The Software System for controlling the RF Power Plant.

E. Bracke

Preface.

When joining the SPS ARF-po section in spring 1978 I was asked by H. P. Kindermann, my section leader, to see if something could be done to improve the remote control of the RF Power Plant. The powerplant then consisted of 3 Siemens 500kW / 200MHz transmitters providing the required RF power to the 3 accelerating cavities of the SPS. Indeed, in the foreseeable future, this powerplant was to be extended, first with another Siemens transmitter and a cavity and later even with another set of 4 transmitters and one of 2 transmitters of make Philips and Valvo respectively.

User friendly control was one of the main demands of my 'customers', my direct colleagues. In those days control of the SPS equipment went via a star formed network of Nord mini computers, installed in all service buildings around SPS. One of which was the RF computer; in the star's center was the Message Handling Computer (another Nord machine) that did the communication between the star 'branches'. Later this network grew out to a net of several of these stars. One branch was e.g. the Library machine that had a 'huge' (ahem) drum (and later harddisk) as mass storage device where all application programmes were stored. Other branches were the Main Control Room Console machines. From the (in the beginning: ASR33 teletype) console of e.g. the RF computer an application could be launched on the latter but it was first delivered by the Library machine before it could be started. The programme, written in an interpreter language called NODAL (extended language from originally DEC's FOCAL - FOrmulaCALculator interpreter for PDP 8 and up computers), ran in the so called Nodal buffer, which was quite small. Too small to house and execute comfortable, user friendly, programmes...

Computer access to the hardware passed via an equipment multiplexer that addressed the hardware by sending and retrieving data to or from it. This multiplexer was hooked up to the computer via CAMAC and interfaced via its driver to Data Module Subroutines which in turn presented themselves as simple, 2 or 3 parameter subroutines or functions (Nodal 'Elements') in the Nodal programming language. They were integrated into the operating system of the computer to which the hardware to be controlled was connected and thus only known in that computer. The DMS served mainly as the input / output model of the hardware, possibly with some data normalizing; generally little control algorithm however. The equipment groups, like e.g. the RF group, had the responsibility for the remote computer access interface and application programmes starting at the DMS level upwards. DMS'es were written in the Nord's assembly language.

It was also possible to 'export' programme parts to a remote computer and had them executed over there. This was required because some of the programme parts had to act on hardware that physically existed only in a specific machine. In this way it was possible for the MCR Console machines, which had no SPS hardware connected to them, to control remotely this hardware never the less.

So I sat down and devised a 3 layered scheme, keeping in mind that my programmes needed to configure themselves while executing: blocks that were not needed anymore could be deleted, freeing valuable buffer space, and new blocks could thus be loaded. The 3 layers consisted of a presentation layer (rich text for human interface), then a process layer where the programme's control algorithm executed on a virtual model of the RF hardware and finally the layer that created, from the DMS model of the hardware, the aforementioned virtual model required by the process layer. By making the third layer 'exportable' any computer in the network that ran an RF application programme could have this 'typical RF' part sent over to the RF computer (the only one in the net that 'knew' about that RF equipment), have it executed, controlling RF hardware and let it send back the data that was asked by the application programme.

Today, modern control systems have client / server connections and the like but currently (and in the future for LHC with FESA etc.) the control of RF hardware is still done with this 3 layered control model. Still we virtualize hardware to a functional 'job-relative' part, still the 'equipment-relative' software part is executing in a 'FrontEnd' computer, as we call these in 2005 and still we present the data in a user friendly way with a 'main / sub program' control application programme to our customers.

I had fun porting this note from the original (still in my possesion) paper note, nothing changed in the text and with its modern (!), 8 needle, lineprinter output scanned into this document.

Have fun reading it!

Erik Bracke (AB/RF-cs) 17th march 2005.

TABLE OF CONTENTS.

1 INTRODUCTION....4

2 PHILOSOPHY....4

3 DEFINITION OF THE MODULES. 4

3.1 THE PROGRAM SELECTOR. 4
3.2 MAIN PROGRAMS. 5
3.3 JOB-RELATIVE SUBROUTINES. 5
3.4 EQUIPMENT-RELATIVE SUBROUTINES. 5
3.5 SUB-PROGRAMS. 5
3.6 MESSAGE OVERLAY. 6

4 EXAMPLE....6

5 APPENDIX I. 7

6 APPENDIX II. 8

7 APPENDIX IIIA. 9

8 APPENDIX IIIB....10

The Software System for controlling the RF Power Plant.

1 Introduction.

This description gives the philosophy behind the software system for the RF power plant, and shows the way in which this system has been realized.

2 Philosophy.

Although in principle intended for use in the RF section, the control software for the RF power plant has been conceived computer-independent. This implies that control of RF equipment can also be done with the software system from the main control room and thus could be useful to the Operations Group.

As far as possible a modular approach was adopted in order to maintain a "one job - one program" relation in spite of transmitters of (in the near future) three different makes, each one having its own characteristics.

The best way of explaining the philosophy of conception of the software system might be by defining the modules from which the system is composed and by giving a description of each of their particular tasks.

3 Definition of the modules.

3.1 The Program Selector.

The program selector is the unique entry point to the software system. Its existence makes it unnecessary for the operator to remember N programmes for N jobs to be executed. (One is sufficient - the name of the program selector itself.)

The program selector asks the operator in plain language which job (program) is to be executed and then starts the relevant program.

The program selector can be called by:

```
RUN <92>RF:PCS
```
(**RF:P**owerplant **C**ontrol program **S**elector)

The control program selector occupies line numbers 1 to 9 inclusive in the buffer.

Appendix I shows the way in which the program selector presents itself to the operator. At present time, there is space for 70 different programs.

3.2 Main programs.

These are the software descriptions of the jobs needed for execution by the computer. The analysis of the problem (job) has resulted in an algorithm which forms the basis of the main program.

The algorithm itself can be found in flow-chart form in the documentation of each main program.

Main programs occupy line numbers 10 to 25 inclusive in the buffer.

3.3 Job-relative subroutines.

If a main program has to execute the same task several times on the hardware (transmitters) or its associated files or for readability reasons, a subroutine is dedicated to this task. Several main programs may also share the same job-relative subroutine.

The job-relative subroutine, once called upon by the main program, executes its task and, in general, checks the hardware to see whether the task has been executed. Then it passes the status of the execution (e.g. task executed) or other data relative to the task on to the main program (feedback from the point of interest: the hardware).

The algorithm of the execution of each task can be found in the documentation of the job-relative subroutine. Job-relative subroutines occupy line numbers 34 to 26 inclusive in the buffer.

3.4 Equipment-relative subroutines.

These subroutines are necessary to make it possible for a job-relative subroutine to operate on three different makes of transmitter. This means that for each job-relative subroutine operating the hardware there must be three corresponding equipment-relative subroutines.

The equipment-relative subroutine (job independent) forms the interface between job-relative subroutine (equipment independent) and the actual hardware (transmitter).

One can thus conclude that equipment-relative subroutines are slaves of job-relative subroutines. They communicate with each other through variables. Equipment-relative subroutines occupy line numbers 65 to 50 inclusive in the buffer.

3.5 Sub-programs.

These are program parts which are usually called only once by a main program. They can be considered as the interface between the human operator and the main program (machine), or vice versa.

Due to their dedication they contain lots of text printing and therefore occupy much buffer space.

Examples of sub-programs are, e.g. initial dialogues and status overviews. The first one gathers information by means of plain text from the operator about the job he wants the computer to perform and compresses this into variables, understandable by the main program. The second one expands the information (which is gathered and stored in variables by the main program), into plain text and presents this to the operator. Once executed a sub-program can disappear in order to release the buffer space it occupies for, for example, a job-relative subroutine.

Sub-programs use line numbers 35 to 49 inclusive in the buffer. Appendix II gives and example of an initial dialogue.

3.6 Message overlay.

The message overlay is a program capable of printing a short message (one sentence) on the interactive terminal. The message overlay is used by a main program (or its job-relative subroutines) every time a block of its algorithm has been executed, thus informing the operator about the current status of the main program.

The message is determined by the main program and contained in a program-relative message pack which is loaded by the overlay.

The information of the main program is passed to the overlay through task global variables (arguments 1, 2 and 15, 16).

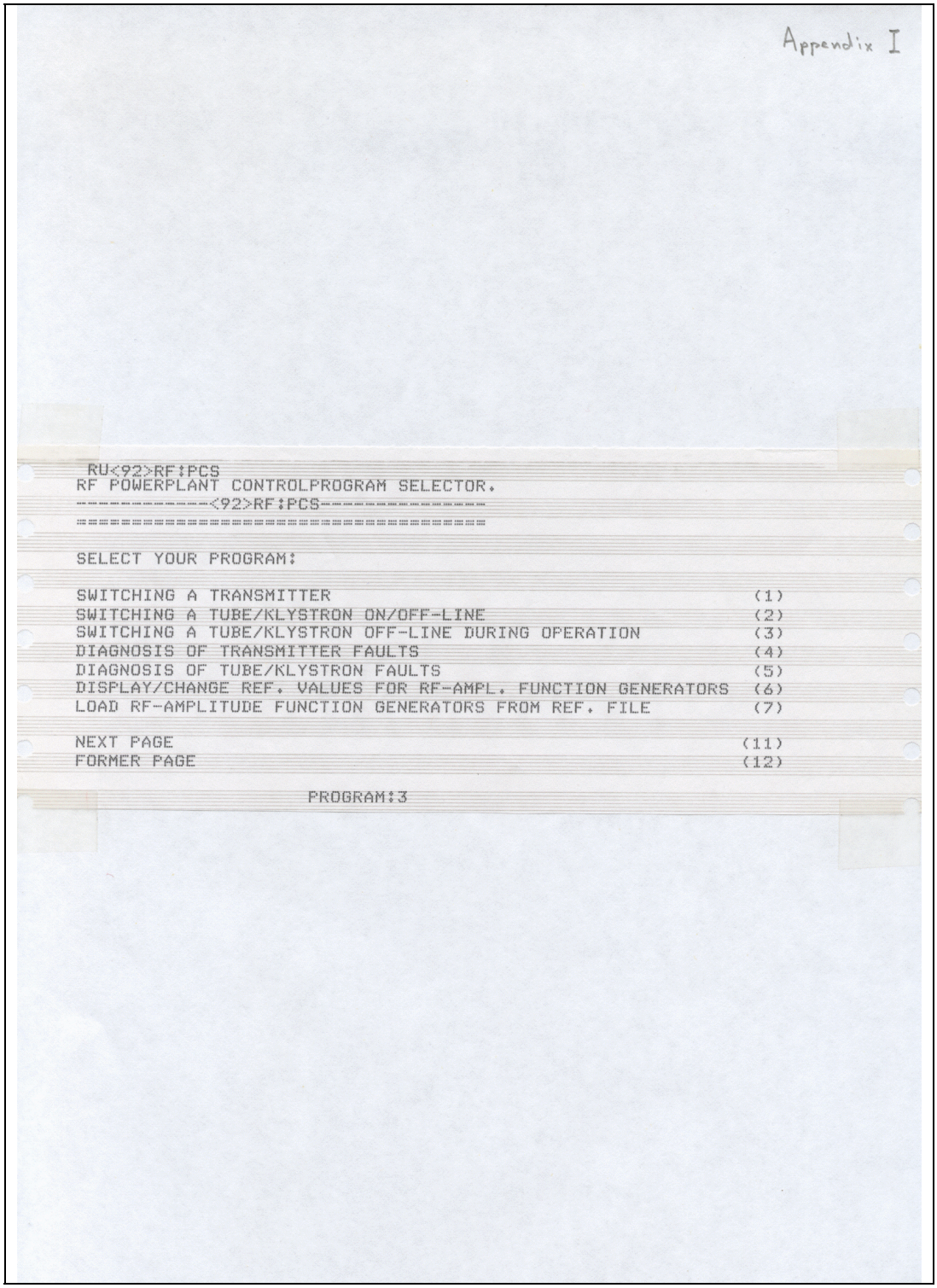
Using the message program as an overlay makes it disappear with the message pack, once executed, thus freeing the occupied buffer space.

The message overlay, together with the program-relative messge pack, uses line numbers 15, 16 and 17 in the buffer. Being in overlay mode, it does not however interfere with the same line numbers used in the main program.

4 Example.

Appendix III a & b give an example of a program execution. Underlined text is typed in by the operator.

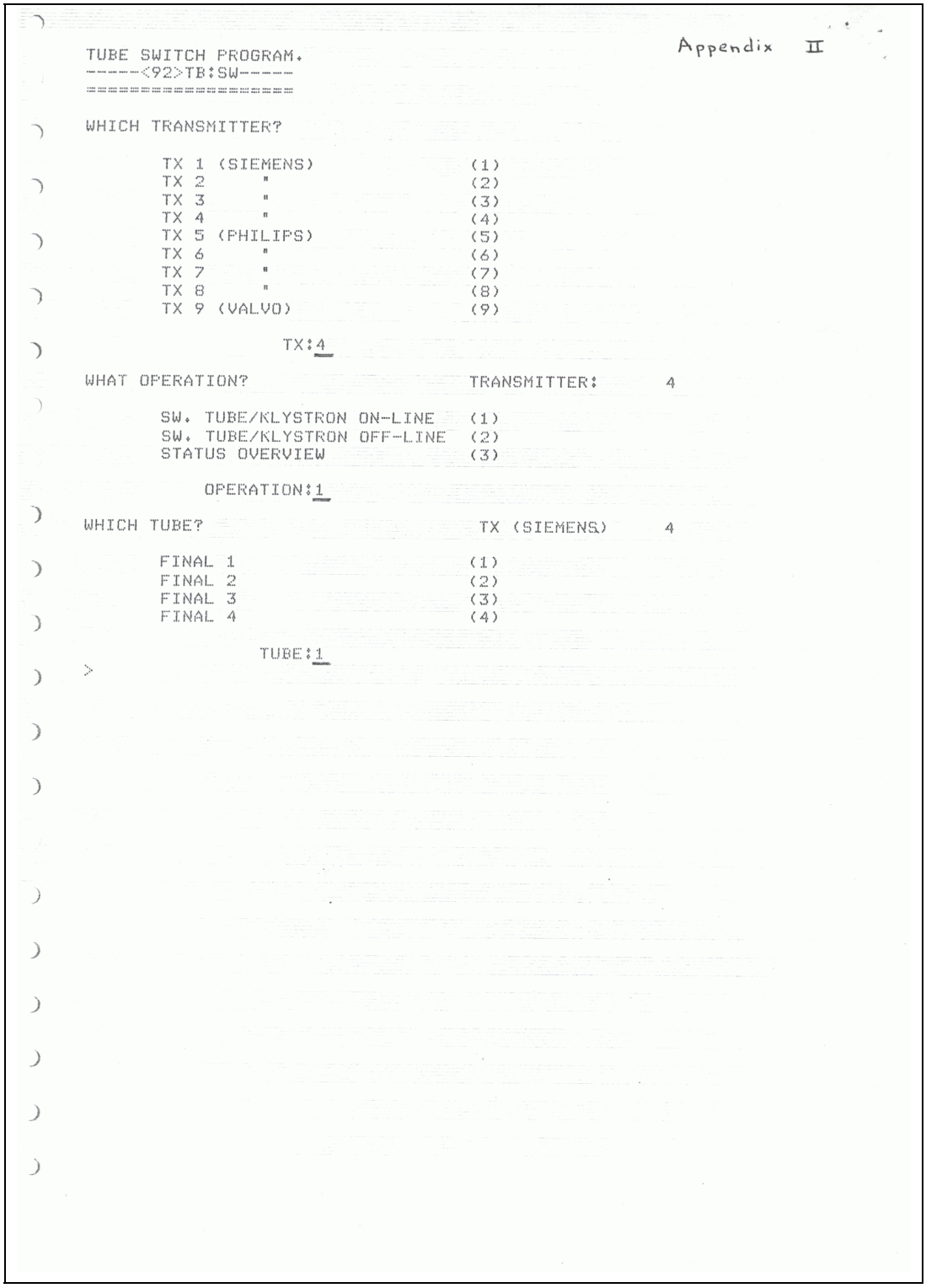
5 Appendix I.



Appendix I

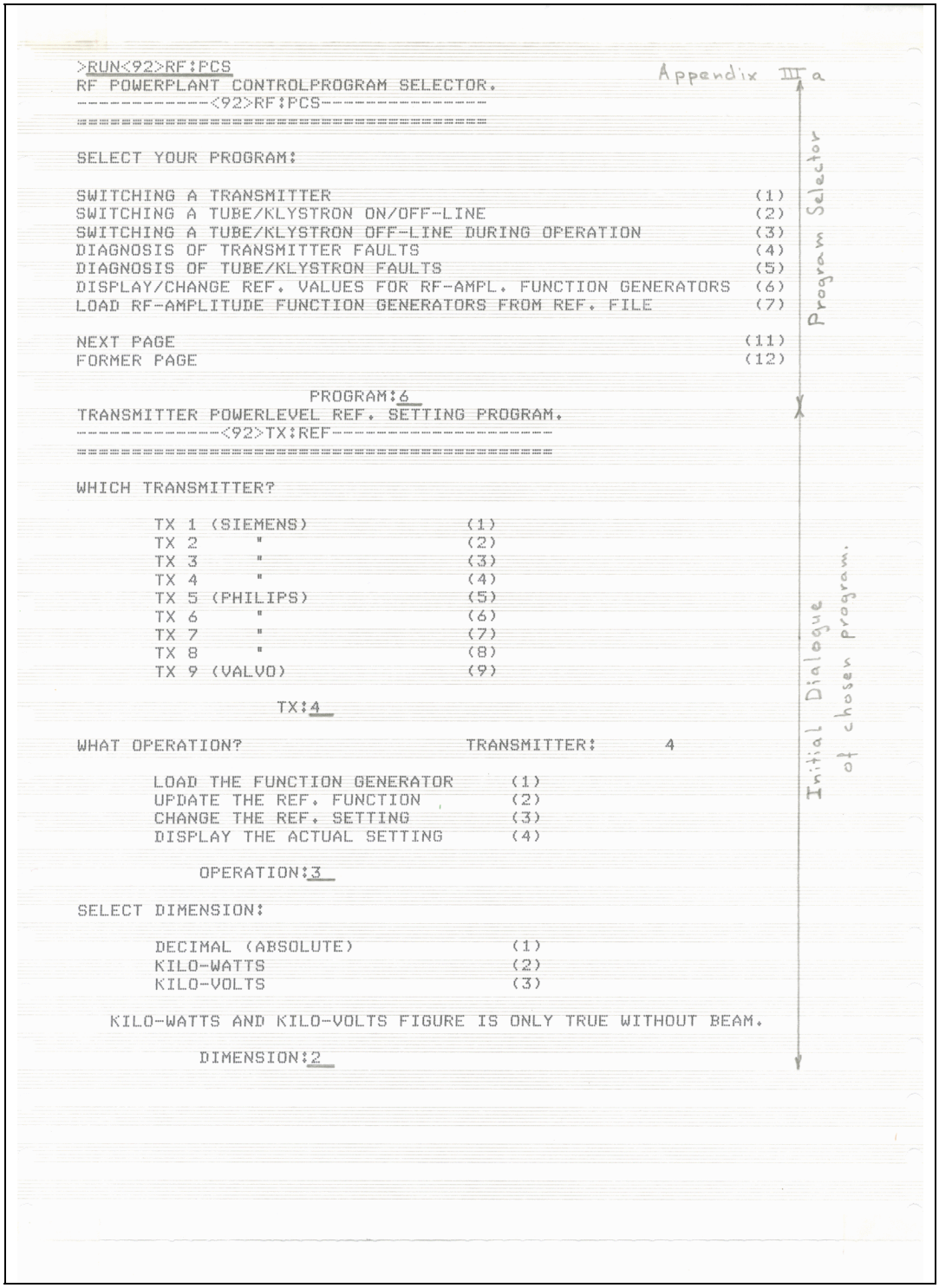
```
 RU<92>RF:PCS
RF POWERPLANT CONTROLPROGRAM SELECTOR.
~~~~~~~~~~~~~~~~~~~~<92>RF:PCS~~~~~~~~~~~~~~~~~~~~
==========================================================

SELECT YOUR PROGRAM:

SWITCHING A TRANSMITTER                                    (1)
SWITCHING A TUBE/KLYSTRON ON/OFF-LINE                      (2)
SWITCHING A TUBE/KLYSTRON OFF-LINE DURING OPERATION        (3)
DIAGNOSIS OF TRANSMITTER FAULTS                            (4)
DIAGNOSIS OF TUBE/KLYSTRON FAULTS                          (5)
DISPLAY/CHANGE REF. VALUES FOR RF-AMPL. FUNCTION GENERATORS (6)
LOAD RF-AMPLITUDE FUNCTION GENERATORS FROM REF. FILE       (7)

NEXT PAGE                                                 (11)
FORMER PAGE                                               (12)

                    PROGRAM:3
```

6 Appendix II.



```
                                                        Appendix II

TUBE SWITCH PROGRAM.
------<92>TB:SW------
==========================

WHICH TRANSMITTER?

        TX 1 (SIEMENS)              (1)
        TX 2      "                 (2)
        TX 3      "                 (3)
        TX 4      "                 (4)
        TX 5 (PHILIPS)              (5)
        TX 6      "                 (6)
        TX 7      "                 (7)
        TX 8      "                 (8)
        TX 9 (VALVO)                (9)

                    TX:4

WHAT OPERATION?                     TRANSMITTER:      4

        SW. TUBE/KLYSTRON ON-LINE   (1)
        SW. TUBE/KLYSTRON OFF-LINE  (2)
        STATUS OVERVIEW             (3)

             OPERATION:1

WHICH TUBE?                          TX (SIEMENS)     4

        FINAL 1                     (1)
        FINAL 2                     (2)
        FINAL 3                     (3)
        FINAL 4                     (4)

                  TUBE:1
>
```

7 Appendix IIIa.



8 Appendix IIIb.