2200
SDS-Extended BASIC-2

RELEASE 2.7

sds

SDS-Extended BASIC-2

Release 2.7

SOFTWARE BULLETIN

Multiuser Operating System
--
BASIC Language Extensions
for
Wang 2200 Series Processors

First Edition -- January, 1985

DISCLAIMER OF WARRANTIES AND LIMITATION OF LIABILITIES

The staff of Southern Data Systems, Inc., has taken due care in preparing this manual; however, nothing contained herein modifies or alters in any way the standard terms and conditions of the Southern Data Systems, Inc. license agreement by which this software was acquired, nor increases in any way Southern Data System's liability to the customer. In no event shall Southern Data Systems, Inc. be liable for incidental or consequential damages in connection with or arising from the use of this manual, the associated equipment, any software programs contained herein, or any related materials.

NOTICE

All Southern Data Systems, Inc., program products are licensed to customers in accordance with the terms and conditions of the Southern Data Systems, Inc. Software License Agreement. No ownership of Southern Data Systems, Inc. software is transferred and any use beyond the terms of the Software License Agreement, without the written authorization of Southern Data Systems, Inc. is prohibited.

PREFACE

This Software bulletin describes SDS Extended BASIC-2 Release 2.7, its installation procedures and the software enhancements to Release 2.5 of the 2200 BASIC-2 Multiuser Operating System.

Chapter 1 describes SDS-Extended BASIC-2 and the installation procedures for Release 2.7.

Chapter 2 describes the seven software enhancements in Release 2.7:

- a new ERR$ command
- a new =SELECT command
- changes to the HEXPRINT function
- changes to the IF END command
- a new, selective LIST DC command
- ability to Time and Date Stamp program files
- ability to display Descriptive Error Messages

This documentation is intended to be used in conjunction with the following manuals:

- Wang BASIC-2 Language Reference Manual
- Wang BASIC-2 Disk Reference Manual
- Wang BASIC-2 Utilities Reference Manual
- Wang Release 2.5 Software Bulletin

CONTENTS

CHAPTER 1 SDS-Extended BASIC-2

CHAPTER 2 Release 2.7 Enhancements

CHAPTER 1

SDS-Extended BASIC-2

What is SDS-Extended BASIC-2?

SDS-Extended BASIC-2 is a modified operating system and enhanced Basic language processor the purpose of which is to extend the usefulness, the effectiveness, the efficiency and the user friendliness of the BASIC-2 Multiuser Operating System. SDS-Extended BASIC-2 contains modifications to the machine language instructions which implement the Basic language and the operating system on the Wang 2200 Series processor.

Each release of SDS-Extended BASIC-2 will contain one or more enhancements and will operate on all Wang 2200 series SVP, MVP, LVP, MVPC and LVPC processors. Installation requires application of "patches" or modifications to Release 2.5 of the BASIC-2 Multiuser Operating System from Wang Laboratories.

Release 2.7 is the first release of the SDS-Extended BASIC-2 Operating System. Users must have a minimum of 24K of control memory and Release 2.5 of the BASIC-2 Multiuser Operating System from Wang Laboratories.

The INSTALL Program

In order to create your version of SDS-Extended BASIC-2 Release 2.7, you must have an unmodified version of BASIC-2 Release 2.5. The changes necessary to create Release 2.7 are applied against the unmodified copy of Release 2.5 and a new file is created.

The SDS Release 2.7 INSTALL program verifies that your copy of the operating system is an unmodified version of Release 2.5 of the WANG Operating System. The INSTALL program makes permanent changes to it. The resulting version, SDS Extended BASIC-2 Release 2.7, allows you to utilize the software enhancements in Chapter 2.

Each copy of SDS-Extended BASIC-2 is created with a unique registration number assigned to the licensee. This registration number and the licensee's name are embedded in the machine readable code on the installation disk supplied. The contents of the installation disk are:

INSTALL	Special Installation Program
@P27	machine language patches to Rel. 2.5
@PR7	machine language Preloader Module

The Hardware and Software Requirements to use Release 2.7

The following hardware and software is required:

- Wang 2200 series SVP, MVP, LVP, MVPC or LVPC processor
- BASIC-2 Release 2.5 with:
  - @@ - UNMODIFIED menu program
  - MVP - UNMODIFIED Release 2.5 Operating System
- Minimum of 24K of Control Memory

The Results of Running the INSTALL Program

The SDS Release 2.7 INSTALL program simultaneously verifies and updates your BASIC-2 Release 2.5 Operating System, and the following new files are created:

- @MVP - user's unique Preloader Module
- @27 - SDS-Extended BASIC-2 Release 2-7

> **CAUTION**
>
> It is very important that you make a BACKUP COPY of your MVP 2.5 Operating System disk before you begin the process of running the SDS Release 2.7 INSTALL program.

If the installation process is interrupted for any reason... a power fluctuation, for example... you cannot use the partially updated version of Release 2.5 to reboot and try it again. You must restore an intact, UNMODIFIED version of MVP 2.5 to begin the process.

Release 2.7 Installation Procedure

	ACTION	COMMENTS
1	Make a backup copy of your Rel 2.5 disk.	Store your original Release 2.5 disk in a safe place.
2	Press CLEAR Press RETURN	Clears memory
3	Type SELECT DISK Daa Press RETURN	where "Daa" is the device address of the Rel 2.7 disk.
4	Type LOAD RUN "INSTALL" Press RETURN	"SDS-Extended BASIC-2 Release 2.7 Installation Program" screen is displayed. An input prompt is given at the bottom of the screen, "Enter the Device Address of the disk containing @MVP (BASIC 2.5)."
5	Type in the Device Address of your @MVP 2.5 disk. Press RETURN	A second input prompt is given, "Enter the Device Address of the disk containing SDS-Extended BASIC-2 Release 2.7 Patches".
6	Type in the Device Address your SDS Release 2.7 disk. Press RETURN	There is no turning back after this point. Your copy of Rel 2.5 is being verified and the new SDS-Extended BASIC-2 Release 2.7 is being created.

When the process is complete the display will read,

"Installation has been successfully completed".

If your version of @MVP 2.5 is damaged or modified, the INSTALL program will stop and display the following message:

"File does not appear to be Unmodified @MVP !"

In this case you must obtain an unmodified copy of Release 2.5, reboot your system using Release 2.5 and start again ... but not before making a backup copy.

This completes the installation of SDS Release 2.7. Your personal copy of the SDS Release 2.7 INSTALL disk can now be discarded.

SDS-Extended BASIC-2 Release 2.7 Startup Sequence

In order to utilize the new features which SDS-Extended BASIC-2 provided, you must master initialize your system using the Release 2.7 Operating System. SDS-Extended BASIC-2 is designed so that the start-up procedures are the same as your Release 2.5 operating system.

You must master initialize your system from Terminal 1 just as you usually do. The following are the procedures for initializing and running your system under SDS-Extended BASIC-2 Release 2.7:

	ACTION	COMMENTS
1	Mount the system platter which contains SDS-Extended BASIC-2 Release 2.7	This disk may already be mounted if you specified a fixed disk in the INSTALL program.
2	Master Initialize the system from the disk which contains your personalized copy of the SDS-Extended BASIC-2 Release 2.7 Operating System	This allows the system startup menu program to display a selection of "DIAGNOSTICS" or "BASIC-2" on your screen.
3	Select "BASIC-2" Press RUN.	The Release 2.7 preloader displays the SDS-Extended BASIC-2 Registration Screen for about 10 seconds while it loads the Release 2.7 operating system

When the operating system is loaded, the system loads and runs the partition gererator program "@GENPART". This will either automatically configure the system for you or the interactive system configuration screen will be displayed, depending on the options set up on your system. You should complete this operation as you usually do.

This completes the procedure for master initializing your system using SDS-Extended BASIC-2 Release 2.7.

You are now ready to utilize the software enhancements described in Chapter 2.

CHAPTER 2

RELEASE 2.7 ENHANCEMENTS

INTRODUCTION

This chapter describes the software enhancements provided with SDS-Extended BASIC-2 Release 2.7. The format and style of the function descriptions is presented in the familar form used by the BASIC-2 Language Reference Manual.

The seven enhancements contained in this chapter are:

- a new ERR$ command
- a new =SELECT command
- changes to the HEXPRINT function
- changes to the IF END command
- a new, selective LIST DC command
- ability to Time and Date Stamp program files
- ability to display Descriptive Error Messages

System Requirements

- Wang 2200 SVP, MVP, LVP, MVPC or LVPC Central Processor
- Minimum of 24K of control memory
- Release 2.5 of BASIC-2 Multiuser Operating System

The ERR$ Command

General Form:

```
          alpha-variable = ERR$ (error-code)
```

Where:

```
          error-code =  the 2-digit numeric portion of the
                        error code.
```

Purpose:

The ERR$ statement provides a convenient means of providing you with an English description of an error. The ERR$ statement places the description of the error code requested in the alpha-variable specified.

When used in conjunction with the ERR function, a program can describe the most recent error to the operator with an appropriate recovery instruction.

Example:

```
100 DIM E$80

200 DATASAVE DC #1, A$(),B$
  : ERROR GOSUB '100
210 . . .

700 DEFFN '100
  : E$=ERR$(ERR)
  : PRINT AT(24,0,80);E$
  : ON ERR-80 GOTO 780,781
  : RETURN

780 REM Error Handler Routines
```

If an error occurs following execution of the DATASAVE DC statement on line 200 a branch occurs to Subroutine DEFFN '100 at line 700 which displays the description of the error on line 24 and branches to an error handler routine starting at line 780.

The =SELECT Function

General Form:

```
alpha-variable =SELECT parameter
```

Where:

alpha-variable = an alpha string variable which will receive the value requested.

parameter = a keyword defined by the following table:

parameter	value returned	length of value returned	format of data returned
CI	Console Input device address	2	0taa
INPUT	INPUT device address	2	0taa
PLOT	PLOT device address	2	0taa
TAPE	TAPE device address	2	0taa
CO	Console Output device address	4	0taaww00
PRINT	PRINT device address	4	0taaww00
LIST	LIST device address	4	0taaww00
#n	0<=n<=15 file-status parameters	8	ftaa ssss cccc eeee
ALL	Master Device Table for current partitions device selections	64	aaup aaup ...

The symbols used in the data format column are defined below:

t = one hex digit specifying the device-type
aa = two hex digits specifying the physical device address
ww = two hex digits expressing the current maximum width of a line
f = file status (0=not open, 1=open on "F" drive, 2=open on "R" drive)

ssss = four hex digits specifying the starting sector address (disk only)
cccc = four hex digits specifying the current sector address (disk only)
eeee = four hex digits specifying the ending sector address (disk only)

u = one hex digit where the binary bits represent device status
bit 1 =1 - device is a disk device
bit 2 =1 - device is assigned to exclusive use of partition (p)
bit 3 =1 - device is currently in use by partition (p)
bit 4 =1 - device is currently hogged by partition (p)

p = one hex digit specifying the number of the partition using the device

Purpose:

Typically a program tracks device selection by setting up a variable or array and maintaining a copy of the current SELECT parameters for decisions on various program actions required by certain devices. An example is in a program producing printed output to either a printer or the the CRT screen. A variable would be set to tell the program to stop every 24 lines if the CRT were selected, or the skip to a new page every 60 or so lines if a printer were selected.

The =SELECT statement provides a convenient means of reading the values contained in the device table. This allows the program to know any SELECTED device at any time without having to maintain a seperate set of varaibles for this purpose.

Example:

```
DIM A$4
SELECT PRINT 215(132)
A$=SELECT PRINT
HEXPRINT A$
```

the printed output is

```
'02158400'
     → Width is Hex(84) = Dec(132)
  →Device is 215
```

HEXPRINT- and HEXPRINT+

```
General Form:     Form 1:    HEXPRINT-          (alpha-variable)
                                                (literal-string)

                  Form 2:    HEXPRINT+          (alpha-variable)
                                                (literal-string)
```

Purpose:

The regular HEXPRINT statement is used to print the value of an alpha-variable or literal-string in hexadecimal notation. The format of the printed value is a continuous string of hexadecimal characters and frequently difficult to read.

The HEXPRINT- statement also prints the value of an alpha-variable or literal- string in hexadecimal notation, however, the format is changed to show pairs of hexadecimal digits seperated by spaces.

The HEXPRINT+ statement prints the value of an alpha-variable or literal-string in both hexadecimal and ASCII format. The printed output is arranged in typical dump format with up to 16 hexadecimal characters seperated by spaces followed by their equavalent ASCII characters for each line. Values which do not convert to printable ASCII characters are displayed as periods.

Examples:

```
Assume:    DIM A$32
           A$="SOUTHERN DATA SYSTEMS, INC."
```

then:

HEXPRINT A$ prints

```
534F55544845524E20444154412053505354454D532C20494E432E2020202020
```

HEXPRINT- A$ prints

```
53 4F 55 54 48 45 52 4E 20 44 41 54 41 20 53 59 53 54 45 4D 53 2C
20 49 4E 43 2E 20 20 20 20 20
```

HEXPRINT+ A$ prints

```
53 4F 55 54 48 45 52 4E 20 44 41 54 41 20 53 59   SOUTHERN DATA
53 54 45 4D 53 2C 20 49 4E 43 2E 20 20 20 20 20   SYSTEMS, INC.
```

The IF -END THEN Statement

General Form:

```
              IF -END THEN (line-number)[ELSE statement]
                           (statement  )
```

Purpose:

The IF END THEN statement is used to test for the presence of an end-of-file record when reading records from a disk.

The IF -END THEN statement is used to test for the ABSENCE of an end-of-file record.

Examples:

Using IF END

```
100 DATALOAD DC A,B,C$
110 IF END THEN 130
120 PRINT A,B,C$
130 . . .
```

Using IF -END

```
100 DATALOAD DC A,B,C$
110 IF -END THEN PRINT A,B,C$
.
130 . . .
```

LIST DCT "mask" Command

```
                                    [<D>]
General Form:                       [<P>]    ["mask"]
                     LIST [S] DC platter [<S>] [-][alpha-variable]
                                    [<SD>]   [(sector)]
                                    [<SP>]
```

Where:

<D> selects only data files for the list

<P> selects only program files for the list

<S> selects only scratched files for the list

<SD> selects only scratched data files for the list

<SP> selects only scratched program files for the list

'-' a minus sign ('-') reverses or negates the selection logic

mask a literal-string or an alpha-variable to define the files to be selected. The mask can contain exact match characters or by using a leading asterik "*", the search will look for any occurance of the mask-string in the file name. Don't-care character positions are defined by the "?" character.

sector a starting sector (or ending sector if a minus is used) will cause the list to begin (or end) at this sector number.

Purpose:

The LIST DC command now allow you to search a disk for selected groups of files. Files may be grouped by name (mask), type (program, data), status (scratched, active) or location (before/after a specified sector).

Examples:	LIST SDCT/310, "GL"	lists all files with "GL" characters in positions 1 and 2
	LIST S DCT "*ASK"	lists all files which have the "ASK" character string in any position of their name
	LIST S DCT "??AB??"	lists all files which have "AB" as the third & forth characters.
	LIST S DCT <SD>(2300)	lists all scratched data files residing above sector 2300.
	LIST S DCT <D>-"*@"	lists all data files except ones with an "@" character anywhere in their name.

TIME & DATE STAMPING of Program Files

General Form:

```
          SAVE [---standard parameters ---] [/text-string]
```

Where:

text-string = an optional literal-string or variable of up to 234 characters in length which is written in the program header sector on the disk.

Purpose:

If the system has an MXE terminal controller or is an SVP with an Option-W, the save statement will automatically write the DATE and TIME in the program header. If the DATE and TIME are not available in the system, zeros are written in the record.

Additionally, an optional text-string of up to 234 characters can be written in the program header record each time a program file is saved. This will allow the programmer to track program revisions and descriptions of programs as required.

The program header can be read using a DATA LOAD BA statement. The format of the program header is as follows:

```
Record  | NNNNNNNNxddmmyyhhmmssttttttttttttttt . . . ttttttttt |
          |      | |     |     |                                |
Byte      1      8 10    16    22                              256
```

```
              N = Program name
              x = control character
         ddmmyy = date (day/month/year)
         hhmmss = time (hour/minute/second)
    ttt . . . ttt = text string of 234 characters
```

Examples:

- SAVE DC T/310, "PROGRAM1"/"Revision 1.7 JRE add range check"

- A$ = "Revision 1.8 by Jim Smith, Modified customer history data
       fields per Change Order number 123-J17"
  SAVE T "PROGRAM2"/A$

- LIMITS T "PROGRAM1",A,B,C,D
  DATA LOAD BA T (A), A$()
  PRINT STR(A$(),22,234)   Prints comment saved with program

Descriptive Error Messages

This enhancement displays descriptive error messages immediately below the normal coded error indicator. This provides on-screen prompting which is more informative and helpful to the user.

```
Example:       10 STR(A$(2,17,20)=STR(B$(J),2,20)
                         ↑
                         ERROR S11
               Missing Right Parenthesis
```

The following table lists the error descriptions displayed with each possible error number:

Error Number	Description
1	Memory Overflow (Text <--> Variable Table)
2	Memory Overflow (Text <--> Value Stack)
3	Memory Overflow (LISTDC, MOVE or COPY)
4	Stack Overflow (Operator Stack)
5	Program Line too long
6	Program Protected
7	Illegal Immediate mode statement
8	Statement not legal here
9	Program not Resolved
10	Missing Left Parenthesis
11	Missing Right Parenthesis
12	Missing Equal Sign
13	Missing Comma or mis-spelled atom
14	Missing Asterisk
15	Missing ">" character
16	Missing letter
17	Missing Hex Digit
18	Missing Relational Operator
19	Missing Required Word
20	Expected End of Statement
21	Missing Line Number
22	Illegal PLOT arguement
23	Invalid Literal String
24	Illegal Expression or Missing Variable
25	Missing Numeric-Scalar variable
26	Missing Array-Variable
27	Missing Numeric Array
28	Missing Alpha-Array
29	Missing Alpha-Variable
30	Not Currently Defined
31	Not Currently Defined
32	Start > End
33	Line-Number Conflict
34	Illegal Value
35	No Program in Memory
36	Undefined Line-Number or CONTINUE illegal
37	Undefined Marked Subroutine
38	Undefined FN Function
39	FNs Nested too Deep
40	No corresponding FOR for NEXT statement
41	RETURN with GOSUB

Error Number	Description
42	Illegal Image
43	Illegal Matrix Operand
44	Matrix not square
45	Operand dimensions not compatable
46	Illegal Microcommand
47	Missing Buffer Variable
48	Illegal Device Specification (Not in device table)
49	Interrupt table full
50	Illegal Array Dimensions or Variable Length
51	Variable or Value too short
52	Variable or Value too long
53	Noncommon Variables Already defined
54	Common Variable required
55	Undefined Variable (Program not Resolved)
56	Illegal Subscripts
57	Illegal STR Arguments
58	Illegal Field-Delimiter Specification
59	Illegal Redimension
60	Underflow
61	Overflow
62	Division by Zero
63	Zero divided by Zero or Zero © Zero
64	Zero raised to a negative power
65	Negative number raised to Noninteger Power
66	Square root of a Negative Value
67	LOG of Zero
68	LOG of a Negative value
69	Arguement too large
70	Insufficient Data
71	Value exceeds Format
72	Singular Matrix
73	Illegal INPUT data
74	Wrong Variable type
75	Illegal number
76	Buffer exceeded
77	Invalid Partition Reference
78	Not currently defined
79	Not currently defined
80	File not OPEN
81	File is Full
82	Requested File is not in Selected Disk Catalog
83	File already exists in Catalog
84	File is not Scratched
85	Index is Full
86	Catalog END error
87	No End of File
88	Wrong Record Type
89	Sector Address beyond End of File
90	Incorrect or no response during Selection sequence
91	Disk may not be mounted
92	Timeout Error
93	Disk Header Format problem
94	Format key engaged
95	Device fault
96	Disk Data error
97	Longitudinal Redundancy Check error (Communication with Disk)
98	Illegal Sector Address or Platter not Mounted
99	Read after Write Error

southern data systems, inc.

5115 Holly Ridge Drive
Raleigh, North Carolina 27612
Phone (919) 781-7603

Printed in U.S.A.