p. 4-42

INCLUDE IN *index-name* USING $\left\{ \begin{array}{l} \textit{alpha-literal} \\ \textit{field} \\ \textit{variable} \end{array} \right\}$. p.4-42

INSERT $\left\{ \begin{array}{l} \textit{num literal} \\ \textit{variable} \end{array} \right\}$ $\left\{ \begin{array}{l} \textit{Before} \\ \textit{After} \end{array} \right\}$ *index-name* . p. 4-41

LINK $\left\{ \begin{array}{l} \textit{literal} \\ \textit{field} \\ \textit{variable} \end{array} \right\}$. p. 4-20

LOAD *variable* . p. 4-35

MOVE $\left\{ \begin{array}{l} \textit{literal} \\ \textit{field} \\ \textit{variable} \\ \textit{expression} \end{array} \right\}$ TO $\left\{ \begin{array}{l} \textit{field} \\ \textit{variable} \end{array} \right\}$. p. 4-12

MULTIPLY $\left\{ \begin{array}{l} \textit{literal} \\ \textit{field} \\ \textit{variable} \\ \textit{expression} \end{array} \right\}$ TIMES *variable* . p. 4-14

NOTE *comment* . p. 4-36

OPEN *index-name* $\left[\left[\left\{ \begin{array}{l} \textit{UPD} \\ \textit{INC} \end{array} \right\} \right] \right]$. p. 4-37

OUTPUT $\left\{ \begin{array}{l} \textit{literal} \\ \textit{field}[\textit{modifier}] \\ \textit{variable}[\textit{modifier}] \\ \textit{<control-function>} \end{array} \right\}$ $\left[\left[\left\{ \begin{array}{l} \textit{literal} \\ \textit{field}[\textit{modifier}] \\ \textit{variable}[\textit{modifier}] \\ \textit{<control-function>} \end{array} \right\} \right] \right]$

Control Functions

Modifiers

<ALL>	<BLK <i>n</i> >	<JOB>	PK
<ALL <i>n</i> >	<BSP <i>n</i> >	<LABEL>	LS
<ALL <i>n - n</i> >	<COUNT <i>nnnn</i> >	<LF>	LZ
<@ALL>	<DATE <i>x</i> >	<PAGE <i>nn</i> >	SG
<@ALL <i>n</i> >	<DEFER>	<REC <i>n</i> >	TS
<@ALL <i>n - n</i> >	<EOF>	<RWND>	TZ
<APPEND>	<FMT>	<SKIP <i>nn</i> >	'mask'
<BATCH>	<HEX <i>nn</i> >	<TOP>	

PAUSE [<LOC { *literal* / *variable* } [, { *literal* / *variable* }] > { *literal* / *field* [|*modifier*] / *variable* [|*modifier*] / <*control-function*> } p. 4-31

PERFORM *label* . p. 4-28

POSTION *field* . p. 4-16

PROTECT *field* [ELSE *instruction* [; *instruction*] . . .] . p. 4-43

RELEASE [ELSE *instruction* [; *instruction*] . . .] . p. 4-15

REMOVE *index-name* ELSE *instruction* [; *instruction*] p. 4-43

REMOVE *index-name* USING { *alpha-literal* / *field* / *variable* } ELSE *instruction* [; *instruction*] p. 4-43

SET *variable* . p. 4-34

SHOW [<LOC { *literal* / *variable* } [, { *literal* / *variable* }] > { *literal* / *field* [|*modifier*] / *variable* [|*modifier*] / <*control-function*> } p. 4-31

SORT { literal
field [|modifier]
variable [|modifier]
<control-function> } [{ literal
field [|modifier]
variable [|modifier]
<control-function> }]

Control Functions

Modifiers

Same as for OUTPUT

|AK
|DK

SUBTRACT { literal
field
variable
expression } FROM variable . p. 4-14

TYPE [LOC terminal] { literal
field
variable
control function } p. 4-34

WHEN condition instruction [; instruction] p. 4-23

Conditions

BATCH	[NOT] FMT
[NOT] ENTRY	[NOT] @FMT
EOT	OVERFLOW
[NOT] EXAMINE	START
[NOT] FIELD	[NOT] UPDATE
FLAG	[NOT] VALIDATE
@FLAG	[NOT] VERIFY

NOTE

AT END can be used in place of ELSE.
PGM can be used in place of FMT.