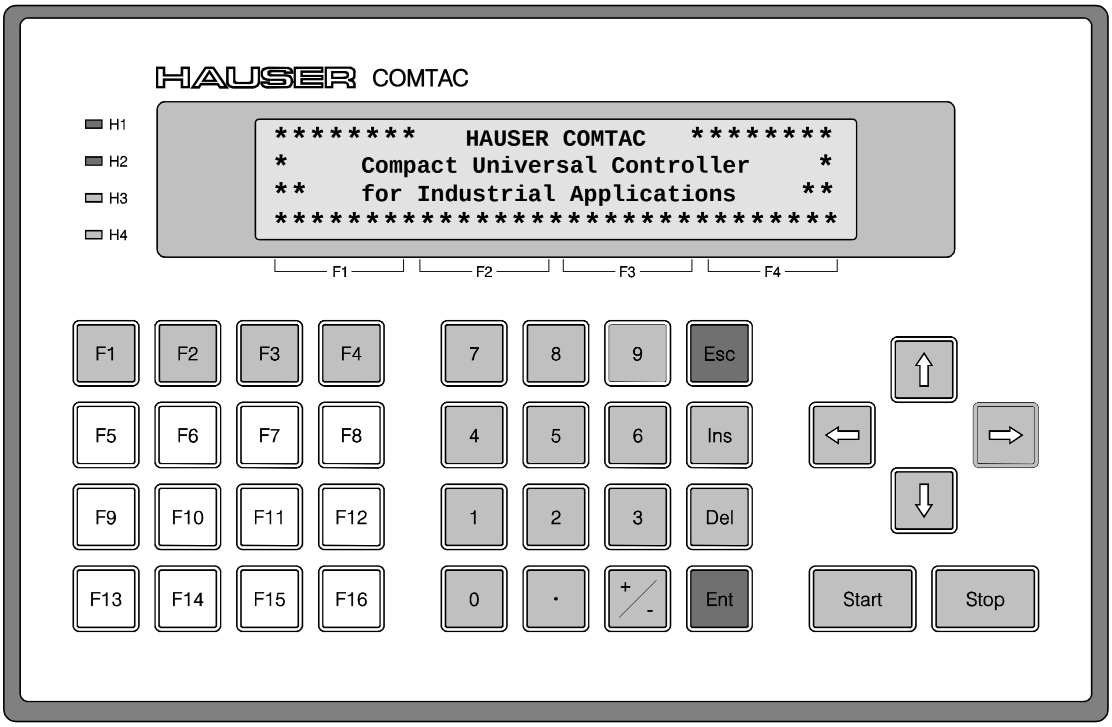
Command Description COMTAC

Compact Industrial Computer



From Software version 2.60

March 2001

HAUSER
We automate motion

CERTIFIED
DIN EN ISO 9001
DQS
QUALITY SYSTEM
Reg. Nr. 36 38

Parker Hannifin GmbH
EMD HAUSER
P.O. Box: 77607-1720
Robert-Bosch-Str. 22
D-77656 Offenburg, Germany
Phone: +49 (0)781 509-0
Fax: +49 (0)781 509-176
http://www.parker-emd.com

Parker Hannifin plc
EMD Digiplan
21 Balena Close
Poole, Dorset
England. BH17 7DX UK
Phone: +44 (0)1202 69 9000
Fax: +44 (0)1202 69 5750
http://www.parker-emd.com

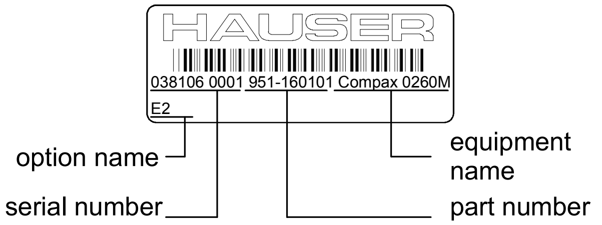
1. Table of contents

2. Device Reference

This documentation applies to the:

COMTAC 2000
COMTAC 3000

HAUSER name plate



Documentation available:

- ◆ Device description.
- ◆ Operating instructions for the Programming Tool.

Notation

Control words: COMTAC parameters (short form: CWs)

The expressions ´Control word´ and ´Parameter´ have the same meaning. The command ´CONTROL´ changes or displays the values of the parameters.

The expression ´Parameter´ is used to be compatible with COMPAX.

A ‘statement’ is a command in a BASIC program line.

Differences from previous Software Versions before V2.01 :

Interrupt processing

- An interrupt subroutine always has to be terminated with the ´RETI´ statement. The ´RETURN´ statement is no longer allowed.
- After power on and program start the interrupt priority level is now equal for all interrupt sources. Thus no interrupt routine can interrupt another. Of course, the priority level of each interrupt source can be adjusted by the user. In previous software versions the priority of the counter, RS232 and the function key interrupt was higher. This fact has to be taken into consideration when running existing programs.

Other Changes:

- The ‘statement’ TIMER has been added. It has the same meaning as the COUNTER statement and both may be used. In a listing the expression COUNTER is changed to TIMER.
- The system clock has been increased from 11,059 MHz to 24,576Mhz, reducing the execution time of each statement.

New software version V2.50

1. Known errors resolved

2. Updates

Program generation

- ◆ During the compiling process, Led H4 flashes.
- ◆ Specific on/off switching of time display (Info-line of terminal).
  - ◆ Ctrl+W activates the time display in the Info-line, while Ctrl+O deactivates it.
  - ◆ The time display is no longer activated automatically by Ctrl+P.

Interfaces

- ◆ OUTPUT x

  The operating system now automatically checks the Bit OUTPUT-Ready using the command OUTPUT before a new text output. As occurs with the ENTER instruction, the command is cancelled after the set timeout period if the OUTPUT-Ready Bit is not set within this time.
  The OUTPUT-Ready Bit does not have to be interrogated any more before each OUTPUT instruction in STSCTR#x.

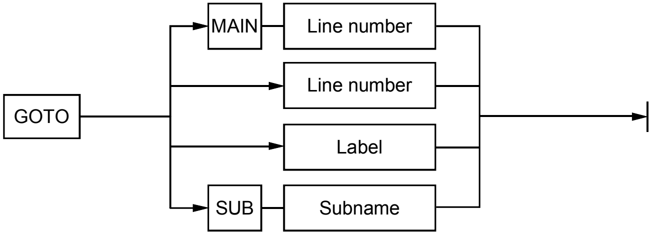
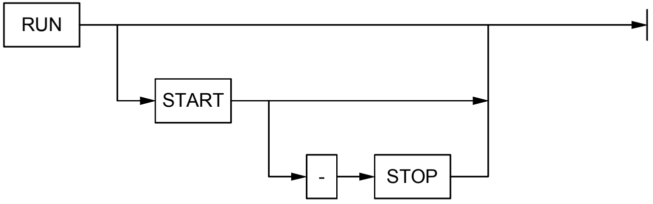
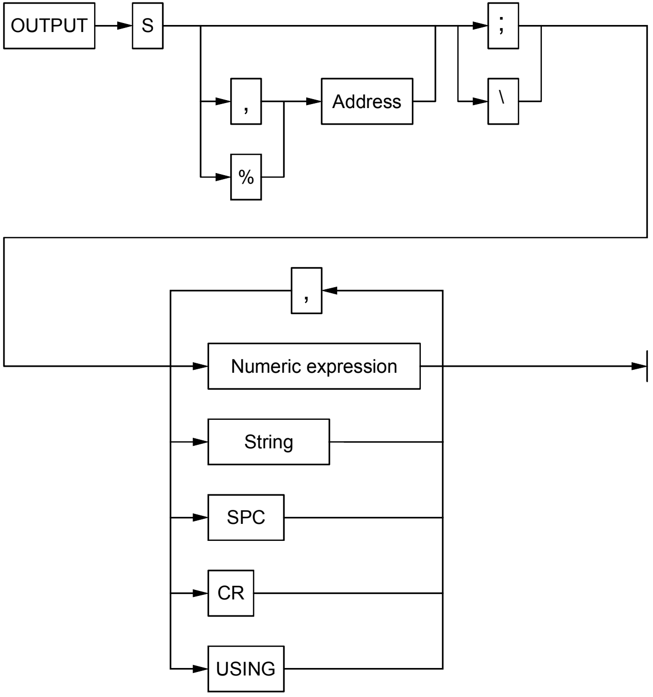
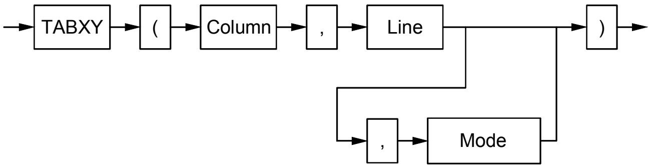
Terminal/membrane keyboard input

- ◆ INPUT, INPUT TABXY, INPKBD, INPKBD TABXY

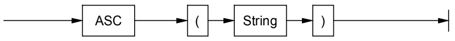
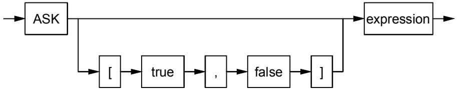
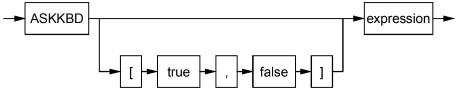
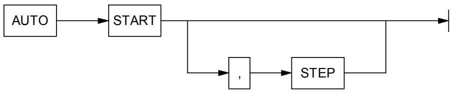
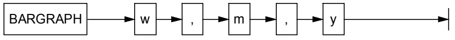
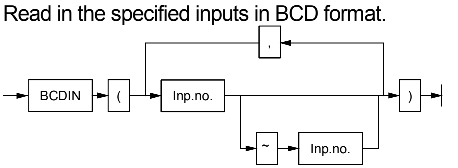
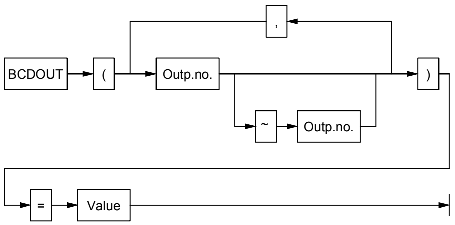
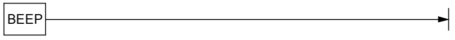
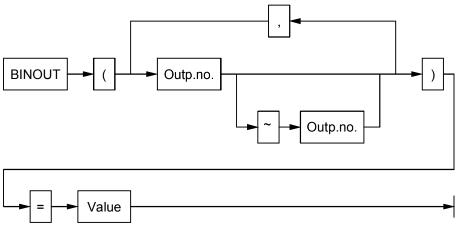
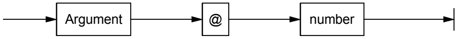
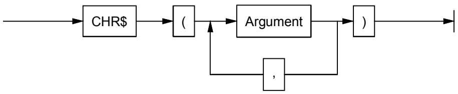
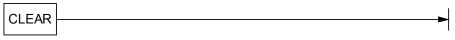
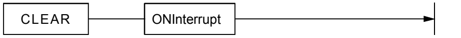
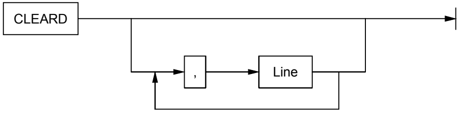
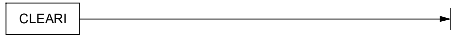
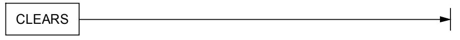
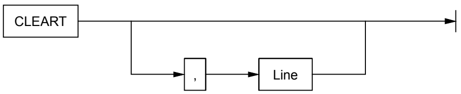
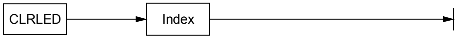
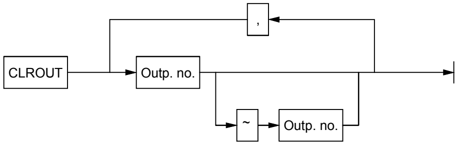
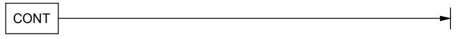
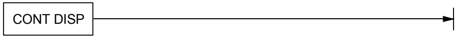
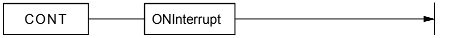
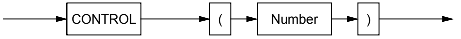
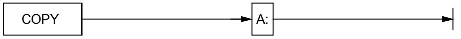
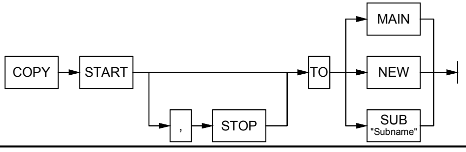
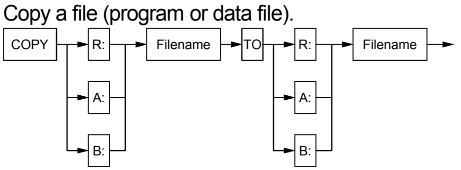
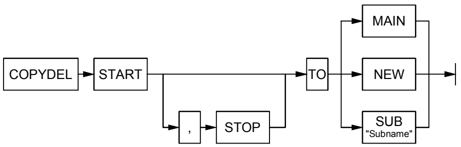
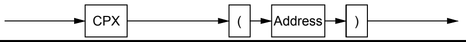
  Using P13 Bit 0 = 1, the parallel Terminal/membrane keyboard input can be activated; i.e. the input can be via the terminal keyboard or the membrane keyboard. The display is in accordance with the programmed Basic instruction; either on the terminal (INPUT, INPUT TABXY) or on the LCD (INPKBD, INPKBD TABXY).

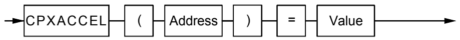
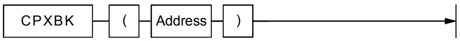
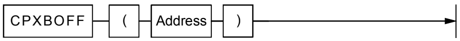
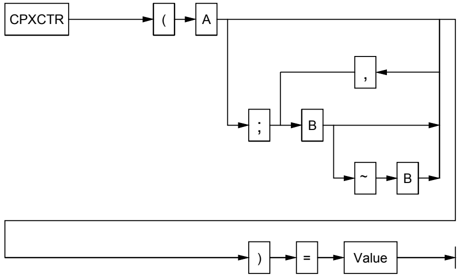
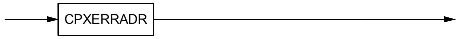
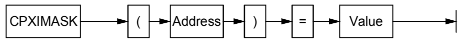
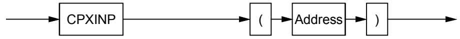
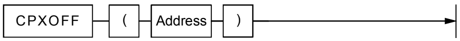
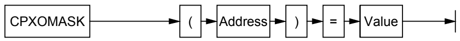
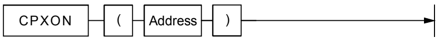
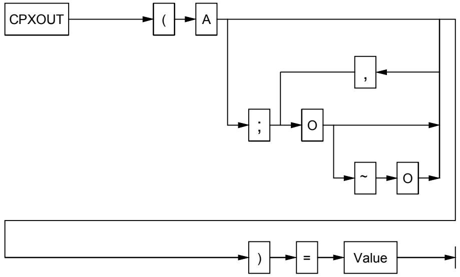
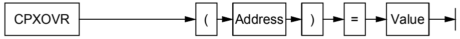
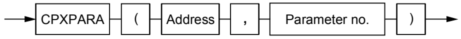
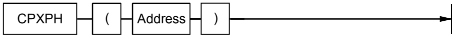
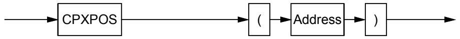
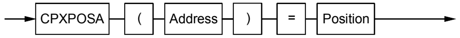
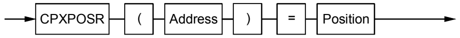
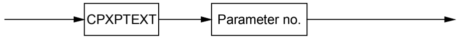
3. Command Syntax

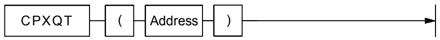
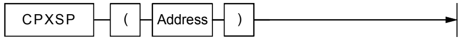
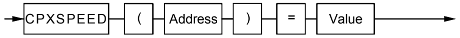
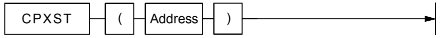
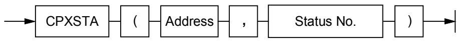
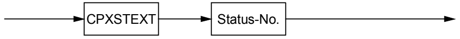
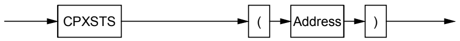
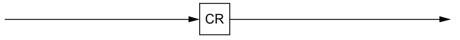
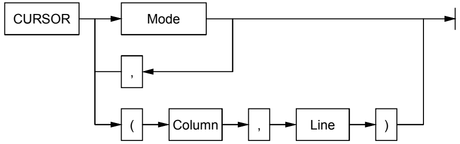
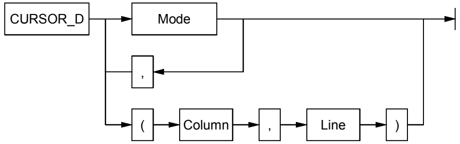
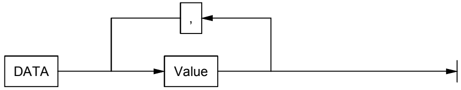
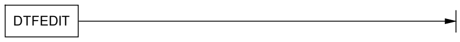
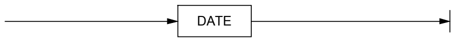
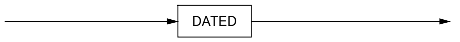
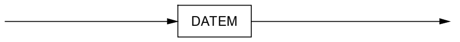
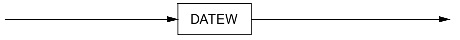
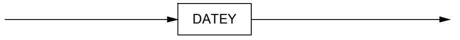
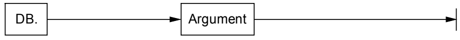
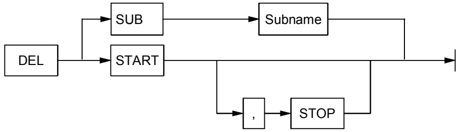
COMTAC uses the line orientated language BASIC, the syntax of which, is described with the help of special structure diagrams and examples.

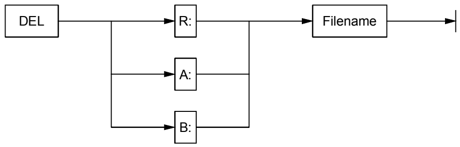
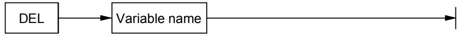
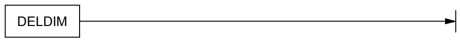
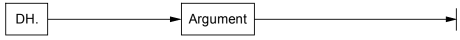
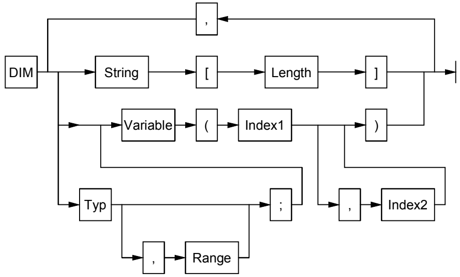
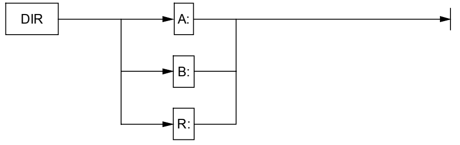
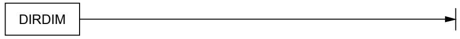
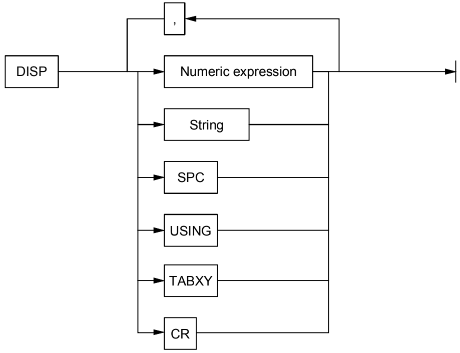
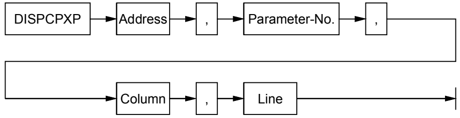
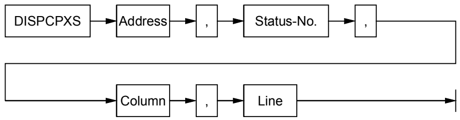
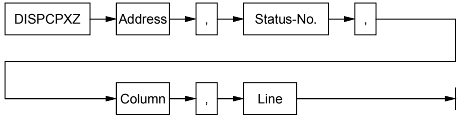
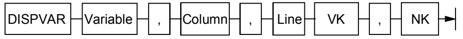
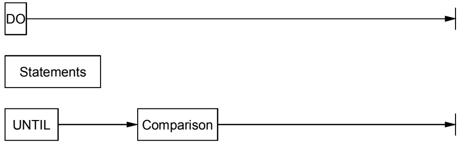
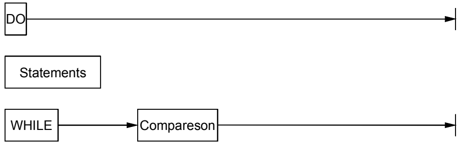
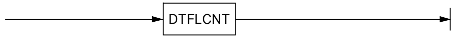
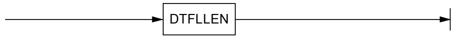
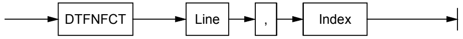
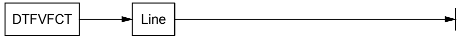
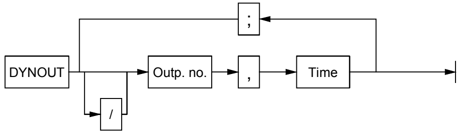
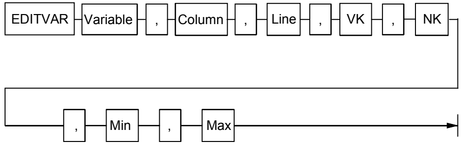
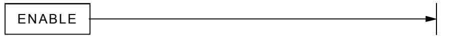
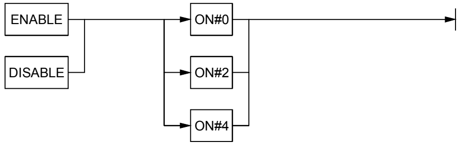
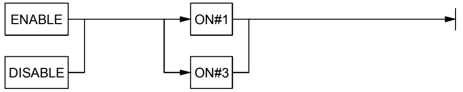
Com- mand	Command syntax	Description
GOTO Line number	Branch to the specified line number.  GOTO MAIN Line number Line number Label SUB Subname	After the GOTO statement the line number must follow. The statement is terminated after the line number (indicated with the bar at the end). **Examples:** GOTO 30, GOTO 400
RUN	Clear all variables, resets all interrupts and starts the user program. 	The command syntax may look like the following: ◆ RUN ◆ RUN 10 ◆ RUN 10-100 These three options are all possible.
OUTPUT (RS232)	Output of strings / values via the RS232 interface  OUTPUT S , Address % ; \\ , Numeric expression String SPC CR USING	After OUTPUT the interface number "S" has to be entered. After this an optional device address with a preceding comma or percent sign may follow. Then a ";" or a "\\" has to follow (different functions). After this some alternative functions, separated by "," are possible: ◆ numerical expression ◆ string ◆ 3 functions (written in capital letters) **Example:** OUTPUT 0; "ANGEL=",A OUTPUT 2,5; "SPA",XPOS,CHR$(10)
TABXY	This statement is used in combination with a PRINT or DISP statement. 	This statement is part of another statement. There is no bar at the end of the arrow. This means that another syntax follows. "Column" and "line" have to be entered, "mode" is optional **Example:** PRINT TABXY(10,10,4),A PRINT TABXY(10,15),B

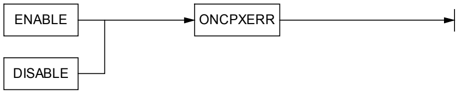
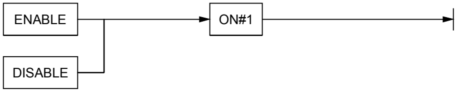
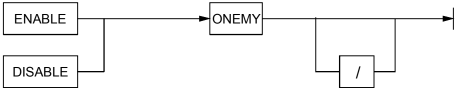
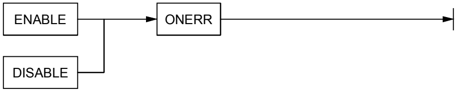
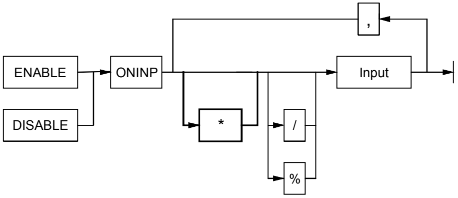
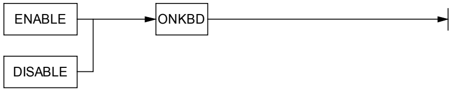
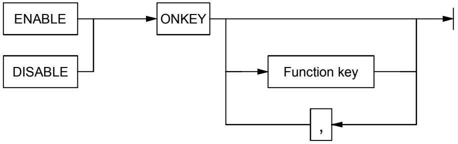
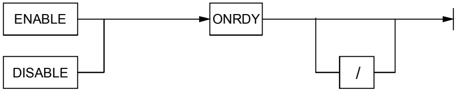
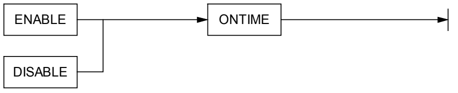
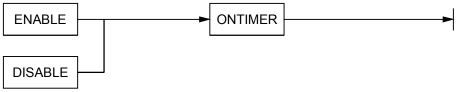
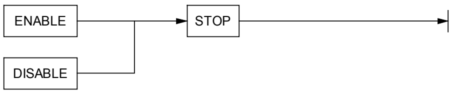
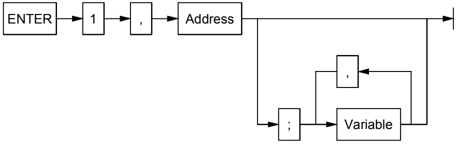
4. Alphabetical Command Overview

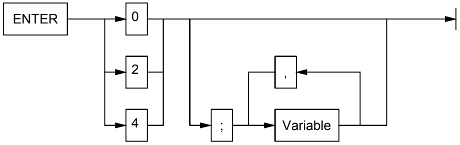
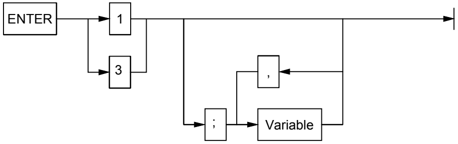
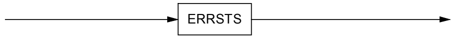
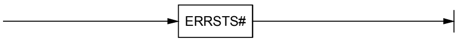
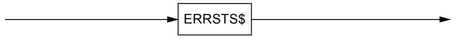
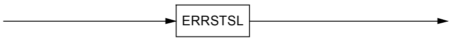
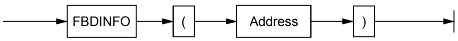
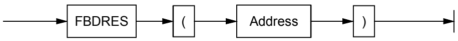
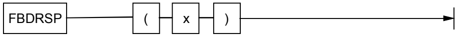
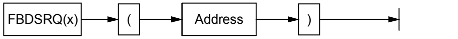
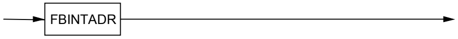
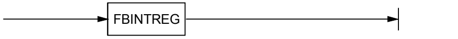
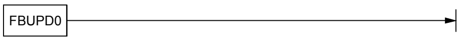
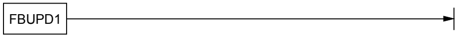
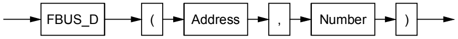
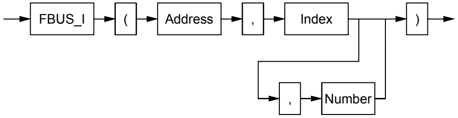
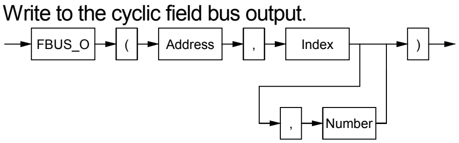
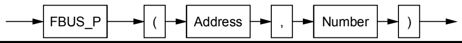
Command	Function. Syntax	Page
ASC	Transforms the first character of a string to a decimal value. 	63
ASK	Keyboard query, true when a key is pressed. 	54
ASKKBD	Keyboard query (COMTAC keyboard) TRUE when a key is pressed. 	57
AUTO	Switch on the auto line function. 	29
BARGRAPH	Generate a bargraph. 	46
BCDIN	Read in the specified inputs in BCD format. 	65
BCDOUT	Read out a value to the specified outputs in BCD format.  BCDOUT (Outp.no. , ~ Outp.no.) = Value	67
BEEP	Sound signal tone of the terminal. 	46
BINOUT	Output a value to the specified outputs in binary format.  BINOUT (Outp.no. , ~ Outp.no.) = Value	67
Bit query	Detects the logic state of a bit . 	23
CHR$	Convert a numeric expression to an ASCII character. 	63
CLEAR	Clear all variables, strings and pending interrupts. 	38
CLEAR ON *Interrupt*	Clear the specified interrupt 	73
CLEARD	Clear the COMTAC display. 	48
CLEARI	Delete all pending interrupts. 	38
CLEARS	Reset the control and argument stack. 	38
CLEART	Clear the terminal screen. 	41
CLRLED	Switch off the specified LED (H1 - H4). 	49
CLROUT	Switch off the specified output(s). 	67
CONT	Continue a previously stopped program. 	31
CONT DISP	COMTAC display: continue a previous stopped automatic display of numbers. 	56
CONT ON *Interrupt*	Enable the specified interrupt source. 	73
CONTROL	Access to system parameters. 	36
COPY (Disk)	Copy a disk. 	35
COPY Basic-line(s)	Copy BASIC line(s) from source to destination.  MAIN COPY START TO NEW , STOP SUB "Subname"	29
COPY File	Copy a file (program or data file).  COPY R: Filename TO R: Filename A: A: B: B:	35
COPYDEL	Copy BASIC line(s) from source to destination. Delete the source.  MAIN COPYDEL START TO NEW , STOP SUB "Subname"	30
CPX	Check for a COMPAX connected to the field bus. 	105

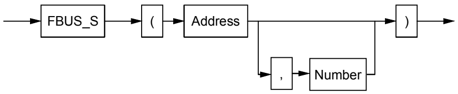
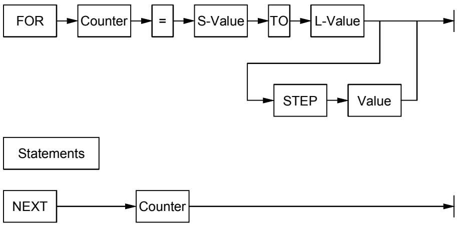
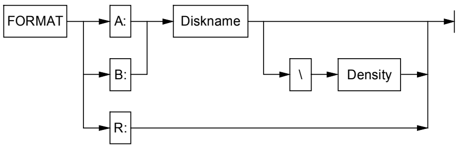
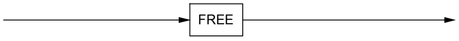
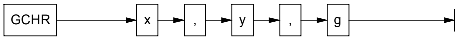
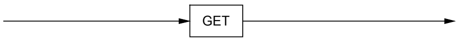
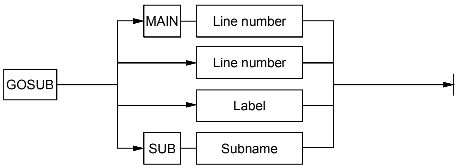
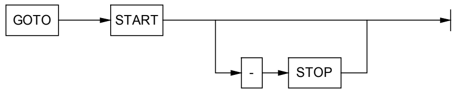
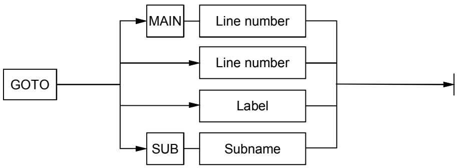
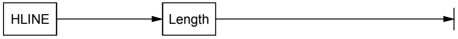
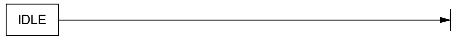
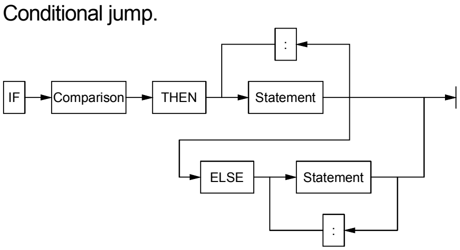
CPXACCEL	ACCEL command to COMPAX. 	100
CPXBK	BREAK command to COMPAX. 	97
CPXBOFF	Switch off COMAPX, motor brake active. 	98
CPXCTR	Activate a COMPAX input function via fieldbus  CPXCTR (A , ; B ~ B) = Value	102
CPXERRADR	Device address of the COMPAX which produced an error message. 	106
CPXIMASK	Enable COMPAX inputs for free access via field bus. 	102
CPXINP	Read the COMPAX inputs I16 to I1. 	104
CPXOFF	Switch off COMPAX, motor brake not active. 	102
CPXOMASK	Enable COMPAX outputs for free access via field bus. 	103
CPXON	Switch COMPAX on, motor break not active. 	101
CPXOUT	Read out a value to the COMPAX output(s).  CPXOUT (A , ; O ~ O) = Value	103
CPXOVR	Override the COMPAX velocity value. 	104
CPXPARA	Read or write a COMPAX parameter. 	97
CPXPH	Command the COMPAX to search for the home position. 	100
CPXPOS	Read the actual position from the COMPAX. 	104
CPXPOSA	Absolute position command sent to the COMPAX. 	100
CPXPOSR	Relative position command sent to the COMPAX. 	100
CPXPTEXT	Term of the specified COMPAX parameter. 	100

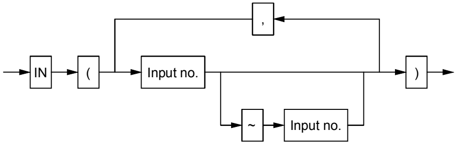
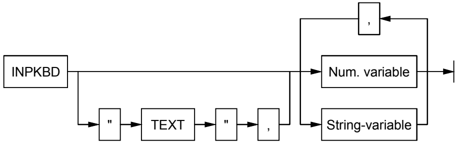
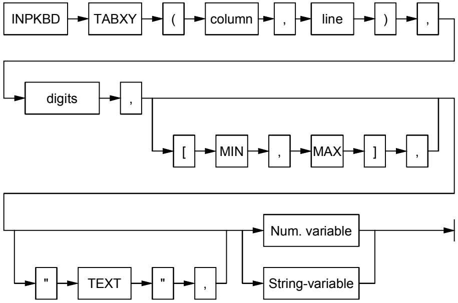
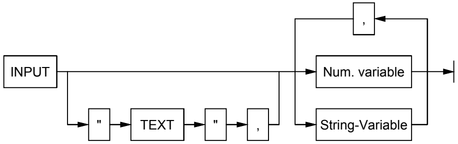
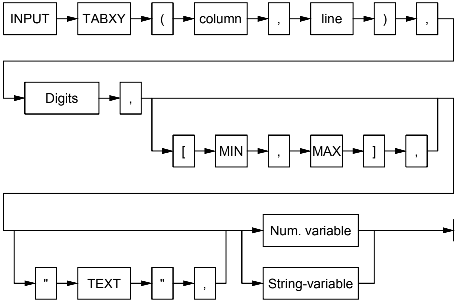
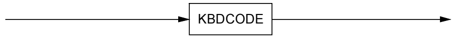
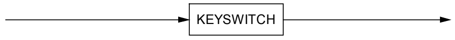
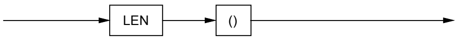
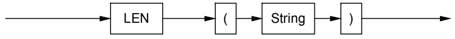
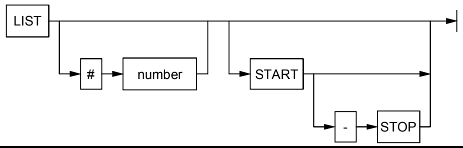
CPXQT	Quit a COMPAX error. 	101
CPXSP	Stop a COMPAX movement or the line interpreter. 	101
CPXSPEED	Speed command to COMPAX. 	100
CPXST	Start a previously stopped COMPAX movement or start the COMAPX line interpreter. 	101
CPXSTA	Read COMPAX status. 	101
CPXSTEXT	Display text of the specified COMPAX status. 	99
CPXSTS	Read the status of the COMPAX outputs O1 to O16. 	103
CR	Carriage Return. 	50
CURSOR	Change cursor mode and position (terminal). 	47
CURSOR_D	Change cursor mode and position of the COMTAC display. 	49
DATA	Make an array of variables. 	36
Data-Text-Editor: DTFEDIT	Start the Data Text Editor. 	87
DATE	Reads out the actual date. 	60
DATED	Read out or set the date. 	60
DATEM	Read out or set the month. 	60
DATEW	Read out or set the weekday. 	60
DATEY	Read out or set the year. 	60
DB.	Display a value in binary format within the COMTAC display. 	44
DEL Basic line(s)	Delete BASIC line(s). 	30

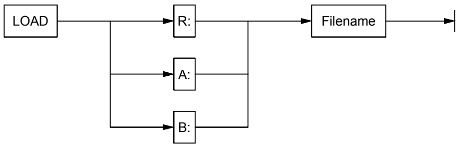
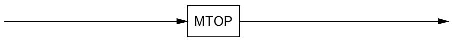
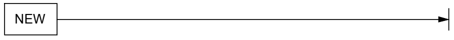
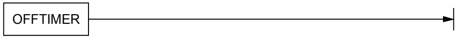
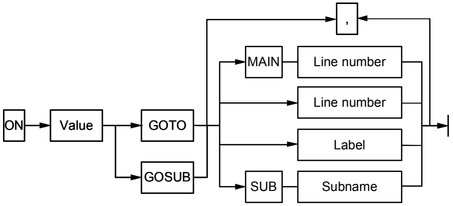
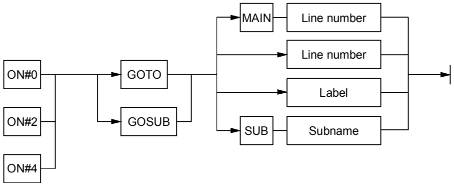
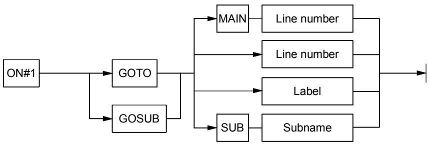
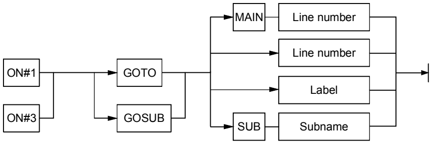
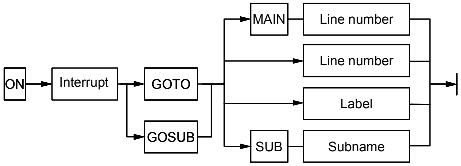
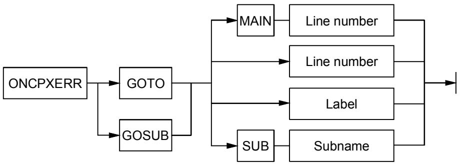
DEL Datei	Deletes a file on the drive R, A or B. 	35
DEL Variable	Delete a variable or an array which is stored in the non-volatile memory (drive R). 	39
DELDIM	Deletes all arrays on drive R which were previously defined. 	39
DH.	Display a value in hexadecimal format within the COMTAC display. 	48
DIM	Reserve memory space for variables, arrays or strings in the specified memory area.  , DIM String [Length] Variable (Index1) Typ ; , Index2 , Range	34
DIR	Display the contents of the specified drive. 	34
DIRDIM	Displays the defined variables, arrays and strings (DIM command). 	39
DISP	Display characters, variables, strings within the COMTAC display.  , DISP Numeric expression String SPC USING TABXY CR	48
DISPCPXP	Display a COMPAX parameter in the COMTAC display once.  DISPCPXP Address , Parameter-No. , Column , Line	99
DISPCPXS	Display a COMPAX status in the COMTAC display once.  DISPCPXS Address , Status-No. , Column , Line	99
DISPCPXZ	Continuous display of COMPAX status in the COMTAC display.  DISPCPXZ Address , Status-No. , Column , Line	99
DISPVAR	Continuous display of a BASIC-variable in the COMTAC-display.  DISPVAR Variable , Column , Line VK , NK	48
DO ... UNTIL	Conditional program loop. 	43
DO ... WHILE	Conditional program loop. 	43
DTFLCNT	System parameter: Number of lines in a Data Text Array. 	87
DTFLLEN	System parameter: Length of a line in the Data Text Array. 	87
DTFNFCT	Read the with number specified Value of a line in a Data Text Array. 	83
DTFVFCT	Assign a value to the specified variable in a Data Text Array 	87
DYNOUT	Set a COMTAC output for a defined time. 	69
EDITVAR	Input a value via the COMTAC keyboard without interrupting the BASIC program.  EDITVAR Variable , Column , Line , VK , NK , Min , Max	55
EMY_STOP	Check the "Emergency Stop" input 	69
Enable	Global enable of interrupts. 	73
ENABLE / DISABLE (RS232)	Enable / Disable the RS232/1, /2 or /3 interrupt. 	85
ENABLE / DISABLE (RS485)	Enable / Disable the RS485/1 or/2 interrupt. 	81

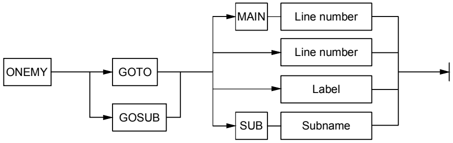
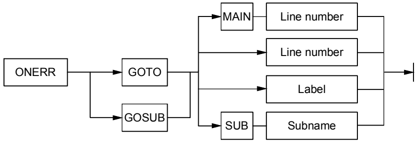
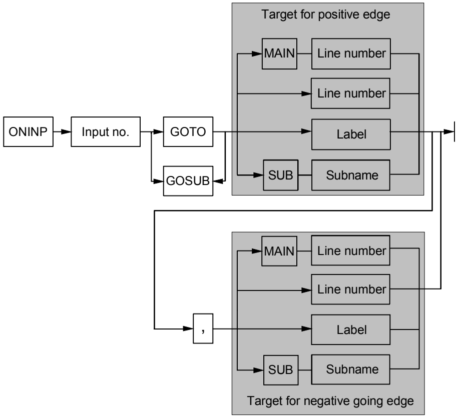
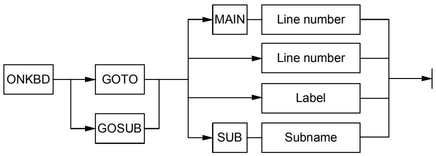
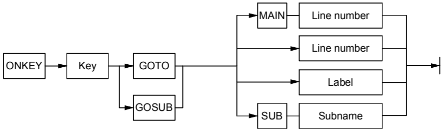
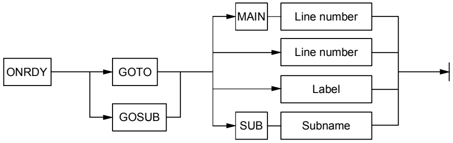
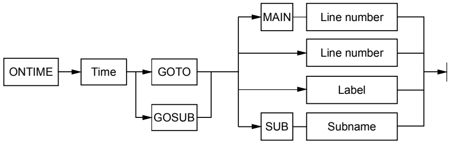
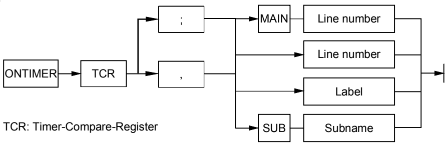
ENABLE / DISABLE ONCPXERR	Enable / Disable the ONCPXERR interrupt.  	105
ENABLE/ DISABLE Feldbus	Enable / Disable the field bus interrupt.  	97
ENABLE/ DISABLE ONEMY	Enable / Disable the Emergency Stop interrupt.  	70
ENABLE/ DISABLE ONERR	Enable / Disable the ONERR interrupt.  	90
ENABLE/ DISABLE ONINP	Enable / Disable the binary input interrupt.  	66
ENABLE/ DISABLE ONKBD	Enable / Disable the keyboard interrupt.  	58
ENABLE/ DISABLE ONKEY	Enable / Disable the onkey interrupt.  	56
ENABLE/ DISABLE ONRDY	Enable / Disable the ready interrupt.  	70
ENABLE/ DISABLE ONTIME	Enable / Disable the time interrupt.  	59
ENABLE/ DISABLE ONTIMER	Enable / Disable the counter interrupt.  	61
ENABLE/ DISABLE STOP	Enable / Disable of Ctrl-C, F8.  	32
ENTER (Field Bus)	Copy a string which was received via the field bus, to the string (#1).  	94

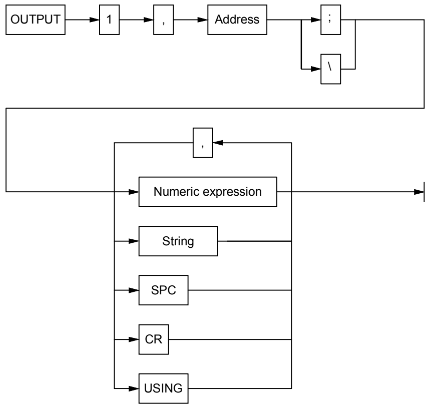
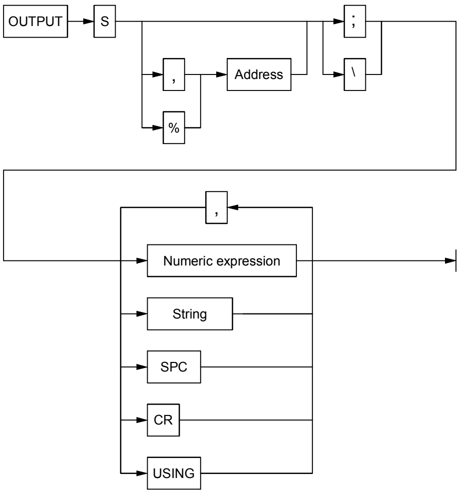
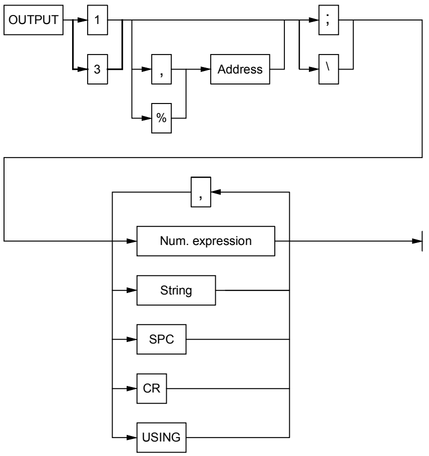
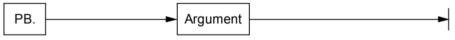
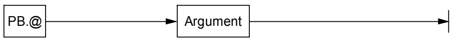
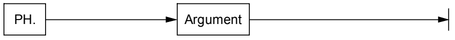
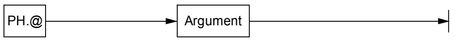
ENTER (RS232)	Copy a string which was received via RS232, to the string $(#0), $(#2) or $(#4).  	85
ENTER (RS485)	Copy a string which was received via RS485, to the string $(#1) or $(#3).  	80
ERRSTS	System variable: Number of the last error.  	90
ERRSTS#	System variable: Name of the subroutine in which the error occurred.  	90
ERRSTS$	System variable: Error message of the last error.  	90
ERRSTSL	System variable: Line number in which the last error occurred..  	90
FBDINFO	Info text about a field bus device.  	97
FBDRES	Reset a field bus device.  	97
FBDRSP	Read field bus device response to $(#1).  	97
FBDSRQ	Read service request data of a field bus device to $(#1).  	97
FBINTADR	Address of the field bus device which generated the ON#1 interrupt.  	97
FBINTREG	Interrupt status register for the ON#1 interrupt.  	97
FBUPD0	Disable the continuous update of the field bus Host.  	93
FBUPD1	Enable the continuous update of the field bus Host.  	98
FBUS_D	Read / Write to parameters of a field bus device  	96
FBUS_I	Read the cyclic field bus input data.  	95
FBUS_O	Write to the cyclic field bus output.  	96
FBUS_P	Read and write to a specified parameter of a field bus device.  	96

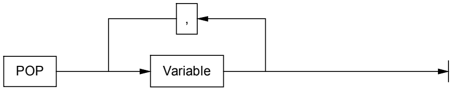
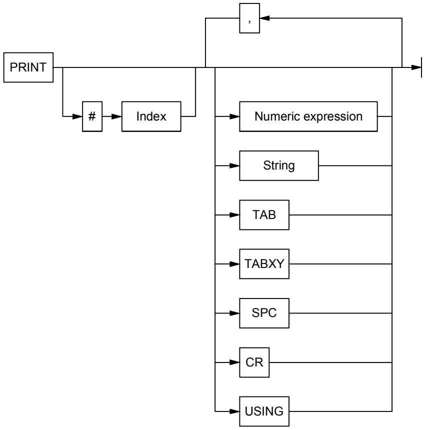
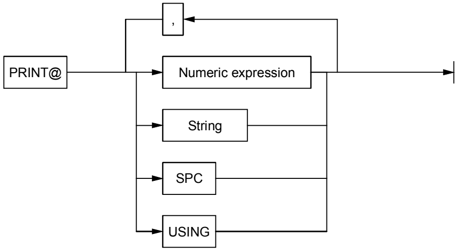
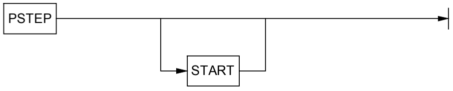
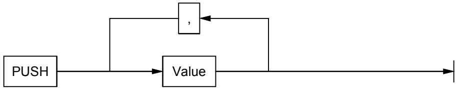
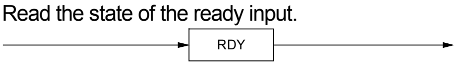
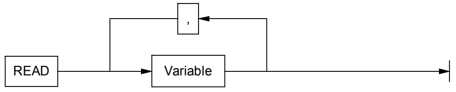
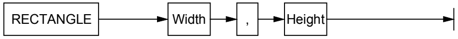
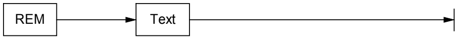
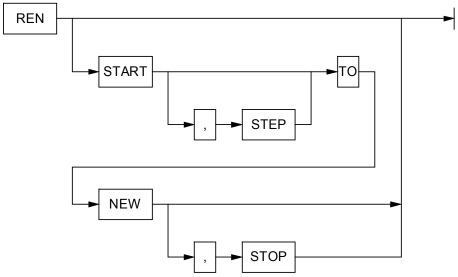
FBUS_S	Read a status value of a field bus device.  	95
FOR ... **NEXT**	Counter loop: to repeat statements a specified number of times.  FOR Counter = S-Value TO L-Value STEP Value Statements NEXT Counter	44
FORMAT	Format the drives R, A or B.  	34
FREE	System variable: Displays the available memory space.  	32
GCHR	Display the character ´g´ at column x and line y of the terminal. ´G´ is a special character.  	47
GET	Copies the code of a pressed key of the keyboard to a variable  	53
GOSUB	Calls a subroutine.  MAIN Line number Line number GOSUB Label SUB Subname	42
GOTO	Start a BASIC program at the specified line number. The variables are not cleared.  	31
GOTO Line number or SUB	Unconditional jump to a BASIC line or a subroutine.  MAIN Line number Line number GOTO Label SUB Subname	42
HLINE	Draws a horizontal line with the specified length.  	47
IDLE	Wait for an interrupt.  	77
IF - THEN - (ELSE)	Conditional jump.  : IF Comparison THEN Statement ELSE Statement :	39

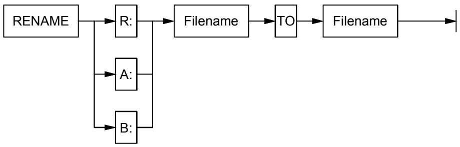
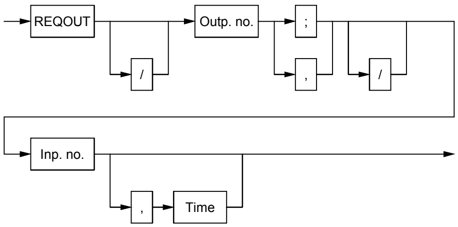
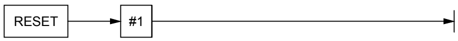
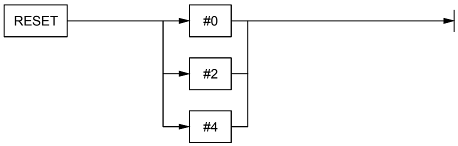
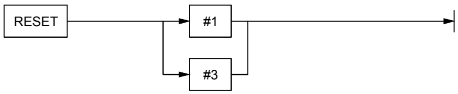
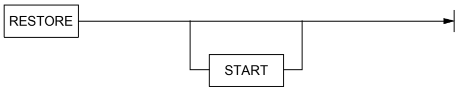
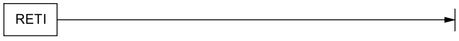
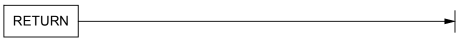
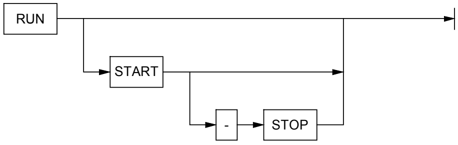
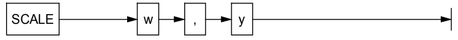
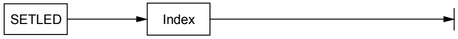
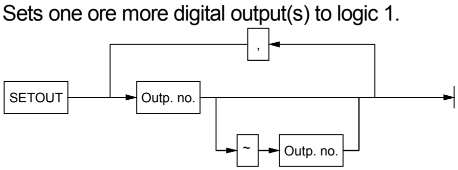
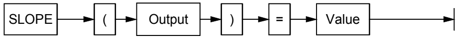
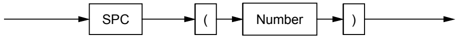
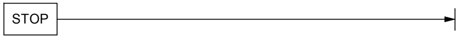
IN	Read one or more digital inputs.  	61
INPKBD	Input of a value via the COMTAC keyboard. The BASIC-program waits till the input is finished.  , INPKBD Num. variable " TEXT " , String-variable	51
INPKBD-TABXY	Position the cursor of the COMTAC display and input a value via the COMTAC keyboard. The BASIC-program waits till the input is finished.  INPKBD TABXY (column , line) , digits , [MIN , MAX] , Num. variable " TEXT " , String-variable	51
INPUT	Input via the keyboard.  , INPUT Num. variable " TEXT " , String-Variable	49
INPUT-TABXY	Input via the keyboard.  INPUT TABXY (column , line) , Digits , [MIN , MAX] , Num. variable " TEXT " , String-variable	49
KBDCODE	Returns the code of the pressed key on the COMTAC keyboard.  	53
KEYSWITCH	Read the state of the key switch (Input of COMTAC).  	65
LEN()	System variable: Length of the BASIC program stored in the RAM.  	28
LEN(String)	Returns the length of a specified string.  	59
LIST	List the BASIC program stored in the RAM from START line to STOP line to the terminal or an interface.  	26

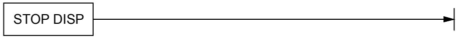
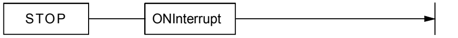
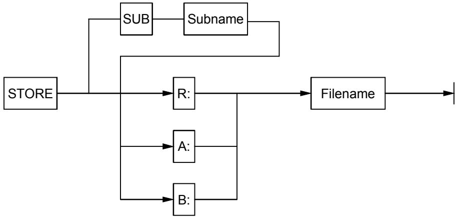
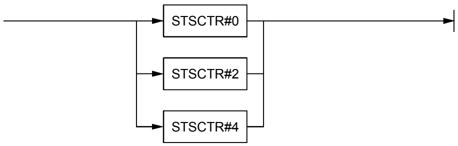
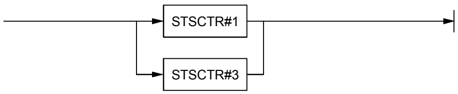
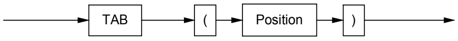
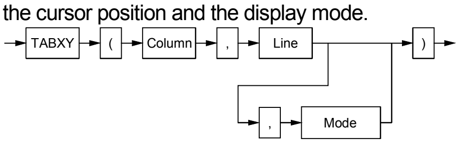
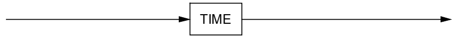
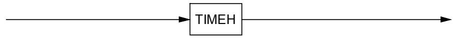
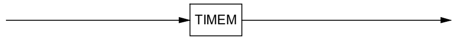
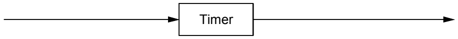
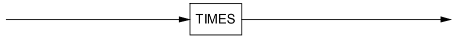
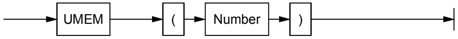
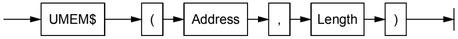
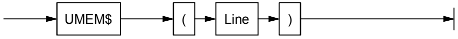
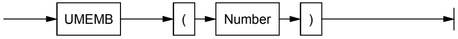
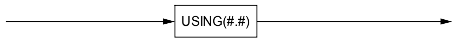
LOAD	Load a program from the specified drive to the RAM.  	31
MTOP	System variable: Returns the highest available address of the RAM.  	28
NEW	Deletes the program in the RAM, resets all interrupts and strings and sets all variables to 0.  	25
OFFTIMER	Cancels the ONTIMER command. The TIMER keeps running.  	57
ON	Multiple conditional jump  , MAIN Line number Line number ON Value GOTO Label GOSUB SUB Subname	38
ON#0 / #2/ #4	Branch, if a valid RS232 interrupt condition occurs.  MAIN Line number Line number ON#0 GOTO Label ON#2 GOSUB SUB Subname ON#4	81
On#1	Branch, if a valid field bus interrupt condition occurs.  MAIN Line number Line number ON#1 GOTO Label GOSUB SUB Subname	92
ON#1/#3	Branch, if a valid RS485 interrupt condition occurs.  MAIN Line number Line number ON#1 GOTO Label ON#3 GOSUB SUB Subname	77
ON-Interrupt	Branch, if any interrupt occurs.  MAIN Line number Line number ON Interrupt GOTO Label GOSUB SUB Subname	38
ONCPXERR	Branch, if an error on a COMPAX device occurs.  MAIN Line number Line number ONCPXERR GOTO Label GOSUB SUB Subname	101

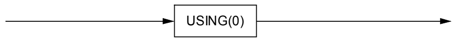
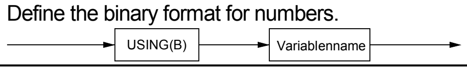
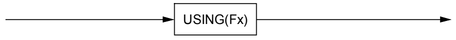
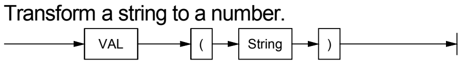
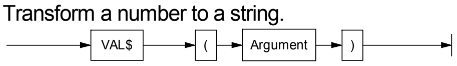
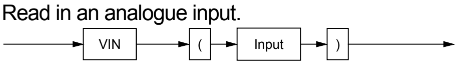
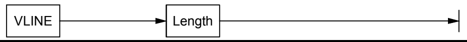
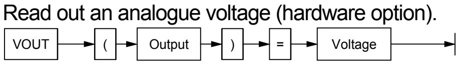
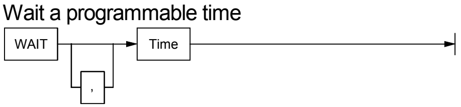
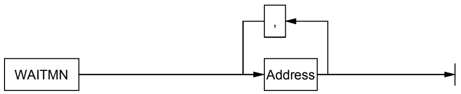
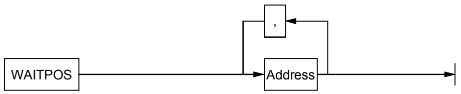
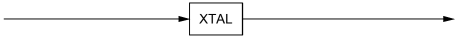
ONEMY	Branch, if the emergency input is activated.  MAIN Line number Line number ONEMY GOTO Label GOSUB SUB Subname	65
ONERR	Branch, if a COMTAC system error occurs.  MAIN Line number Line number ONERR GOTO Label GOSUB SUB Subname	86
ONINP	Branch, if a positive or negative going edge is detected at the selected input.  Target for positive edge MAIN Line number Line number ONINP Input no. GOTO Label GOSUB SUB Subname MAIN Line number Line number , Label SUB Subname Target for negative going edge	62
ONKBD	Branch, if any key on the COMTAC keyboard is pressed.  MAIN Line number Line number ONKBD GOTO Label GOSUB SUB Subname	53
ONKEY 19	Branch, if the specified function key is pressed.  MAIN Line number Line number ONKEY Key GOTO Label GOSUB SUB Subname	52
ONRDY	Branch, if the ready input is activated.  MAIN Line number Line number ONRDY GOTO Label GOSUB SUB Subname	66
ONTIME	Branch, if the actual time of the real time clock equals the preset value.  MAIN Line number Line number ONTIME Time GOTO Label GOSUB SUB Subname	55
ONTIMER	Branch, if the actual value of the timer equals the preset value.  ; MAIN Line number Line number ONTIMER TCR , Label TCR: Timer-Compare-Register SUB Subname	57

OUTPUT (Feldbus)	Read out characters, Strings or numbers via the field bus.  OUTPUT 1 , Address ; \ , Numeric expression String SPC CR USING	90
OUTPUT (RS232)	Read out characters, Strings or numbers via the RS232 interface.  OUTPUT S , Address % ; \ , Numeric expression String SPC CR USING	80
OUTPUT (RS485)	Read out characters, Strings or numbers via the RS485 interface.  OUTPUT 1 3 , Address % ; \ , Num. expression String SPC CR USING	76
PB.	Output a value in binary format. 	42
PB.@	Output a value in binary format at line 22 of the terminal. 	41
PH.	Output a value in hexadecimal format. 	41
PH.@	Output a value in hexadecimal format at line 22 of the terminal. 	42

POP	Assign a value of the argument stack to a variable. 	36
PRINT	Output characters, strings, values to the terminal or an interface (printer).  , PRINT # Index Numeric expression String TAB TABXY SPC CR USING	41
PRINT@	Output a value at line 22 of the terminal. 	41
PSTEP	Single step mode for program test. 	27
PUSH	Store a value temporarily on the stack. 	36
RDY	Read the state of the ready input. 	66
READ	Read a data array. 	36
RECTANGLE	Draw a rectangle at the terminal. 	43
REM	Insert remarks in a BASIC program. 	37
REN	Renumber a BASIC program. 	26

RENAME	Rename a file. 	31
REQOUT	Activate a digital output for a programmable time and poll a digital input during this same period of time.  REQOUT / Outp. no. ; , / Inp. no. , Time	64
RESET Fieldbus	Reset the field bus interface. 	93
RESET RS232	Reset the RS232 1 / 2 or 4 interface. 	82
RESET RS485	Reset the RS485 interface. 	77
RESTORE	Reset the read pointer of a DATA array to the first index. 	36
RETI	Terminate an interrupt subroutine. 	39
RETURN	Terminate a subroutine. 	39
RUN	Clear all variables, reset all interrupts and start a BASIC program. 	27
SCALE	Displays a scale at the terminal. 	42
SETLED	Switch on a COMTAC LED (H1 - H4) 	45
SETOUT	Sets one ore more digital output(s) to logic 1. 	63
SLOPE	Set the ramp time of an analogue output. 	67
SPC	Read out spaces. 	46
STOP	Stop a BASIC program whilst it is running. 	36

STOP DISP	Temporarily store the contents of the COMTAC display. 	52
STOP ON Interrupt	Disables an interrupt source. 	69
STORE	Store programs or data. 	30
STSCTR#0 /#2 /#4	System variable: Status register of the RS232 interfaces.. 	82
STSCTR#1 /#3	System variable: Status register of the RS485 interfaces. 	77
TAB	Used with the PRINT and DISP statement. Sets the cursor to the specified position. 	46
TABXY	Used with the PRINT and DISP statement. Sets the cursor position and the display mode. 	46
TIME	Display the actual time. 	55
TIMEH	Output or set the hours of the real time clock. 	55
TIMEM	Output or set the minutes of the real time clock. 	55
TIMER	Output or set the BASIC timer 	56
TIMES	Output or set the seconds of the real time clock. 	55
UMEM (X)	Read or store a floating point value in the Data Text Array. 	84
UMEM$ (Address, Length)	Read or store a string in the Data Text Array. Index is a byte address. 	85
UMEM$ (Line)	Read or store a string in the Data Text Array. Index is a line of the Data Text Array. 	84
UMEMB (X)	Byte wise access to the Data Text Array. 	84
USING(#.#)	Define a fixed format for numbers. 	47

USING(0)	Define a free format for numbers.  	47
USING(B)	Define the binary format for numbers.  	48
USING(Fx)	Define the exponential format for numbers.  	47
VAL	Transform a string to a number.  	60
VAL$	Transform a number to a string.  	60
VIN	Read in an analogue input.  	67
VLINE	Draw a vertical line at the terminal.  	42
VOUT	Read out an analogue voltage (hardware option).  	67
WAIT	Wait a programmable time  	37
WAITMN	Wait for ´Home Position Found´. COMPAX function.  	101
WAITPOS	Wait for ´Target Position Reached´. COMPAX function  	100
XTAL	Clock frequency of the BASIC system.  	28

5. General

5.1 Operating Modes

Two operating modes are possible with COMTAC: The COMMAND mode and the RUN mode.
In the COMMAND mode a BASIC program can be written or edited and a command can be executed immediately.
In the RUN mode a BASIC program, stored in the RAM, is executed.

5.2 Programming Language

The operating system of COMTAC consists of an interpreter which executes the BASIC program.
Aside from the usual BASIC commands there are versatile commands to operate with the available hardware components of COMTAC, e.g. the digital inputs and outputs and the serial interfaces.
Additionally, there are special commands used to operate with the digital axis controller COMPAX. These commands support the functions of the controller.
A command with a preceding line number is called a statement. This statement is stored in the RAM following an "Enter".
A command is executed immediately if no line number preceds.
Some commands like NEW or LIST can´t be executed in a BASIC program. Thus they can´t be stored in a line number.
A few commands like GOSUB, RETURN can´t be executed immediately.

5.3 Data Format and Accuracy

The operating system of COMTAC can operate with numbers in the range:
± 1E-127 to ±0,999 999 99 E+127
The mantissa of the floating point format has 8 places. All numbers are rounded up or down corresponding to this accuracy. Numbers can be enterd in 5 different formats:

integer	decimal	hexadecimal	binary	exponential
Example:				
129	88.32	0E7B5H	01011B	1.2345E+3

- ◆ The valid value range for the hexadecimal and binary numbers is 0 - 65 535.
- ◆ If a hexadecimal number begins with a letter, a 0 has to precede the number (e.g. 0a245).
- ◆ All numbers are stored in the floating point format. Each number needs 6 bytes of memory. The number π (3.1415926), for example, is stored as follows:

Memory location	Value	Meaning	
X	81H	Exponent	81H=10^1,82H=10^2 80H=10^0,7FH=10^{-1} etc.
X-1	00H	Sign	00H = positive 01H = negative
X-2	26H	7. and 8. position of the mantissa	
X-3	59H	5. and 6. position of the mantissa	
X-4	41H	3. and 4. position of the mantissa	
X-5	31H	1. and 2. position of the mantissa	

5.4 Numeric Variables

5.4.1 Variable Names

A variable name may consist of 1 to 10 letters, numbers or the underscore. The first character has to be a letter, e.g. POSITION1 or POS1.

⚠	The first and the last character and the length of the name define the variable. Uppercase or lowercase letters are equivalent.

This means that the names ABCDE, AFgZE, A_t_E are the same variable.

⚠	It´s important not to use reserved terms (Command syntax) for variables.

5.4.2 Indexed Variables

One- dimensional and two-dimensional variables (arrays) are possible.
The index numbers are put into brackets, two numbers are separated by comma. Examples:
A(7) , B(9,4) , ZX(I) , VPOS(I,7) , WINKEL(2,N)
Arrays without an index number always consist of 121 elements:
Element(0,0), element(0,1),... element(1,0), ...,element(10,10).
Arrays have to be dimensioned before they can be used:
DIM A(30)
◆ defines the array A with 31 elements: A(0) ... A(30).
DIM B(20,15)
◆ defines the array B with 126 elements: B(0,0) ... B(20,5).
The maximum index number is 254.
meaning that a one-dimensional arrays may consist of 255 elements E(0) to E(254).
The index may be a number, variable or a numeric expression.
Optionally arrays can be defined to be stored in the ZPRAM (non volatile memory).

5.5 Strings

A string is any series of characters used to operate with non-numerical information. COMTAC-BASIC allows one dimensional strings. A string is identified with a $ followed by the string index in brackets, e.g. $(1).
This index can be represented by a value, a variable or a numeric expression. The range for this number is 1 to 255.
Examples: $(7) ; $(A) : $ (A-8/4)
The maximum length of a string is 255 characters.
The default length is 16 characters. Within the DIM-statement variable length can be declared. The actual length of a string can be 0 (empty string).
Example: DIM $(1)[10] declares a string with the index 1 and a length of 10 characters.

5.6 COMTAC-BASIC Keywords

⚠ The BASIC comands and the following terms must not be used as variable names!

CBY(
CKCON
COUNT
CWO(
DBY(
DWO(
IEREG
OFF
ONCOUNT
PBY(
PREG

PWO(
RCAP2
T1REG
T2CON
T2REG
TMOD
TOREG
WDPCON
XBY(
XWO(

5.7 Notations

BCD	Binary coded decimal numbers each 4bits of a binary number represent one decimal place. Example:								
Decimal place	10^1					10^0			
Binary digits	2^3	2^2	2^1	2^0		2^3	2^2	2^1	2^0
Number 35, BCD format	0	0	1	1		0	1	0	1

5.8 Command Abbreviations

Statement	Abbreviation
BINOUT	B.
CLEAR	CL.
CONTROL	CT.
COPY	C.
CPXACCEL	AL.
CPXBK	BK.
CPXBOFF	POB.
CPXCTR	CC.
CPXCTR	CC.
CPXIMASK	CIM.
CPXIMASK	CIM.
CPXINP	CI.
CPXINP	CI.
CPXOFF	OFF.
CPXOMASK	COM.
CPXON	ON.
CPXOUT	CO.
CPXOVR	CV.
CPXPARA	PAR.
CPXPH	ZP.
CPXPOS	CP.
CPXPOSA	PA.
CPXPOSR	PR.
CPXQT	QT.
CPXSP	SP.
CPXSPEED	SD:
CPXST	ST.
CPXSTA	STA.
CPXSTS	CS.
CURSOR	CU.
DISABLE	DI.
DISP	D.
ENABLE	EN.
ENTER	E.
ERRSTS	ES.
FBUS_D	FD.
FBUS_I	FI.
FBUS_O	FO.
FBUS_P	FP.
FBUS_S	FS.
GOSUB	GS.
GOTO	GT.
HLINE	H.
INPUT	I.
LOAD	L.
ONERR	OE.
ONINP	OP.
ONKEY	OK.
ONTIMER	OC.
OUTPUT	O.
PRINT	P.
PRINT	?
PSTEP	PS.
RECTANGLE	RECT.
REM	!
RESET	RS.
RETURN	R.
STORE	S.
TABXY	T.
UMEM	¦
USING	U.
VLINE	V.

⇨ These abbreviations also can be found in the detailed command description.

5.9 Arithmetic Operators and Functions

Operator	Description/Function	Example
+	Addition	A = B **+** C
-	Subtraction	X = B **-** 8
*	Multiplication	Z = 6 * 6
/	Division	T = X **/** Y
'DIV'	Integral part of a division	D = X'DIV'Y
'MOD'	Modulus of a division	M = X'MOD'Y
**	Power function (x^y)	A = X ** Y
SQR(*x*)	Square root of *x*	V = SQR(9)
EXP(*x*)	Exponential function (e^x) e = 2.7182818	Q = EXP(T)
LOG(*x*)	Natural logarithm($\log_e x$)	T = LOG(Q)
ABS(*x*)	Modulus of *x*	B = ABS(X/Y-8)
INT(*x*)	Integral part of *x*	Z = INT(W/12)
SGN(*x*)	Sign of *x* SGN(*x*) = 1 when *x* > 0 SGN(*x*) = 0 when *x* = 0 SGN(x) = -1 when x < 0	V = SGN(A-B)
NOT(*x*)	16 Bit one´s complement of *x.* *0<= x <=65535.* NOT(0) = 65535 NOT(1) = 65534	C = NOT(S*4)
RND	Random value in the range of 0 .. 1	Z = RND*100

For these calculations the floating point format is used.

5.10 Trigonometric Functions

COMTAC-Basic calculates with the Arcus (0 - 2 Π). Trigonometrical functions are calculated by Taylor´s series in the range of 0 to Π/2.

The argument of a trigonometrical function has to be as small as possible to get reach a accuracy.

Operator	Description/Function	Example
SIN(*x*)	Sine of *x* *-200000* <= *x* <= +200000	U=SIN(PI/4)
COS(*x*)	Cosine of *x* *-200000* <= *x* <= +200000*x*	C=COS(A*B+4)
TAN(*x*)	Tangent of *x* *-200000* <= *x* <= +200000*x*	N=TAN(D+V)
ATN(*x*)	1/Tangent of *x* range of the result: +/-PI/2	T=ATN(Z/8)
PI	number π (3.1415927)	A=PI/4
RAD	180degrees / PI	W=SIN(30/RAD)

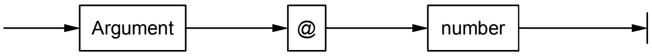
5.11 Compare and Logical Operators

The result of a comparison equals 65535 if it´s TRUE and equals 0 if it´s FALSE.
This result is stored in the argument stack to be displayed immediately or can be assigned to a variable.
Additionally it´s possible to combine the result together with other comparisons (e.g. IF A<B 'OR' A>C THEN ....).

Operator	Description/Function	Example
=	equal to	IF C=4 THEN ...
<>	not equal to	IF X<>Y THEN ...
>	greater than	PRINT X > 4
>=	greater or equal to	T = 5 >= X
<	less than	WHILE I < 10
<=	less or equal to	UNTIL N <= 30
'AND'	logical AND	X = B'AND'C
'OR'	logical OR-	PRINT 4'OR'Z
'NOT'	Bit change	IF NOT (A) ...
'XOR'	logical EXCLUSIVE-OR	DISP Q'XOR'30
@	Bit test	DISP Q@3

The arguments for the functions ('AND','OR','XOR') must be in the range of 0 to 65535.

5.12 Bit-Query



Function:

This function is also called a bit test.
It determines the logical state of a bit.
Index specifies the bit position of an argument.
The result is TRUE if the specified bit is set (log. 1) and FALSE if the bit is not set (log. 0).

Parameter	Input	Range	Description
Argument	num. expr.	0 - 65535	number
Index	num. expr.	0 - 15	Bit number of the argument. (15 = MSB, 0 = LSB)

Example: IF CPXSTS(1)@4 then GOTO 100
WHILE A@1

Note:

Bit 0 is the least significant bit (LSB), bit 15 is the most siginficant bit (MSB).

5.13 Priority of the Operators and Functions

COMTAC uses the following mathematical priorities when calculating a numerical expression:
1) Functions with brackets
2) Bit query (@)
3) Power function (**)
4) Multiplication (*) and division (/)
5) Addition (+) and subtraction (-)
6) Compare functions (=,<>,>,>=,<,<=)
7) logical AND
8) logical OR
9) logical EXCLUSIVE-OR

6. Programm Structure

6.1 Subroutines

A user program may consist of code modules, containing:

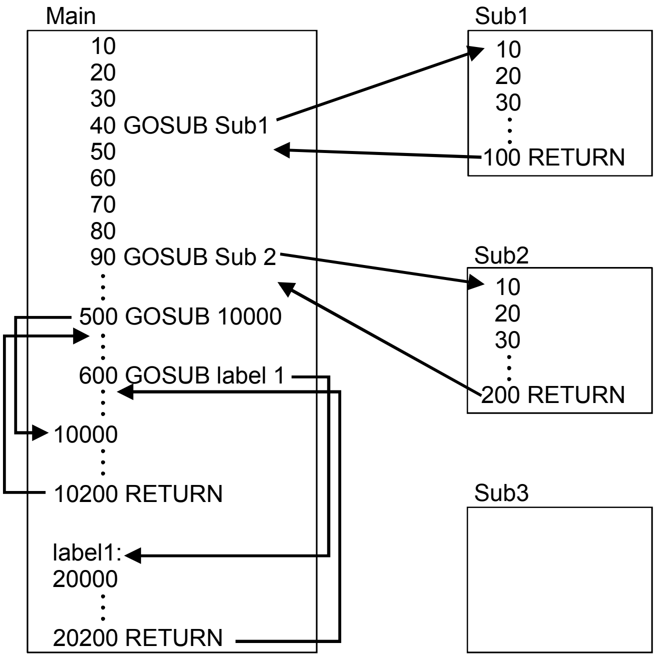
- a MAIN program
- one or more subroutines (SUBs).

All of these modules can be stored in one file.

Each program line must have a line number in the range 0 to 65535.

These line numbers are "local" to each module i.e., each single module can use this range of line numbers.

Example:



A subroutine can call another subroutine. Pay attention to the nesting of subroutines.

Creating a SUB

The instruction **SUB"** subname**"** creates a new subroutine.

- The length of the subname is limited by the maximum length of the line.
- The interpreter distinguishes between small and capital letters (is case sensitive).
- A SUB is created automatically with the COPY statement. In this case a range of existing program lines can be copied to a new subroutine.
  e.g. COPY 1000,1999 TO SUB "StartMe".

Store a SUB

The instruction:
STORE SUB "subname" drivename:"filename.extension"
stores the specified SUB in the file "filename.extension".
The instruction:
STORE drivename: "filename.extention"
stores the MAIN and all SUBs in the file "filename.extension"

Reference to the Programming Tool

- When the DIR-mode is active, the sub which has just been edited is stored with the store function (Function Key).
- When the MAIN program has just been edited, this and all subroutines are stored.
- Before switching to the DIR mode the file(s) to be stored have to be selected.

Load a SUB

- The statement: **LOAD drivename: "filename.extension"** loads an existing SUB.

A program in the compiled format can´t be loaded (refer to page 27 in chapter 7.3.1 RUN).

Call a SUB

GOSUB SUB "subname"
The RETURN statement terminates the SUB.

Jump to a SUB

GOTO SUB "subname".

Jump back to the MAIN module

The statement **GOTO MAIN lineNo.** executes a jump back to the specified line **No**. in MAIN.

Call of a subroutine in the MAIN

The statement **GOSUB MAIN ZNo.** This call is executed in a subroutine SUB 1. It calls a subroutine SUB 2 in the MAIN. This subroutine SUB 2 begins at the specified line number. With the RETURN statement the program run continues in SUB 1.

Edit an existing SUB

The command **SUB"*subroutine*"** selects the *subroutine*.
The program lines can be listed with the LIST command.
If this *subroutine* dosen´t exist, a new one will be created.

Delete an existing SUB

The command **DEL SUB"*subroutine*"** deletes the specified *subroutine*.

Copy program lines to a SUB

The command **COPY/COPYDEL line number,line number TO SUB "*subroutine*"** copies the specified lines of a SUB or MAIN to the *subroutine.*

Copy program lines of a SUB to the MAIN

The command **COPY/COPYDEL line number,line number TO MAIN** copies the specified lines of a SUB to the MAIN.

Edit the MAIN

The command **MAIN** selects and lists the MAIN module.

List SUB Names

The command **LIST SUB** lists all existing SUBs.

Nesting of statements

To control nested statements there exists a memory range of 254 Byte. The programmer has to take care not to exceed this number.
The different statements require the memory space listed below:

GOSUB:	4 Byte
FOR/ NEXT-loop:	18 Byte
DO UNTIL- loop:	4 Byte
DO WHILE- loop:	4 Byte

Only the momentary active GOSUBs and loops have to be considered.

6.2 Labels

Branches executed with GOTO or GOSUB are related to line numbers or labels.
Labels like line numbers are, only valid in the local MAIN or one SUB.

Set Labels

A label is inserted to an existing line after the line number with the command **LABEL"n*ame*":**.
After the colon another BASIC statement must follow.

Example:

Input:
100 label "label 1": Print "Test"
With that the following structure is generated:
label 1:
100 Print "Test"
Label 1 corresponds to line number 100.

Delete Labels

A label is deleted when rewriting or editing a line without the label command.
Example: line 100 is edited. After ENTER, the label is deleted. The contents of line 100 are:

```
100 Print "Test"
```

To retain the label use a remark statement after the label statement. It´s now unnecessary to edit this line again.
Example: 100 label "label 1": ! label 1.

Search for a Label

The statement **LABEL"n*ame*"** searches for the specified label and lists the corresponding line.
If the label doesn´t exist an error is displayed.
Function key F12 lists the program from the label onwards.

Call of a Subroutine with a LABEL (local Subroutine)

The statement **GOSUB "n*ame*"** calls the subroutine with the specified name.

Branch to a LABEL

The statement **GOTO "*Name*"** executes a jump to the specified label.

7. Edit Programs

The operating system of COMTAC includes a terminal program. A simple terminal is sufficient to write programs for the COMTAC, which supports the following terminals or terminal emulations:

- TV905
- VT100

In addition it is possible to use the “COMTAC PROGRAMMING TOOL“ or "WIN TVS05" software that runs under DOS and has additional functions:

- Store and load COMTAC-files on/from the PC (hard disk or diskette).
- A parameter editor to edit COMTAC parameters.
- A macro editor.

⇨ The detailled functions of this software are described in a separate document.

7.1 Terminal Program

A terminal or a terminal emulation always works via a serial interface together with the online operating system. All user inputs are interpreted and executed by COMTAC.

Set Up:

- Baudrate (default): 9600 Bd
  (When the function key F12 of the COMTAC keyboard is pressed during power on, all COMTAC parameters are set to their default values.)
- Interface: RS232/1.
- Terminal type. The default type is TV905 (parameter 2=0).
  Terminal type VT100: parameter 2=3.

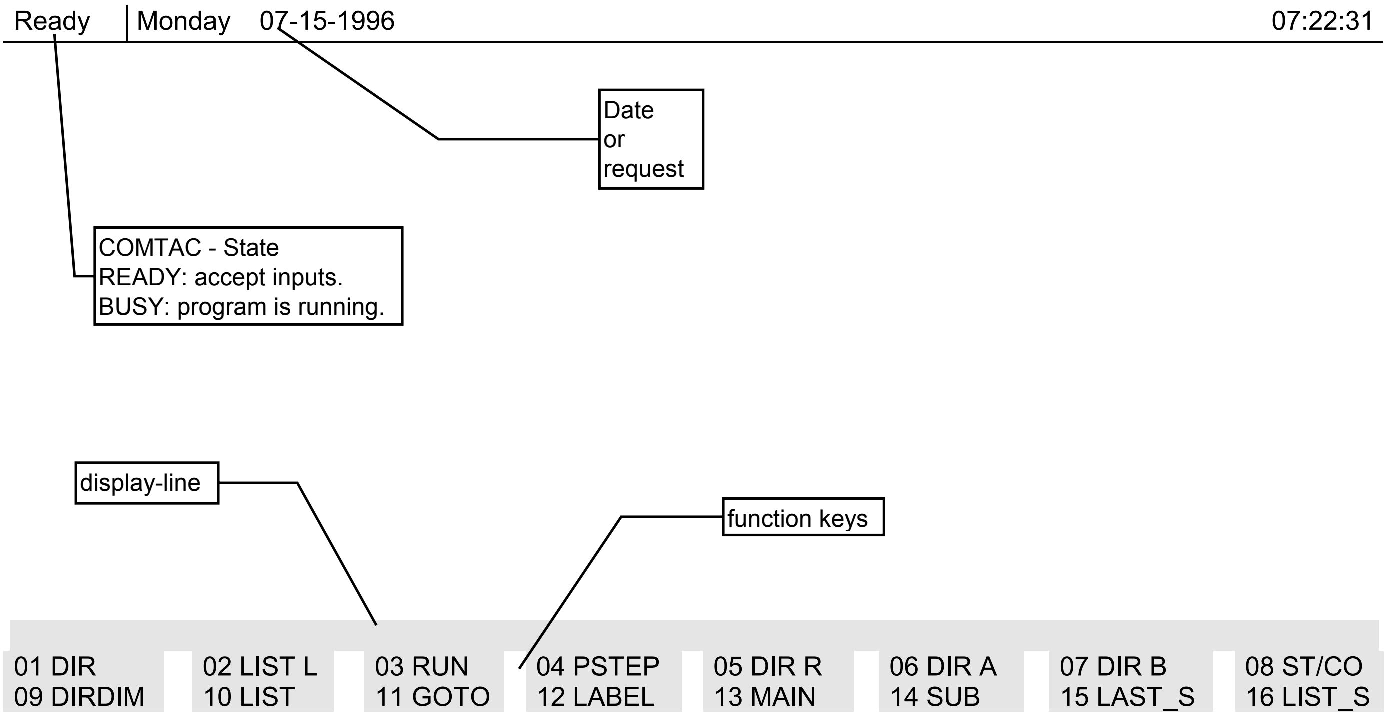
Function Overview

- Input of a command without a line number: the command is executed immediately.
- Write programs.
  A preceding line number to a command (= statement) is interpreted as a program line. This line is temporarily stored in the RAM.
  e.g.:
  10 GOSUB 1000
- Start a program.
- Store or load programs from the nonvolatile ZPRAM in the COMTAC or from a floppy drive (option HFM2).

⇨ The return key terminates an input procedure and executes the command typed in. All received inputs are stored in an input buffer and can be repeated or edited with the usual cursor, insert and delete functions.

Monitor Mask

After power on the operating system of COMTAC starts the command mode (if there is no auto start program running) and sets up the following mask:



```
Ready | Monday   07-15-1996                                              07:22:31

01 DIR       02 LIST L   03 RUN    04 PSTEP   05 DIR R   06 DIR A   07 DIR B    08 ST/CO
09 DIRDIM    10 LIST     11 GOTO   12 LABEL   13 MAIN    14 SUB     15 LAST_S   16 LIST_S
```

- Line 1: Info line
- Line 3 - 21: Input area
- Line 22: Display line
- Line 23-24: Function keys

Info Line

The first part of line 1 displays the COMTAC state.
The second part of line 1 is used to display error messages, date and time and dialog text.

Input Area

This area is used to input and edit commands or program lines and list program lines.

Edit Functions of the Input Area

The input area consists of 19 input lines. 18 of these are stored in an input buffer and thus displayed on the screen. After the input of the 18th. line the screen is scrolled causing the first line to disappear.

Within these 18 lines the cursor can be moved to any line to edit or repeat commands. The cursor marks the actual line to be edited or executed. With the RETURN key the contents of this line are stored in the input buffer and the command in this line is executed. When the line starts with a number it is stored as a program line in the program memory. Without a line number the command is executed immediately. Program errors are recognized after the start of a program.
The following paragraphs describe the detailled edit functions of COMTAC.

Display Line

Display line for the PRINT@-command.

Function Key Code

These functions apply to terminals which can be programmed with the key codes listed below (e.g. TV905, TV910, TV925, TV950, TV9065).

Key	Code				Function	Description
	Keyboard		hex			
F1	Ctrl_A	0	0x01	0x30	DIR	Switch on the DIR mode - display the directory of a drive.
F2	Ctrl_A	1	0x01	0x31	LIST_L	List a program (SUB/MAIN). The listing starts with the line number which was entered with the latest LIST command.
F3	Ctrl_A	2	0x01	0x32	RUN	Start a program. The program starts with the line number marked by the cursor (SUB/MAIN). If there is no line number in this line the program starts with line 1 of the MAIN. Variables are cleared.
F4	Ctrl_A	3	0x01	0x33	PSTEP	Single step. Execute one program line and wait for F4 again. The program line marked by the cursor is executed first. If this line has no line number the next BASIC statement of the program is executed.
F5	Ctrl_A	4	0x01	0x34	DIR R	Switch on the DIR-Mode. Display the contents of drive R
F6	Ctrl_A	5	0x01	0x35	DIR A	Switch on the DIR-Mode. Display the contents of drive A
F7	Ctrl_A	6	0x01	0x36	DIR B	Switch on the DIR-Mode. Display the contents of drive B
F8	Ctrl_A	7	0x01	0x37	ST/CO	A running program is stopped (STOP) or a stopped program is continued (CONTinue). This function can be enabled/disabled via COMTAC parameter 13.
F9	Ctrl_A	8	0x01	0x38	DIRDIM	Display the arrays stored in drive R.
F10	Ctrl_A	9	0x01	0x39	LIST	List a program. The listing starts with the line marked by the cursor. If there is no line number in the edit line the display continues with the last displayed line
F11	Ctrl_A	:	0x01	0x3A	GOTO	Start a program (SUB/MAIN) at the line marked by the cursor. If this line has no line number the program starts with line 1 of the MAIN. Variables are not cleared.
F12	Ctrl_A	;	0x01	0x3B	LABEL	List a program (SUB/MAIN) beginning with the label which appears in the line marked by the cursor. If there is no label the command is not executed.
F13	Ctrl_A	<	0x01	0x3C	MAIN	List the MAIN beginning with the first line.
F14	Ctrl_A	=	0x01	0x3D	SUB	List a SUB beginning with the sub which appears in the line marked by the cursor. If this line has no SUB name the next possible SUB is listed
F15	Ctrl_A	>	0x01	0x3E	LAST_S	List the last but one SUB.
F16	Ctrl_A	?	0x01	0x3F	LIST_S	List all SUB names.

HOME Key

Moves the cursor to the home position (first line of the input area). The contents of the edit buffer are not affected.

CursorKeys

Move the cursor one line in the vertical or one character in the horizontal direction.
Cursor key and Home Key: horizontal keys - move to the beginning or the end of a line. Vertical keys - move to the first or last line of the input field-

{CLEAR SPACE}[1] or Ctrl+T or Ctrl+Z

Delete the input buffer and move the cursor to the home position.
If COMTAC was in the RUN mode the screen is deleted and renewed.

{PAGE ERASE} or Ctrl+P

The screen is deleted and renewed.

DEL or {CHAR DELETE}

Delete the character at the cursor position.

[1] The keys in brackets "{}"are for use with a TV905 Terminal.

{CHAR INSERT} or Ctrl+B
Insert one or more characters at the cursor position
Press this key again to switch off the insert function.

{LINE DELETE} or Ctrl+Y
The line marked by the cursor is deleted.

{LINE INSERT} or Ctrl+I or TAB
Insert a new line after the cursor position.

{LINE ERASE} or Ctrl+E
The line marked by the cursor is deleted from the cursor position to the end of the line.

Ctrl+C
Stop a running program. Switch from the RUN mode to the COMMAND mode.

Ctrl+R

Cancel all active functions, reset all interfaces.
The user program in the RAM is retained.

Ctrl+A then R
Cold start of COMTAC.

Ctrl+S
Stop the output of a listing.

Ctrl+Q
Continue a stopped output of a listing.

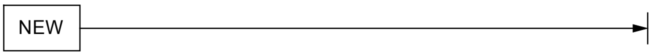
Ctrl+U
Convert a program from the compiled format to the uncompiled format.

{PAGE} or Ctrl+X
Start the DATA-TEXT-EDITOR.

Ctrl+O
Interrupts the display of the real time in the info line.

7.2 Edit functions

7.2.1 NEW



Function:

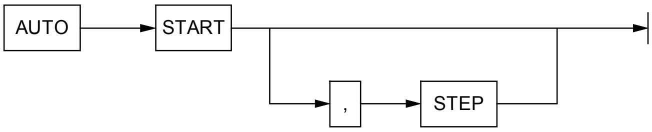
Delete a program which is stored in the RAM (MAIN and SUB(s)). Clear all variables (including those created with the DIM statement), strings. Reset all interrupts, the control and argument stack.

Example: NEW

Note:

The contents of the Data Text Array, the actual value of the TIMER, the Strings $(#0), $(#1), $(#2), $(#3), $(#4) and matrixes in the ZP-RAM (drive R) are not deleted.

7.2.2 AUTO



Function:

Activate the function to count program lines automatically. After each RETURN a new line is prompted.

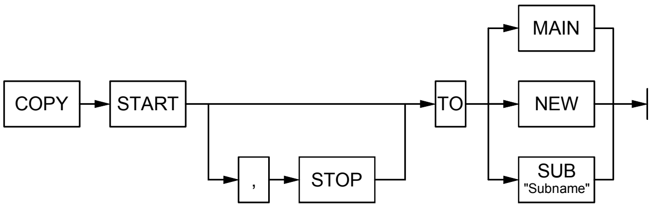
Parameter	Input	Range	Description
Start	Number	0 - 65535	Start line number
Step	Number	1 - 65535	Step value

Example: AUTO 100
AUTO 100,5

Note:

The default step value is 10.
Ctrl-C cancels this function.

7.2.3 COPY [C.] Basic Lines



Function:

Copy one or more BASIC lines.

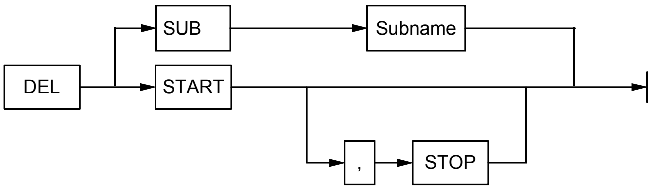
Parameter	Input	Range	Description
Start	Number	0 - 65535	First line to be copied
Stop	Number	0 - 65535	Last line to be copied
New	Number, Name	0 - 65535	Target where the lines are copied to.

Example: COPY 30 TO 200 or COPY 100, 260 TO 1000

Note:

If they exist, lines at the destination will be overwritten.
The step value is stored.
If the specified SUB doesn´t exist a new one is created.

7.2.4 DEL Basic Lines



Function:

Delete

- one or more line(s) in the active program (MAIN or SUB).
- a complete SUB.

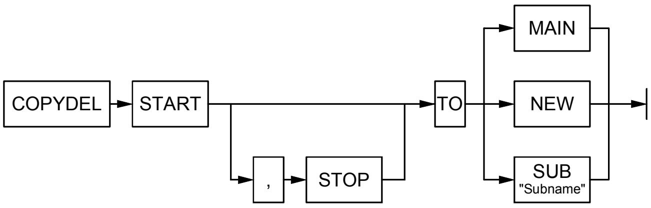
Parameter	Input	Range	Description
Start	Number	0-65 535	First line to be deleted
Stop	Number	0-65 535	Last line to be deleted

Examples: DEL 30 or

DEL 70,200

DEL SUB "Test"

7.2.5 COPYDEL



Function:

Copy one ore more BASIC lines and delete the source lines.

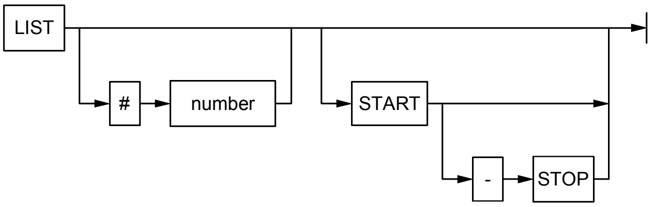
Parameter	Input	Range	Description
Start	Number	0 - 65535	First line to be copied
Stop	Number	0 - 65535	Last line to be copied
New	Number	0 - 65535	Target where the lines are copied to..

Example: COPYDEL 100 TO 500
COPYDEL 230,440 TO 20000

Note:

If they exist, lines at the destination will be overwritten.
The step value is stored.
If the specified SUB doesn´t exist a new one is generated.

7.2.6 LIST



Function:

List the active program (MAIN or SUB)

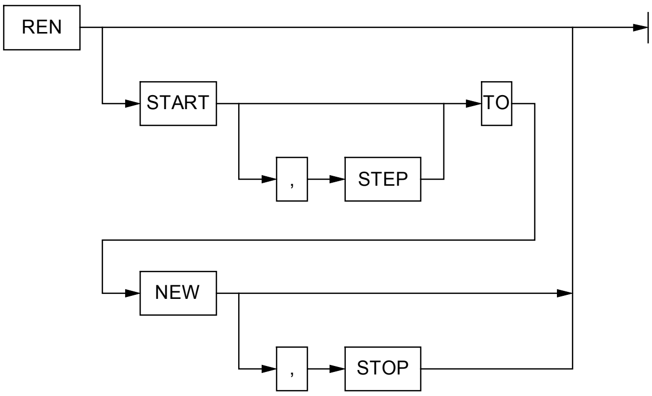
Parameter	Input	Range	Description
Index	Number	0 - 9	Interface number
Start	Number	0 - 65535	Start listing with line number ...
Stop	Number	0 - 65535	Stop listing with line number ...

Example: LIST or LIST#1 10-200

Note:

- If no interface is specified the lines are listed at the terminal.
- Ctrl-S stops the listing, Ctrl-Q continues it, Ctrl-C finishes the listing.
- After 18 lines are listed the output stops automatically. Another LIST command continues the listing for the next 18 lines.
- Function key SHIFT F2
  - releases a listing
  - continues a listing
  - starts a listing at the line number typed in
- Type in a line number and press RETURN releases a listing, beginning at this line number.
- Function key F2 repeats the last LIST command.

7.2.7 REN



Function:

Renumber the BASIC lines in the active program (MAIN or SUB).

Parameter	INPUT	Range	Description
Start	Number	0 - 65535	Renumber starts with this line number
Step	Number	1 - 65535	Step value
New	Number	0 - 65535	First line after renumber
Stop	Number	0 - 65535	Last line to be renumbered

Example: REN or REN 10,5 TO 100,90

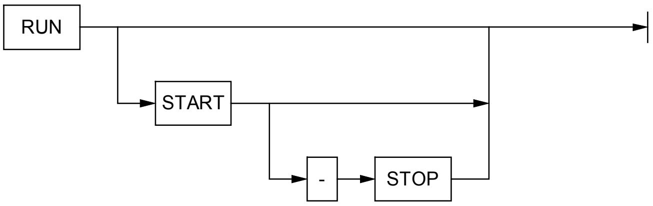
Note:

Default values for the REN function:
START = first line of the program
STEP = 10 NEW = 10
STOP = last line of the program

➡ GOTO & GOSUB line number branches are calculated automatically.

7.3 Program Run

7.3.1 RUN



Function:

Clear all variables, reset all interrupts, compile the program and start it running. .
An optional start address starts the program at this line number.
An optional stop address stops the program run at this line number. The stop line is not executed.

Parameter	Input	Range	Description
Start	Number	0 - 65535	Line **No**. program start
Stop	Number	0 - 65535	Line **No**. program stop

Example: RUN or RUN 10 - 100

Note:

- The program run can be stopped with the function key F8 of the COMTAC keyboard, if enabled.
- The program run can be stopped with the function key F8 or CNTL C of the terminal key board.
- CNTL R always stops a program run and resets the COMTAC without deleting the program.
- The command RUN (without line number) starts the MAIN program.
- Starting a SUB:
  - Change to the SUB
  - Enter „RUN *line number*“.

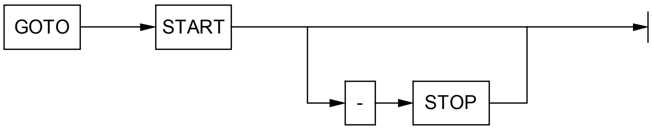
Compiling a program

A program is compiled automatically when it is started. Here, all logical jump addresses (GOTO, GOSUB) are converted into physical addresses in order to reduce the execution times of these branches. The required time to carry out this function depends on the size of the program. To avoid the compilation time, a program can be stored in the compiled format after being run for the first time.
A program in the compiled format can´t be loaded into a SUB, it first has to be recompiled.

Recompile a Program

Load the program as a MAIN program, with no SUB loaded at the same time.
CNTL U will recompile the program.

7.3.2 GOTO [GT.]



Function:

Start a program at the specified START line.
The program run stops at the STOP line
The STOP line isn´t executed.

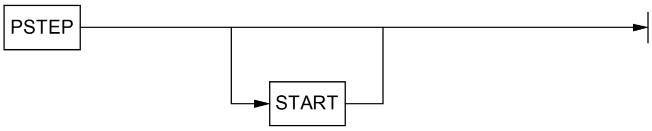
Parameter	Input	Range	Description
Start	Number	0 - 65535	START line number
Stop	Number	0 - 65535	STOP line number

Example: GOTO 30
GOTO 30 - 400

Note:

- The program run can be stopped with the function key F8 of the COMTAC keyboard, if enabled.
- The program run can be stopped with the function key F8 or CNTL C of the terminal key board.
- CNTL R always stops a program run and resets the COMTAC without deleting the program.
- Starting a SUB:
  - Change to the SUB
  - Enter “GOTO *linenumber*“.

7.3.3 PSTEP [PS.]



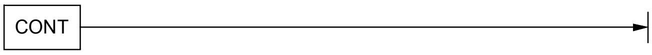
Function:

One program line is executed. The next PSTEP command executes the following program line (single step mode).

Example: PSTEP or PSTEP 50

- Single step mode of a SUB:
  - Change to the Sub.
  - Enter “PSTEP *linenumber*“.

7.3.4 CONT



Function:

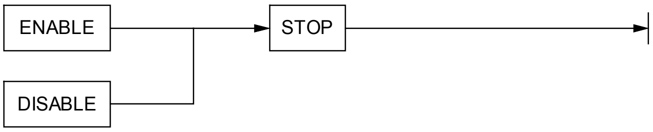
Continue a previously stopped program run.

Example: CONT

Note:

The program run will not be continued if the program was edited or an error occured.

7.3.5 ENABLE [EN.]/DISABLE [DI.] STOP



Function:

Enable / Disable, CNTL C and function key 8.

Example: ENABLE STOP
DISABLE STOP

Note:

After the DISABLE STOP has been executed the program run can´t be stopped with the CNTL C or function key 8.

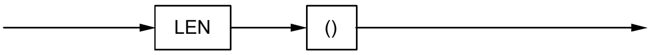
7.4 System Variable

7.4.1 FREE

Function:

This variable displays the free memory space for user programs(RAM)
FREE = MTOP - LEN()
Example: PRINT FREE
DISP FREE

7.4.2 LEN()

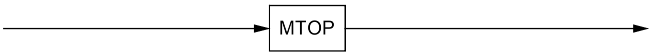


Function:

This variable displays the length (=required memory space) of the active program in the RAM.

Example: PRINT LEN()
DISP LEN()

7.4.3 MTOP

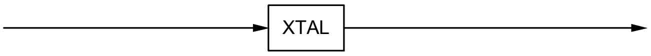


Function:

This variable displays the maximun available memory space of the COMTAC (RAM).
After power on COMTAC checks this RAM.
MTOP has the value 65535 (COMTAC2000) or 327679 (COMTAC 3000).

Example: PRINT MTOP
IF MTOP <> 65535 THEN...

7.4.4 XTAL



Function:

This variable displays the system clock of the BASIC system.

8. Storing and Loading Programs and Data

8.1 Storage Media

Using COMTAC, BASIC-programs and data can be stored in the internal ZPRAM (drive R) or on a floppy disk (external drive). The file name may consist of 8 characters and an additional extension of 3 characters, separated with a decimal point: nnnnnnnn.eee.
The drive is selected by a drive name:
R = ZPRAM
A = floppy disk A (HFM2)
B = floppy disk B (HFM2). HFM2 is an option.

Nonvolatile memory for Programs and Data:

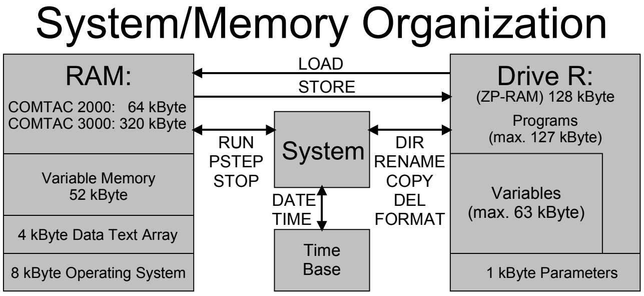
ZPRAM (Drive R): 128kByte.

RAM:

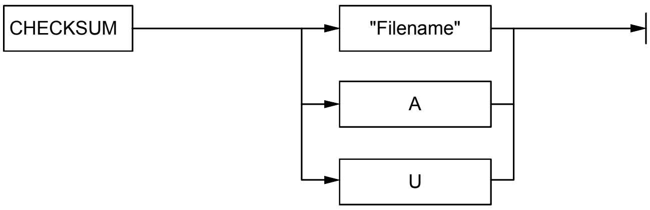
S-RAM with 128kByte (384kByte with COMTAC 3000).

- The maximum possible length of a BASIC program is 64kByte (320kByte with COMTAC 3000).
- 64kByte are reserved for variables.

Memory Organization



8.2 Checksums



To prevent data loss in the ZPRAM COMTAC calculates checksums for the following types of file:

- each program file
- each data file
- each array
- the system parameters (System CW's; Ct. 0 ... Ct. 100)
- the user parameters (User CW's; Ct. 101 ... Ct. 200)

COMTAC handles these different checksums as follows:

Program - file	Data file	Array/ User CW's	System CW's
Calculation of the checksum			
when storing	1. always when storing 2. automatically at any access to an element*1 3. by command	1. automatically at any access to an element*1 2. by command	1. automatically at any access to an element
Check of the checksum*1			
when loading	1. At power on or reset 2.when loading	1. At power on or reset	1. At power on or reset
Displayed Error message			
Checksum Error in Program	Checksum Error in Data	Checksum Error in Data / User CWs	Checksum Error in System CWs

*1 This function can be disabled

Modes of the Checksum Generation

Set up of CT.(13)	Program/ System CWs	Data file /Arrays	User CWs
CHECKSUM-check off	Bit 11=0	Bit 4=0	Bit 6=0
CHECKSUM-check on	Bit 11=1	Bit 4=1	Bit 6=1
CHECKSUM-automatically generated	-	Bit 5=0	Bit 7=0
CHECKSUM-generated by command	-	Bit 5=1	Bit 7=1

Each modification of a data file, array or parameter (CW) initiates the calculation of the checksum.
The automatic calculation of checksums extends the program execution times therefore, this function should be disabled after all program changes are complete. It is useful to calculate checksums by command.

Error Detection

If a checksum error occurs in a data file or a matrix during power on or reset the name of this file is stored in the string $(#0).

8.3 File Names

File names can consist of letters, numbers and the following special characters:
) $ ^ ! (- @ } ' & _ { ~ ^ %
Small letters are changed into capital letters.
The COMTAC commands
FORMAT; DIR; STORE; LOAD; DEL; COPY; RENAME
are used to support file handling. These commands are described below

8.3.1 Program-File

A program file name may consist of 8 characters and an extension of 3 characters max., separated with a decimal point: nnnnnnnn.eee.
Two extension types are reserved for special files:

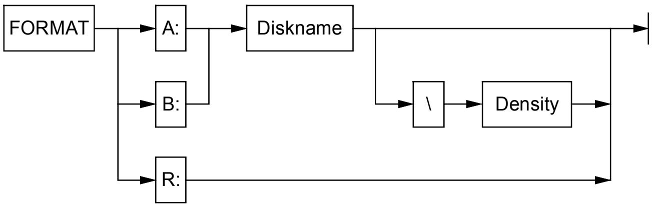
- '.ASP' characterises a BASIC program with the Auto Start Function.
- '.DAT' characterises a COMTAC data file.

Other extensions can be used for any other files.

8.3.2 Data-File

A data file name may consist of 8 characters. The extension for this type of file is always .DAT. COMTAC recognises files with a .DAT extension as data files.

8.4 Format



Function:

Format the ZPRAM (drive R) or a disk. The FORMAT command deletes all data stored on the specified drive.

Density parameter:

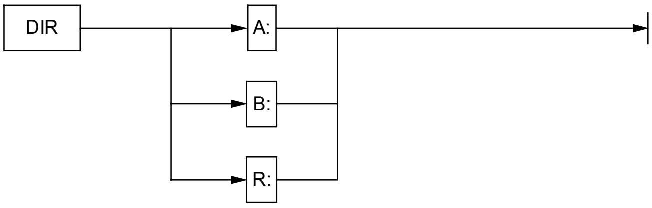
- 'N' = double density (720k Byte)
- 'H' = high density (1.44 M Byte).

Default is double density
Example :
FORMAT B: "COMTAC.001" \ N
FORMAT A: $(4)
FORMAT R:

Note:

A disk in drive A or B can be labeled.

8.5 DIR



Function:

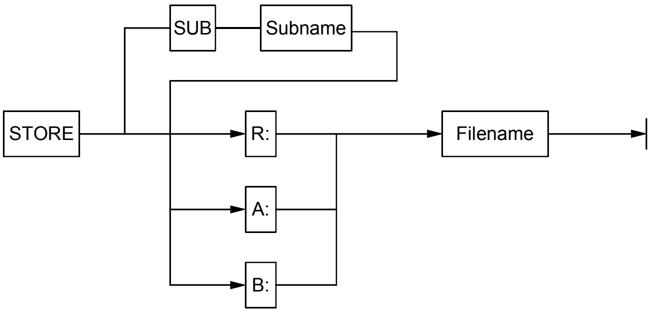
Activates the directory (DIR) mode of COMTAC. The contents of the specified drive (R, A, B) are displayed at the treminal and the COMTAC display

Example: DIR A: (external drive)
DIR B: (external drive)
DIR R: (ZPRAM)

Note:

In this mode a file can be selected by the cursor keys. The commands COPY, DEL, STORE, LOAD, DIR, RENAME and FORMAT can be executed. These commands are menu driven.

8.6 STORE [S.]



Function:

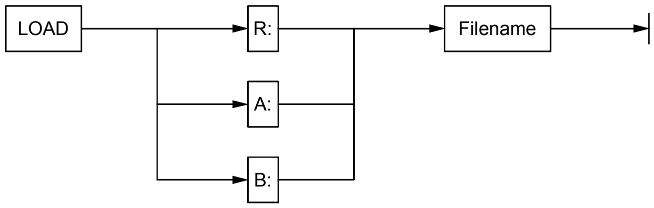
Store a program or data from the RAM on the selected drive (R, A, B)

Example: STORE A: "BAHN.TXT"
STORE R: UMEM$(1) [1,8]

Note:

This command can be used in a BASIC program.
The file name can be entered directly or a string can be used.
The Data Text Array (4kByte of the RAM) is allocated when the extension .DAT is used.

8.7 LOAD [L.]



Function:

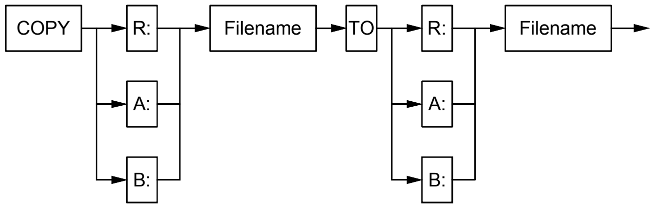
Load a program or data file from a drive(R, A, B) into the RAM of the COMTAC.

Example: LOAD A:"DEMO.BAS"
LOAD R:"TEST.DAT"
LOAD A:$(7)

Note:

This command can be used in a BASIC programs.
The file name can be entered direct or a string can be used.
If this command is executed in a BASIC program, the program which was loaded starts when the load procedure has finished.
Attention: Variables are cleared.

8.8 COPY [C.] File



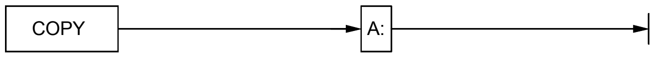
Function:

Copy program or data files- form drive to drive.

Example: COPY R: "DEMO.TXT TO A: "TEST2.TXT"
COPY R: $(1) TO A: $(1)

Note:

8.9 COPY Disk



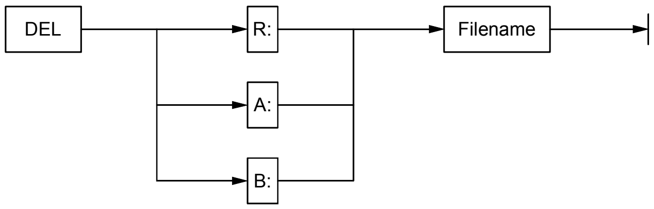
Function:

Copy the contents of the disk in drive A to the disk in drive B.

Note:

This command can be used in a BASIC program.

8.10 DEL File



Function:

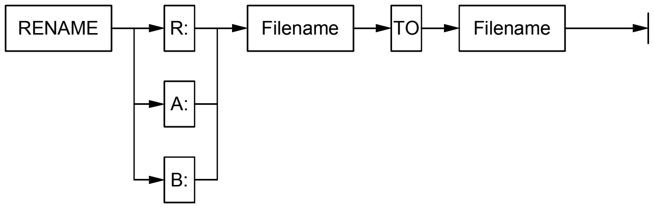
Delete specific program or data files on the drives A, B, or R.

Example: DEL A:"AUTOKOM.BAS"
DEL R:$(A)

Note:

This command can be used in a BASIC program.
The file name can be entered direct or a string can be used.

8.11 RENAME



Function:

Change a file name .

Example: RENAME R: "MASCH" TO "MASCH1"
RENAME B: "RTEST" TO "RTEST.TXT"

Note:

This command can be used in a BASIC program.
The file name can be entered direct or a string can be used.

8.12 AUTO-START Function

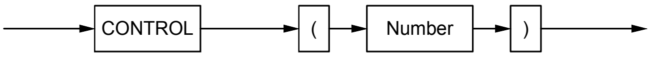
After power on COMTAC searches first in the ZPRAM for a file with the extension .ASP. If there is no such file, the disk in drive A will be searched.
If an .ASP file is found: this program is copied to the RAM and is started.
The Auto Start function can be disabled or enabled with the autostart flag in parameter 13.

8.13 DEBUG-Function

During testing the executed line number can be displayed on the COMTAC display .
This function is enabled with parameter 16. Parameter 16 = 0 disables the DEBUG function.

9. Parameter

9.1 CONTROL [CT.]



Function:

Parameters 0 to 100 are used to hold system and interface information. These parameters can be changed within their limits.
These Parameters are described on the following pages.
Parameters 101 to 200 are free, e.g. to store user program or machinery data.

Parameter	Input	Range	Description
Number	numeric Expression	0 - 200 0-100 101-200	Parameter -Number COMTAC System Parameters User Parameters

Example 1: PRINT CONTROL(10)

Example 2: CONTROL(X) = 100

9.2 Parameter Overview

No.	Usage
00...09	COMTAC System parameters
10...19	COMTAC - System parameters
20...29	reserved
30...39	reserved
40...49	RS232/3 - Interface parameters
50...59	RS232/1 - Interface parameters
60...69	RS485/1 - Interface parameters
70...79	RS485/1 - Interface parameters (field bus)
80...89	RS485/2 - Interface parameters
90...99	RS232/2 - Interface parameters
100	Assigns an interface to the terminal functions
101...200	user specific

9.3 System Parameters

No.	Function	Default
00	COMTAC-Type; set with power on	2000/ 3000
01	Date of the operating system software	-
02	Terminal-Typ 0 = TV 905 3 = VT100	0
03	Manufacturing number	-
04	Date of test	-
05	Power on *value* for O16 ... O1 After power on or CNTL R the digital outputs O16 .. O1 are set to *value*: BINOUT(16~1)=Parameter 5	0
06	Power on *value* for O32 ... O17 After power on or CNTL R the digital outputs O32 .. O17 are set to *value*: BINOUT(32~17)=Parameter 6	0
07	Power on delay 1 - 255 1=100ms	50
08	REQOUT Time-Out 1 - 255 1 = 100 ms	10
09	Repeat time for COMTAC-key board 1 - 255 1 = 100 ms	1
10	Assign an interface to the floppy drive 0 = RS232/1 is linked to the drive 1 = RS485/1 is linked to the drive 2 = RS232/2 is linked to the drive 3 = RS485/2 is linked to the drive 4 = RS232/3 is linked to the drive 99 = no floppy drive connected	99
11	Floppy-Timeout value 0 - 255 (1 = 0.1s) 0 = no timeout control	10
12	Floppy baudrate 4 = 2400 5 = 4800 6 = 9600 7 = 19200 8 = 38400	7
13	System-Flags	0
	Bit Function	
	0 **Parallel-Input** 0 = off, 1 = on (Terminal / Keyboard)	0
	1 **Listing-Option**; 0 = Normal, 1 =abbreviations	0
	2 **Keyboard autorepeat**; 0 = off, 1 = on	1
	3 **F8-Function** 0=Stop 1:disabled	0
	4 **Checksum-Data** 0=off 1=on	0
	5 0=auto 1=Com.	0
	6 **Checksum-User Param.** 0=off 1=on	0
	7 0=auto 1=Com.	0
	8 **Auto start function ZPRAM**; 0 = off, 1 = on	1
	9 **Auto start function diskette**; 0 = off, 1 = on	0
	10 -	0
	11 **Checksum: program and system parameter**; 0 = off, 1 = on	0
	12	0
	13 **Date output**; 0 =engl. 1 = ger.	0
	14	0
	15 **Ready beep**; 0 = off, 1 = on	0

14	Position of the time display in the COMTAC display (LCD). 0 = display disabled 1...160 = position of the first character	33
15	reserved	-
16	Position of the BASIC line display in the COMTAC display (LCD) for the DEBUG function 0 = display disabled 1...160 = position of the first character	0
17	Timeout (0=,5s)	0
18	Password Protection	-
19	Extended Password Protection	-

Parameters 10 to 12 determine the functions of the floppy drive HFM2. All other adjustments for the drive are performed automatically by COMTAC.

The COMTAC parameters are set to the default values when the function key F12 is pressed during power on.

9.4 Variable Timeout for Up/Download

With the aid of P17 you may select the Timeout for receiving signals from the PC in steps from 0,1 to 25,5s.
P17 = 0 implies a default value of 0,5s.

This is necessary in order to enable an Up/Download by modem as well.

9.5 Password Protection

With the aid of P18 you may select a so-called password protection in order to protect the basic program from unauthorised access.

P18 = 0 ⇨ password protection off
P18 <> 0 ⇨ password protection on

Activate/deactivate password protection

The password protection may be activated by entering a value (password) in P18.
By once again entering the same value (password) in P18, the password protection is deactivated.

Limited functions in case of activated password protection

If the password protection is activated, the following functions are not available:

- Program editing
- Program listing (LIST)
- Program download

Extended password protection

With the aid of P19 you may select the extended password protection.
P19 = 0 ⇨ extended password protection off.
P19 <> 0 ⇨ extended password protection on.

Activate/deactivate extended password protection

The extended password protection is activated by entering a value (password) in P19.
By once again entering the same value (password) in P19, the password protection is deactivated.

Limited functions in case of activated extended password protection

If the extended password protection is activated, the following functions are not available:

- Program Upload

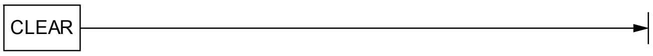
P18 and P19 give the value TRUE (65535) if they contain a password.

NV-Ram as working memory in COMTAC 3000 with device No. > 9082600000

The operating system is able to use also a NV-Ram (ZP-Ram) as working memory. This has the advantage that the current program must not be always memorised in the ZP-Ram or on disk.
The NV-Ram may be defined as working memory to the operating system with CT.(13) Bit 12 = 1
It is possible to start the program in the NV-Ram after Power On with the help of the autostart function.
This funciton is activated with CT.(13) Bit 10 = 1
In order to display possible loss of data in the working memory (NV-Ram) you may create a checksum by the following command: CHECKSUM P. This checksum will be newly calculated and compared to the old sum after Power On, if Bit 10 and Bit 11=1 in CT. (13).

10. General Commands

10.1 CLEAR [CL.]



Function:

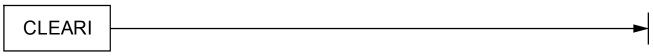
All variables , strings and interrupts are cleared. Previous executed DIM statements are no longer valid. The control and argument stack are reset.
Exception: Arrays defined in the ZPRAM are not changed.

Example: CLEAR

Note:

The contents of the Data Text Array, the TIMER and the strings $(#0), $(#1), $(#2), $(#3), $(#4) are **not** deleted.

10.2 CLEARI



Function:

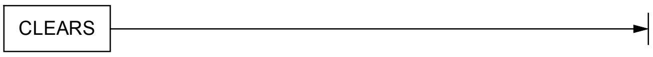
Clear all interrupts.

Example: CLEARI

Note:

The TIMER is not stopped.

10.3 CLEARS



Function:

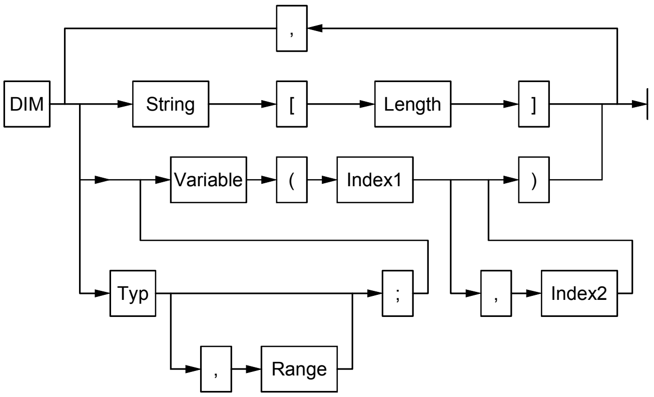
Resets the control and argument stack.

Example: CLEARS

Note:

This statement can be used to avoid a control stack overflow if there were FOR NEXT (or other) loops which were not closed correctly.

10.4 DIM



Function:

1. reserves memory space for strings.
2. reserves memory space for one or two dimensional arrays.

The option ´type´ defines the array's data type. The default type is floating point..

The option ´range´ determines to which pysical memory range the defined matrix is allocated. The default range is the variable memory (RAM).

Parameter	Input	Range	Description
$	-	-	stringname
Length	num. expr.	1-255	max. length of the string
Variable	-	-	variable name
Index 1	num. expr.	1 - 254	number of the elements of the matrix-1
Index 2	num. expr.	1 - 254	number of the elements of the matrix-1
Type	number	0 (1 Byte) 1 (1 Byte) 2 (2 Byte) 3 (2 Byte) 4 (4 Byte) 5 (4 Byte) 6 (6 Byte) 7 (4 Byte) 8 (4 Byte) 9 (6 Byte)	unsigned character character (+/- 127) unsigned integer integer long unsigned integer long integer floating point fix point, 3 places for the fractional part fix point, 6 places for the fractional part DSP Real (COMPAX)
Range	number	0 1 2 3 4 5 6	Variable memory RAM range 0...63k ZPRAM (drive R) (nonvolatile) RAM range 64...127k[*1] RAM range 128...191k[*1] RAM range 192...255k[*1] RAM range 256...319k[*1]

[*1] Only COMTAC3000

Example: DIM $(5)[80]
Reserves memory for the string $(5). The maximum string length is 80 characters.
DIM A(20)
defines a one dimensional array with 21 elements:
A(0), A(1), ... A(20).
DIM 1;POS(10,2)
defines a two dimensional array with 33 elements. The elements are POS(0,0) ... POS(10,0)
POS(0,1) ... POS(10,1)
POS(0,2) ... POS(10,2).
DIM 6,2;Z(5,0)
defines a one dimensional array with 6 elements:
Z(0), Z(1), ... Z(5).
The data type is the floating point format. The variables are stored in the nonvolatile ZPRAM (drive R).

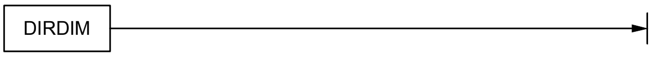
Note:

The DIM statement has to be used for:

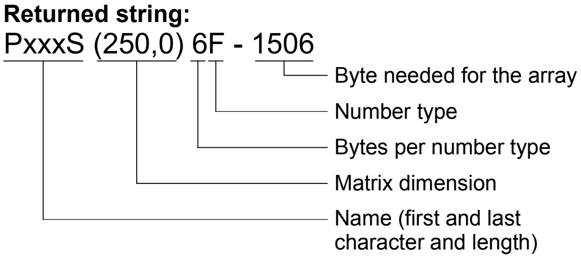
- Strings with a maximum string length different to the default length of 16 characters.
- one dimesional arrays.
- two dimensional arrays, when the number of elements is different to the default number of 121.

The commands NEW and CLEAR do not affect arrays delared in the ZPRAM. Therefore they have to be deleted with the DEL command.

10.5 DIRDIM



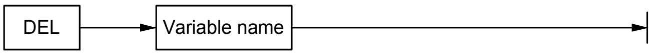
Function:
Display the arrays which are defined in the ZPRAM (drive R).
Returned string:



Notation:

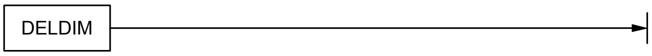
Format	Short form
unsigned character	UC
character	C
unsigned integer	UI
integer	I
long unsigned integer	UL
long integer	L
floating point	F
long int., 3 places for the fractional part	L.3
long int. 6 places for the fractional part	L.6
DSP Real	D

10.6 DEL Variable



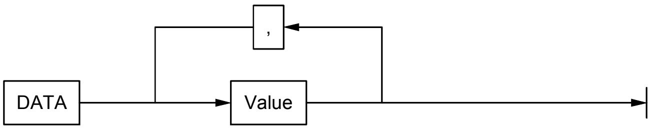
Function:
This statement delets the specified array in the ZPRAM.

10.7 DEL DIM



Function:
This statement deletes all arrays in the ZPRAM.

10.8 DATA



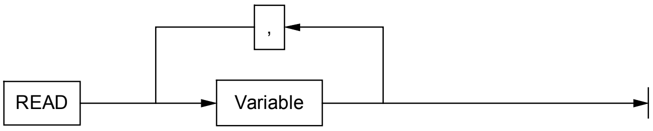
Function:

Statement to create an array in the BASIC MAIN module (Not allowed in a SUB!). The data can be read with the READ statement.

Parameter	Input	Range	Description
Value	num. expr.		data

Example: DATA 10,PI/2,A*B
DATA 50,100,150,200,250,300

10.9 READ



Function:

Statement to assign the data of the data array to variables.

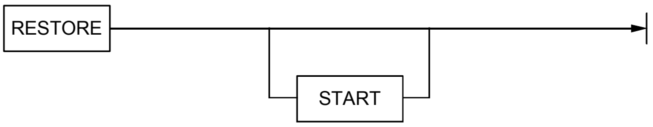
Parameter	Input	Range	Description
Variable	variable name		variable assigned a value from the data array.

Example: READ X,Y,Z or READ D1,D2,D3,D4,D5

Note:

After a program start (RUN, GOTO) the internal READ pointer points to the first DATA value.
Following a READ operation (one value was read) the READ pointer is incremented by 1.
The RESTORE statement resets the READ pointer to the first DATA value or to any specified line number.

10.10 RESTORE

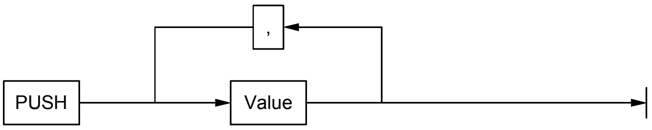


Function:

The RESTORE statement resets the READ pointer to the first DATA value or to any specified line number.

Example: RESTORE
RESTORE 100

10.11 PUSH



Function:

This statement is used to pass variables to subroutines and to store variables temporarily.
The value of the variable is stored on the argument stack.

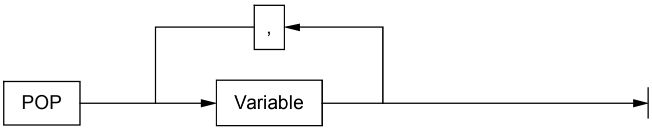
Parameter	Input	Range	Description
Value	num. expr.		Value to be stored on the argument stack

Example: PUSH A,B,C

Note:

Writing to the argument stack is organized as a FILO (first in last out) memory and can store a maximum of 39 numbers. The operating system also uses this stack.
Note: The POP statement assigns a value from the stack to a variable.

10.12 POP



Function:

The POP statement assigns a value stored on the stack to a variable.

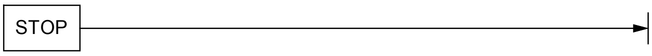
Parameter	Input	Range	Description
Variable	variable name		this variable gets a value from the stack

Example: POP X,Y,Z
FOR I=0 TO 10:POP X(I):NEXT I

Note:

The argument stack is organized as a FILO (first in last out) memory.

10.13 STOP



Function:

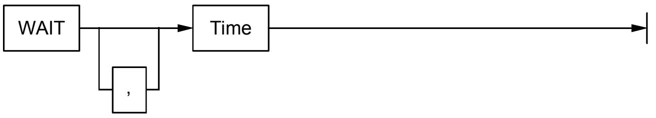
Stops the program run at any line in order to display or change variables.

Example: STOP

Note:

The CONT statement restarts the program.

10.14 WAIT



Function:

The program run stops, and waits the programmed time before the next statement is executed.

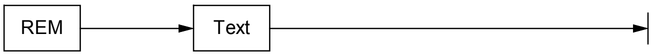
Parameter	Input	Range	Description
Time	no. express.	1 - 65535	Input in milli seconds

Example: WAIT 1000 or WAIT Z1

Note:

This statement is interrupted with CNTL C or an interrupt event.
Placing a comma before the number prevents an interrupt from cancelling the WAIT function (not the CNTL C interrrupt).

10.15 REM [!]



Function:

Insert a comment into the program.

Parameter	Input	Range	Description
Text			any text

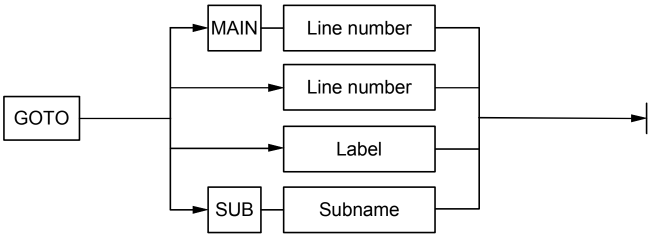
Example: REM This is a comment
! Start of subroutine 1
I=I+1:!a=I

Note:

Statements **after** the **REM** or **!** are **not** executed (REM ! have the same meaning).

11. Program Branches and Loops

11.1 GOTO [GT.] Line number



Function:

Unconditional jump to a line number or a SUB

Parameter	Input	Range	Description
Line number	Number	0 - 65535	Target

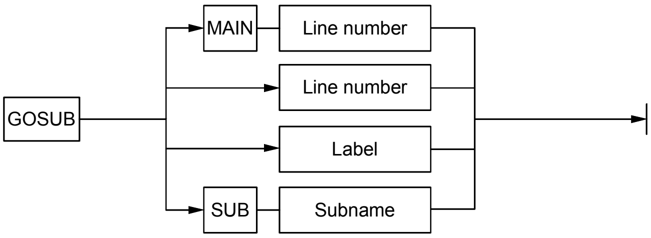
Example 1: GOTO 30

Example 2: GOTO SUB "Test"

Example 3: GOTO "Labelname"

Example 4: GOTO MAIN 2000

11.2 GOSUB [GS.]



Function:

Calling a subroutine.

Parameter	Input	Range	Description
Line number	Number	0 - 65535	First line of the subroutine

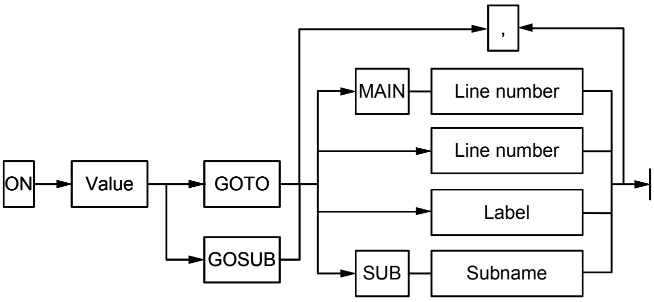
Example: GOSUB 500 oder GOSUB 30000

GOSUB SUB “subroutine“

Note:

A subroutine is terminated with the RETURN statement.

11.3 ON



Function:

Conditional branch function.
The entered value determines which given line number the program jumps to.
If the value is 0 the program jumps to the first given line number.
If the value is 1 the program jumps to the second given line number.
If the value is 2 the program jumps to the third given line number, and so on.

Parameter	Input	Range	Description
Value	number	0 - 100	branch condition
Linenumber	umber	0 - 65535	target line number

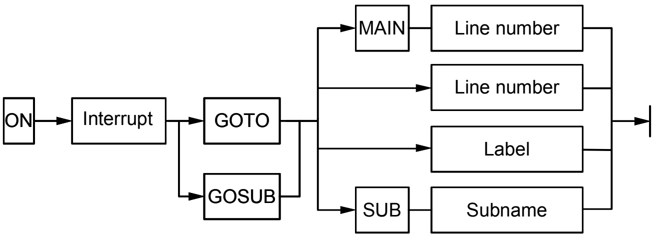
Example:

ON Q GOSUB 1000,2000,3000,4000,5000,6000,7000,8000
ON V-100 GOTO 300,400,500,600,10000

Note:

If the entered value is < 0 a ´Bad Argument Error´ occurs.
If the entered value is greater than the number of line numbers a`Syntax Error´ occurs.

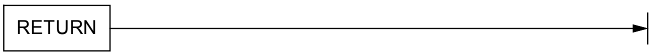
11.4 ON Interrupt



Function:

Program branch triggered by an interrupt event. Further discription of this function can be found in the chapter "Interrupt Handling“.

11.5 RETURN [R.]

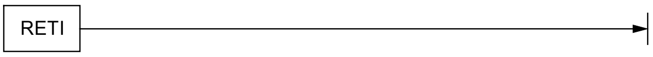


Function:

This statement terminates a subroutine.
The program jumps back to the place in the program from which the subroutine was called (GOSUB) and continues with the statement following the GOSUB statement.

Example: RETURN

11.6 RETI

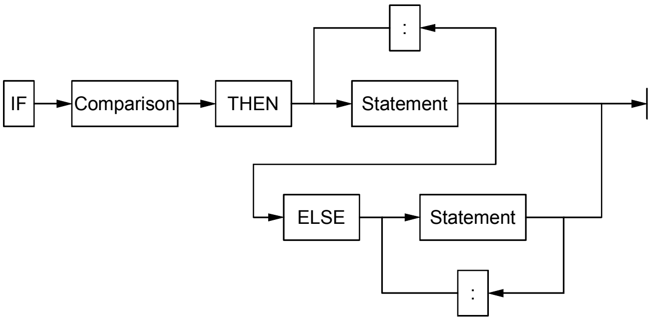


Function:

This statement terminates an interrupt subroutine.
The program jumps back to the place in the program which was interrupted and contiunues the program.

Example: RETI

11.7 IF THEN (ELSE)



Function:

This statement provides a conditional execution of statements.

1. If the result of the comparison is TRUE the sequence of statements following the THEN part is executed.
2. If the result of the comparison is FALSE the sequence of statements following the ELSE part is executed. The ELSE part is optional.

A compare expression may be numeric, a string, a logical function or a combination of these.
The COMTAC functions which have a TRUE or FALSE result may be used too: ASK- ,REQOUT, CPX_PE or WAITMN.

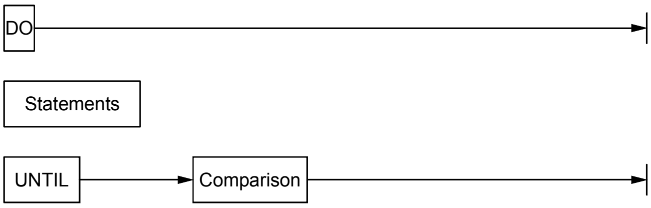
Examples:

```
IF A=10 THEN B=1 ELSE B=2
IF X>5 'AND' X<10 'OR' $(3)[1,1]="U" THEN GOTO 500
IF WFS 2 THEN GOSUB 1000
IF ROUT 1,5 THEN PRINT "OK" ELSE PRINT "Error"
IF (IN(1)=1'AND'IN(2)=0) THEN 200 ELSE 300
IF WAITMN(1) THEN GOTO 100
```

Note:

The ´GOTO´ after THEN and ELSE can be omitted. Complex comparisons should be separated with brackets to avoid errors.

11.8 DO ... UNTIL



Function:

This statement provides a conditional program loop..
The sequence of statements is executed until the comparison becomes TRUE.
A compare expression may be numerical, a string, a logical function or a combination of these.
The COMTAC functions which have a TRUE or FALSE result may be used to: ASK- ,REQOUT, CPX_PE or WAITMN.

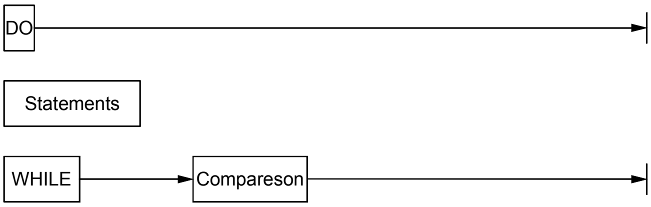
Example:

```
10 DO
20 A=A+1
30 PRINT A
40 UNTIL A=10    !    Jump to 10 till A=10
```

Note:

The sequence of statements is executed at least once.
Complex comparisons should be separated with brackets to avoid errors.

11.9 DO ... WHILE



Function:

This statement provides a conditional program loop.
The sequence of statements is executed repeatedly as long as the comparison is TRUE.
A compare expression may be numeric, a string, a logical function or a combination of these.
The COMTAC functions which have a TRUE or FALSE result may be used to: ASK- ,REQOUT, CPX_PE or WAITMN.

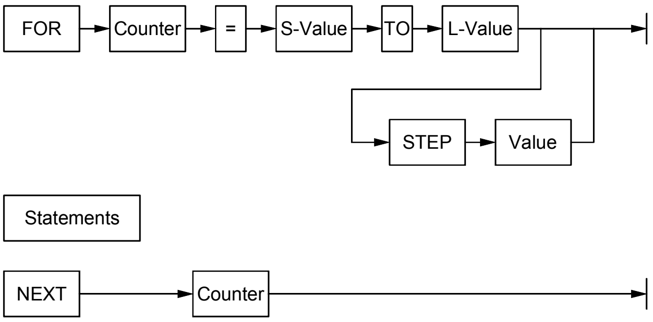
Example:

```
10 DO
20 A=A+1
30 PRINT A
40 WHILE A<10    !  Jump to 10 as long as A<10
                 !
```

Note:

The sequence of statements is executed at least once.
Complex comparisons should be separated with brackets to avoid errors.

11.10 FOR ... NEXT



Function:

This statement provides an iterative program loop.

The sequence of statements enclosed by the FOR and NEXT is repeated as long as the counter is greater than the limit value(l-value).

The start value (s-value) is initially assigned to the counter. The NEXT statement increments the counter by the STEP value and compares the result with the limit value. If the result is greater than the limit value the loop is terminated.

Parameter	Input	Range	Description
Counter	Variable		Variable name
s-value	num. expr.		Start value
l-value	num. expr.		Limit value
step-value	num. expr.		Step value

Example:

```
10 FOR I=1 TO 100 STEP 10
20 PRINT I
30 NEXT I
```

FOR-NEXT-loops can be nested:

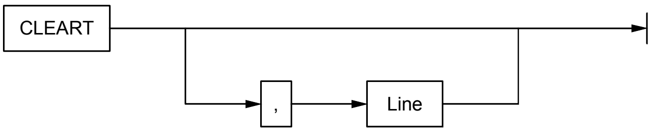
```
10 FOR I=1 TO 10
20 FOR N=1 TO 10
30 PRINT I,N
40 NEXT N
50 NEXT I
```

Note:

The step value may be omitted. The default value is 1.
The step value may be negative or positive.
The sequence of statements is executed at least once.

12. Terminal Output

12.1 CLEART



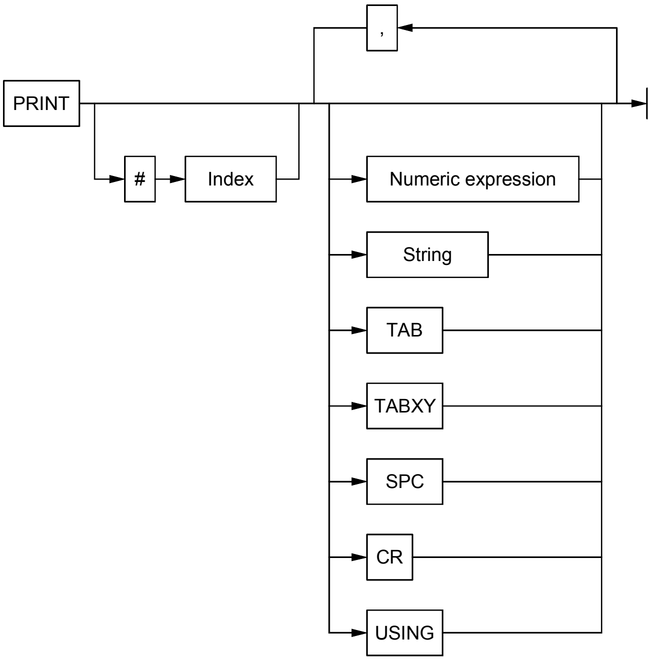
Function:

CLEART
Clear the whole screen.
CLEART,
Clear the screen starting at the actual cursor position.
CLEART,line
Clear the specified line
If the given line number is 0 the line marked by the cursor is deleted, beginning at the actual cursor position.

Parameter	Input	Range	Description
Line	**num. expr..**	**1-24**	**-**

Example: CLEART
CLEART,7

12.2 PRINT [P.]

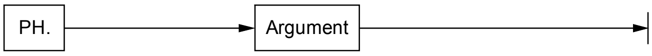


Function:

Read out characters, strings, values to the terminal or an interface (printer).

Example: PRINT A/30
PRINT#2 "COMTAC 2000"

12.3 PH.



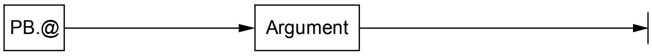
Function:

Read out a value in the hexadecimal format.

Parameter	Input	Range	Description
Argument	num. expr.	0-65535	displayed value

Example: PH. 55
PH. M/330

12.4 PB.@



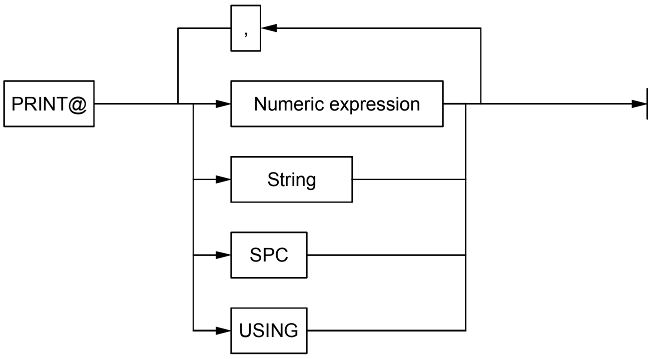
Function:

Read out a value in the binary format at line 22 of the terminal.

Parameter	Input	Range	Description
Argument	num. expr.	0-65535	displayed value

Example: DB. A*B+300
DB. 45000

12.5 PRINT@ [P.@]



Function:

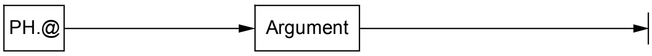
Read out a value at line 22 of the terminal.

Example: PRINT@ A
PRINT@ 800*33

Note:

A description of the SPC and USING functions can be found in the Format Commands Section.

12.6 PH.@



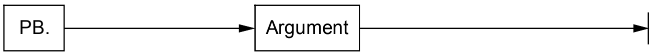
Function:

Read out a value in hexadecimal format at line 22 of the terminal.

Parameter	Input	Range	Description
Argument	num. expr.	0-65535	displayed value

Example: PH. 34
PH. X/2

12.7 PB.



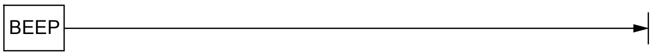
Function:

Read out a value in binary format.

Parameter	Input	Range	Description
Argument	num. expr.	0-65535	displayed value

Example: PB. V-845
PB. 377

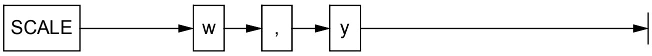
12.8 BEEP



Function:

Signal tone of the PC or a terminal.
COMTAC sends out the character 07 (BEL) to the terminal.

12.9 SCALE



Function:

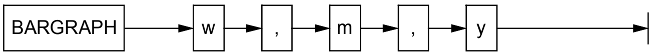
Display at line y and line y-1 a scale for the BARGRAPH function.
The scale beginns at column 30 and is lettered from 0 up to the entered value w*10.

Parameter	Input	Range	Description
w	num. expr.	1-10	maximum value of the scale
y	num. expr.	2-24	line number

Example: SCALE 3,10
With w=3 and y=10 the following scale is displayed at line 9 and 10 beginning at column 30:

```
0    3    6    9   12   15   18   21   24   27   30
|....|....|....|....|....|....|....|....|....|....|
```

12.10 BARGRAPH



Function:

Display a bar graph of the measured value at line y relative to the 100% value w. The bargraph begins at column 31.

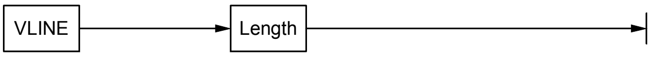
Parameter	Input	Range	Description
w	num. expr.	-	100%-value
m	num. expr.	-	measured value
y	num. expr.	1-24	line number

Example: BARGRAPH 30,15,11
With w = 30, m = 15 und y = 11 a bargraph is displayed at line 11.

Note:

The SCALE statement is used to add the measurement scale points.

12.11 VLINE [V.]



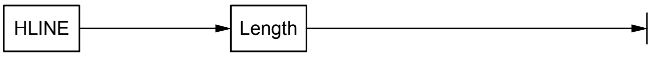
Function:

Draw a vertical line with the given length beginning at the actual cursor position.

Parameter	Input	Range	Description
Length	num. expr.	1 - 24	-

Example: VLINE Y
VLINE 8

12.12 HLINE [H.]



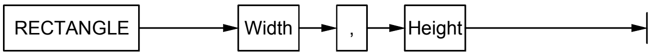
Function:

Draw a horizontal line with the given length beginning at the actual cursor position.

Parameter	Input	Range	Description
Length	num. expr.	1 - 80	-

Example: HLINE 5
HLINE X

12.13 RECTANGLE [RECT.]



Function:

Draw a rectangle beginning at the actual cursor position.

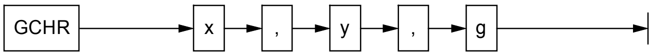
Parameter	Input	Range	Description
Width	num. expr.	1 - 79	
Height	num. expr.	1 - 23	

Example: RECTANGLE X,Y
RECTANGLE 7,9

Note:

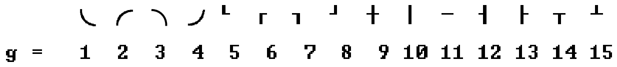
The rectangle will be drawed from the left to the right side and from the top to the bottom.

12.14 GCHR



Function:

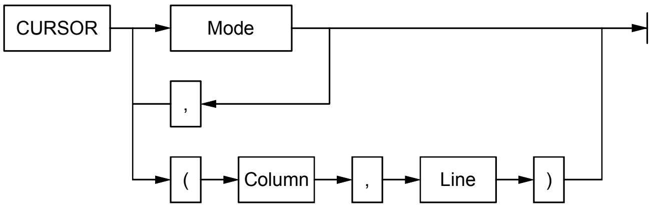
Display the character ´g´ at column x and line y. The following special characters are available:



Parameter	Input	Range	Description
x	num. expr.	1 - 80	Column
y	num. expr.	1 - 24	Line
g	num. expr.	1 - 15	Number of the character

Example: GCHR 5,5,10
With x = 5, y = 5 and g = 10 the special character “|” is displayed.

12.15 CURSOR [CU.]



Function:

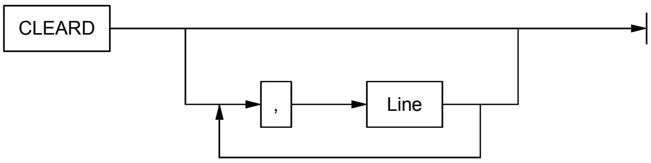
This statement sets the cursor mode and the cursor position.

Parameter	Input	Range	Description
Mode	num. expr.	0 - 4	Cursor mode 0 = off 1 = Block flashing 2 = Block static 3 = underscore flashing 4 = underscore static
Column		1 - 80	-
Line	num. expr.	1 - 24	-

Example: CURSOR 4, (10,10)
CURSOR 1

13. LCD Output

13.1 CLEARD



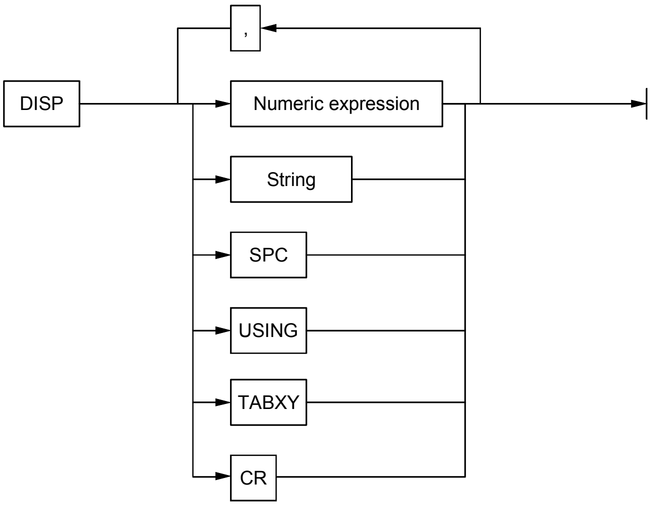
Function:

CLEARD
Clear the whole display.
CLEARD,
Clear the display beginning at the actual cursor position.
CLEARD,line
Clear the specified lines.
If the given line number is 0 the line marked by the cursor is deleted, beginning at the actual cursor position.

Parameter	Input	Range	Description
Line	num. expr.	1-4	-

Example: CLEARD
CLEARD,2
CLEARD,2,3

13.2 DISP [D.]

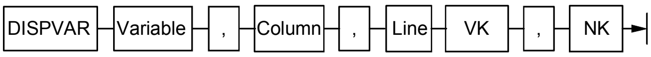


Function:

Display characters, variables, strings on the COMTAC display.

Example: DISP A
DISP 800*33

13.3 DISPVAR

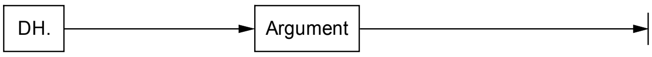


Function:

Cyclic display of a BASIC variable.
The integral part of the number (BD) and the fractional part (AD) must not exceed 8 digits.
The display can be stopped and stored with the statement STOP DISP. The statement CONT DISP continues the display (see page 51)

Example: DISPVAR A, 20, 2, 2, 4

13.4 DH.



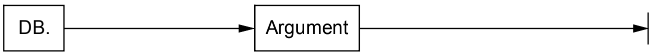
Function:

Display a value in the hexadecimal format on the COMTAC display.

Parameter	Input	Range	Description
Argument	num. expr.	0-65535	Displayed value

Example: DH. 34
DH. X/2

13.5 DB.



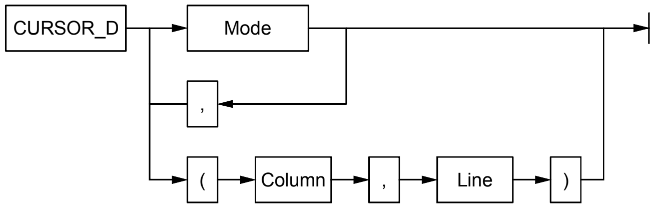
Function:

Display a value in the binary format on the COMTAC display.

Parameter	Input	Range	Description
Argument	num. expr.	0-65535	Displayed value

Example: DB. A*B+300
DB. 45000

13.6 CURSOR_D



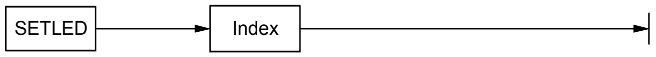
Function:

This statement sets the cursor mode and the cursor position.

Parameter	Input	Range	Description
Mode	num. expr.	0 - 4	Cursor mode 0 = off 1 = Block flashing 2 = underscore static 3 = underscore static 4 = underscore static
Column		1 - 40	-
Line	num. expr.	1 - 4	-

Example: CURSOR_D 4, (10,1)
CURSOR_D 1

13.7 SETLED



Function:

Switch on the specified LED (H1 - H4) of the COMTAC display.

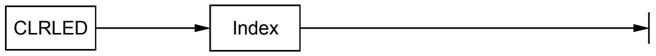
Parameter	Input	Range	Description
Index	num. expr.	1...4	Number of the LED

Example: SETLED 2 : REM switch on H2

Note:

A comma placed after the LED index makes the LED flash.

13.8 CLRLED



Function:

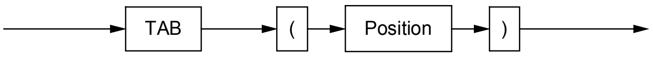
Switch off the specified LED (H1 - H4) of the COMTAC display.

Parameter	Input	Range	Description
Indexr	num. expr.	1...4	Number of the LED

Example: CLRLED 2 : REM switch off H2

14. Format Commands

14.1 TAB



Function:

This statement is used together with the PRINT statement in order to move the cursor. The entered number is the new cursor position.

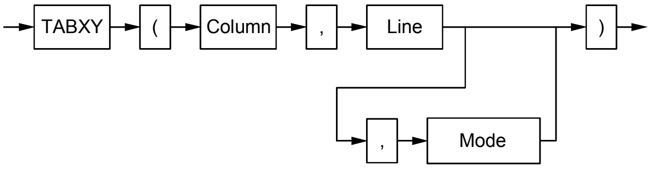
Parameter	Input	Range	Description
Position	num. expr.	1 - 79	Position of the Cursors

Example: PRINT TAB(30), "X"
PRINT#2 TAB(71),A

Note:

If the actual cursor position is equal to or greater than the given position the command is ignored.

14.2 TABXY [T.]



Function:

This statement is used together with the PRINT/DISP statement in order to set the cursor position and the display mode of the screen. The display mode is only for the terminal, not for the COMTAC display.

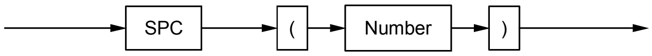
Parameter	Input	Range	Description
Column	num. expr.	1 - 80 (1-40)	Values in brackets are for the COMTAC display
Line	num. expr.	1 - 24 (1-4)	Values in brackets are for the COMTAC display
Mode (only for the terminal)	num. expr.	0 - 14	Mode of the display 0= normal 2= flashing 4= inverse 6= inverse,flashing 8= underscored 10= underscored, flashing 12= underscored, inverse 14= underscored, flashing, inverse

Example: PRINT TABXY(10,10,4),A
PRINT TABXY(10,15),B

Note:

The *´MODE´* parameter is only used with the PRINT statement.
This statement only works with a PRINT or DISP statement.

14.3 SPC



Function:

This statement displays the specified number of blanks.

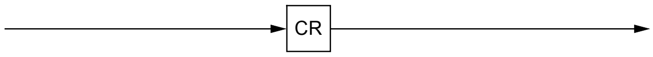
Parameter	Input	Range	Description
Number	num. expr.	0 - 14	Number of blanks

Example: PRINT SPC(90)
PRINT% A, SPC(5),3

Note:

This statement only works with a PRINT or DISP statement.

14.4 CR



Function:

This statement reads out a Carriage Return (0Dhex).

Example: PRINT CR
PRINT% A,CR,X

Note:

This statement only works with a PRINT or DISP statement.

14.5 USING Commands

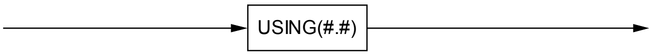
General:

The format statements USING (#.#), USING (Fx) and USING (0) are valid till a new USING statement is executed.
The statemets USING (;#.#), USING (;Fx) und USING (;0) (semicolon in brackets) store the given format. The statement USING(*) recalls the stored format.

Usage:

This recall function can be used if a second USING(...) format is necessary for a temporary output. After this output the original format can be activated with the USING(*) statement. This function often is used in interrupt subroutines.

14.6 USING [U.] (#.#)



Function:

This statement activates the decimal numeral representation i.e. numbers are displayed in the decimal format. The number of # before the decimal point determines the places for the integral part. The number of # behind the decimal point stands for the fractional part. The maximum total number for the #s is 8. At least one must be defined for the integral part.

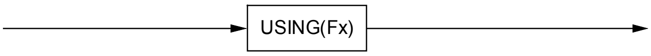
Example: The variable has the value 12.3456

Statement	Output
PRINT USING (##),A	12
(##.#),A	12.3
(##.#####),A	12.34560
(#.##),A	? 12.23456

Note:

USING(#,#) is active till a new USING statement is executed.
If the integral part of the number is greater than the defined number of places before the decimal point the number is displayed with a ? and the free format.
USING(#.#) only works with a PRINT, DISP, or OUTPUT statement.

14.7 USING [U.] (Fx)



Function:

Activate the exponential numeral representation. Value x determines how many places of the mantissa are displayed. If x = 0 the following zeros of the mantissa are suppressed. COMTAC diplays a minimum of three mantissa places, also if x = 1 or 2.

Parameter	Input	Range	Description
x	Number	0 - 8	Number of places of the mantissa

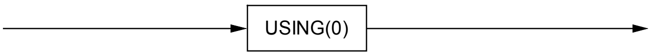
Example: The variable has the value 12345

Statement	Output
PRINT USING (F0),A	1.2345 E+4
(F1)	1.23E+4
(F2)	1.23E+4
(F3)	1.23E+4
(F4)	1.234E+4
(F5)	1.2345E+4
(F6)	1.23450E+4
(F7)	1.234500E+4
(F8)	1.2345000E+4

Note:

USING(#,#) is active till a new USING statement is executed.
USING(Fx) only works with a PRINT, DISP, or OUTPUT statement.

14.8 USING [U.] (0)



Function:

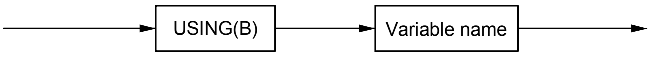
Activate the free numeral representation. The decimal format is used for numbers between ±9999 9999 and ±1. Otherwise the exponential format (USING(F0)) is used.

Example: decimal format: 12.345
exponential format: 1.0 E+8

Note:

After power on this format is active.
USING(0) only works together with a PRINT, DISP, or OUTPUT statement.

14.9 USING [U.] (B)



Function:

Activate the internal binary format. This format is only active for the OUTPUT, PRINT or DISP statement.
The output or display is performed byte per byte, following the way the variable is stored in the COMTAC.
The following table shows the order in which the different data types are displayed.

Datatype	1.Byte	2.Byte	3.Byte	4.Byte	5.Byte	6.Byte
Floating Point	Mantissa					Exponent
	1/2. Ste.	3/4. Ste.	5/6. Ste.	7/8. Ste.	sign	
DSP Fractional	places integral part			placesfractional part		
	LSB		MSB	LSB		MSB
long Integer	LSB			MSB		
Integer	LSB	MSB				

Example:

OUTPUT 1,4;CHR$(88h),"A",USING(B),POS(1)
A binary value, stored in the variable POS(1) is transmitted via RS485 to the COMPAX with the device address 4. The data type is DSP fractional, the value is 1000.

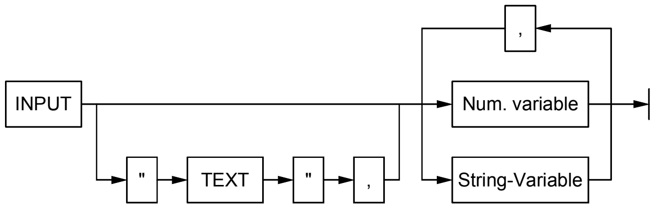
Byte NO.	dec	hex	Description
1	52	34h	Address (ASCII-format)
2	136	88h	CHR$(88h)
3	65	41h	"A"
4	0	00h	POS(1) (place behind dp. LSB)
5	0	00h	POS(1) (place behind dp.)
6	0	00h	POS(1) (place behind dp. MSB)
7	232	e8h	POS(1) (place before dp. LSB)
8	3	03h	POS(1) (place before dp.)
9	0	00h	POS(1) (place before dp. MSB)

Note:

USING(0) only works with a PRINT, DISP, or OUTPUT statement.
USING(B) is only active for one output.

15. Terminal input

15.1 INPUT [I.] (Terminal)



Function:

Input of a value to a variable.

Example: INPUT "SPEED=",S
INPUT $(5)

Note:

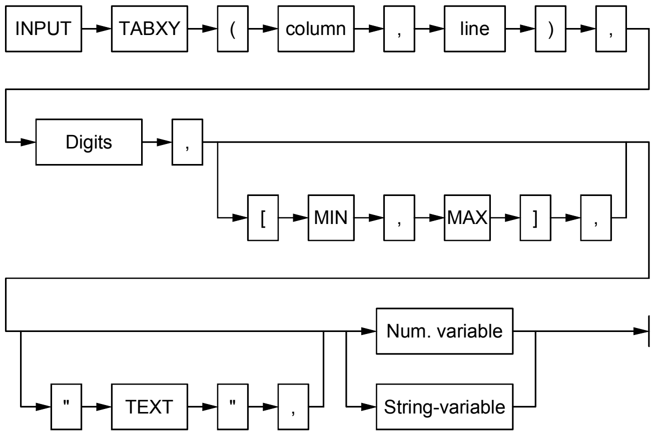
Only a numeric value can be assigned to a numeric variable. Only a string can be assigned to a string variable.
When there is a wrong input a signal tone (BEEP) sounds and the INPUT statement is repeated.

It is possible to input several variables with one input statement. This requires the variables and/or strings of the INPUT statement to be separated by a comma. When executing the INPUT a question mark prompts for the input of the values. The values also need to be separated by a comma. After the input of a string a Carriage Return (↵) has to be entered.

Example: INPUT a,b,$(0),D
Program execution:
?10,20,Hello↵
?30

Note: A string is always terminated with a Carriage Return ↵ (CR).

15.2 INPUT [I.] TABXY [T.]



Function:

This statement adds to the INPUT statement. The TABXY statement sets the cursor. At this cursor position the latest valid value of the variable is displayed left justified. This value can be edited within the limits of the maximum number of digits (Digits) and the minimum (Min) and maximum (Max) values. Following a Carriage Return the value is checked. It is stored in the variable if it keeps within limits. Then the INPUT statement is terminated. Otherwise the latest valid value is displayed and the INPUT is executed again.

Parameter	Input	Range	Description
Column	num. expr.	1 - 80	X-Position
Line	num. expr.	1 - 24	Y-Position
Digits	num. expr.	1 - 80	Number of digits
Min	num. expr.		Minimum value
Min	num. expr.		Maximum value

Example: INPUT TABXY (X,Y),5,[100,M],"Velocity = ",V
INPUT TABXY (10,10),S,[0.4,0.97],"Pieces = ",P

Note:

The maximum number of digits includes the sign and the decimal point.

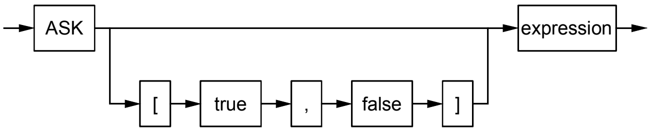
15.3 GET

Function:

Returns the key code of the pressed key to the GET function.
If no key is pressed the return value is 0.

Example: T = GET
10 IF GET = 0 THEN GOTO 10

15.4 ASK



Function:

Polls the terminal keyboard.
´Expression´ is displayed on the screen. Then the keyboard is polled until the key y/Y or n/N is pressed. y/Y returns a TRUE (=65535), n/N a FALSE (=0) result. Any other key causes a signal tone (BEEP) and is ignored.
If other keys instead of nN, yY want to be used the code of these keys has to be added to the statement in square brackets.

Parameter	Input	Range	Description
true	num. expr.	0 - 255	ASCII code of the key
false	num. expr.	0 - 255	ASCII code of the key

Example:

```
IF ASK TABXY(10,5), Do you want to change the value?(Y/N)" THEN....
X = ASK [49,50]"Choose axis 1 or 2"
```

(49/50 is the ASCII-Code of 1 / 2)

Note:

The expression following the ASK is handled like a PRINT statement.

Caution

The ASK statement can be interrupted and will be terminated.
In this case the result is 1 instead of TRUE or FALSE.
The result can be checked in the program line which follows the ASK statement.

Example:

```
100    A = ASK [49,50]"Coose axis 1 or 2"
110    If A=1 THEN 100
```

15.5 Use of the Function Keys

COMTAC provides up to 18 function keys.
They are handled according to the following rules:

- To each function key used in the program a separate sequence of statements (GOTO) or a subroutine (GOSUB) has to be assigned.
  Example: ONKEY 1 GOTO 500
  ONKEY 12 GOSUB 1000
- This program part begins with the specified program line.
- The statement ENABLE ONKEY x enables the specified function key, DISABLE ONKEY x disables it. These statements can be used anywhere in the program more than once.
- The function key generates an interrupt. This interrupt will be carried out when the execution of the actual program line has finished. The program then branches to the defined interrupt sequence.

Example:

```
50 ONKEY 1 GOTO 500:ENABLE ONKEY 1
60 ONKEY 12 GOSUB 1000:ENABLE ONKEY 12
.
.
```

```
500    DISABLE ONKEY.1
510
.
.
.
580    ENABLE ONKEY 1:GOTO xxx
```

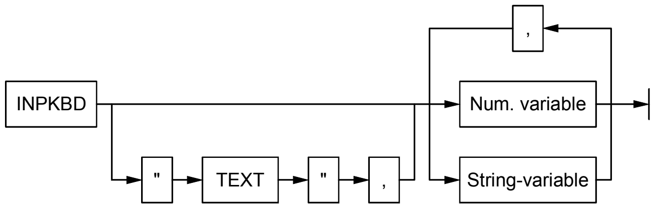
Program part for function key F1

```
1000   DISABLE ONKEY 12
1010
.
.
.
1090   ENABLE ONKEY 12:RETURN xxx
```

Program part for function key F12

16. COMTAC Keyboard - Input

16.1 INPKBD



Function:

Input of a value to a variable.

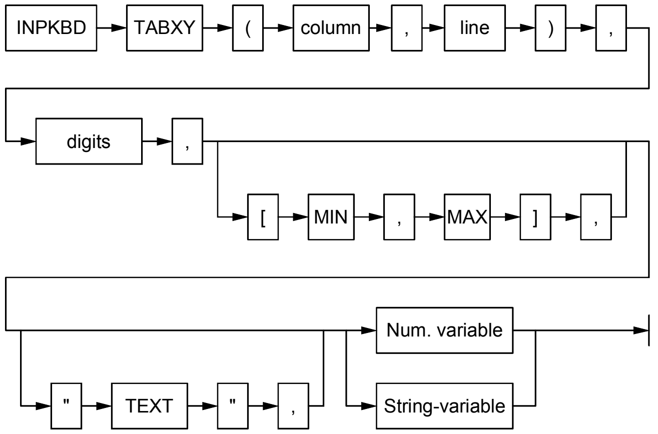
Example: INPKBD "Position =",POS
INPKBD $(32)

Note:

With the COMTAC's Keyboard it is only possible to input numbers. Each input has to be terminated with the ENTER key "Ent"

Example: INPUT a
Program run
?10 key "Ent"

16.2 INPKBD TABXY [T.]



Function:

This statement adds to the INPUT statement.
The TABXY statement sets the cursor. At this cursor position the latest valid value of the variable is displayed left justified. This value can be edited within the limits of the maximum number of digits (Digits) and the minimum (Min) and maximum (Max) value. Following ENTER the value is checked and is stored in the variable if it keeps within the limits. Then the INPUT statement is terminated. Otherwise the latest valid value is displayed and the INPUT is executed again.

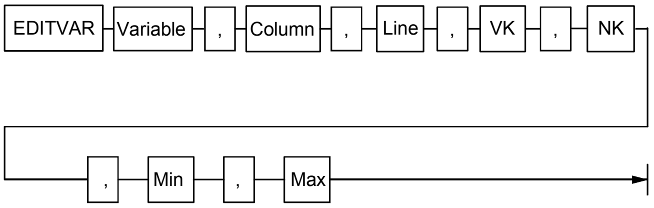
Parameter	Input	Range	Description
Column	num. expr.	1 - 40	X-position
Line	num. expr.	1 - 4	Y-position
Digits	num. expr.	1 - 40	Number of digits
MIN	num. expr.		Minimum value
MAX	num. expr.		Maximum value

Example: INPKBD TABXY 1 (10,3),[0,5], "OFFSET=",A

Note:

The maximum number of digits includes the sign and the decimal point.

16.3 EDITVAR



Function:

COMTAC Keyboard input without program stop . In contrast to the INPUT statements listed above, this input function doesn´t stop the BASIC program run during keyboard input.
The latest valid value of the specified variable is displayed left justified at the given position (column, line).
This value can be edited within the limits of the minimum (Min) and maximum (Max) value and the maximum places defined by the number of digits before and after the decimal point (BD, AD). Following ENTER the value is checked and is stored in the variable if it keeps within the limits. Otherwise the latest valid value is displayed.
This function can be aborted with ESC without changes.
This function keeps running till the command:

EDITVAR STOP *column,line*: This statement terminates the input procedure for the specified variable displayed at *column,line.*

Parameter	Input	Range	Description
Column	num. expr.	1 - 40	X-position
Line	num. expr.	1 - 4	Y-position
(BD)		0 - 8	places integral part
(AD)		0 - 8	places fractional part
	Condition: BD + AD ≤ 8		
Min	num. expr.		minimum value
Max	num. expr.		maximum value

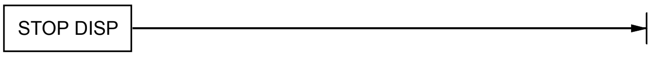
Example: EDITVAR A,20,1,3,4,-100,100
EDITVAR B,20,2,2,0,0,10

Two variables are displayed. Variable A at column 20 in line 1, variable B at column 20 in line 2.

Note:

- ◆A maximum of 9 input values can be activated and displayed on the COMTAC display in order to create input masks. However, this maximum number includes the cyclic display of BASIC variables (DISPVAR statement) and COMPAX info (DISPCPXZ statement).
- ◆If there is more than one variable the cursor up / cursor down keys of the COMTAC display allow you to toggle between the different variables.

16.3.1 STOP DISP



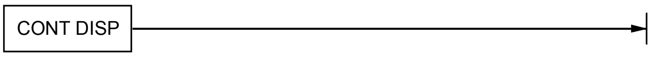
The contents of the complete COMTAC display is stored. All display functions (EDITVAR, DISPVAR, DISPCPXZ) are stopped. So the display can be used to display some messages, e.g. errors or actual values.

The statement CONT DISP continues the previously stopped display.

Note

When an EDITVAR, DISPVAR or DISPCPXZ statement is executed during STOP DISP all previous activated displays are cleared.
These new statements will be activated after CONT DISP.

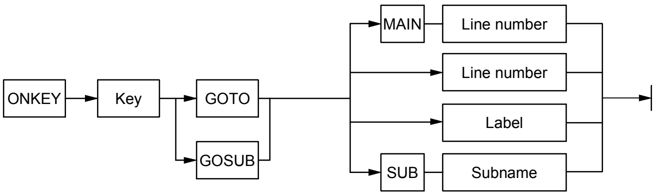
16.3.2 CONT DISP



The statement CONT DISP continues the previously stopped display.

Before "CONT DISP" a "STOP DISP" must be carried out.

16.4 ONKEY [OK.]



Function:

Function key interrupt. This statement determines a target address where the program branches to when the specified function key interrupt is actioned.

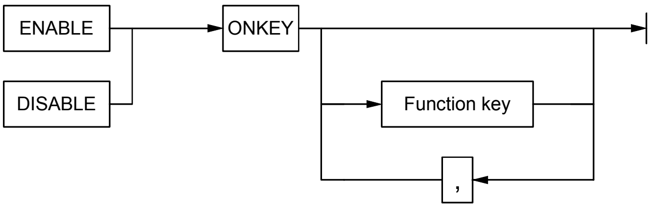
Function keys are the keys F1 to F16 and the START and STOP key

- ◆START-key = function key 17
- ◆STOP-key = function key 18.

Parameter	Input	Range	Description
Key	num. expr.	1 - 18	function key number
Line no.	number	0-65535	target line number

Example: ONKEY 3 GOTO 25500
ONKEY 17 GOSUB 8880

16.5 ENABLE [EN.]/DISABLE [DI.] ONKEY [OK.]



Function:

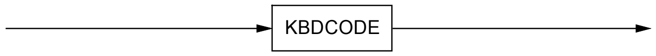
Enable or disable a function key interrupt

Parameter	Input	Range	Description
Key	num. expr.	1 - 18	function key number

Note:

The statement ENABLE / DISABLE without a function key number relates to all previous defined function key interrupts (ONKEY statement).

16.6 KBDCODE

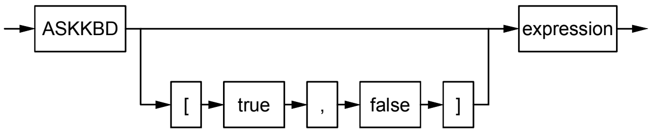


Function:

Sample the COMATC keyboard code.
Returns the key code of the pressed key to the KBDCODE function.
If no key is pressed the return value is 0.
Example: DISP KBDCODE
10 IF KBDCODE = 0 THEN GOTO 10
Keycodes,assigned to the variable KBDCODE.

Taste	Dezimal-Wert	Hex-Wert
Function key 1	176dec	0b0hex
Function key 2	177dec	0b1hex
Function key 3	178dec	0b2hex
Function key 4	179dec	0b3hex
Function key 5	180dec	0b4hex
Function key 6	181dec	0b5hex
Function key 7	182dec	0b6hex
Function key 8	183dec	0b7hex
Function key 9	184dec	0b8hex
Function key 10	185dec	0b9hex
Function key 11	186dec	0bahex
Function key 12	187dec	0bbhex
Function key 13	188dec	0bchex
Function key 14	189dec	0bdhex
Function key 15	190dec	0behex
Function key 16	191dec	0bfhex
Function key START	192dec	0c0hex
Function key STOP	193dec	0c1hex
ESC (Escape)	026dec	01a hex
cursor left	008dec	008 hex
cursor right	012dec	00c hex
cursor up	011dec	00b hex
cursor down	022dec	016 hex
ENT (enter)	013dec	00d hex
INS (insert)	002dec	002 hex
DEL (delete)	127dec	07f hex
Key 0	048 dec	030 hex
Key 1	049 dec	031 hex
Key 2	050 dec	032 hex
Key 3	051 dec	033 hex
Key 4	052 dec	034 hex
Key 5	053 dec	035 hex
Key 6	054 dec	036 hex
Key 7	055 dec	037 hex
Key 8	056 dec	038 hex
Key 9	057 dec	039 hex
Key decimal point	046 dec	02e hex
Key + / -	045 dec	02d hex

16.7 ASKKBD



Function:

Polls the COMTAC keyboard.
´Expression´ is displayed on the screen. The keyboard is polled until the specified key for TRUE or FALSE is pressed. A TRUE returns the value 65535, a FALSE returns 0. Any other key is ignored.

Parameter	Input	Range	Description
True	num. expr.	0 - 255	ACII code of the key
False	num. expr.	0 - 255	ASCII code of the key

Example:
IF ASKKBD ["0","1"] "Is this input correct?(0/1)" THEN...

Note:

The expression following the ASK is handled like a DISP statement.

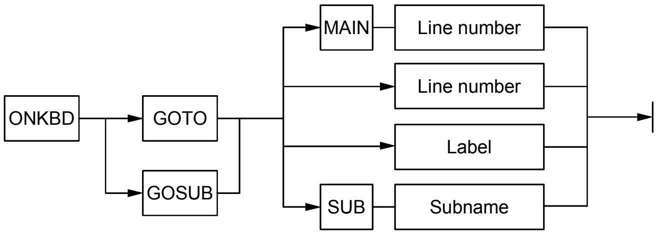
Caution

The ASKKBD statement can be interrupted which will cause it to be terminated.
In this case the result is 1 instead of TRUE or FALSE.
The result can be checked with a statement in the program line which follows the ASKKBD statement.

Example:

```
100   A = ASKKBD [49,50]Choose axis 1 or 2"
110   If A=1 Then 100
```

16.8 ONKBD



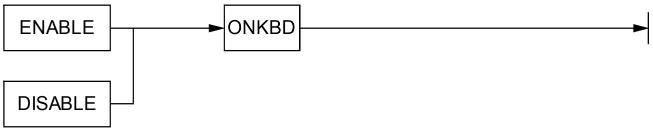
Function:

This statement defines a program branch. When a key of the COMTAC keyboard is pressed or released this branch is executed provided the interrupt has been enabled.
The system variable KBDCODE returns the value of the pressed key or 0 when the key had been released.

Parameter	Input	Range	Description
Line no.	number	0-65535	Target line number

Example: ONKBD GOTO 1000

16.9 ENABLE [EN.]/DISABLE [DI.] ONKBD



Function:

Enable / disable the COMTAC keyboard interrupt.

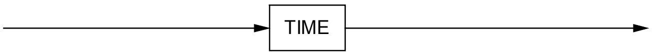
Example: ENABLE ONKBD
DISABLE ONKBD

Note:

Releasing a key also produces an interrupt. The return value for KEYCODE is 0 in this case.

17. Real Time Clock and Timer

17.1 TIME



Function:

Returns the time of the real time clock (RTC) in the format HH:MM:SS

Example: PRINT TIME
DISP TIME

Note:

The TIME statement can only be used in combination with a PRINT-, DISP- or OUTPUT-statement.

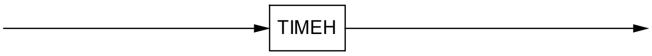
Exceptions:

- Copy the TIME to a string, e.g. $(4) = TIME.
- Set the RTC: TIME = H,M [S]

Parameter	Input	Range	Description
H	num. expr.	0 - 23	Hour
M	"	0 - 59	Minute
S	"	0 - 59	Second

Seconds are set to 0 if this value is left out.

17.2 TIMEH



Function:

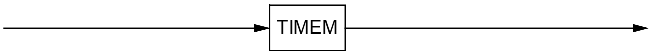
System variable to set or read the hours of the RTC.

Example: TIMEH = 10
IF TIMEH = 12 THEN ....

Note:

This variable has to be in the range of 0 to 23.

17.3 TIMEM



Function:

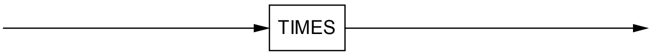
System variable to set or read the minutes of the RTC.

Example: TIMEM = 33
IF TIMEM = 0 THEN ....

Note:

This variable has to be in the range of 0 to 59.

17.4 TIMES



Function:

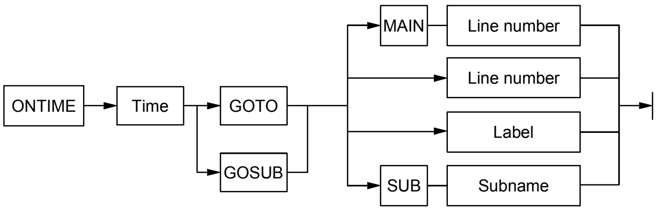
System variable to set or read the seconds of the RTC.

Example: TIMES = 0
IF TIMES = 15 THEN ....

Note:

This variable has to be in the range of 0 to 59.

17.5 ONTIME



Function:

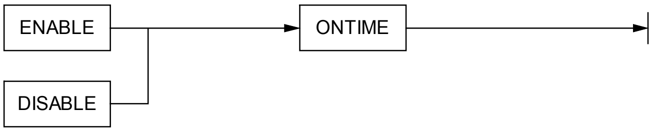
This statement defines a program branch which is executed when the time of the RTC equals the preset time value (See ENABLE/DISABLE ONTIME)

Example: ONTIME 12,30,0 GOTO
ONTIME H,M GOSUB 3300

Note:

The time value must be entered in the format
Hours, minutes [,seconds] (see TIME).

17.6 ENABLE [EN.]/DISABLE [DI.] ONTIME



Function:

Enable/disable the RTC interrupt.

Example: ENABLE ONTIME / DISABLE ONTIME

17.7 DATE

Function:

This statement returns the actual date in the format *weekday, MM-DD-XXAA* or *weekday, DD.MM.XXAA.*
XX = 19 for 1980 - 1999
XX = 20 for 2000 - 2079
Parameter 13 defines the format.

Example: PRINT DATE
OUTPUT 32; $(0), DATE

Note:

The DATE statement only can be used in combination with a PRINT-, DISP- or OUTPUT-statement.

Exceptions:

- Copy the DATE to a string, e.g. $(7) = DATE.
- Set the date: DATE = W, M, D, A

Parameter	Input	Range	Description
W	num. expr.	1-7	Weekday
M	"	1-12	Month
D	"	1-31	Day
A	"	0-99	Year

17.8 DATEW

Function:

System variable to set or read the weekday of the actual date.

Example: DATEW = 1
IF DATEW = 5 THEN ....

Note:

The value assigned to this variable has to be in the range of 0 to 7.

Sunday= 1 Monday= 2 Tuesday= 3 Wednesday= 4
Thursday= 5 Friday= 6 Saturday= 7

17.9 DATED

Function:

System variable to set or read the day of the actual date

Example: DATED = 25
IF DATED = 4 THEN ....

Note:

The value assigned to this variable has to be in the range of 0 to 31.

17.10 DATEM

Function:

System variable to set or read the month of the actual date.

Example: DATEM = 7
IF DATEM = 12 THEN ....

Note:

The value assigned to this variable has to be in the range of 0 to 12.

17.11 DATEY

Function:

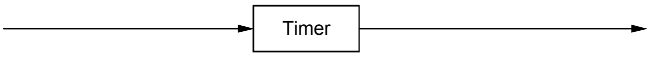
System variable to set or read the year of the actual date.

Example: DATEY = 97
IF DATEY = 99 THEN ....

Note:

The value assigned to this variable has to be in the range of 0 to 99.

17.12 TIMER



Function:

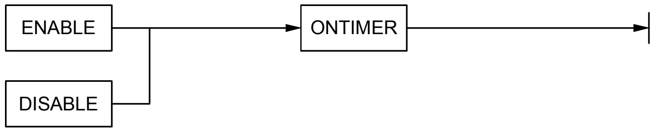
System variable to set/read the timer of the BASIC system.

Example: PRINT TIMER
X = TIMER

Note:

This variable is incremented every 5 milliseconds (if enabled) until it has the value of 65535995sec. Then it restarts with a value of 0.

17.13 ENABLE [EN.]/DISABLE [DI.] ONTIMER [OC.]

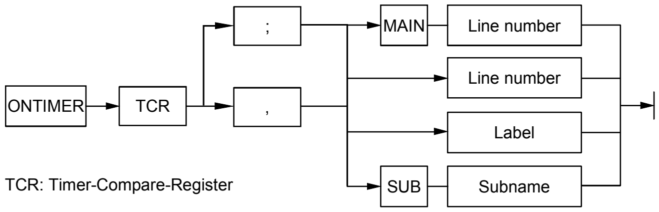


Function:

Enable/disable the system timer. Enable: the timer is incremented. Disable: the timer stops.

Example: ENABLE ONTIMER
DISABLE ONTIMER

17.14 ONTIMER [OC.]



Function:

This statement defines a subroutine call. This call is executed when the TIMER value is equal to or greater than the preset value.
To execute this branch the timer has to be enabled. (see ENABLE/DISABLE ONTIMER).

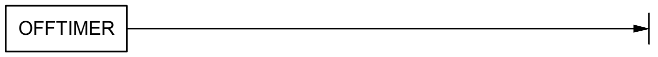
Example: ONTIMER 1 , 200
ONTIMER TIMER+2,1000

Note:

The TIMER value is incremented every 5 milliseconds. The interrupt is executed when the value of the timer is equal to or greater than the value of the timer compare register (TCR). The resolution for the compare function is one second. The interrupt event is executed once. An auto reload function makes it possible to maintain the interrupt generation e.g. for time slots, by using a semicolon instead of a comma. A GOTO branch is not possible with this function.

ONTIMER can be cleared with the statement OFFTIMER

17.15 OFFTIMER



Function:

Disables the TIMER interrupt. The timer keeps on running if enabled.

18. Strings

18.1 Character Strings

Strings allow non numerical information to be processed. COMTAC BASIC uses one dimensional strings. The syntax to define a string is $(index). Index can be a number, a variable or a numeric expression. The maximum value for an index is 255.
Examples: $(7) ; $(A) : $ (A-8/4)
A string could be an empty string - string without characters: $(1)="".
The string length for the definition of a string can be 1 to 255. The actual length can be 0 to 255.
The actual length of a string is the number of characters stored in a string variable.

18.2 Storing strings

The default length of a string is 16 characters. Other lengths have to be defined, e.g.:

DIM$ (5) [100]

This definition reserves memory for a string with 100 characters.

	1	2	3	4	5	6	7	8	9	...	99	100
$(5)												

18.2.1 Linked Strings

Single strings can be linked by assigning these strings to another string, separated by a comma:

```
    10 $(0) = "COMPUTER"
    20 $(1) = "GAMES"
    30 $(2) = $(0), $(1)
    40 PRINT $(0)
    50 PRINT $(1)
    60 PRINT $(2)
    70 END
```

Program run:

```
COMPUTER
GAMES
COMPUTERGAMES
```

The statement in line 30 links the strings $(0) and $(1) and assigns it to string $(2).

18.3 Parts of Strings

A part of a string can be the whole string or only some characters of the string. A part of a string is specified by one, two or three indices enclosed in square brackets.

Example:

$(0)[5] defines a string which starts with the fifth character of string $(0) till the end of this string

$(15)[3,6] defines a string which starts with the third and ends with the sixth character of string $(15).

$(7)[9,15,13] defines a string which starts with the ninth and ends with the fifteenth character or with the character which has the value of 13.

$(7)[9,,13] defines a string which starts with the ninth and ends with the character which has the value of 13.

The first index defines the first character in a string, the second the last. The third index defines the value of the character with which the string ends. All indices are optional.
Definition:
$(x)[START,END,ENDCHARACTER] or
$(x)[START,END] or
$(x)[START,,ENDCHARACTER]
$(x)[START]

A part of a string can be assigned to another part of a string, e.g.
$(0) [8,15] = $(93) [5,9]

A "**Bad Argument**" error occurs when:

- The START or END index is less than 1.
- The START index on the left side of the “equals“ sign is greater than the actual length+1 of this string
- The START index on the right side of the “equals“ sign is greater than the actual length of this string.
- The START or END index is greater than the string length.

18.4 Predefined Strings

COMTAC BASIC has the following predefined strings:

$(#0)	**RS232/1- buffer**
$(#1)	**RS485/1- buffer**
$(#2)	**RS232/2- buffer**
$(#3)	**RS485/2- buffer**
$(#4)	**RS232/3- buffer**
UMEM$	**Access to the Data Text Array**

These strings are unique.

$(#Interface number)

The number of these strings corresponds to the channel number of the interfaces of COMTAC. Each interface uses one of these strings in order to store the characters received via this interface. The storing is executed by the ENTER statement. The string length can be up to 255 characters.

Example: ENTER 0,2
DISP $(#0)

UMEM$

This string is used to gain access to strings in the Data Text Array. A complete line (line orientated) of the Data Text Array or a specified number of characters (absolute addressed) can be read or written.

1. Line orientated

Any line in the Data Text Array is a string with the term UMEM$(line *no*.). The length of these strings is set by the definition of the Data Text Array.
These strings can be handled like all other strings.

2. Absolute addressed

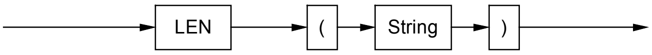
The Data Text Array has a length of 4096 characters (byte). Within this range a string can be defined by the statement UMEM$(address,length). Address is 0 to 4095, length is 0 to 255. These strings must not be parts of other strings.
For a description see page 84.

18.5 Comparing Strings

To compare strings the compare operations “=“ and “<>“ are possible. They can be used together with an IF-statement or a DO-WHILE/DO-UNTIL-loop.

Example: 1.) IF $(0) = "COMTAC 8000" THEN ...
2.) DO
...
UNTIL $(10) <> UMEM $(7)[3,8]
3.) DO
...
WHILE $(#1)[5,5] = "1"

18.6 LEN($(x))



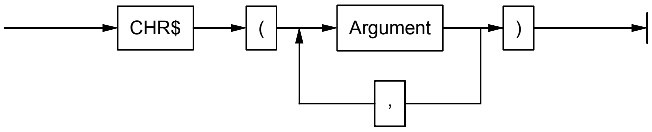
Function:

Returns the actual length of a string.

Parameter	Input	Range	Description
String	Character string		string index

Example 1: A = LEN($(7))
Example 2: PRINT LEN($(#2)[5,,13])

18.7 CHR$



Function:

This statement changes a number to an ASCII character.

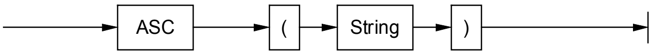
Parameter	Input	Range	Description
Argument	num. expr.	0 - 65535	value to be changed

Example 1: $(B)[4,4] = CHR$(97)
Example 2: OUTPUT 1,3; CHR$(13)

Note:

Only the low byte of the argument is changed. The high byte is ignored.

18.8 ASC



Function:

Changes the first character of an ASCII string to a number.

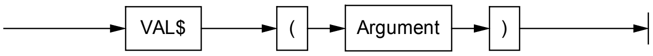
Parameter	Input	Range	Description
String	character string		character to be changed

Example 1: V = ASC($(#2))
Example 2: IF ASC($(3)[4]) <> 17 THEN...

Note:

The string must not be an empty string.

18.9 VAL$



Function:

Changes the specified number to a string.

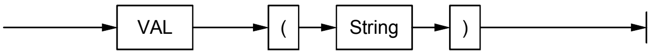
Parameter	Input	Range	Description
Argument	num. expr.		value to be changed

Example 1: PRINT VAL$(A)
Example 2: UMEM$(8) = VAL$(X)

Note:

The string is in a PRINT format.
Exception: In the case of a positive number the first character must not be blank.
The format can be changed by the PRINT USING statement.

18.10 VAL



Function:

Changes a string to a number.

Parameter	Input	Range	Description
String	character string		string to be changed

Example: Q = VAL($(13))
IF VAL($(B)[8]) = 137 THEN ...

Note:

The first character of this string, if it´s not a blank, has to be a number, +, -, or the decimal point.
All numerical characters following the first character till the first non-numerical character are transformed.

19. Digital In/Outputs

COMTAC 2000 has 16 digtal inputs and 16 outputs, whilst the COMTAC 3000 has 32 digtal inputs and 32 outputs. A multi axis system using COMPAX axis controllers and the field bus connection to COMTAC can be formed (see page 89). In such a system, the digital inputs and outputs of 14 COMPAX can be used with the commands: IN(x), BCDIN(x), BINOUT (x) = y, BCDOUT(x) = 0, CLROUT x, SETOUT x und REQOUT x,y,t.

19.1 Field Bus I/O Organisation

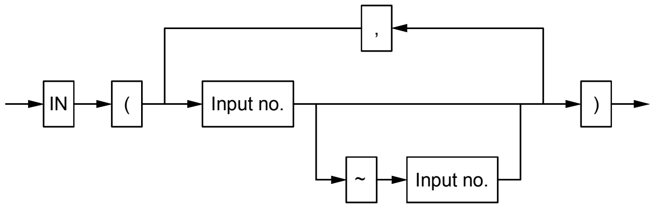
The COMTAC and COMPAX inputs and outputs are addressed in a linear manner.

A maximum of 256 inputs and 256 outputs can be addressed. The numerical order is as follows:

I/O *no.*	Device
01 ... 16	COMTAC 2000/3000 I/O's
17 ... 32	COMTAC 3000 I/O's
33 ... 48	COMPAX field bus address *no*.1
49 ... 64	COMPAX field bus address *no*.2
65 ... 80	COMPAX field bus address *no*.3
81 ... 96	COMPAX field bus address *no*.4
97 ... 112	COMPAX field bus address *no*.5
113 ... 128	COMPAX field bus address *no*.6
129 ... 144	COMPAX field bus address *no*.7
145 ... 160	COMPAX field bus address *no*.8
161 ... 176	COMPAX field bus address *no*.9
177 ... 192	COMPAX field bus address *no*.10
193 ... 208	COMPAX field bus address *no*.11
209 ... 224	COMPAX field bus address *no*.12
225 ... 240	COMPAX field bus address *no*.13
241 ... 256	COMPAX field bus address *no*.14

This mode only can be used with the field bus protocol!

19.2 IN (digital Input)



Function:

Read the state of a digital input or a group of inputs.

The value of this read operation is a decimal value. If only one input is read the value is 0 or 1. When reading several inputs the significance ot these inputs is determined by the order of the input numbers given by the IN statement.

The first input number is the most significant bit, the last input number is the least significant bit.

The number of inputs is limited to 16.

A range of inputs is separated by the ~ .

Single input numbers are separated by a comma.

Parameter	Input	Range	Description
Input.No.	num. expr.	1 - 16 (32)	Number of the input

The following examples help to understand the input function:

Example 1: PRINT IN(8~1)

Inputs 1 to 8 are read.

With inputs 3, 5, and 8 set to logic 1.

The inputs get the following bit order:

Binary places	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Significance	128	64	32	16	8	4	2	1
Input no.	8	7	6	5	4	3	2	1
state of the inputs	1	0	0	1	0	1	0	0
displayed decimal value	148 (128+16+4)							

Example 2: PRINT IN(1~3,16,10,13)

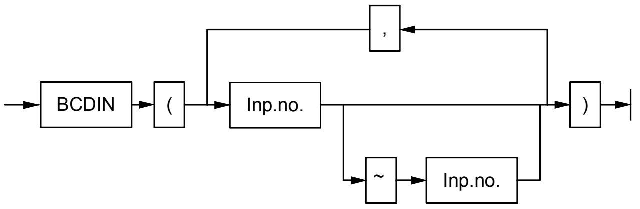
Inputs 1 to 3,10,16 and 13 are read.

With inputs 1 and 10 set to logic 1.

The inputs get the following bit order:

Binary places	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Significance	128	64	32	16	8	4	2	1
Input no.			1	2	3	16	10	13
state of the inputs			1	0	0	0	1	0
displayed decimal value	34 (32+2)							

19.3 BCDIN



Function:

Returns the state of digital inputs in binary coded decimal. One digit consists of four bits.

- The read value is changed into the BCD format.
- Four input bits form one digit.
- The significance of the bits for one digit and the significance of the digits is determined by the order of the input numbers given by the BCDIN statement.
- The input number in the BCDIN statement is the least significant bit of the least siginficant digit. The input numbers following are assigned in ascending order.
- The number of inputs is limited to 16 (4 digits).
- A range of inputs is separated by the ~ .
- Single input numbers are separated by a comma.

Parameter	Input	Range	Description
Input No.	num. expr.	1 - 16 (32)	Number of the input

The following examples help to understand the input function:

Example 1: A=BCDIN(8~1)

Inputs 1 to 8 are read. With inputs 1, 3, 5, 6 set to logic 1. The return value is 35.

Digits	10^1					10^0			
Binary places	2^3	2^2	2^1	2^0		2^3	2^2	2^1	2^0
Significance	8	4	2	1		8	4	2	1
Input No.	8	7	6	5		4	3	2	1
Value of 35 in the BCD format	0	0	1	1		0	1	0	1
				3					5

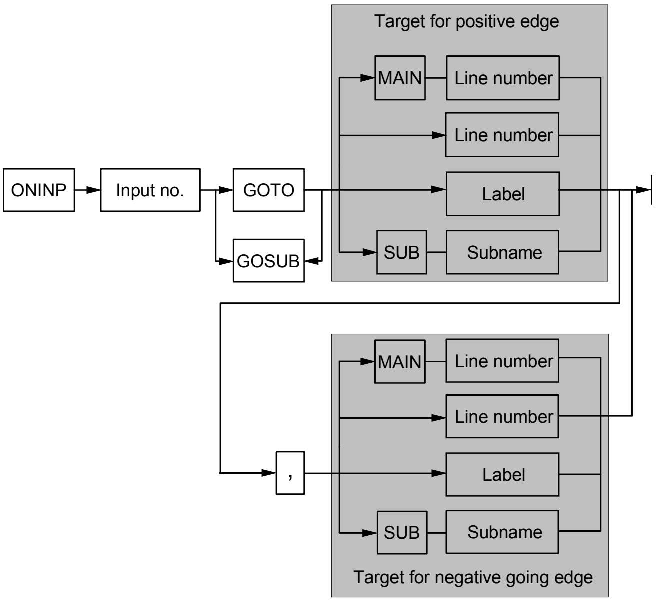
Example 2: A=BCDIN(1~3,16,10,22)

Inputs 1,2,3,16,10 and 22 are read. With inputs 22, 3 and 2 set. The return value is 19.

The inputs get the following bit order:

Digits	10^1					10^0			
Binary places	2^3	2^2	2^1	2^0		2^3	2^2	2^1	2^0
Significance	8	4	2	1		8	4	2	1
Input No.			1	2		3	16	10	22
Value of 19 in the BCD format	0	0	0	1		1	0	0	1
				1					9

19.4 ONINP [OP.]



Function:

Program branch (interrupt) to the specified line number or subroutine if the corresponding input is defined.

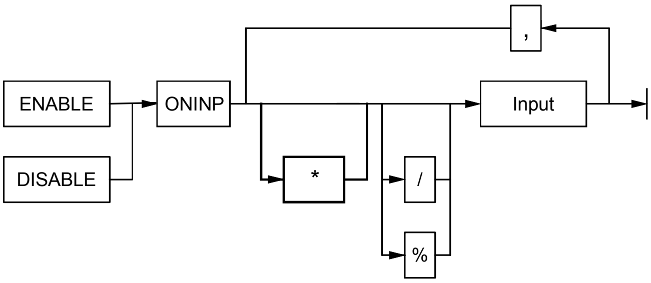
Parameter	Input	Range	Description
Input	num. expr.	1 - 16	Input No.
Line No. positive (or negative)[2] going edge	number	0-65535	Target line number for the positive (or negative) going edge
Line No. negative going edge	number	0-65535	Target line number for the negative going edge

Example: ONINP 2 GOTO 300
ONINP X GOSUB 10000

Note:

The inputs are transition sensitive.

19.5 ENABLE [EN.]/DISABLE [DI.] ONINP [OP.]



Function:

Disable / enable the ONINP interrupts and define the type of sensitivity.

Parameter	Input	Range	Description
Input	num. expr.	1 - 16	Number of the input

Example 1: ENABLE ONINP/8
Example 2: DISABLE ONINP4

Note:

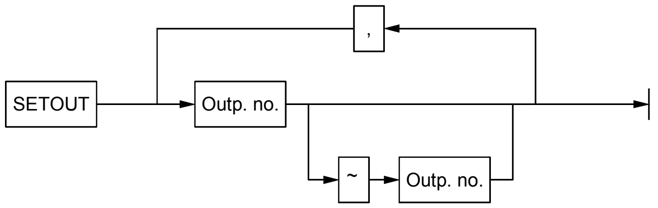
The inputs can be defined to be level or transition sensitive.

Sensitivity

Sign	Meaning
none	Each rising edge generates an interrupt
/	Each falling edge generates an interrupt
%	Both edges generate an interrupt.
*	If the input has the specified level at the time the statement ENABLE ONINP is excecuted, an interrupt is generated.

[2] If only one line number is given an both edges are enabled it´s the target line for both .

19.6 SETOUT



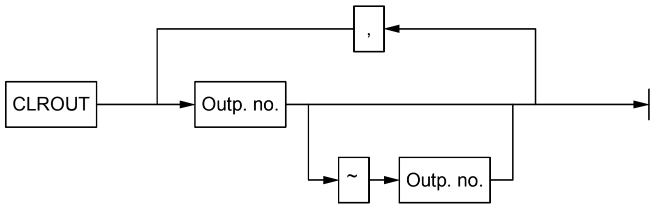
Function:

Set one or several digital outputs.
A range of outputs is separated by the ~ .
Single output numbers are separated by a comma.

Parameter	Input	Range	Description
Output No.	num. expr.	1 - 16 (32)	Number of the digital output

Example: SETOUT 3,5,18~21
SETOUT X~Y,Z

19.7 CLROUT



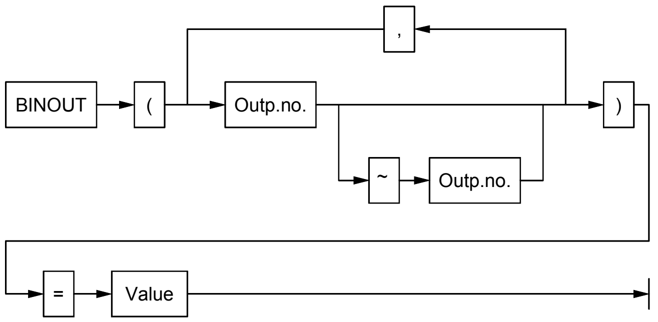
Function:

Clear one or several digital outputs.
A range of outputs is separated by the ~ .
Single output numbers are separated by a comma.

Parameter	Input	Range	Description
Output No.	num. expr.	1 - 16 (32)	Number of the output

Example: CLROUT 10,4~8,A,B
CLROUT V,8~4

19.8 BINOUT [B.]



Function:

Assign a binary value to the digital outputs.

- The decimal value to be read out is transformed into a binary format.
- This value is read out to the specified outputs.
- The last named output of the output statement is assigned the siginficance of 2^0 the next one 2^1 and so on.
- The siginficance of the ouputs is determined by the order given to the output statement.
- The number of outputs is limited to 16.
- A range of outputs is separated by the ~ .
- Single output numbers are separated by a comma.

Parameter	Input	Range	Description
Output No.	num. expr.	1 - 16 (32)	Nunmber of the output
Value	num. expr.	0 - 65535	assigned value

The following examples help to understand the output function:

Example 1:

BINOUT(8~1)=35
The value of 35 is assigned to the outputs 1 to 8.
The outputs are set as follows:

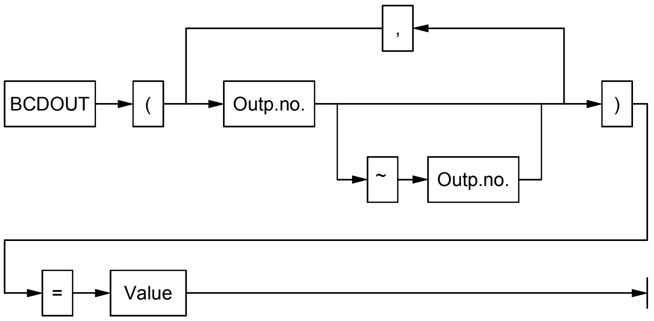
Binary places	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Significance	128	64	32	16	8	4	2	1
Output No.	8	7	6	5	4	3	2	1
Value of 35 in the binary format	0	0	1	0	0	0	1	1

Example 2:

BINOUT(1~3,16,10,13)=99
The value of 49 is assigned to the outputs 1, 2, 3, 16, 10 and 13.
The outputs are set as follows:

Binary places	**2^7**	**2^6**	**2^5**	**2^4**	**2^3**	**2^2**	**2^1**	**2^0**
Significance	128	64	32	16	8	4	2	1
Output No.			1	2	3	16	10	13
Value of 49 in the binary format	0	0	1	1	0	0	0	1

19.9 BCDOUT



Function:

Read out a value in binary coded decimal digits (BCD format). One digit consists of four bits.

- The decimal value to be read out is transformed into BCD format.
- This value is read out to the specified outputs.
- Four bits form one digit.
- The significance of the bits for one digit and the significance of the digits are determined by the order in which the input numbers appear in the BCDIN statement.
- The output number last named in the BCDOUT statement is the least significant bit of the least siginficant digit. The output numbers following are assigned in ascending order.
- The number of outputs is limited to 16 (4 digits).
- A range of outputs is separated by the ~ .
- Single output numbers are separated by a comma.

Parameter	Input	Range	Description
Output No.	num. expr.	1 - 16 (32)	Number of the output
Value	num. expr.	0 - 9999	assigned value

The following examples help to understand the output function:

Example 1: BCDOUT(8~1)=35

The value of 35 is assigned to the outputs 1 to 8:

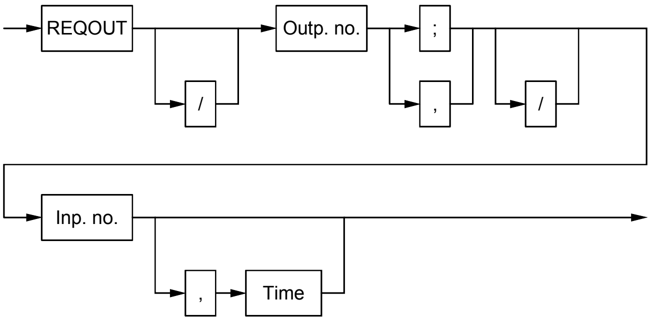
Digits	10^1					10^0			
Binary places	2^3	2^2	2^1	2^0		2^3	2^2	2^1	2^0
Siginficance	8	4	2	1		8	4	2	1
Output No.	8	7	6	5		4	3	2	1
Value of 35 in the BCD format	0	0	1	1		0	1	0	1

Example 2: BCDOUT(1~3,16,10,22)=19

The value of 19 is assigned to the outputs 1,2,3,16,10 and 22:

Digits	10^1					10^0			
Binary places	2^3	2^2	2^1	2^0		2^3	2^2	2^1	2^0
Significance	8	4	2	1		8	4	2	1
Output No.			1	2		3	16	10	22
Value of 19 in the BCD format	0	0	0	1		1	0	0	1

19.10 REQOUT



Function:

Set an output to the specified logic level and poll an input for a period of *Time*.

If the input gets the specified logic level before the *Time* has run out the return value of REQOUT is logic 1 (=65535) and the output is reset. Otherwise the return value is logic 0 (=0).

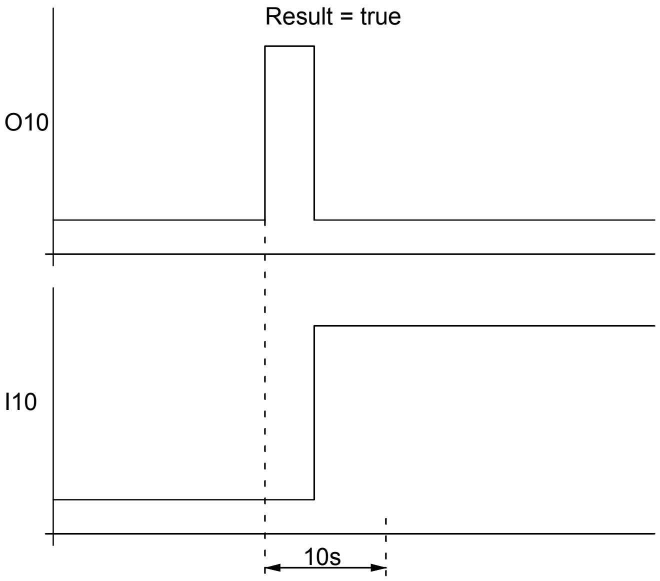
In this case the output is reset only if the character ";" is given after the output number.

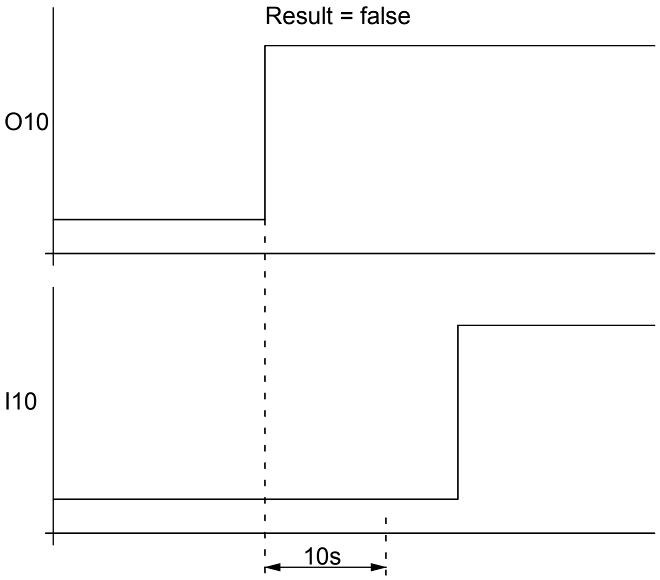
Parameter	Input	Range	Description
Output No.	num. expr.	1 - 16 (32)	Number of the output
Input No.	num. expr.	1 - 16 (32)	Number of the input
Time	num. expr.	1 - 255	Time (1 = 100msec)

Example 1: IF REQOUT 2;7,30 THEN ...

Example 2: W = REQOUT A,/E

Example 3: W = REQOUT 10,10,100

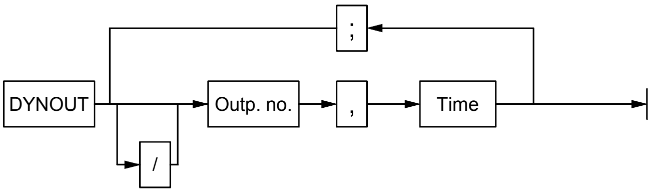




Note:

The character "/" defines the logic level of 0.
If the value for *Time* is left out the default value in paramter 8 is taken.

19.11 DYNOUT



Function:

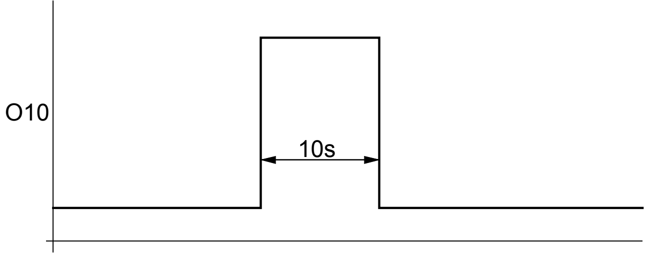
Set an output for a period of *Time* to the specified logic level. After *Time* is run out the output is reset.

Parameter	Input	Range	Description
Output No.	num. expr.	1 - 16	Number of the digital output.
Time	num. expr.	1 - 255	1 = 100 ms

Example 1: DYNOUT 7,50

Example 2: DYNOUT A,T1;8,T2;2,100

Example 3: DYNOUT 10,100



Note:

This time procedure runs in-step with the BASIC program.

19.12 KEYSWITCH

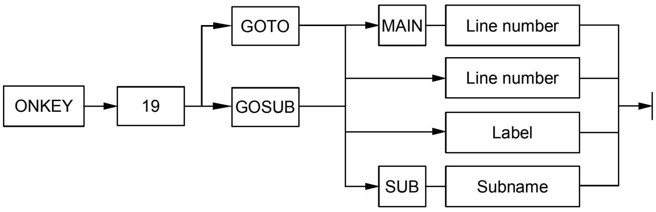
Function:

Reads in the state of the key switch input (connector X7, S1).

If S1 is closed the return value is TRUE(=65535). Otherwise the return value is FALSE(=0).

Example: IF KEYSWITCH THEN GOTO 100

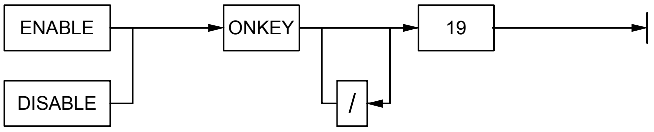
19.13 ONKEY 19 (Keyswitch)



Key switch interrupt

Example: ENABLE ONKEY 19

19.14 ENABLE [EN.]/DISABLE [DI.] ONKEY 19 (Key switch)



Enable the key switch interrupt.

ONKEY 19 signals a key switch operation. The interrupt can be generated when closing and/or opening the switch.

- ◆Interrupt when closing: ENABLE ONKEY 19
- ◆Interrupt when opening: ENABLE ONKEY /19

19.15 EMY_STOP

Function:

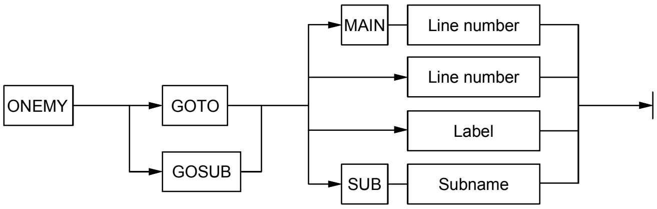
Reads in the state of the emergency stop input (connector X7, PLC input).

If this input is ≥10V and < 24V → EMY_STOP = TRUE (=65535)

If this input is <10V → EMY_STOP = FALSE (=0).

Example: IF EMY_STOP THEN GOTO 100

19.16 ONEMY



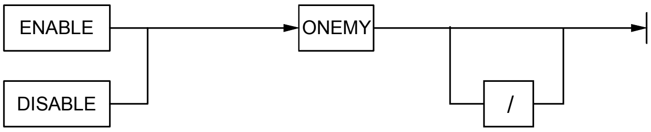
Function:

This statement defines a program branch to the specified line number or subroutine in case of an interrupt generated by the Emergency Stop input. (connector X7 pin 5).

Parameter	Input	Range	Description
Line number	number	0-65535	target line number

Example: ONEMY GOTO 25500
ONEMY GOSUB 8880

19.17 ENABLE [EN.]/DISABLE [DI.] ONEMY



Function:

Enable or disable the Emergency Stop interrupt. The input is transition sensitve.

- Enable the interrupt for a positive going edge: ENABLE ONEMY
- Enable the interrupt for a negative going edge: ENABLE ONEMY /

Example: ENABLE ONEMY
DISABLE ONEMY

19.18 RDY

Function:

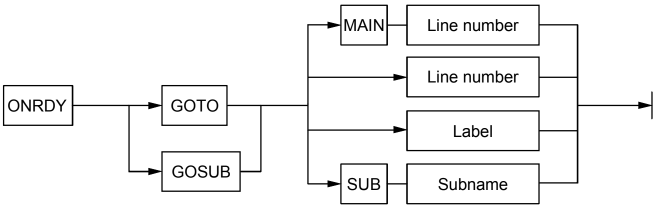
Read the state of the Ready input(connector X7).

If this input is ≥10V and < 24V → RDY = TRUE (=65535)

If this input is <10V → RDY = FALSE (=0).

Example: IF RDY THEN GOTO 100

19.19 ONRDY



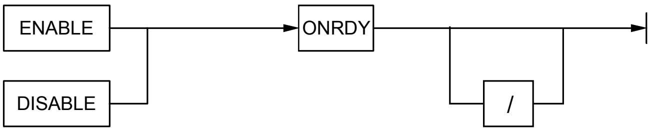
Function:

This statement defines a program branch to the specified line number or subroutine in case of an interrupt generated by the Ready input. (connector X7 pin 6).

Parameter	Input	Range	Description
Line number	number	0-65535	target line number

Example: ONRDY GOTO 25500
ONRDY GOSUB 8880

19.20 ENABLE [EN.]/DISABLE [DI.] ONRDY



Function:

Enable or disable the Ready interrupt. The input is transition sensitve.

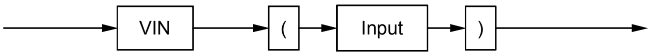
- Enable the interrupt for a positive going edge: ENABLE ONRDY.

Enable the interrupt for a negative going edge: ENABLE ONRDY /.

Example: ENABLE ONRDY
DISABLE ONRDY

20. Analogue Inputs and Outputs

20.1 VIN



Function:

With this statement three analogue inputs can be read.

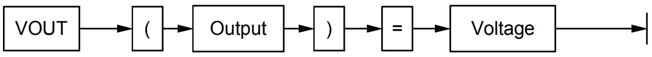
Parameter	Input	Range	Description
Input	num. expr.	0-2	Number of the input

Input ranges

Input	Input range	Read value	Connector
Input 0	0...5V	0-255	X5/25- X5/23
Input 1	-10V ... +10V	-10.00- +10.00 Volts	X6/12-X6/10
Input 2	0 ...+10V	0 - +10.00 Volts	X6/13-X6/10

Example: PRINT;VIN(1)
X=VIN(V)

20.2 VOUT



Function:

With this statement two analogue values can be read out.

Parameter	Input	Range	Description
Output	num. expr.	1-2	Output number
Voltage	num. expr.	-10.00...+10.00	Output voltage

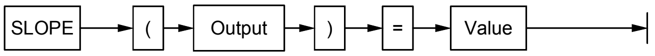
Example: VOUT(1) = -3.4
VOUT(2) = SPG(3)

Note:

The voltage output can be done with or without a linear slope function. The resolution of this slope is 10mV.

These analogue outputs are only available with the option D2.

20.3 SLOPE



Function:

Slope function for the analogue outputs.

Parameter	Input	Range	Description
Output	num. expr.	1-2	Number of the analogue output
Value	num. expr.	0...2000 1=10msec	Slope time. Resolution = 10msec.

Value = 0 switches off the slope function.

The slope function applies to all specified analogue outputs which follow the slope statement.

Note:

The programmed time relates to a voltage change. If an output changes from the value of a to the value of b, the slope follows the difference b-a.

Example: SLOPE(2)=7 or SLOPE (A)=X

These analogue outputs are only available with the option D2.

21. Interrupt Handling

21.1 Interrupt Sources

COMTAC has 48 different interrupt sources:

Source	Interrupt service statement	Description
Error	ONERR GOTO/GOSUB	COMTAC system error
COMPAX-Error	ONCPXERR GOTO/GOSUB	A COMPAX device connected via field bus to COMTAC reports an error or a warning.
Emergency Stop input (EMY_STOP)	ONEMY GOTO/GOSUB	The programmed transition of the Emergency Stop input.
Timer	ONTIMER value, target line.	The BASIC timer reached the programmed preset value.
COMTAC key board	ONKBD GOTO/GOSUB	A key has been pressed or released.
RS232/1-interface	ON#0 GOTO/GOSUB	According to the interface protocol characters have been received and the flag "Input-Rdy" in the control register STSCTR#0 has been set.
RS485/1-interface	ON#1 GOTO/GOSUB	**No field bus mode:** According to the interface, protocol characters have been received and the flag "Input-Rdy" in the control register STSCTR#1 has been set. **Field bus Host:** A field bus device generated one of the following messages: - Time out: Device failiure, bus connectors disconnected - Alarm: Device reports an event; service request - Error: Device reports an error - Data: the requested data is available and can be read by the Host - Quit: the command quit can be read by the Host
RS232/2-interface	ON#2 GOTO/GOSUB	According to the interface protocol, characters have been received and the flag "Input-Rdy" in the control register STSCTR#2 has been set.
RS485/2-interface (Option F6)	ON#3 GOTO/GOSUB	According to the interface protocol, characters have been received and the flag "Input-Rdy" in the control register STSCTR#3 has been set.
RS232/3-interface (Option F6)	ON#4 GOTO/GOSUB	According to the interface protocol, characters have been received and the flag "Input-Rdy" in the control register STSCTR#4 has been set.
RS485/5-interface (not available at the moment)	ON#5 GOTO/GOSUB	According to the interface protocol, characters have been received and the flag "Input-Rdy" in the control register STSCTR#5 has been set.
Real time clock (RTC)	ONTIME GOTO/GOSUB	The RTC time is equal to or greater than the programmed preset value.
Ready input (RDY)	ONRDY GOTO/GOSUB	The Ready input has been activated.
Key switch (KEYSWITCH)	ONKEY 19 GOTO/GOSUB	The key switch has been closed or opened.
Function key 1 to 18	ONKEY x GOTO/GOSUB	The specified key has been pressed
Digital inputs 1 to 16	ONINP GOTO/GOSUB	The specified input has been activated

21.1.1 Initialize Interrupts

Each interrupt source has to be initialized in order to be serviced by COMTAC.
The statement **ON*Interupt* GOTO/GOSUB line-no.** defines the **target address** and prepares the **operating system** to acknowledge an interrupt.

21.1.2 Interrupt Control

Each interrupt which is initialized and enabled is supervised by the operating system.
Each interrupt source has its own interrupt flag. This flag is set when the programmed interrupt condition is set true.

21.1.3 Enable the Interrupt Control

The interrupt source is supervised if the interrupt is enabled. The statement **ENABLE ON*Interrupt*** enables the interrupt source provided that this source has been initialized (ON*Interupt*...) Otherwise the error 56 occurs.
The ENABLE statement clears the interrupt flag.

Important

The statement **ON*Interrupt* GOTO/GOSUB line no.** must precede the statement **ENABLE ON*Interrupt***.

21.1.4 Disable the Interrupt Control

The statement **DISABLE ON*Interrupt*** suspends the interrupt supervision and clears a possible pending interrupt.

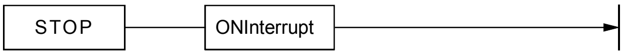
21.1.5 Interrupt Branch

After the operating system has finished the execution of a BASIC line an interrupt control routine starts. This routine checks for pending interrupts and carries out a branch in the case of an interrupt.

Note

Interrupts are always recognized at the end of a BASIC line. An interrupt occurence can not interrupt an immediately executed statement. For this reason, avoid programming more than one statement in one BASIC line, in order to provide fast interrupt response times.

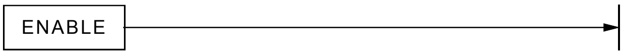
21.1.6 Disable Interrupt execution



The statement **DISABLE** disables each interrupt execution . (exception: ONERR). The statement **STOP ON*Interrupt*** disables the specified interrupt execution. The interrupt control (interrupt flag) remains set.

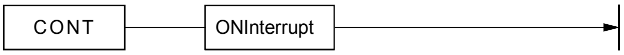
21.1.7 Enable Interrupt Execution

Enable



The statement **ENABLE** enables each interrupt execution.

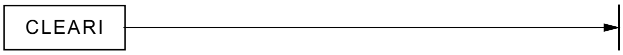
CONT ON*Interrupt*



The statement **CONT ON*Interrupt*** enables the execution of the specified interrupt.

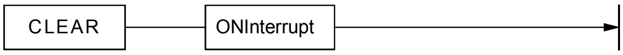
21.1.8 Reset Interrupts

CLEARI



The statement **CLEARI** clears all pending interrupts, disables the supervision of the interrupts, enables the interrupt execution and resets the priority settings for the interrupts.
The initializitation of the interrupt is maintained.

CLEAR ON*Interrupt*



The statement **CLEAR ON*Interrupt*** resets the specified interrupt: the interrupt is cleared and disabled, the interrupt execution is enabled. If this interrupt is in process while the statement CLEAR ON Interrupt is carried out, other interrupts with the same priority level can interrupt this one.
The initializitation of this interrupt is maintained.

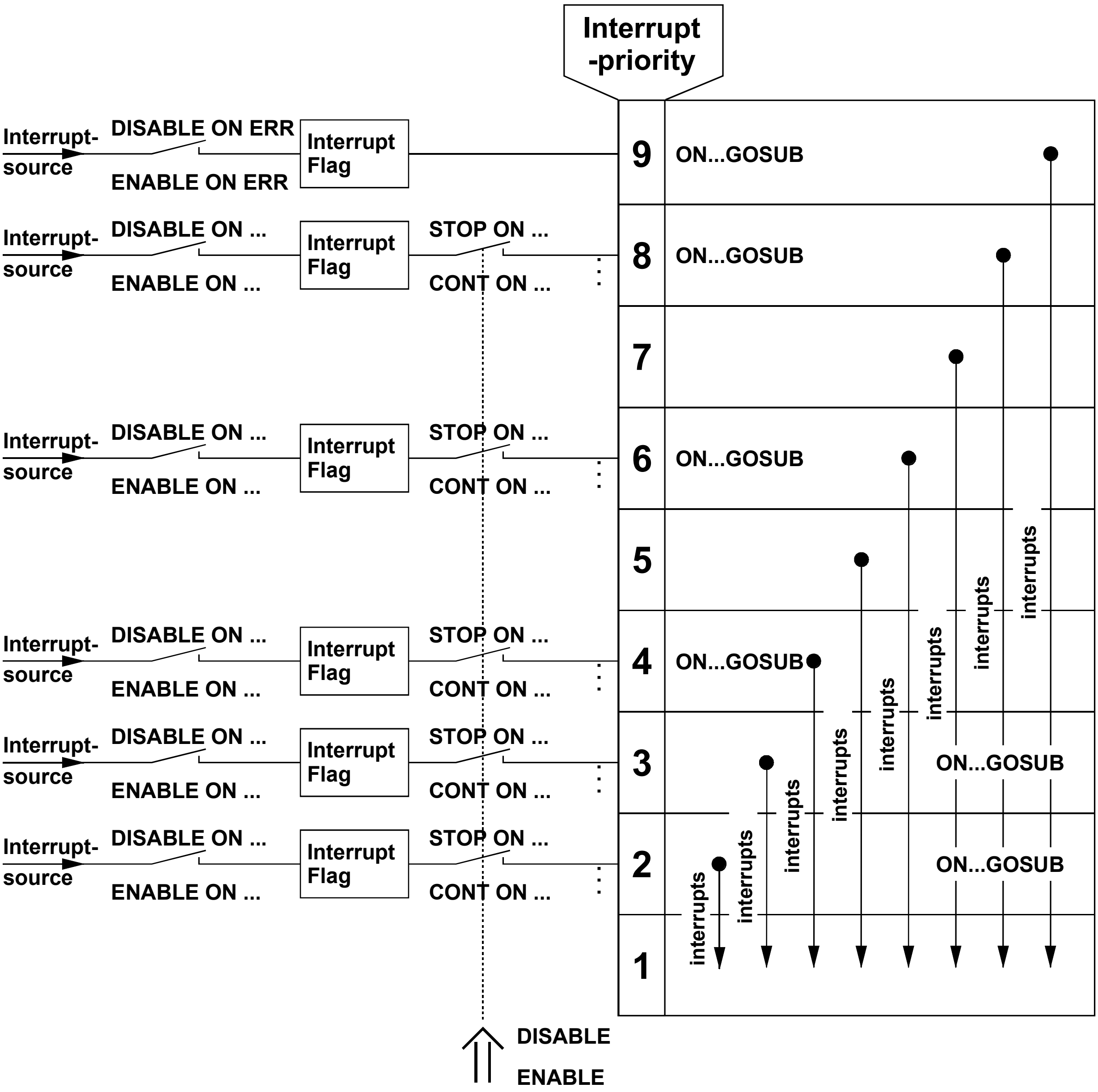
21.1.9 Return from Interrupt

The statement **RETI** closes an interrupt subroutine. Pending interrupts with the same priority level as the closed interrupt now can be carried out.

21.2 Interrupt Priority

- COMTAC provides nine levels of interrupt priority. The error interrupt (**ONERR GOSUB ...**) has the highest level (9) and can´t be interrupted. The priority for this interrupt is fixed.
- To any other interrupt a priority level of 1 (lowest) to 8 (highest) can be assigned.
- If an interrupt source of higher priority makes a service request while a lower priority request is pending, or an interrupt subroutine of a lower priority request is carried out, the higher priority request is serviced.
- An interrupt request with a priority equal to or lower than the current interrupt is not serviced, but the interrupt flag is set.
- An interrupt subroutine has to be initialized with the **ON*Interupt*.GOSUB ...**and has to be closed with the **RETI** statement.
- A Goto-branch caused by an interrupt has the priority settings of (**ON*Interrupt*.GOTO ...**) but the statements carried out following the branch can be interrupted by occurences with the same or lower priorities.
- After power on or reset the interrupts are set to the lowest priority (1) with the exception of the Error interrupt.

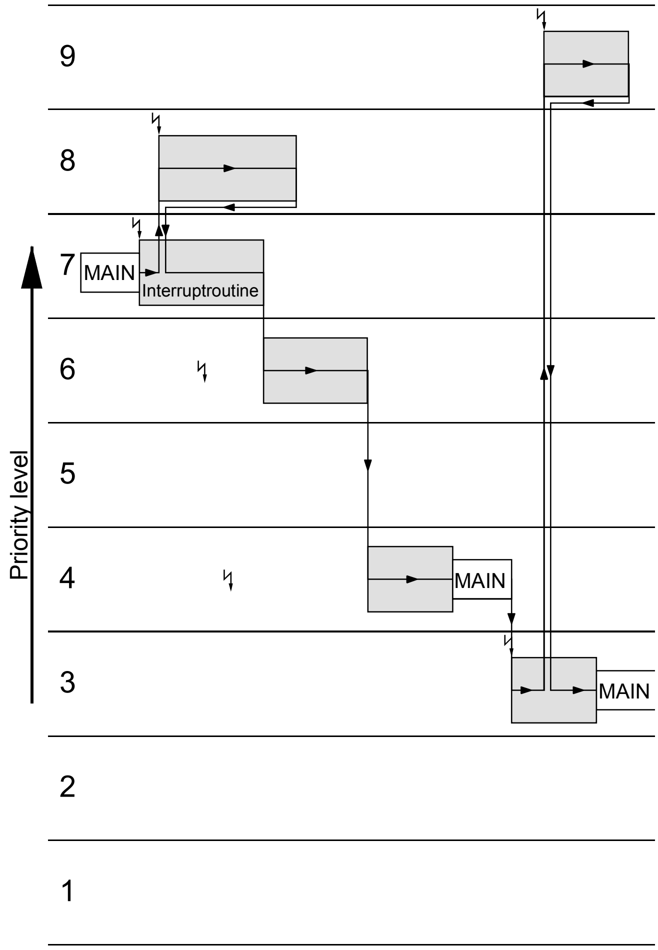
Descrition of the Interrupt functions



Description:

- The error interrupt always has the highest priority (9).
- Other Interrupt sources can be set to a priority level.
  More than one interrupt may have the same priority.
- ENABLE ON...enables an interrupt source and resets the interrupt flag.
- DISABLE ON... disables the interrupt source and resets the interrupt flag.
- The interrupt flag stores an interrupt occurence.
- STOP ON ... prevents an interrupt from being serviced. The interrupt flag stores the service request (one only).
  CONT ON... resets the STOP ON condition.
- DISABLE prevents all interrupts from being serviced, but the interrupt flags store the service requests.
  ENABLE resets the DISABLE condition.

Example: Order of interrupt services



The arrows (↯) mark interrupt occurrences.

Interrupt priority for one priority level.

The interrupts are serviced in the following order if they are at the same priority level.

(1)	ONCPXERR
(2)	ONEMY-Interrupt
(3)	ONTIMER
(4)	ONKBD
(5)	ON#0
(6)	ON#1
(7)	ON#2
(8)	ON#3
(9)	ON#4
(10)	ON#5
(11)	ONTIME
(12)	ONRDY
(13)	ONKEY 19
(14)	ONKEY 17
(15)	ONKEY 18
(16)	ONKEY 1
(17)	ONKEY 2
(18)	...
(19)	ONKEY 16
(20)	ONINP 1
(21)	ONINP 2
(22)	...
(23)	ONINP 16
(24)	ONINP 1
(25)	ONINP 2
(26)	...
(27)	ONINP 16

21.2.1Error Interrupt

All COMTAC system errors can release this interrupt.
The error number and the BASIC line in which the error occurred are stored.

Initialization:	ONERR GOTO/GOSUB *Line no.*
Enable :	ENABLE ONERR
Disable:	DISABLE ONERR
Disable interrupt service:	not possible
Enable interrupt service:	not possible
Reset:	CLEAR ONERR
Priority set/read:	not possible
Error *no*.:	ERRSTS
Error line:	ERRSTSL
Error subroutine:	ERRSTS#
Description	see page 86

21.2.2 COMPAX Error Interrupt

If a COMAPX, via the field bus, reports an error. The device address of this COMPAX is stored.

Initialization:	ONCPXERR GOTO/GOSUB *Line no.*.
Enable:	ENABLE ONCPXERR
Disable:	DISABLE ONCPXERR
Disable interrupt service:	STOP ONCPXERR
Enable interrupt service:	CONT ONCPXERR
Reset:	CLEAR ONCPXERR
Set priority level:	PRIORITY ONCPXERR = *value*
Device address	CPXERRADR
Description	see page 102

21.2.3 Emergency Stop Input Interrupt

This input is transition sensitive. A rising or a falling edge can cause an interrupt.

Initialization:	ONEMY GOTO/GOSUB *Line no.*.
Enable:	ENABLE ONEMY or ENABLE ONEMY / (falling edge)
Disable:	DISABLE ONEMY
Disable interrupt service:	STOP ONEMY
Enable interrupt service:	CONT ONEMY
Reset:	CLEAR ONEMY
Set priority level:	PRIORITY ONEMY = x
Description	see page 65

21.2.4 Timer Interrupt:

Initialization:	ONTIMER *compare value* **,** *Line no..*
With reload function	ONTIMER *compare value* **;** *Line no..*
Enable (=start) timer:	ENABLE ONTIMER
Disable (=stop) timer	DISABLE ONTIMER
Disable interrupt service:	STOP ONTIMER
Enable interrupt service:	CONT ONTIMER
Reset:	CLEAR ONTIMER
Set priority level:	PRIORITY ONTIMER = x
Description	see page 57

21.2.5 COMTAC keyboard Interrupt

Each key can cause an interrupt. This keycode is stored. Releasing a key also causes an interrupt. The key code in this case is 0.

Initialization:	ONKBD GOTO/GOSUB *Line no..*
Enable:	ENABLE ONKBD
Disable:	DISABLE ONKBD
Disable interrupt service:	STOP ONKBD
Enable interrupt service:	CONT ONKBD
Reset:	CLEAR ONKBD
Set priority level:	PRIORITY ONKBD = *x*
Key code	KBDCODE
Description	see page 53

21.2.6 Interface Interrupt

According to the interface protocol characters have been received and the flag "Input-Rdy" in the control register STSCTR #x has been set.

Initialization:	ON#x GOTO/GOSUB *Line no..*
Enable:	ENABLE ON#x
Disable:	DISABLE ON#x
Disable interrupt service:	STOP ON#x
Enable interrupt service:	CONT ON#x
Reset:	CLEAR ON#x
Set priority level:	PRIORITY ON#x = *value*
Description	see page 77 und 81

21.2.7Real Time Clock Interrupt

The RTC time is equal to or greater than the programmed preset value. This interrupt is repeated daily.

Initialization:	ONTIME *preset value* GOTO/GOSUB *Line no..*
Enable:	ENABLE ONTIME
Disable:	DISABLE ONTIME
Disable interrupt service:	STOP ONTIME
Enable interrupt service:	CONT ONTIME
Reset:	CLEAR ONTIME
Set priority level:	PRIORITY ONTIME = *Wert*
Description	see page 55

21.2.8 Ready Input Interrupt

This input is transition sensitive. A rising or a falling edge can cause an interrupt.

Initialization:	ONRDY GOTO/GOSUB *Line no..*
Enable:	ENABLE ONRDY or ENABLE ONRDY / (falling edge)
Disable:	DISABLE ONRDY
Disable interrupt service:	STOP ONRDY
Enable interrupt service:	CONT ONRDY
Reset:	CLEAR ONRDY
Set priority level:	PRIORITY ONRDY = x
Description	see page 66

21.2.9 Key Switch Interrupt

This input is transition sensitive. A rising or a falling edge can cause an interrupt.

Initialization:	ONKEY 19 GOTO/GOSUB *Line no..*
Enable:	ENABLE ONKEY 19 or /19 (falling edge)
Disable:	DISABLE ONKEY 19
Disable interrupt service:	STOP ONKEY 19
Enable interrupt service:	CONT ONKEY 19
Reset:	CLEAR ONKEY 19
Set priority level:	PRIORITY ONKEY (19) = x
Description	see page 52

21.2.10 Function Key Interrupt

Each of the 16 function keys, the START (F17) and STOP(F18) key can cause an interrupt.

Initialization:	ONKEY x GOTO/GOSUB *Line no..*
Enable:	ENABLE ONKEY x
Disable:	DISABLE ONKEY x
Disable interrupt service:	STOP ONKEY x
Enable interrupt service:	CONT ONKEY x
Reset:	CLEAR ONKEY x
Set priority level:	PRIORITY ONKEY (x) = x
Description	see page 62

21.2.11 Digital Input Interrupt

Each of the 16 digital inputs can cause an interrupt. These can be programmed to be transition (rising or falling edge) or level sensitive (ONINP *). Level sensitive means that one interrupt is generated if the digital input has already been set to the programmed level when the statement ENABLE ONINP is executed.

The target for the rising and falling edge may be different or the same.

Example

ONINP 1 GOSUB 100,200

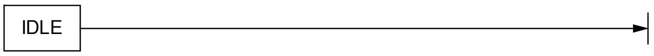
ENABLE ONINP %1 Differerent targets: Line *no*. 100 for the rising edge, line *no*. 200 for the falling edge.

ONINP 1 GOSUB 500

ENABLE ONINP %1 The target is line *no*. 500 for both edges.

Initialization:	ONINP x GOTO/GOSUB *Line no..*
Enable:	ENABLE ONINP x (rising edge) or ONINP * x
	ENABLE ONINP /x (falling edge) or ONINP * /x
	ENABLE ONINP %x (rising and falling edge) or ONINP * %x
Disable:	DISABLE ONINP 5
Disable interrupt service:	STOP ONINP 5
Enable interrupt service:	CONT ONINP 6
Reset:	CLEAR ONINP 7
Set priority level:	PRIORITY ONINP (10) = x
Description	see page 62

21.3 IDLE (Wait for an Interrupt)



Function:

This allows the BASIC operating system to synchronise a system by use of the IDLE statement.

Example: IDLE

Note:

It is important to enable an interrupt before executing the IDLE statement.

22. RS485-Interfaces

COMTAC can work with two RS485 interfaces, RS485/1 and RS485/2. RS485/1 is standard, RS485 is optional and requires the additional hardware module F6. This module includes an RS232 interface.
The protocol of RS485/1 is adjusted with the parameters P60 to P79. As well as ASCII protocol, this interface can operate with the field bus protocol, which enhances the control of COMPAX devices with its wide range of special commands adapted for COMPAX functions.
The RS485/2 is adjusted with parameters P80 to P89 and operates with the ASCII protocol.
The maximum number of devices that can be connected to an RS485 interface is 32.

22.1 Description of the Functions

The functional description of the field bus protocol applies only if it is enabled (parameter P73=1).

⇨ Field bus protocol is only possible with the RS485/1.

Any RS485 device can be connected to COMTAC.
The wiring of this interface is described in the device description for COMTAC.
The interface protocol must correspond to the settings of the devices connected to this interface (baudrate, startbit, stopbits ...).
After power on the RS485/1/2 is set up with the parameters named above. These are stored in the ZPRAM.
Notation:
OUTPUT-Ready (indicates readiness to transmit data)=Statusbit 0 of the system variable STSCTR#1(3).
INPUT-Ready (indicates that data was received)=Statusbit 1 of the system variable STSCTR#1(3).

22.2 Parameters of the RS485/1 Interface (#1)

No.	Function		Default
60	Device address 0 - 255		0
61	Start character of the received string 0 - 255		2
62	End character of the received string 0 - 255		13
63	Receive protocol		2
	Bit	Function	
	0	Protocol Bit 0	0
	1	Protocol Bit 1	1
	2	Protocol Bit 2	0
	3		0
	4	Block-Check-Character; 0 = off, 1 = on	0
	5		1
	6		1
	7	Software handshake; 0 = off, 1 = on	0
	The protocol bits define one of these receive protocols:		
	PBitno	INPUT-Rdy = 1,	
	2 1 0		0 1 0
	0 0 0	after each received character	
	0 0 1	after a number of received characters (P66)	
	0 1 0	after the end of string character (P62)	
	0 1 1	after the start of string (P61) and end of string (P62) character	
	1 0 0	after the start of string (P61) and end of string (P62) character and the device address(P60)	
64	Baudrate 0=150 3=1200 6=9600 9=57600 1=300 4=2400 7=19200 10=172800 2=600 5=4800 8=28800 11=345600		6
65	Stopbits/Parity/Hardware Bit 0...2= Parity/Stopbit selection 0 without parity, 1 stopbit 1 without parity, 2 stopbit 2 with parity Even, 1 stopbit 3 with parity Odd, 1 stopbit Bit 3...5 = reserved Bit 6 = 4 wire 0 Comtac=sub device:TxD = Pin 2, 7; RxD = Pin 1, 6 1 Comtac=Host:TxD = Pin 1, 6; RxD = Pin 2, 7 Bit 7 = 2/4 wire =0:2 wire =1:4 wire		0
66	Number of characters per received string. 1 - 255		1
67	End character of the output string 0 - 255		13
68	Output protocol 0: the end character isn´t added automatically 1: CR is added automatically 2: CR LF is added automatically 3: (P67) is added automatically 4: CR LF and (P67) is added automatically 5: (P61) and (P62) is added automatically		1
69	Time out value for ENTER 1 statement 0 - 255 (1 = 0.1sec) 0 = no time out check		0

22.3 Parameters of the RS485/2 Interface (#3) (Option F6)

No.	Function		Default
80	Device address 0 - 255		0
81	Start character of the received string 0 - 255		2
82	End character of the received string 0 - 255 (must be 62 for COMPAX)		13
83	Receive protocol		2
	Bit	Function	
	0	Protocol Bit 0	0
	1	Protocol Bit 1	1
	2	Protocol Bit 2	0
	3		0
	4	Block-Check-Character; 0 = off, 1 = on	0
	5		1
	6		1
	7	Software-Handshake; 0 = off, 1 = on	0
	The protocol bits define one of these receive protocols:		
	PBitno	INPUT-Rdy = 1,	
	2 1 0		0 1 0
	0 0 0	after each received charater	
	0 0 1	after a number of received characters (P86)	
	0 1 0	after the end of string character (P82)	
	0 1 1	after the start of string (P81) and end of string (P82) character	
	1 0 0	after the start of string (P81) and end of string (P82) character and the device address(P80)	
84	Baudrate 0=150 3=1200 6=9600 9=57600 1=300 4=2400 7=19200 10=76800 2=600 5=4800 8=38400 11=115200		6
85	Parity / stopbits / character length		0
	Bit 0	Number of stop bits: 0 = 1 stop bit 1 = 2 stop bit	
	Bit 1	Parity Enable: 0 = Disable 1 = Enable	
	Bit 2/ Bit 3	Parity Select: 0/0 = Odd 1/0 = Even 0/1 = Mark 1/1 = Space	
	Bit 4/ Bit 5	Character length: 0/0 = 8Bit 1/0 = 7Bit 0/1 = 6Bit 1/1 = 5Bit	
	Bit 6	reserved	
	Bit 7	2/4-wire 0 = 2-wire 1 = 4-wire	
86	Number of characters per received string. 1 - 255		1
87	End character of the output string 0 - 255		13

No.	Function	Default
88	Output protocol 0: the end character isn´t added automatically 1: CR is added automatically 2: CR LF is added automatically 3: (P87) is added automatically 4: CR LF and (P87) is added automatically 5: (P81) and (P82) is added automatically	1
89	Time out value for the ENTER statement 0 - 255 (1 = 0.1sec) 0 = no time out check	0

22.4 Receive Characters

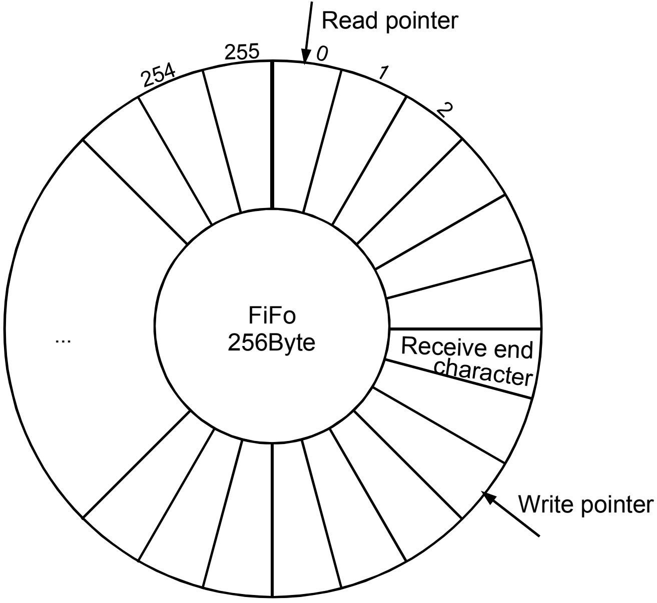
The driver routine for the receive procedure consits of a FIFO (first in first out) buffer with a capacity of 256 byte (characters) and a read and write pointer to access this buffer.

All received characters but not including control characters are stored in this buffer. The storing is controlled by the receive protocol (P63/83).

When the receive condition, defined with P63/83, becomes true (INPUT READY=1, STSCTR#1/#3) one received string (without the control characters) is copied from the FIFO to the corresponding string $(1) / $(3). This is done with the ENTER statement. These strings can be used with other BASIC statements.

The ENTER statement clears the INPUT READY bit if there are no more strings to be received.

The ENTER statement polls the INPUT READY bit and waits till this gets to logic 1(TRUE) for the period of time defined in P69/89. If this time runs out before INPUT READY gets to a 1, a time out error occurs.



FIFO: The characters received are stored with the write pointer and read with the read pointer. If the ENTER statement isn´t carried out after characters are received the buffer becomes full and the characters will be overwritten.

22.5 Output Characters

The OUTPUT *string* statement copies the *string* to the output buffer. This buffer has a capacity of 256 bytes. A string is always copied by writing the first character to the first location of the output buffer (and so on).

The driver routine for the output of strings reads out the characters and adds, if necessary, control characters according to P68/88.

The bit OUTPUT-Ready (STSCTR#1/#3) remains at 0 for as long as the output takes place. When the output has finished this bit changes to a 1. Then the next string can be sent out. The OUTPUT READY has to be polled in the user program.

22.6 BCC RS485

The Block Check Character (BCC) enhances the reliability of data transmission. This function can be enabled / disabled with P63/83.

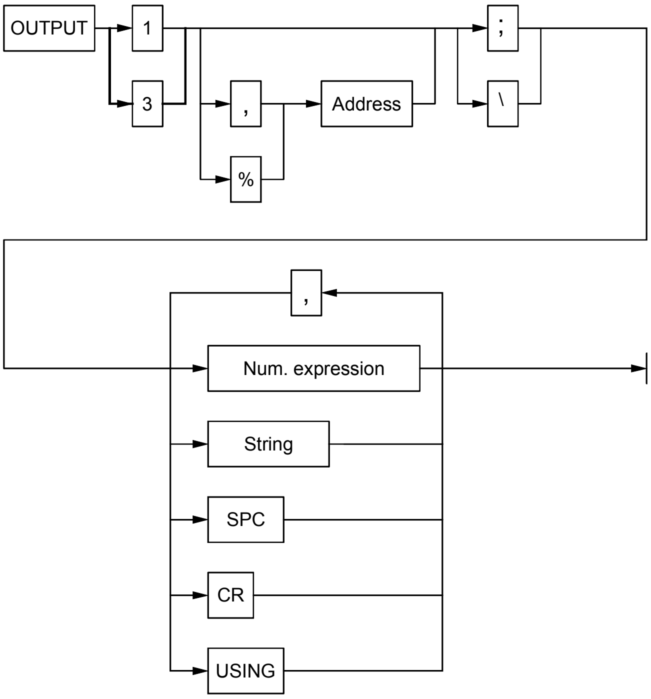
BCC XORs all characters to be send except the end character. The result of this logical combination is a byte which is added to the output string after the end character.

If COMATC receives a string with BCC the received BCC character is compared with the calculated sum.

If the BCC is enabled the devices connected to this interface must support this function.

If the result of the comparison is negative an error occurs.

22.7 OUTPUT [O.] RS485



Function:

This statement reads out characters, strings, and numerical expressions.

This statement reads out characters, strings and numerical expressions.

It´s possible to receive a response ($(1),$(3)) simultaneously with this statement by using the back slash instead of the semicolon in the OUTPUT statement.

Parameter	Input	Range	Description
Address	num. expr.	0 - 255	device address

The device address is optional.

Two output formats for the address are possible:

- comma (,) = ASCII format
- Percent (%)= binary format.

Example: OUTPUT 1;"Angle=",A
OUTPUT 1,5;"PA",XPOS,CHR$(13)

Note:

The operating system now automatically checks the OUTPUT-Ready flag.

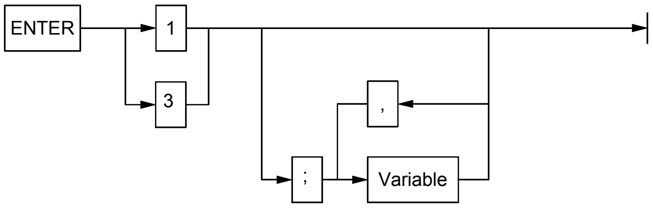
As occurs with the ENTER instruction, the command is cancelled after the timeout period.

Only then is the output buffer of the corresponding RS232 interface empty.

Parameter 68(88) defines the end charatcer which is added automatically.

A description of SPC, CR, USING can be found in chapter 14.

22.8 ENTER [E.] (RS485)



Function:

This statement copies a received string from the receive buffer (FIFO) to the string $(1) or $(3).

A string may contain letters and numbers. An optional function of the input statement allows the copying of numbers direct to BASIC variables.

The first series of numbers detected in the string is assigned to the first variable name of the ENTER statement, the second to the second variable and so on.

The number of variables need not correspond to the number of numerical strings found in the received string.

Example: Received string = "10V200X-30"
ENTER 1,address; A,B,C,D

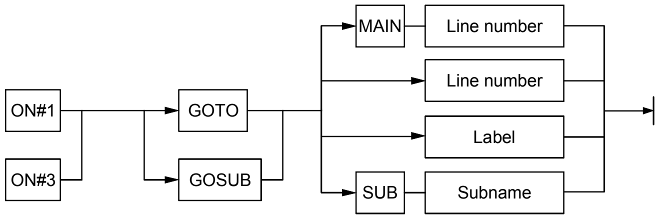
After the execution of ENTER: $(#1) = "10V200X-30"
A = 10
B = 200
C = -30 and D has not changed.

Note:

While the ENTER statement is executed the BASIC program waits until a string has been received, or for a *time out*.

In this case an error message occurs.

22.9 ON#1/#3



Function:

This statement defines the interrupt conditions for the RS485 interface(s). The interrupt is generated if a string was received (INPUT READY=1).See ENABLE/DISABLE ON#1/#3).

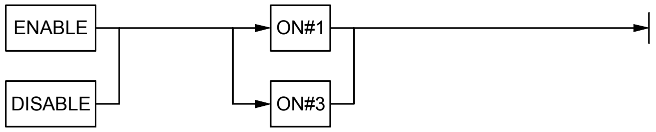
Parameter	Input	Range	Description
Line number	number	0 - 65535	target line number

Example: ON#1 GOTO ...
ON#1 GOSUB ...

Note:

Parameter 63 (83) defines the conditions for a valid string.

22.10 ENABLE [EN.] / DISABLE [DI.] (RS485)

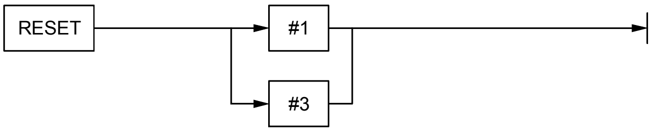


Function:

Enable / disable the RS485 interrupt.

Example: ENABLE ON#1
DISABLE ON#1

22.11 RESET [RS.] RS485

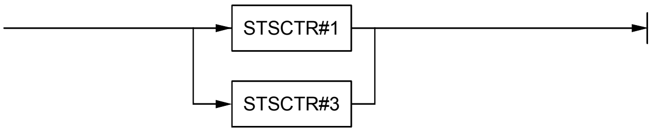


Function:

Reset the RS485 interface to the default parameters.

Examples: RESET #1, RESET#3

22.12 STSCTR#1/#3



Function:

This system variable reports the actual state of the RS485 interface(s). The bits of this register have the following meaning. They are active high:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
BCC Error	Parity Error	Overrun Error	0	0 oder 1	0 oder 1	INPUT Ready	OUTPUT Ready
128	64	32	16	8	4	2	1

- Bit 0: output buffer empty. A string can be sent out.
- Bit 1: a string was received and can be read with the ENTER statement.
- Bit 2: reserved
- Bit 3: reserved
- Bit 4: reserved
- Bit 5: Input buffer overflow
- Bit 6: Parity error
- Bit 7: BCC error

Examples: PB.STSCTR#1: displays the register contents in binary formt.

IF (STSCTR#1 'AND' 10B) = 0 THEN ...: Condtional jump if bit 1 is set to logic 1.

Note:

- The errorbits BCC-, Parity-, Overrun are cleared when reading STSCTR#1/#3.
- STSCTR#3 is the status register for the RS485/2 interface.

23. RS232 Interface

23.1 Description

Any device with an RS232 interface can be connected to the COMTAC.

The wiring of this interface is described in the device description for COMTAC.

The interface protocol must correspond to the settings of the devices connected to this interface (baudrate, startbit, stopbits ...).

After power on the RS232 is set up with the following named parameters. These are stored in the ZPRAM.

RS232/1 with parameters 50...59

RS232/2 with parameters 90...99.

RS232/3 with parameters 40...49 (Option F6).

Parameter 100 defines which RS232 is used for the terminal communication. Default is RS232/1.

Notation:

RTS	Hardware handshake output of COMTAC (1 = ready to receive data, 0 = not ready).
CTS	Hardware handshake input;(1 = Comtac can send out data, 0 = disabled).
OUTPUT-Ready	Status bit 0 of the system variable STSCTR#0 resp. STSCTR#2 resp. STSCTR#4
INPUT-Ready	Status bit 1 of the system variable STSCTR#0 reps. STSCTR#2 resp. STSCTR#4

⇨ The RS232 interfaces operate with the ASCII protocol. The RS232/3 has the ability to operate with 3964(R) protocol. (see P41, P42, P43).

* The settings 9,10,11 of the parameters P54 (RS232/1) and P94 (RS232/1) are valid for a device clock frequency of 29,4912 MHz.
For older devices with a lower clock frequency the baud rate is as follows:

	P54/P94=9	P54/P94=10	P54/P94=11
11,0592MHz			
RS232/1	57600	57600	115200
RS232/2	57600	57600	57600
24,5760MHz			
RS232/1	38400	76800	76800
RS232/2	38400	76800	76800

the clock frequency of teh device may be determined via the command XTAL

23.2 Parameters of the RS232/1 interface (#0)

No.	Function		Default
50	Device address 0 - 255		0
51	Start character of the received string 0 - 255		2
52	End character of the received string 0 - 255 (must be 62 for COMPAX)		13
53	Receive protocol		34
	Bit	Function	
	0	protocol Bit 0	0
	1	protocol Bit 0	1
	2	protocol Bit 0	0
	3	-	0
	4	Block-Check-Character; 0=off, 1=on	0
	5	Auto-RTS; 0 = off, 1 = on	1
	6	Hardware-Handshake; 0 = off, 1 = on	0
	7	Software-Handshake; 0 = off, 1 = on	0
	The protocol bits define one of these receive protocols:		
	PBitno	INPUT-Rdy=1,	
	2 1 0		0 1 0
	0 0 0	after each received charater	
	0 0 1	after a number of received characters (P56)	
	0 1 0	after the end of string character (52)	
	0 1 1	after the start of string (P51) and end of string (P52) character	
	1 0 0	after the start of string (P51) and end of string (P52) character and the device address(P50)	
54	Baudrate 0=150 3=1200 6=9600 9=57600* 1=300 4=2400 7=19200 10=76800* 2=600 5=4800 8=38400 11=115200*		8
55	Parity and Stopbits 0 = without Parity, 1 Stopbit 1 = without Parity, 2 Stopbit 2 = with Parity EVEN, 1 Stopbit 3 = with Parity ODD, 1 Stopbit		0
56	Number of characters per received string. 1 - 255		1
57	End character of the output string 0 - 255		13
58	Output protocol 0: the end character isn´t added automatically 1: CR is added automatically 2: CR LF is added automatically 3: (P57) is added automatically 4: CR LF and (P57) is added automatically 5: (P51) and (P52) is added automatically		1
59	Time out value for the ENTER statement 0 - 255 (1 = 0.1sec) 0 = no time out check		0

23.3 Parameters of the RS232/2-Interface (#2)

No.	Function		Default
90	Device address 0 - 255		0
91	Start character of the received string 0 - 255		2
92	End character of the received string 0 - 255		13
93	Receive protocol		34
	Bit	Function	
	0	protocol Bit 0	0
	1	protocol Bit 0	1
	2	protocol Bit 0	0
	3	-	0
	4	Block-Check-Character; 0=off, 1=on	0
	5	Auto-RTS; 0 = off, 1 = on	1
	6	Hardware-Handshake; 0 = off, 1 = on	0
	7	Software-Handshake; 0 = off, 1 = on	0
	The protocol bits define one of these receive protocols:		
	PBitn o	INPUT-Rdy=1,	
	2 1 0		0 1 0
	0 0 0	after each received charater	
	0 0 1	after a number of received characters (P96)	
	0 1 0	after the end of string character (92)	
	0 1 1	after the start of string (P91) and end of string (P92) character	
	1 0 0	after the start of string (P91) and end of string (P92) character and the device address(P90)	
94	Baudrate 3=1200 6=9600 9=57600* 4=2400 7=19200 10=76800* 5=4800 8=38400 11=115200*		8
95	Parity and Stopbits 0 = without Parity, 1 Stopbit 1 = without Parity, 2 Stopbit 2 = with Parity EVEN, 1 Stopbit 3 = with Parity ODD, 1 Stopbit		0
96	Number of characters per received string. 1 - 255		1
97	End character of the output string 0 - 255		13
98	Output protocol 0: the end character isn´t added automatically 1: CR is added automatically 2: CR LF is added automatically 3: (P97) is added automatically 4: CR LF and (P97) is added automatically 5: (P91) and (P92) is added automatically		1
99	Time out value for the ENTER statement 0 - 255 (1 = 0.1sec); 0 = no time out check		0
100	Defines which RS232 is used as terminal interface: 0 = RS232/1 1 = RS485/1 2 = RS232/2 3 = RS232/3		

* see page 78.

23.4 Parameters of the RS232 /3 Interface (#4) (Option F6)

No.	Function		Default
40	Device address 0 - 255		0
41	Start character of the received string 0 - 255 or Character delay time (with 3964 protocol) (1⇔50ms)		2
42	End character of the received string 0 - 255 or Quit delay time (with 3964 protocol) (1⇔50ms)		13
43	Receive protocol		34
	Bit	Function	
	0	protocol Bit 0	0
	1	protocol Bit 1	1
	2	protocol Bit 2	0
	3	-	0
	4	Block-Check-Character; 0=off, 1=on	0
	5	Auto-RTS; 0=off, 1=on	1
	6	Hardware-Handshake; 0 = off, 1 = on	0
	7	Software-Handshake; 0 = off, 1 = on	0
	The protocol bits define one of these receive protocols:		
	PBitn o	INPUT-Rdy = 1,	
	2 1 0		0 1 0
	0 0 0	after each received character	
	0 0 1	after a number of received characters (P46)	
	0 1 0	after the end of string character (42)	
	0 1 1	after the start of string (P41) and end of string (P42) character	
	1 0 0	after the start of string (P41) and end of string (P42) character and the device address(P40)	
	1 0 1	3964 protocol	
44	Baudrate 0=150 3=1200 6=9600 9=57600 1=300 4=2400 7=19200 10=76800 2=600 5=4800 8=38400 11=115200		6
45	Parity / stopbits / character length		0
	Bit 0	number of Stopbits: 0 = 1 Stopbit 1 = 2 Stopbit	
	Bit 1	Parity Enable: 0 = Disable 1 = Enable	
	Bit 2/ Bit 3	Parity Select: 0/0 = Odd 1/0 = Even 0/1 = Mark 1/1 = Space	
	Bit 4/ Bit 5	character length: 0/0 = 8Bit 1/0 = 7Bit 0/1 = 6Bit 1/1 = 5Bit	
	Bit 6	no function	
	Bit 7	no function	
46	Number of characters per received string. 1-255		1

No.	Function	Default
47	End character of the output string 0 - 255 repeat factor (with 3964 protocol)	13
48	Output protocol 0: the end character isn´t added automatically 1: CR is added automatically 2: CR LF is added automatically 3: (P497) is added automatically 4: CR LF and (P47) is added automatically 5: (P41) and (P42) is added automatically	1
49	Timeout value 0 - 255 (1 = 0.1s) 0 = no timeout check	0

23.5 Receive Characters

The driver routine for the receive procedure consists of a FIFO (first in first out) buffer with a capacity of 256 bytes (characters) and a read / write pointer to access this buffer.
All received characters, without the control characters defined by receive protocol P53/93/43, are stored in this buffer.
When the receive condition, defined by P53/93/43, becomes true (INPUT READY=1, STSCTR#0/#2#4) one received string (without control characters) is copied from the FIFO to the corresponding string $(0) / $(2) / $(4). This is done with the ENTER statement. These strings can also be used with other BASIC statements.
The ENTER statement clears the INPUT READY bit if there are no more strings received.
The ENTER statement polls the INPUT READY bit and waits till this becomes logic 1(TRUE) for the period of time defined in P59/99/49. If this time runs out before INPUT READY becomes a 1, a time out error occurs.

23.6 Output Characters

The OUTPUT *string* statement copies the *string* to the output buffer. This buffer has a capacity of 256 bytes. A string is always copied by copying the first character to the first location of the output buffer (and so on).
The driver routine for the output of strings reads out the characters and adds, if necessary, control characters according to P58/98/48.
The bit OUTPUT-Ready (STSCTR#0/#2/#4) remains at logic 0 whilst the output is active. When the output has finished this bit changes to 1. Then the next string can be sent out. The OUTPUT READY has to be polled in the user program.
The output of characters can be controlled by hardware or software handshake:

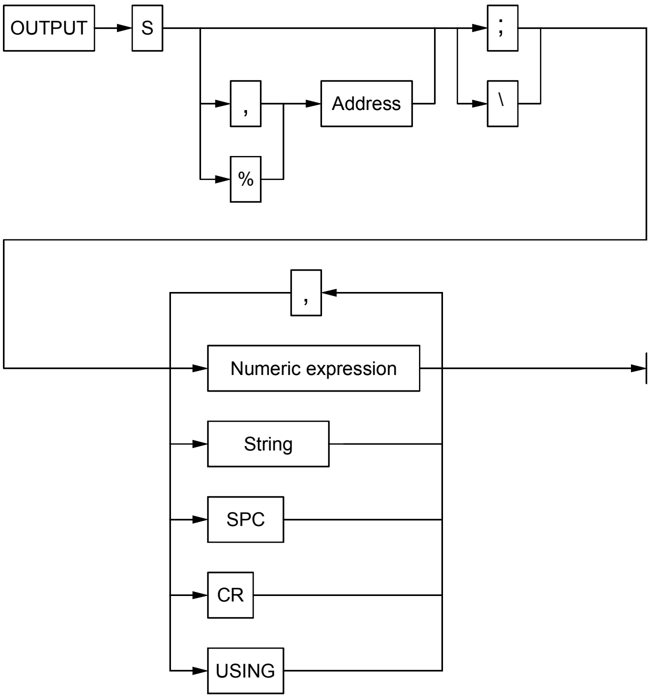
Hardware-Handshake
CTS = 0 output disabled
CTS = 1 output enabled

Software-Handshake
Receiver sends X-OFF (13H) --> output disabled.
Receiver sends X-ON (11H) -->output enabled.

23.7 BCC (RS232)

The Block Check Character (BCC) enhances the reliability of the data transmission. This function can be enabled / disabled with P53/93/43.
BCC XORs all characters to be sent except the end character. The result of this logical combination is a byte which is added to the output string after the end character.
If COMATC receives a string with BCC the received BCC character is compared with the calculated sum.
If the result of the comparison is negative an error occurs.
If the BCC is enabled the devices connected to this interface must support this function.

23.8 OUTPUT [O.] (RS232)



Function:

This statement reads out characters, strings, and numerical expressions.
OUTPUT 0... output via RS232/1
OUTPUT 2... output via RS232/2
OUTPUT 4... output via RS232/3
It´s possible to receive a response ($(0),$(2),$(4)) simultaneously with this statement by using the back slash instead of the semicolon in the OUTPUT statement.

Parameter	Input	Range	Description
Address	num. expr.	0 - 255	Device address of the receiver
S	Number	0,2 or 4	0 RS232/1 2 RS232/2 4 RS232/3

The device address is optional.
Two output formats for the address are possible:

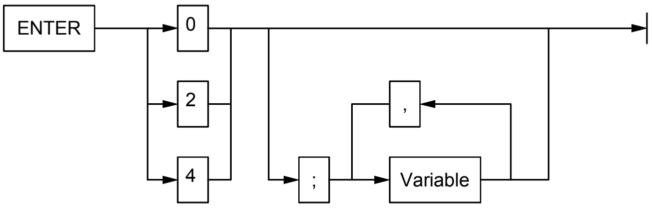
- comma(,): ASCII format
- decimal point(.): binary format

Example: OUTPUT 0; "Angle=",A
OUTPUT 2,5; "SPA",XPOS,CHR$(10)

Note:

Das Betriebssystem prüft nun selbständig das OUTPUT-Ready-Flag.
Wie bei der ENTER-Anweisung wird nach Ablauf der Time-out-Zeit der Befehl abgebrochen.
Nur dann ist der Ausgabepuffer der entsprechenden RS232-Schnittstelle leer.
Parameter 58(98,48) defines the end charatcer which is added automatically.

23.9 ENTER [E.] (RS232)



Function:

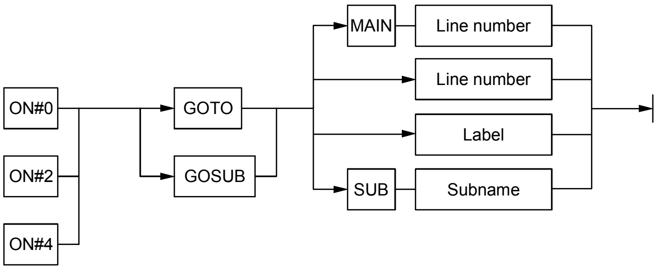
This statement copies a received string from the receive buffer (FIFO) to the string $(0), $(2) or $(4).
A string may contain letters and numbers. An optional function of the input statement allows numbers to be passed direct to BASIC variables.
The first series of numbers detected in the string are assigned to the first variable name of the ENTER statement, the second to the second variable and so on.
It´s not nescessary for the number of variables to correspond to the number of numerical strings found in the received string.

Example: Received string = "10V200X-30"
ENTER 1,address; A,B,C,D
After the execution of ENTER: $(#1) = "10V200X-30"
A = 10
B = 200
C = -30 and D has not changed.

Note:

While the ENTER statement is executed the BASIC program waits till a string has been received or times out and generates an error message.

23.10 ON#0 / #2 /#4



Function:

This statement defines the interrupt conditions for the RS232 interface(s). The interrupt is generated if a string was received (INPUT READY=1).See ENABLE/DISABLE ON#0'2/#4).

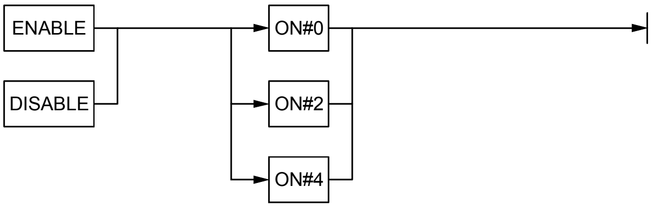
Parameter	Input	Range	Description
Line number	No.	0 - 65535	target line number

Example: ON#0 GOTO 100
ON#2 GOSUB 150

Note:

Parameter 53 (93,43) defines the conditions for a valid string.

23.11 ENABLE [EN.]/DISABLE [DI.] RS232



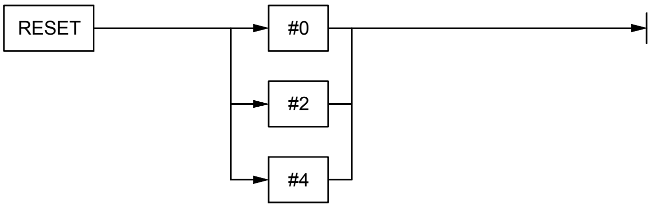
Function:

Enable / disable the RS232/1, RS232/1 or RS232/2 -Interrupt.

Example:

ENABLE ON#0 (enables the RS232/1 interrupt)
DISABLE ON#2 (disables the RS232/2 interrupt)

23.12 RESET [RS.] RS232

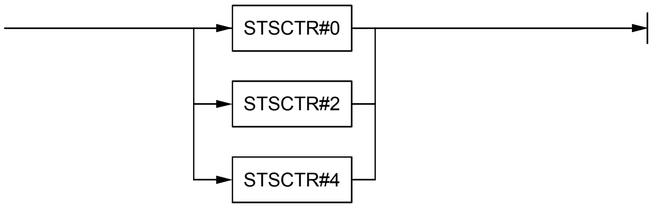


Function:

Reset the RS232 interface(s) to the default parameters.

Example: RESET #0
RESET #2

23.13 STSCTR#0 /#2 /#4



Function:

This system variable informs about the actual state of the RS232 interface(s). The bits of this register have the meaning described below. They are active high:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
BCC Error	Parity Error	Overrun Error	Framing Error	CTS	RTS	INPUT Ready	OUTPUT Ready
128	64	32	16	8	4	2	1

- Bit 0: output buffer empty. A string can be send out..
- Bit 1: a string was received and can be read with the ENTER statement..
- Bit 2: State of the handshake line RTS.
- Bit 3: State of the handshake line CTS.
- Bit 4: Framing error
- Bit 5: Input buffer overflow
- Bit 6: Parity error
- Bit 7: BCC error.

Example: PB.STSCTR#0: displays the register contents in the binary formt.

```
IF (STSCTR#2 'AND' 10B) = 0 THEN ....
STSCTR#2 = STSCTR#2 'OR' 4 : REM set the RTS output to log. 1
```

Note:

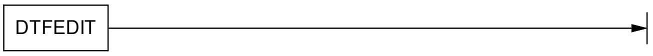
The errorbits BCC-, Parity-, Timeout are cleared when reading STSCTR#0/#2#4.
RTS can be set with the assignment STSCTR#no=value.

24. DATA-TEXT-ARRAY

24.1 General

To store machine or production data COMTAC has 4096 bytes of space in the RAM.
All user programmes can have access to this data.
The statements NEW, CLEAR or RUN have no affect on the contents of the Data Text Array.
This array can be stored like a program file from the RAM to the ZPRAM (drive R) or on diskette. The extension of a file name must be *name*.DAT

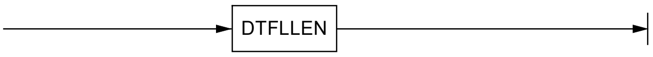
24.2 Data-Text-Editor: DTFEDIT



Function:

Start the Data Text Editor.
In the command mode the editor can also be started with CNTL+X.
The editor is line orientated. The number of lines depends on the number of characters per line. This number can be defined by the user with the statement DTFLLEN. The maximum number of characters in this array is 4000.
The Data Text Array consists of a header in which the length of a line is defined. The maximum number of lines is calculated automatically.
If an array is already defined when starting the editior the contents of this array will be displayed and can be edited.
Characters which can´t be displayed are indicated by ~ characters.
If no line length is defined it has to be entered. The editor initialises the array by calculating the number of lines and filling each line with blanks.
Attention! All existing Data Text Arrays (in the RAM) are cleared.
The first character (byte) of the first line is stored at location 0. The line length is stored at location 4000, the number of lines at location 4001(high byte) and 4002(low byte). Locations 4003 to 4015 are used by the editior. The headline is stored at location 4016 to 4095.

24.3 DTFLLEN

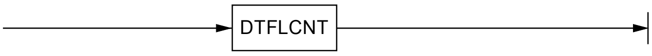


Function:

System variable for the line length, allowing it to be read or set to a particular value.

Example: DISP DTFLLEN
DTFLLEN = 60

24.4 DTFLCNT



Function:

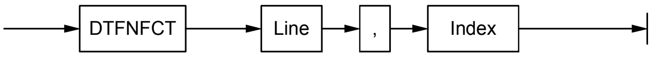
System variable for reading a line length.

Example: DISP DTFLCNT

Note:

The number of lines can only be changed by the number of characters per line.

24.5 DTFNFCT



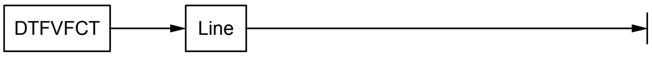
Function:

Read a number (specified by *index*) at a specified line number.
If no number is found in the specified line the return value is 0.
A separator between two numbers could be any ASCII character except a number, decimal point or comma.

Parameter	Input	Range	Description
Line	num. expr.	1...	Line of the Data Text Array
Number	num. expr.		Index of the number

Example: DISP DTFNFCT 10,3
A = DTFNFCT Y,N
At line 8 the text "3000;80;5480;4700;988" is stored. When Y=8 and N=2 the value 80 is assigned to the variable A.

24.6 DTFVFCT



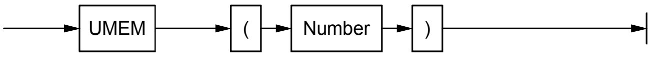
Function:

Assigns a value to a BASIC variable**. In this case only capital letters are possible.**

Parameter	Input	Range	Description
Line	num. expr.	1...	Line of the Data Text Array

Example: DTFVFCT 15
At line 15 the text "B3413 X88 Z250 Q9000" is stored. The statement DTFVFCT 15 assigns the value of 3413 to the BASIC variable B, 88 to X , 250 to Z and 9000 to Q.

24.7 UMEM(x) [¦(x)]



Function:

In this case the Data Text Array is used to store numerals in the floating point format. Each numeral needs 6 bytes of memory. Thus, 4096/6 = 682 numerals can be stored. The indexes for the first byte of these numerals are UMEM(0) to UMEM(681).

Parameter	Input	Range	Description
Index	num. expr.	0 - 681	Location of the first byte of the numeral

To calculate the location I of the first byte of a numeral the formula I=x*6 an be used.

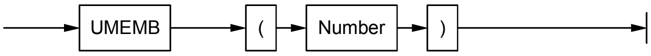
Example: UMEM(A)=B
A = 3 and B = 5233 the numeral 5233 is stored at location 18 to 23.

Note:

The UMEM statement makes direct access to Data Text Arrays stored in the ZPRAM (drive R) possible.
Therefore the filename of the Data Text Array has to be added to the statement.
The name must be placed between quotation marks or can be represented by a string.
The extension .DAT has to be left out.

Example: DISP UMEM "TEST" (105)
UMEM ;$(1) (333) = 7000
A = UMEM "W_DATA" (200)

24.8 UMEMB(x) [¦B(x)]



Function:

Byte addressing of the Data Text Array allows 4096 bytes to be used: UMEMB(0) to UMEMB(4095).

Parameter	Input	Range	Description
Index	num. expr.	0 - 4095	Location of the byte

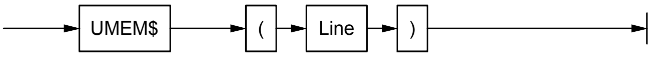
Example: UMEMB(SP) = SPEED
SP = 8 and SPEED = 123 the value of 123 is assigned to byte 8.

Note:

The UMEMB statement allows direct access to Data Text Arrays stored in the ZPRAM (drive R) possible.
Therefore, the filename of the Data Text Array has to be added to the statement.
The name must be placed between quotation marks or can be represented by a string.
The extension .DAT has to be left out.

Example: PRINT UMEMB "DATEN" (5)
UMEMB ;$(8) (33) = 10
Z = UMEMB "CHAR_S" (4002)

24.9 UMEM$(Line) [¦$(Line)]



Function:

This statement is used to read / write strings to / from the Data Text Array. All string commands are possible.
The maximum string length is defined by the DTFLLEN statement.

Parameter	Input	Range	Description
Line	num. expr.	1-4000	Line number

Example: UMEM$(***y***)=*String*
UMEM$(Y) = $(8)

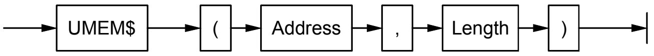
Y = 15 and $(8) = "CUSTOMER" stores the string "CUSTOMER" in the first 8 bytes of line 15. Other characters (bytes) of this line remain unchanged.

Note:

The UMEM$ statement allows direct access to Data Text Arrays stored in the ZPRAM (drive R) possible.
Therefore, the filename of the Data Text Array has to be added to the statement.
The name must be placed between quotation marks or can be represented by a string.
The extension .DAT has to e left out.

Example: DISP UMEM$ "TEXT" (7)
UMEM$;$(13) (33) = "HELLO"

24.10 UMEM$(Address, Length) [¦$(Address, Length)]



Function:

This statement allows free format storage of strings in the Data Text Array. The line length is variable for each string.
All string commands are possible.
The first location of the string and the string length is defined with UMEM$(address, length).

Example:
UMEM$(***a***,***l***)=*String*

Parameter	Input	Range	Description
Address	num. expr.	0 - 4095	Location in the Data Text Array
Length	num. expr.	1 - 255	Length of the string

Example: DISP UMEM$(10,4)
UMEM$(A,L)=$(4)

A = 30, L = 20 and $(4) = "COMTAC 2000". 20 bytes are reserved with L=20. $(4) is stored beginning at location 30. $(4) occupies location 30 to 40. The other bytes 41 to 50 remain unchanged.

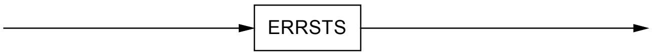
Note:

The UMEM$ statement allows direct access to Data Text Arrays stored in the ZPRAM (drive R) possible.
Therefore, the filename of the Data Text Array has to be added to the statement.
The name must be placed between quotation marks or can be represented by a string.
The extension .DAT has to be left out.

Example: DISP UMEM$ "TXT001" (0,10)
UMEM$;$(25) (10,10) = "PROCESS"

25. Error Handling

25.1 ERRSTS [ES.]



Function:

System variable which stores the last displayed error number.

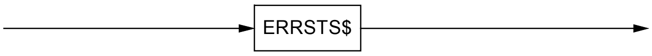
A description of the error numbers can be found on the next page.

Example: PRINT ERRSTS
A = ERRSTS

Note:

ERRSTS can only be read.

25.2 ERRSTS$



Function:

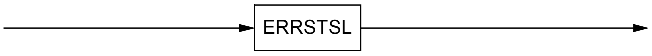
This system variable stores the text of the last displayed error.

Example: PRINT ERRSTS$
OUTPUT 2,4;ERRSTS$

Note:

ERRSTS$ can only be used in combination with the PRINT, DISP, or OUTPUT statement.

25.3 ERRSTSL



Function:

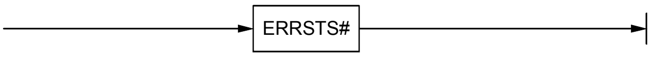
System variable which stores the line number of the BASIC program in which the error occurred.

Example: PRINT ERRSTSL
A = ERRSTSL

Note:

ERRSTSL can only be read.

25.4 ERRSTS#



Function:

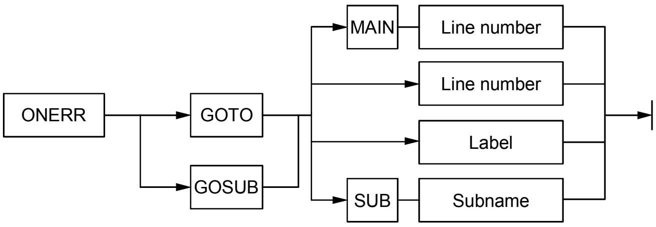
System variable which stores the name of the subroutine in which the error occurred.

Example: PRINT ERRSTS#
A = ERRSTS#

Note:

ERRSTS# can only be read.

25.5 ONERR [OE.]



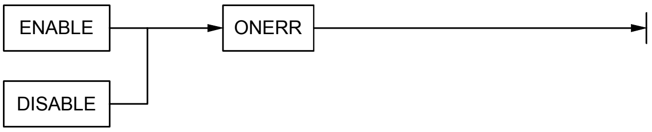
Function:

Defines a program branch. The branch is executed if an error occurs and the interrupt ONERR is enabled. In this case the BASIC program will keep running. The program will stop if the ONERR interrupt is disabled.

Parameter	Input	Range	Description
Liner No.	Zahl	0 - 65535	Target line *No.*

Example: ONERR GOTO 200
ONERR GOSUB 700

25.6 ENABLE [EN.]/DISABLE [DI.] ONERR [OE.]



Function:

Enable / disable the ONERR Interrupt.

Example: ENABLE ONERR
DISABLE ONERR

25.7 Error Description

1 **Bad Syntax**
Incorrect syntax.

2 **Renumber not possible**
The renumbering can´t be executed.

3 **Copy not possible**
The specified program lines cannot be copied.

6 **No Data**
A READ statement without a DATA statement was executed, or all data was read and no RESTORE statement followed.

7 **Can't continue**
The program cannot continue following an error.

8 **Memory Allocation**
Not enough program or data memory.

9 **A-Stack**
Too many PUSHes executed or POP without a preceding PUSCH.

10 **C-Stack**
this error occurs when:
- ◆RETURN without GOSUB
- ◆ WHILE- or UNTIL-statement without a preceding DO.
- ◆NEXT-Statement without a preceding FOR.
- ◆the C-stack has no more free memory.

The C-stack can store 254 bytes.
Each FOR...NEXT-loop needs 18 Byte, each DO...WHILE-, DO...UNTIL- 4 Byte. Each GOSUB-statement needs 4 Byte.
Therefore maximum 14 FOR...NEXT loops can be nested.

11 **I-Stack**
Failure of the operating system.

12 **Array Size**
Access to an array which isn´t defined.

13 **Invalid Line Number**
Branch statement to a line which doesn´t exist.

14 **Transfer Parity.** Error while sending/receiving data to/from the floppy disk drive.

15 **Floppy-interface not defined.** Interface for the floppy disk drive not defined (Parameter 10).

16 **Floppy report**.Floppy disk drive reports an error.

18 **No F-Bus Host.** COMTAC is not defined to be fieldbus Host (Parameter 73).

19 **This is no COMPAX**
Field bus command to a device which is not a COMPAX.

20 **F_Bus address missing**
The device address used isn´t defined.

21 **F_Bus device alert**
The addressed field bus device reports an ALARM.

22 **F_Bus device missing**
The bus connection to the addressed field bus device is defective.

23 **COMTAC-Slave hardware failure**
Hardware failiure of the slave processor system of COMTAC

24 **FBUS timeout**
Addressed field bus device timeout.

25 **F_Bus device no data**
The addressed field bus device didn´t place the requested data to the bus.

26 **F_Bus device no command**
The addressed field bus device can´t execute the received command.

27 **F_Bus device error**
The addressed field bus device reports ERROR.

28 **Parenthesis missing**
A parenthesis is missing in the statement.

29 **Quotation mark missing**
A quotation mark is missing in the statement.

30 **Bad Argument**
Inadmissible index or number.

31 **Divide by Zero**
Denominator of a division = 0.

32 **Arith. underflow**
The result of an arithmetical operation is less than +/-1*10-127.

33 **Arith. overflow**
The result of an arithmetical operation is graeter than +/-.99999999*10127.

34 **File already exists.** Delete?
The file already exists (drive R, A or B).

35 **File not found**
The file doesn´t exist on drive R, A or B.

36 **Insufficient space**
Not enough memory space to store the file.

37 **No disk in drive**
No disk in the addressed floppy drive.

38 **Write protected**
The disk is write protected.

39 **Read/Write**
Read / Write error when accessing the disk.

40 **RS232 timeout**
RS232 programmed time, timeout.

41 **RS485/1 timeout**
RS485/1 programmed time, timeout.

42 **Not formated**
The disk is not formatted. Use the FORMAT instruction.

43 **No Data File**. Access with a Data Text Array statement to a file without the extension .DAT has been attempted.

44 **Floppy not present**
The floppy is already defined but not connected to COMTAC..

45 **RS232/1 timeout**
RS232/1 programmed time, timeout.

46 **PC Link timeout**.Error during file upload / download.

47 **Checksum Error in System CWs**
System parameter failure.

48 **Checksum Error in User CWs**
User parameter failure.

49 **Checksum Error in Program**
Program memory failure.

50 **Array already defined**
This error occurs if
- ◆An array is defined a second time (DIM statement).
- ◆A string is defined a second time (DIM statement).

51 **Array not found**
An attempt was made to delete a non-existent array in the ZPRAM.

52 **Output 1-8 is overloaded**. One of the outputs 1 to 8 is overloaded .

53 **Output 9-16 is overloaded**. One of the outputs 9 to 16 is overloaded.

54 **RS485/2 timeout.**

55 **RS232/3 timeout**.

56 **Jump address missing**

The target address for the specified interrupt routine doesn´t exist.

57 No connection. Error within the 3964(R) protocol (RS232/3 interface).

58 Invalid LABEL. The LABEL doesn´t exist.

59 Invalid SUB. The SUB doesn´t exist

60 Output 17-24 overloaded.

61 Output 25-32 overloaded.

6**2 Checksum Error in Data**. Error in a data file or array (Data Text Array). $(#0) contains the name of this file.

63 F-Bus device no response. The addressed field bus device does not release the received command.

64 Add a file not possible. The file to be added has been compiled, so it can´t be added to the existing file.

65 CPX Response negative, CPX default No. may be determined via ASC($(#1)).

Break Down of a Field Bus Device

The break down of a field bus device is reported immediately by the error: "F_Bus device fail".
The error number is the device address number + 100.

26. Field Bus Interface

26.1 Description

The RS485/1 is used for the field bus interface. The field bus protocol enhances the use of COMTAC in combination with COMPAX devices.

The field bus is controlled by the COMATC which acts as the Host. The other devices which are connected to the field bus are the Sub Devices. There can only be one Host in the system. The Sub Devices must be able to operate with the field bus protocol.

Parameter 74 adjusts the baudrate .

A maximum of 31 Sub Devices can be connected to the Host. Each Sub Device has its own address (1 - 99). Address 0 is reserved for the Host.

During power on, when COMTAC gets initialised, the operating system searches for Sub Devices connected to the fieldbus. The detected Sub Device addresses are stored in the COMTAC. Thus COMTAC always knows the available devices. If a Sub Device is connected after power on, the interface has to be reset with the RESET#1 instruction.

Parameter 70 determines the maximium device address.

The data of each Sub Device listed below is stored in the COMTAC:

- Type of Sub Device (Classification),
- Sub Device address,
- number of bytes which are read from the Sub Device: input bytes
- number of bytes which are written to the Sub Device: output bytes.

After this initialisation the bus protocol starts the cyclic update. This means that all Sub Devices are polled and within a fixed time period the input and output bytes of all Sub Devices are updated sequentially device by device. This procedure is repeated periodically.

The time needed for this update depends on the number of Sub Devices. The cycle is determined by the baud rate. If the update of all Sub Devices takes longer than the time period of the baud rate, update time is extended.

After this cyclic update other data can be transmitted from or to the COMATC, e.g OUTPUT, ENTER, FBUS+P, FBUS+D commands. This procedure is called non cyclic because it happens only at definite times when a command is executed.

COMTAC automatically repeats commands if there is a transmission error or the addressed device can´t accept commands at present.

Parameter 75 determines the number of allowed repetitions. After this an error occurs.

Parameter 76 determines the number of allowed transmission errors for one transmission. After this an error occurs.

26.2 Parameters of the Field Bus Interface

70	Maximum Sub Device address number. COMTAC checks all devices up to this number during power on.	10
71	Fieldbus interrupt mask Bit 0 = device timeout 0=off, 1=on Bit 1 = device event 0=off, 1=on Bit 2 = device error 0=off, 1=on Bit 3 = device data 0=off, 1=on Bit 4 = device write response 0=off, 1=on Bit 5 = disable error message 21 Bit 6 = disable error message 27 Bit 7 = disable auto device write response check	0
72	Field bus time period 0...255 (1 = 1ms) 0: determined by the baud rate	0
73	Field bus protocol 0: no field bus protocol 1: COMTAC is field bus Host	1
74	Field bus Baudrate 0=150 3=1200 6=9600 9=57600 1=300 4=2400 7=19200 10=172800 2=600 5=4800 8=28800 11=345600	11
75	Number of command repetitions 0...250 COMTAC automatically repeats commands if there is a transmission error or the addressed device can´t accept commands at present. After this an error occurs	10
76	Maximum number of allowed transmission errors.(Field bus Host). After this an error occurs.	10
77	Display of the field bus initialisation on the COMATC display. = 0 --> ON = 1 --> OFF	0
78	reserved	0
79	0: no time out check 1-255: time out 0,1 - 25,5 sec.	0

26.3 Cycle Times

The cycle time for the update of the Sub Devices depends on the baud rate:

Baudrate [KBaud]	Cycle time[msec]	time per character [µsec]
345.60	7	32
172.80	10	64
57.60	15	192
28.80	30	384

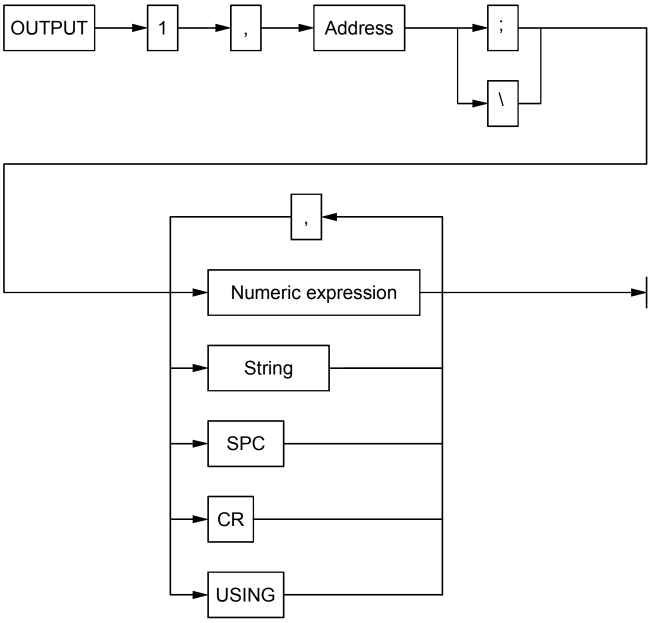
For each bus configuration consisting of the Host and the Sub Devices the required update time for all Sub Devices can be calculated:

Tcyc = ((5 * nSD + nIO) * Tchar) + (nSD * Toffs)
Tcyc = Required update time
nSD = number of Sub Devices
nIO = number of input and output bytes
Tchar = time per character(e.g. 32 µs # 345,6kBaud)
Toffs = offset time (= 100 µs)

COMPAX consists of 10 input bytes and 6 output bytes.
COMTAC can manage a maximum of 496 inout/output data bytes. This number corresponds to a maximum of 31 COMPAX devices on the field bus.
Cycle time for n COMPAXes at a baudrate of 345.6 kBaud:

Tcyc(n) = (((5 * n) + (16 * n)) * 32 µs) + (100 µs * n)
Tcyc(n) = 772 µs * n
Tcyc(1) = 0.772 msec
Tcyc(2) = 1.544 msec
Tcyc(3) = 2.316 msec
Tcyc(4) = 3.088 msec
Tcyc(5) = 3.860 msec
Tcyc(6) = 4.632 msec
Tcyc(31) = 23.93 msec

26.4 OUTPUT [O.](Fieldbus)



Function:

This statement reads out characters, strings, and numerical expressions.
It´s possible to receive a response ($(1) simultaneously with this statement by using the back slash instead of the semicolon in the OUTPUT statement..

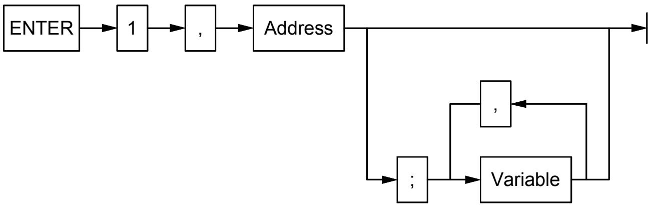
Parameter	Input	Range	Description
Address	num. expr.	0 - 255	Target device address

Example: OUTPUT 1,7;"ANGLE=",A
OUTPUT 1,255;"PA",XPOS,CHR$(13)

Note:

Address no. 255 is a broadcast address. With this all Sub Devices are addressed simultaneously.

26.5 ENTER [E.] (Fieldbus)



Function:

This statement copies a received string from the buffer to the string $(1).

Parameter	Input	Range	Description
Address	num. expr.	0 - 255	Sub Device address

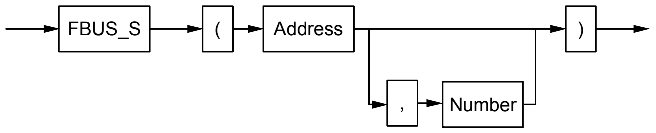
A string may contain letters and figures. An optional function of the input statement allows numbers in the string to be stored in BASIC variables.
The first series of numbers detected in the string is assigned to the first variable name of the ENTER statement, the second to the second variable and so on.
The number of variables do not have to agree with the number of numerical strings found in the received string

Example: Received string = "10V200X-30"
ENTER 1,address; A,B,C,D
After the execution of ENTER: $(#1) = "10V200X-30"
A = 10
B = 200
C = -30 and D has not changed

Note:

The status bit STG7 of the STATUS register of the Sub Device indicates that a string is available.

26.6 FBUS_S [FS.]



Function:

This system variable returns the status information of the addressed Sub Device.
If the index number is left out, the return value is TRUE (65535) if the addressed Sub Device exists or FALSE (0) if not.

Parameter	Input	Range	Description
Address	num. expr.	01 - 99	Sub Device address
Number	num. expr.	0 - 3	Status Index

Status-Index	Description
0	STG STATUS
1	STI number of input bytes
2	STO number of output bytes
3	TYPSub device type (classification)

Example: PRINT FBUS_S(4,2)
S = FBUS_S(3,0)

STG = STATUS of a field bus device

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
STG7	STG6	STG5	STG4	STG3	STG2	STG1	STG0
128	64	32	16	8	4	2	1

STG0: identifies the Sub Device (classification)
0: non cyclic type with cyclic status information
1: cyclic type with constant input / output bytes
STG1: no function
STG2: no function
STG3: no function
STG4: Quit flag for a received command
0: no command quit
1: quit a command
STG5: initalises an alarm
0: no alarm (Event)
1: alarm (Event) occurred
STG6: Field bus device failiure
0: no failiure
1: failiure or reduction of the functions of a field bus device
STG7: Data of the field bus device is available and can be read
0: no data available
1: data available

TYP = Byte which represents the classification of a field bus device

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TYP7	TYP6	TYP5	TYP4	TYP3	TYP2	TYP1	TYP0
128	64	32	16	8	4	2	1

Bit 5 - 7 are for global classification:

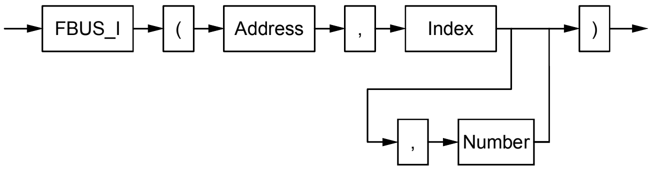
TYP7	TYP6	decribe the data exchange of a field bus device
0	0	device with only cyclic data exchange
0	1	device with only non cyclic data exchange
1	0	device with cyclic and non cyclic data exchange
1	1	device with a special protocol

TYP5	describes the data structure of a field bus device
0	byte structure
1	word structure

With this the following range types are defined:

1 - 31	device with byte structure, cyclic data exchange
33 - 63	device with word structure, cyclic data exchange
65 - 95	device with byte structure, non cyclic data exchange
97 - 127	device with word structure, non cyclic data exchange
129-159	device with byte structure, cyclic and non cyclic data exchange
161-191	device with word structure, cyclic and non cyclic data exchange
193-255	device with special features

26.7 FBUS_I [FI.]



Function:

With this statement the Host reads the cyclic input data of a Sub Device, specified by index and number.
Index points to the LSB (least siginificant byte).

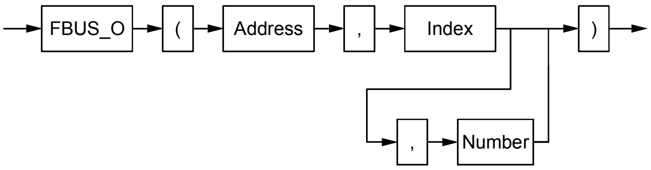
Parameter	Input	Range	Description
Address	num. expr.	01 - 99	Sub Device address
Index	num. expr.	1 - STI	points to the first byte to be read
Number	num. expr.	1,2 o. 4	Number of bytes to be read

Example: PRINT FBUS_I(7,1,4)
IF FBUS_I(1,0)@4 THEN ...

Note:

COMTAC always reads two bytes if no number of bytes is specified.

26.8 FBUS_O [FO.]



Function:

With this statement the Host writes the cyclic output data to a Sub Device, specified by index and number.
Index points to the LSB (least siginificant byte).

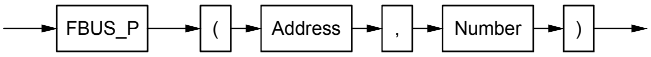
Parameter	Input	Range	Description
Address	num. expr.	01 - 99	Sub Device address
Index	num. expr.	1 - STO	points to the first byte to be written
Number	num. expr.	1,2 o. 4	Number of bytes to be written

Example: FBUS_O(3,1,1) = 01000100B
FBUS_O(5,2) = SOMETHING

Note:

COMTAC always writes two bytes if no number of bytes is specified.
The output data can be read back.
The broadcast address 255 writes the data to all connected devices.

26.9 FBUS_P [FP.]



Function:

Read or write a field bus parameter specified by address and number

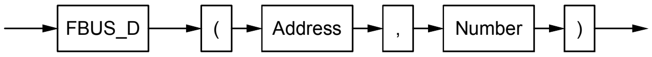
Parameter	Input	Range	Description
Address	num. expr.	01 - 99	Sub Device address
Number	num. expr.	1 - 255	Parameter number

Example: PRINT FBUS_P(70,2)
FBUS_P(33,10) = 3000

Note:

A parameter has a valid range of 0 - 65535.
The broadcast address can be used.

26.10 FBUS_D [FD.]



Function:

Read or write two field bus parameters specified by address, number and number+1.
The two parameters form one value with a valid range of -2 147 483 648 - +2 147 483 647.

- Parameter(n)stores the lower part
- Parameter(n+1) = stores the higher part and the sign of the value.

Parameter	Input	Range	Description
Address	num. expr.	01 - 99	Sub Device address
Number	num. expr.	1 - 255	Parameter number

Example: PRINT FBUS_D(55,128)
FBUS_D(13,24) = -250000

Note:

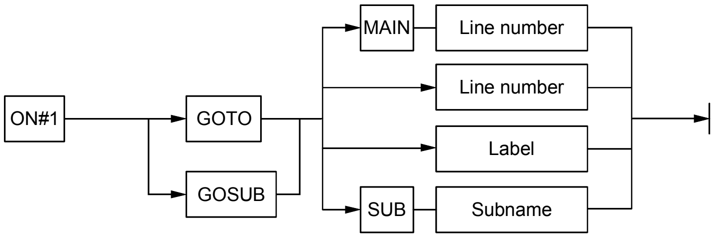
Remember that parameter(n) and parameter(n+1) will be changed.
The broadcast address can be used.
Assigning several parameters:
It´s possible to write several parameters when listing these parameters in the FBUS_D statement. The values to be written must be separated by a comma or semicolon.
The separators have the following meaning:

- comma - write to a parameter
- semicolon - write to parameter(n) and to parameter(n+1)

Example: FBUS_P(3,0)=18;100000,10,555
After the execution of this statement:

Parameter 0 = 18	Parameter 1,2 = 100000
Parameter 3 = 10	Parameter 4 = 555.

26.11 On#1



Function:

Defines a program branch if one of the following conditions are reported from a field bus device:

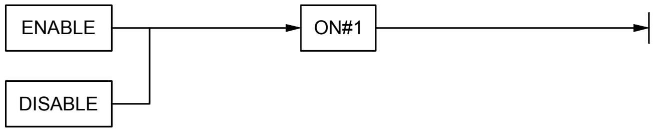
- Timeout	device break down or disconnected
- Alarm (Event)	service request of a device
- Error	device failiure
- Data	requested data is available
- Quit	the command quit can be read

Parameter	Input	Range	Description
Line no.	Number	0 - 65535	Target line number

Example: ON#1 GOTO ...
ON#1 GOSUB ...

Note: Parameter 71 enables / disables the interrupts

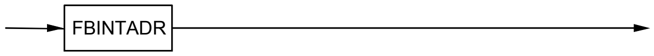
26.12 ENABLE [EN.]/DISABLE [DI.] Field Bus



Function:

Enable / disable the field bus interrupt.
Example: ENABLE ON#1
DISABLE ON#1

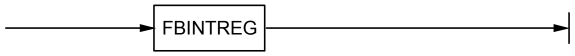
26.13 FBINTADR



Function:

This system variable returns the address of the field bus device which released the field bus interrupt.
Example: ADR = FBINTADR
DISP FBINTADR

26.14 FBINTREG



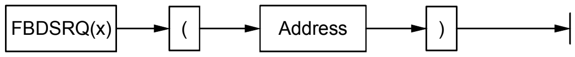
Function:

This system variable returns a number which describes the reason for the ON#1 interrupt.

0 = Timeout
1 = Alarm (Event)
2 = Error
3 = Data
4 = Quit

Example: ON FBINTREG GOSUB ....
IF FBINTREG = 0 THEN ...

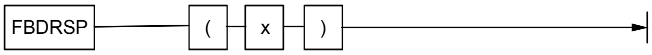
26.15 FBDSRQ(x)



Function:

Read the alarm (Event) data of a field bus device. The response is stored in $(#1).
Example: FBDSRQ(2)

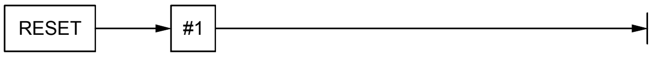
26.16 FBDRSP(x)



Function:

Read the quit data of a field bus device. The response is stored in $(#1)[1].
ASC($(#1)) = 0 --> command executed
ASC($(#1)) <> 0 -->command not executed. See error handling.
Example: FBDRSP(2)

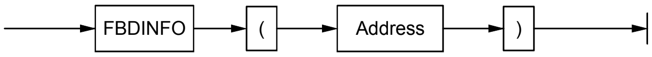
26.17 RESET [RS.] Fieldbus



Function:

Resets the field bus interface to the default values and starts polling the Sub Devices.
Example: RESET #1

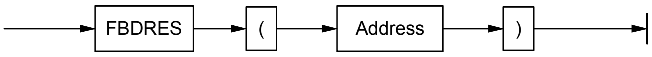
26.18 FBDINFO(x)



Function:

Read an info string from a field bus device The response is stored in $(#1).
Example: FBDINFO(2)

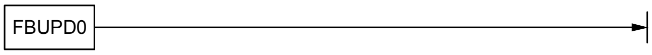
26.19 FBDRES(x)



Function:

Reset the Sub Devices.
Example: FBDRES(4)

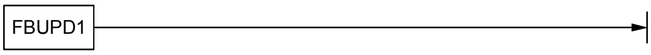
26.20 FBUPD0



Function:

Switch off the cyclic update of the Host (COMTAC).
Example: FBUPD0

26.21 FBUPD1



Function:

Switch on the cyclic update of the Host (COMTAC).
Example: FBUPD1

Reaction to negative command acknowledgement by COMPAX

If a fieldbus-command to a COMPAX device is negatively acknowledged as a result of an error, (e.g. E55), the operating system will check if an ONCPXERR-Interrupt is already due. In this case the BASIC-Text indicator moves to the beginning of the fieldbus-command and branches out to:

- ONCPXERR-Interrupt routine (if the selected interrupt priorities allow)
- any other interrupt routine with a higher priority due

After returning from the CONCPXERR-interrupt subroutine (where the COMPAX-error was eliminated), the fieldbus-command will be repeated automatically.

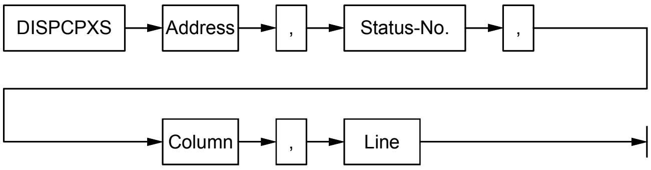
If no ONCPXERR-interrupt is due (e.g. not enabled) the fieldbus-command will be repeated x times according to the value in P75.
Afterwards the BASIC text indicator will also move to the beginning of the fieldbus-command and the COMTAC fault management routine (Error No, 65) is initialised.
If the ONERR-interrupt is enabled, you may analyse the COMTAC-error Nos. (ERRSTS) as well as (ASC($#1))) and, where appropriate, rectify the COMPAX-error.
After return from the ONERR-interrupt subroutine, the fieldbus-command will now be repeated automatically.

27. COMPAX Commands

These COMPAX commands can only be used with field bus protocol.

All strings used with these COMAPX commands have to be written in capital lettes.(Example: OUPUT 1,1;"POSA100")
Please note the comment on "negative command acknowledgement" on page 94!

27.1 DISPCPXS



Function:

Read COMPAX status once . The value of the specified status and corresponding text (optional) is displayed on the COMTAC display.

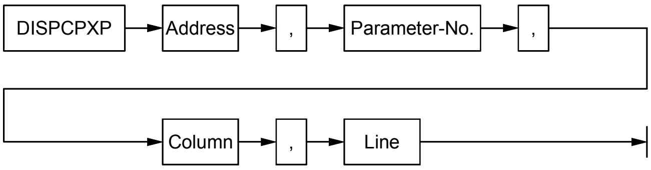
Parameter	Input	Range	Description
Address	num. expr.	1-99	Address of COMPAX device
S-No.	num. expr.	1-250	COMPAX status number
Column	num. expr.	0-40	column
Line	num. expr.	1-4	line

Example: DISPCPXS 1,1,1,1
DISPCPXS 5,1,0,20

Note:

The optional text is displayed if the column value is 0.

27.2 DISPCPXP



Function:

Read a COMPAX parameter once The value of the specified parameter and corresponding text (optional) is displayed on the COMTAC display.

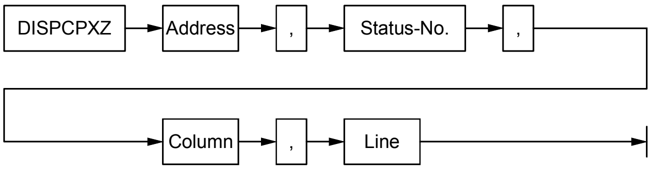
Parameter	Input	Range	Description
Address	num. expr.	1-99	Address of COMPAX device
Parameter-No.	num. expr.	1-250	COMPAX parameter No.
Column	num. expr.	0-40	column
Line	num. expr.	1-4	line

Example: DISPCPXP 31,4,0,1
DISPCPXP 10,100,X,Y

Note:

The optional text is displayed if the column value is 0.

27.3 DISPCPXZ



Function:

This statement starts the cylic reading of the specified COMPAX status. The status is updated automatically.

This type of display can be stopped with the STOP DISP statement and continued with the CONT DISP statement. (see page 51)

Parameter	Input	Range	Description
Address	num. expr.	1 - 99	Address of COMPAX device
Status no.	num. expr.	0...	COMPAX status*no.* The display is finished with status *no.* = 0
Column	num. expr.	0...40	column 0: Display of an optional text >0: Only the value is displayed.
Line	num. expr.	1...4	Display-Line

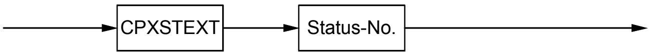
Example: DISPCPXZ 1,1,1,3 Start a cyclic display
DISPCPXZ 1,0,1,3 Stop the cyclic display

Note:

The optional text is displayed if the column value is 0.

The cyclic display can be terminated by executing the same statement but with status number = 0. A maximum of 9 values (including the DISPVAR and EDITVAR functions) can be displayed.

27.4 CPXSTEXT



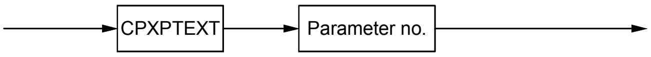
Function:

This function is used together with the DISP statement in order to display the text of a COMPAX status discription .

Parameter	Input	Range	Description
Statusno.	num. expr.	1...250	COMPAX status *no.*

Example: DISP CPXSTEXT 5
DISP TABXY(10,3),CPXSTEXT 1

27.5 CPXPTEXT



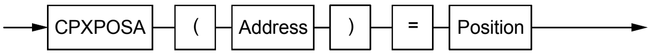
Function:

This function is used together with the DISP statement in order to display the text of a COMPAX parameter description .

Parameter	Input	Range	Description
Parameter number	num. expr.	1...250	COMPAX parameter number

Example: DISP CPXPTEXT 94
DISP TABXY(10,3),CPXPTEXT 100

27.6 CPXPOSA [PA.]



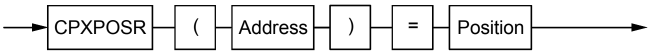
FFunction:

Absolute position command to COMPAX.

Parameter	Input	Range	Description
Address	num. expr.	1 - 99	Address of COMPAX device
Value	num. expr.		target position

Example: CPXPOSA(5) = 1000

27.7 CPXPOSR [PR.]



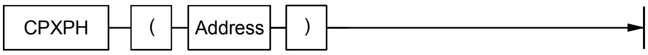
Function:

Relative position command to COMPAX.

Parameter	Input	Range	Description
Address	num. expr.	1 - 99	Address of COMPAX device
Value	num. expr.		target position

Example: CPXPOSR(AX) = 500

27.8 CPXPH [ZP.]



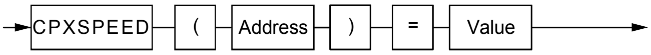
Function:

Command COMPAX to search for the home position.

Parameter	Input	Range	Description
Address	num. expr.	1 - 99	Address of COMPAX device

Example: CPXPH(7)

27.9 CPXSPEED [SD.]



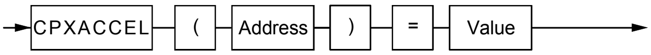
Function:

Speed command to COMPAX.

Parameter	Input	Range	Description
Addresse	num. expr.	1 - 99	Address of COMPAX device
Value	num. expr.		Speed

Example: CPXSPEED(1) = 80

27.10 CPXACCEL [AL.]



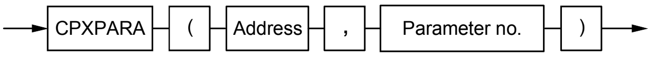
Function:

ACCEL command to COMPAX.

Parameter	Input	Range	Description
Address	num. expr.	1 - 99	Address of COMPAX device
Value	num. expr.		Accelleration

Example: CPXACCEL(8) = 2500

27.11 CPXPARA [PAR.]



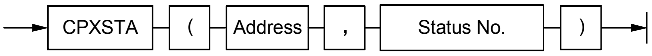
Function:

Read or write a COMPAX parameter.

Parameter	Input	Range	Description
Address	num. expr.	1 - 99	Address of COMPAX device
Parameter number.	num. expr.	1...250	COMPAX paramter number

Example: A = CPXPARA (1,1)

27.12 CPXSTA [STA.]



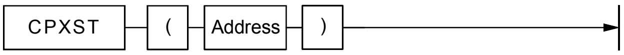
Function:

Read a COMPAX status.

Parameter	Input	Range	Description
Address	num. expr.	1 - 99	Address of COMPAX device
Statusno.	num. expr.	1...250	COMPAX statue no.

Example: A = CPXSTA (1,1)

27.13 CPXST [ST.]



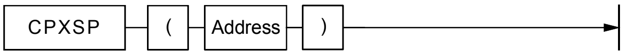
Function:

Start a previous stopped COMPAX movement or start the COMPAX line interpreter.

Parameter	Input	Range	Description
Address	num. expr.	1 - 99	Address of COMPAX device

Example: CPXST(8)

27.14 CPXSP [SP.]



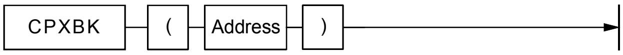
Function:

Stop a COMPAX motion or the line interpreter.

Parameter	Input	Range	Description
Address	num. expr.	1 - 99	Address of COMPAX device

Example: CPXSP(1)

27.15 CPXBK [BK.]



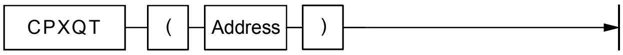
Function:

BREAK command to COMPAX. Finishes a command which the COMPAX is currently executing.

Parameter	Input	Range	Description
Address	num. expr.	1 - 99	Address of COMPAX device

Example: CPXBK(10)

27.16 CPXQT [QT.]



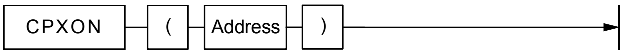
Function:

Quit a COMPAX error.

Parameter	Input	Range	Description
Address	num. expr.	1 - 99	Address of COMPAX device

Example: CPXQT(9)

27.17 CPXON [P1.]



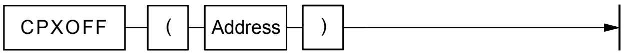
Function:

Switch COMPAX on, motor brake not active.

Parameter	Input	Range	Description
Address	num. expr.	1 - 99	Address of COMPAX device

Example: CPXON(1)

27.18 CPXOFF [P0.]



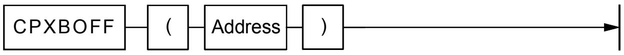
Function:

Switch off COMPAX, motor brake not active.

Parameter	Input	Range	Description
Address	num. expr.	1 - 99	Address of COMPAX device

Example: CPXOFF(6)

27.19 CPXBOFF [P0B.]



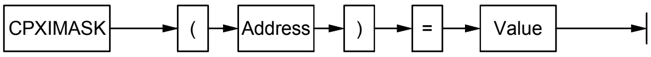
Function:

Switch off COMAPX, motor brake active.

Parameter	Input	Range	Description
Address	num. expr.	1 - 99	Address of COMPAX device

Example: CPXBOFF(7)

27.20 CPXIMASK [CIM.]



Function:

Enable function for the COMPAX digital inputs I1 to I6. Normally COMPAX uses these 6 inputs for fixed functions. These COMAPX hardware input functions can be replaced by setting the inputs via the COMTAC field bus. In this case the COMPAX hardware inputs can be used for other functions. The CPXCTR command activates the fixed input functions.

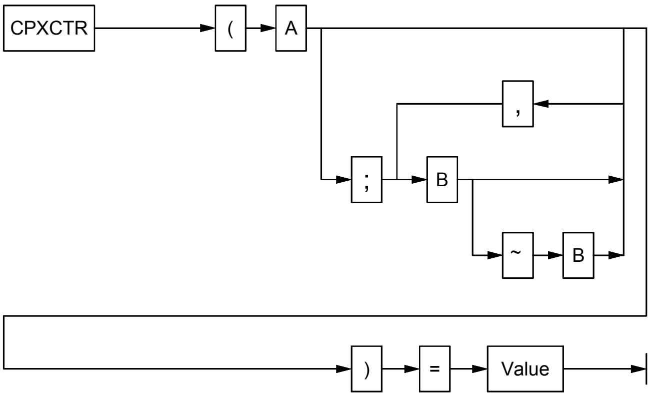
Bit 0=1 enables I1, Bit 1=1 enables I2, ... Bit 6=1 enables I6.

Parameter	Input	Range	Description
Address	num. expr.	0...99	Address of COMPAX device
alue	num. expr.	0..65535	assigned value

Example: CPXIMASK (12) = 0003H

In this example Input I1 and input I2 of the COMPAX with the device address 12 are able to be set by COMTAC.

27.21 CPXCTR [CC.]



Function:

This statement activates COMPAX digital input functions (from COMPAX Software version V2.10 on the virtual COMPAX-input functions) via the field bus (COMPAX control word).The access to these functions has to be enabled with the CPXIMASK command.

The bit number for the least significant bit is 1 in this case (not 0 !)

The bits of the COMPAX control word correspond to the hardware input functions of COMPAX.

- The decimal value to be read out is transformed into the binary format.
- The last named input of the CPXCTR statement gets the siginficance of 2^0 , the next one 2^1 and so on.
- If there is no bit number entered, the value is assigned to the complete word (Bit 16 = MSB, Bit 1 = LSB). Example: CPXCTR(1)=3→E17=E18="1"
- The number of inputs is limited to 16.
- A range of inputs is separated by the ~ .
- Single input numbers are separated by a comma.

Parameter	Input	Range	Description
A	num. expr.	0...99	Address of COMPAX device
B	num. expr.	1...16	Bit-Number Virtual input functions 17-32
Value	num. expr.	0..65535	assigned value

Example 1: CPXCTR(1;3~1)=3

Bit 1 to bit 3 of COMPAX with the device address 1 gets the value of 3 (E17=E18=1)

The allocation of the different binary places to the fiven bits is made as follows:

Binary places	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Significance	128	64	32	16	8	4	2	1
Bit number						3	2	1
Value of 3 in the binary format						0	1	1

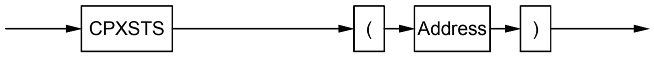
Example 2: CPXCTR(5)=2

Bit 2 of COMPAX with the device address 5 is set. All other bits are reset, because no bit number was entered.

Note:

The COMAPX control word can also be read:
A = CPXCTR(3).

27.22 CPXSTS [CS.]



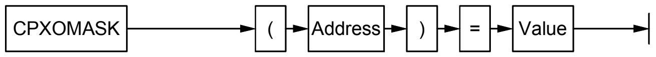
Function:

Read the status of the COMPAX outputs O1 to O16.

Parameter	Input	Range	Description
Address	num. expr.	0...99	Address of COMPAX device

Example: DISP CPXSTS(4)
IF CPXSTS(1)@5 THEN ...

27.23 CPXOMASK [COM.]



Function:

Enable function for the COMPAX digital outpus O1 to O6. Normally COMPAX uses these 6 outputs for fixed functions. If the access to these outputs is enabled for COMTAC they can be set by the CPXOUT command via the field bus. COMPAX itself then doesn´t have access to these outputs. Bit 0=1 enables I1, Bit 1=1 enables I2, ... Bit 6=1 enables I6.

Parameter	Input	Range	Description
Address	num. expr.	0...99	Address of COMPAX device
Value	num. expr.	0..65535	assigned value

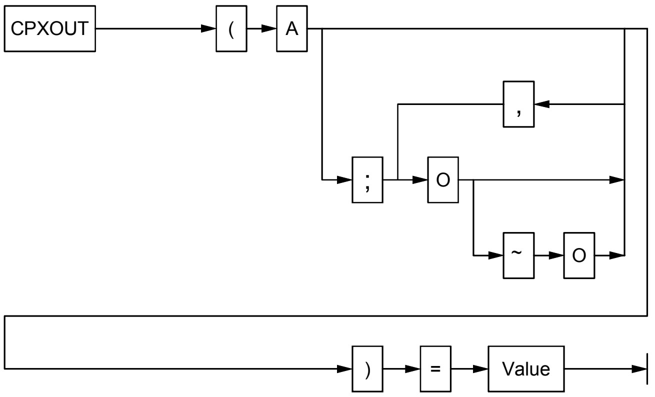
Example: CPXOMASK(15) = 0007h

In this example the outputs 1 to 3 are enabled for COMTAC access.

Note:

The value of CPXOUT can´t be read.

27.24 CPXOUT [CO.]



Function:

This statement sets the COMPAX outputs one output or a group of outputs. This function has to be enabled with the CPXOMASK command.
The decimal value to be read out is transformed into the binary format.
The binary value is assigned to the COMPAX outputs.
The last named output of the CPXOUT statement gets the siginficance of 2^0 , the next one 2^1 and so on.
The number of outputs which are set or reset depends on the value.
If there is no bit number entered, the value is assigned to the complete word (Bit 16 = MSB, Bit 1 = LSB).
The number of inputs is limited to 16.
A range of inputs is separated by the ~ .
Single input numbers are separated by a comma.

Parameter	Input	Range	Description
Address	num. expr.	0...99	Address of COMPAX device
O	num. expr.	1..16	Output number
Value	num. expr.	0..65535	assigned value

Example 1: CPXOUT(1;8~1)=11

The value of 11 is assigned to the outputs 1 to 8 of the COMPAX with the device address 1.

Binary places	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Significance	128	64	32	16	8	4	2	1
Output numbers	8	7	6	5	4	3	2	1
Value of 11 in the binary format	0	0	0	0	1	0	1	1

Example 2: CPXOUT(5;1~3,8,10,15)=60

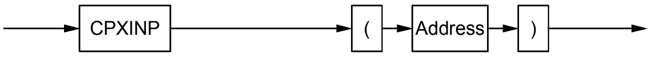
The value of 60 is assigned to the outputs 1, 2, 3 8,10 and 15 of the COMPAX with the device address 5.

Binary places	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Significance	128	64	32	16	8	4	2	1
Output numbers	-	-	1	2	3	8	10	15
Value of 60 in the binary format	0	0	1	1	1	1	0	0

Note:

CPXOUT can also be read. All 16 outputs are read:
A=CPXOUT(2).

27.25 CPXINP [CI.]



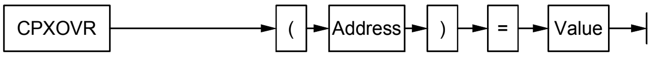
Function:

Read the COMPAX inputs I16 to I1. Input I16 is MSB, input I1 is LSB.

Parameter	Input	Range	Description
Address	num. expr.	0...99	Address of COMPAX device

Example: DISP CPXINP(8)
IF CPXINP(2)@3 THEN ...

27.26 CPXOVR [CV.]



Function:

Override value for the velocity of the COMPAX..

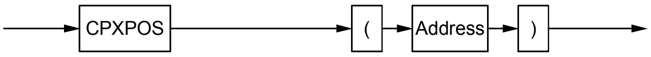
Parameter	Input	Range	Description
Address	num. expr.	0...99	Address of COMPAX device
Value	num. expr.	0..255	Assigned value

Example: CPXOVR (7) = 100

Note:

CPXOVR also can be read.

27.27 CPXPOS [CP.]



Function:

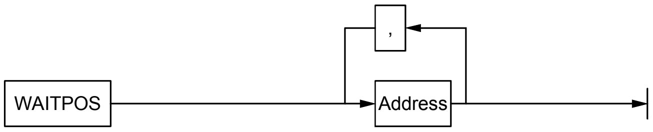
Read the actual position of COMPAX..

Parameter	Input	Range	Description
Address	num. expr.	0...99	Address of COMPAX device

Example: DISP CPXPOS(8)

27.28 WAITPOS

27.28.1 Wait for target Position reached"



Function:

Wait for ´Target Position Reached´ of the specified COMPAX.
The WAITPOS command is finished when all specified COMPAXEs have reached the target position.

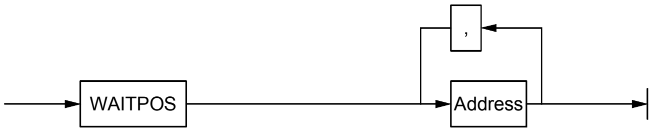
Parameter	Input	Range	Description
Address	num. expr.	0...99	Address of COMPAX device

Example: WAITPOS 1,2
WAITPOS A1,A2,A3

Note:

WAITPOS can be interrupted by an interrupt program.
After the return from the interrupt program the WAITPOS command is continued.

27.28.2 Query "Target Position reached"



Function:

Query the ´Position reached´status of COMPAX.

The system variable WAITPOS returns a TRUE (=65535) if all specified COMPAXes have reached their target position. The return value is FALSE (=0) if one or more of the specified COMPAXes are still moving.

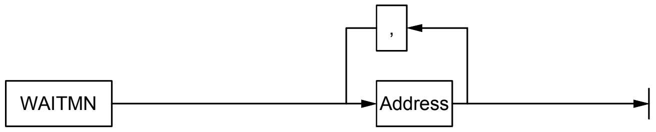
Parameter	Input	Range	Description
Address	num. expr.	0...99	Address of COMPAX device

Example: IF WAITPOS 1,2,3 THEN ....

```
DO
...
...
UNTIL WAITPOS A1,A2,A3,A4
```

27.29 WAITMN

27.29.1 Wait for "Home Position found"



Function:

Wait for ´Home Position found of the specified compax devices.
The WAITMN command is finished when all specified COMPAXes have found their home position.

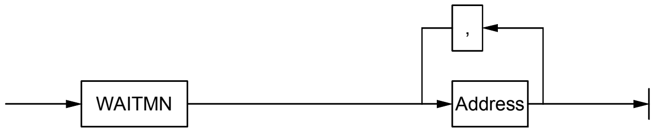
Parameter	Input	Range	Description
Address	num. expr.	0...99	Address of COMPAX device

Example: WAITMN 1,2
WAITMN A1,A2,A3

Note:

WAITMN can be interrupted by an interrupt program.
After the return from the interrupt program the WAITMN command is continued.

27.29.2 Query for "Home Position found"



Function:

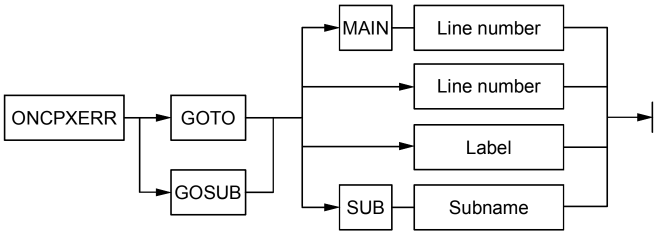
Query of the ´Home Position found´status of COMPAX.

The system variable WAITMN returns a TRUE (=65535) if all specified COMPAXes have found their home position. The return value is FALSE (=0) if one or more of the specified COMPAXes are still moving.

Parameter	Input	Range	Description
Address	num. expr.	0...99	Address of COMPAX device

Example: IF WAITMN 1,2,3 THEN ....
DO
...
...
UNTIL WAITMN A1,A2,A3,A4

27.30 ONCPXERR



Function:

Defines a program branch if a COMPAX reports an error and the interrupt for the COMPAX error is enabled.

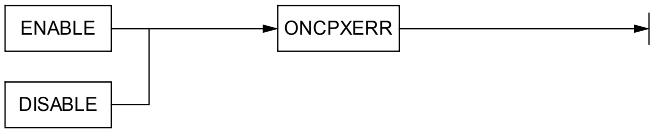
Attention:

This interrupt request has to be reset by reading the device address of the COMPAX which generated the interrupt: A = CPXERRADR.
Otherwise another COMPAX error interrupt isn´t recognised.

Parameter	Input	Range	Description
Line number	number	0..65535	target line number

Example: ONCPXERR GOTO ...
ONCPXERR GOSUB ...

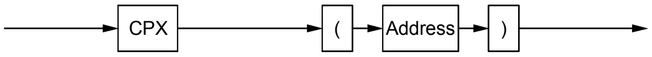
27.31 DISABLE [DI.] / ENABLE [EN.] ONCPXERR



Function:

Enable / disable the COMPAX error interrupt.
Example: ENABLE ONCPXERR
DISABLE ONCPXERR

27.32 CPX



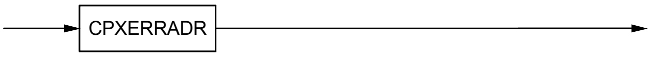
Function:

System variable to check the COMPAX devices connected to the field bus. The return value of the specified COMPAX is TRUE (=65535) if it´s available on the bus. If not the return value is FALSE (=0).

Parameter	Input	Range	Description
Address	num. expr.	1...99	Feldbus Address of COMPAX device

Example: IF CPX(3) THEN ...
DISP CPX(5)

27.33 CPXERRADR



Function:

System value. It returns the device address of the COMPAX which caused the COMAPX error interrupt.

Example: IF CPXERRADR=5 THEN ...
DISP CPXERRADR

28. Programing Example

Functions of the program:

- COMPAX moves to the home position
- Execute a position command.
- Cyclic display of a COMPAX status information at the LCD.

Tasks:

- Execute the movement to the home position. Wait till the home position is found
- Output a position command to COMPAX and display the actual position.
- Wait till the target position is reached..

Program

```
100  CLEAR D .........................................................................! Clear display
110  A=1 ..................................................................................! A = Device address of COMPAX
130  DISP TABXY(5,1),"Move to the home position!"
140  CPXPH(A) .......................................................................! Home position command to COMPAX
150  WAIT 100 .........................................................................! Wait for execution of COMPAX
160  IF WAITMN(A) THEN 170 ELSE 160 ...............................! Query of ´Home position reached´
170  OUTPUT 1,A;"SD10" ......................................................! SPEED = 10%
180  CLEAR D
190  DISP TABXY(1,2),"Actual position: "                         ! String output to the COMTAC display
200  DISP TABXY(1,3),"Speed: "
220  INPKBD "Please enter the target position!",POS              ! Asks the user to enter the target position
230  OUTPUT 1,A;"PA",POS                                        ! Position command to COMPAX
240  WAIT 100 ..........................................................................! Wait for execution of COMPAX
250  DISP TABXY(18,2),CPXPOS(A) ......................................! Display the actual position
260  OUTPUT 1,A;"S4" ............................................................! Request the actual speed from COMPAX
280  ENTER 1,A ......................................................................! Read the actual speed
290  DISP TABXY(18,3),$(#1)[6] .............................................! Display the actual speed
300  WAITPOS(A) ....................................................................! Wait till COMPAX has reached the target postion
310  DISP TABXY(18,2),CPXPOS(A) ......................................! Display the actual position
320  OUTPUT 1,A;"S4" ............................................................! Request the actual speed from COMPAX
340  ENTER 1,A ......................................................................! Read the actual speed
350  DISP TABXY(18,3),$(#1)[6] .............................................! Display the actual speed
360  WAIT 1000
370  DISP TABXY(10,4),"Target Position reached!"
380  WAIT 1000
390  CLEAR D,4
400  CLEAR D,1
410  GOTO 220
```

29. Parameter, sorted by number

Parameter Overview

No.	Usage
00...09	COMTAC System parameters
10...19	COMTAC - System parameters
20...29	reserved
30...39	reserved
40...49	RS232/3 - Interface parameters
50...59	RS232/1 - Interface parameters
60...69	RS485/1 - Interface parameters
70...79	RS485/1 - Interface parameters (field bus)
80...89	RS485/2 - Interface parameters
90...99	RS232/2 - Interface parameters
100	Assigns an interface to the terminal functions
101...200	user specific

System Parameters

No.	Function	Default
00	COMTAC-Type; set with power on	2000/ 3000
01	Date of the operating system software	-
02	Terminal-Typ 0 = TV 905 3 = VT100	0
03	Manufacturing number	-
04	Date of test	-
05	Power on *value* for O16 ... O1 After power on or CNTL R the digital outputs O16 .. O1 are set to *value:* BINOUT(16~1)=Parameter 5	0
06	Power on *value* for O32 ... O17 After power on or CNTL R the digital outputs O32 ... O17 are set to *value:* BINOUT(32~17)=Parameter 6	0
07	Power on delay 1 - 255 1=100ms	50
08	REQOUT Time-Out 1 - 255 1 = 100 ms	10
09	Repeat time for COMTAC-key board 1 - 255 1 = 100 ms	1
10	Assign an interface to the floppy drive 0 = RS232/1 is linked to the drive 1 = RS485/1 is linked to the drive 2 = RS232/2 is linked to the drive 3 = RS485/2 is linked to the drive 4 = RS232/3 is linked to the drive 99 = no floppy drive connected	99
11	Floppy-Timeout value 0 - 255 (1 = 0.1s) 0 = no timeout control	10
12	Floppy baudrate 4 = 2400 5 = 4800 6 = 9600 7 = 19200 8 = 38400	7
13	System-Flags	0
	Bit Function	
	0 **Parallel-Input** 0 = off, 1 = on (Terminal / Keyboard)	0
	1 **Listing-Option;** 0 = Normal, 1 =abbreviations	0
	2 **Keyboard autorepeat;** 0 = off, 1 = on	1
	3 **F8-Function** 0=Stop 1:disabled	0
	4 **Checksum-Data**	0
	0=off 1=on	
	5 0=auto 1=Com.	0
	6 **Checksum-User Param.** 0=off 1=on	0
	7 0=auto 1=Com.	0
	8 **Auto start function ZPRAM;** 0 = off, 1 = on	1
	9 **Auto start function diskette;** 0 = off, 1 = on	0
	10 -	0
	11 **Checksum: program and system parameter;** 0 = off, 1 = on	0
	12	0
	13 **Date output**; 0 =engl. 1 = ger.	0
	14	0
	15 **Ready beep;** 0 = off, 1 = on	0
14	Position of the time display in the COMTAC display (LCD). 0 = display disabled 1...160 = position of the first character	33
15	reserved	-
16	Position of the BASIC line display in the COMTAC display (LCD) for the DEBUG function 0 = display disabled 1...160 = position of the first character	0

Parameters 10 to 12 determine the functions of the floppy drive HFM2. All other adjustments for the drive are performed automatically by COMTAC.

The COMTAC parameters are set to the default values when the function key F12 is pressed during power on.

Parameters of the RS232 /3 Interface (#4) (Option F6)

No.	Function	Default
40	Device address 0 - 255	0
41	Start character of the received string 0 - 255 or Character delay time (with 3964 protocol) (1⇔50ms)	2
42	End character of the received string 0 - 255 or Quit delay time (with 3964 protocol) (1⇔50ms)	13
43	Receive protocol	34
	Bit Function	

No.	Function		Default
	0	protocol Bit 0	0
	1	protocol Bit 1	1
	2	protocol Bit 2	0
	3	-	0
	4	Block-Check-Character; 0=off, 1=on	0
	5	Auto-RTS; 0=off, 1=on	1
	6	Hardware-Handshake; 0 = off, 1 = on	0
	7	Software-Handshake; 0 = off, 1 = on	0
	The protocol bits define one of these receive protocols:		
	PBitno	INPUT-Rdy = 1,	
	2 1 0		0 1 0
	0 0 0	after each received character	
	0 0 1	after a number of received characters (P46)	
	0 1 0	after the end of string character (42)	
	0 1 1	after the start of string (P41) and end of string (P42) character	
	1 0 0	after the start of string (P41) and end of string (P42) character and the device address(P40)	
	1 0 1	3964 protocol	
44	Baudrate 0=150 3=1200 6=9600 9=57600 1=300 4=2400 7=19200 10=76800 2=600 5=4800 8=38400 11=115200		6
45	Parity / stopbits / character length		0
	Bit 0	number of Stopbits: 0 = 1 Stopbit 1 = 2 Stopbit	
	Bit 1	Parity Enable: 0 = Disable 1 = Enable	
	Bit 2/ Bit 3	Parity Select: 0/0 = Odd 1/0 = Even 0/1 = Mark 1/1 = Space	
	Bit 4/ Bit 5	character length: 0/0 = 8Bit 1/0 = 7Bit 0/1 = 6Bit 1/1 = 5Bit	
	Bit 6	no function	
	Bit 7	no function	
46	Number of characters per received string. 1 - 255		1
47	End character of the output string 0 - 255 repeat factor (with 3964 protocol)		13
48	Output protocol 0: the end character isn´t added automatically 1: CR is added automatically 2: CR LF is added automatically 3: (P497) is added automatically 4: CR LF and (P47) is added automatically 5: (P41) and (P42) is added automatically		1
49	Timeout value 0 - 255 (1 = 0.1s) 0 = no timeout check		0

Parameters of the RS232/1 interface (#0)

No.	Function		Default
50	Device address 0 - 255		0
51	Start character of the received string 0 - 255		2
52	End character of the received string 0 - 255 (must be 62 for COMPAX)		13
53	Receive protocol		34
	Bit	Function	
	0	protocol Bit 0	0
	1	protocol Bit 0	1
	2	protocol Bit 0	0
	3	-	0
	4	Block-Check-Character; 0=off, 1=on	0
	5	Auto-RTS; 0 = off, 1 = on	1
	6	Hardware-Handshake; 0 = off, 1 = on	0
	7	Software-Handshake; 0 = off, 1 = on	0
	The protocol bits define one of these receive protocols:		
	PBitno	INPUT-Rdy=1,	
	2 1 0		0 1 0
	0 0 0	after each received charater	
	0 0 1	after a number of received characters (P56)	
	0 1 0	after the end of string character (52)	
	0 1 1	after the start of string (P51) and end of string (P52) character	
	1 0 0	after the start of string (P51) and end of string (P52) character and the device address(P50)	
54	Baudrate 0=150 3=1200 6=9600 9=57600* 1=300 4=2400 7=19200 10=76800* 2=600 5=4800 8=38400 11=115200*		8
55	Parity and Stopbits 0 = without Parity, 1 Stopbit 1 = without Parity, 2 Stopbit 2 = with Parity EVEN, 1 Stopbit 3 = with Parity ODD, 1 Stopbit		0
56	Number of characters per received string. 1 - 255		1
57	End character of the output string 0 - 255		13
58	Output protocol 0: the end character isn´t added automatically 1: CR is added automatically 2: CR LF is added automatically 3: (P57) is added automatically 4: CR LF and (P57) is added automatically 5: (P51) and (P52) is added automatically		1
59	Time out value for the ENTER statement 0 - 255 (1 = 0.1sec) 0 = no time out check		0

* see page 78.

Parameters of the RS485/1 Interface (#1)

No.	Function		Default
60	Device address 0 - 255		0
61	Start character of the received string 0 - 255		2
62	End character of the received string 0 - 255		13
63	Receive protocol		2
	Bit	Function	
	0	Protocol Bit 0	0
	1	Protocol Bit 1	1
	2	Protocol Bit 2	0
	3		0
	4	Block-Check-Character; 0 = off, 1 = on	0
	5		1
	6		1
	7	Software handshake; 0 = off, 1 = on	0
	The protocol bits define one of these receive protocols:		
	PBitno	INPUT-Rdy = 1,	
	2 1 0		0 1 0
	0 0 0	after each received character	
	0 0 1	after a number of received characters (P66)	
	0 1 0	after the end of string character (P62)	
	0 1 1	after the start of string (P61) and end of string (P62) character	
	1 0 0	after the start of string (P61) and end of string (P62) character and the device address(P60)	
64	Baudrate 0=150 3=1200 6=9600 9=57600 1=300 4=2400 7=19200 10=172800 2=600 5=4800 8=28800 11=345600		6
65	Stopbits/Parity/Hardware Bit 0...2 = Parity/Stopbit selection 0 without parity, 1 stopbit 1 without parity, 2 stopbit 2 with parity Even, 1 stopbit 3 with parity Odd, 1 stopbit Bit 3...5 = reserved Bit 6 = 4 wire 0 Comtac=sub device:TxD = Pin 2, 7; RxD = Pin 1, 6 1 Comtac=Host:TxD = Pin 1, 6; RxD = Pin 2, 7 Bit 7 = 2/4 wire =0:2 wire =1:4 wire		0
66	Number of characters per received string. 1 - 255		1
67	End character of the output string 0 - 255		13
68	Output protocol 0: the end character isn´t added automatically 1: CR is added automatically 2: CR LF is added automatically 3: (P67) is added automatically 4: CR LF and (P67) is added automatically 5: (P61) and (P62) is added automatically		1

No.	Function	Default
69	Time out value for ENTER 1 statement 0 - 255 (1 = 0.1sec) 0 = no time out check	0

Parameters of the Field Bus Interface

No.	Function	Default
70	Maximum Sub Device address number. COMTAC checks all devices up to this number during power on.	10
71	Fieldbus interrupt mask Bit 0 = device timeout 0=off, 1=on Bit 1 = device event 0=off, 1=on Bit 2 = device error 0=off, 1=on Bit 3 = device data 0=off, 1=on Bit 4 = device write response 0=off, 1=on Bit 5 = disable error message 21 Bit 6 = disable error message 27 Bit 7 = disable auto device write response check	0
72	Field bus time period 0...255 (1 = 1ms) 0: determined by the baud rate	0
73	Field bus protocol 0: no field bus protocol 1: COMTAC is field bus Host	1
74	Field bus Baudrate 0=150 3=1200 6=9600 9=57600 1=300 4=2400 7=19200 10=172800 2=600 5=4800 8=28800 11=345600	11
75	Number of command repetitions 0...250 COMTAC automatically repeats commands if there is a transmission error or the addressed device can´t accept commands at present. After this an error occurs	10
76	Maximum number of allowed transmission errors.(Field bus Host). After this an error occurs.	10
77	Display of the field bus initialisation on the COMATC display. = 0 --> ON = 1 --> OFF	0
78	reserved	0
79	0: no time out check 1-255: time out 0,1 - 25,5 sec.	0

Parameters of the RS485/2 Interface (#3) (Option F6)

No.	Function	Default
80	Device address 0 - 255	0
81	Start character of the received string 0 - 255	2
82	End character of the received string 0 - 255 (must be 62 for COMPAX)	13

No.	Function		Default
83	Receive protocol		2
	Bit	Function	
	0	Protocol Bit 0	0
	1	Protocol Bit 1	1
	2	Protocol Bit 2	0
	3		0
	4	Block-Check-Character; 0 = off, 1 = on	0
	5		1
	6		1
	7	Software-Handshake; 0 = off, 1 = on	0
	The protocol bits define one of these receive protocols:		
	PBitno	INPUT-Rdy = 1,	
	2 1 0		0 1 0
	0 0 0	after each received charater	
	0 0 1	after a number of received characters (P86)	
	0 1 0	after the end of string character (P82)	
	0 1 1	after the start of string (P81) and end of string (P82) character	
	1 0 0	after the start of string (P81) and end of string (P82) character and the device address(P80)	
84	Baudrate 0=150 3=1200 6=9600 9=57600 1=300 4=2400 7=19200 10=76800 2=600 5=4800 8=38400 11=115200		6
85	Parity / stopbits / character length		0
	Bit 0	Number of stop bits: 0 = 1 stop bit 1 = 2 stop bit	
	Bit 1	Parity Enable: 0 = Disable 1 = Enable	
	Bit 2/ Bit 3	Parity Select: 0/0 = Odd 1/0 = Even 0/1 = Mark 1/1 = Space	
	Bit 4/ Bit 5	Character length: 0/0 = 8Bit 1/0 = 7Bit 0/1 = 6Bit 1/1 = 5Bit	
	Bit 6	reserved	
	Bit 7	2/4-wire 0 = 2-wire 1 = 4-wire	
86	Number of characters per received string. 1 - 255		1
87	End character of the output string 0 - 255		13
88	Output protocol 0: the end character isn´t added automatically 1: CR is added automatically 2: CR LF is added automatically 3: (P87) is added automatically 4: CR LF and (P87) is added automatically 5: (P81) and (P82) is added automatically		1
89	Time out value for the ENTER statement 0 - 255 (1 = 0.1sec) 0 = no time out check		0

Parameters of the RS232/2-Interface (#2)

No.	Function		Default
90	Device address 0 - 255		0
91	Start character of the received string 0 - 255		2
92	End character of the received string 0 - 255		13
93	Receive protocol		34
	Bit	Function	
	0	protocol Bit 0	0
	1	protocol Bit 0	1
	2	protocol Bit 0	0
	3	-	0
	4	Block-Check-Character; 0=off, 1=on	0
	5	Auto-RTS; 0 = off, 1 = on	1
	6	Hardware-Handshake; 0 = off, 1 = on	0
	7	Software-Handshake; 0 = off, 1 = on	0
	The protocol bits define one of these receive protocols:		
	PBitno	INPUT-Rdy=1,	
	2 1 0		0 1 0
	0 0 0	after each received charater	
	0 0 1	after a number of received characters (P96)	
	0 1 0	after the end of string character (92)	
	0 1 1	after the start of string (P91) and end of string (P92) character	
	1 0 0	after the start of string (P91) and end of string (P92) character and the device address(P90)	
94	Baudrate 3=1200 6=9600 9=57600* 4=2400 7=19200 10=76800* 5=4800 8=38400 11=115200*		8
95	Parity and Stopbits 0 = without Parity, 1 Stopbit 1 = without Parity, 2 Stopbit 2 = with Parity EVEN, 1 Stopbit 3 = with Parity ODD, 1 Stopbit		0
96	Number of characters per received string. 1 - 255		1
97	End character of the output string 0 - 255		13
98	Output protocol 0: the end character isn´t added automatically 1: CR is added automatically 2: CR LF is added automatically 3: (P97) is added automatically 4: CR LF and (P97) is added automatically 5: (P91) and (P92) is added automatically		1
99	Time out value for the ENTER statement 0 - 255 (1 = 0.1sec); 0 = no time out check		0
100	Defines which RS232 is used as terminal interface: 0 = RS232/1 1 = RS485/1 2 = RS232/2 3 = RS232/3		

* see page 78.

Parameters of the RS232/2-Interface (#2)

No.	Function		Default
90	Device address 0 - 255		0
91	Start character of the received string 0 - 255		2
92	End character of the received string 0 - 255		13
93	Receive protocol		34
	Bit	Function	
	0	protocol Bit 0	0
	1	protocol Bit 0	1
	2	protocol Bit 0	0
	3	-	0
	4	Block-Check-Character; 0=off, 1=on	0
	5	Auto-RTS; 0 = off, 1 = on	1
	6	Hardware-Handshake; 0 = off, 1 = on	0
	7	Software-Handshake; 0 = off, 1 = on	0
	The protocol bits define one of these receive protocols:		
	PBitno	INPUT-Rdy=1,	
	2 1 0		0 1 0
	0 0 0	after each received charater	
	0 0 1	after a number of received characters (P96)	
	0 1 0	after the end of string character (92)	
	0 1 1	after the start of string (P91) and end of string (P92) character	
	1 0 0	after the start of string (P91) and end of string (P92) character and the device address(P90)	
94	Baudrate 3 = 1200 5 = 4800 7 = 19200 4 = 2400 6 = 9600 8 = 38400		6
95	Parity and Stopbits 0 = without Parity, 1 Stopbit 1 = without Parity, 2 Stopbit 2 = with Parity EVEN, 1 Stopbit 3 = with Parity ODD, 1 Stopbit		0
96	Number of characters per received string. 1 - 255		1
97	End character of the output string 0 - 255		13
98	Output protocol 0: the end character isn´t added automatically 1: CR is added automatically 2: CR LF is added automatically 3: (P97) is added automatically 4: CR LF and (P97) is added automatically 5: (P91) and (P92) is added automatically		1
99	Time out value for the ENTER statement 0 - 255 (1 = 0.1sec) 0 = no time out check		0
100	Defines which RS232 is used as terminal interface: 0 = RS232/1 1 = RS485/1 2 = RS232/2 3 = RS232/3		

30. Index