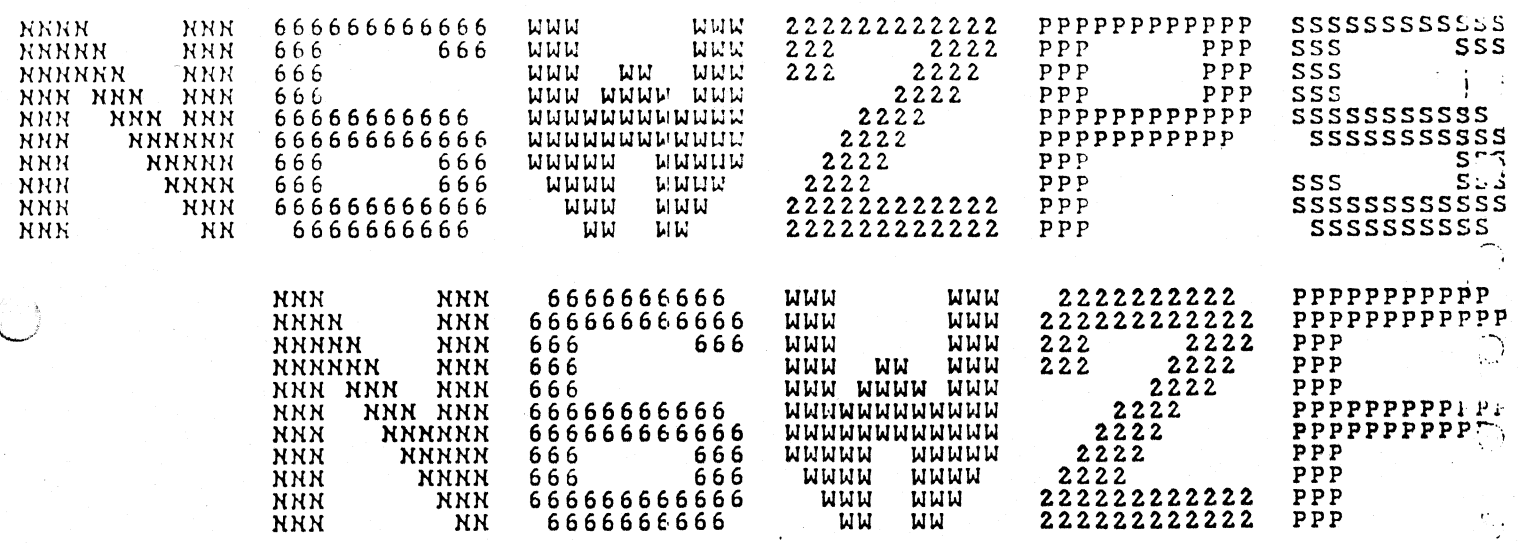
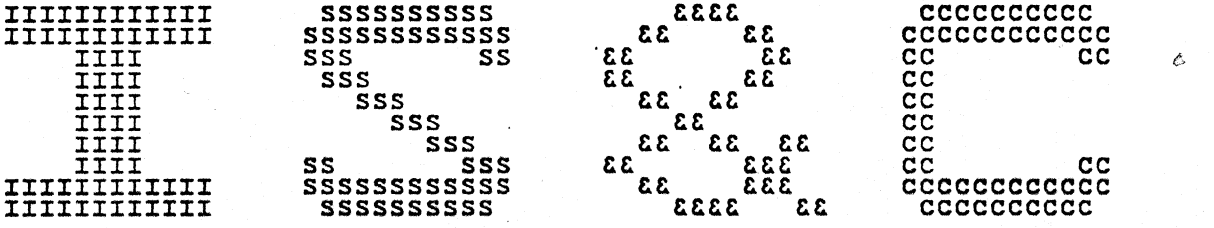
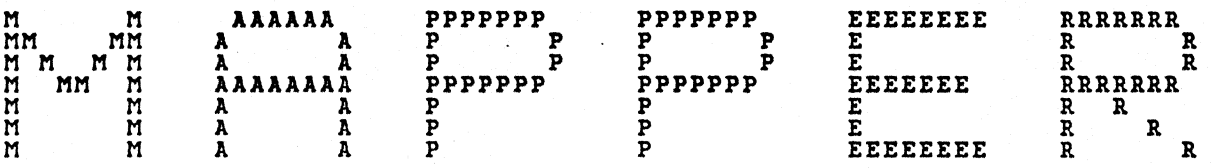


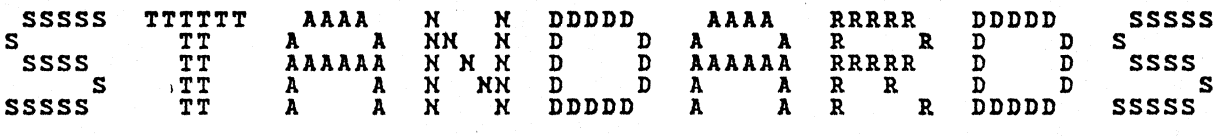
* * * * * * UNIVAC 0777 PRINTING SYSTEM VERSION 1R4.0-P4 * * * * *

RUNID * N6W2PS USER ID * N6W2PS PART NUMBER * 00 INPUT DEVICE

DATE CREATED * 12/06/83 DATE PRINTED * 01/25/84







PREPARED BY:

PETER SCHAMBIL
I. S. & C. INTERNATIONAL DIVISION
EUROPEAN SYSTEM SUPPORT GROUP

NOVEMBER 1983

Table of contents

Table of contents

1. INTRODUCTION

1.1. SCOPE

THE PURPOSE OF THIS DOCUMENT IS TO ESTABLISH STANDARDS FOR THE USE OF MAPPER IN THE DEVELOPMENT OF MAJOR BUSINESS APPLICATIONS. THROUGHOUT THE DOCUMENT A STRONG EMPHASIS IS PLACED ON THE ASPECT OF DEVELOPING COMMON DATA PROCESSING SYSTEMS FOR USE BY VARIOUS ORGANIZATIONS. THIS DOES NOT MEAN THAT THESE STANDARDS DO NOT APPLY TO SYSTEMS DEVELOPED FOR USE IN ONE LOCATION ONLY.

THESE STANDARDS APPLY TO ALL MAPPER BASED DATA PROCESSING APPLICATIONS DEVELOPED BY INTERNATIONAL DIVISION I. S. & C.

THE SECTION ON SYSTEM DEVELOPMENT STANDARDS IS ADDRESSED MAINLY TO THE PROJECT LEADER AS THE PERSON RESPONSIBLE FOR CREATING THE ENVIRONMENT IN WHICH THE SYSTEM DEVELOPMENT WILL TAKE PLACE AND FOR ESTABLISHING THE GUIDELINES WHICH APPLY TO THE SPECIFIC SYSTEM.

PROGRAMMING STANDARDS AND RUN EFFICIENCY GUIDELINES WERE PREPARED MAINLY FOR THE BENEFIT OF THE RUN DESIGNER

SECTION 5 COVERS THE ASPECT OF SEPARATING TEST FROM PRODUCTION ENVIRONMENT, AND CONTROLLING CHANGES AFTER SYSTEM IMPLEMENTATION. IT DESCRIBES A SIMPLE BUT EFFECTIVE METHODOLOGY OF ORGANIZING A SYSTEM SUPPORT ENVIRONMENT FOR MAPPER BASED SYSTEMS.

SECTION 6 DESCRIBES THE MINIMUM REQUIREMENTS OF DOCUMENTATION FOR MAPPER BASED SYSTEMS. THESE DOCUMENTATION STANDARDS ARE LOOSELY BASED ON THE UTILITIES AVAILABLE AS PART OF THE SYSTEM DEVELOPMENT AND ARCHITECTURE (SDA) PACKAGE.

SECTION 7 GIVES A BRIEF DESCRIPTION OF THE VARIOUS UTILITIES MENTIONED THROUGHOUT THIS DOCUMENT. FURTHER DETAILS OF THESE UTILITIES ARE

AVAILABLE ON REQUEST FROM I. S. & C. ID.

1.2. ACKNOWLEDGEMENTS

MANY OF THE TOPICS IN THIS DOCUMENT HAVE BEEN COVERED BEFORE IN OTHER PLACES SUCH AS THE MAPPER 1100 USER GUIDE, THE MAPPER 'HELP' DOCUMENT, THE SYSTEM DEVELOPMENT AND ARCHITECTURE MANUAL AND OTHER DOCUMENTS.

THIS DOCUMENT REPRESENTS AN ATTEMPT TO COMBINE ALL ASPECTS OF SYSTEM DEVELOPMENT UNDER MAPPER IN ONE PLACE.

THE CODING EXAMPLES USED THROUGHOUT THIS DOCUMENT HAVE ALSO BEEN TAKEN FROM A VARIETY OF SYSTEMS.

COMMENTS ARE INVITED AND SHOULD BE SENT TO THE AUTHOR AT

SPERRY COMPUTER SYSTEMS
INTERNATIONAL DIVISION
SPERRY CENTRE
STONEBRIDGE PARK
LONDON NW10 8LS

2. SYSTEM DEVELOPMENT STANDARDS

2.1. SYSTEMS DESIGN

THE MAJORITY OF MAPPER RUNS TO BE DEVELOPED DO NOT STAND ALONE BUT COMBINE WITH OTHER MAPPER RUNS OR COBOL PROGRAMS TO FORM A SYSTEM OR SUBSYSTEM. RUN-DESIGN SHOULD THEREFORE ALWAYS BE DONE IN THE SYSTEM CONTEXT, BEARING IN MIND SUCH OBJECTIVES AS UNIFORMITY BOTH FOR THE BENEFIT OF THE USER AND THE PROGRAMMER.

DURING THE PROCESS OF SYSTEMS DESIGN THE ANALYST CREATES THE ENVIRONMENT FOR ALL FURTHER STEPS IN THE SYSTEM DEVELOPMENT ACTIVITY. HE/SHE WILL LAY DOWN THE GUIDELINES WHICH DETERMINE THE METHODS USED TO ACCOMPLISH THE GOALS ESTABLISHED AT THE OUTSET OF THE PROJECT.

IN THE AIM OF DEVELOPING APPLICATIONS WHICH SATISFY THE REQUIREMENTS FOR TRANSPORTABILITY, MAINTAINABILITY, FLEXIBILITY AND SECURITY, THE SYSTEMS DESIGNER WILL SELECT THE METHODS TO BE USED BY THE PROGRAMMERS TO ATTAIN THESE GOALS.

THIS DOCUMENT DOES NOT PRESENT ONE SIMPLE SET OF RULES FOR SYSTEM DEVELOPMENT, BUT AIMS TO PROVIDE THE BACKGROUND KNOWLEDGE ON WHICH THE DECISIONS FOR THE EFFECTIVE AND EFFICIENT USE OF MAPPER CAN BE BASED. IT HIGHLIGHTS A NUMBER OF CRITICAL ISSUES OF SYSTEM DEVELOPMENT USING MAPPER AND SUGGESTS VARIOUS SOLUTIONS FROM WHICH THE SYSTEMS DESIGNER HAS TO CHOOSE THOSE MOST SUITABLE TO THE APPLICATION IN QUESTION.

THE PROJECT LEADER SHOULD ISSUE A SET OF GUIDELINES ON HOW THESE STANDARDS ARE TO BE APPLIED TO THE PARTICULAR SYSTEM. IT MAY BE USEFUL TO PREPARE AND DISTRIBUTE A SAMPLE RUN DEMONSTRATING THE VARIOUS TECHNIQUES TO BE EMPLOYED.

SYSTEM DEVELOPMENT STANDARDS

2.1.1. THE USE OF COPY LIBRARIES

VARIABLE DEFINITIONS OR STANDARD ROUTINES THAT ARE TO BE USED THROUGHOUT A SYSTEM SHOULD BE ESTABLISHED AS ELEMENTS IN A COPY LIBRARY. THE CONCEPT OF COPY LIBRARIES IS NOT A FEATURE OF MAPPER AS A PROGRAMMING LANGUAGE BUT HAS BEEN CREATED IN THE FORM OF SIMPLE UTILITIES WHICH ALLOW THE INCLUSION OF COPY ELEMENTS INTO RUNS.

OBVIOUS AREAS FOR USING COPY ELEMENTS ARE DEFINITION OF STANDARD VARIABLES USED THROUGHOUT THE SYSTEM, STANDARDIZED SECURITY CHECKS AND ERROR PROCEDURES. APART FROM THE OBVIOUS WORK-SAVING BENEFITS, COPY LIBRARIES CAN BE USED TO ACHIEVE THE GOALS OF TRANSPORTABILITY, FLEXIBILITY AND LANGUAGE-INDEPENDENCE. THE SUBJECT OF COPY ELEMENTS WILL RECUR THROUGHOUT THIS DOCUMENT AND EXAMPLES OF COPY PROCEDURES WILL BE SHOWN UNDER THE VARIOUS HEADINGS.

THE COPY LIBRARY MUST BE ESTABLISHED BEFORE RUN DESIGN STARTS. THE RUN DESIGNERS MUST BE INFORMED ABOUT THE COPY ELEMENTS AVAILABLE AND THEIR PURPOSE. THIS IS PARTICULARLY IMPORTANT AS A NUMBER OF VARIABLES WILL BE RESERVED FOR USE IN THESE COPY ELEMENTS AND MUST NOT BE USED FOR ANY OTHER PURPOSE.

2.1.2. DESIGN SYSTEMS FOR TRANSPORTABILITY

DURING THE INITIAL PHASE OF ANY MAPPER BASED DATA PROCESSING APPLICATION THE MODE(S) AND TYPE(S) WHERE THE DATA AND THE MAPPER RUNS WILL BE LOCATED ARE ASSIGNED. MOST MAPPER RUN STATEMENTS REQUIRE THE SPECIFICATION OF THE DATA MODE AND TYPE. OFTEN THIS INFORMATION WAS HARD-CODED INTO EACH STATEMENT. WHEN THE APPLICATION WAS TRANSPORTED TO ANOTHER INSTALLATION WHERE THE SAME MAPPER MODE WAS NOT AVAILABLE, IT WAS NECESSARY TO CHANGE EACH STATEMENT. IN ORDER TO MAKE SYSTEMS MORE TRANSPORTABLE THE RULE WAS ESTABLISHED THAT VARIABLES MUST BE USED FOR ALL MODES AND TYPES. THESE VARIABLES WERE SET TO THE REQUIRED

VALUES IN THE INITIALIZATION SECTION OF EACH RUN. WHEN THE RUN IS TRANSPORTED TO ANOTHER LOCATION, ONLY THE INITIAL VALUES OF THESE VARIABLES NEED TO BE CHANGED. ALTHOUGH THIS METHOD REPRESENTS AN IMPROVEMENT, IT STILL REQUIRES A SIGNIFICANT AMOUNT OF WORK TO TRANSPORT A LARGE SCALE SYSTEM. THE NEXT STEP WAS TO INTRODUCE THE CONCEPT OF A COPY LIBRARY. THE VARIABLE DEFINITION WAS BUILT AS A COPY ELEMENT. EACH LOCATION MAINTAINS AN ALTERNATE COPY LIBRARY CONTAINING THE MODES AND TYPES USED AT THAT LOCATION. UTILITIES WERE DEVELOPED WHICH MAKE IT POSSIBLE TO SWAP THESE COPY ELEMENTS ON ONE PARTICULAR RUN OR ON AN ENTIRE SYSTEM.

AN EXAMPLE OF THIS METHOD IS SHOWN BELOW.

```
 00010      12   %%%%%%%%%%   MODE-TYPE-RID DEFINITIONS   %%%%%%%%%%%%%%%%%%
.====== 80 - 99 = MODE - TYPE - RID =.================================
:V80I4 46     = RUN MODE              ;V81I4 600   = RUN TYPE
:V82I4 12     = DATA MODE             ;V83I4 156   = DATA TYPE H (CUST)
:V84I4 154    = DATE TYPE G (COURSE)  ;V85I4 146   = DATA TYPE D (CLASS)
:V86I4 150    = DATA TYPE E (STUDENT) ;V87I4 152   = DATA TYPE F (HIST)
:V88I4        = OPEN                  ;V89I4 160   = DATA TYPE I (TABLES)
:V90I4 17     = MKTG BRANCH TABLE     ;V91I4 13    = RID NR ERROR MESS
:V92I4 14     = RID NR ERROR EXPL     ;V93I4 10    = RID NR SECURITY FILE
:V94I4 11     = RID NR CONSTANT FILE  ;V95I4 1?    = RID NR LOG FILE
:V96I4 16     = RID NR INSTRUCTOR TAB ;V97I4 23    = RID NR COST CENTRE TAB
:V98I4 9      = LOCK RID CUST FILE    ;V99I4 25    = RID NR HOTEL ADDRESSES
.  END OF COPY ELEMENT 00010   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

USING COPY ELEMENTS AND THE ASSOCIATED UTILITIES MAKES IT POSSIBLE TO GET A TRANSPORTED SYSTEM READY FOR USE WITHIN MINUTES.
ONE MINOR DISADVANTAGE REMAINS: EVERY TIME A NEW VERSION OF A CENTRALLY MAINTAINED RUN IS RELEASED, THE PROCESS OF SWAPPING THE COPY ELEMENT HAS TO BE REPEATED. THIS NECESSITY WAS REMOVED BY A NEW

SYSTEM DEVELOPMENT STANDARDS

METHOD: MODES AND TYPES ARE STORED IN A TABLE WHICH THE RUN ACCESSES. THE ONLY REQUIREMENT IS THAT THIS TABLE IS KEPT IN A PRE-DETERMINED RID IN THE RUN TYPE. NEW VERSIONS OF CENTRALLY MAINTAINED RUNS CAN BE PUT INTO PRODUCTION WITHOUT ANY CHANGE TO THE RUN. THIS METHOD ALSO MAKES IT POSSIBLE TO USE THE SAME RUNS FOR SEVERAL ORGANIZATIONS BUT TO MAINTAIN DIFFERENT SETS OF DATA FILES.

THE CODE WHICH HAS TO BE INCLUDED IN THE BEGINNING OF EACH RUN IS SHOWN BELOW:

```
 00006/V01 012 LINES ===> DEPN$ TBL LOOKUP CORE MODE&TYPES <==============
∂IF V141 = 0 CHG V141 EMODE$ IF V151 = 0 CHG V151 ETYPE$ .
∂IF V142 = 0 GTO LIN +1 ; GTO LIN +11 .                  **LINE +11 USED
∂CHG V185 DEPN$ CHG V38 100 .          GET-DEPT-NMBR-AND-SET-RID-FOR-TAB
∂FND,V141,V151,V38,6 ' ' 2-4,12-4  ,V185,'CORE'  ,V39 .
∂IF V39 = 0 GTO LIN +1 ; GTO LIN +5 .
∂BRK,V141,V151 .
DEPARTMENT NOT SET UP IN DEPN$ TABLE FOR USE OF THIS PROGRAM-NOTIFY IS&C
∂BRK,V141,V151 DSP,V141,V151,-0 .
∂GTO END .
∂RDL,V141,V151,V38,V39 7-4,17-4,27-4,32-4,37-4,42-4,47-4,52-4,57-4,62-4 \
V37,V142,V143,V144,V145,V146,V147,V148,V149,V150 .  LOAD-SCOPE-CORE-MD&TY
∂ . %% END %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

THE TABLE USED IN THIS EXAMPLE LOOKS LIKE THIS:

```
*DEPT.DUMP.PROG.     .
*NMBR.UNIT.TYPE.MODE.TYPE <C O R E   T Y P E S   B   T H R U   I>
*====.====.====.====.====.====.====.====.====.====.====.====.====.====.
   28  485 CORE 0440       6702 6704 6706 6710 6712 6714 6716 6720
```

FROM THE NEED FOR TRANSPORTABILITY RESULT THE FOLLOWING REQUIREMENTS:

- VARIABLES MUST BE USED FOR ALL MODES AND TYPES
- THESE VARIABLES MUST BE DEFINED IN A SEPARATE COPY ELEMENT
- A TABLE LOOK-UP METHOD MUST BE USED TO LOAD THE RELEVANT VALUES AT EACH LOCATION.
- THE NECESSARY COPY ELEMENTS MUST BE CREATED BEFORE RUN DESIGN STARTS AND MUST BE INCLUDED INTO EVERY RUN.

ANOTHER ASPECT OF TRANSPORTABILITY IS THE CHOICE OF THE CHARACTER SET. PRACTICE HAS SHOWN THAT THERE ARE NO PROBLEMS WITH TRANSPORTING RUNS FROM AN ASCII (FCS OR FCSU) TYPE INTO A FIELDATA (LCS) TYPE, BUT SERIOUS PROBLEMS HAVE BEEN ENCOUNTERED WHEN ATTEMPTING TO MOVE FROM LCS TO FCS.

ALL COMMON SYSTEMS MUST THEREFORE BE DEVELOPED IN FCSU TYPES. THE RUNS MUST BE CODED TO FUNCTION CORRECTLY IN EITHER CHARACTER SET. THIS IS PARTICULARLY IMPORTANT IN RANGE SEARCHES AND SEARCHES FOR BLANK OR NON-BLANK LINES.

2.1.3. FLEXIBILITY

AMONGST THE PROBLEMS IN DEVELOPING AND MAINTAINING COMMON SYSTEMS WHICH ARE USED IN VARIOUS LOCATIONS ARE THE DIFFERENCES IN PROCESSING REQUIREMENTS WHICH, ALTHOUGH MINOR, WOULD IMPACT THE COMMON PART OF THE SYSTEM. LOCAL MODIFICATIONS WOULD MAKE THE CENTRAL MAINTENANCE OF THESE RUNS IMPOSSIBLE AND WOULD DEFEAT THE OBJECTIVE OF COMMON SYSTEMS.

THE DEVELOPMENT OF COMMON SYSTEMS MUST THEREFORE PROVIDE SUFFICIENT FLEXIBILITY TO ALLOW FOR LOCAL DIFFERENCES WHILE STILL MAINTAINING ONE SET OF COMMON PROGRAMS.

ALL COMMON SYSTEMS MUST THEREFORE INCLUDE AN OPTION AND CONSTANTS

TABLE. THIS TABLE CAN BE USED TO STORE CONSTANT VALUES USED IN A SYSTEM, SUCH AS TAX RATE, LOCATION CODE ETC. IT CAN ALSO BE USED TO STORE PROCESSING OPTIONS. THE VALUES IN THIS TABLE ARE SET UP AT IMPLEMENTATION TIME ACCORDING TO THE PARTICULAR LOCATION'S REQUIREMENTS. AT EXECUTION TIME, EACH RUN WILL ACCESS THIS TABLE, OBTAIN THE CONSTANTS AND OPTIONS REQUIRED AND PERFORM THE PROCESSING INDICATED IN THE TABLE.

2.1.4. LANGUAGE INDEPENDENT SYSTEMS

LANGUAGE INDEPENDENCE IS JUST ANOTHER ASPECT OF FLEXIBILITY. COMMON SYSTEMS MUST INCLUDE THE CAPABILITY TO TRANSLATE SCREENS AND OTHER TEXTS INTO THE LOCAL LANGUAGE WITHOUT INTERFERING WITH THE CONCEPT OF COMMON SYSTEMS. UNLESS THIS ASPECT HAS BEEN CONSIDERED DURING THE SYSTEM DESIGN, TRANSLATION OF SCREENS WHICH ARE EMBEDDED IN THE RUNS, MEANS MODIFICATION OF THESE RUNS. WHEN A NEW VERSION OF THE RUN IS RELEASED, THIS WORK HAS TO BE REPEATED.

LANGUAGE-INDEPENDENCE CAN BE ACHIEVED BY PLACING ALL SCREENS AND TEXTS INTO A COPY LIBRARY. THE SUBSIDIARY CREATES A LOCAL LANGUAGE VERSION OF THESE COPY ELEMENTS AND USES THE AVAILABLE UTILITIES TO SWAP COPY ELEMENTS. NEW RELEASES OF RUNS CAN BE CONVERTED TO LOCAL LANGUAGE BY USING THE SAME PROCESS.

LIKEWISE, ERROR MESSAGES MUST NOT BE EMBEDDED IN THE RUNS BUT STORED IN AN ERROR TABLE. THIS ALLOWS SUBSIDIARIES TO TRANSLATE THE ERROR TEXTS INTO THE LOCAL LANGUAGE WITHOUT CHANGING THE RUNS.

2.1.5. SECURITY

THERE ARE TWO ASPECTS TO THE GENERAL SUBJECT OF SECURITY:

A) TO PREVENT THE USE OF A RUN BY UNAUTHORIZED USERS.

B) TO PREVENT MANUAL ACCESS/UPDATES TO DATA BY USERS BYPASSING THE RUN-CONTROLLED SECURITY AND AUDIT FEATURES.

THE SYSTEM DESIGNER HAS TO DECIDE WHICH KIND OF SECURITY CHECK IS REQUIRED FOR EACH PARTICULAR SYSTEM. A STANDARDIZED SECURITY CHECK ROUTINE SHOULD BE PREPARED AND INCLUDED INTO EACH RUN. THE ROUTINE SHOWN BELOW USES A TABLE CONTAINING THE USER-ID'S OF ALL AUTHORIZED USERS. THE SECURITY ROUTINE WILL THEREFORE PREVENT UNAUTHORIZED USE OF THE SYSTEM. EACH USER-ID HAS A SECURITY LEVEL ASSIGNED. THIS MAKES IT POSSIBLE TO DEFINE FOR EACH USER WHICH FUNCTIONS HE/SHE MAY USE.

```
 00105     04   %%%%%%%%%   SECURITY CHECK ROUTINE        %%%%%%%%%%%%%%%%%
 @CHG V165 USER$ FND,V82,V89,V93,,187 ' ' 2-12  ,V165 ,V190 .      GET USER
 @RDL,V82,V89,V93,V190 15-2 V195 .                    . FIND SECURITY LEVEL
 @IF V195 LT V194 CHG V197 008 CHG V198 169 GTO 188 . LEVEL NOT HIGH ENOUGH
 .  END OF COPY ELEMENT 00105  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

UNAUTHORIZED ACCESS TO THE DATA CAN BE PREVENTED BY RESTRICTING ACCESS TO THE MODE IN WHICH THE DATA IS LOCATED.

PASSWORDS SHOULD BE PLACED ON RIDS, WHEN UNCONTROLLED UPDATING IS NOT PERMITTED. THE PASSWORDS SHOULD BE BASED ON AN ALGORITHM AND A NEW PASSWORD SHOULD BE WRITTEN, EACH TIME A RID IS UPDATED.

2.1.6. RECOVERY

MAPPER RECOVERY PROCEDURES ARE MAINLY GEARED TO MANUAL UPDATING. ALL UPDATES WITH THE POSSIBLE EXCEPTION OF THE LAST TRANSMIT ARE AVAILABLE AFTER RECOVERY. THE PRESENT UPDATE STATUS CAN BE ESTABLISHED BY CHECKING THE RID IN QUESTION. THIS PROCEDURE IS ADEQUATE FOR UPDATING UNDER RUN CONTROL, AS LONG AS THE RUN UPDATES ONLY ONE RID. HOWEVER, WHEN A RUN UPDATES SEVERAL RIDS IT IS POSSIBLE THAT ONE RID MAY HAVE BEEN UPDATED BUT THE CORRESPONDING UPDATE TO ANOTHER RID HAS NOT YET

BEEN APPLIED. THIS POSES A POTENTIAL THREAT TO THE INTEGRITY OF THE DATA BASE, WHENEVER A RUN DOES NOT COME TO A NORMAL TERMINATION.

THE PRESENT RELEASE (LEVEL 30) OF MAPPER DOES NOT PROVIDE A RUN-UNIT ROLL BACK FEATURE - A LIMITED VERSION IS EXPECTED WITH LEVEL 31.

IT IS THEREFORE UP TO THE SYSTEMS DESIGNER TO DETERMINE WHAT KIND OF RECOVERY IS NECESSARY AND HOW IT CAN BE ACCOMPLISHED.

RECOVERY IN THIS CONTEXT CONSISTS OF TWO ASPECTS:

A) DETECT THAT THERE IS A POTENTIAL DANGER OF DATABASE CORRUPTION AND PREVENT FURTHER UPDATES.

B) RECOVER THE SITUATION PRIOR TO THE LAST RUN.

THE FIRST OBJECTIVE CAN BE ACHIEVED BY USE OF SO-CALLED 'SOFT LOCKS': EACH RUN WILL PLACE A LOCK (USUALLY THE RUN NAME) IN A PRE-DETERMINED PLACE IN THE RID(S) TO BE UPDATED. THIS LOCK IS REMOVED AFTER ALL UPDATES HAVE BEEN COMPLETED SUCCESSFULLY. EACH RUN, PRIOR TO UPDATING A RID, WILL CHECK THAT RID FOR THE PRESENCE OF A LOCK, WAIT FOR A PRE-DETERMINED PERIOD OF TIME AND, IF THE LOCK IS STILL PRESENT, TERMINATE WITH AN APPROPRIATE MESSAGE TO THE USER.

THE PRESENCE OF A LOCK ON A RID INDICATES THE POTENTIAL CORRUPTION OF THE DATA BASE. THIS IS WHERE THE RECOVERY PROCEDURE COMES INTO THE PICTURE.

IN ORDER THAT THE RECOVERY RUN CAN PERFORM A 'RUN UNIT ROLL BACK', EACH RUN MUST PERFORM A PRE-DEFINED BACK-UP PROCEDURE BEFORE APPLYING THE FIRST UPDATE. ONE WAY OF ACHIEVING THIS WOULD BE TO SAVE A COPY OF THE RID TO BE UPDATED (BEFORE-LOOKS). IF THIS METHOD CREATES AN UNACCEPTABLY LARGE OVERHEAD, ALTERNATIVE METHODS SHOULD BE DEVELOPED, SUCH AS SAVING THE LINE NUMBER OF THE LAST SUCCESSFUL UPDATE, ETC.

THE RECOVERY PROCEDURE SHOULD BE CONSTRUCTED IN SUCH A WAY THAT NO

MANUAL UPDATE OF THE DATABASE IS REQUIRED. THIS CAN BE ACHIEVED BY A RUN WHICH, BASED ON THE RUN NAME AND OTHER INFORMATION ABOUT THE UNSUCCESSFUL TRANSACTION RE-INSTATES THE SITUATION PRIOR TO THE ABORTIVE RUN.

2.1.7. RUN LOGGING

IT MAY BE BENEFICIAL TO ESTABLISH SYSTEM MONITORING TOOLS ALREADY DURING THE SYSTEM DEVELOPMENT PROCESS. ONE SUCH TOOL CAN BE AN ACTIVITY LOG, INTO WHICH EACH RUN WRITES AN ENTRY CONTAINING THE RUN NAME, USER-ID, DATE, START TIME AND, AFTER COMPLETION, THE END TIME. THIS GIVES THE SYSTEM ADMINISTRATOR A VALUABLE TOOL FOR MONITORING THE ACTIVITY ON THE SYSTEM. IT ALSO HELPS TO IDENTIFY EXECUTIONS WHICH DID NOT COME TO A SUCCESSFUL END, AS THE END TIME FOR THESE ENTRIES WOULD BE MISSING. AS NO HISTORICAL INFORMATION IS REQUIRED, THE RUN LOG CAN BE ESTABLISHED AS A WRAP-AROUND RID WITH A FIXED NUMBER OF LINES.

AN EXAMPLE OF A LOG-ON AND LOG-OFF PROCEDURE IS SHOWN BELOW.

```
 00120     06   %%%%%%%%%   RUN LOG ENTRY (LOG-ON)      %%%%%%%%%%%%%%%%%%
∂CHG V161 TIME$ CHG V174 DATE1$ RDL,V80,V82,V85,3 3-4 V192 LOK,V80,V82,V85
∂WRL,V80,V82,V85,V192,Y  2-6,9-8,18-8,27-12,40-1,42-4,47-12,60-40  ,V174,\
 V161,'     ',V165,' ',' ',V199,V189/ ,'*','*','*','*','*','*','*','*' .
∂CHG V190 V192 +1 IF V190 GT 309 CHG V190 9 .
∂WRL,V80,V82,V85,3,Y  3-4 *,V190 ULK .
∂ . END OF COPY ELEMENT 00120   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 00121     02   %%%%%%%%%   RUN LOG ENTRY (LOG-OFF)     %%%%%%%%%%%%%%%%%%
∂CHG V161 TIME$ LOK,V80,V82,V85 WRL,V80,V82,V85,V192,Y 18-8  ,V161  ULK .
∂ . END OF COPY ELEMENT 00121   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

2.1.8. TEST AND DEMONSTRATION ENVIRONMENT

DURING THE DEVELOPMENT OF A DATA PROCESSING SYSTEM, A TEST DATA BASE IS REQUIRED TO TEST THE CORRECT FUNCTIONING OF ALL PROGRAMS. THIS DATA BASE MAY BE CREATED EXPLICITLY FOR THIS PURPOSE OR MAY BE A COPY OF A LIVE DATA BASE. PROGRAMMERS ARE ABLE TO CHANGE THIS DATA BASE FREELY TO CREATE THE VARIOUS CONDITIONS THE PROGRAMS ARE EXPECTED TO COPE WITH, WITHOUT DISTORTING ANY LIVE DATA.

IN MANY CASES, THE MODES USED DURING THE DEVELOPMENT WILL BE LOADED WITH THE LIVE DATABASE DURING SYSTEM IMPLEMENTATION AND WILL FROM THAT POINT ON BE USED IN A PRODUCTION MODE.

THIS HAS THE CONSEQUENCE THAT ALL FURTHER TESTING (DEBUGGING AND ENHANCEMENTS) WILL BE DONE WITH THE LIVE DATABASE. THIS PRACTICE CREATES A POTENTIAL HAZARD TO THE INTEGRITY OF THE LIVE DATA BASE - AS CHANGES APPLIED DURING THE TEST WILL HAVE TO BE MANUALLY UNDONE - AND MAKES THOROUGH TESTING OF CHANGES DIFFICULT.

LIKEWISE, ANY DEMONSTRATION OF A SYSTEM WOULD USE THE LIVE DATABASE.

IT IS STRONGLY RECOMMENDED TO INCLUDE TEST AND DEMONSTRATION CAPABILITIES INTO THE SYSTEM DESIGN FROM THE BEGINNING. FOR MAJOR SYSTEMS, THIS WOULD REQUIRE ASSIGNING DIFFERENT MODES FOR THE TEST AND PRODUCTION DATA. WITH SMALL SYSTEMS, IT WILL PROBABLY BE SUFFICIENT TO RESERVE A SEPARATE SET OF RIDS FOR TEST PURPOSES.

EACH RUN MUST DETERMINE FROM THE OUTSET WHETHER THE TEST OR PRODUCTION DATABASE IS TO BE USED. IN SYSTEMS WHICH ALREADY INCLUDE A USER TABLE FOR SECURITY CHECKS, THIS CAN BE EASILY ACCOMPLISHED BY DEFINING THE MODE(S) AND TYPES TO BE USED FOR EACH USER IN THIS TABLE, OR BY PLACING A CODE NEXT TO EACH USER TO DISTINGUISH TEST USERS (PROGRAMMERS AND ANALYSTS) FROM PRODUCTION USERS.

AN EXAMPLE OF SUCH A ROUTINE IS SHOWN BELOW:

```
 PIM10     04   SECURITY ROUTINE  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 ∂CHG V165A12 USER$ FND,V80,V81,8 ' ' 2-12  ,V165 ,V197 . FIND USER
 ∂IF V197 E0 0 RUN PIMS-SEC,V199 .   NOT FOUND - DO VIOLATION STUFF
 ∂RDL,V80,V81,8,V197 15-2,19-3,23-4,28-4 V196,V80,V81,V82 . READ MD,TP
 ∂ . %%%%%%%%%%%%%%%%%%%%  END OF COPY ELEMENT PIM10   %%%%%%%%%%%%%%%%
```

2.2. DATABASE DESIGN

2.2.1. GENERAL

THE DESIGN OF THE DATA BASE IS ONE OF THE MOST CRITICAL STEPS DURING THE DEVELOPMENT OF ANY DATA PROCESSING SYSTEM. THE USE OF MAPPER DOES NOT ALTER THIS SITUATION.

MAPPER OFFERS A VARIETY OF WAYS TO ACCESS DATA BUT POSES PROBLEMS WITH LARGE RECORDS AS THE MAXIMUM LINE LENGTH IS FIXED AT 132 CHARACTERS. WHENEVER THIS NUMBER IS EXCEEDED, DATA MUST BE SPLIT INTO SEVERAL LINES AND A DECISION MUST BE MADE ON HOW TO STORE THE VARIOUS LINE TYPES. ANOTHER RESTRICTION IS THE NUMBER OF LINES PER RID. WHENEVER THE NUMBER OF LINES IN A RID IS LIKELY TO EXCEED THE OPTIMAL RID SIZE, DATA MUST BE DISTRIBUTED OVER A RANGE OF RIDS AND A METHOD MUST BE FOUND TO EFFECT THIS DISTRIBUTION.

A WRONG DECISION CAN SERIOUSLY IMPACT PERFORMANCE AND RESPONSE TIME, WHEN THE COMPLETED SYSTEM IS TURNED OVER TO THE USERS. DATA BASE DESIGN TAKES PLACE AT A VERY EARLY STAGE OF THE PROJECT, BUT EFFECTS OF THE DESIGN WILL BE NOTICED MUCH LATER, OFTEN ONLY AFTER SYSTEM TURN-ON. CORRECTING ANY MISTAKES AFTERWARDS BY RE-DESIGNING DATA ORGANIZATION AND ACCESS METHODS CAN BE A DIFFICULT AND COSTLY JOB.

IT CAN BE SAID THAT THE DATA BASE DESIGN WILL ENABLE OR PREVENT THE

USE OF EFFICIENT RUN DESIGN TECHNIQUES. IF, FOR EXAMPLE, THE DATA FIELDS REQUIRED FOR A CALCULATION ARE NOT PART OF THE SAME RECORD, THIS WOULD PREVENT USE OF THE 'TOT' FUNCTION AND WOULD REQUIRE SEQUENTIAL PROCESSING WITH ALL THE ASSOCIATED OVERHEAD. LIKEWISE, IF THE DATA FIELDS USED FOR A RECORD SELECTION ARE NOT PART OF THE SAME RECORD, THIS WOULD PREVENT USE OF THE 'SRH' FUNCTION.

OBVIOUSLY, NO DATA BASE DESIGN CAN BE IDEAL FOR ALL PROCESSING REQUIREMENTS. THE DATA BASE DESIGNER MUST THEREFORE - AS IS THE CASE WITH TRADITIONAL DATA BASES - PREPARE A LIST OF ALL PROCESSING REQUIREMENTS, STATING THE FREQUENCY AND PRIORITY OF EACH REQUIREMENT. BASED ON THIS INFORMATION THE DATA BASE MUST BE DESIGNED IN SUCH A WAY THAT THE MOST IMPORTANT AND MOST FREQUENT PROCESSES USE THE SHORTEST AND MOST EFFICIENT ACCESS PATHS.

2.2.2. DATA ORGANIZATION

WHENEVER THE NUMBER OF DATA LINES EXCEED THE CAPACITY OF ONE RID, THE DATA MUST BE DISTRIBUTED OVER VARIOUS RIDS AND A METHOD BE ESTABLISHED TO ACCESS THE DATA.

THE MOST EFFICIENT METHOD WOULD BE TO DETERMINE THE RID NUMBER FROM A PART OF THE RECORD KEY ITSELF. ANOTHER METHOD COULD BE TO USE AN INDEX TO DETERMINE THE TARGET RID. USING THE INDEXED METHOD WOULD REQUIRE FREQUENT MONITORING OF THE DATA BASE TO DETERMINE THE NEED FOR A RE-ORGANIZATION AND EXPANSION.

ANOTHER REASON FOR DISTRIBUTING DATA OVER VARIOUS RIDS CAN BE THE REQUIREMENT TO ALLOW CONCURRENT ON-LINE UPDATING BY SEVERAL USERS. SPLITTING THE DATA OVER A SERIES OF RIDS REDUCES THE PROBABILITY OF UPDATE CONFLICTS.

MOST SYSTEMS INCORPORATE A SET OF STANDING DATA TABLES FOR VALIDATION AND OTHER PURPOSES. THESE TABLES ARE MAINLY USED FOR 'LOOK-UPS' AND

ARE UPDATED ONLY VERY INFREQUENTLY. IT BECOMES THEREFORE AN IMPORTANT CONSIDERATION TO MAKE THE SEARCHING OF THESE TABLES AS EFFICIENT AS POSSIBLE. ONE WAY TO ACHIEVE THIS IS TO STORE THE DATA SORTED IN ASCENDING ORDER SO THAT THE BINARY FIND (BFN) CAN BE USED.

2.2.3. LONG RECORDS

WHENEVER A DATA RECORD EXCEEDS 132 CHARACTERS THE DATA FIELDS MUST BE SPLIT INTO SEVERAL LINES - A SITUATION WHERE ONE LOGICAL RECORD CONSISTS OF MULTIPLE PHYSICAL RECORDS. THE DATA BASE DESIGNER HAS TO DECIDE WHETHER THE RECORDS SHOULD BE SPLIT ACROSS TYPES OR 'FOLDED' AS MULTIPLE LINES WITHIN THE SAME RID. BOTH METHODS HAVE THEIR ADVANTAGES AND DISADVANTAGES AND THE METHOD CHOSEN SHOULD BE DETERMINED BY THE PROCESSING REQUIREMENTS.

STORING MULTIPLE RECORD TYPES IN ONE RID PRECLUDES PROCESSING VIA MANUAL FUNCTIONS.

IF DATA IS SPLIT OVER VARIOUS RIDS, EACH RECORD MUST CONTAIN THE RECORD KEY.

EVEN WHEN THE DIFFERENT DATA RECORDS ARE STORED IN THE SAME RID, IT IS OFTEN FOUND THAT THE RECORD KEY IS REPEATED IN ALL RECORDS. THIS IS PARTLY DUE TO RESTRICTIONS IN EARLIER MAPPER LEVELS. IT IS NOW POSSIBLE TO CREATE 'SETS' OF ONE TAB-CODE HEADER LINE FOLLOWED BY MULTIPLE ASTERISK LINES. NEW OPTIONS IN THE 'SRH' STATEMENT ALLOW TO PERFORM SEARCHES ON THE ASTERISK LINES WHICH INCLUDE EITHER THE HEADER OR THE ENTIRE SET IN THE SEARCH RESULT. THE POSSIBILITY OF USING THIS TECHNIQUE SHOULD BE CAREFULLY CONSIDERED DURING DATA BASE DESIGN.

2.2.4. OTHER CONSIDERATIONS

A NUMBER OF OTHER ASPECTS HAVE TO BE CONSIDERED DURING THE DATA BASE

DESIGN.

- CONCURRENT UPDATES

  IF CONCURRENT UPDATES TO THE DATA BASE ARE POSSIBLE, A METHOD HAS TO BE DEFINED TO AVOID CONFLICTS, SUCH AS LOCKING THE RID TO BE UPDATED.

- UPDATE METHOD

  TO AVOID THE OVERHEAD CAUSED BY RID EXPANSION OR REDUCTION, THE UPDATE METHOD HAS TO BE DEFINED. IF A 'NEXT AVAILABLE LINE' POINTER IS TO BE USED, ITS POSITION HAS TO BE DEFINED. ALTERNATIVELY, UPDATING COULD USE THE 'ADD' AND 'REP' STATEMENTS. IF RECORD DELETION FORMS PART OF THE PROCESSING REQUIREMENTS, THE METHOD TO EFFECT THESE DELETIONS HAS TO BE DECIDED. ONE WAY IS TO OVERWRITE THE DELETED LINE WITH SPACES. IF THIS METHOD IS USED, A 'PURGE' PROCESS HAS TO BE DESIGNED TO REMOVE THE BLANK LINES ON A REGULAR BASIS.

- MASTER LOCK

  IF ANY PROCESSING (SUCH AS A PURGE) REQUIRES THAT THERE IS NO OTHER ACTIVITY ON THE ENTIRE DATA BASE, A MASTER LOCK HAS TO BE ESTABLISHED. INSTEAD OF LOCKING EACH RID, THE PURGE PROCESS WOULD ONLY NEED TO DISABLE THE MASTER LOCK TO PREVENT FURTHER ACTIVITY AS LONG AS THE PURGE IS IN PROGRESS.

3. PROGRAMMING STANDARDS

3.1. PREPARATIONS

3.1.1. RUN SPECIFICATIONS

THE 'WORK TRIGGER' FOR WRITING A NEW MAPPER RUN ARE THE RUN SPECIFICATIONS. THESE SHOULD BE PREPARED ACCORDING TO THE SYSTEM DOCUMENTATION STANDARDS. (HOWEVER, IT IS RECOGNIZED THAT SOMETIMES RUN SPECIFICATIONS WILL BE INFORMAL AND RUDIMENTARY.) SPECIFICATIONS SHOULD INCLUDE ALL RECORDS PROCESSED BY THE RUN AND THE LOCATION AND ACCESS METHOD OF THESE RECORDS.

IT IS IMPORTANT THAT THE RUN DESIGNER GETS A COMPLETE UNDERSTANDING OF THE RUN'S PURPOSE AND THE PROCESSING NECESSARY TO ACHIEVE THE DESIRED RESULT.

THE RUN SPECS SHOULD ALWAYS BE REVIEWED WITH THE ANALYST WHO WROTE THEM. THIS REVIEW SHOULD CONSIST OF TWO PARTS: FIRSTLY THE ANALYST WILL CLEAR UP ALL QUESTIONS WHICH AROSE FROM THE LECTURE OF THE RUN SPECS. SECONDLY THE RUN DESIGNER WILL DESCRIBE IN HIS/HER OWN WORDS THE PROPOSED SOLUTION. EVEN IF THERE DID NOT SEEM TO BE ANY QUESTIONS, THIS SECOND PART WILL OFTEN REVEAL THAT WHAT THE RUN DESIGNER UNDERSTOOD WAS NOT WHAT THE ANALYST INTENDED.

3.1.2. RUN FLOW CHART

THE NEXT STEP OF THE PREPARATIONS IS TO PREPARE A RUN FLOW CHART. FLOW CHARTS ARE NO LONGER REQUIRED AS PART OF THE DOCUMENTATION, BUT STILL SERVE A USEFUL FUNCTION DURING THE RUN DEVELOPMENT PHASE. FOR THIS PURPOSE, FLOW CHARTS MAY BE INFORMAL, IN FACT, A LIST OF ALL

PROCESSING STEPS, DEPICTING THE SEQUENCE IN WHICH THEY ARE PERFORMED AND THE LOGIC CONDITIONS WHICH CAUSE THEM TO BE PERFORMED OR BYPASSED MAY BE COMPLETELY ADEQUATE.

THE COMPLETED RUN FLOW CHART PROVIDES THE RUN DESIGNER WITH A CLEAR IDEA OF THE RUN STRUCTURE, A GUIDANCE ON THE ASSIGNMENT OF LABELS AND HELPS TO DETECT ANY AREA NOT COVERED BY THE RUN SPECS.

3.1.3. RECORD DEFINITIONS

AS THE NEXT STEP THE RUN DESIGNER SHOULD OBTAIN COPIES OF THE RECORD DEFINITIONS FOR ALL RECORDS USED BY THE RUN.

3.1.4. VARIABLE ASSIGNMENT

DEPENDING ON THE SYSTEM OF WHICH THE RUN FORMS A PART, A NUMBER OF VARIABLES WILL BE RESERVED AS COMMON VARIABLES, MODE AND TYPE ASSIGNMENT AND FOR USE IN COMMON ROUTINES. THIS MUST BE TAKEN INTO CONSIDERATION WHEN ASSIGNING VARIABLES FOR DATA FIELDS, COUNTERS AND WORK FIELDS. THE USE OF A VARIABLE WORK SHEET MAY PROVE BENEFICIAL AT THIS STAGE.

3.2. EASE OF MAINTENANCE

LIKE EVERY OTHER PROGRAM, MAPPER RUNS WILL AT ONE STAGE OR ANOTHER NEED TO BE DEBUGGED AND/OR MODIFIED.

PERHAPS THE GREATEST WEAKNESS OF MAPPER IS THE DIFFICULTY INVOLVED IN MAINTAINING RUNS. THIS IS BECAUSE THE SYNTAX OF THE RUN LANGUAGE IS AWKWARD, VARIABLE NAMES ARE NON-DESCRIPTIVE, MODULARIZATION IS WEAK, AND STRUCTURING CONSTRUCTS ARE POOR. THESE SHORTCOMMINGS IN MAPPER ITSELF MUST BE COMPENSATED BY SUPERIOR TECHNICAL DOCUMENTATION AND ADHERENCE TO PROGRAMMING GUIDELINES.

MAINTAINABILITY IS THEREFORE AN IMPORTANT OBJECTIVE DURING RUN DESIGN. THE STANDARDS LAID DOWN IN THIS DOCUMENT WERE ESTABLISHED, TO MAKE SURE ALL MAPPER RUNS ARE DESIGNED IN A STANDARDIZED WAY, SO THAT MAINTENANCE CAN BE CARRIED OUT QUICKLY AND EFFICIENTLY.
EXTENSIVE USE OF COMMENTS IS ENCOURAGED TO PROVIDE FOR BETTER READABILTY.

3.3. RUN ORGANIZATION

LABELS MUST BE ASSIGNED IN ASCENDING ORDER.
IT IS THEREFORE ADVISED THAT WHEN ASSIGNING LABELS, THEY BE INCREMENTED IN INTERVALS OF 3 OR 5, THUS ALLOWING FOR THE INSERTION OF ADDITIONAL LABELS DURING RUN EXPANSION AND MAINTENANCE.

EACH MAPPER RUN SHOULD BE STRUCTURED INTO FIVE MAIN SECTIONS AS DEFINED BELOW. THE RANGE OF LABEL NUMBERS GIVEN FOR EACH SECTION SHOULD BE UNDERSTOOD AS A GENERAL GUIDELINE FROM WHICH DEVIATIONS ARE POSSIBLE, ACCORDING TO THE CIRCUMSTANCES.

3.3.1. PROGRAM INITIATION

THIS SECTION CONTAINS ALL RUN VARIABLE DEFINITIONS AND INITIAL VALUE ASSIGNMENTS. FOLLOWING THE INITIAL VALUE, A BRIEF VARIABLE DESCRIPTION. THESE DEFINITIONS SHOULD APPEAR FIRST IN THE RUN.

THE FOLLOWING RULES APPLY TO THE DEFINITION OF VARIABLES:

A) SEVERAL VARIABLES MAY BE DEFINED IN ONE LINE, USING A SEMICOLON IN PLACE OF THE COLON FOR THE SECOND AND ALL FOLLOWING DEFINITIONS. IF THIS IS DONE, CARE SHOULD BE TAKEN, NOT TO USE A PERIOD IN THE VARIABLE DESCRIPTION BECAUSE THE SYSTEM WILL IGNORE ALL FURTHER DEFINITIONS IN THIS LINE, REGARDING THEM AS COMMENTS.

B) VARIABLES SHOULD NOT BE USED FOR MORE THAN ONE PURPOSE. THIS

MAKES MAINTENANCE UNNECESSARILY DIFFICULT AND IS REALLY TRYING TO SAVE IN THE WRONG PLACE.

C) VARIABLES SHOULD BE GROUPED TOGETHER ACCORDING TO THEIR USE AS SHOWN IN THE STANDARD RUN SKELETON.

D) VARIABLES 1 TO 19 SHOULD BE AVOIDED, IF POSSIBLE. IF USED, THEY SHOULD BE CODED AS 001 TO 019. THIS MAKES IT POSSIBLE TO USE THE 'LOC' AND 'CHG' FUNCTION ON THE RUN CONTROL RID.

E) IN ORDER THAT THE VARIABLE DESCRIPTION MAY APPEAR IN THE CROSS REFERENCE LIST PRODUCED BY XREF, IT SHOULD BE PRECEEDED BY AN = SIGN.

ANY INITIAL PROCESSING SUCH AS SECURITY CHECKS, RUN LOGGING, TABLE LOOK-UP FOR MODE AND TYPE ASSIGNMENT SHOULD ALSO BE INCLUDED IN THE INITIALIZATION SECTION OF THE RUN.
THE INITIAL MENU OR OPTION SCREEN COULD ALSO BE INCLUDED HERE.

LABELS USED IN THIS SECTION SHOULD BE IN THE RANGE OF 1 TO 19.

NOTE: IN GENERAL, ONCE COMPLETED, THE RUN INITIALIZATION SECTION SHOULD NOT BE EXECUTED AGAIN DURING RUN EXECUTION.

3.3.2. PROCESSING SECTION

THIS IS THE HEART OF THE MAPPER RUN. IT IS USUALLY EXECUTED OVER AND OVER AGAIN. STRAIGHT LINE CODING SHOULD BE USED WHERE POSSIBLE. THIS MEANS THE 'LONGEST LEG' OF THE PROCESSING CODE SHOULD READ STRAIGHT THROUGH. DON'T USE GTO'S TO JUMP THRU IT. GTO'S IN THE RUN SHOULD BE USED TO BRANCH AWAY FROM THE MAIN LINE CODE TO PERFORM ERROR ROUTINES AND OTHER SHORTER PROCESSING LEGS.

IT MAY BE ADVISABLE TO VISUALLY SUB-DIVIDE THE PROCESSING SECTION INTO SUB-SECTIONS TO IMPROVE MAINTAINABILITY.

LABELS USED IN THIS SECTION SHOULD BE IN THE RANGE OF 20 TO 149.

3.3.3. FINALIZATION SECTION

ANY HOUSEKEEPING THE RUN NEEDS TO PERFORM, AFTER COMPLETING ALL PROCESSING, SHOULD BE DONE HERE; SUCH THINGS AS LOGGING THE TERMINATION OF THE RUN, CLEARING ANY 'RUN LOCKS' NOT ALREADY CLEARED, ETC.

LABELS USED IN THIS SECTION SHOULD BE IN THE RANGE OF 150 TO 169.

NOTE: IN GENERAL, ANY ROUTINE WHICH NEEDS TO BE PERFORMED ONCE ALL PROCESSING HAS BEEN COMPLETED, SHOULD BE INCLUDED IN THIS SECTION.

3.3.4. ERROR SECTION

THE PRIMARY USE FOR THIS SECTION IS TO GIVE ERROR MESSAGES BACK TO THE USER ON LINE ZERO OF THE SCREEN WHEN INVALID DATA WAS ENTERED. WHEN AN ERROR IS DETECTED IN THE PROCESSING SECTION THE USE OF A 'GTO' TO A SPECIFIC ERROR LABEL OR A STANDARDIZED ERROR ROUTINE SHOULD BE PERFORMED.

LABELS USED IN THIS SECTION SHOULD BE IN THE RANGE OF 170 TO 189

3.3.5. VARIABLE DUMP SECTION

THIS SECTION IS PRIMARILY USED TO AID IN BUILDING AND DEBUGGING A MAPPER RUN. ALL VARIABLES USED IN THE RUN SHOULD BE INCLUDED IN EDIT LINES WITH A BRIEF DESCRIPTION OF WHAT THEY ARE. VARIABLES SHOULD BE PRESENTED IN THE SEQUENCE IN WHICH THEY ARE DEFINED IN THE INITIALIZATION SECTION.

AS THE RUN IS BEING BUILT, THE RUN DESIGNER CAN USE A 'GTO' TO THE VARIABLE DUMP SECTION TO VERIFY THAT THE VARIABLES CONTAIN THE EXPECTED VALUES.

ONCE THE RUN HAS BEEN COMPLETED, THE VARIABLE DUMP SECTION IS USED TO SEND THE VARIABLE DUMP ALONG WITH AN ERROR MESSAGE TO THE PERSON RESPONSIBLE FOR THE APPLICATION, WHEN UNFORESEEN CONDITIONS OCCUR. SUCH CONDITIONS MIGHT BE WHEN THE RUN PERFORMS A READ OF A DATA LINE IN A SYSTEM TABLE THAT FOR SOME REASON IS NO LONGER PRESENT, OR WHEN MAX TIME HAS EXPIRED ON THE WAIT FOR A DATA RID TO BE UNLOCKED.

A VARIABLE DUMP SHOULD ALSO BE PRODUCED WHEN THE RUN BRANCHES TO THE REGISTERED ERROR ROUTINE DUE TO A PROGRAM ERROR. (SEE 3.4.4.)

LABELS USED IN THIS SECTION SHOULD BE IN THE RANGE OF 190 TO 199.

3.4. EASE OF USE

3.4.1. THE 'HUMAN FACTOR'

UNLIKE MOST OTHER APPLICATION PROGRAMS, MAPPER RUNS WILL TO A GREAT EXTENT BE USED BY NON-DATA-PROCESSING PERSONNEL. THE PRIME C´NCERN OF THESE USERS IS NOT OPERATING A COMPUTER SYSTEM, BUT GETTING THEIR PARTICULAR JOB DONE. THIS REQUIRES A CHANGE OF ATTITUDE FROM THE DATA PROCESSING PROFESSIONAL. THE FINAL PRODUCT MUST NOT ONLY PRODUCE THE REQUIRED RESULTS, IT MUST DO SO IN A WAY WHICH IS UNDERSTANDABLE TO THE LAYMAN. THIS FACT MUST BE TAKEN INTO CONSIDERATION, WHEN DESIGNING INPUT OR OUTPUT SCREENS, ERROR-MESSAGES ETC. THE USER MUST BE GUIDED THROUGH THE VARIOUS PROCESSING STEPS IN A DIALOGUE BETWEEN SYSTEM AND USER WHICH SEEMS MOST NATURAL TO HIM, BEING GEARED TO HIS WORK BACKGROUND.

APPLICATION RUNS SHOULD BE CAPABLE OF BEING STARTED BY SEVERAL

METHODS, DEPENDING ON THE USER'S EXPERTISE.

1. VIA MENU SELECTION OF RUN NAME. FOR EXAMPLE THE ENTRY POINT RUN WILL DISPLAY A MENU OF ALL FUNCTIONS AVAILABLE WITHIN A SPECIFIC APPLICATION. THE USER CAN THEN ACTIVATE THE DESIRED RUN BY SELECTING IT FROM THE MENU

2. VIA RUN NAME. THE USER, BY ENTERING THE RUN NAME FOR THE DESIRED FUNCTION, CAN START THE RUN DIRECTLY WITHOUT GOING THROUGH THE MENU SELECTION.

3. VIA RUN NAME AND PARAMETERS. BY ENTERING THE REQUIRED INPUT PARAMETERS THE USER MAY START THE REQUIRED PROCESS WITHOUT FURTHER ACTION.

THIS ENABLES THE SYSTEM TO BE TAILORED FOR THE EXPERT USER AND STILL BE ABLE TO ASSIST THE NOVICE.

3.4.2. SCREEN HANDLING

SCREENS ARE THE MEANS OF COMMUNICATION BETWEEN THE USER AND THE APPLICATION. GREAT CARE MUST BE TAKEN WHEN DESIGNING INPUT AND OUTPUT SCREENS TO MAKE THIS COMMUNICATION WORK.

THE LAYOUT OF INPUT AND OUTPUT SCREENS WILL VERY MUCH DEPEND ON THE APPLICATION AND ITS USERS. THERE ARE, HOWEVER, SOME GENERAL RULES TO BE FOLLOWED.

1. LINE 0 MUST BE LEFT FREE FOR ERROR MESSAGES.

2. EVERY SCREEN MUST SHOW THE RUN NAME, SCREEN NUMBER AND ANY PERTINENT INFORMATION FOR USER REFERENCE.

3. THE SCREEN MUST GIVE GUIDANCE ON SIZE, CLASS AND FORMAT OF INPUT FIELDS. DATE FIELDS SHOULD INDICATE THE FORMAT BY PLACING YY FOR YEAR, MM FOR MONTH AND DD FOR DAY IN THE APPROPRIATE COLUMNS.

PROGRAMMING STANDARDS

THE FOLLOWING SYMBOLS MUST BE USED ON INPUT SCREENS:

MANDATORY FIELDS	MULTI POSITION	<	>
	SINGLE POSITION	+	
OPTIONAL FIELDS	MULTI POSITION	(	)
	SINGLE POSITION	-	

4. ALL INPUT SCREENS SHOULD PROVIDE THE POSSIBILITY TO EXIT FROM THE RUN.

5. WHEN DISPLAYING A RID OR RESULT FOR UPDATING, THE 'HOLD LINES' OPTION (AVAILABLE SINCE LEVEL 30) IN THE 'DSP' STATEMENT SHOULD BE USED.

6. WHENEVER POSSIBLE, SCREENS SHOULD BE DESIGNED IN CO-OPERATION WITH THE END USER.

7. IT IS POSSIBLE - PARTICULARLY AFTER AN ERROR - THAT THE USER TRANSMITS WITHOUT PLACING THE CURSOR AFTER THE LAST INPUT FIELD. USING NORMAL PROCEDURES, THE REMAINING INPUT FIELDS WOULD BE LOST. TO AVOID THIS, ALL SCREENS CONTAINING MULTIPLE INPUT FIELDS MUST CONTAIN A TRANSMIT POSITION ON THE BOTTOM LINE.
THE RUN MUST THEN DETERMINE THE VERTICAL POSITION OF THE CURSOR AT THE TIME OF TRANSMIT. IF THE CURSOR WAS NOT POSITIONED AT THE BOTTOM LINE, ANOTHER OUT (AND ANOTHER CHG INVAR$ OR INPUT$) MUST BE EXECUTED, UTILIZING THE FORCED TRANSMIT OPTION. THIS TECHNIQUE IS DEMONSTRATED BELOW:

```
∂CHG INVAR$ V20,V21,V22,V23,V24,V25,V26,V27,V28,V31,V33,V34,V29,V30 .
∂BRK OUT,V80,V81,-0,2,24,0,V103,Y,N,,P .                DISPLAY SCREEN
∂CHG V180 CURV$ IF V180 NOT LT 18 GTO 25 .      CHECK VERT POS CURSOR
∂CHG INVAR$ V20,V21,V22,V23,V24,V25,V26,V27,V28,V31,V33,V34,V29,V30 .
∂OUT,V80,V81,-1,0,0,0,15,N,,,P,Y .                  OUTPUT SCREEN AGAIN
∂25: .
```

3.4.3. ON-LINE VALIDATION

IN MOST CASES INPUT SCREENS WILL BE FOLLOWED BY A VALIDATION OF THE INPUT DATA. IF AN ERROR HAS BEEN DETECTED, THIS MUST BE COMMUNICATED BACK TO THE USER. TREATMENT OF USER ERRORS UNDER MAPPER DIFFERS GREATLY FROM THE TREATMENT UNDER TRADITIONAL BATCH CONDITIONS.

- DON'T ABORT THE RUN BECAUSE ONE INPUT FIELD IS WRONG OR MISSING! REMEMBER THAT YOU ARE IN A DIALOGUE. GIVE THE USER A CHANCE TO CORRECT OR CLARIFY THE REQUEST.
- DON'T ERASE ALL INPUT FIELDS BECAUSE ONE FIELD IS WRONG. SHOW ALL FIELDS EXACTLY AS THEY WERE KEYED IN SO THAT THE USER CAN SEE THE INVALID ENTRY.
- GIVE CLEAR AND COMPREHENSIBLE INFORMATION WHICH FIELD IS WRONG AND WHY.
- POSITION THE CURSOR TO THE (FIRST) FIELD IN ERROR.
- OFFER MORE DETAILED INFORMATION WITH SUGGESTIONS ON HOW TO CORRECT THE ERROR, IF THE PROCESSING IS COMPLEX AND MERITS THIS KIND OF TREATMENT.

THERE ARE TWO METHODS OF VALIDATING INPUT DATA:

A) TO VALIDATE ALL FIELDS ON THE SCREEN BEFORE RETURNING AN ERROR MESSAGE TO THE USER.

B) RETURNING AN ERROR MESSAGE AS SOON AS AN ERROR IS DETECTED, WITHOUT HAVING CHECKED THE SUBSEQUENT FIELDS.

METHOD A) AVOIDS REPETITIVE ERROR PROCESSING BY DETECTING AND INDICATING ALL ERRORS IN ONE PROCESS. THE DISADVANTAGE OF THIS METHOD IS THAT THE ERROR TEXT CAN ONLY BE VERY GENERAL. THE ERRONEOUS FIELDS

HAVE TO BE FLAGGED WITH AN ERROR INDICATOR ('?' OR SIMILAR).

METHOD B) CAN PROVIDE THE USER WITH A MORE SPECIFIC DIAGNOSIS AS EACH ERROR MESSAGE CAN BE GEARED TO ONE PARTICULAR ERROR CONDITION. THE FACT THAT THE ERROR PROCESSING MAY HAVE TO BE REPEATED SEVERAL TIMES IS CONSIDERED LESS IMPORTANT AS THERE IS SELDOM MORE THAN ONE ERROR ON ONE INPUT SCREEN.

METHOD B), BECAUSE OF THE MORE SPECIFIC ERROR DIAGNOSIS MAKES IT ALSO POSSIBLE TO OFFER THE USER AN EXPLANATION OF THE ERROR CONDITION AND ITS REMEDIES.

AN EXAMPLE FOR SUCH AN ERROR ROUTINE IS SHOWN BELOW.

```
 00110      10   %%%%%%%%%   COMMON ERROR ROUTINE          %%%%%%%%%%%%%%
∂ . ALL RUNS WISHING TO OUTPUT ERROR MESSAGES, JUMP TO 188
∂188:BRK,V80,V81   RDL,V80,V82,V98,V197 1-80 V188   . READ ERROR TEXT
V188
EXPLAIN (  ,)
∂BRK OUT,V80,V81,-0,2,2,0,V196,N,N,,P,,A,Y .       . OUTPUT ERROR
∂CHG V180 CURV$  IF V180 GT 1 GTO V198 .           . RETURN
∂SRH,V80,V82,V99 DH 2-3  ,V197                     . FIND EXPLANATION
∂DSP,V80,V82,-0 .                                  . DISPLAY IT
∂CHG V196 V196 -1  GTO V187 .                      . RETURN
∂ . END OF COPY ELEMENT 00110   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

3.4.4. ERROR HANDLING

DURING THE EXECUTION OF A MAPPER RUN, THREE TYPES OF ERRORS MAY OCCUR. EACH TYPE HAS DIFFERENT CAUSES AND MUST BE DEALT WITH DIFFERENTLY.

1. USER ERRORS

A USERS REQUEST CAN'T BE PROCESSED BECAUSE DATA ENTERED DOES NOT COMPLY WITH ESTABLISHED RULES, MANDATORY FIELDS HAVE NOT BEEN ENTERED, A REQUESTED ITEM IS NOT ON FILE, OR THE REQUEST WAS NOT ALTOGETHER INTELLIGIBLE. THIS TYPE OF ERROR IS CAUSED BY THE USER AND CAN BE CORRECTED BY HIM.

TREATMENT OF THESE ERRORS HAS BEEN DEALT WITH IN THE PREVIOUS SECTION.

2. INTERNAL ERRORS

PROCESSING CANNOT BE COMPLETED BECAUSE AN INTERNAL TABLE USED BY THE RUN IS INCOMPLETE OR INCONSISTENT, DATA TO BE UPDATED IS LOCKED BY ANOTHER RUN, ETC. THIS TYPE OF ERROR IS NOT CAUSED BY THE USER AND HE CAN DO NOTHING TO RECTIFY THE SITUATION.

THIS TYPE OF ERROR CAN BE DETECTED BY THE RUN. IN MOST CASES HOWEVER, SOME INVESTIGATION AND CORRECTIVE ACTION BY THE DATA ADMINISTRATOR OR PROGRAMMER IS NECESSARY BEFORE THE PROCESSSING CAN BE COMPLETED. IF THE RUN CANNOT RECTIFY THE SITUATION WHICH CAUSED THE ERROR, THE RUN MUST BE TERMINATED, GIVING THE USER AN EXPLANATION WHY PROCESSING CANNOT BE COMPLETED AND SENDING A VARIABLE DUMP TO THE ASSIGNED SCOPE.

THE USER WILL THEN CONTACT THE RESPONSIBLE SYSTEMS LEADER TO GET THE PROBLEM SORTED OUT.

AN EXAMPLE OF A VARIABLE DUMP SECTION IS SHOWN BELOW:

```
∂199:BRK,V142,V149 .
∂CHG V182 DATE1$ CHG V196 STNUM$ CHG V197 TIME$ CHG V199 USER$
                    I O S   P R O G R A M   D U M P
  PROGRAM ID  V36                          USER ID          V199
  DATE/TIME   V182 V197                    STATION NUMBER   V196
.COMMON IOS VARIABLES==================.COMMON IOS VARIABLES=======
/V1/1       = 2001 SOE/CUST     /V11/11        = 2073 TRAFFIC REF .
/V2/2       = 2002 LOG NO       /V12/12        = 2118 TWX CHG CD .
/V3/3       = 2124 AMD NO       /V13/13        = 2120 LN INV STAT CD
       .
       .
       .
/V188/188     = LINE$             /V198/198          = TYPE$
/V189/189     = LLP$              /V199/199          = USER$
/V190/190     = MODE$             .
∂BRK,V142,V149 RNM -1 SEN,V142,V149,-1,V37 .
∂AUX,V142,V149,-1,V37,COP,Y .
===>>>   S Y S T E M   E R R O R   -   N O T I F Y   I S & C  <<<===
∂GTO END .
```

3. PROGRAM BUGS

MAPPER ITSELF DETECTS AN ERROR CONDITION DUE TO INCORRECT OR INCOMPLETE CODING AND BRANCHES TO ITS INTERNAL ERROR ROUTINE. THIS TYPE OF ERROR IS NOT CAUSED BY THE USER AND THERE IS NOTHING HE CAN DO ABOUT IT.

CAREFUL RUN DESIGN AND DEBUGGING SHOULD REDUCE THE ERRORS OF THIS TYPE TO A MINIMUM. UNLESS THE 'REGISTER ERROR ROUTINE' IS USED, MAPPER BRANCHES TO ITS INTERNAL ERROR ROUTINE, CONTROL IS NOT RETURNED TO THE RUN AND THE PROGRAMMER CANNOT CHECK THE CONDITIONS THAT CAUSED THE ERROR. THE USER WILL ONLY GET THE STANDARD ERROR MESSAGE WHICH GIVES NO CLUE ABOUT THE CAUSE OF THE

ABORT.

THE FUNCTION 'RER' (REGISTER ERROR ROUTINE) WHICH IS AVAILABLE SINCE RELEASE OF MAPPER LEVEL 30 GIVES THE PROGRAMMER THE POSSIBILITY TO OBTAIN MORE DETAILED INFORMATION ABOUT THE RUN ERROR, INFORM THE USER AND BRING THE RUN TO A CONTROLLED END. THIS CONDITION SHOULD ALSO RESULT IN A VARIABLE DUMP BEING PRODUCED WHICH INCLUDES THE INFORMATION OBTAINED IN THE ERROR ROUTINE.

A SAMPLE OF AN ERROR ROUTINE CAN BE SEEN BELOW:

```
 00007/V01 003 LINES ===> MAPPER ERROR ABORT ROUTINE      <=========
∂LDV V31=497 CHG V170A6 XERR$ CHG V171A4 XFUN$ CHG V172I4 XLINE$ .
∂CHG V173I4 XRID$ CHG V174I6 XTYPE$ LSM,V170 V175S87 .
∂ . %% END OF COPY PROC 00007 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

THE VARIABLE DUMP SECTION WOULD INCLUDE THE FOLLOWING LINES:

```
.MAPPER ABORT INFORMATION =============.==========================
/V170/170  = XERR$ /V175/175
/V171/171          = XFUN$          /V172/172          = XLINE$
/V173/173          = XRID$          /V174/174          = XTYPE$
```

4. RUN EFFICIENCY GUIDELINES

NOTE: THIS SECTION IS ALSO AVAILABLE AS A SEPARATE DOCUMENT

4.1. INTRODUCTION

4.1.1. SCOPE

THE PURPOSE OF THIS SECTION IS TO PROVIDE MAPPER RUN DESIGNERS WITH GUIDELINES AND USEFUL HINTS TO ENABLE THEM TO PRODUCE EFFICIENT MAPPER RUNS.

4.1.2. WHY ARE RUN EFFICIENCY GUIDELINES NEEDED?

ANY COMPUTER, NO MATTER HOW POWERFUL IT MAY BE, CAN ONLY OFFER LIMITED RESOURCES TO ITS USERS. WHEN SEVERAL APPLICATIONS WITH A MULTITUDE OF USERS COMPETE FOR THE USE OF THESE RESOURCES, RESPONSE TIMES INCREASE AND A GERNERAL DEGRADATION OF THE SERVICE PROVIDED BY THE APPLICATIONS CAN BE REGISTERED.

IN THIS ERA OF REAL-TIME DATA PROCESSING IT IS NO LONGER ONLY THE DATA PROCESSING STAFF WHO SUFFER FROM UNSATISFACTORY SYSTEM PERFORMANCE. LARGER AREAS OF BUSINESS LIFE HAVE BEEN AUTOMATED AND DEPEND ON COMPUTER APPLICATIONS.

SLOW RESPONSE, SYSTEM OVERLOADING AND, CONSEQUENTLY, INSTABILITY NOT ONLY FRUSTRATE THE USER COMMUNITY, THEY ALSO SEVERELY REDUCE THE PRODUCTIVITY OF THE INDIVIDUAL USER AND THEREFORE BECOME A CONCERN TO MANAGEMENT.

IT IS OFTEN FOUND THAT PROBLEMS WITH SYSTEM RESPONSE CAN PARTLY BE

RUN EFFICIENCY GUIDELINES

BLAMED ON THE INEFFICIENT USE OF COMPUTER RESOURCES.

MAPPER AS A PROGRAMMING LANGUAGE DIFFERS IN MANY WAYS FROM OTHER LANGUAGES. PARTICULARLY OUTSTANDING FACTORS ARE THE SHORT TIME IT TAKES TO LEARN THE LANGUAGE, TO DEVELOP AND DEBUG SIMPLE RUNS AND THE EASE OF MODIFICATIONS. ALL THESE CHARACTERISTICS OF MAPPER REPRESENT A TREMENDOUS INCREASE IN PROGRAMMER PRODUCTIVITY.

HOWEVER, AS THE SIZE AND COMPLEXITY OF MAPPER-BASED APPLICATIONS INCREASE, THE DEMAND FOR SYSTEM RESOURCES REACHES A LEVEL WHERE IT EFFECTS THE GENERAL SYSTEM PERFORMANCE. IT BECAME EVIDENT THAT THE RUN DESIGNER IN CHOOSING A PROCESSING TECHNIQUE CAN SIGNIFICANTLY INFLUENCE THE SYSTEM PERFORMANCE.

TO WRITE EFFICIENT MAPPER RUNS REQUIRES A DEEPER UNDERSTANDING OF MAPPER THAT GOES BEYOND THE SIMPLE KNOWLEDGE OF THE AVAILABLE COMMANDS.

THESE GUIDELINES HIGHLIGHT SOME INTERNAL CONSIDERATIONS AND AIM TO DIRECT THE RUN DESIGNERS TOWARDS A BETTER UNDERSTANDING AND THEREFORE MORE EFFICIENT USE OF SOME MAPPER CHARACTERISTICS.
THIS DOCUMENT DOES NOT PURPORT TO COVER EVERY POSSIBLE ASPECT OF RUN EFFICIENCY. IT ATTEMPTS TO PROVIDE THE BACKGROUND KNOWLEDGE AND SOME EXAMPLES OF EFFICIENT SOLUTIONS TO ENABLE THE RUN DESIGNER TO DECIDE ON THE MOST SUITABLE PROCESSING TECHNIQUES FOR EACH PARTICULAR PROBLEM.

4.1.3. HOW IS RUN EFFICIENCY MEASURED?

IT HAS BEEN RECOGNIZED FOR SOME TIME THAT A PROGRAMMER CAN BE PRODUCTIVE WITHOUT ACTUALLY WRITING PROGRAM CODE ALL THE TIME. WRITING EFFICIENT RUNS REQUIRES CAREFUL ANALYSIS OF THE BUSINESS PROBLEM. THE PROGRAMMER HAS TO CONSIDER ALTERNATIVE METHODS TO SOLVE THE PROBLEM AND CHOOSE THE MOST SUITABLE TECHNIQUE. SOMETIMES IT MAY EVEN BE

NECESSARY TO WRITE EXPERIMENTAL RUNS TO DETERMINE THE MOST EFFICIENT SOLUTION.

THIS INDICATES ALREADY: EFFICIENCY DOES NOT MEAN FASTER RUN PRODUCTION; IT MEANS PRODUCTION OF FASTER RUNS. THE FACT THAT RUNS ARE DEVELOPED ONLY ONCE AND THEN USED MANY TIMES, MAKES ANY EFFORT SPENT IN SPEEDING-UP RUNS A GOOD AND WORTHWHILE INVESTMENT.

THE MOST OBVIOUS MEASUREMENT FOR EFFICIENCY APPEARS TO BE THE RESPONSE TIME; IN OTHER WORDS: THE TIME WHICH ELAPSES FROM THE MOMENT A REQUEST IS TRANSMITTED UNTIL THE RESPONSE IS RECEIVED. ALTHOUGH THIS IS THE CRITERIA BY WHICH MOST USERS JUDGE THE PERFORMANCE OF A SYSTEM, IT CAN NOT BE THE PRIMARY AND CERTAINLY NOT THE ONLY MEASURE FOR EFFICIENCY. THE RESPONSE TIME MAY VARY GREATLY DEPENDING ON THE SYSTEM LOAD AT THE TIME OF EXECUTION. A USER MAY IN FACT GET A SLOW RESPONSE NOT BECAUSE THE RUN HE IS USING IS INEFFICIENT, BUT BECAUSE SOMEBODY ELSE IS EXECUTING A BADLY WRITTEN RUN AND BLOCKS OTHER USERS' ACCESS TO THE SYSTEM RESOURCES.

THE PRIMARY MEASUREMENT USED THROUGHOUT THIS DOCUMENT IS THE NUMBER OF INPUT/OUTPUT OPERATIONS (I/O'S) PERFORMED BY THE RUN. AN INPUT/OUTPUT OPERATION IN THIS CONTEXT IS EVERY REQUEST TO READ FROM OR WRITE TO EXTERNAL MEMORY.

MAPPER RUNS ARE NOT RESIDENT IN MEMORY. A BLOCK OF CODE IS READ WHENEVER REQUIRED. THIS READING OF THE RUN CONTROL RID (RCR) ITSELF CAN ACCOUNT FOR PART OF THE I/O'S USED DURING EXECUTION. THE NUMBER OF I/O'S USED TO READ THE RCR IS SHOWN SEPARATELY IN ALL TEST RESULTS. A HIGH PROPORTION OF RCR I/O'S OUT OF THE TOTAL I/O'S WOULD NORMALLY INDICATE THE NEED FOR EFFICIENCY IMPROVEMENTS TO REDUCE THIS OVERHEAD.

ANOTHER MEASUREMENT IS THE NUMBER OF LINES PROCESSED. THIS REFERS TO THE NUMBER OF LOGICAL I/O'S (AS OPPOSED TO PHYSICAL I/O'S) THE RUN HAS PERFORMED. THIS MEASUREMENT IS ONLY OF LIMITED USE, AS THE IMPACT OF

RUN EFFICIENCY GUIDELINES

Table 4-1. EXAMPLE 1 TEST RESULTS

TEST/VERSION	I/O'S USED TOTAL	I/O'S USED RCR	LOGIC LINES TOTAL	LOGIC LINES RCR	ELAPSED TIME	RUN TIME SECONDS
TEST A						
VERSION 1	292	188	2545	741	7 SEC	3.458
VERSION 2	95	6	1551	46	2 SEC	0.121
TEST B						
VERSION 1	1850	1382	10491	5434	51 SEC	28.265
VERSION 2	425	6	5053	46	10 SEC	0.272

APART FROM THE OBVIOUS CONCLUSION THAT THE REPORT PROCESSING FUNCTIONS ARE MORE EFFICIENT THAN THE RECORD-ORIENTATED APPROACH, THERE ARE SOME IMPORTANT LESSONS TO BE LEARNED FROM THIS EXPERIMENT:

- THE NUMBER OF I/O'S USED TO READ THE RUN CONTROL RID DOES NOT INCREASE WHEN REPORT PROCESSING FUNCTIONS ARE USED, REGARDLESS OF VOLUME.
- IN THE RECORD PROCESSING VERSION MORE THAN 50 PERCENT OF THE TOTAL I/O'S ARE USED TO PROCESS THE RUN CONTROL RID.
- VERSION 1 USED THREE TIMES AS MANY I/O'S AS VERSION 2 IN THE SMALL VOLUME TEST BUT ALREADY MORE THAN FOUR TIMES AS MANY IN THE MEDIUM VOLUME TEST.
- **INEFFICIENCY INCREASES AT A HIGHER RATE THAN THE DATA VOLUME!**

4.2.2. USE THE INPUT BUFFER - RDL AND RLN

EVERYTIME A 'READ LINE' INSTRUCTION (RDL) IS ISSUED THE MAPPER RUN ACCESSES THE EXTERNAL MEMORY AND PLACES ONE BLOCK OF 448 WORDS INTO THE RUN'S I/O BUFFER. (THIS AVERAGES 19 LINES OF 132 CHARACTERS IN LCS

RIDS OR 12 LINES IN FCS OR FCSU RIDS).
THE 'READ NEXT LINE' COMMAND (RLN) WILL READ THE NEXT LINE FROM THE BUFFER UNTIL ALL LINES HAVE BEEN PROCESSED. ONLY THEN WILL THE NEXT PHYSICAL I/O OCCUR.

WHENEVER MAPPER REPORT PROCESSING FUNCTIONS CANNOT BE USED AND IT IS NECESSARY TO PROCESS A RID SEQUENTIALLY, LINE BY LINE, ~~RRL~~ RLN MUST BE USED AFTER THE INITIAL RDL INSTRUCTION.

READING A RID OF 200 132-CHARCTER LINES USING RDL WILL COST 200 I/O'S VERSUS 11 I/O'S USING BUFFERED READS (17 ON FCS RIDS). THIS REPRESENTS AN INCREASE OF MORE THAN 1000 % IN I/O OVERHEAD, WHICH IS KNOWN TO BE THE MOST COSTLY PART OF A RUN ALREADY.

THERE IS ONLY ONE I/O BUFFER AVAILABLE PER ACTIVE MAPPER RUN. WHENEVER ANOTHER RID OR RESULT IS REFERENCED, THE DATA IN THE BUFFER IS REPLACED BY DATA FROM THIS NEW RID AND BUFFERED PROCESSING OF THE ORIGINAL RID BY RLN FUNCTION IS NO LONGER POSSIBLE.
EXECUTION OF A WRITE LINE (WRL) STATEMENT WILL ALSO CAUSE THE INPUT BUFFER TO BE LOST, EVEN IF THE WRITE IS TO THE SAME RID WHICH IS BEING READ.

NOTE THAT THE ARITHMETIC STATEMENT (ART) CALLS A FORTRAN-BASED SUBROUTINE WHICH ALSO CAUSES THE CONTENT OF THE I/O BUFFER TO BE LOST. IT IS HIGHLY RECOMMENDED TO USE THE CHG FUNCTION INSTEAD OF ART, PARTICULARLY AS IT NOW ALSO INCLUDES A DIVIDE CAPABILITY.

4.2.3. HOW TO LIVE WITH ONE BUFFER - USE YOUR INGENUITY

THESE I/O BUFFER LIMITATIONS HAVE TO BE RECOGNIZED BY THE RUN DESIGNER AND SOLUTIONS HAVE TO BE FOUND TO AVOID NEGATIVE SIDE-EFFECTS ON RUN EFFICIENCY. THIS MAY SOMETIMES REQUIRE A CERTAIN AMOUNT OF INGENUITY AND THE ADOPTION OF METHODS WHICH MAY AT FIRST SIGHT APPEAR UNORTHODOX. IT WOULD BE IMPOSSIBLE TO PRESENT ALL THE POSSIBLE

TECHNIQUES WHICH CAN BE EMPLOYED TO USE BUFFERED PROCESSING AS MUCH AS POSSIBLE. TWO EXAMPLES ARE INCLUDED TO INSPIRE THE READERS IMAGINATION.

4.2.3.1. TABLE ACCESS - THE READ SWITCH

THE BUSINESS PROBLEM FOR THIS EXAMPLE IS THE SAME AS FOR EXAMPLE 1 ABOVE. THE RECORD ORIENTED VERSION PROCESSED THE INPUT RID SEQUENTIALLY AFTER IT HAD BEEN SORTED. HOWEVER, TO OBTAIN THE COST CENTRE NAME THE COST CENTRE TABLE HAD TO BE ACCESSED, WHENEVER THE COST CENTRE NUMBER IN THE DATA RID CHANGED. THE RUN DESIGNER OF VERSION 1 DECIDED THEREFORE THAT THE BUFFERED READ COULD NOT BE USED.

VERSION 3 OF THE SAME RUN SHOWS HOW THE RLN COMMAND CAN STILL BE USED UNDER THESE CIRCUMSTANCES. THE COST CENTRE TABLE IS ONLY ACCESSED WHEN A CHANGE IN COST CENTRE NUMBER IS ENCOUNTERED. THIS MEANS THAT A SUBSTANTIAL NUMBER OF LINES MAY BE PROCESSED IN THE DATA RID UNTIL A BREAK OCCURS. IN FACT USE OF THE RDL STATEMENT IS ONLY NECESSARY AFTER THE COST CENTRE TABLE HAS BEEN REFERENCED. THE RUN DESIGNER OF VERSION 3 USES VARIABLE 102 AS A READ SWITCH WHICH INDICATES WHETHER THE NEXT READ HAS TO BE BY RDL OR RLN FUNCTION. AFTER THE COST CENTRE TABLE IS ACCESSED THE SWITCH IS SET TO PERFORM A RDL; AFTER THE RDL IS EXECUTED THE SWITCH IS SET TO USE THE RLN STATEMENT. THE RESULT IS THAT THE NON-BUFFERED READ VIA RDL STATEMENT IS ONLY EXECUTED WHEN ABSOLUTELY NECESSARY.

RUN EFFICIENCY GUIDELINES

VERSION 3 - THE READ SWITCH

```
∂60:SOR,V80,V81,V86 ' ' 59-4  ,1  RNM -1 .    SORT DATA RID
∂BRK,V80,V81 CHG V100 6  .                    BRK TO OUTPUT HEADERS
.                SUMMARY BY COST CENTRE
*    C O S T    C E N T R E          .          .
* NO .        DESCRIPTION            .  AMOUNT  .
*====.============================.==========.
∂RDL,V80,V81,-1,V100   39-10,59-4 V22,V20 .    READ FIRST LINE
∂CHG V100 V100 + 1 LDV V24=V20,V23=0.00 .      SAVE OLD KEY - ZERO TOT
∂64:CHG V102 72 .                              SET SWITCH FOR RDL
∂FND,V80,V82,V83,,66 ' ' 2-4   ,V20 ,V101 .    FIND CC IN TABLE
∂RDL,V80,V82,V83,V101 13-30 V21 GTO 68 .       READ CC NAME
∂66:LDV V21='****UNKNOWN****' .                CC NOT FOUND
∂68:CHG V23 V23 + V22 .                        ADD AMOUNT
∂GTO V102 .          R E A D   S W I T C H :   GTO RLN/RDL
∂70:RLN,V100,75  39-10,59-4 V22,V20 .          READ NEXT LINE
∂CHG V100 V100 + 1 IF V20 EQ V24 GTO 68 .      SAME KEY - ADD
∂72:RDL,V80,V81,-1,V100,75  39-10,59-4 V22,V20 .  READ NEXT LINE
∂CHG V100 V100 + 1 LDV V102=70 IF V20 EQ V24 GTO 68 . SAME KEY - ADD
 V24 V21 V23
∂LDV V24=V20,V23=0.00 GTO 64 .    ZERO TOT - MOVE NEW KEY TO OLD
∂75: .                            OUTPUT LAST LINE
 V24 V21 V23
∂BRK DSP,V80,V81,-0 .
```

VERSION 3 WAS PUT THROUGH THE SAME TESTS AS THE FIRST TWO SOLUTIONS. THE COMBINED RESULTS ARE PRESENTED BELOW.

Table 4-2. EXAMPLE 2 TEST RESULTS

TEST/VERSION	I/O'S USED TOTAL	I/O'S USED RCR	LOGIC LINES TOTAL	LOGIC LINES RCR	ELAPSED TIME	RUN TIME SECONDS
TEST A						
VERSION 1	292	188	2545	741	7 SEC	3.458
VERSION 2	95	6	1551	46	2 SEC	0.121
VERSION 3	153	58	2504	875	5 SEC	1.882
TEST B						
VERSION 1	1850	1382	10491	5434	51 SEC	28.265
VERSION 2	425	6	5053	46	10 SEC	0.272
VERSION 3	652	185	11732	5030	21 SEC	8.708

ALTHOUGH VERSION 3 DOES NOT COME CLOSE TO THE EFFICIENCY OF THE REPORT PROCESSING FUNCTIONS USED IN VERSION 2, IT REPRESENTS A TREMENDOUS IMPROVEMENT AGAINST VERSION 1. IN CASE MORE COMPLEX REPORT PROCESSING REQUIREMENTS WOULD PREVENT THE USE OF REPORT PROCESSING FUNCTIONS, VERSION 3 SHOULD BE USED.

4.2.3.2. MULTIPLE OUTPUTS

THE PROBLEM OF ONE I/O BUFFER MANIFESTS ITSELF NOT ONLY WHEN MULTIPLE INPUTS ARE PROCESSED. IT IS ALSO NOTICED WHEN MULTIPLE OUTPUTS HAVE TO BE PRODUCED. EXAMPLE 3 IS AN EXTRACT FROM A RUN WHICH CALCULATES DEPRECIATION FOR FIXED ASSETS. THERE IS ONLY ONE INPUT SOURCE BUT THE RUN HAS TO OUTPUT THE UPDATED DATABASE AND ACCOUNTING TRANSACTION FOR THE CALCULATED DEPRECIATION AMOUNTS. A TRADITIONALLY-MINDED PROGRAMMER WOULD HAVE USED A WRITE LINE STATEMENT (WRL) FOR AT LEAST ONE OF THE TWO OUTPUT TYPES. THIS WOULD NOT ONLY REQUIRE ADDITIONAL I/O'S FOR THE WRL STATEMENT ITSELF, IT WOULD ALSO PREVENT THE USE OF THE RLN STATEMENT ON THE INPUT RID.

RUN EFFICIENCY GUIDELINES

IN THE SOLUTION SHOWN HERE BOTH THE UPDATED FIXED ASSET RECORDS AND THE ACCOUNTING TRANSACTIONS ARE OUTPUT INTO THE CURRENT RESULT RID, SIMPLY PREFIXED BY A DIFFERENT RECORD TYPE (A AND B FOR ACCOUNTING ENTRIES; TAB-CODE AND ASTERIX FOR DATABASE RECORDS). AFTER THE ENTIRE INPUT HAS BEEN PROCESSED THE TWO OUTPUT TYPES CAN BE SEPARATED BY TWO SIMPLE SEARCH STATEMENTS FOR FURTHER PROCESSING OF EACH INDIVIDUAL TYPE.

EXAMPLE 3 MULTIPLE OUTPUT

```
∂20:RDL,V80,V81,V86,V104,120 1-132,2-9,12-3,16-5,22-7,30-3,\
34-4,39-10,50-8,59-7,67-30,98-9,108-4,113-9,123-9  V40,\
V20,V21,V22,V23,V24,V25,V26,V27,V28,V29,V30,V31,V32,V33 .
∂IF V20(1-6) EQ ' ' GTO 120 .  FOUND FIRST BLANK LINE - END OF DATA
∂22:RLN,,171 1-132,22-3,26-1,28-2,31-4,79-9 V41,V34,V35,V36,V37,V38
∂IF V41(1-1) NOT EQ '*' GTO 177 .                ERROR IN INPUT RID
∂CHG V104 V104 + 2 .                          INCREASE LINE COUNTER
∂IF V32 NOT GT 0.00 GTO 30 .     CURRENT VALUE 0 - DON'T DEPRECIATE
∂CHG V101 V25(3-2) *12 + V25(1-2) .          CALCULATE CURRENT MONTH
∂CHG V103 V100 - V101 + 1 .                CALC MONTHS TO DEPRECIATE
∂IF V101 GT V102 GTO 30 .   PURCH DATE IN FUTURE - DON'T DEPRECIATE
∂IF V35 EQ 2 GTO 50 .            EXCEPTION: SUM-OF-THE-YEARS-DIGITS
∂IF V35 NOT EQ 1 GTO 30 .       INVALID DEPRECIATON METHOD - BYPASS
∂CHG V106 V30 * V103 / V34 .              CALCULATE NEW DEPR TO DATE
∂IF V106 GT V30 CHG V106 V30 . IF DEPR GT PURCH COST USE PURCH COST
∂CHG V107 V106 - V33  .                 CALC DEPRECIATION THIS MONTH
∂CHG V108 V30 - V106 .                   CALC NEW CURRENT BOOK VALUE
∂LDV V40(108-4)=V60,V40(113-9)=V108,V40(123-9)=V106 .
∂LDV V41(54-4)=V60,V41(59-9)=V108,V41(69-9)=V106,V41(79-9)=V107 .
A1V23       ID000 V36V42 81 F.A.R  V107(1-6)V107(8-2)
B1       V23IDV24 V36V42 81 F.A.R  V107(1-6)V107(8-2)
∂30: .              LABEL TO GO TO IF NO DEPRECIATION IS CALCULATED
```

```
V40
V41
∂RLN,,120 1-132,2-9,12-3,16-5,22-7,30-3,34-4,39-10,50-8,59-7,\
67-30,98-9,108-4,113-9,123-9 V40,V20,V21,V22,V23,V24,V25,V26,\
V27,V28,V29,V30,V31,V32,V33 .
∂IF V20(1-6) EQ ' ' GTO 120 ; GTO 22 .        CHECK FOR END OF DATA
```

4.2.4. REDUCE RCR-READS

AS STATED ABOVE, ONLY A PART OF THE MAPPER RUN IS LOCATED IN MEMORY DURING RUN EXECUTION. WHEN CONTROL PASSES TO A STATEMENT WHICH IS NOT IN MEMORY THE REQUIRED BLOCK OF CODE IS READ FROM EXTERNAL MEMORY.

AS THESE READS OF THE RCR CONSTITUTE I/O'S IN ADDITION TO THOSE NEEDED TO PROCESS THE DATA RIDS, CARELESS CODING CAN SEVERELY IMPACT THE PERFORMANCE OF MAPPER RUNS. RUN DESIGNERS MUST BE AWARE OF THE OVERHEAD CREATED BY THE RCR READS AND WRITE RUNS SO THAT UNNECESSARY I/O'S ARE AVOIDED.

4.2.4.1. COMPACT LOOPS

THE IMPACT OF RCR READS BECOMES PARTICULARLY CRITICAL WHEN THEY OCCUR WITHIN A LOOP WHICH IS EXECUTED MANY TIMES, E. G. FOR EVERY DATA RECORD PROCESSED. IF THE CODE TO PROCESS A DATA RECORD EXCEEDS THE ABOVE MENTIONED NUMBER OF LINES, TWO I/O'S ON THE RUN CONTROL RID ARE NEEDED FOR EACH DATA RECORD PROCESSED!

THESE PROCESSING LOOPS SHOULD BE CODED AS COMPACTLY AS POSSIBLE, WITH SEVERAL STATEMENTS ON THE SAME RCR-LINE. COMMENTS WITHIN THE LOOPS SHOULD BE USED SPARINGLY AND ONLY WHERE THEY DO NOT INTERFERE WITH THE EFFICIENCY OF THE CODE. IF READABILITY OF THE RUN IS A CONCERN, A EXPLANATORY PARAGRAPH OF COMMENTS CAN BE PLACED BEFORE THE LOOP.

THE ABOVE GUIDELINE HAS NOT BEEN FOLLOWED IN THE EXAMPLES INCLUDED IN THIS DOCUMENT FOR SAKE OF READABILITY.

4.2.4.2. BRANCH ON EXCEPTIONS ONLY

WHEN CONDITIONAL BRANCHING IS REQUIRED IN A RUN, THE MOST OFTEN USED PATH OR 'NORMAL' CASE SHOULD BE TREATED IN THE NEXT LINE. GTO OR CONTINGENCY LABEL BRANCHING SHOUID BE RESERVED FOR EXCEPTIONS AND ERROR PROCESSING.

4.2.4.3. AVOID SUBROUTINES

THE USE OF SUBROUTINES (RSR OR GTO STATEMENTS) WILL ALSO CAUSE ADDITIONAL I/O'S ON THE RUN CONTROL RID AND SHOULD THEREFORE BE AVOIDED. THIS AGAIN BECOMES PARTICULARLY CRITICAL WHEN THE SUBROUTINE IS CALLED FROM A PROCESSING LOOP. IN CASE THE SAME CODE NEEDS TO BE EXECUTED FROM SEVERAL PARTS OF THE RUN IT IS PREFERABLE TO PLACE THE SAME CODE SEVERAL TIMES INTO THE RCR RATHER THAN USING A SUBROUTINE. THE SIZE OF THE RUN CONTROL RID DOES NOT IMPACT RUN EFFICIENCY! RUN DESIGNERS SHOULD NOT USE SUBROUTINES UNLESS THEY HAVE A VERY GOOD REASON FOR IT.

4.2.4.4. USE 80-CHARACTER RUN CONTROL RIDS

WHEN USING 80 CHARACTER RUN CONTROL RIDS A LARGER PART OF THE RUN IS READ INTO MEMORY THAN WITH 132 CHARACTER RUN CONTROL RIDS. THE RESULT IS FEWER I/O'S TO READ THE RCR.

4.2.5. USE EFFICIENT UPDATE TECHNIQUES

4.2.5.1. RID EXPANSION/REDUCTION - THE NAL POINTER

EVERY TIME LINES ARE ADDED TO, DUPLICATED OR DELETED FROM AN EXISTING RID, THE ENTIRE RID IS WRITTEN TO THE SYSTEM RECOVERY TAPE. WHEN EXISTING LINES ARE MODIFIED, ONLY THOSE LINES ARE WRITTEN TO THE TAPE (EXCEPT FOR THE FIRST UPDATE OF THE DAY WHICH ALWAYS CAUSES THE ENTIRE RID TO BE WRITTEN). THIS CAN HAVE A SIGNIFICANT IMPACT ON SYSTEM PERFORMANCE, PARTICULARLY WHEN RIDS ARE UPDATED FREQUENTLY DURING A GIVEN DAY.

INSTEAD OF ADDING ONE LINE AT A TIME, IT IS RECOMMENDED TO INITIATE RIDS WITH A NUMBER OF BLANK LINES AND TO KEEP A NEXT AVAILABLE LINE (NAL) POINTER IN THE RID HEADER. RUNS WHICH WOULD NORMALLY ADD LINES TO THE RID WILL USE THIS POINTER TO FIND THE NEXT FREE LINE AND WRITE THE DATA INTO THIS LINE WHICH WILL CAUSE ONLY THIS LINE TO BE WRITTEN TO THE LOG FILE. AFTER COMPLETION THE RUN UPDATES THE NAL POINTER FOR THE NEXT USER. WHEN ALL FREE LINES HAVE BEEN USED ANOTHER BLOCK OF BLANK LINES IS ADDED (MAXIMUM 99 IN ONE LN+ STATEMENT).

THIS METHOD CAUSES THE RID TO BE WRITTEN TO THE LOG FILE MUCH LESS FREQUENTLY THAN IF NEW LINES WERE ADDED FOR EVERY TRANSACTION.

4.2.5.2. AVOID DATE/TIME STAMP UPDATES

CURRENTLY WHENEVER A LINE IN A RID IS UPDATED, LINE 1 OF THE RID IS ALSO UPDATED WITH THE DATE, TIME, AND USER-ID. THEREFORE EVERY WRITE LINE STATEMENT COSTS 4 I/O'S - ONE TO READ THE DATA LINE, ONE TO MODIFY IT, ONE TO READ LINE 1 AND ONE TO UPDATE LINE 1.

TO AVOID THIS OVERHEAD IT IS RECOMENDED TO USE THE 'Y' OPTION IN THE 'NO-TIME-ID' SUBFIELD OF THE WRL STATEMENT. THIS IS PARTICULARLY ADVISABLE WHEN THE NAL POINTER IS USED. THE LAST WRITE TO UPDATE THE NAL POINTER SHOULD BE CODED WITHOUT THE 'Y' OPTION SO THAT DATE, TIME

AND USER-ID OF THE LAST UPDATE ARE RECORDED IN LINE 1.

4.2.5.3. CONSIDER ALTERNATIVE UPDATE METHODS

IT SHOULD ALSO BE NOTED THAT THE WRL STATEMENT OFFERS THE POSSIBILITY TO WRITE UP TO 26 LINES IN ONE STATEMENT. THIS CAN ALSO REDUCE THE NUMBER OF I/O'S REQUIRED.

WHEN A LARGE NUMBER OF LINES HAVE TO BE ADDED TO AN EXISTING RID IT IT IS MORE EFFICIENT TO CREATE A RESULT AND ADD IT TO THE RID RATHER THAN USING THE WRITE LINE STATEMENT.

LIKEWISE, IF A LARGE NUMBER OF LINES ARE UPDATED, IT MAY BE ADVANTAGEOUS TO CREATE A RESULT, UPDATE IT AS REQUIRED, AND THEN REPLACE THE ORIGINAL RID. THE ORIGINAL RID MUST BE LOCKED PRIOR TO UPDATING.

4.2.6. OTHER CONSIDERATIONS

4.2.6.1. USE START LINE ON MULTIPLE FINDS

UNLESS A START LINE IS SPECIFIED IN THE FIND STATEMENT, THE SEARCH WILL START AT THE TOP OF THE RID BEING SEARCHED. WHENEVER MULTIPLE FINDS IN THE SAME RID ARE NECESSARY, THE INPUT RID SHOULD BE SORTED INTO THE SEQUENCE OF THE MASTER RID AND THE LINE NUMBER OF THE PREVIOUS FIND SHOULD BE USED AS THE START LINE FOR CONSECUTIVE FINDS. THIS WILL SAVE UNNECESSARY SEARCHING OF THE TOP PART OF THE MASTER RID. SAVINGS, PARTICULARLY ON LARGE RIDS CAN BE QUITE SUBSTANTIAL!

4.2.6.2. USE LDV INSTEAD OF CHG OR INS

USE THE LDV INSTEAD OF CHG STATEMENT TO LOAD A VARIABLE WITH CONSTANT

DATA. IN THIS CASE LDV CAN BE TWICE AS FAST.

THE LDV STATEMENT CAN BE USED TO LOAD MANY VARIABLES WITH ONE EXECUTION WHERE CHG OR INS WILL REQUIRE THE INTERPRETATION OF SEVERAL FUNCTION CALLS. THE RESULT IS LESS FUNCTIONS TO INTERPRET AND LESS LINES IN THE RCR. THIS CAN BE SIGNIFICANT ESPECIALLY IN LOOPS.

LDV IS THE FASTEST WAY TO RIGHT JUSTIFY, LEFT JUSTIFY, AND ZERO-FILL VARIABLES. PREVIOUSLY IT WAS NECESSARY TO USE AN IF LOOP WITH CHG OR INS STATEMENTS WHICH SHOULD BE ELIMINATED WHENEVER POSSIBLE.

LDV CAN BE USED TO LOAD OR MOVE VARIABLES AND SUB-STRINGS OF MORE THAN 12 CHARACTERS.

4.2.6.3. PRE-SORT RIDS AND USE 'P' OPTION ON MATCH

UNLESS THE 'P' OPTION IS USED THE MATCH FUNCTION WILL FIRST TEST THE DATA RIDS TO SEE IF BOTH ARE SORTED ON THE MATCH FIELDS, SORT THEM WHEN NECESSARY, AND THEN PROCEED WITH THE MATCH.
IT IS 50 % MORE EFFICIENT TO PRE-SORT THE DATA RIDS AND THEN USE THE 'P' OPTION IN THE MATCH STATEMENTS. IN A CASE WHERE THE RIDS ARE NORMALLY EXPECTED TO BE SORTED THE 'P' OPTION SHOULD BE USED IN COMBINATION WITH THE CONTINGENCY LABEL WHICH WILL PERFORM THE SORT ONLY IN THE EXCEPTIONAL CASE.

4.2.6.4. USE HEADER-DIVIDER ON APPEND

WHEN APPENDING RESULTS CREATED IN A RUN, BE SURE THERE IS A HEADER-DIVIDER (*=) LINE IN THE BEGINNING OF THE RESULT. THIS LINE INDICATES TO MAPPER THE END OF HEADERS AND START OF DATA. THE APPEND FUNCTION SCANS THE RESULT FOR THE HEADER-DIVIDER LINE TO DETERMINE THE BEGINNING OF DATA. IF NO HEADER-DIVIDER LINE IS FOUND THE ENTIRE RESULT IS TREATED AS DATA.

IN CASE THERE IS NO HEADER-DIVIDER LINE IN THE RESULT MAPPER HAS TO SCAN THE WHOLE RESULT BEFORE EXECUTING THE APPEND FUNCTION. TO PREVENT THIS ENTIRE REPORT SCAN, PLACE A HEADER-DIVIDER (*=) LINE IN THE RESULT BEFORE BUILDING THE DATA WHICH WILL BE APPENDED TO ANOTHER REPORT.

4.2.6.5. AVOID LARGE RESULTS

WHEN A RESULT IS STARTED WITHIN A RUN, THE SOFTWARE ALLOCATES SPACE FOR 200 LINES OF DATA. ONCE THE RUN ATTEMPTS TO WRITE A RECORD AT LINE 201, THE SOFTWARE ALLOCATES AN AREA FOR 4000 LINES, COPIES THE ORIGINAL 200 LINES TO IT AND THEN ALLOWS THE ADDITION OF LINE 201. SUBSEQUENTLY WHENEVER THE ALLOCATED SPACE IS EXHAUSTED, AN AREA WHICH IS 4000 LINES LARGER THAN THE PREVIOUS AREA IS ASSIGNED AND ALL DATA IS COPIED TO THAT NEW AREA BEFORE THE NEXT LINE CAN BE WRITTEN. IN THE GENERATION OF A 100,000 LINE RESULT THE FIRST 4000 LINES WILL BE HANDLED 20 TIMES, THE NEXT 4000 LINES 19 TIMES, ETC, OR EACH LINE WILL BE HANDLED AN AVERAGE OF 10.5 TIMES.

4.2.6.6. USE CAL INSTEAD OF MULTIPLE TOT'S

WHEN A CALCULATION REQUIRES MULTIPLE 'TOT' STATEMENTS, 'CAL' SHOULD BE USED INSTEAD. CAL CAN ALSO BE USED TO JUSTIFY AND ZERO-FILL COLUMNS. THE AVAILABILITY OF LOGICAL CONDITIONS MAKE CAL A POWERFULL ADDITION TO THE MAPPER COMAND REPERTORY.

4.2.6.7. USE CHG INSTEAD OF ART FOR ARITHMETIC

THE ARITHMETIC FUNCTION (ART) PASSES THE DATA TO THE OPERATING SYSTEM WHERE A COMMON BANK (DMATH$) PERFORMS THE MATHEMATICS THEN PASSES THE RESULT BACK TO THE MAPPER SYSTEM. THE CHANGE FUNCTION IS DONE WITHIN THE MAPPER SYSTEM.

RUN EFFICIENCY GUIDELINES

4.2.6.8. BE AWARE OF PRE-RUN OVERHEAD

EVERY TIME A RUN IS CALLED, PRE-RUN SCANS THE RUN REGISTRATION RID FOR THE MAPPER DEPARTMENT TO FIND THE REGISTRATION OF THE REQUESTED RUN. THIS CAN USE UP TO ONE SECOND WHEN MANY RUNS ARE REGISTERED FOR ONE DEPARTMENT. TO AVOID UNNECESSARY OVERHEAD, THE MOST FREQUENTLY USED RUNS SHOULD BE PLACED IN THE BEGINNING OF THE RUN REGISTRATION RID.

AS A CONSEQUENCE, RUNS SHOULD NEVER RE-START THEMSELF VIA A ∂RUN STATEMENT BUT GO TO THE BEGINNING OF THE RUN CONTROL RID VIA A GTO FUNCTION.
WHEN CALLING ANOTHER RUN, USE OF GTO RPX INSTEAD OF ∂RUN SHOULD BE CONSIDERED.

4.2.6.9. USE AUX SPARINGLY

WHENEVER AN AUX STATEMENT IS EXECUTED, A NEW RID IS ADDED IN A SYSTEM TYPE AND QUEUED TO THE SPECIFIED STATION NUMBER FOR SUBSEQUENT PRINTING. EVERY AUX THEREFORE GENERATES MORE AND MORE ENTRIES IN THE TYPE WHICH CAUSES A GROWTH IN THE TIME REQUIRED TO DETERMINE THE PLACEMENT OF EACH AUX RID.

IF A RUN HAS TO PRINT A SERIES OF RIDS, THOSE RIDS SHOULD BE PLACED INTO A RESULT (SEPARATED BY .EJECT LINES) AND ONLY THE RESULT BE PRINTED. THIS WILL GENERATE ONLY ONE RID IN THE AUX QUEUE.

4.2.6.10. USE D AND H OPTIONS WHERE AVAILABLE

WHERE AVAILABLE (SRH, MCH) THE D AND H OPTIONS SHOULD BE USED TO ELIMINATE THE DETAIL AND HEADER LINES IN THE RUN.

RUN EFFICIENCY GUIDELINES

4.2.7. A WORD ABOUT CHARACTER SETS

SINCE THE RELEASE OF MAPPER LEVEL 30, THE INTERNAL CODE OF MAPPER IS BASED ON ASCII, OR FULL CHARACTER SET. RUNS AND DATA WHICH RESIDE IN LIMITED CHARACTER SET RIDS HAVE TO BE TRANSLATED BEFORE THE STATEMENTS ARE EXECUTED, THUS CREATING AN ADDITIONAL OVERHEAD IN PROCESSOR TIME.

ON THE OTHER HAND, FEWER LINES OF DATA OR RUN CONTROL RIDS CAN BE PLACED INTO ONE BLOCK, WHEN FULL CHARACTER SET IS USED. THIS WORKS OUT AT 19 VERSUS 12 LINES FOR 132 CHARACTER RIDS AND 30 VERSUS 20 LINES FOR 80 CHARACTER RIDS. THE SAVINGS IN I/O'S USING LIMITED CHARACTER SET ARE SUBSTANTIAL AND MORE THAN COMPENSATE THE OVERHEAD IN PROCESSOR TIME.

CURRENT INDICATIONS ARE THAT FUTURE MAPPER RELEASE WILL PROCESS ONE CHARACTER SET ONLY, WHICH IN MOST CASES WILL LEAD TO A MIGRATION TO ASCII.

5. SYSTEM ADMINISTRATION

5.1. INTRODUCTION

WHEN A DATA PROCESSING SYSTEM HAS BEEN COMPLETED AND IS USED IN A 'PRODUCTION MODE', PROCEDURES HAVE TO BE ESTABLISHED TO CONTROL ALL FURTHER CHANGES TO THE RUNS. IN THE MAJORITY OF LOCATIONS, ADEQUATE PROCEDURES HAVE BEEN IN OPERATION FOR SOME TIME TO CONTROL CHANGES TO BATCH SYSTEMS.

MAPPER BASED SYSTEMS REQUIRE A SOMEWHAT DIFFERENT APPROACH FROM SYSTEMS WRITTEN IN COBOL OR OTHER LANGUAGES WHICH REQUIRE A COMPILATION PROCESS.
FOR ONE, THERE IS NO DISTINCTION BETWEEN 'SOURCE' AND 'ABSOLUTE' PROGRAM. THE RUN A PROGRAMMER IS TRYING TO CHANGE, MAY BE EXECUTED BY A USER AT THE SAME TIME. THE CONSEQUENCES CAN BE IRRITATING AT BEST AND DISASTROUS AT WORST.

5.2. CONTROL PRINCIPLES

BASED ON THE CONCEPTS THAT GOVERN THE PROVEN PROCEDURES FOR BATCH PROGRAMS, A SET OF RULES CAN BE ESTABLISHED TO ACCOMPLISH MODIFICATIONS TO MAPPER RUNS WITHOUT DISRUPTING THE ACTIVITY OF THE USERS.

- ALL RUNS ARE LOCATED IN A MODE WHICH IS DIFFERENT FROM THE DATA MODE AND CANNOT BE ACCESSED BY THE USERS.
- A TEST VERSION OF EACH MAPPER RUN MUST BE KEPT SEPARATE FROM THE PRODUCTION VERSION.
- CHANGES ARE APPLIED TO THE TEST VERSION ONLY - NEVER TO THE PRODUCTION VERSION.

- TRANSFER OF TEST RUNS TC THE PRODUCTION SYSTEM MUST BE DOCUMENTED.

- A SEPARATE BACK-UP VERSION MUST BE AVAILABLE FOR RECOVERY.

ONE OF THE METHODS PUT FORWARD TC MAINTAIN TEST AND PRODUCTION RUNS IN SEPARATE RIDS IS TO STORE PRODUCTION RUNS IN EVEN AND TEST RUNS IN ODD RID NUMBERS. AS LONG AS THIS METHOD IS NOT BACKED UP BY AN AUTOMATED PROCEDURE TO RECORD CHANGES AND TO PREVENT DIRECT UPDATES OF THE PRODUCTION RUNS, IT DOES NOT SATISFY ALL THE REQUIREMENTS STATED ABOVE.

A FURTHER SHORTCOMING OF THIS METHOD ARISES WHENEVER A 'GTO RPX' IS USED IN A RUN. THE TEST RUN WOULD HAVE TO GO TO THE TEST VERSION OF THE RPX RID (ODD RID NUMBER). BEFORE THE TESTED RUN IS RELEASED FOR PRODUCTION, ALL THE 'GTO RPX' STATEMENTS HAVE TO BE ALTERED TO GO TO THE PRODUCTION VERSION (EVEN RID NUMBER).

IT IS RECOMMENDED TO USE TWO DIFFERENT MAPPER TYPES FOR TEST AND PRODUCTION RUNS (AND A THIRD TYPE FOR BACKUP), USING THE SAME RID NUMBER FOR ONE RUN IN EACH TYPE. THIS ELIMINATES THE NEED TO CHANGE 'GTO RPX' STATEMENTS.

BASED ON THE CONTROL REQUIREMENTS STATED ABOVE, A MAPPER RUN 'RUN-ADMIN' HAS BEEN DEVELOPED AS PART OF AN APPLICATION DEVELOPMENT AND ADMINISTRATION METHODOLOGY.

5.3. RUN-ADMIN - CHANGE IN A CONTROLLED ENVIRONMENT

RUN-ADMIN PROVIDES THE FOLLOWING TOOLS TO MONITOR SYSTEM DEVELOPMENT AND MAINTENANCE ACTIVITIES:

- AN INDEX OF ALL RUNS BY SYSTEM

- A LOG OF OPEN ACTIVITY (ALL RUNS FOR WHICH PRESENTLY UPDATES ARE

POSSIBLE.

- A LOG OF ALL UPDATES INCLUDING DATE AND USER-ID AND UPDATE REASON. THIS LOG CAN BE INTERROGATED FOR ALL CHANGES BY SYSTEM, RUN NAME, USER-ID, ETC.

ACCESS TO THE RUN INDEX AND THE LOGS IS CONTROLLED BY A SYSTEM INDEX WHICH MAKES IT POSSIBLE TO COMBINE VARIOUS SYSTEMS IN ONE MODE, OR TO MAINTAIN SEPARATE MODES FOR EACH SYSTEM. IT IS ALSO POSSIBLE TO ESTABLISH A SEPARATE RUN SKELETON FOR EACH SYSTEM, CONTAINING THE STANDARD VARIABLE DEFINITIONS TO BE USED. AUTOMATIC MAINTENANCE OF VERSION NUMBERS CAN ALSO BE SELECTED AT SYSTEM LEVEL.

THE FUNCTIONS OFFERED BY RUN-ADMIN ACCOMPANY THE RUN DEVELOPMENT AS IT PROGRESSES FROM INITIATION THRU IMPLEMENTATION TO THE MAINTENANCE PHASE. COMPREHENSIVE SAFEGUARDS ARE PROVIDED AT EACH STAGE TO PREVENT USERS FROM BYPASSING THE BUILD-IN CONTROLS.

1. OPEN NEW RUN

   WHEN THIS FUNCTION IS SELECTED, THE RUN INDEX IS SCANNED FOR THE NEXT AVAILABLE RID NUMBER, AND THE RUN SKELETON FOR THE SYSTEM IS COPIED INTO THIS RID. THE RUN NAME, RUN DESCRIPTION, USER-ID AND DATE ARE ENTERED INTO THE RUN INDEX AND INTO THE HEADLINES OF THE RUN RID. THE RUN IS ALSO ENTERED INTO THE OPEN ACTIVITY LOG.

2. COPY RUN TO PRODUCTION SYSTEM

   AFTER RUN DEVELOPMENT AND TESTING HAVE BEEN COMPLETED, THE RUN IS COPIED INTO THE TYPE ASSIGNED FOR PRODUCTION RUNS. THIS FUNCTION REQUESTS THE INPUT OF AN UPDATE REASON WHICH IS ENTERED INTO THE UPDATE LOG ALONG WITH THE DATE AND USER-ID. IF AUTOMATIC UPDATE OF VERSION NUMBERS HAS BEEN SELECTED, THE VERSION NUMBER IS INCREASED DURING THIS PROCESS. THE USER-ID AND

DATE IN THE RUN INDEX AND IN THE HEADLINES OF THE RUN RID ARE UPDATED, AND BOTH THE TEST AND PRODUCTION VERSION OF THE RUN ARE PROTECTED WITH PASSWORDS AGAINST FURTHER UPDATES. THE ENTRY FOR THIS RUN IN THE OPEN ACTIVITY LOG IS REMOVED.

3. RE-OPEN RUN FOR MODIFICATION

THIS FUNCTION IS USED, WHENEVER MODIFICATIONS ARE REQUIRED TO A RUN WHICH HAD ALREADY BEEN COPIED TO THE PRODUCTION SYSTEM. THE SYSTEM FIRST CHECKS WHETHER THE RUN NAME IS STILL PRESENT IN THE OPEN ACTIVITY LOG, WHICH WOULD INDICATE THAT ANOTHER PROGRAMMER/ANALYST IS PRESENTLY WORKING ON THE RUN IN QUESTION. IN THIS CASE THE REQUESTOR IS INFORMED AND NO FURTHER ACTION IS TAKEN.

BEFORE THE UPDATE PASSWORD IS REMOVED FROM THE TEST RUN, THE RUN IS SAVED INTO THE TYPE ASSIGNED FOR THE BACKUP VERSION. THE RUN NAME IS ALSO ENTERED INTO THE OPEN ACTIVITY LOG.

4. RECOVER RUN FROM BACKUP

THIS FUNCTION IS USED, SHOULD IT BECOME NECESSARY TO FALL BACK ONTO THE VERSION THAT WAS PREVIOUSLY SAVED IN THE BACKUP TYPE. THE BACKUP VERSION IS SIMPLY COPIED INTO THE TEST RUN RID.

5. ENTER NEW SYSTEM INTO INDEX

THIS FUNCTION CREATES A NEW ENTRY IN THE SYSTEM INDEX. THE USER HAS TO SPECIFY THE MODE AND TYPES WHERE THE RUNS WILL BE LOCATED, TYPE AND RID NUMBERS FOR THE SYSTEM INDEX AND LOGS, AND THE RID NUMBER OF THE RUN SKELETON TO BE USED FOR THIS SYSTEM. THE ENTRY ALSO CONTAINS THE MAPPER DEPARTMENT NUMBER WHICH IS ENTITLED TO UPDATE RUNS IN THE NEW SYSTEM. IF SEVERAL DEPARTMENTS MAY ACCESS RUNS OF A SYSTEM, ONE ENTRY HAS TO BE CREATED FOR EACH DEPARTMENT.

6. CHANGE ENTRY IN RUN INDEX.

THE RUN INDEX IS PASSWORD PROTECTED AGAINST MANUAL UPDATES. WHENEVER A RUN NAME OR DESCRIPTION HAS TO BE CHANGED, THIS FUNCTION MUST BE USED. RUNS WHICH ARE NO LONGER REQUIRED CAN BE REMOVED FROM THE RUN INDEX BY SETTING THE RUN NAME AND SYSTEM NAME TO 'OPEN'. THIS WILL CAUSE AN ENTRY TO BE WRITTEN INTO THE UPDATE LOG.

6. DOCUMENTATION STANDARDS

DOCUMENTATION IS NOT A GOAL IN ITSELF, BUT AN AID TO ACHIEVE OTHER GOALS.

TWO DISTINCTLY DIFFERENT TYPES OF DOCUMENTATION ARE REQUIRED, EACH SATISFYING THE NEEDS OF A SPECIFIC GROUP OF PEOPLE.

PROGRAMMERS AND ANALYSTS REQUIRE THE TECHNICAL INFORMATION TO DEVELOP AND LATER MAINTAIN THE SYSTEM.

USERS REQUIRE INFORMATION WHICH HELPS THEM TO UNDERSTAND THE FUNCTIONING OF THE SYSTEM SO THAT THEY CAN PRODUCE THE DESIRED RESULTS.

6.1. TECHNICAL DOCUMENTATION

THE DOCUMENTATION ITEMS LISTED HERE ARE THOSE WHICH MAKE UP THE FORMAL DOCUMENTATION FOR THE SYSTEM AND ARE TO BE RETAINED.

MOST OF THIS DOCUMENTATION IS MAINTAINED WITHIN THE MAPPER SYSTEM. IT CAN BE REFERENCED MANUALLY OR BY RUN FUNCTIONS. IN ORDER TO KEEP TRACK OF THE INDIVIDUAL DOCUMENTS, THREE INDICES ARE MAINTAINED. THESE ARE THE SPECIFICATION INDEX, THE PROGRAM INDEX AND THE DATA STRUCTURE INDEX.

6.1.1. SYSTEM SCOPE

BASED UPON THE SERVICE REQUEST AND INITIAL RESEARCH, THE ANALYST PREPARES THE PRELIMINARY SYSTEM SCOPE. THIS SCOPE OUTLINES THE FUNCTIONAL AREAS TO BE SURVEYED AND THE INFORMATION PROBLEMS TO BE COVERED. THE PORPOSE OF THE SCOPE IS TO ESTABLISH AMONG ALL CONCERNED:

- WHAT IS TO BE STUDIED?

- WHY IS THE EFFORT EXPENDED?

- FUNCTIONAL AREAS INVOLVED?

THE FORMAT CONSISTS OF A 'GENERAL' SECTION DEVOTED TO A DESRIPTION OF PURPOSE AND OBJECTIVES AND A LIST OF 'INVOLVED FUNCTIONS' OR DEPARTMENTS. THE 'GENERAL' SECTION SHOULD PROVIDE A BRIEF DESCRIPTION OF THE SYSTEM INCLUDING WHAT IT IS TO BE USED FOR AND WHAT IT DOES IN ADDITION TO THE ABOVE ITEMS.

THE SCOPE DOCUMENT SHOULD BE PREPARED INITIALLY AS A WORK PAPER AND THEN FINALIZED AT THE COMPLETION OF THE PROJECT AS NECESSARY.

6.1.2. SYSTEM DATA FLOW DIAGRAM

THE SYSTEM DATA FLOW DIAGRAM DEFINES THE LOGISTICS OF HOW DATA FLOWS AND HOW INFORMATION IS DISSEMINATED.

THE SYSTEM DATA FLOW DIAGRAM IS DEVELOPED TO PRESNT THE PROPOSED SYSTEM TO USER MANAGEMENT. IT IDENTIFIES INTERFACES TO AND FROM THE PROPOSED BUSINESS FUNCTION BY DEPARTMENT, SYSTEM, WHAT TYPES OF DATA RESOURCES ARE REQUIRED AND THE CATEGORIES OF INFORMATION AVAILABLE FROM THE SYSTEM.

THE SYSTEM DATA FLOW NARRATIVE COMBINES WITH THE SYSTEM DATA FLOW DIAGRAM TO EXPLAIN THE ESSENTIAL UNDERLYING CONCEPTS OF THE SYSTEM.

6.1.3. PROCESS SPECIFICATION

THE PURPOSE OF THE PROCESS SPECIFICATION IS TO PROVIDE THE LOGIC NARRATIVE TO EXPLAIN THE FUNCTIONS PERFORMED BY PROGRAM. THIS PORTION OF THE PROGRAM SPECIFICATION IS SIGNIFICANT IN THAT IT PROVIDES OTHER ANALYSTS AND PROGRAMMERS WITH THE BACKGROUND THAT WILL ENABLE THEM TO PROVIDE SUSTAINED SUPPORT AND THE DEVELOPMENT OF FUTURE ENHANCEMENTS.

THE BASIC INFORMATION CONTAINED IN THE PROCESS SPECIFICATION IS THE REFERENCES TO THE INPUT/OUTPUT FILES AND AN EXPLANATION OF THE DYNAMIC PROCESSES INVOLVED IN THE MANIPULATION OF THE DATA. OTHER TECHNIQUES SUCH AS H.I.P.O (HIERARCHICAL INPUT PROCESS OUTPUT) HAVE BEEN DEVELOPED TO CONVEY THIS INFORMATION. SOME PROGRAMMERS AND ANALYSTS PREFER THIS TECHNIQUE TO THE LOGIC NARRATIVE APPROACH OF THE PROCESS SPECIFICATION. THE ADVANTAGE OF A H.I.P.O. IS A MORE DIRECT CORRELATION OF THE FUNCTIONS ASSOCIATED WITH THE INPUT AND OUTPUT FILES. THE DISADVANTAGE IS THAT IT LACKS A SYNOPSIS OF THE PROGRAM. THERE APPEARS TO BE NO MAJOR ADVANTAGE OR DISADVANTAGE IN EITHER APPROACH, SO THE CHOICE IS LEFT TO THE DESIGNER.

6.1.4. DATA DICTIONARY

USE OF A DATA DICTIONARY MEANS THAT THERE IS ONE PLACE FOR RECORDING THE NATURE OF ALL DATA ELEMENTS USED IN THE SYSTEM. ANYONE NEEDING INFORMATION ABOUT A PARTICULAR ELEMENT, OBTAINS IT FROM THAT STORE AND NOT FROM PIECES OF PAPER OR VERBAL DEFINITIONS WHICH BECOME OBSOLETE WHEN CHANGES ARE MADE. DATA RECORDS ARE GENERATED FROM THE SAME DATA STORE AND CAN BE QUICKLY REGENERATED WHEN SOMEONE MAKES A CHANGE.

TO OBTAIN THE UNIQUE BENEFITS IT PROVIDES, THE DATA DICTIONARY MUST BE UPDATED AS SOON AS DATA INFORMATION IS DEFINED OR CHANGED.

INCLUDED IN THE DATA DICTIONARY IS ALL THE INFORMATION CONCERNING DATA ELEMENTS NORMALLY NEEDED FOR DOCUMENTATION: DATA DICTIONARY SEQUENCE NUMBER, DATA ELEMENT DESCRIPTION, DEFINITION AND VALIDATION RULES, TYPE AND SIZE.

6.1.5. RECORD DESCRIPTION

THE RECORD DESCRIPTION PROVIDES THE ANALYSTS AND PROGRAMMERS WITH A LIST OF THE DATA ELEMENTS WHICH COMPRISE THE RECORD. A RUN NAMED

'$RECDEF' HAS BEEN PROVIDED TO AUTOMATICALLY GENERATE THE RECORD DESCRIPTION FROM THE INFORMATION IN THE DATA DICTIONARY. THE DATA ELEMENT NAME AND NUMBER, TYPE, SIZE, STARTING POSITION AND ENDING POSITION ARE PROVIDED IN THE RECCRD DESCRIPTION.

6.2. USER DOCUMENTATION

THE DOCUMENTATION PREPARED FOR THE END USER IS EXPECTED TO TELL THE TERMINAL OPERATOR EVERYTHING HE/SHE NEEDS TO KNOW TO USE THE SYSTEM. THE USER MANUAL MUST SERVE BOTH FOR THE TRAINING OF NEW USERS AND AS A REFERENCE FOR EXPERIENCED USERS. IF A SYSTEM IS USED BY SEVERAL DEPARTMENTS THE USER INSTRUCTIONS SHOULD BE BROKEN INTO SECTIONS BY USER GROUP, GIVING EACH GROUP ONLY THE INFORMATION REQUIRED BY THAT GROUP.

IT IS ALSO RECOMMENDED TO PREPARE ON-LINE DOCUMENTATION IN THE FORM OF A SIMPLE 'HELP' RUN WHICH OFFERS THE USER A MENU OF TOPICS FROM WHICH THE REQUIRED INFORMATON CAN BE SELECTED.

BOTH USER MANUAL AND ON-LINE DOCUMENTATION MUST BE WRITTEN WITH THE READER IN MIND. IT SHOULD FOLLOW THE PROCEDURES AND TERMINOLOGY THE USERS ARE ACCUSTOMED TO. IT IS OFTEN BENEFICIAL WHEN THE USER DOCUMENTATION IS PREPARED BY OR IN COOPERATION WITH A USER COORDINATOR.

7. UTILITIES

A NUMBER OF MAPPER UTILITY RUNS HAVE BEEN DEVELOPED TO ASSIST THE PROGRAMMERS AND ANALYSTS DURING SYSTEM DEVELOPMENT, DOCUMENTATION AND ADMINISTRATION. HERE FOLLOWS A BRIEF DESCRIPTION OF THESE UTILITIES. FURTHER DETAILS ARE AVAILABLE, ON REQUEST, FROM I. S. & C. ID.

7.1. LIB-IN

A RUN WHICH IS USED TO COPY A PREVIOUSLY ESTABLISHED COPY ELEMENT FROM A MAPPER COPY LIBRARY INTO A MAPPER RUN.

THE POSITION WHERE THE COPIED LINES ARE TO BE PLACED MUST BE IDENTIFIED BY A '∂COPY XXXXX' LINE, WHERE XXXXXX IS THE FIVE CHARACTER IDENTIFIER OF THE COPY ELEMENT TO BE COPIED.

THE RUN REQUESTS MODE, TYPE AND RID LOCATION OF THE RUN AND MODE TYPE AND RID LOCATION OF THE COPY LIBRARY.

7.2. LIB-OUT

A RUN WHICH IS USED TO REMOVE CODING FROM A MAPPER RUN WHICH HAD PREVIOUSLY BEEN COPIED FROM A PROCEDURE LIBRARY. THE USER MAY SPECIFY A PARTICULAR COPY ELEMENT OR 'ALL' .

THE RUN REQUESTS MODE, TYPE AND RID LOCATION OF THE RUN AND THE COPY ELEMENT IDENTIFIER.

7.3. LIB-CHG

A RUN WHICH COMBINES THE FUNCTIONALITY OF LIB-OUT AND LIB-IN. IT IS USED TO INSERT A NEW VERSION OF A COPY ELEMENT INTO A RUN BY FIRST REMOVING THE OLD CODE AND THEN INSERTING THE NEW ELEMENT. THIS RUN IS

UTILITIES

VERY USEFULL WHEN A RUN IS TRANSPORTED TO SWAP THE COPY ELEMENTS DEFINING MODES AND TYPES, OR TO REPLACE SCREENS BY LOCAL LANGUAGE VERSION. THE USER MAY SPECIFY A FARTICULAR COPY ELEMENT OR 'ALL' .

THE RUN REQUESTS MODE, TYPE AND RID LOCATION OF THE RUN, MODE, TYPE AND RID LOCATION OF THE COPY LIBRARY AND THE COPY ELEMENT IDENTIFIER.

7.4. LIB-SYS

THIS RUN PERFORMS THE SAME FUNCTION AS LIB-CHG ON AN ENTIRE SYSTEM. IN ORDER TO DETERMINE THE RUNS WHICH BELONG TO THE SPECIFIED SYSTEM AND THEIR LOCATION, THE RUN ACCESSES THE RUN INDEX MAINTAINED BY RUN-ADMIN. THE USER MAY SPECIFY A PARTICULAR COPY ELEMENT OR 'ALL' . THE RUN ALLOWS THE USER TO DECIDE FOR EACH RUN WHETHER IT SHOULD BE PROCESSED OR NOT.

THE RUN REQUESTS THE SYSTEM NAME, MODE AND TYPE WHERE THE RUNS ARE STORED, MODE, TYPE AND RID LOCATION OF THE RUN INDEX, THE COPY ELEMENT IDENTIFIER, AND MODE, TYPE AND RID LOCATION OF THE COPY LIBRARY.

7.5. DITTO

A RUN WHICH IS USED TO DUPLICATE OR MOVE LINES OF CODING FROM A SPECIFIED RID TO A SPECIFIED POSITION IN THE RID ON DISPLAY WHEN THE UTILITY IS CALLED. WHEN THE M OPTION (M = MOVE) IS USED, THE LINES ARE DELETED FROM THEIR ORIGINAL LOCATION.

THE RUN REQUESTS THE STARTING LINE, NUMBER OF LINES, THE LINE AFTER WHICH THE LINES ARE TO BE INSERTED, THE OPTION, AND THE TYPE AND RID NUMBER WHERE THE LINES ARE COMMING FROM.

7.6. RUN-ADMIN

THE PURPOSE OF THIS RUN HAS BEEN DISCUSSED EXTENSIVELY IN SECTION 5 OF THIS DOCUMENT.

7.7. $SYSUPD

$SYSUPD HAS BEEN PROVIDED TO ADD A RECORD TO THE SYSTEM INDEX FOR A NEW APPLICATION SYSTEM. $SYSUPD PROVIDES THE CAPABILITY TO CREATE THE RID HEADERS FOR EACH OF THE FOLLOWING STRUCTURES WHICH ARE REQUIRED TO DOCUMENT APPLICATION SYSTEM WITH THE SDA UTILITIES:

PROGRAM INDEX
DATA STRUCTURE INDEX
SPECIFICATION INDEX
DATA DICTIONARY
ERROR TABLE

7.8. $DDUPD

$DDUPD HAS BEEN PROVIDED TO FACILITATE THE ADDITION, MODIFICATION OR DELETION OF ELEMENTS INTO THE DATA DICTIONARY. A TRANSACTION SKELETON IS PROVIDED TO PROMOTE CONSISTENT DATA ELEMENT FORMATTING.

COMMENT LINES MAY BE ADDED TO OR DELETED FROM THE TRANSACTION SKELETON AS NEEDED.

7.9. $RECDEF

$RECDEF HAS BEEN PROVIDED TO GENERATE A RECORD DESCRIPTION FROM THE DATA ELEMENTS IN THE DATA DICTIONARY. DATA ELEMENTS ARE SELECTED FROM THE DATA DICTIONARY BY DATA DICTIONARY NUMBERS WHICH ARE ENTERED INTO A DISPLAY PROVIDED BY $RECDEF. $RECDEF MAINTAINS A DATA STRUCTURE

INDEX WHICH PROVIDES REFERENCES TO THE RIDS WHERE THE RECORD DESCRIPTIONS ARE STORED.

7.10. $DATAXREF

$DATAXREF PREPARES A DATA ELEMENT CROSS REFERENCE LISTING, SHOWING ALL RECORDS WHICH CONTAIN A SPECIFIED DATA ELEMENT.

7.11. $SPECPROG

THE $SPECPROG RUN PROVIDES THE SYSTEM DESIGNER WITH A TOOL TO FACILITATE THE DEVELOPMENT AND MAINTENANCE OF THE PROGRAM SPECIFICATION. OPTIONS ARE PROVIDED TO GENERATE SCREEN DESCRIPTION MASTER, GTOC MASTERS, PROCESS SPECIFICATION MASTERS AND HIPO MASTERS FOR INCLUSION IN THE PROGRAM STRUCTURE INDEX.