Proposal for the Input and the Notation of the System.

0. Introduction.

Input of the system will take place in two stages.

From the written documents we shall punch on Flexowriters the tape HACOSYS (Hand Coded System), an assembler will read HACOSYS and punch the tape BICOSYS (Binary Coded System). The tape BICOSYS will load the system into a virgin machine. BICOSYS, which will be handled daily by the operator will be very much shorter than HACOSYS, which is only handled at system modification.

1. The General Structure of BICOSYS.

Paper tape can be read

1) for the sake of initial input into the virgin machine under complete control of Charon
2) under normal program control (with Charon as program controlled slave).

We call the two forms of input primary and secondary input.

Primary input is unchecked; for this reason we shall restrict it to a minimum, i.e.

1) an interrupt jump in M[24] to the starting address of the secondary input program
2) the secondary input program itself. The latter will be stored in a future core page.

The tasks of the secondary input program are the following ones

1) It has to verify its correct presence in core store, say by means of a sum check. It has to clear the remainder of the memory.
2) It has to clean all the necessary teeth in the normal manner but for the real time clock, which has to remain dormant (say by LVCA <u>false</u>)
3) All LVIF's can be set in their future position and IV can be set <u>true</u>
4) Now the secondary input program starts to read paper tape over the golden reader in a most unorthodox manner, viz.
   4.1) it will give tape read commands directly to the golden reader
   XX 4.2) it will keep the IFT-value negative, it will not rely on the interrupt mechanism but will inspect the IFT increases.
5) Secondary reading will consist of two phases, core filling, followed by segment filling.

During core filling all core locations with permanent meaning are filled. They are filled in such a way that at the end of core filling the primary input program looks as a program in PM1. All mechanisms to dump segments are then present, the clock, if necessary, can be started and secondary reading continues to fill the segments. The multirunning system is then working but for the information streams.

6) Finally the golden reader is set in accordance with the initial state of the corresponding CM and PM1 ends its activity as normal programs do.

The information on BICOSYS consists of

a) the text of the secondary input program
b) the information to be read during core filling
c) the information to be read during segment filling.

2. The Structure of HACOSYS.

HACOSYS will consist of two parts
1) the Table, followed by
2) the Contents, the latter subdivided into
   2.1) the secondary input program
   2.2) the core information
   2.3) the segment information

2.1. The Structure of the HACOSYS Table.

The identifiers occurring on the HACOSYS Table identify parameters, the HACOSYS Table specifies their constant type and value according to rules to be explained in this section. The identifiers have the complete HACOSYS Contents as their scope.

They are of three different types
1) constants
   1.1) fixed constants
   1.2) derived constants
   1.3) invariant addresses
2) dynamic addresses
3) static addresses

On the HACOSYS Table they are introduced in the following order
1) the fixed constants
2) the dynamic addresses
3) the static addresses, the invariant addresses and the derived constants in order of increasing CCA (Current Core Address)

The CR (if desired,preceded by comment introduced by quotes) acts as separator.

2.1.1. The Fixed Constants.

The fixed constant is given by its identifier, followed by "=", followed by its value; its value -a single word- may be given by a (signed or unsigned) decimal integer or by a (signed or unsigned) octal number (at most 9 octal digits enclosed within apostrophes). Then one or more separators have to follow.

2.1.2. The dynamic Addresses.

Dynamic addresses are defined in terms of the value of CDA (Current Dynamic Address); CDA is initialized by "<u>d</u> p[q]", where "p" (unsigned decimal integer) and "q" (decimal integer) specify a dynamic address in the usual manner. Then one or more separators must follow.

A dynamic address is then given by its identifier, followed by an unsigned decimal number enclosed within parentheses. The value of CDA is then assigned to the parameter identified, whereafter CDA is increased by the amount given within the parentheses. One or more separators follow.

2.1.3. The Static Addresses.

The static addresses can only be given after the first CCA initialization, that specifies the value of CCA (Current Core Address)

The CCA initialization consists of c, followed by an address value, given as unsigned (decimal or octal) number, followed by one or more separators. This value is the next CCA value, CCA must be monotonically non decreasing.

A static addres is then given by its identifier, followed by an unsigned decimal number within parentheses, followed by one or more separators. The value of CCA is assigned to the parameter identified, whereafter CCA is increased by the amount given within the parentheses.

2.1.4. The Derived Constants.

After the definition of a static address, one or more "derived constants" may be given. Each derived constant is given by its identifier, followed by "=", followed by a word, given as decimal or octal number. The value of the derived ~~word~~ constant is this word increased by the static address value just assigned to the last static address. (In a -be it clumsy- manner, this facility enables us to introduce "SE1" with the value "SUBCD(:PSE1)" as soon as PSE1 has been defined.)

2.1.5. The Invariant Addresses.

When CCA points to a location that shall contain an SV, then the moment has come to give the invariant addresses, related to that corresponding segment.

By "s" invariant address definition is announced and the variable CLN(Current Line Number) is set to zero. From then onwards invariant addresses related to this segment can be given by giving their identifier, followed by an unsigned decimal number between parentheses.

The value assigned to the patameter is the invariant address, composed of CCA and CLN (which must be less than 512), whereafter CLN is increased by the amount given within the parentheses. Invariant address definition with respect to this segment is ended~~X~~ by "t", which ~~XXXX~~ causes CCA to be increased by 1.

The individual invariant address definitions belonging to the same segment must be separated by one or more separators. Following ~~XXX~~ "s" and preceding "t" no separator is required.

2.2. The Structure of the HACOSYS Contents.

On the ~~XXXX~~ HACOSYS Contents we find in order on the tape
1) the primary input
2) the core information
3) the segment information.

2.2.1. The Primary Input.

A piece of primary input is announced by "p", followed by an unsigned (decimal or ~~XX~~ octal) number or a static address, ~~XXXXXXXX~~ (i.e. the identifier of a static address) indicating the starting address, followed by an unsigned decimal number between parentheses indicating the number of words that follow, followed by one or more separators. Then the succesive words will follow, all separated from one another.

On account of such part of the HACOSYS Contents, a part of BICOSYS will be made, intended for primary input, on successive locations, starting at the address given. This BICOSYS portion will contain at the end a word more, computed in such a way that the total sum of this portion is = 0.

A number of such pieces of primary input may follow. (E.g. one for M[24] and one for the secondary input program).

2.2.2. A Piece of Core Filling.

Such a piece is announced by the identifier of its static starting address, followed by an unsigned decimal integer between parentheses, indicating the number of words that follow. Then the words follow, announcements and words all followed by one or more separators.

Many such pieces may follow.

2.2.3. A Piece of Segment Filling.

Such a piece is as a piece of core filling but for the fact that it is announced by the identifier of an invariant address. Many such pieces may follow.

2.2.4. The End of HACOSYS.

An "e" indicates the end of HACOSYS.

2.2.5. The Words.

We shall now describe how the words on the HACOSYS Contents are written down. They are of two different layouts, called "constants" and "instructions". In the following description an "optional sign" is "+", "-" or "empty"; an octal number is an apostrophe, followed by at most 9 octal digits, followed by an apostrophe.

2.2.5.1. The Layout of HACOSYS constants.

The following 6 basic forms are admissible
a) an optional sign, followed by an unsigned decimal integer
b) an optional sign, followed by an unsigned octal number
c) an optional sign, followed by the identifier of a fixed constant
d) an optional sign, followed by the identifier of a derived constant
e) an optional sign, followed by the identifier of an invariant address
f) an optional sign, followed by a colon ":", followed by the identifier of a static address.

All six may be followed by the optional increment; this consists of (zero or more times) a "[", followed by one of the six constant formats, follwed by "]".

Remark 1. The character sequences "][+" and "][-", that may thus arise may be contracted into "+" and "-" respectively.

Remakr 2. If the constant starts with a minus sign, this minus sign applies to the complete constant: all increments are added (mod 2 ↑ 27 - 1) and at the end the result is inverted.

2.2.5.2. The Layout of HACOSYS instructions.

The instruction notation is an adaptation of the ELAN conventions.

Instructions with an address of type "STAT" may have this address in the form
a) M
b) the identifier of a static address

followed by the optional increment as described in 2.2.5.1. (The identifier "M" is regarded as static address = 0). The resulting address must ~~XX~~ fit into the 15 bits available.

Instructions with an address of type "STATB" may have this address in the form
a) M[B]
b) identifier of a static address, followed by "[B]

followed by the optional increment (also after "B", the contraction convention applies). The resulting address must fit into the 15 bits available.

instructions with an address of type "DYN" may have this address in the form
a) MG, MA, MS, MC, MT, MD, Mp
b) the identifier of a dynamic address

followed by the optional increment.

In the above cases the corrections are applied to the future address part of the instruction.

The addres operand of type ":DYN" has the form ":" followed by one of the forms dynamic address just described; the adress operand ":STAT" has the form of an unsigned constant (see 2.2.5.1). Both followed by the optional increment.

All increments that follow are added to the unsigned operand thus described. The result must be less than 32768 in absolute value, ~~X~~ an initial plus or minus sign applies to the result thus obtained, the resulting sign is processed in building up the instruction part.