HANDOUT 19 [illegible]

SECURITY GUIDELINES FOR COMSEC SOFTWARE DEVELOPMENT

THE MILITARY COMMITTEE

COMMUNICATIONS SECURITY AND EVALUATION AGENCY

(SECAN)

Washington, D.C.

January 1988

CONTENTS

1. Purpose

This document, SECAN Data Requirement 3 (SECAN DR-3), specifies life cycle security guidelines for the development of COMSEC software. These guidelines are for use in preparing data to be provided in satisfaction of SECAN DR-2.

2. Scope

These guidelines are generic in the sense that they apply to all COMSEC software system developments. They include minimum design and implementation requirements which, in unusual or high risk software development environments, may be complemented by specific software security requirements. It should be noted that even rigorous compliance with all specified requirements and goals does not guarantee official "certification", the greater the degree of compliance, the more assessable, and hence certifiable, the software becomes. If a developer believes that any guidance provided herein is inappropriate, he should apply for a case-by-case exemption.

3. Control

This document, in its entirety, is NATO UNCLASSIFIED.

4. Glossary

<u>Alias</u>
An alternative name for an identifier established via equate, equivalence, common, or similar mechanisms. Aliased identifiers share the same memory location(s).

<u>Assessable</u>
Capable of being evaluated by a person with average knowledge of the technology involved. Characteristics of assessability include:
- o Specifications (the lowest level of which is the code itself) are traceable to requirements.
- o Requirements are testable.
- o Software modules have high cohesion, low connectivity, and low complexity.

<u>Certification</u>
The process of review leading to confirmation that

software:

o Performs its intended COMSEC functions in a secure fashion.

o Performs no unintended, security detrimental functions.

It is a technical process that produces a corporative judgement or statement of opinion.

Closed security environment

An environment in which both of the following conditions hold:

o Applications developers have sufficient clearances and authorizations to provide an acceptable presumption that they have not introduced malicious logic into the design or operation of a secure system.

o Configuration control provides sufficient assurance that applications are protected against the introduction of malicious logic prior to and during the operation of system applications.

COMSEC

COMmunications SECurity.

COMSEC algorithm

A procedure which performs encryption, decryption, or authentication.

COMSEC boundary

That minimal collection of all hardware/firmware/software whose correct operation is necessary in order to meet the security requirements.

COMSEC control software

Any software/firmware which directs, monitors, invokes, or otherwise interfaces the application system to the COMSEC function.

COMSEC function

Any operation whose purpose is to provide secure communications through the use of COMSEC algorithms or other signal conversions (e.g., spread spectrum). Also included are any operations which, although not having overtly secure communications functions, can cause the compromise of sensitive data, with a resulting adverse COMSEC impact, if not correctly performed.

COMSEC software

All software/firmware which implements, or contributes to the correct operation of, a COMSEC function.

Critical routines
Any collection of software/firmware whose correct operation is necessary in order to meet the security requirements.

Critical software checks
Those checks which directly guard against conditions which can lead to a violation of functional requirements.

Firmware
Software residing in read-only memory.

Software Security Guidelines
The Security Guidelines for COMSEC Software Development, i.e. SECAN DR-3.

Storage class
An attribute associated with a variable which determines the location and lifetime of the variable's storage. Examples are static, external, and automatic.

Thread
A continuous and traceable data and/or control flow path description of some system function, with corresponding documentation.

Validation
To establish the fitness of software for its operational mission; informally, is the right job being done?

Verification
To establish the correspondence between higher level and lower level software specifications; informally, is the job being done right?

5. Discussion

In recent years, the use of general purpose, programmable computers and microprocessors to perform COMSEC functions has become increasingly common. This approach to the design of secure COMSEC devices opens new security concerns not present in special purpose hardware implementations. The major new difficulty stems from embedding software in a possibly preexisting environment containing non-COMSEC related hardware and software. Often this environment has been designed without security as a goal, or with performance requirements that work counter to security requirements. The guidelines given here are meant to apply to a range of software development environments, but have reduced effectiveness in general purpose environments.

5.1 Designer Tasks

The COMSEC software designer has two tasks: design software that accurately performs its intended functions, and provide conclusive evidence which will allow certification personnel to easily confirm that the software performs only its intended functions.

5.1.1 Performance Task

The first task requires that the software designer understand the security requirements to be implemented in software. He must design and implement COMSEC software that performs only its intended cryptographic functions, and provide safeguards which ensure that the COMSEC software cannot be circumvented, modified, or degraded. This is a performance task. Of major concern is proper performance of all cryptographic functions, and correct handling of potential hardware faults. Although there should be no software faults, a good design will also check for potential software errors.

5.1.2 Assurance Task

The second task is one of assurance. The design must provide simple, clear, incontrovertible evidence that COMSEC software cannot be bypassed, modified, misused, or in any way compromised due to the effects of other system components (or even the COMSEC software itself). Key to providing assurance is process isolation. Isolation may be accomplished through physical and logical (program logic) means. Generally, physical isolation provides the greater assurance of correct COMSEC operation. Physical isolation may be accomplished through a combination of

hardware features, such as memory mapping circuits, special access circuitry, privileged or supervisor states, separate processors and busses, and memory tagging hardware. Logical isolation may be accomplished through software features such as security kernels, structured topdown software design, use of modern high-order languages (HOL's) such as Ada, modularity, and proper programming techniques in the handling of interrupt processing and I/O operations. The goal of process isolation is to reduce the totality of software dealing with critical COMSEC functions (and consequently reduce the totality of software that needs to undergo COMSEC analysis) to a manageable level. What is manageable is subjective, but is generally accepted to be on the order of 4000 statements. With the use of formal development methodologies, this may be expanded depending on the level of abstraction employed. Equally important in providing assurance is documentation. A simple, clear, and convincing argument must be presented which describes how and why the design achieves COMSEC integrity. The documentation must allow for an easy, independent evaluation and verification of all claims.

5.2 Operating Environment

The operating environment in which the COMSEC software is embedded (that is, all software and hardware upon which it relies for correct operation) will have a significant impact on the integrity of the COMSEC software. These range from dedicated stand alone custom designed microprocessor circuits, to COMSEC functions implemented on multiuser mainframe computers. Listed in the order of decreasing COMSEC potential are samples of environments in which COMSEC software could be embedded:

5.2.1 Dedicated Custom Hardware

In this environment, the COMSEC software will be operating on dedicated custom hardware. The COMSEC designer has total control over both the hardware and software. An example is the use of dedicated microprocessors or microcomputers which are custom designed for a specific COMSEC function. Security features can be implemented in both the hardware and software to ensure the integrity of the implemented COMSEC software.

5.2.2 Dedicated General Purpose Hardware

In this environment, the COMSEC software will be operating on general purpose hardware, such as a general purpose microprocessor board, personal computer, or perhaps a minicomputer. The COMSEC function, however, is the <u>only</u> process

for which the hardware is to be used. The COMSEC software is the only software running on the system, and even handles all normal operating system functions. The COMSEC designer has total control only over the software, not the hardware.

5.2.3 Dedicated Function on General Purpose Hardware

This is similar to the Dedicated General Purpose Hardware environment. It differs in that the COMSEC function is not the only function for which the hardware is to be used. However, when the COMSEC function is invoked, it assumes control of the entire machine, displacing the operating system, and/or any other software that may have been in use on the hardware at that time. This environment is not as controlled as those previously discussed. The COMSEC designer may have total control of the COMSEC software only when it is invoked. The designer must pay particular attention to the applications running <u>before</u> <u>and</u> <u>after</u> the COMSEC software executes.

5.2.4 COMSEC Function on General Purpose Hardware

This is the least controlled environment. COMSEC software is invoked and executed under the General Purpose Hardware's operating system. In some cases, this might even be a multiuser, multitasking environment. This means that the COMSEC designer has no control over the operating system, what functions have run before or after the COMSEC software, and what functions might be co-resident when the COMSEC software is executing.

6. Requirements and Guidelines

6.1 General

The development of COMSEC software consists of four major phases. The requirements phase, the architecture phase, the design phase, and the implementation phase. Dividing the development phases into separate functions (with perhaps different personnel, etc.) should be considered as contributing to assurance. Details which bind a design to a specific implementation should be postponed as long as possible. Concurrent with all four phases are three major activities: COMSEC analysis, quality assurance, and configuration control. These latter two activities continue over the entire life cycle of the COMSEC product.

a. Requirements Phase: During this phase, the technical requirements for both the software, and the hardware on which it will operate must be determined. These requirements must include the security requirements for the software. It is extremely important that during this phase the set of security requirements and the rationale for their choice be clearly defined and analyzed to determine the strengths and weaknesses of the proposed system.

b. Architecture Phase: In this phase, the security requirements are used to develop a security architecture and to determine a set of security features. Specific attention must be given to the isolation of the COMSEC functions from other system functions. The security architecture must address both the operating and physical environments for software development. This includes the system in which the software is to be embedded, the security of the development facility, personnel clearances, the testing plans, quality assurance plans, and configuration control plans.

c. Design Phase: This phase is a refinement of the architecture phase, where functionality becomes apparent. In addition to designing functions which implement the COMSEC functions, mechanisms must be designed which ensure the integrity of the COMSEC functions. At this point, the quality of the security functions can begin to be analyzed. An evaluation of the strengths and weaknesses of candidate programming languages should be performed in the context of candidate hardware and requisite security

mechanisms.

d. Implementation Phase: During this phase, the developer must implement the functions to be used to provide the security features described in the design phase. The programmer will take the COMSEC functions specified using the formal specification or program design language and code them using either a highorder language, or in certain cases, an assembly language.

6.2 Minimum Essential Requirements

This section provides both requirements and goals, grouped by phase and activity. Additional standards and/or guidelines may be specified, depending upon the application. All requested information and documentation must be supplied in the requested/required format, unless a convincing argument can be presented why the required document, information, or format is inappropriate, unnecessary, redundant, or available in another manner acceptable to SECAN.

6.2.1 Requirements Phase

During this phase all relevant regulations, standards, and manuals must be gathered and studied. In conjunction with the operational requirements for the system, these form the basis for all subsequent development. This is the proper point at which to clarify the interpretation of the guidance provided in this document, decide which requirements apply and, if the application warrants, obtain special case exemptions. Once the requirements are defined, they must be carefully tracked throughout the successive levels of system definition. At each of the major development stages/reviews, the mechanisms which fulfill each requirement must be documented. Where appropriate, specific modules and routines must be associated with the requirements which they support. Specific tests must be provided for each of the requirements as part of the final product acceptance testing.

6.2.1.1 Goals

The following references provide useful background reading in the spirit of these guidelines.

a. NATO Trusted Computer System Evaluation Criteria,AC/35-D/1012, dated 25 May 1987

b. Computer Security Requirements - Guidance for Applying the Department of Defense Trusted Computer System Evaluation Criteria in Specific Environments, CSC-STD-003-85, 25 June 1985.

6.2.2 Architecture Phase

During this phase, a plan must be developed which describes the strategy and approach to security certification. The software designer's two major tasks, performance and assurance, must be interpreted in view of all environmental constraints. When less controlled environments are used, more elaborate assurance techniques need to be employed. Success in this phase is measured by the ease with which the two tasks may be addressed and satisfied throughout the remainder of system development and use.

6.2.2.1 Requirements

a. A COMSEC boundary must be defined. That is to say, COMSEC functions and data must be physically or logically (through program logic) separated from non-COMSEC functions and data. The separation must isolate the COMSEC functions and data from all outside influence which may:

   (1) Modify either program code/data or correct performance of cryptographic functions.

   (2) Bypass cryptographic functions.

   (3) Compromise cryptographic functions or data (e.g., disclose program logic, key, subkey, key variables, plain text, etc.).

b. The COMSEC boundary must be verifiable. Documented, well-reasoned substantiation must be provided, including arguments which will convince one with average skill in the technology involved that the boundary is indeed as described. Care must be taken not to trust unverified claims concerning process and user isolation of commercial general purpose operating systems.

c. The totality of software and firmware within the COMSEC boundary must be assessable. It should be a small percentage of the overall system, and of low

complexity. It is unlikely that more than 4000 statements will be assessable.

d. For the COMSEC Function on General Purpose Hardware operating environment, the operating system and non-COMSEC software will be required to meet a level of trust as described in the NATO Trusted Computer System Evaluation Criteria. The exact level of trust required will be application dependent.

6.2.2.2 Goals

a. Hardware implementation of COMSEC functions.

b. Use of hardware to implement COMSEC features and/or enforce security requirements.

c. Use of previously certified components.

6.2.3 Design Phase

Having established the security strategy, the design phase must provide hardware components and program logic which are consistent with the overall strategy. It must address both performance and assurance. Reliability and failsafe performance must be assured. Redundant hardware, software checks, and hardware alarms may be employed to provide the requisite assurances. At this level, it is possible to enumerate specific software security requirements and goals.

6.2.3.1 Requirements

a. The COMSEC software must:

(1) Enforce or monitor correct performance of hardware and software within the COMSEC boundary; and

(2) be properly invoked.

b. Positive notification of a failed critical software check must be provided to the system operator. This may be in the form of a visual and/or audible indicator.

c. When a critical software check fails, all cryptographic operation must immediately cease and

remain inhibited until:

(1) The software check passes; and

(2) The system operator overtly causes cryptographic operation to resume. Automatic recovery from critical failures will be allowed only in exceptional circumstances and with explicit SECAN approval.

d. Critical routines must be checked before and after they are executed. This check can be the execution of a sequence check, known answer test, or other authentication technique to verify correct performance of both software and hardware.

e. All sumchecks used to verify the integrity of data in memory or being transferred from one medium to another must have a minimum of 16 bits precision, and be computed in a nonlinear fashion.

f. The integrity of critical routines must be checked by the performance of periodic sumchecks and CPU health checks.

g. Where alarms exist, either in hardware or software, their operation must be periodically verified.

h. The design must protect sensitive data, which may include part of the program itself, as well as key variables, intermediate computational states during encryption/decryption, etc.

i. Sensitive data must be stored in a minimal number of locations. Locations reserved for sensitive data must not be used for nonsensitive data. If possible, physical barriers should be employed to isolate sensitive from nonsensitive data to prevent inadvertent access.

j. Locations used for temporary storage of sensitive data must be overwritten after use and, if appropriate, prior to interrupt servicing. This may include stack locations. (In some cases, deleting a data set does not necessarily delete the original data from storage.)

k. Software must be written so that an unexpected interrupt or power failure does not result in a security weakness. The design should treat all

possible exceptional conditions and interrupt causes.

l. If possible, lock out interrupts while critical routines are being executed.

m. A known answer test must be executed immediately before and after an encryption/decryption function which is performed in software or firmware.

n. A formal development methodology (requirements/ specification language, verification system, and/or program design notation) must be used to describe the COMSEC functions.

o. The criteria and rationale for the selection of programming language(s) used must be supplied for SECAN review.

6.2.3.2 Goals

a. Special software checks should be used to detect not only hardware errors, but also operator errors and (consistent with design simplicity) software errors (bugs). That is to say, <u>defensive</u> and <u>robust</u> programming techniques should be used.

b. Memory health checks should be designed to detect the failure of an address or memory unit to arbitrary bit patterns over the full width of the address or memory word.

c. The format of inputs should be checked.

d. Modules should check that parameters passed to them have acceptable values. There should be as few permissible values as possible.

e. Flags can be used to check that code is executed in the correct order and that routines have been entered and exited at the proper places.

f. Count the number of loop iterations by more than one method, and check that they agree.

g. When locations have been overwritten with a fixed pattern, check, by reading from them, that overwriting was successful.

h. Check RAM for addressing failures by setting each location to a different value (e.g., its address) and then checking that every location contains the correct value. If there are more locations than possible values, then more than one check will be necessary.

i. Check RAM for memory errors by writing a fixed pattern in each location and then reading back the pattern. Repeat the test with various fixed patterns.

j. In addition to known answer tests, encryption/decryption routines may be checked by decrypting cipher and checking that the plain text is obtained. The encryption and decryption processes ideally should be independent (i.e., different but equivalent code). For two key (public) systems, repeat the operation using an independent method, and check that the two agree.

k. The number of modules which access sensitive data should be minimized.

l. Set a flag (in some cases, a nonvolatile flag) before a sensitive operation starts, and do not reset it until the operation is complete. Before another sensitive operation can begin, check to see that all critical flags are in the proper state.

m. Update all critical counters (volatile and non-volatile) before a sensitive operation takes place. If an interrupt or power failure causes the operation to be aborted, then the next attempt will start with the new counter settings, preventing repeats of critical operations. We assume here that the current value of the counter defines which operation needs to be performed next.

n. To minimize the effects of single bit memory failures, values assumed by a flag should differ in as many bits as possible (have good Hamming distances).

o. Storage areas for sensitive data should be chosen so that failure of a single address bit does not cause a security weakness.

p. Unused memory should be set to a value which would cause an alarm or system halt if executed.

q. Any formal methodology used should be machine based (i.e., automated/processable) and "standard" (i.e., independently developed, commercially available, etc.).

r. Thread or scenario descriptions for major system functions should be supplied or be readily obtainable.

6.2.4 Implementation Phase

There are two aspects that need to be addressed during the implementation phase, one dealing with hardware and the other with software. We present only goals for hardware, recognizing that choices may have already been limited due to strategies and approaches taken in the architecture and design phases. We state very strongly that these goals should be followed whenever possible, in order to reduce the complexity of the assurance task.

6.2.4.1 Requirements

6.2.4.1.1 General

a. The software must perform the COMSEC functions for which is was designed, and nothing else.

b. The software must fulfill all of the security requirements attendant (e.g., alarms, rekey, etc.) to the correct operation of the function.

c. A High-Order Language must be used for all COMSEC control software. All COMSEC algorithms must be encoded with assembly language. Only in specific, SECAN approved cases will exceptions be made.

d. Software inspections and/or walkthroughs must be a part of the software development methodology.

e. Structured/modular design, parameterization, use of reusable standard components, and other modern software engineering techniques must be employed.

f. "Unadvertised", undocumented, or anomalous features/behavior of the hardware and software systems employed must not be used.

6.2.4.1.2 Data Declaration

a. All variables must be explicitly declared.

b. Default attributes (including initialization) must be documented. This may be done for classes of variables if appropriate.

c. Variables which are intended to be local must have storage classes which do not conflict with their scope.

6.2.4.1.3 Data Reference

a. Each data item (variable, array element, structure field, etc.) reference must have a value intentionally set (initialization, assignment, I/O, argument/parameter, set via an alias) consistent with its attributes and in conformance with data storage conventions.

b. Array subscripts and string indices must be within defined bounds and have appropriate attributes.

c. Indexing expressions must be correct (proper type, not "off by one," appropriate for storage mechanism used, etc.).

d. All storage referenced by pointers or reference variables (addresses, etc.) must exist at the time of reference.

e. Data storage and formats must be consistent (type/alignment) when used in multiple contexts (aliased, pointer/address referenced, I/O, etc.).

f. Variables containing sensitive data must be reset to nonsensitive values before reuse or deallocation.

6.2.4.1.4 Computation

a. Variable ranges must be within nominal, as well as normal, bounds.

b. Intermediate expressions must not cause exceptions (overflow, division by zero, etc.) or unexpected

conversions. For example, (25 + 1/3) in PL/I will cause a fixed point overflow.

6.2.4.1.5 Interface

a. Arguments and parameters (or actual and formal parameters) must match in number, order, attributes and represented units system.

b. When using multiple entry points, only parameters associated with the current point of entry may be referenced (single entry/exit is preferred).

c. Parameters intended as input only must not be altered (e.g., using call-by-reference instead of call-by-value).

d. Global variables must have consistent declarations. However, their use should be minimized.

6.2.4.1.6 Control Flow

a. Unreachable code or variables declared/set but not used must be removed or made into comments.

b. Decision sequences must be exhaustive.

6.2.4.1.7 Input/Output

a. I/O data, data format, and target structure must be consistent.

(1) The data in memory or on an external medium must be represented in the proper bit level format (integer, floating point, ASCII, etc.).

(2) The format conversion performed upon the data must present it in a form expected by the internal variable or output medium.

b. Sumchecks must be performed on all data when transferring from one medium to another.

c. I/O conditions (end of file, transmission error, etc.) must be sensed, and appropriate actions explicitly provided for each error possibility.

6.2.4.1.8 Documentation

Unit level module documentation must include configuration information (version identification, date, author/point of contact, reason for update, etc.), and interface specifications.

6.2.4.1.9 Other Checks

a. Techniques which result in compiler/assembler diagnostic messages at an "error" or higher level must not be used.

b. Error conditions for which there are not explicit checks, must cause graceful degradation (via a general error handler, etc.).

c. During testing, critical software checks must be tested using values that exercise all decision outcomes. If this is not possible, the check must be identified and rationale given for the correct performance of the check.

6.2.4.2 Goals

6.2.4.2.1 General

a. Commercially available (independently developed) operating systems, language processors, development and analysis tools, and utilities should be used. "Standard" programming languages should be used when available.

b. No more than one programming language statement should appear on a text line unless clarity is enhanced by doing so.

c. Programming language features and compiler/ assembler options which assist in the detection of erroneous conditions during execution (such as array bounds checking) should be used, consistent with performance requirements.

6.2.4.2.2 Data Declaration

a. Variables should have the smallest scope consistent with their use, and should not be used for more than one purpose within the scope of their

declarations. An example of multiple purpose is using a variable as an integer in one context and simultaneously as a string in another.

(1) In assembly languages, avoid external variables when local variables could be used.

(2) In high order languages, avoid declaring variables in an outer block which are local to the inner block. Restrict the scope of temporary variables.

b. Variables should be properly initialized (including consistency with storage class).

c. Verify proper variable type, length, precision, dimension, storage class, etc.

d. Avoid variable names that are nonmnemonic, have confusingly similar names, or are reserved words.

e. Check structure/matching of initialization lists.

f. Avoid using constants in array dimensions, loop limits, and like situations where changes may occur through program maintenance.

6.2.4.2.3 Data Reference

a. Verify that variables of sizes smaller than units of storage addressability are accessed correctly (Bit variables are typically in this category).

b. Check for appropriate variable "finalization"; upon leaving the scope of definition of a variable, make sure that its final value is as desired.

c. Avoid inserting control or nonprintable characters directly into the source program. Use alias techniques (numeric values, etc.) for such characters.

6.2.4.2.4 Computation

a. Inconsistent expressions (mixed base/scale/ precision mode and/or nonarithmetic operands) should be avoided.

b. Check for type conversion problems at intermediate stages of computations, assignment statements, I/O operations, etc.

c. Check for peculiarities of integer arithmetic.

(1) For example, does the identity (A/B) = (A/B) = (A)/B = A/(B) hold?

(2) What is the range of integer representation?

(3) Are loop counters implemented using integer arithmetic?

d. Verify grouping of subexpressions.

e. Units and sign convention should be checked (arithmetic base, etc.).

f. Verify operator precedence and order of evaluation in complex expressions.

g. The most natural operation should be used within the context of the program logic; e.g., when doing arithmetic calculations, do not "arithmetic left shift X by 2" in place of "X*4".

h. Self-modifying code should be avoided. If used, each case must be justified and well documented.

6.2.4.2.5 Interface

a. Check for implied use of dummy arguments. Call by value or expression arguments usually imply dummy actual arguments. This is of possible security concern for duplicated storage of classified information.

b. Check validity/consistency of multiple returns (value/no value, alternate, etc.).

c. Modules should check input for validity.

d. Adhere to standard calling conventions.

(1) Do not pass arguments in registers, thus bypassing the formal argument list.

(2) Do not make use of results resident in

operating registers or intermediate variables upon subroutine return.

6.2.4.2.6 Control Flow

a. Every loop (explicit and implicit) should have explicit termination condition documentation. This can be accomplished by inserting a comment describing the intended exit conditions.

b. Verify that consequences of loop exit or "fallthrough" are as intended.

c. Verify loop iteration counts.

d. Verify correct matching of statement groupings (DO END, IFTHENELSE, etc.).

e. Check logical expressions for correct comparison operators, subexpression grouping, order of evaluation, unexpected conversions, etc.

f. Branches should not be made conditional on the value of sensitive data, because:

(1) If a hardware error causes the branch to be taken to the wrong address, then the data might be compromised.

(2) The branch could lead to differences in the timing of output and hence convey information about the data.

g. Sensitive data should not be kept in operating registers. To the maximum extent possible, operations should be used which reference sensitive data while in memory. For example, instead of loading the accumulator with sensitive data and then adding "immediate" or "through memory" to non sensitive data, load the nonsensitive data, and add the sensitive data.

h. When sensitive data is kept in a permanent store, the store should be sumchecked, or compared with an independent copy, before data is used.

i. When feasible, hardware memory mapping circuitry should be used in order to isolate sensitive code and data. In the absence of suitable hardware

memory mapping circuitry, sensitive code may be isolated by storing it in an area where it cannot be executed and copying it to another area to execute. (The copy should be erased immediately after use.) This will decrease the chance that the code will be executed unintentionally.

6.2.4.2.7 Input/Output

a. Files implicitly declared should not be explicitly declared unless attributes are being changed.

b. Check for correct sequencing and span of open, reference, and close file operations.

c. Verify correspondence and consistency of data and format items.

d. Check for missing and/or incorrect input.

e. Check that data area(s) for I/O transmission are consistent with record size, format, etc.

6.2.4.2.8 Documentation

a. Highlight (via comments) labels used as transfer points (branch targets).

b. Insert commented references to specifications where appropriate.

c. Make commented assertions about relations, conditions, etc., in addition to normal descriptive comments.

d. Unit development folders should be employed, and be available for SECAN inspection.

e. Important software development information, such as unit development folders and test data sets, which are not normally a part of deliverable documents, should be archived by the developer for a period of one year after acceptance or for the software warranty period, whichever is longer.

6.2.4.2.9 Other Checks

a. Check compiler/assembler cross reference listings for identifiers not referenced or referenced only once.

b. Check compiler/assembler attribute listing for consistency with intended attributes.

c. Check for potential portability difficulties, which may indicate poor technique.

d. Duplication of code should be avoided, consistent with clarity.

e. Unit testing should include the evaluation of all linearly independent control paths (this implies the execution of all statements at least once).

6.2.4.2.10 Hardware for Critical Operations

a. Duplicate (redundant) processors with comparators (at least data and address bits) should be used.

b. Processors should operate in privileged (supervisor) and user hardware states. More states may be useful in some applications.

c. Memory mapping units should be used. These units should control read/write/execute privileges based upon user hardware state.

d. All memory should contain at least single bit error correction and double bit error detection checks.

e. COMSEC program code and static tables should be contained in and executed from read-only memory.

6.2.5 Analysis Activity

Each phase of the development process will be analyzed to assure that the two tasks of the COMSEC software designer are being achieved. This analysis will not be a separate phase, but an ongoing effort, commencing with the requirements phase. The analysis will be systematic and thorough. The security analysis is in addition to the typical Software Quality Assurance (SQA), and Independent Verification and Validation (IV&V) efforts, although these efforts may indeed be a significant input to the security analysis effort.

a. The requirements analysis will ensure that all of the security requirements are included and appropriately addressed.

b. The architecture analysis includes a review of the COMSEC isolation approach and the security of the development facility, with special attention to the integrity of software while under development.

c. The design analysis will review the correctness, completeness, and consistency of the COMSEC software functions. The implications of the choice of hardware and programming language(s) are examined.

d. The implementation analysis includes review of the actual implementation of the software and the security features embodied in that software.

6.2.5.1 Requirements

a. The following documentation must be provided to SECAN for review. When in conflict, the requirements of this document shall prevail.

(1) Software Software Product.

(2) Software Program End Item.

(3) Software Hierarchical Structure/Attribute List.

(4) A correspondence mapping, which traces each security-related system functional requirement through every level of specification, down to the actual source code which satisfies the requirement. Each requirement should map to as few software entities (subprograms, routines, units, etc.) as possible.

(5) Hardware Reference Manual (commercial product or equivalent must include timing information).

(6) Programmer's Reference Manual (commercial product or equivalent).

(7) Software Language Reference Manual(s) (commercial product or equivalent).

(8) Software End-Product Acceptance Plan

(9) Data Dictionary Document.

(10) Technical Reports - Independent Software Verification and Validation Report.

(11) A "Theory of Compliance" document, demonstrating how the software meets requirements.

b. There must be an independent software verification and validation effort, addressing at least all software within the COMSEC boundary, managed by the developer in parallel with the software development. Care must be taken to assure that the effort is truly independent of the development effort, and that reporting channels are set up to keep SECAN apprised of the developer's conformance to the guidelines provided herein. The developer shall deliver to SECAN a Software Independent Verification and Validation Report, which details all evidence and rationale contributing to the <u>assurance</u> task as described earlier in the Discussion section of this document. This report shall be updated at each formal review or major project milestone. In addition to this effort, there will also be a SECAN evaluation, which will be the basis for final certification.

6.2.5.2 Goals

a. There should be either periodic deliveries of interim software or dumps of the software development facility prior to final delivery to include:

(1) Program design notation specifications.

(2) Source code with:

(a) Identifier attributes listing (declared, initialized, aliased, argument, parameter, assigned, I/O, referenced, etc.).

(b) Control flow connectivity matrix.

(c) External data flow connectivity matrix.

(d) "Inverse" data dictionary ("cross

reference" list; places of use for each identifier).

6.2.6 Configuration Control Activity

In order to preserve security while under development and throughout the life of the system, a complete and accurate master copy of all software and documentation must be maintained. Not only must the software under development be controlled, but so also should the development facility, personnel, tools, compilers, assemblers, preprocessors, and operating system software used on the software development system.

6.2.6.1 Requirements

a. Software modifications to correct errors or provide system updates must be evaluated for their impact upon security. Changes which involve security related code must receive particular attention. Changes to security related code must be analyzed for their impact upon the system and to ensure that other problems are not introduced. These changes must be evaluated by testing to ensure that they perform as intended.

b. A formal configuration control system must be established. Configuration control procedures must be instituted at the inception of a new system development. These procedures must be documented in a Configuration Control Plan. This plan includes a description of the internal procedures for controlling software configuration.

c. The hardware and software used as tools in the development of a product are important to the maintenance of the security integrity of that product. The hardware configuration used must be controlled during the life cycle of the product being developed to prevent the introduction of flaws. Similarly, the software tools used to develop a product, including operating systems, editors, compilers, assemblers, linkers, etc., must also be controlled during the product life cycle.

d. Access must be limited to prevent the unauthorized release of information or the unauthorized modification of security related data, software, and equipment. Physical control of the development

facility is necessary, including techniques to control remote access. When a general purpose computer is shared among numerous groups, techniques must be used to provide internal access control. Such techniques may include periods processing (which limits physical and logical access to system resources while sensitive data is being processed), system passwords, and file protection mechanisms.

e. All critical COMSEC software must be developed and maintained by people cleared at least NATO SECRET in closed security environments.

f. All classified application software must be developed in its entirety within secure facilities.

g. All operational software must be placed under configuration control from the beginning of development, or at its time of integration for software not specifically developed for this project.

h. Baseline documentation which describes product design and operation must be controlled. This documentation includes approved descriptions of requirements, system architecture, software source listings, operating instructions, etc.

i. Any files which contain approved executable code to be loaded into memory or burned into ROMs must be controlled. Since it is this information which represents the final product, its integrity must be absolutely assured. A control version of this information must be maintained as a reference.

j. Software change proposals must be used to maintain strict accountability of software changes. A record of each fielded system configuration must be maintained along with the documentation of each system modification.

k. Software updates must be evaluated by performing system tests. The test sets and configurations which are used must be maintained under configuration control to ensure that correct system performance is maintained. Any changes may require reverification, revalidation, and/or redoing formal derivations.

l. If the operational requirements of the system

permit, all software development tools (editors, debuggers, etc.) must be purged from operational software to minimize the potential for unauthorized modification of operational software by operating personnel.

6.2.6.2 Goals

a. Periodic comparisons of fielded software to the controlled reference should be performed to assure the continued integrity of the security mechanisms.

b. A mechanism should be established to solicit problem reports from users of the product. This information should be used to investigate potential security problems. If practical, the disposition of each report should be recorded.

c. Information should be maintained on the authorized users of any product sold. This information should be used to inform users of potential problems and to provide system update information when available.

7. Minimal Requirements for Existing Software

The following guidelines are intended for software that was not developed under the formal Software Security Guidelines. It is recognized that, due to various constraints, not all of them may be applicable. Failure to supply necessary information will delay analysis and evaluation efforts, and possibly jeopardize certification.

a. A formal, written response to the Security Guidelines for COMSEC Software Development, providing an assessment of the extent to which specific areas are addressed.

b. A Descriptive Top Level Specification, identifying all security related and COMSEC related software, preferably in a machine based program design notation.

c. A detailed description of how the hardware and/or software supports security/COMSEC boundary isolation.

d. A correspondence mapping of security related specifications to source code.

e. A Software Hierarchical Structure Attribute List, identifying all unit level source code and associated security related characteristics (supervisor/user state, isolation mechanism, etc.).

f. Convenient access to all Quality Assurance and Verification and Validation reports on security related components.

g. A description of Quality Assurance, Configuration Management, and Integration and Test plans and procedures.

h. All software development tools, including language processors, utilities, and operating systems, as well as source code, must be deliverable in a readily machine readable magnetic form.

i. Unrestricted, convenient access to all developer internal hardware and software development systems (including data bases, libraries, and test beds).

8. Compliance Checklist

Tailoring these guidelines to a specific application is an essential feature of compliance, and is the joint responsibility of the software developer and the host nation. The following requirements (R) and goals (G) index can be used to make sure all relevant guidelines have been addressed, and indicate applicability and degree of compliance with the referenced items. A completed checklist can be used as an initial and/or summary response to the Guidelines.

8.1 Status Indication

For each item, the developer must indicate actual or intended status:

- Not Applicable (N)
- Not Required ()
- Noncompliance (X)
- Partial Compliance (P)
- Compliance (C)
- Pending/Unknown/etc. (?)

Items for which compliance (C) is not indicated must have supplemental documentation explaining why. Items indicating compliance must include a brief description of how the requirement is or will be met.

8.2 Tailored Requirements

After SECAN agreement is reached on the status of each checklist item, the tailored guidelines will then be considered requirements against which delivered software will be evaluated.

8.3 Checklist

General

R 6.2	Documentation content/format

Requirements Phase

G 6.2.1.1.a	AC/35-D/1012
G 6.2.1.1.b	CSC-STD-003-85

Architecture Phase

R 6.2.2.1.a	Define COMSEC boundary
R 6.2.2.1.b	Verify COMSEC boundary
R 6.2.2.1.c	Assessable software/firmware
R 6.2.2.1.d	Trusted computer system level
G 6.2.2.2.a	Hardware COMSEC functions
G 6.2.2.2.b	Hardware COMSEC features
G 6.2.2.2.c	Reuse certified components

Design Phase

R 6.2.3.1.a	Correct performance/invocation
R 6.2.3.1.b	Failure notification
R 6.2.3.1.c	Critical software check failure
R 6.2.3.1.d	Critical routine correctness check
R 6.2.3.1.e	Data integrity sumchecks
R 6.2.3.1.f	Critical routine integrity check
R 6.2.3.1.g	Periodic alarm verification
R 6.2.3.1.h	Protect sensitive data
R 6.2.3.1.i	Minimize/isolate sensitive data
R 6.2.3.1.j	Temporary storage of sensitive data
R 6.2.3.1.k	Interrupts/exceptions
R 6.2.3.1.l	Interrupt lock out
R 6.2.3.1.m	Known answer test
R 6.2.3.1.n	Formal development methodology
R 6.2.3.1.o	Language criteria and rationale
G 6.2.3.2.a	Defensive/robust programming
G 6.2.3.2.b	Memory health test check groups
G 6.2.3.2.c	Check input format
G 6.2.3.2.d	Parameter check
G 6.2.3.2.e	Flags
G 6.2.3.2.f	Loop iterations
G 6.2.3.2.g	Overwriting
G 6.2.3.2.h	RAM addressing failure check
G 6.2.3.2.i	RAM memory error check
G 6.2.3.2.j	Encryption/decryption check
G 6.2.3.2.k	Minimize sensitive data access
G 6.2.3.2.l	Sequence flags
G 6.2.3.2.m	Sequence counts
G 6.2.3.2.n	Flag values
G 6.2.3.2.o	Storage security for sensitive data
G 6.2.3.2.p	Unused memory
G 6.2.3.2.q	Automated/standard methodology
G 6.2.3.2.r	Thread/scenario descriptions

Implementation Phase

R 6.2.4.1.1.a	Performance only as designed
R 6.2.4.1.1.b	Attendant security requirements
R 6.2.4.1.1.c	HOL for COMSEC control software
R 6.2.4.1.1.d	Inspections/walkthroughs
R 6.2.4.1.1.e	Modern SW engineering techniques
R 6.2.4.1.1.f	Anomalous HW/SW features/behavior
R 6.2.4.1.2.a	Explicit variable declaration
R 6.2.4.1.2.b	Documented default attributes
R 6.2.4.1.2.c	Local variables
R 6.2.4.1.3.a	Data item value consistency
R 6.2.4.1.3.b	Subscripts/indices
R 6.2.4.1.3.c	Indexing
R 6.2.4.1.3.d	Pointer references
R 6.2.4.1.3.e	Data/format context consistency
R 6.2.4.1.3.f	Variable reset/reuse
R 6.2.4.1.4.a	Variable ranges
R 6.2.4.1.4.b	Intermediate expressions
R 6.2.4.1.5.a	Arguments/parameters
R 6.2.4.1.5.b	Entry/exit
R 6.2.4.1.5.c	Input parameters
R 6.2.4.1.5.d	Global variables
R 6.2.4.1.6.a	Unreachable/unused code
R 6.2.4.1.6.b	Exhaustive decision sequences
R 6.2.4.1.7.a	I/O consistency
R 6.2.4.1.7.a.(1)	Proper bit level representation
R 6.2.4.1.7.a.(2)	Format conversion
R 6.2.4.1.7.b	Data transfer sumchecks
R 6.2.4.1.7.c	I/O conditions
R 6.2.4.1.8	Unit level module documentation
R 6.2.4.1.9.a	Language processor diagnostics
R 6.2.4.1.9.b	Graceful degradation
R 6.2.4.1.9.c	Critical software check testing
G 6.2.4.2.1.a	Commercial availability
G 6.2.4.2.1.b	Text lines
G 6.2.4.2.1.c	Executiontime diagnostics
G 6.2.4.2.2.a	Variable scope
G 6.2.4.2.2.a.(1)	External assembly variables
G 6.2.4.2.2.a.(2)	HOL local variables in outer blocks
G 6.2.4.2.2.b	Initialization
G 6.2.4.2.2.c	Verify attributes
G 6.2.4.2.2.d	Variable names
G 6.2.4.2.2.e	Initialization lists
G 6.2.4.2.2.f	Avoid constants
G 6.2.4.2.3.a	Storage addressability
G 6.2.4.2.3.b	Finalization
G 6.2.4.2.3.c	Nonprintable or control characters
G 6.2.4.2.4.a	Inconsistent expressions

G 6.2.4.2.4.b	Type conversion
G 6.2.4.2.4.c	Integer arithmetic
G 6.2.4.2.4.c.(1)	A/B = A/B = A/B = A/B?
G 6.2.4.2.4.c.(2)	Range of integer representation?
G 6.2.4.2.4.c.(3)	Integer loop counters?
G 6.2.4.2.4.d	Subexpression grouping
G 6.2.4.2.4.e	Unit/sign convention
G 6.2.4.2.4.f	Operator precedence
G 6.2.4.2.4.g	Natural operations
G 6.2.4.2.4.h	Selfmodifying code
G 6.2.4.2.5.a	Dummy arguments
G 6.2.4.2.5.b	Multiple returns
G 6.2.4.2.5.c	Input validity
G 6.2.4.2.5.d	Standard calling conventions
G 6.2.4.2.5.d.(1)	Don't pass arguments in registers
G 6.2.4.2.5.d.(2)	Avoid register/intermediate values
G 6.2.4.2.6.a	Loop termination documentation
G 6.2.4.2.6.b	Loop exit
G 6.2.4.2.6.c	Verify loop iteration counts
G 6.2.4.2.6.d	Statement grouping matching
G 6.2.4.2.6.e	Logical expressions
G 6.2.4.2.6.f	Branches
G 6.2.4.2.6.g	Sensitive data in registers
G 6.2.4.2.6.h	Sensitive data integrity
G 6.2.4.2.6.i	Hardware memory mapping
G 6.2.4.2.7.a	File declaration
G 6.2.4.2.7.b	File operation sequencing
G 6.2.4.2.7.c	Data and format consistency
G 6.2.4.2.7.d	Missing/incorrect input
G 6.2.4.2.7.e	Data area consistency
G 6.2.4.2.8.a	Highlight transfer point labels
G 6.2.4.2.8.b	Commented references to specs.
G 6.2.4.2.8.c	Commented assertions
G 6.2.4.2.8.d	Unit development folders
G 6.2.4.2.8.e	Information archiving by developer
G 6.2.4.2.9.a	Reference listing
G 6.2.4.2.9.b	Attribute listing
G 6.2.4.2.9.c	Portability
G 6.2.4.2.9.d	Code duplication
G 6.2.4.2.9.e	Unit control path testing
G 6.2.4.2.10.a	Duplicate processors
G 6.2.4.2.10.b	Processor states
G 6.2.4.2.10.c	Memory mapping
G 6.2.4.2.10.d	Memory errors
G 6.2.4.2.10.e	COMSEC code/tables in ROM

Analysis Activity

R 6.2.5.1.a	Precedence of requirements

R 6.2.5.1.a.(1)	Computer Software Product
R 6.2.5.1.a.(2)	Computer Product End Item
R 6.2.5.1.a.(3)	SW Hierarchical Attribute List
R 6.2.5.1.a.(4)	Requirements/specifications mapping
R 6.2.5.1.a.(5)	Hardware Reference Manual
R 6.2.5.1.a.(6)	Programmer's Reference Manual
R 6.2.5.1.a.(7)	Software Language Manuals
R 6.2.5.1.a.(8)	SW End Product Acceptance Plan
R 6.2.5.1.a.(9)	Data Dictionary Document
R 6.2.5.1.a.(10)	IV&V Technical Report
R 6.2.5.1.a.(11)	Theory of Compliance
R 6.2.5.1.b	Independent verification/validation
G 6.2.5.2.a	Periodic delivery/dumps
G 6.2.5.2.a.(1)	PDN specifications
G 6.2.5.2.a.(2).(a)	Attribute listing
G 6.2.5.2.a.(2).(b)	Control flow connectivity matrix
G 6.2.5.2.a.(2).(c)	External data flow matrix
G 6.2.5.2.a.(2).(d)	Inverse data dictionary

Configuration Control Activity

R 6.2.6.1.a	Change evaluation
R 6.2.6.1.b	Configuration control system
R 6.2.6.1.c	Hardware/software tool control
R 6.2.6.1.d	Access limitation
R 6.2.6.1.e	Closed security environment
R 6.2.6.1.f	Secure facilities
R 6.2.6.1.g	Operational software control
R 6.2.6.1.h	Baseline documentation control
R 6.2.6.1.i	Executable code control
R 6.2.6.1.j	Change accountability
R 6.2.6.1.k	Update evaluation
R 6.2.6.1.l	Purging of development tools
G 6.2.6.2.a	Fielded software integrity
G 6.2.6.2.b	Problem reports
G 6.2.6.2.c	Authorized user list

9. Reference

a. C-M(55) 15(Final).

b. AC/35-D/1012, NATO Trusted Computer System Evaluation Criteria, Dated 25 May 1987

BRUK

AV

PCer ved IK

24-08-87 JJ, TVO

Rev 14-09-87 ER
Rev 16-11-87 ER,JJ

INNHOLDSFORTEGNELSE

1.Innledning

Dette notatet er et resultat av arbeide som er gjort for å samordne PC-aktivitetene ved IK. Arbeidet ble igangsatt etter å ha erfart følgende:

- Stadig flere PCer av ulike varianter dukket opp i huset.

- Fra å være innstallasjoner for bestemte formål har PCer utviklet seg til verktøy på linje med slegge og spett.

- Ingen har oversikt over hvilken programvare og maskinvare som allerede er kjøpt inn.

- Kunnskaper og erfaring omkring produkter og metoder blir ikke systematisk akkumulert.

- Driften av alle PCene krever totalt stadig større innsats.

For å bøte på dette forsøker en nå å samordne aktivitetene. Vi har kokt av arbeide som er gjort av Hydrac og resultatet ventes å bli effektivisering av driftsaktivitetene og gladere brukere.

ER er ansvarlig for dette notatet. En anmoder om skriftlige reaksjoner. Jeg beklager det tvilsomme språket nedenfor: I utgangspunktet er notatet Norsk, men jeg benytter engelske ord uhemmet der dette gjør teksten mer forståelig.

2.Sammendrag

Notatet består av to deler: Organisasjon og PC konfigurering. I første kapittel beskrives en organisasjon som skal yte tjenester til brukerene og pålegge restriksjoner. Innkjøp, opplæring, innstallasjon og sikkerhetskopiering av innkjøpt programvare er her sentrale emner.

Resten av notatet beskriver filstruktur, batch filer, enviroment variabler og annet som skal standardiseres. Filstruktur deler opp systemet i utviklingsverktøy, tekstbehandlere, andre programpakker og brukerfiler. Brukerkatalogene deles også opp etter hvilke programpakker som benytter dem.

Batchfilene inneholder menysystem og filkataloghåndtering og skal bevirke at brukerene skal slippe å plages med installasjonsdetaljer.

3.Organisasjon

Hele PC-organisasjon vil bestå av to deler: en person som er utpekt som ansvarlig koordinator med nødvendig myndighet (heretter kalt PCguru) og alle oss andre. Jeg vil beskrive organiseringen ved å følge en programpakke fra kremmerens disk til den støver ned i hylla hos oss:

3.1.Innkjøp

Et innkjøp begynner alltid med et behov(?), og er derfor noe som kan starte hos hvem som helst. Det er ikke praktisk å sentralisere noe særlig grad av kunnskap om produkter og produktnyheter hos PCguruen. Enhver må selv forsøke å holde seg rimelig orientert om de produktene som er interessante i den aktuelle jobbsituasjonen.

Når en har et begrep om hva en tror en trenger oppsøker en PCguruen. PCguruen vil vite hva som fins i huset av liknende saker og ting eller vil kunne ha en ide om andre som sitter inne med aktuelle erfaringer. Det forventes alle tar hverandres spørsmål seriøst og at vi hjelper hverandre og utveksler erfaringer slik at vi unngår å gjøre samme tabbene om og om igjen.

Kontakten med leverandørene kanaliseres gjennom PCguruen bare så langt dette er praktisk. Den som skal noe får gjøre det selv. PCguruen skal allikevel bistå når det gjelder f.eks å samle innkjøpene og oppnå rabatter hos leverandørene.

3.2.Første innstallajon og utprøving

Den som har kjøpt noe foretar første utprøving selv, eventuelt med noe bistand fra PCguruen. Det er viktig at PCguruen nå blir gjort kjent med hva som forgår slik at det kan nyttes av andre. Etter en stund vil en vite om det som er kjøpt er noe tess eller om det bør legges på hylla og oppføres i arkivet som en vond opplevelse.

Om en derimot mener at dette er brukbart skal den som har handlet sammen med PCguruen utarbeide en standardinstallasjon. PCguruen fører notater om at denne programpakken fins i huset og hvem som har erfaringer med den. Han oppbevarer originaldiskettene og distribuerer backupkopier til de som ønsker dette.

3.3.Bruk og kassering

Noen form for organisert opplæring ser en ikke behov for riktig ennå. Allikevel understreker jeg at vi skal hjelpe hverandre, og de som er involvert får selv holde dette i den form som er egnet.

Etter noen tids bruk er muligens ikke programpakken aktuell lenger. Det er da viktig at håndbøker m.m. leveres til PCguruen som så vil holde et lager. Dersom en programpakke skal følge et prosjekt gjøres det selvfølgelig unntak her.

PCguruen er også ansvarlig for å ta ut av sirkulasjon programvare som er uaktuell, f.eks. forsøke å begrense bruk av Turbo Pascal og heller pålegge bruk av Modula 2/86. Det er viktig å begrense antall programpakker som er i bruk.

3.4. Utskriftsstasjon.

PC ved DAK-anlegg er ment som en felles maskin for å lage utskrifter. Den er pr. idag (6-11-1987) tilkoblet den store plotteren som hører til DAK anlegg (HP 7580B). Denne kan benyttes til å tegne ut skjemaer fra OrCAD. Det er bestilt en OKI matriseskriver. Denne er rimelig rask, skriver brede listinger (132 char) og vil benytte listepapir. Planen er å tilknytte også en liten plotter (A3) med arkmater og en skjønnskriver (sannsynligvis laserskriver) for A4 ark.

3.4.1. Flytting av tekst og data til utskriftsstasjon.

Før utskrift må data eller tekst flyttes til utskriftsstasjonen ved hjelp av floppydisk. PC'en ved DAK har kun double density floppydisk drive, ikke high capacity. Det er derfor viktig at en benytter double density disker, ikke high capacity (se på merkingen på disken). Når double density disker skal formatteres på en IMB PC-AT eller klon, må en spesifisere /8 for å få riktig format.

3.4.2. Bruk av stor plotter (HP 7580B).

For å få plottet ut tegninger må følgende prosedyre følges:

1. Sett plotter i "Listen Only".
2. GP-IB kabelen kobles mellom plotter og PC.
3. Slå på PC'en.
4. Strømmen på plotteren settes på.
5. Legg i ark.
6. Trykk "HOLD".
7. Trykk "REMOTE".
8. Det er nå to muligheter for å plotte filen.
   1. DiskHp
      Enter file name: A:<filename>
   2. Copy A:<filename> LPT1
9. Går noe galt kontakt Per Erik.

3.4.3. Bruk av linjeskriver.

Denne skriveren kan skrive linjer på 132 karakterer.
For skrive ut brede listinger må man skrive :

LPWIDE A:<filename>

3.5. Backup av egne filer

Det er opp til den enkelte å lage backup av det arbeidet som bedrives. En bør benytte minst to sett med floppydisker som backup som en kopierer til annen hver gang. Bacthfiler som skal hjelpe til med å lage backuper er beskrvet nedenfor.

PCguruen er soussjef for det ulåste brannsikre skapet (det lengst til venstre ved den gamle inngangen oppe). Han kan anvise lagringsplass der slik at alle kan ha disketter der. ALT SOM VI ARBEIDER MED SKAL LAGRES I DET BRANNSIKRE SKAPET. Disketter er svært sårbare og det er lett å påføre seg selv unødige tap. Dersom en har disketter med gradert informasjon på kan dette lagers brannsikkert på annen måte, spør PCguruen.

Disketter som lagres skal ha en minimums merking:

- Navn:	Hvem er det som har puttet denne disketten inn her?.
- Prosjekt:	Alt merkes med prosjektnummeret.
- Innhold:	Hva er innholdet, gjerne med en liten kommentar som gjør det mulig for utenforstående å få en ide om innholdet. Også diskettformatet (vanlig eller backup) er viktig.
- Nummerering:	Diskene nummereres dersom flere hører sammen i en sekvens, f.eks ved backup.
- Dato:	Når ble diskene skrevet sist? Normalt vil det bli en lang liste med datoer hvor en stryker ut den gamle datoen og skriver en ny for hver gang en ta en ny backup.

Jeg skal gi et eksempel:

Erik Rosness
IK/6423 KRYSTLE
Modula2 source til PPDT (parallel program development tool)
#2 av 4
23/8/87 30/8/87 5/9/87

Med denne merkingen bør det være mulig å holde en viss oversikt over innholdet i skapet. Når en først har fått plass kan den enkelte selv gå i skapet og gjøre sin plikt.

NB! Begge dørene må alltid lukkes skikkelig, ellers er det ingen vits i å bruke brannsikkert skap.

3.6. Generelle oppfordringer

PCguruen vil sørge for sikker lagring og kopiering av all distribuert programvare. Sikkerhetskopiering av brukerfiler er overlatt til den enkelte. For den praktiske delen av backup vil jeg få henvise til bekrivelse nedenfor av vårt utmerkede backupopplegg.

PCguruen yter bare begrenset hjelp i først omgang. Dersom noen krever mer bør dette organiseres under der aktuelle prosjektet.

Det er heller ikke meningen at han skal kjenne alle program-pakkene, han står fritt til å henvise folk videre.

Noe videreutvikling vil sikkert finne sted både når det gjelder organisering og konfigurering. Alle kommentarer, snill kritikk og skamros mottas med takk, så kan dette om mulig bli enda bedre.

Det er forståelig at enkelte brukere vegrer seg for å inrette seg inn under et slikt system, men vi føler sterk trang til å være konsekvente. Det er allerede brukt opp mye penger til ingen nytte på grunn av manglende styring fra sentralt hold og, en regner med lojal oppslutning om dette (jfr. Einar Førde: Me er alle sosial-demokratar).

4. PCguruens plikter

PCguruens arbeidsoppgaver er ikke særlig omfattende. Han skal koordinere heller enn å utføre alle oppgaver (eller sitte inne med all kunnskap) selv. Formålet med alle aktivitetene er å forenkle tilværelsen for den enkelte bruker og å effektivisere driften.

4.1. Håndtering av innkjøpte programpakker

All programvare som kjøpes inn skal PCguru sørge for at blir forsvarlig testet, eventuelt innstallert, lagret og sikkerhets-kopiert. Alle originaldisketter lagres i låst brannsikkert skap. Backup av alle distribusjonsdisketter lagres i ulåst brannsikkert skap. PCguru holder oversikt over hvem som kjenner hvilke programpakker, og anbefaler hvilke pakker som bør brukes til ulike formål.

Vi skiller mellom to slags programpakker: vanlige og sære. Av de vanlige pakkene kan nevnes:

- MS-DOS operativsystem
- Brief program editor
- Lotus 123 regneark
- dBASE III+ database
- Word Perfect tekstbehandling
- Norton Utilities

Dette er produkter som PCguruen sørger for å kjøpe inn og håndtere slik at de blir tilgjenglige for alle. Håndbøker skal også skaffes i nødvendig antall.

Andre programpakker regnes som sære, og PCguruen skal bare hjelpe til på oppfordring.

PCguruen er også ansvarlig for avtaler med leverandørene, f.eks rabattordninger og rammeavtaler som sikrer gunstige innkjøp. PCgugren vil også bistå dersom det er ønsket av andre som kjøper inn programvare.

4.2. Håndtering av dokumentasjon

PCguruen skal ha oversikt over hva vi har av dokumentasjon, både den som følger med programpakkene og den som er kjøpt ved siden av. Han har skal også ha en rimelig oversikt over hvem som har de forskjellige bøkene, og om nødvendig supplere med nye.

4.3. Veiledning av brukere

PCguruen skal bidra til at alle brukerene opparbeider en kom-petanse som gjør PCene til effektive verktøy. Særlig skal han se til at PCene er riktig utstyrt for bruken og opplyse brukerene om ny programvare, nye metoder og muligheter m.m.

4.4. Drift av utskriftsstasjon.

PCguruen skal sørge for at PC ved DAK er i drift og kan benyttes. Han skal sørge for papir, fargebånd og annet nødvendig materiell for å holde installasjonen i drift.

4.5. Vedlikehold av menysystem

PCguruen er ansvarlig for vedlikehold av batchfil og menysystem. Han skal motta forslag, klager og problemer fra brukerene og videreutvikle menysystemet slik at det til enhver tid inneholder de funksjonene som behøves. Dokumentasjonen skal også oppdateres. Dersom vesentlige forbedringer eller endringer finner sted skal PCguruen sørge for fornuftig distribusjon av nyhetene.

5.Standard konfigurasjon for PC

Vi skal forsøke å holde IK PCer til en tilnærmet standard-konfigurasjon. Et forslag til konfigurasjon pr 03/09/87 er

PC AT
287 coprocessor (opsjon)
640 K RAM
EGA grafikk
2 serie porter
2 parallell porter
1 Microsoft serie-mus kompatibel (opsjon)

Pr dato er kostnaden ca 23.000 eks mva for denne konfigurasjonen desom vi handler en klon. Nye innkjøp skal drøftes med PCguru slik at en forvisser seg om at den nye maskinen kan benytte eksisterende programdistribueringsopplegg.

Tillegget for koprosessor, 2 serie/paralell porter og mus er ca 2000 kr + 2000 kr ≈ kr 4000. Det betyr at det er en besparelse for IK å inkludere dette i en standard-konfigurasjon i stedet for å legge til dette etter hvert.

6.Beskrivelse av batch-filer under MS-DOS

Dette avsnittet er en kort introduksjon til hvorfor batch-programmer brukes, samt en oversikt over de vanligste kommandoene som brukes i batch-filer.

En batch-file er DOS kommandoer som er lagret i en file. DOS vil behandle teksten i batch-fila som om den ble skrevet inn på keyboardet.

Enkelte ganger kan det være at vi skal ha uført en oppgave på PCen som krever at det tastes inn flere MS-DOS kommandoer etter hverandre. Dermed blir vi sittende å vente på at en og en kommando skal bli ferdig slik at vi kan få tastet inn den neste. Hvis vi i tillegg benytter denne serien av MS-DOS kommandoer ofte, vil bruk av batch-filer gjøre arbeidet mye lettere for oss. Alle de MS-DOS kommandoene som ønskes utført legges i en batch-fil, og kommando-sekvensen startes ved å skrive navnet på programmet. MS-DOS kommandoene blir da kjørt sekvensielt.

Foruten alle de mer eller mindre vanlige MS-DOS kommandoer finnes det en del spesielle kommandoer som kan brukes i batch-filer. Her kan blant annet nevnes:

REM - brukes for enten å vise fram meldinger på skjermen (ECHO = on), eller for å skille ut kommentarer fra vanlige MS-DOS kommandoer (ECHO = off).

ECHO - bestemmer om REM-setninger og navnet på de DOS kommandoene som blir kjørt skal vises på skjermen.

GOTO - med denne kan vi foreta hopp mellom de ulike MS-DOS kommando i batch-filen.

IF - gir mulighet for betinget prosessering i en batch fil.

FOR - gir mulighet for gjentatt prosessering.

ASK - Dette er en kommando som inngår i Norton Utilities 4.0, og som brukes for å lage interaktive batchprogrammer. Dvs. at man kan legge inn spørsmål til brukerne forskjellige steder i programmet.

PAUSE - Stopper batchfile og sier, ..strike any key to continue.. . Brukes til å lage pauser i batchprogrammet.

Forøvrig henvises det til "DOS - The Complete Reference" [1] av Kris Jamsa, samt "Supercharging MS-DOS" [2] av Van Wolwerton for nærmere studie av batch- filer/programmer.

6.1.Retningslinjer for skriving av batch-filer

For å få en mest mulig enhetlig og oversiktlig struktur på batch-filene i dette systemet, er det noen få retningslinjer som bør følges ved utvikling av nye batch-programmer som skal inngå i

systemet.

1. Identifikatorer skal gis entydige og selvforklarende navn slik at man slipper "unødvendige" kommentarer i programmet.

2. Alle identifikatorer skrives med liten bokstav først. Dersom identifikatoren er sammensatt av flere ord, skal første bokstav i de påfølgende ord skrives med stor bokstav, f.eks directoryDoesNotExist.

3. De "environment"-variable som settes når batch-programmet startes må nullstilles (tømmes) før batch-filen forlates.

4. Alle batch-programmer skal, når de startes, vise en heading på skjermen. Denne headingen skal inneholde navnet på programmet, eller en forklaring til hva programmet gjør. Videre skal headingen vise hvem som har laget programmet, samt programversjon. Denne siste informasjonen skal være lagret i variabelen "date-Revised". "dateRevised" skal alltid oppdateres når det gjøres endringer i programmet.

5. Headingen skal ha et bestemt format. På hver side av teksten skal det skrives ut blå stjerner (*), mens selve teksten skal skrives ut i gult. For å få til dette må ANSI.SYS driveren være innstallert i Config.sys. Samtidig må selve ANSI.SYS-programmet ligge under katalogen \APPS\DOS\. Se "Supercharging MS-DOS" av Van Wolwerton for beskrivelse av skjerm-/cursor- kommandoer i ANSI.SYS.

7.Beskrivelse av IK-meny system

Når systemet er installert vil det komme opp en hovedmeny på skjermen hver gang maskinen startes. Nedenfor er det vist et eksempel på en slik hovedmeny:

```
 Volume in drive C is SNOOPY 3
*****Programming Languages Installed*****************************
Type MOD2    [ program-directory ] to enter Logitech Modula2
Type PAS     [ program-directory ] to enter TurboPascal
*****Utility Programs Installed**********************************
Type LOTUS   [ 123     -directory ] to enter Lotus-123
Type REFLEX  [ reflex  -directory ] to enter Borland REFLEX
Type ABEL    [ abel    -directory ] to enter Data I/O ABEL
Type WP      [ text    -directory ] to enter SSI Word-processing WP
Type B       [          file      ] to enter Brief-editor
*****Management Utilities***************************************
Type BACK group [    ] [    ] to make a backup of user files
Type USER group [    ] [    ] for directory-management utilities
Type SHUT to prepare the system for moving
******24/08/87 TVO/JJ*******************************************
```

Fra ROOT-katalogen kan alle kommandoene i hovedmenyen velges. De fleste kommandoer krever spesifisering av en til tre parametere (katalog, fil, funksjon) forat de skal bli uført. Dersom parameterene ikke spesifiseres vil det komme opp en hjelpe-meny for den aktuelle kommandoen, eller en oversikt over hvilke parametere (kataloger) som kan velges.

Når det gjelder de enkelte kommandoene på hovedmenyen så ansees ikke ytterligere forklaring av kommandoene i de to første hovedguppene nødvendig. For den siste delen er det imidlertid på sin plass med en nærmere spesifisering.

BACK er et program for å ta backup av brukernes egne filer, uansett type. For å starte BACK må de aktuelle parameterene spesifiseres. Dersom parameterene ikke angis kommer en hjelpe-meny fram på skjermen. I disse dager (ca. 1/11/87) er en i ferd med å skifte backupprogram. Tidligere har BACKUP og RESTORE i MS-DOS vært benyttet. Til alles store overraskelse er disse kommandoene befengt med feil, og vi skal innføre bruk av DSBACKUP+. DSBACKUP+ vil produsere backup som kan benyttes uavhengig av MS-DOS versjon. Dog kan backup laget med DSBACKUP+ <u>ikke</u> blandes med backup laget i MS-DOS (eller omvendt). Bortsett fra dato-spesifikasjon (som ikke lenger vil være mulig) skal bruken av BACK kommandoen skal ikke endres. DSBACK, som det nye backup-programmet heter, har en ekstra funksjon sammenlignet med det gamle backup systemet. Denne funksjonen startes ved å skrive DSBACK USERDEF. Brukeren kan da selv kommunisere interaktivt med DSBACKUP+ programmet, dette gir mulighet for å ta backup av filer som ligger utenfor \USER katalogen.

Det er tre forskjellige parametere som må/kan spesifiseres:

BACK gruppe [directory] [type]

1. parameter:
[gruppe] - hvor gruppe er en av mulighetene som er vist nedenfor.Denne parameteren må alltid spesifiseres. Den angir enten hvilken hoved-katalog under USER som er utgangspunkt for backup-operasjonen, eller om backupen skal gjelde alle underkatalogene evt. bare kildekoden til de enkelte programmene. Følgende parameter-verdier er gyldige:

123 - hovedkatalog under USER til bruk for LOTUS-123.
text - hovedkatalog under USER til bruk for de forskjellige tekstbehandlings-pakkene.
generic - hovedkatalog under USER til bruk for GENERIC.
reflex - hovedkatalog under USER til bruk for REFLEX.
abel - hovedkatalog under USER til bruk for ABEL.
dbase3 - hovedkatalog under USER til bruk for DBASE3.
st80 - hovedkatalog under USER til bruk for ST80.
timeline - hovedkatalog under USER til bruk for TIMELINE.
all - alle bruker-filer i alle hovedkataloger under USER.
help - Viser meyen.
program - alle spesifiserte filer på hovedkatalogen PROG-RAM.(com-filer, mod-filer, osv.)
source - bare kildekoden til de programmene som spesifiseres.

2. parameter:
[xx] - hvor xx er en path fra hovedkatalog til den akuelle katalogen. Parameteren benyttes for å spesifisere eventuell underkatalog til den gruppekatalogen som ble angitt ovenfor.

3. parameter:
[type] - hvor type er en mulighetene nedenfor. Denne parameteren brukes for videre spesifisering/beg-rensning av antall filer som det skal tas backup av. Følgende parametere er gyldige:

[dd.mm.yy] - hvor dd er dag, mm er måned og yy er år. Ta backup av alle filer som er endret etter spesifisert dato. Denne muligheten fins <u>ikke</u> i DSBACKUP+.

mod - backup alle filer som er endret siden siste backup.

Når det tas backup av egenproduserte filer vha BACK.BAT vil backup-disketten i tillegg til brukerfilene inneholde en eller flere <gruppe>.LSTfiler.

Disse <gruppe>.LST-filene inneholder tid/dato for backup-operasjonen samt navn på filene som det ble tatt backup av. <gruppe>.LST-filen vil også bli liggende på tilhørende

underkatalog under \USER.

<gruppe>.LST-filene gis samme navn som den katalogen under \USER som er utgangspunkt for backup-operasjonen. Hvis det f.eks tas backup av tekst-filer vil både disketten og \USER\TEXT inneholde filen TEXT.LST.

<gruppe>.LST-filene kan brukes for å finne ut hvilke filer som ligger på den aktuelle backup-disketten uten at alle filene på disketten må "restores". Ved å bruke DOS-kommandoen RESTORE på bare <gruppe>.LST-filen og deretter liste ut innholdet av denne, fås en komplett oversikt over alle filene på backup-disketten.

<gruppe>.LST-filen som ligger på aktuell katalog under \USER brukes til å undersøke når det sist ble tatt backup fra denne katalogen, og hvilke filer som ble tatt backup av.

Når DSBACK.BAT benyttes vil disketten ikke inneholde slike <gruppe>.LST-filer. Derimot vil det bli lagret tilsvarende filer på katalogen \APPS\DSBACKUP. Disse filene har extention REP.

Som følge av at DSBACKUP+ har et annet format enn DOS' BACKUP-kommando kan ikke RESTORE kommandoen i DOS brukes for å hente filene som ligger på DSbackup-disketten. Til dette formålet finnes det en egen batch-fil, DSGET.BAT.
DSGET.BAT brukes på tilnærmet samme måte som DSBACK.BAT.
Det er to parametere som må spesifiseres når filene på en DSbackup-diskett skal hentes. Disse to parameterne er de samme som de to første parameterne for DSBACK.BAT.
F.eks må man skrive

DSGET PROGRAM KRYSTLE\PPDT

for å hente program filer som skal ligge på katalogen \USER\-PROGRAM\KRYSTLE\PPDT.

USER er et batch-program som inneholder en del funksjoner som brukes i forbindelse med kataloger. Ved hjelp av USER kan det lages nye kataloger, fjernes kataloger samt listes ut innholdet av spesifisert katalog.
Det er tre parametere som må/kan spesifiseres, dersom dette ikke gjøres vil det vises en hjelpe-meny på skjermen.

USER [gruppe] [function] [directory]

1. parameter:
[gruppe] - Denne parameteren må alltid spesifiseres. Den angir hvilken hovedkatalog under USER som er utgangspunkt for den aktuelle operasjonen.

Følgende er gyldige:

program - hovedkatalog under USER til bruk for kilde-filer for de ulike programmeringsspråkene.

123	-	hovedkatalog under USER til bruk for LOTUS-123.
text	-	hovedkatalog under USER til bruk for de - forskjellige tekstbehandlingspakkene.
reflex	-	hovedkatalog under USER til bruk for REFLEX.
abel	-	hovedkatalog under USER til bruk for ABEL.
dbase3	-	hovedkatalog under USER til bruk for DBASE3.
generic	-	hovedkatalog under USER til bruk for GENERIC.
timeline	-	hovedkatalog under USER til bruk for TimeLine.
st80	-	hovedkatalog under USER til bruk for SmallTalk.
orcad	-	hovedkatalog under USER til bruk for OrCAD.

2. parameter:

[function] - hvor function er valgt fra mulighetene som er vist nedenfor. Denne parameteren angir hvilke funksjoner som ønskes utført. Det kan velges mellom følgende funksjoner:

mkdir	-	lage en ny katalog.
mkdir	-	fjerne en eksisterende, tom katalog.
list	-	liste ut alle underkatalogene under den spesifiserte katalogen.
copy	-	kopiere filer fra spesifisert katalog til diskett i drive a:

3. parameter:

[xx] - hvor xx er et pathnavn. Denne parameteren brukes for å spesifisere aktuell katalog. Tegnene "*" og "?" kan benyttes ved kopiering av filer til a-driven.

Meny-systemet bygger på en rekke ulike batch-filer. Disse programmene er til for å gjøre den daglige bruk av MS-DOS og de ulike program-pakkene enklere for brukerne. Selve hovedmenyen genereres av programmet MAKEHELP. Dette programmet undersøker hvilke program-pakker som er installert i systemet, og lager hovedmeny på bakgrunn av dette. Måten dette gjøres på er at for hver batch-fil som skal starte en program-pakke, må det også finnes en tilhørende .HLP-fil. .HLP-filen er installert sammen med det kjøpte programsystemet. For eksempel vil 123.hlp ligge på katalogen \UTILITY\123. Man kan på en måte si at .HLP-filen fungerer som et "flag" som detekteres av MAKEHELP. Dersom .HLP-filen mangler vil ikke den aktuelle program-pakken komme fram på menyen.
MAKEHELP må alltid kjøres etter at en ny program-pakke er installert i systemet. Dette gjøres ved å skrive MAKEHELP.

7.1.AUTOEXEC.BAT

Dette er en batch-fil som ligger under ROOT-katalogen. Hver gang maskinen startes finner MS-DOS AUTOEXEC.BAT og utfører den. I AUTOEXEC.BAT legges det inn kommandoer og programmer som ønskes utført hver gang maskinen starter.
AUTOEXEC.BAT inneholder setting av "environment"-variable samt installasjons-/konfigurasjons-prosedyrer for drivere, kort og IK

meny system.
Det er laget en standard AUTOEXEC.BAT fil som brukes på IK. Den skal inneholde alle aktuelle oppsett brukt på IK, slik at på hver PC skreddersys standard filen til den aktuelle brukers individuelle behov.

Den 24/08/87 er det lagt inn følgende prosedyrer i Autoexec.bat:

```
echo off
set dateRevised=230787 TVO
echo *****Autoexec SetUp  %dateRevised%************************
rem Sets the hardware configuration and installation on the PC-AT
rem SNOOPY 3

set ramdrive=noRamDrive
set helppath=c:\bat_help;c:\apps\dos;c:\apps\nu;c:\apps\dosutil;
path %helppath%

                    SETTING AV MODULA2 PARAMETERE

set moddrive=c:
set moddir=%moddrive%\devtool\modula2
:noModula2Setup

cls

              SETTING AV TEGNSETT PÅ SKJERM OG TASTATUR

rem Sets the Norwegian Characters on The EGA
eganor
:noEGACopam

rem Sets Norwegian Character on Copam Turbo AT
coptast /E
:noCoptast

goto noKeybno
rem sets norwegian charaters
keybno g/9/5
:noKeybno

goto noIBMmodel3Setup
NLSFUNC
MODE CON CODEPAGE PREPARE=((850) c:\apps\dos\ega.cpi)
MODE CON CODEPAGE SELECT=850
KEYB NO,850,c:\apps\dos\keyboard.sys
rem Include in config.sys
rem country=047,850,c:\apps\dos\country.sys
rem DEVICE=c:\apps\dos\DISPLAY.SYS CON:=(EGA,850,1)
:noIBMmodel3Setup

                INSTALLASJON AV RESIDENTE PROGRAMMER

goto noSidekick
rem enable sidekick
```

```
Ask "Do you want SIDEKICK? ",nNyY
if errorlevel 3 goto sidekick
if errorlevel 1 goto noSidekick
:sidekick
rem sidekick notes
call skn
:noSidekick

goto noResidentCalculator
rem Sets Resident Calculator on
rem and marks current memory position
rem the resident memory allocation is deallocated using
rem RELEASE.COM
mark
rescalc
: noResidentCalculator
```

DIVERSE

```
goto noFastCursor
rem Sets the cursor to move faster
quick
:noFastCursor

rem Sets ced to provide command buffering
ced -f\apps\dosutil\ced.cfg
:noCommandEditor

goto noSpool
rem Enable spooling
Ask "Do you want spooling for line printer? ",nNyY
if errorlevel 3 goto spool
if errorlevel 1 goto noSpool
:spool
call spool lo
:noSpool

rem type help-menu
cls
type c:\bat_help\help.hlp

rem Sets coloured prompt
SETPROM 2
set dateRevised=
```

7.2.CONFIG.SYS

CONFIG.SYS ligger under ROOT-katalogen. Maskinen bruker denne filen hver gang den startes.
CONFIG.SYS informerer MS-DOS om en del system parametere samt eventuelle drivere som måtte være installert i maskinen.
Den 24/08/87 inneholder en typisk CONFIG.SYS følgende:

```
shell=command.com /p /e:2500
break=on
```

```
country=047,,c:\apps\dos\country.sys
lastdrive=N
files=30
buffers=30
device=c:\apps\dos\ansi.sys
```

Eksemplet på CONFIG.SYS viser et minimum for IK menysystem. Det er definert et utvidet "enviroment space", 2500 byte, satt norsk landskode, definert diskdrives fra A: til N:, et utvidet antall filer og buffere og den utvidete skjermdriver ANSI.SYS.
Setup for ramkort og IBM enhanced keyboard er ikke inkludert i eksemplet.

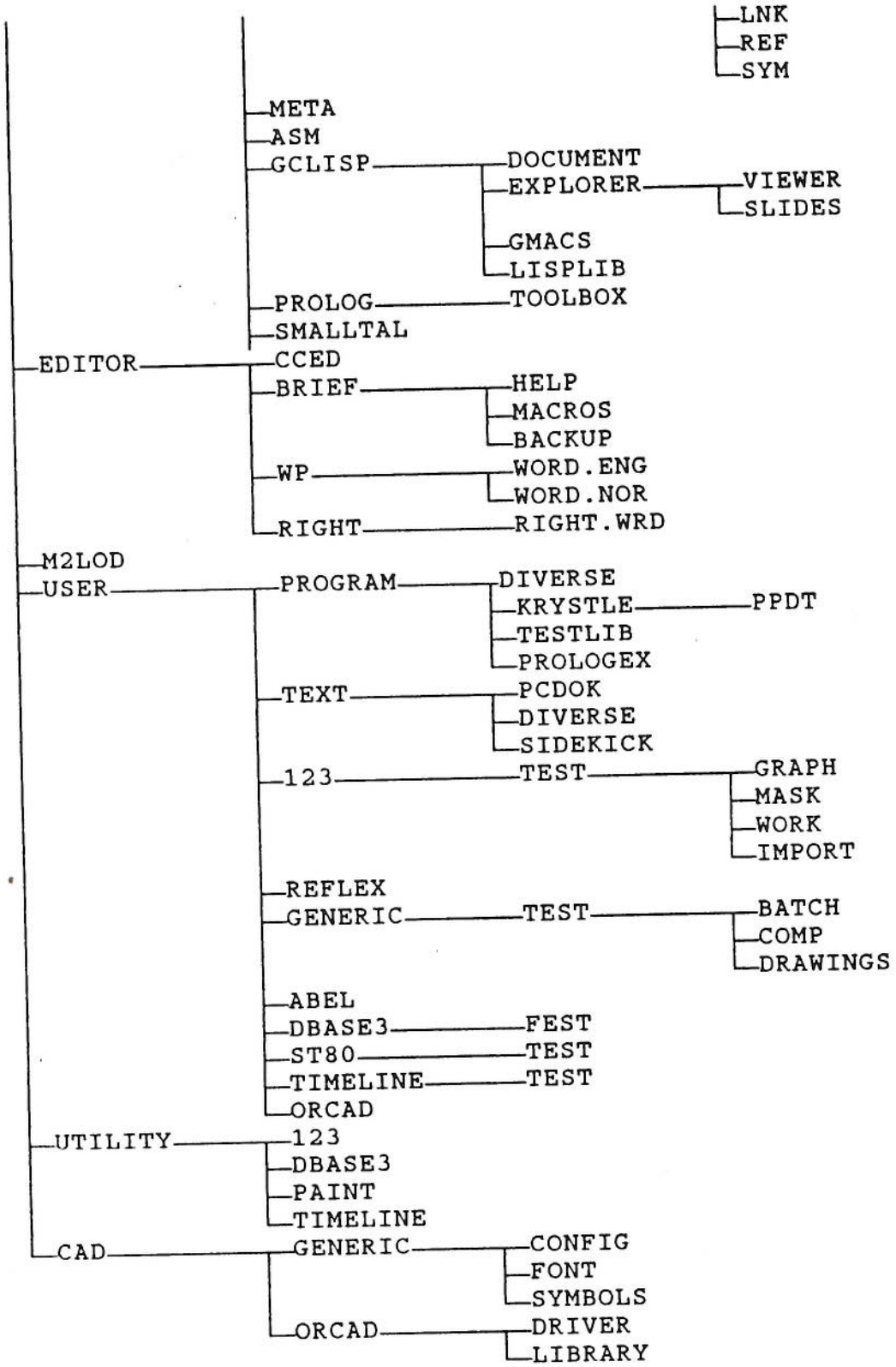
8.Filstruktur

Det er definert opp en filstruktur som alle PCer på IK skal inneholde. Den er definert generelt slik at alle typer programmer kan legges inn i strukturen.

8.1.Bilde av generell filstruktur

En typisk filstruktur pr. 16/11/87 er vist nedenfor:

```
C:\ ─┬─APPS──────────┬─DOS
     │               ├─NU
     │               ├─DOSUTIL
     │               ├─SIDEKICK
     │               ├─GPIB
     │               ├─TEST
     │               ├─COMPUTER
     │               └─DSBACKUP
     ├─BAT_HELP
     ├─DEVTOOL───────┬─TURBOPAS
     │               ├─MODULA2──────┬─GRAPHIX──────┬─DEF
     │               │              │               ├─LNK
     │               │              │               ├─REF
     │               │              │               └─SYM
     │               │              ├─HGRAPH────────┬─DEF
     │               │              │               ├─LNK
     │               │              │               ├─REF
     │               │              │               └─SYM
     │               │              ├─ISAM──────────┬─DEF
     │               │              │               ├─LNK
     │               │              │               ├─MOD
     │               │              │               ├─REF
     │               │              │               └─SYM
     │               │              ├─M2LIB─────────┬─DEF
     │               │              │               ├─LNK
     │               │              │               ├─REF
     │               │              │               └─SYM
     │               │              ├─PAS2MOD───────┬─DEF
     │               │              │               ├─LNK
     │               │              │               ├─MOD
     │               │              │               ├─REF
     │               │              │               └─SYM
     │               │              ├─REP──────────┬─DEF
     │               │              │               ├─LNK
     │               │              │               ├─MOD
     │               │              │               ├─REF
     │               │              │               └─SYM
     │               │              ├─RTK──────────┬─DEF
     │               │              │               ├─LNK
     │               │              │               └─SYM
     │               │              ├─SUPER────────┬─DEF
     │               │              │               ├─LNK
     │               │              │               ├─MOD
     │               │              │               ├─REF
     │               │              │               └─SYM
     │               │              └─META─────────┬─DEF
```



8.2. \BAT_HELP

\BAT_HELP katalogen inneholder batchprogrammene for IKs meny - system. På katalogen finnes også en versjonskontroll i filen

MENYIK.V**. Denne filen inneholder installert versjon samt en endrings-protokoll for meny-systemet.
Batch-filene kan deles inn i to hovedgrupper. Den første gruppen består av batch-filer som brukes for å starte opp de ulike program-pakkene som er installert i systemet. Den andre hoved-gruppen inneholder en del kommandoer som er en videreutvikling av enkelte MS-DOS kommandoer, slik at de er mer egnet til bruk i dette systemet.
Den 24/08/87 finnes følgende batchprogrammer i hovedgruppe 1:

MOD2	BAT
ABEL	BAT
ASM	BAT
ED	BAT
CAD	BAT
DBASE3	BAT
FASTM2	BAT
FTN	BAT
LOTUS	BAT
PAS	BAT
REFLEX	BAT
WP	BAT

Hva de forskjellige programmene gjør burde gå klart fram av navnene.
Hovedgruppe 2 består av følgende programmer:

USER	BAT	-	Se "Beskrivelse av meny system"
BACK	BAT	-	Se "Beskrivelse av meny system"
ROOT	BAT	-	Fører brukeren tilbake til ROOT-katalogen.
SETPROM	BAT	-	Setter forskjellige farger på maskinens prompt.
HELP	BAT	-	Får fram hovedmeny på skjermen.
MAKEHELP	BAT	-	Genererer hovedmeny.
HOME	BAT	-	'Blanker' skjermen og venter på et taste-trykk. Brukes for å spare skjermen, slik at skjermbildet ikke 'brenner' seg fast.
DSBACK	BAT	-	Lag backup på diskett ved hjelp av DS-BACKUP+.
DSGET	BAT	-	Hent inn fra diskett som inneholder backup laget ved hjelp av DSBACKUP+.

8.3. \APPS

Denne katalogen skal bare inneholde kataloger som gjelder drift av PCen, støtte og setup-programmer.

8.3.1. \APPS\DOS

Denne katalogen skal inneholde IBM siste versjon av PC-DOS.
Den 24/08/87 er det PC-DOS 3.30

8.3.2. \APPS\DOSUTIL

Denne katolgen skal inneholde programmer og drivere som kommer i tillegg til DOS. Dette er programmer som er samlet opp over lang tid. Denne katalogen vil være under oppdatering da noen av programmene etter en tid blir erstattet av nye DOS-versjoner.

8.3.3. \APPS\COMPUTER

Dette er programmer som brukes bare på denne maskinene og ikke hører inne på apps\dosutil.

8.3.4. \APPS\NU

Denne katalogen skal inneholde Norton Advanced Utilities 4.0. Norton Utilities er et av de faste støtteprogrammene som ligger i IK meny-system. Det kan blant annet brukes til å finne tilbake slettede filer, søke etter filer i hele filstrukturen, sjekke hastighet på PC og defragmentere harddisken. Dette er nyttige hjelpeprogrammer som ikke finnes i PC-DOS.

8.3.5. \APPS\TEST\<computer>

Denne katalogen skal inneholde testprogrammer som følger med den enkelte PC. Det finnes slike testprogrammer for alle PC-typer. Den 02/09/87 finnes følgende test/diagnose-programmer:

CAF
COPAM
IBM AT
IBM ADVANCED DIAGNOSTIC

Hver PC type skal bare ha testprogrammer som er beregnet for denne typen. For eksempel skal \APPS\TEST\COPAM inneholde test og diagnose programmer for PC 501 Turbo.

8.3.6. \APPS\GPIB

\APPS\GPIB skal inneholde de programmer og drivere som følger med National PC2A GPIB-controller. Denne driveren er testet av IK og fungerer utmerket.
24/08/87 TVO
Forhandler er Elektronix A/S. IK har en 10% rabattavtale.
Kontaktmann er Oppheim.

8.3.7. \APPS\SIDEKICK

Borlands Sidekick er lagret på denne katalogen. Sidekick er et resident program som inneholder en avtaleliste, kalkulator, editor og en modemdriver.
Standard brukerkatalog er \USER\TEXT\SIDEKICK.

8.3.8. \APPS\DSBACKUP

På denne katalogen ligger alle filer som har tilknyttning til

DSBACKUP+ pakken, dvs program-filer, setup-filer samt <gruppe>.-REP-filer.
Som nevnt tidligere i dette notatet skal DSBACUP+ brukes istedenfor DOS' backupkommando. Grunnen til dette er at det er oppdaget feil i DOS, videre er DSBACKUP+ uavhengig av DOS versjon. I tillegg er DSBACKUP+ raskere enn DOS sin backup kommando.
De som ønsker mærmere kjennskap til DSBACKUP+ henvises til PCguruen som har manual for program-pakken.

8.4. \USER

Under \USER-katalogen skal alle brukerfiler være lagret. Det skal eksistere en katalog under \USER for enhver kategori av programpakker installert i systemet. For eksempel \USER\TEXT for tekstbehandling og \USER\123 for Lotus 123. Meny programmene USER og BACK, beskrevet i "Beskrivelse av Meny-system", letter håndteringen av brukerfilene under \USER.

8.4.1. \USER\123

Her lagres alle bruker-filer som LOTUS 123 anvender. Dette forutsetter at 123 programmet er konfigurert til å lete på katalog c:\user\123. Dette gjøres ved å bruke /F(ile)D(irectory) kommandoen i 123. Dette gjøres en gang for alle ved første gangs installasjon. Under user\123 katalogen legges underkataloger som ordnes etter prosjekt, oppgave e.l. Hver enkelt av disse deles i 4 underkataloger som er beskrevet i det følgende.

8.4.1.1. \USER\123\<dir>\GRAPH

Her legges alle filer som skrives ut fra 123 for å taes inn i Printgraph programmet for grafisk utskrift på HP-plotter eller matriseskriver. Alle disse filene har extension .PIC.

8.4.1.2. \USER\123\<dir>\MASK

Her legges alle worksheet filer brukes som mal. De inneholder f.eks. sett av makroer som brukes som mal for flere regneark. De vil i de fleste tilfeller være filer med extension .WK1.

8.4.1.3. \USER\123\<dir>\WORK

Her legges alle worksheet filer. Dette er filer som inneholder et helt regneark. De har extension .WK1.

8.4.1.4. \USER\123\<dir>\IMPORT

Her legges alle filer som inneholder data som skal taes inn med /F(ile)I(mport) kommandoen i 123. Dette er ASCII filer med extension .PRN.

8.4.2. \USER\GENERIC

Under denne katalogen ligger bruker filer til Generic CADD. De er ordnet etter prosjekt, deloppgave e.l. For hvert prosjekt finnes 3 forskjellige under kataloger.

8.4.2.1. \USER\GENERIC\<dir>\DRAWINGS

Her ligger tegninger. Dette er filer med extension .DWG.

8.4.2.2. \USER\GENERIC\<dir>\BATCH

Her ligger batch filer som brukes for å utføre serier av kommandoer i Generic CADD. De har extension .TXT.

8.4.2.3. \USER\GENERIC\<dir>\COMP

Her ligger egne bibliotek av komponenter. De kan inkluderes i en hvilken som helst tegning. Dette er filer med extension .DWG.

8.4.3. \USER\TEXT

Under denne katalogen skal alle filer som genereres i forbindelse med bruk av de ulike tekstbehandlingspakkene ligge. Filene er ordnet etter prosjekt e.l . For hvert prosjekt finnes en under-katalog.

8.4.4. \USER\PROGRAM

Under denne katalogen skal alle typer filer som genereres i forbindelse med utvikling av programmer ligge. Det betyr at denne katalogen vil inneholde alt fra kildekode til eksekverbare programmer.
Filene er ordnet etter prosjekt e.l. Brukerene må selv finne en egnet struktur for å ordne sine underkataloger. Normalt vil disse katalogene inneholde mange filer og det er viktig å skape litt orden i kaoset. Dersom filer skal flyttes mellom maskiner er backup en bra metode, men da må katalogstrukturene være like. Dette må en ta hensyn til.

8.4.5. \USER\DBASE3

Under denne katalogen ligger kataloger med brukerfiler til DBASE3. De er ordnet etter prosjekt, deloppgave e.l. For hvert prosjekt finnes en underkatalog. DBASE3 skiller mellom de ulike filetype ved hjelp av sitt eget system med ulike extentions. Ved import og eksport til/fra TimeLIne benyttes en DBASE3 katalog med samme navn som TimeLine katalogen. Se TimeLine for dypere innsikt.

8.4.6. \USER\REFLEX

Under denne katalogen ligger bruker-filene for Borlands REFLEX. De er ordnet etter prosjekt e.l. For hvert prosjekt finnes en underkatalog.

8.4.7. \USER\ABEL

Under denne katalogen ligger katalogene med bruker-filene for Data I/O ABEL. De er ordnet etter prosjekt e.l. For hvert prosjekt finnes en underkatalog. De ulike filetypene som ABEL bruker eller produserer skilles fra hverandre ved hjelp av extentions.

8.4.8. \USER\ST80

Under denne katalogen ligger katalogene med bruker-filene for SmartSoft SmallTalk AT. De er ordnet etter prosjekt e.l. For hvert prosjekt finnes en underkatalog. Kun imagefiler vil bli lagretr her i første omgang. Dersom andre typer filer skal benyttes må en gjøre mere arbeid på menysystemet. Kontakt eventuelt PCguru.

8.4.9. \USER\TIMELINE

Under denne katalogen ligger katalogene med bruker-filene for Breakthrough TimeLine. For hvert prosjekt finnes en underkatalog. De ulike filetypene som TimeLine bruker eller produserer skilles fra hverandre ved hjelp av extentions.

8.4.10. \USER\ORCAD

Under denne katalogen ligger filene som genereres ved bruk av ORCAD skjemategnings verktøy.

8.5. \DEVTOOL

Under \DEVTOOL katalogen skal alle program-utvikling verktøyer ligge. Dette er først og fremst programmeringsspråk som Modula2, Pascal og Fortran.

8.5.1. \DEVTOOL\MODULA2

På denne katalogen er Logitechs Modula2 installert. Det er i tillegg definert et sett batch programmer som håndterer kompilering og lenking av programmer samt oppbygging av egne biblioteker.

8.5.1.1. \DEVTOOL\MODULA2\META

Logitechs overbyggning på grafikkpakken MetaWindow. Dette er en vel utbygd og svært rask grafikkpakke for bruk sammen med Logitechs Modula2. Ulempen er størrelsen på den medfølgende driver (som en må innstallere) som stjeler ca. 130k av memory.

8.5.1.2. \DEVTOOL\MODULA2\SUPER

I dette biblioteket er Ødegaards Elektronikks toolkit bibliotek for Modula2 installert.

8.5.1.3. \DEVTOOL\MODULA2\M2LIB

Dette er biblioteket som følger med Modula2. Det er Wirths definisjon av Modula2.

8.5.2. \DEVTOOL\TURBOPAS

Borlands TurboPascal 3.0 er installert på denne katalogen. Vi anbefaler at Modula2 benyttes framfor Pascal på grunn av store fordeler ved større systemer og ved gjenbruk av programmer.

8.5.3. \DEVTOOL\MSFTN

MicroSoft FORTRAN versjon 3.31 er installert på denne katalogen, sammen med LINK og PLINK86, som brukes for å linke sammen program. Det finnes også 2 batch-filer: F og CLOAD.

F <fil> brukes for å kompilere en fil. Filnavnet gis uten extension, da det forutsettes at denne er .FOR. Ved kompileringsfeil kalles BRIEF editoren automatisk opp med 2 filer i hvert sitt vindu: <fil>.FOR og <fil>.ERR.

CLOAD <fil> brukes for å kompilere og linke ved hjelp av PLINK86. Ved kompileringsfeil kalles BRIEF editoren automatisk opp med 2 filer i hvert sitt vindu: <fil>.FOR og <fil>.ERR.

8.5.4. \DEVTOOL\FTN

Ryan McFarland FORTRAN versjon 2.0 er installert på denne katalogen, sammen med LINK og PLINK86, som brukes for å linke sammen program. Det finnes også 2 batch-filer: F og CLOAD.
I disse dager, 13/11/87, er det kommet en ny fortran versjon, 2.4, fra Ryan McFarland. Denne versjonen kan brukes selv om maskinen ikke har co-prosessor.
F <fil> brukes for å kompilere en fil. Filnavnet gis uten extension, da det forutsettes at denne er .FOR. Ved kompileringsfeil kalles BRIEF editoren automatisk opp med 2 filer i hvert sitt vindu: <fil>.FOR og <fil>.ERR.

CLOAD <fil> brukes for å kompilere og linke ved hjelp av PLINK86. Ved kompileringsfeil kalles BRIEF editoren automatisk opp med 2 filer i hvert sitt vindu: <fil>.FOR og <fil>.ERR.

8.5.5. \DEVTOOL\GCLISP

Dette er Gold Hills Common Lisp, liten versjon (uten virtual memory mulighet). En meget omfattende hjelpe og opplæringspakke (San Marco Explorer) følger med.

LISP er et av de eldste dataspråkene med røtter langt tilbake i 50 åra. Utgangspunktet for lisp er en liste. (LISP står for list processing) som er en ordnet samling med objekter. For den som er interressert kan en benytte den medfølgende opplæringspakka uten forhåndkunskaper.

8.5.6. \DEVTOOL\PROLOG

Detter Borlands TurboProlog med Toolbox. Prolog er et AI språk med innebygget backtracking. Borlands versjon har også bra gafikk og muligheter for kobling til andre språk.

Når PROLOG startes for første gang vil ens egen private konfigurasjonsfile automatisk bli generert. Dersom en endrer oppsettet og saver dette vil endringen bli permanent, men begrenset til denne katalogen.

8.5.7. \DEVTOOL\ST80

Dette er programmeringsspråket som ble utviklet ved Xerox PARC i 70årene. SmallTalk var en del av utgangspunktet da Apple designet MacIntosh.

Smalltalk vil ha sin egen type directory for brukerfiler hvor et image av smalltalk lagres. Dette er gjort slik fordi programmering i SmallTalk egentlig er modifisering og utbygging av det eksisterende systemet. En imagefile er stor, ca 600k byte.

SmallTalk krever AT, seriemus med tre kapper, EGA grafikk og minst 500kbyte extended memory i tillegg til 640kbyte i vanlig DOS bruker memory område. Under kjøring av SmallTalk settes 80286 over i virtual addressing mode.

8.6. \UTILITY

Under \UTILITY katalogen skal programpakker som inneholder database, prosjektstyring, regneark osv være lagret. Eksempler er Borlands Reflex, Ashton Tates dBaseIII og Generic Cadd 3.0.

8.6.1. \UTILITY\REFLEX

Dette er et enkelt databaseprogram. Det er pr 03/09/87 laget et system for generering av komponentlister. REFLEX bør kun benyttes for enkle lister og liknende, større oppgaver bør løses med dBASE3+.

8.6.2. \UTILITY\DATAIO

Denne katalogen benyttes av ABEL fra DataIO.

8.6.2.1. \UTILITY\DATAIO\ABEL

ABEL er et høynivåspråk for å programmere PLD's (programmable logic devices). Programpakken lager output-filer i JEDEC format som kan overføres til brenner med RS232 kabel. Alle PLDer som

svis bør programmeres med ABEL av hensyn til vedlikehold og endringer.

8.6.2.2. \UTILITY\ABEL\LIB

Denne katalogen inneholder et bibeliotek med beskrivelse av de kretsene som ABEL kan programmere.

8.6.3. \UTILITY\DBASE3

DBASE3+ er et databaseprogram. En kan strukturere og lagre ulike type informasjon. DBASE3+ kan søke, liste ut og lage rapporter. I tillegg fins det et eget høynivåspråk.

8.6.4. \UTILITY\123

Lotus 123 er et meget godt regneark. Det er på IK brukt til å lage et prosjektoppfølgingsystem, presentasjon og bearbeiding av simuleringsresultater og som en enkel database. 123s gode grafikkmuligheter må fremheves med driver til plotter og matrise-skrivere.

8.6.5. \UTILITY\TIMELINE

TimeLine er et prosjektplanleggings og oppfølgingsvektøy. TimeLine er i første rekke egnet til å planlegge og følge opp aktiviteter i tid, og er også nyttig ved kalkulering av pro-sjekter. TimeLine er lite egnet til oppfølgning på kostnadssiden.

Første gang timeline startes fra en katalog må en gjennom en runde med oppsett for å fortelle hvor brukerfilene ligger. Batchfilen vil skrive un hjelpeinformasjon mens dette skjer. Om en senere ønsker hjelpeinformaskjon kan en skrive

TL HELP.

Timeline kan utveksle datafiler med Lotus123 og dBASE3+. For dette fins det et eget hefte som fulgte med timelinbe. Om en skal benytte denne muligheten <u>må</u> katalognavnene under TimeLine og dBASE3+ være like. Nå TimeLine startes skriver en

TL [gruppe] [katalog] [schedule]

hvor gruppe er LOTUS eller DBASE3, katalog er katalognavnet (både i dBASE3+/LOTUS og TimeLine), og scehdule er filenavn på schedule i TimaLine.

Før TimeLine startes vil de aktuelle filene bli kopiert til TIMELINE katalogen (i hvert fall gjøres et forsøk). Når en forlater TimeLine blir en spurt om disse filene skal kopieres tilbake.

8.6.6. \UTILITY\FILTER

Dette er en digital filter designpakke utviklet i Trondheim på NTH.

Den er godt egnet til å designe og dokumentere enklere digitale filtre.

8.6.7. \UTILITY\1STCLASS

Dette er et program for å lage enklere ekspertsystemer. Programmet er ikke installert generelt, kontakt Sverre Holm for informasjon.

8.7. \EDITOR

Under \EDITOR katalogen skal alle editorer og tekstbehandlings-pakker være installert. Installerte pakker er WordPerfect 4.1 og Brief 2.0 programeditor.

8.7.1. \EDITOR\BRIEF

Brief editor brukes i forbindelse med programvare-utvikling. Editoren er svært generell, og kan brukes for mange programmerings-språk. Grunnen til dette er at det kan lages makroer som er tilpasset for de ulike programmerings-språkene. Disse makroene loades automatisk når editoren startes. Hvilke makroer som loades avhenger av filnavn extension. Også tabulatorsetting avhenger av hvilken type fil som editeres.

Brief editoren er relativt enkel å bruke. Editoren har angremuligheter for en rekke kommandoer, og søkemulighetene er veldig sterke.
Dokumentasjon for å skrive makroer er dårlig, og det kan derfor være litt vanskelig å komme igang med å skrive egne makroer. Når disse vanskelighetene er overvunnet har man imidlertid store muligheter med egne makroer.

I tillegg til standard biblioteket er det laget makroer for:

- merking/kopiering/sletting av boks markering.
- listing av søke-linjer til en fil.
- uppercase/lowercase konvertering.
- grafiske bokser og linjer.

For FORTRAN er det laget makroer med følgende funksjoner:

- utfylling av standard heading for funksjoner og subrutiner.
- definering av variable, parameterlist, eksterne funksjoner.
- dokumentasjon av innholdet på filer: subrutiner, funksjoner samt purpose og kalle sekvens entrier.
- logging av alle nye rutiner på egen fil.

Hjelpemenyen er enkel å forandre, og andre menyer kan lett lages.

Installasjonen av Brief er noe særegen. Path til Brief ligger i helppath, derfor kan Brief alltid startes uansett hvor man befinner seg.

8.7.2. \EDITOR\WP

WordPerfect 4.1 programsystem er lagret på denne katalogen. Når WordPerfect startes fra vil \USER\TEXT bli substituert med h:, dette medfører at man slipper å spesifisere hele path'en ved endring av arbeids-fil.

Dersom ordlista skal brukes behøver en å kunne velge mellom norsk og engelsk ordliste. Dette gjøres ved å gå inn i setup i WordPerfect (start WP med WP /S) og velg 1. Riktig path til ordlista settes opp. For pathnavn bruk \EDITOR\WP\WORD.NOR eller \EDITOR\WP\WORD.NOR for henholdsvis norsk og engelsk ordliste.

For engelsk tekst kan vi tilby en engelsklærer ved navn RightWriter som analyserer teksten overraskende nøye. For å aktivere RightWriter skrives RWRITE [directory] hvor [directory] er navnet på den tekstkatalogen hvor den aktuelle teksten er lagret. Før tekstanalysen startes vil en få en liste over filene på katalogen, og en må fortelle hvilken file som skal analyseres. RightWriter lager en ny file som inneholder teksten samt kommentarer.

8.7.2.1. \EDITOR\WP\WORD.NOR

Den norske ordlisten og stavekontrollen til WordPerfect er lagret på denne katalogen.

8.7.2.2. \EDITOR\WP\WORD.ENG

Den engelske ordlisten og stavekontrollen til WordPerfect er lagret på denne katalogen.

8.7.3. \EDITOR\RIGHT

Denne katalogen inneholder et program som sjekker engelsk tekst, se forklaring under beskrivelsen av WP.

8.7.4. \EDITOR\CCED

Dette er en enkel programeditor som er Wordstar kompatibel. Den regnes ikke som særlig brukervennlig, og anbefales kun for de som er vant med Wordstar.

8.8. \CAD

Under denne katalogen skal alle CAD-pakker ligge. Pr. 13/11/87 er det Generic Cadd og Orcad som ligger her.

8.8.1. \CAD\GENERIC

Dette er et DAK program for tegning av nøyaktige tegninger. Ved siden av PC med grafisk skjerm kreves en mus. Uegnet til skjema-tegning.

8.8.1.1. \UTILITY\GENERIC\CONFIG

Her ligger alle filer som har med installasjon og oppsett av Generic CADD 3.0.

8.8.1.2. \UTILITY\GENERIC\SYMBOLS

Her ligger Generic CADDs komponentbibliotek med komponenter for elektro, eksempel er transistorsymboler o.a. Disse kan taes inn i hvilken som helst tegning.

8.8.1.3. \UTILITY\GENERIC\FONT

Her ligger alle skrift type filer som brukes ved generering av tekster i Generic CADD.

8.8.2. \CAD\ORCAD

Dette er en pakke for skjema-tegning av elektronikk. Gir i tillegg til selve tegningen(e) også komponentlister og komponent-plassering.

8.8.2.1. \CAD\ORCAD\DRIVER

Under denne katalogen ligger drivere for printer, plotter og display.

8.8.2.2. \CAD\ORCAD\LIBRARY

Under denne katalogen ligger det en del komponent biblioteker som brukes ved skjemategningen.

9.Retningslinjer for utvidelse av filstruktur

For å bevare den ryddige og oversiktlige filstrukturen som er innført på IKs PCer er det satt opp noen retningslinjer som skal følges ved endring av og tilføyelser i filstrukturen.

1. Nye program-utviklings verktøy legges i egen katalog under \DEVTOOL. Bruker-filer som genereres i forbindelse med den nye program pakken legges under \USER\PROGRAM.

2. Nye editorer og tekstbehandlings-pakker legges i egen katalog under \EDITOR. Bruker-filer som genereres i forbindelse med den nye program pakken legges under \USER\TEXT.

3. Programpakker som ikke hører inn under de to kategoriene som er nevnt foran plasseres i egen underkatalog under \UTILITY.
   I tillegg opprettes det en tilhørende brukerkatalog for den nye programpakken under \USER.

4. Alle batch-programmer som tilhører en av kategoriene beskrevet i avsnittet C:\BAT_HELP legges under katalogen \BAT_HELP.

5. Programmer som har med drift av PCen å gjøre samt støtte-/ setup- programmer, legges i passende underkatalog under \APPS.

6. Alle CAD-pakker legges inn under \CAD.

PC-brukere bør ikke installere nye programpakker etter eget forgodt-befinnende. Dette vil medføre alminnelig anarki.

Derom alle følger de punktene som er nevnt foran vil vi bedre den enkeltes mulighet til å gjøre en effektiv jobb på grunn av de muligheten PCen gir. IK vil da være sikret en oversiktlig, veldefinert og effektiv håndtering av PCene.

QU [filespec] [/A]

Switch

/A Automatic (non-interactive) mode

SA *main-setting* [/N]

or

SA [*intensity*] [*fore*] [ON *back*] [/N]

Choices for *intensity*

Bright Bold Blinking

Choices for *main-setting*

Normal Reverse Underline

Choices for *fore* and *back*:

White Black Red Magenta
Blue Green Cyan Yellow

Switch

/N Do not set border color

SD [*d*:]

or

SD [*item-to-report-on*] /REPORT [/S]

Switches

/Report Generate a report, do not rearrange
/S Include subdirectories in report

SI [*d*:] [*switches*]

Switches

/LOG Format output for printer or file
/N No live-memory probe

TM [START] [REPORT] [*comment*] [*switches*]

Switches

/C*n* Select Counter *n* (*n* is 1, 2, 3 or 4)
/L Write TM info on Left side of screen
/LOG Format output for printing or file
/N No display of current time or date

TS [*filespec*] [*search-text*] [/S] [/T] [*switches*]

or

TS [*search-text*] /D *or* /E [*switches*]

Common Switches (for all command formats)

/A Automate search; yes to all prompts
/EBCDIC Files are EBCDIC encoded
/LOG Format output for file or printer
/N Not 100% IBM compatible
/WS WordStar—no extended characters

Switches When Searching Files

/S Search through Subdirectories also
/T Totals (non-interactive)

Switches When Searching Disk or Erased Space

/D Search entire Disk
/E Search Erased space
/C*n* Start the search at cluster *n*

UD [*directory pathname*]

VL [*d*:] [*label*]

WIPEDISK *d*: [*switches*]

Switches

/E Wipe erased space only
/G Follow government rules for wiping
/LOG Format output for file or printer
/R*n* Repeat wiping *n* times; default *n* = 1
/V*n* Wipe value; default *n* = 0

WIPEFILE *filespec* [*switches*]

Switches

/G Follow government rules for wiping
/LOG Format output for file or printer
/N Non-wiping mode
/P Pause before wiping each file
/R*n* Repeat wiping *n* times; default *n* = 1
/S Wipe or delete files in subdirectories
/V*n* Wipe Value; default *n* = 0

The Norton Utilities
Advanced Edition / 4.0

ASK
Create interactive batch files

BEEP
Sound a tone, or play a tonefile

DS—Directory Sort
Sort file directories

DT—Disk Test
Test disks for physical errors

FA—File Attributes
Set, reset, and scan file attributes

FF—File Find
Find lost files and directories

FI—File Info
Attach and view filename comments

FR—Format Recover (Advanced Edition)
Recover an accidentally formatted hard disk

FS—File Size
List or total file sizes

LD—List Directories
Graphic display of directory tree

LP—Line Print
Print files with line numbers and headers

NCD—Norton Change Directory
Graphic directory navigation
Short-hand command-line Change Dir

NI—Norton Integrator
Integrated environment—Point and Shoot

NU—The Norton Utility
Full disk exploration and editing
Powerful UnErasing tools
Edit Directory and FAT (Advanced Edition)

QU—Quick UnErase
Automatic recovery of erased files

SA—Screen Attributes
Set screen colors and attributes

SD—Speed Disk (Advanced Edition)
Reorganize disks to increase performance

SI—System Information
Report performance indexes and other info

TM—Time Mark
Automatic timing with four stopwatches

TS—Text Search
Locate text in files, disks, and erased files

UD—UnRemove Directory
Recover removed directories

VL—Volume Label
View, add, change and delete disk labels

WIPEDISK
Overwrite entire disks, for maximum security

WIPEFILE
Obliterate files, for maximum security

Command Formats

ASK "*prompt*", [*key-list*]

Returns errorlevel code equal to the position of the chosen key in key-list. Arrange branches in descending order of errorlevel codes.

BEEP [*switches*]
or
BEEP [*filespec*]

Switches

/D*n*	Duration of the tone in *n*/18 seconds
/F*n*	Sound a tone of frequency *n*
/R*n*	Repeat the tone *n* times
/W*n*	Wait between tones *n*/18 seconds

DS [*directory name*] ⇒ *Full-screen mode*
or
DS *sort-key(s)* [*directory name*] [/S]

Sort keys

N—name	D—date
E—extension	S—size
T—time	– reverse sort order

DT [*d*:] [*filespec*] [*switches*]

Switches

/B	Perform both a disk test and file test
/C*n*	Mark Cluster *n* as bad
/C*n*–	Mark Cluster *n* as good
/D	Test entire disk
/F	Test files only
/LOG	Format output for printer or file
/M	Relocate doubtful clusters
/S	Test subdirectories also

FA [*filespec*] [*attribute switches*] [*switches*]

Attribute switches

/A	archive	/R	read-only
/HID	hidden	/SYS	system

Follow switch with + or – to set or reset.

Switches

/Clear	Clear all attributes
/P	Pause mode
/S	Act on subdirectories
/T	Show totals only
/U	Unusual files—any attribute set

FF [*d*:] [*filename*] [*switches*]

Switches

/A	Search for files on All drives
/P	Pause mode
/W	List files in a Wide format

FI [*filespec*] [*comment*] [*switches*]

Switches

/C	List only files with comments
/D	Delete filename comment.
/E	Edit or add a comment (full-screen)
/L	Long format
/N	Not 100% IBM-compatible
/P	Pause mode
/Pack	Compress the FILEINFO.FI file
/S	Include subdirectories in listing

FR [*d*:][/SAVE]

Switch

/SAVE	Save reformatting information

FS [*filespec*] [*target-drive*:] [*switches*]

Switches

/P	Pause mode
/S	Include subdirectories
/T	Display totals only

LD [*d*:] [*pathname*] [*switches*]

Switches

/A	List the directory of all drives
/G	Graphic directory tree
/N	Non-IBM compatible printer
/P	Pause mode
/T	List totals only

LP *filespec* [*where-to-print*] [*switches*]

/N	Number lines	/P*n*	Page start #
/T*n*	Top margin	/S*n*	line Spacing
/B*n*	Bottom margin	/80	80-col print
/L*n*	Left margin	/132	Condensed
/R*n*	Right margin	/EXT	extended chars
/H*n*	page Height	/HEADER*n*	Level
/W*n*	page Width	/EBCDIC	coding

/SET:*filespec* File of Lotus-style setup strings

NCD [*end-point directory name*][/R]
or
NCD MD *dirname*
or
NCD RD *dirname*

Switch

/R	Reread disk directory

NU [*filespec*][*switches*]

Switches

/D0	Driver for 100% IBM compatibles
/D1	Screen driver for BIOS-compatibles
/D2	ANSI.SYS screen driver
/Bn	Set Background color; n=0 to 15
/Fn	Set foreground color; n=0 to 15
/BW	Use with mono monitor on a CGA
/EBCDIC	Use EBCDIC encoding
/EXT	Display EXTended characters
/TV	Topview-compa[illegible]e environments
/M	Maintenance mode (Advanced Ed.)