August 99

6. Programming the Test Set XS92b

6.1. Program instructions (syntax) 6-2
6.1.1. General 6-2
6.1.2. Programming language structure 6-3
6.1.3. General 6-3
6.1.4. Assigning values to parameters 6-4
6.1.5. Conversion of parameters 6-6
6.1.6. Program branches and loops 6-7
6.1.7. Sub-routines 6-10
6.1.8. Instructions for controlling injection 6-11
6.1.9. Printing instructions 6-12
6.1.10. Examples of test programs 6-14

6.2. Writing a program 6-19
6.2.1. EDT mode (editor) 6-19
6.2.2. Editing a program using a text editor 6-21
6.2.3. Printing programs 6-21
6.2.4. Executing programs 6-22

6.3. Program management 6-24
6.3.1. Initialising the memory unit YS92 6-24
6.3.2. Storing a program 6-25
6.3.3. Program directory 6-26
6.3.4. General reset 6-26

6.4. Terminal control 6-27
6.4.1. Principle 6-27
6.4.2. TDM functions 6-27
6.4.3. Line input 6-27
6.4.4. Switching between terminals 6-28
6.4.5. Kind of terminal 6-29
6.4.6. Line termination 6-29
6.4.7. Alphabetic list of control codes and commands 6-30

6. Programming the Test Set XS92b

6.1. Program instructions (syntax)

6.1.1. General

The test set XS92b can be programmed very simply by the user to execute a series of test instructions automatically which otherwise would have to be entered manually one after the other for every test. For this purpose, there is a list of program instructions from which user test programs can be assembled and entered line by line via a (handheld) terminal connected to the test set.

When using the operator program **PCX92** or **XSWIN**, the functions for entering the parameters are accessed by clicking on the "Terminal mode" button.

Test programs can also be written with a text editor such as Notepad or Wordpad (see Section 6.2.2.).

The format of these instructions (also referred to as statements) corresponds largely to the way in which the test set would be controlled manually.

Instructions are also included for converting parameter values, introducing conditional or compulsory branches, looping certain parts of a program and calling sub-routines.

Some test set modules are specifically designed to support the creation and handling of user programs:

"EDT"	an editing function for creating, correcting and changing programs
"STO"	enables complete programs to be stored on the memory board YS92
"RUN"	starts the execution of a program.

The following Sections describe this and other parts of the test set in more detail.

6.1.2. Programming language structure

6.1.3. General

A program is a list of individual statements with each statement on a separate line. It must have a 'name' comprising a maximum of eight characters, the first of which must be a letter and the remainder any desired character.

The **beginning of a program** is simply its first line, but the end must be especially indicated:

End of program EP "End of program".

EP is then the last line of the program.

Following certain commands, several statements can be combined to form an instruction block.

Instruction block	BEG	"begin"
	END	"end"

An instruction block is treated as a single statement, e.g. after a condition, in the case of a loop or when calling a complete subroutine.

BEG marks the beginning of an instruction block
END marks its end.

Structure:

```
BEG
    statement
    statement
    •
    •
    statement
END
```

Example:

```
BEG
U = 20.5
DU = 1.0
A = 2
END
```

A maximum of 30 instruction blocks can be boxed inside each other.

Example:

```
BEG
    DU=1.0
    ST=-1
    A=2
    TI=2
     REP 3            program loop (see Section 6.1.6.)
       BEG
         U=20.5
         F
         PHI ADD 30
       END
END
```

The programs must be properly structured. It is not permitted to leave instruction blocks, boxed routines or loops using a GOTO statement, except when the target (label) address is not in a loop or instruction block.

6.1.4. Assigning values to parameters

All parameters which are required for a particular set of injection signals must have a value assigned to them.

Assignment:

PAR = xx.x "parameter" = "value"

Example:

ZS = 20.5 ZS = source impedance

"PAR" can be any one of the parameters given in the lists in Section 4.3. and "xx.x" any one of the values in the ranges given.

Assignment of values using the terminal

When writing a program, a parameter may be entered without assigning a value to it using the form:

PAR = "parameter" =

Example:

DZL =

In this case, the processor will execute the program once started up to this point and then wait for a value to entered via the terminal (terminated with "ENTER") before continuing.

Assignment of constants

The programming language permits the assignment of values to constants. The following designations are available for this purpose:

N0 to N9	for integers (no decimal places)
X0 to X9 Y0 to Y9 Z0 to Z9	for true form values (with decimal places)

The assignment syntax is the same as for parameters:

CON = xx.x "constant" = "value"

Example:

Y4 = 21.4 or N1 = 324

The values of constants do not vary during the course of a program as do the values of parameters. Constants enable starting values to be transferred or intermediate results to be stored.

Transferring values

The variable "DUM" has to be used in order to transfer values between constants and parameters, or between parameters:

DUM = CON PAR = DUM	The value of the constant is transferred to the parameter.

These two lines must be written consecutively.

Example:

X1 = 60.0	constant for true form values = 60.0
PZL = 30.0	line impedance angle = 30°
DUM = X1	
PZL = DUM	PZL is now = 60°.

Like parameters, constants can be compared with a value or converted by any of the four basic mathematical operations when part of an IF condition (see Sections 6.1.5. and 6.1.6.).

Assignment of substitution variables

If a function or a sub-routine is needed several times, but with different parameters, variables (VAR) may be inserted.

The following designations are reserved for these variables:

V0 to V9

The syntax is as follows:

VAR = PAR "variable" = "parameter"

Once assigned, VAR and the corresponding parameter are identical and VAR assumes the value of the parameter.

A substitution variable can be assigned to different parameters one after the other and then processed in a sub-routine (see Section 6.1.7.).

Example:

```
Main program:   V1 = ZL
                CAL SUBPROGR
                V1 = ZS
                CAL SUPPROGR
                    •
                    •
                    •
Sub-program:    SBR SUBPROGR
                BEG
                V1 ADD 1.0
                END
```

In the above example, the values ZL and ZS are increased one after the other by 1.0.

6.1.5. Conversion of parameters

Addition of an amount to a parameter.

PAR ADD xx.x "parameter" "addition" "value"

The amount xx.x is added to the original value of the parameter.

Example:

```
I = 5.0
I ADD 1.5        subsequently I = 6.5 A
```

Subtraction of an amount from a parameter

PAR SUB xx.x "parameter" "subtraction" "value"

The amount xx.x is subtracted from the original value of the parameter.

Example:

I = 5
I SUB 2.2 subsequently I = 2.8 A

Multiplication of a parameter by a factor.

PAR MUL xx.x "parameter" "multiply" "value"

The value of the parameter is multiplied by the factor xx.x

Example:

I = 5.0
I MUL 3 subsequently I = 15.0 A

Division of a parameter by an amount.

PAR DIV xx.x "parameter" "divide" "value"

The original value of the parameter is divided by the amount xx.x.

Example:

I = 5.0
I DIV 2.5 subsequently I = 2.0 A

6.1.6. Program branches and loops

Program branches can be initiated by parameter values. In this case, the parameter concerned is compared with a specified value. If the truth criterion is fulfilled, the next program instruction is executed, otherwise it will be omitted.

An "ELSE" branch may also be written which is executed when the truth criterion is not fulfilled.

```
IF PAR cond xx.x "if" "parameter" "condition" "value"
    statement
•
•
```

or

```
IF PAR cond xx.x
    statement
ELS
    statement
•
```

PAR can be:
- a parameter name or a constant
- a substitution variable
- "LM" standing for limiting value.

‘cond’ defines the particular condition and can be one of the following:
- EQ equal to
- GT greater than
- GE greater or equal to
- LT less than
- LE less or equal to
- NE not equal to

xx.x is a constant numerical value.

An instruction block may be entered in place of statement which starts with "BEG" and finishes with "END" and may contain further "boxed" routines.

Checking for limits

The limit of a main parameter can be checked by setting an "IF" condition.

IF LM EQ 1

"1" is not a value in this case, but simply symbolises the logical state "TRUE". "0" indicates the logical state "FALSE". This means that the above condition is fulfilled when the limiting value is reached and the next statement (or instruction block) is being executed.

A further possibility is:

IF LM NE 0

"LM" means "limit".

LM itself always designates the limit of the parameter which was varied by the last step parameter to be activated. Whether the upper or lower limit is checked depends on whether the parameter is being varied upwards or downwards.

Program loops enable an instruction to be executed a specified number of times. A program loop will usually contain several statements forming an instruction block.

```
REP n              "repeat" "number"
   statement
```

or

```
REP n
   BEG
         statement
            •
            •
         statement
   END
```

"n" determines the number of repeats (1 to 999).

Aborting a loop:

```
BRK                "break"
```

This statement enables a program loop to be terminated prematurely, e.g. following an "IF" criterion. The program then jumps to the next statement after the interrupted loop. It is only permissible to use the BRK instruction within a loop.

Compulsory program jump:

```
GOTO Lnnn          "go to" "label" "address"
```

nnn is a constant from 0 to 999 and Lnnn defines an address in the program where it continues.

```
Lnnn               "label" "address"
```

Caution!

The program label may not occur within a loop or an instruction block (see Section 6.1.3.). Thus "GOTO" cannot be used to jump to a point in a loop or an instruction block, but it does permit an exit from a loop or an instruction block. This restriction forces correctly structured programming.

6.1.7. Sub-routines

Attention has to be paid when writing sub-routines to defining the parameters used in them, otherwise the corresponding last value of the main program will be used.

However, there is the possibility of using substitution variables (see Section 6.1.4.) which can be assigned to different parameters in the main program.

```
SBR name        "subroutine" "name"
    statement
```

or

```
SBR name
    BEG
    statement
        •
        •
    statement
    END
```

The sub-routine is defined by "SBR name" and the statement or instruction block that follows.

"name"	may comprise a maximum of 8 letters or figures.

To run the sub-routine "name", the main program must contain:

```
CAL name        "call" "name"
```

The sub-routine is then run before continuing at the next line.

To save computing time it is advisable to place sub-routines at the beginning of the program. Should a sub-routine occur later in the program, a GOTO instruction must be used to jump to it.

```
    •
    •
GOTO Lx
SBR 1
    BEG
    •
    •
    END
```

```
SBR 2
    BEG
    •
    •
    END
Lx
    •
    •
CAL 1
    •
    •
```

6.1.8. Instructions for controlling injection

As was the case for manually controlled testing, injection is determined by the settings of control parameters from list FF. The following statements replace the keys 'START', '+', '-' (search direction) and '0' (printing) used in manual operation.

Starting injection:

F	"fault"	injection commences

Following this statement, the test set either generates the injection quantities or, if A ≠ 0 was set, starts the search sequence. The next line of the program is then processed.

Direction of the automatic search:

In both program and manual operation, the automatic search sequence is defined by the 'A' parameter setting: A = 0, 1 or 2. Parameter 'ST' determines the number of steps and the direction.

ST = +n	"step" = +	"number" (increment)
ST = -n	"step" = -	"number" (decrement)

A = 0: **n** is the number of steps by which the main parameter is increased or decreased.

A = 1 or 2: The value of **n** is of no consequence providing it is greater than zero. Only the sign determines whether the parameter must be automatically incremented or decremented.
The search sequence ceases when:

- the variable parameter reaches its limit.
- a test condition set for parameter TI is fulfilled.

ST = ... should always immediately precede the line with the "F" instruction, but be at least within the same loop!

Terminating injection

STP "stop"

This statement switches all outputs to zero. It enables injection to be terminated upon reaching a limit or a steady-state injection signal after a program instruction has stopped a search sequence.
"STP" may be at any point in the program. For reasons of safety, however, "STP" should also be the last instruction **before** "EP" (end of program). This ensures that injection is stopped at the end of the program.
If "STP" immediately succeeds the command "F" during steady-state injection and "A" and "ST" are zero, injection is terminated before the time "TF" expires. A $\neq$ 0 and ST = 1 must be set if a single injection lasting "TF" has to be programmed.

6.1.9. Printing instructions

If a printer is connected to the serial interface, i.e. to connector 7 in Fig. 2.5, the following commands can be used to print parameter designators or parameter lists with their current values and short texts or other information.
In order to execute the printing commands directly (i.e. not in the program) change to the enter mode *XS92b ready*. In manual operation, this is done by pressing the keys 'CTRL' and 'C' (^C).

Commands:

TYP This command switches the (handheld) terminal keyboard through to the printer. All characters (or commands) entered via the (hanheld) terminal which must be terminated by 'ENTER' (the 'ENTER' key functions as "carriage return - line feed", CR/LF) are then printed.
The test set itself does not react to the characters entered while the terminal is switched over.
Pressing the keys 'CTRL' and 'D' (^D) terminates the transmission to the printer and *XS92b ready* appears on the display.

PLP PAR "print parameter" " parameter"

The parameter "PAR" and its value are printed.

PAR can be:
- the name of a parameter, e.g. ZS
- the name of a list, e.g. ZZ (In this case, all the parameters in the list and their values are printed.)

PTP PAR "print parameter" " parameter"

A parameter is printed without actuating line feed (LF). This permits several parameters to be printed on a line (enter spaces using PTT pos.).

PLS text "print string" "text"

Any text up to 80 characters per line is printed for "text". The command must be repeated for every additional line.

PTS text "print string" "text"

Prints any text without line feed.

PTT pos "print tabulator" "position"

pos = number of characters from the left-hand edge or from the first character of the last line printed.

The printing head moves to the right to "pos" following 'CR/LF'. This instruction remains in force until it is cancelled or changed (PTT 0).

PTL nn "print line" "number"

The printer moves forward "nn" lines.
"PTL" must always be the **final instruction** after PLS, PTS, PLP and PTP.

Only the following supplementary commands can occur in the program:

TXT= "print text"

This instruction stops the program and permits text to be entered at the terminal. Upon pressing "Enter", his text is transmitted to the printer, e.g. via the printer port ("PCF").

^D The program continues upon entering "CTRL" and "D".

PPX d "print ASCII code decimal"

Control codes can be generated for controlling the printer, e.g. a printer requires the code 'ESC M' in order to select the font. This corresponds to ASCII decimal (27) and (77) or to PPX27 and PPX77 on the test set.

6.1.10. Examples of test programs

Example 1

Name: TEST

The program TEST searches for the pick-up point of the relay under test in the impedance plane using the step parameter (DLZ). The following values are given:

- PZS=90°, PLZ=90°, K0=1

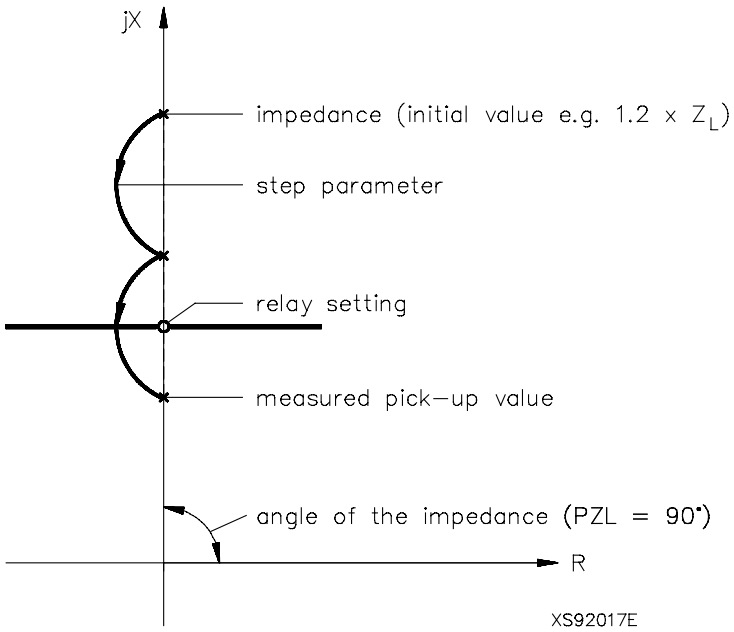

Fig. 6.1 Diagram of Example 1

The values for parameters ZS (source impedance), ZL (fault impedance), DZL (impedance step) and FC (kind of fault) are entered while the program is running.

Program	Comments on the statements
PLS TEST PTL 1	• Program name printed as title
ZS= PZS=90.0 ZL= ZL MUL 1.20 PZL=90.0 DZL= K0=1.0 FC=	• Entry of the parameter designators and those values which are known in any order from the lists ZZ and DZ (input of impedance). • Impedance (initial value e.g. 1.2 x Z_L) • Fault code entry

PLS START: PLP ZZ PLP DZL PLP FC PTL 1	• The selected lists and parameters are printed
SQ=1F1 IN1=1 TF=100 TI=2 MOD=ZZ	• Determination of the selected parameters and values from the FF list.
A=2 ST=-1 F	• Automatic search for a reducing parameter value (decrement) and stop for IN1=1 (interrupt condition).
STP PLS RE- SULT: PLP ZL PLS PTL 4 EP	• The calculated measured values are printed. • End of programme

Program execution

TEST START: ZL=4.00 PZL=90.0 ZS=10.0 PZS=90.0 K0=1.0 PK0=0.0 KS=0.0 PKS=0.0 DZL=0.1 FC=12	• Confirmation of the values entered
RESULT: **ZL=2.00**	• Result for the values entered

(Data entered for this example: ZL = 4.00, ZS = 10.00, DZL = 0.1 and FC = 12 and earth fault L2-N. The protection picked up at ZL = 2.00.)

Example 2

Name: PROGTEST

The program PROGTEST searches for the pick-up values of a protection relay by dynamic injection between 0° and 180° in the complex plane (Fig. 6.2). The initial values of line and source impedances are at the beginning of the program. The search commences with large step changes of impedance which are then reduced twice after detecting a pick-up value. The values are determined for RO, SO, TO, RS, ST and TR. The angular increment is 30°. The values of impedance and phase-angle are printed for the pick-up points determined. A starting contact of the relay under test must be connected to the "TRIP" input of the test set (see Section 1.3.5.).

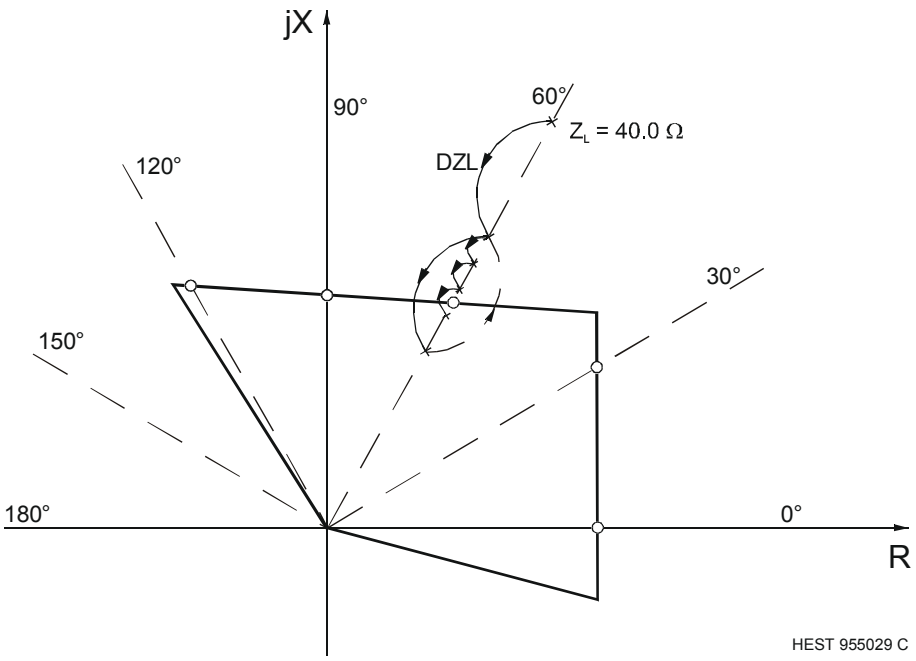

Fig. 6.2 Diagram for the PROGTEST example

Program	**Comments on the statements**
PTS PROGTEST	Print name of program
ZL = 40.0	Assigned test parameter values
PZL = 0	
ZS = 10.0	
PZS = 85	
K0 = 1.0	
FC = 11	
SQ = 1F1	
TF = 100	Test condition: Stop when trip signal received
TI = 2	
OU1 = 1	Sets the binary output as a check.
IN1 = 1	"Trip" input is tested.
TT = 0	Measurement of operating time.
MOD = ZZ	Kind of calculation
PTS DISTANCE	Output of text
RELAY TEST	
PLS PICK-UP VALUES	
PTL 2	
PLP ZZ	Print all impedances and phase-angles
PTL 3	
REP 7	Loop I: Change phase-angle by 30°
BEG	
REP 6	Loop II: Change kind of fault
BEG	
DZL = 5.0	Setting the impedance step
REP 4	Loop III: Search for pick-up value
BEG	
A = 2	Aut. search with response for decrement Test cond.: Stop when trip received
ST = -1	Search dir. towards lower values (decrement)
F	Start injection
IF ZL LT 0.1	If no trip received, check when a value of
ZL=0.1	0.1 Ω is reached
A = 0	Reduce impedance by 1 step and inject.
ST = +1	
F	
DZL DIV 5	Reduce step value by factor 5.
IF DZL LT 0.04	Test the min. value (0.04 Ω)
BRK	
END	End of loop III
STP	Discontinue injection

PTP FC PTL 1 PTP ZL PTT 20 PTP PZL PTL 1 ZL = 40.0	Print the fault code, the value and the angle of the impedance found.
IF FC LT 12 FC = 12 ELS IF FC LT 13 FC = 13 ELS IF FC LT 21 FC = 21 ELS IF FC LT 22 FC = 22 ELS FC = 23	Change fault code for all ground and phase-to-phase faults
END	End of loop II
PZL ADD 30 IF PZL GT 180 BRK	Increase phase-angle by 30° and test max. value.
FC = 11 ZL = 40.0	Reset fault code to initial fault and the value for ZL
END STP	End of loop I
PTS END OF TEST PTL 5	Print text
EP	End of program

When the program is displayed on the screen or printed all lines are justified to the left without indenting.

6.2. Writing a program

6.2.1. EDT mode (editor)

The EDT mode enables a new program to be entered or an existing program to be modified.

A program with the name "name" in the memory unit YS92 is called by entering:

EDT name "edit" "name"

When writing a new program, "name" (max. 8 characters) is then the name of the new program.

It is also called by:

EDT name " edit " "name "

The following is then displayed:

XS92 "name "
XS92 end prog new program

A program already in the memory of the processor unit XC92 (always the case once a program has been edited, run or printed) is recalled without the program name by simply entering:

EDT

The following is then displayed:

```
XS92 "name     "
        statement
                •
                •
        statement
XS92 end prog
```

Only two lines at a time are displayed on the handheld terminal.

The lines 'XS92 "name "' and "XS92 end prog" indicate the start and the finish of a program and the statements are written between them.

Each new line entered is inserted in front of the last one displayed and shifts upwards upon pressing "ENTER", so that the previous last line is displayed once again with the new line above it.

Inserting lines:

If it is desired to insert one of more lines in a program then it is necessary to manoeuvre the line **above** which the new ones are to be inserted such, that it is the last one on the display.

The following commands are required in the EDT mode:

DEL	cancels a character already entered
^L	'CTRL' 'L' cancels the last line on the display and shifts the remainder of the program up one line.
+	shifts the display one line towards the end.
-	shifts the display one line towards the beginning.
n +	shifts the display n lines towards the end (n < 1000).
n -	shifts the display n lines towards the beginning (n < 1000).
ENTER	terminates a statement or a numerical correction.
^R	'CTRL' 'R' initiates a numerical correction. Where only the numerical values on a program line have to be corrected, the corresponding line must be the last one on the display before entering ^R. The new values can then be typed and the operation terminated by pressing 'ENTER' (does not apply to control parameter codes).
^H text	searches for a program line. In the case of command lines, "text" must be the first eight characters of the line, or in the case of actual text the first two words including PTS or PLS. The search commences at the last line displayed and proceeds towards the end of the program.
^C	'CTRL"C' quits the EDT mode.
INC name	Inserting programs • This command copies the program "name" from the memory unit YS92 to the program being processed by EDT. The copied section is inserted **above** the last line on the display. • Press either the "+" or the "-" key to continue editing.

6.2.2. Editing a program using a text editor

Use only a straightforward text editor that does not generate (possibly invisibly) any formatting characters as do word processors like “WORD” or “WRITE”.

The first line must be a program line without any title, name or comment before it.

The other program lines must follow immediately and justified to the left.

The last line must be:

EP terminated by “Enter”.

No spaces or empty lines may follow.

To transfer the program to the XS92b, open the folder “RS-232C/Transfer” when running the XSWIN operator program and then select “Transfer program from PC to test set”.

During the transfer of the program, the XS92b checks the syntax at the end of every line and discontinues the transfer if there is an error.

6.2.3. Printing programs

PRT "print"

The PRT instruction enables programs in the memory to be printed.

It may be entered either with or without a program name:

a) Without name: PRT
If there is a program in the RAM of the processor unit XC92, it will be printed, otherwise the entry of a program name will be requested (error message). The program named will then be transferred from the memory unit YS92 to the RAM before being printed.

b) With name: PRT name
The program named is first copied from the ancillary memory unit YS92 into the RAM of the processor unit and then printed. An error is signalled if a program is already in the RAM.
Printing can be interrupted or the "PRT" mode aborted by entering ^C 'CTRL' 'C'.

6.2.4. Executing programs

RUN "name"

Test programs in the memory can be started by entering RUN which may be entered with or without a program name:

a) Without name: RUN
This procedure is used when a program is already in the RAM of the processor unit XC92, otherwise an error is signalled.

b) With name: RUN name
The program "name" is first copied from the memory unit YS92 to the RAM. An error is signalled if a program is already in the RAM.

A program being executed can be interrupted or the "RUN" mode be aborted by entering ^C 'CTRL' 'C'. Injection which was started using "F" cannot be interrupted by ^C, but only by ^E (resets the test set), to which the test set reacts immediately after the next fault period (MT x TF).

Execution of a program line-by-line

A program can be run line-by-line. Commencing from '*XS92b ready*' (after "^C" or switching on) enter

TSS 4 "test single step"

and then execute the program starting with

RUN (with or without "name")

The program can now be gone through line-by-line using "^D" ('CTRL"D').

At the end of the program the program line pointer (program instruction pointer) is on the last line (normally EP). Because of the instruction TSS 4, RUN does not automatically reset the program line pointer and the program will therefore be executed not from the first line, but from the last position of the cursor (in the EDT or RUN modes).

Before beginning a further line-by-line run at a particular line, the line concerned must be manoeuvred onto the display using the editor. The run can then be started (by "^C", "RUN").

The instruction "TSS 4" is effective until

TSS 0

is used to switch back to normal program operation.

6.3. Program management

Programs are saved according to name in the memory unit YS92.

How many programs can be stored is limited by the capacity of the memory. The available space can be occupied by a few long programs (max. program size 16 kBytes or up to 62 kBytes with new YS92 boards) or many short programs.

6.3.1. Initialising the memory unit YS92

The memory PCB can accommodate blocks of PROM's, EEPROM's (non-volatile read/write memories) and RAM's. Apart from actual programs, non-volatile memories usually contain data relating to their size, programs already stored, test codes (see Section 7.1.7.2.) etc. The latter are read by the micro-processor every time the test set is switched on. All memory blocks having an incorrect test code are considered to be RAM's (if data can be written in them) or not fitted.

Commencing from the *XS92b ready* state, the memory blocks (in the case of EEPROM's) are given a test code by entering

INI "initialisation"

Since this operation overwrites read/write memories, the system requests confirmation that initialisation should be carried out:

Delete all (y/n)?

The memory is only initialised if the answer is "Y".

The "INI function" can be executed at any time, but has to be executed:

- when a new PCB YS92 is being used for the first time
- memory units on the PCB have been replaced or added.

6.3.2. Storing a program

STO "store"

The editor mode only enables a program to be compiled or modified in the RAM. The STO function is needed to save it in the memory unit.

Since only a program actually in the RAM can be saved and the name is therefore known, STO is used on its own without stating a program name.

The "STO" command can also be used to save programs which are being edited.

The amount of memory space is checked before a program is saved. If there is not enough space, the new program must be shortened. Programs in the memory can be deleted after ^C ('CTRL"C') to create free space. Saving a program can take more than a minute depending on its length.

DEL "delete"

DEL used independently of the editor deletes a complete program. It can be entered both with and without a name.

a) Without name: DEL
A program in the processor RAM is deleted.

b) With name: DEL name
The program named is deleted from the memory unit YS92

The "DEL" command initialises all the parameters at their default values again.

6.3.3. Program directory

DRC "directory"

The instruction "DRC" displays the list of programs in the memory with their name and length in Bytes and also the amount of free memory remaining.

Display format:

```
name    nnnn
name    nnnn
• •
• •
name    nnnn

free    nnnn
```

where:
name = program name
nnnn = length of program in Bytes

To enable the lines to be viewed on the handheld terminal, they are shown in pairs upon pressing any key.

If a program name is given on the line of the DRC instruction, i.e.

DRC name

then only this program and its length in Bytes are displayed.

Operator program XSWIN: In the terminal mode, the first program name in the list appears after clicking on the “DIR” button. The other names are called using the downwards arrow key “↓”. After the last program name, the XS92b reports:

free (number of free Bytes)

XS92b ready

6.3.4. General reset

The test set XS92b is completely reset (hardware and software) when either the 'RESET' button located on the side (see 16 in Fig. 2.7) is pressed or the instruction 'CTRL' and 'E' is entered at the terminal. Executing a reset interrupts injection, deletes the RAM in the XC92 and sets the default values.

6.4. Terminal control

6.4.1. Principle

The terminal driver (TDM) has two logical ports, one for the console (handheld or computer terminal) and one for the printer (serial interface). Identical sets I/O functions exist for both ports.

Since the printer port, like the console port, can be operated in both directions by the TDM, it can also be used for communicating with a computer system.

TDM operates in both transfer directions using the XON/XOFF protocol. Thus the output of data can be interrupted with XOFF, which permits a high Baud rate even on slow printers or data receiving equipment.

The input of characters is controlled by interrupts and takes place cyclically via a ring buffer. TDM emits an XOFF signal when data arrives quicker than it can be processed to prevent the ring buffer from becoming full. As soon as the ring buffer has been emptied sufficiently, TDM emits an XON signal.

6.4.2. TDM functions

TDM provides a series of I/O functions in the form of Pascal procedures and functions which are independent of hardware and used by the program.

TDM contains everything needed to execute the functions on the hardware available. TDM also takes care of the usual basic functions for communicating with the terminal so the user does not have to include them in his program.

6.4.3. Line input

Line input is a TDM function which transfers a correct line to the program.

Corrections:	While writing a line, mistakes can be corrected using the 'DEL' key. Each time it is operated, the last character on the display is deleted.
Termination:	The input of a line is usually terminated by pressing 'ENTER' (= CR) or LF.

Horizontal scroll: The handheld terminal can display 16 characters. Entering further characters causes the display to shift to the left, so that the last 16 characters are always visible. Upon deleting a character the display shifts to the right.

6.4.4. Switching between terminals

Switching between devices, e.g. console and printer, on the same port is a standard feature. This was included to achieve the maximum possible flexibility.

User instructions:

a) Terminal configuration

TCF p "terminal configuration"

with p = T Operation via the terminal port ("handheld terminal")
with p = P Operation via the printer port ("serial interface")
with p = N All signals to the terminal are suppressed. The control function of the terminal remains effective.

b) Printer configuration

PCF p "printer configuration"

with p = P Printer output at the printer port ("serial interface")
with p = T Printer output at the terminal port ("hand terminal")
with p = N Printer output switched off.

The standard configuration is set when XS92b is switched on:

- Operation via terminal port ("hand terminal")
- Printer output at printer port ("serial interface").

The instructions to switch ports may also be used in programs.

6.4.5. Kind of terminal

Provision is made for specifying the number of lines and characters per line and therefore the operating firmware can be matched to the terminal in use. Whilst all operations can be performed via the handheld terminal, the two-line display has its limitations particularly when editing programs. To take full advantage of a VDU terminal (PC) in relation to the handheld terminal, the terminal in use may be specified.

TCZ zz "terminal control lines" "number"
zz = number of display lines
TCK kk "terminal control column" "number"
kk = number of characters per line

Upon energising the XS92b, the terminal code for the handheld terminal is initialised and corresponds to the input:

TCZ 2
TCK 16

6.4.6. Line termination

Some of the lines from the terminals are not terminated by "LF" or "CR", because an input is expected. The cursor stays at a position after the last character on the right.

This can cause problems when data is being exchanged between the test set XS92b and an external computer via a serial interface. Provision has thus been made for terminating these lines with a "prompt" character:

PMT x "prompt" "code"

x may be any ASCII sign between decimal 43 and 122. The meaning of some ASCII signs is defined as follows:

Code	**Line terminated by (ASCII (decimal))**
x = 0 x = 1 x = 3	CR (carriage return, ASCII (13))
x = 2	LF (line feed, ASCII (10))
x = :	":" (ASCII (58))
x = N	no prompt character

Entry is made after "*XS92b ready*".

PMT can also be set and deleted in the program.

PMT = N should be set when entering lines using the editor!

6.4.7. Alphabetic list of control codes and commands

	Data input for manual operation	(4.5.)
^B	Baud rate: handheld terminal	(2.2.6. / 7.1.6.)
^C	Cancel: man. op., RUN, EDT and PRT	(2.2.6.)
^D	a) Termination of 'TYP' or 'TXT' b) Step instruction for 'TSS'	(2.2.6. / 6.1.9.) (6.2.4.)
^E	General reset	(6.3.4. / 2.2.6.)
^H...text	Search command in 'EDT'	(6.2.1.)
^L	Delete line in 'EDT'	(6.2.1.)
^R	Change value in 'EDT'	(6.2.1.)
ADD	Add parameters	(6.1.5.)
BEG	Begin of an instruction block	(6.1.3.)
BRK	Cancel loop execution	(6.1.6.)
CAL name	Call sub-routine	(6.1.7.)
DEL name	Delete program	(6.3.2.)
DIV	Divide parameters	(6.1.5.)
DRC name	List programs	(6.3.3.)
DUM	Value transfer variable	(6.1.4.)
EDT name	Call editor	(6.2.1.)
ELS	Alternative condition after 'IF'	(6.1.6.)
END	End of instruction block	(6.1.3.)
EP	End of program	(6.1.3.)
F	Start injection	(6.1.8.)
GOTO L...	Unconditional jump	(6.1.6.)
IF	Test for a condition	(6.1.6.)
INC name	Insert program in 'EDT'	(6.2.)
INI	Initialise memory board YS92	(6.3.1.)

^Control codes

Ln	Label No. (GOTO address)	(6.1.6.)
LM	Check limit using 'IF'	(6.1.6.)
MUL	Multiply parameters	(6.1.5.)
PCF	Change printer port	(6.4.4.)
PLP	Print parameters using 'LF'	(6.1.9.)
PLS	Print text using 'LF'	(6.1.9.)
PMT	Set prompt character	(6.4.6.)
PPX	Control character output	(6.1.9.)
PRT name	Print program	(6.2.3.)
PTL	New line ('LF')	(6.1.9.)
PTP	Print parameters without 'LF'	(6.1.9.)
PTS	Print text without 'LF'	(6.1.9.)
PTT	Tabulator	(6.1.9.)
REP	Program loop	(6.1.6.)
RUN name	Call program	(6.2.4.)
SBR name	Sub-routine	(6.1.7.)
ST +/- n	Number of steps and direction for para. A	(6.1.8.)
STO	Save program	(6.3.2.)
STP	Stop injection	(6.1.8.)
SUB	Subtract parameters	(6.1.5.)
TCF	Change terminal port	(6.4.4.)
TCK	Display number of characters	(6.4.5.)
TCZ	Display number of lines	(6.4.5.)
TSS	Test step-by-step	(6.2.4.)
TXT	Print from terminal via program	(6.1.9.)
TYP	Print direct from terminal	(6.1.9.)

The list is divided into the following groups:

a) Baud rate setting on the handheld terminal
b) Immediately after *XS92b ready*
c) Leaving a particular function
d) Effective in programming operation.

a) Control mode which is only used for **setting the baud rate on the handheld terminal**.

^B	Baud rate for handheld terminal	(2.2.6. / 7.1.6.)

b) Control codes and instructions which can be entered **immediately after *XS92b ready***.

	Data input for manual operation	(4.5.)
^C	Cancel: man. op., 'RUN', 'EDT' and 'PRT'	(2.2.6.)
^E	General reset	(6.3.4. / 2.2.6.)
DEL name	Delete program	(6.3.2.)
DRC name	List program directory (XS92)	(6.3.3.)
EDT name	Call editor	(6.2.1.)
INI	Initialise memory board	(6.3.1.)
PCF	Change printer port	(6.4.4.)
PLP	Print parameters using 'LF'	(6.1.9.)
PLS	Print text using 'LF'	(6.1.9.)
PMT	Set prompt character	(6.4.6.)
PPX	Display control codes	(6.1.9.)
PRT name	Print program	(6.2.3.)
PTL	Line feed ('LF')	(6.1.9.)
PTP	Print parameters without 'LF'	(6.1.9.)
PTS	Print text without 'LF'	(6.1.9.)
PTT	Tabulator	(6.1.9.)
RUN name	Start program	(6.2.4.)
STO	Save program	(6.3.2.)
TCF	Change terminal port	(6.4.4.)
TCK	Display number of characters	(6.4.5.)
TCZ	Display No. of lines	(6.4.5.)
TSS	Test step-by-step	(6.2.4.)
TYP	Print direct from terminal	(6.1.9.)

c) Control codes and instructions which permit **a particular function to be aborted** (e.g. manual operation, program control, special functions).

^C	Cancel: man. op, 'RUN', 'EDT' and 'PRT'	(2.2.6.)
^D	a) Termination of 'TYP' or 'TXT' b) Step instruction for 'TSS'	(2.2.6. / 6.1.9.) (6.2.4.)
^E	General reset	(6.3.4. / 2.2.6.)
STO	Save program, close 'EDT'	

d) Control codes and instructions used (only in the editor) under program control.

^C	Close 'EDT'	(2.2.6.)
^E	General reset	(6.3.4. / 2.2.6.)
^L	Delete line	(6.2.1.)
^R	Change value	(6.2.1.)
ADD	Add parameters	(6.1.5.)
BEG	Start of instruction block	(6.1.3.)
BRK	Cancel loop execution	(6.1.6.)
CAL name	Call sub-routine	(6.1.7.)
DIV	Divide parameter	(6.1.5.)
DUM	Value transfer variable	(6.1.4.)
ELS	Alternative condition after 'IF'	(6.1.6.)
END	End of instruction block	(6.1.3.)
EP	End of program	(6.1.3.)
F	Start of injection	(6.1.8.)
GOTO L...	Compulsory jump	(6.1.6.)
IF	Check for condition	(6.1.6.)
INC name	Program insertion	(6.2.)
Ln	Label number (jump address)	(6.1.6.)
LM	Check limit using 'IF'	(6.1.6.)
MUL	Multiply parameters	(6.1.5.)
PCF	Change printer port	(6.4.4.)
PLS	Print text using 'LF'	(6.1.9.)
PMT	Set prompt character	(6.4.6.)
PPX	Display control codes	(6.1.9.)
PTL	Line feed ('LF')	(6.1.9.)
PTP	Print parameters without 'LF'	(6.1.9.)
PTS	Print text without 'LF'	(6.1.9.)
PTT	Tabulator	(6.1.9.)
REP	Program loop	(6.1.6.)
RUN	Call program (and close 'EDT')	(6.2.4.)
SBR name	Sub-routine	(6.1.7.)
ST +/- n	Step number and direction for parameter 'A'	(6.1.8.)
STO	Save program (and close 'EDT')	(6.3.2.)
STP	Stop injection	(6.1.8.)
SUB	Subtract parameter	(6.1.5.)
TCF	Change terminal port	(6.4.4.)
TCK	Display number of lines	(6.4.5.)
TCZ	Display number of characters	(6.4.5.)
TSS	Test step-by-step	(6.2.4.)
TXT	Print from terminal via program	(6.1.9.)
TYP	Print direct from terminal	(6.1.9.)