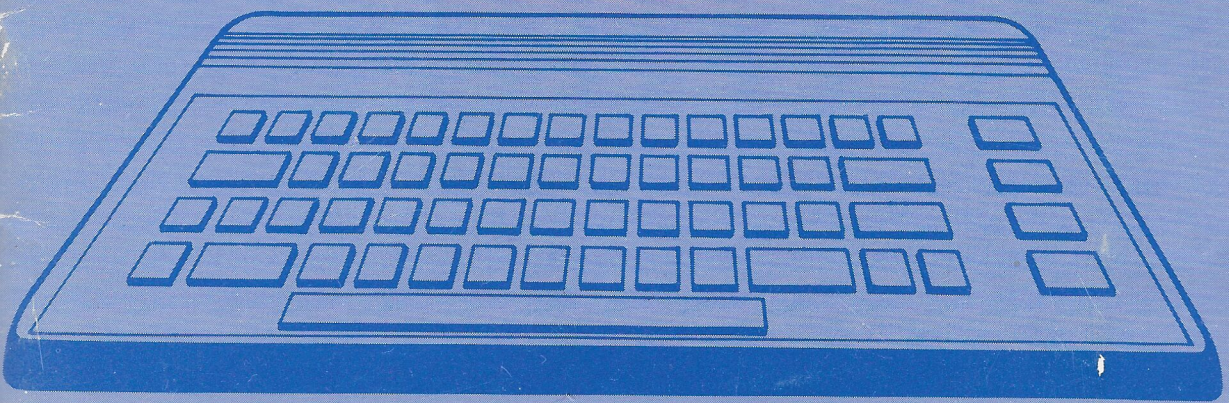
STACK

SUPER HELP

A Programmers Utility Cartridge
for the

COMMODORE 64

20 Additional Basic Commands
Disassembler and Monitor
Disk Operating System Support
2 Pass Assembler



popular computer accessories

HELP / SUPER HELP C64

TABLE OF CONTENTS

INTRODUCTION

INSTALATION

Help and Super Help are cartridges which plug in to the memory expansion port of your CBM 64. To install either one of the units, simply power down your 64. Insert the cartridge into the memory expansion port with the label uppermost. (In a Mother Board with the label facing the keyboard). Then power up the 64. The monitor should now show,
"64K RAM SYSTEM 30719 BASIC BYTES FREE"
"***STACK SUPER HELP***" or "***STACK C64***" for Help.
If this is not the case, power down, remove cartridge, insert cartridge, then power up again.

OVERVIEW

All the commands in HELP/SUPER HELP must be entered in direct mode (not in a program). They also must begin in column 1 of the screen. For the execution of these commands Super Help and Help use memory in zero page (0110 - 0135Hex).
There are no changes which will prevent continuing a program by CONT.

SUPER HELP consists of 4 parts:-
while Help only consists of 3 parts. Some sections are labled 'SUPER HELP ONLY'. These commands are only available on the Super Help option.

1 BASIC ADDITIONAL COMMANDS for Programming

Super Help 22 commands
Help 19 commands :- Not bracketed below
APPEND,(BASIC-RENEW,COMPACT), DELETE, FIND, GENLINE, HELP, SINGLE-STEP, TRACE, END OF SINGLE STEP, and HEXA DECIMAL, DECIMAL to HEXA DECIMAL, (CHECKING for UNDEF'D STATEMENT), OPENing file with CMD, END of CMD, PRINT FRE (Ø), DEVICE No modify.

2 DISASSEMBLER - MONITOR

13 commands for programming, checking, listing and for execution of machine language programs in a C-64 memory and a FLOPPY-DISK memory, providing a MONITOR in combination with a DISASSEMBLER.

3 DOS - commands

8 commands for Floppy-Disk. Featuring simplified disk commands.

Super Help Only
4 2-PASS ASSEMBLER for 65ØØ-6515 MICROPROCESSOR

The operation of each command is explained in the following pages in greater detail.
SUPER HELP restarts automatically after each reset. However you can terminate it by the command: #K. If you want to start it again without turning the C-64 off, just type SYS 33ØØØ.

1. ADDITIONAL BASIC COMMANDS

APPEND

COMMAND-CODE: #A

SYNTAX: #A "NAME",D (similar to LOAD: D = DEVICE)

By means of this command a program from tape or floppy disk may be added behind a program already in memory.

If DEVICE is omitted the append command defaults to TAPE (ie. D=1). File names may only be ommitted for TAPE operations. Line numbers of the new program may be less than or equal to those in the old program. The lengths of the separate programs may differ.

The joined programs can be RENUMBERED with #R. NOTE: If the commands GOTO,GOSUB, THEN or RUN, refer to duplicated line

numbers, the first occuring line number is always used. It is recommended, before joining two or more programs, to RENUMB the single programs so that there are no conflicting lines and so that the line numbers of the new program are higher than the line numbers existing in the memory.

RESTRICTION: Machine language programs which are behind a program in the memory, are deleted by the added program.

HELP

COMMAND-CODE: #H

SYNTAX: #H

If the execution of a program has been interrupted, this command allows listing of the last executed command line. When interruption is caused by a program error (ERROR) the error-causing part is displayed in REVERSE field. This might not be the real error because a calculation error can cause an interruption and only this later interruption address is known.

This command should be used immediately after termination of a program because entering another command may change memory locations.

FIND

COMMAND-CODE : #F

SYNTAX: #F COMMAND (lists all lines containing this command)
#F VARIABLE (lists all line containing this variable name)
#F "STRING (lists all lines containing this string)

This command gives you the ability to list all commands, variables, figures or strings in program. When searching for a VARIABLE with a single letter (eg.A), all lines with a two-letter variable having the same first letter will be also listed (eg.A2, AZ)

When searching for a STRING, it need not be at the beginning

of the inverted commas. Entering of #F" (without string) lists all lines where strings occur.

When searching for a figure (eg.122) all lines with a figure containing 122 will be listed (eg 5122 or 12295). It is also possible to use COMMANDS with FIGURES, eg. GOSUB 5ØØ or with VARIABLES, (eg. IF A.) In this case it is important to enter in the same way as in a program, eg. GOSUB5ØØ if no space had been left or GOSUB 5ØØ with a space.

Listing all displayed lines can be slowed down by the CTRL key and arrested by the SHIFT key (for as long as you depress it). Termination is achieved by the STOP key.

These lines can be printed with a printer by using the ((CMD Command) command

DELETE

COMMAND-CODE: #D

SYNTAX:
#D 1ØØ (erases line 1ØØ)
#D -1ØØ (erases up to line 1ØØ)
#D 1ØØ- (erases all after & including line 1ØØ)
#D 1ØØ-2ØØ (erases all from line 1ØØ to line 2ØØ)
#D without operand is treated as SYNTAX ERROR to prevent erasing the program unintentionally.

With this command a part of the program can be deleted without entering each line number. A machine language program behind this program is NOT DISTURBED by this command.

After execution of this command continuation of a program with CONT is no longer possible.

GENLINE

COMMAND-CODE: #G

SYNTAX:
#G A,B A= INITIAL LINE NUMBER
B= INCREMENT OF LINE NUMBER
#G without A and B; 1ØØ. 1Ø is assumed

After entering this command each press of the return key

prints a new line number. The line number is incremented automatically by the value B. Shift return stops GENLINE.

The initial line number must be in the range Ø-63999. The increment of the line numbers must be in the range 1 to 255.

Other figures will generate; ILLEGAL QUANTITY ERROR. If you reach a value higher than 63999 then 64ØØØ is deducted from this value and the result displayed as the line number. (eg 64001 = 1)

TRACE

COMMAND-CODE: #T

SYNTAX: #T

This command causes the execution of a program line by line. Execution stops for a short time after executing one line. Then it continues at the next line. If the SHIFT key is pressed during execution, the program runs continuously but at a slowed pace.

When staring a BASIC-program with RUN, GOTO, GOSUB or CONT the top screen line displays the line number of the next line be executed. After execution of this line, the line number will be moved to the left.

The contents of the next program lines to be executed are displayed on the second and the third screen lines.

The first three screen lines are deleted each time before a new command is executed. INPUT -commands can cause problems if these commands are in the first three lines, to continue the program without errors put the cursor on the fourth line and then do the entry.

The program can be interrupted at any time with the STOP key and continued with CONT.

This command is operational until it is terminated by the command #E = END TRACE and STEP.

The six line numbers are stored in the memory locations from $0122 - $0130. These memory locations should NOT be modified by POKE during execution of this command.

SINGLE STEP

COMMAND-CODE: #S

SYNTAX: #S

This command causes a Basic-program to be executed line by line. Execution stops until one of the two SHIFT KEYS is depressed. If one SHIFT KEY is held down, Basic-program runs slowly (similar to TRACE).

When starting execution with RUN, GOSUB or CONT or GOTO, the top screen line displays the line number of the next line to be executed. On executions this line number will be moved to the left. The last 6 line numbers are displayed continously.

The contents of the next program lines to be executed are displayed on the second and the third screen lines, the first three screen lines are deleted each time before a new command is executed. INPUT -commands can cause problems if these commands are in the first 3 lines, to continue the program without errors put the cursor on the fourth line and then do the entry.

The Program can be interrupted at any time with the STOP key and continued with CONT.

This command is operational until it is terminated by the command #E = END TRACE and STEP.

The six line numbers are stored in the memory locations from $0122-$0130. These memory locations should NOT be modified

by POKE during execution of this command.

END of TRACE and STEP

COMMAND-CODE: #E

SYNTAX: #E

This command changes the execution of a Basic-program back to the NORMAL-MODE. For switching from SINGLE-STEP to TRACE or vice versa the command #E is not necessary.

RENUMBER

COMMAND-CODE: #R

SYNTAX: #R A,B A = initial line number
B = increment of line numbers

#R Without A and B: 10,10 is assumed.

This program renumbers a program stored in memory. At the same time all GOTO, GOSUB, THEN and RUN statements followed by one or more line numbers, are also changed. Before this command is executed the program should be verified by USING UNDEF'D STATEMENT COMMAND (#U)., If an error occurs. This error should be corrected because, if a none existant line number is referenced it will be numbered to 65535.

With this command a program can be made longer or shorter depending on the initial line number and the increment.

The initial line number has to be in the range 0 to 63999. The increment of the line numbers has to be in the range 1 to 255. Entry of other figures causes the program to be interrupted with; ILLEGAL QUANTITY. NOTE: If there is a line number greater than 63999, 64000 is deducted and the remainder assumed as new line number. The order of these two program lines, however, is the same.

A program can be renumbered repeatedly without being destroyed. A machine language program after the Basic program is not altered. If the running program finds a line number which is

greater than 63999, execution is interrupted and SYNTAX ERROR, generated.

VARIABLE DUMP

COMMAND-CODE: #V

SYNTAX: #V

With this command all non-matrix VARIABLES used in a program can be listed with their current values after execution.

Listing is made in the sequence of the activation of the variables. All variables are printed in such a way that their values can be listed, edited to new values and with the CONT command given continue the program with the new values.

Listing of VARIABLES can be slowed down by CTRL key and held with the SHIFT KEY (as long as you press it). You can interrupt by use of the STOP key.

It is also possible to print the VARIABLES with a printer as a normal listing by (,(CMD).

MATRIX DUMP

COMMAND-CODE: #M

SYNTAX: #M

With this command all ONE DIMENSIONAL or MULTI DIMENSIONAL VARIABLES used in program can be listed with their current values after execution.

Listing is made in the sequence of activation of the variables. All variables are printed in such a way that the values can be listed, edited to new values and with the CONT command given continue the program with the new values.

Listing of the VARIABLES can be slowed down by CTRL key and held with SHIFT key (as long as you press it). You can interrupt a normal listing by ((CMD).

PAGE LIST

COMMAND-CODE: #L

SYNTAX: #Ln n = line number
#L without line number = listing from beginning

Listing begins at the first line number of the program and list the contents page by page (25 lines) on the screen.

To list the next page press the RETURN key.

If you press the UP ARROW KEY ↑ the previous page is listed.

The feature can only be terminated by pressing the stop key, the cursor is always situated in the bottom left corner and the listed program on the screen will not be scrolled.

As this command is only intended to list on the screen an existing CMD will be terminated before execution and not opened again.

CONVERSION from HEX to DECIMAL

COMMAND-CODES: !$

SYNTAX: !$DDDDD

A DECIMAL FIGURE (positive from Ø to 65535) is converted to a HEXIDECIMAL FIGURE up to 4 digits. For the values 1Ø to 15 the letters A to F are used.With this command variables are permitted.

BASIC RENEW [SUPER HELP ONLY]

COMMAND-CODE: #B

SYNTAX: #B

This command can be used in three ways.

1. A basic program deleted by NEW is reactivated with this command.
2. A BASIC program is set free from an appended machine language program or a program part which has been just appended by OVERLAY. Program-end pointer is reset.
3. If after loading a basic program no CURSOR appears. STOP-RESTORE until the cursor appears, then use this command to reset the program pointer.

This command should not be used if there is no basic program in memory because it is possible that program lines with undefined content may be generated.

Opening of FILES and CMD

COMMAND-CODE: (

SYNTAX: (d d = DEVICE NUMBER (4 - 31)
(without d; DEVICE 4 is assumed

This command opens a OUTPUT FILE with the FILE NUMBER 255. If this FILE is already opened it does not lead to a FILE OPEN ERROR with CLOSING OF ALL FILES but the existing open FILE will be used for the CMD. With this command a SECONDARY ADDRESS is not possible.

This command generates the following;

1. OPEN 255,d d = DEVICE
2. CMD 255

CMD and CLOSING OF FILES

COMMAND-CODE:)

SYNTAX:)

This command terminates a CMD and closes the FILE 255.

MODIFYING OF A DEVICE NUMBER for DOS

COMMAND-CODE: £

SYNTAX: £d d = DEVICE NUMBER (4 - 31)

With this command the DEVICE NUMBER for DOS and ASSEMBLER can be modified. Every time you switch on the SUPER HELP the DEVICE NUMBER will be 8.

KILL

COMMAND-CODE: #K

SYNTAX: #K

With this command SUPER HELP is switched off and there is no interruption for other programs using the same SYNTAX. To restart HELP C-64 PLUS use SYS 33ØØØ.

COMPACTOR [SUPER HELP ONLY]

COMMAND-CODE: #C

SYNTAX: #C d d = DENSITY (1 - 24Ø)
#C without d; DENSITY 24Ø is assumed

With this command a BASIC program is condensed. All unnecessary SPACES and all REM's are removed. Lines are joined until the length (density) wanted is acheived except if they are jumped - on lines. (ie. the target of a goto etc)

Before the program is compacted, it is checked for UNDEF'D STATEMENT. If this error occurs, execution is interrupted and the compacting stopped.

After compacting, the program is RENUMBERED by 1,1 to get maximum density.

COMPACTED programs run faster because the time taken to find a line is shorter for jumping commands. Unnecessary SPACES and REMS cost time. Once compacted a program is vey difficult to disentangle and to modify. A compacted program should only be used for the execution of a program and not for the test phase. Always keep a copy of the original un compacted, in case you want to change a line.

CHECKING for UNDEF'D STATEMENT [SUPER HELP ONLY]

COMMAND-CODE: #U

SYNTAX: #U

The BASIC program existing in memory is checked for UNDEF'D STATEMENT. If such an error is found you will see the line number of the program line in this FORM: UNDEF'D STATEMENT ERROR in 1ØØ (if this error has been found in line 1ØØ).

FREE MEMORY LOCATIONS

COMMAND-CODE: *

SYNTAX: *

This command replaces the command: PRINTFRE(Ø)

2 DISASSEMBLER - MONITOR

COMMAND-KEY] <RETURN>

This key starts a MONITOR program where the BASIC-EDITOR is disabled. This program services 13 commands which are described in detail later.

The address of a memory location for this microprocessor is shown by a 16 bit hexadecimal number (ie. 4 digits) and the contents of this address are shown as 8 bit hexadecimal number (ie. 2 digits). With this MONITOR the address will be displayed on the left and the contents (DATA) are displayed two spaces to the right.

For indication and input of a HEXA decimal value the figures Ø - 9 and for the value 1Ø - 15 the letters A - F are used.

The DISASSEMBLER-MONITOR allows 3 MODES which are in a defined position when starting this program.

MODE1 : MODIFYING of ADDRESS or DATA

When starting the DISASSEMBLER MONITOR the ADDRESS MODIFY MODE is on. Because the same keys (Ø-9 and A-F) are used for changing both address and DATA, it is necessary to determine which has to be changed.

MODIFYING of ADDRESS; COMMAND-CODE: +

After typing this command key the keys Ø - 9 and A - F modify the address in the following way: The entered value is written to the fourth digit, all other values are shifted left one position. The CONTENTS (DATA) of this new address is then printed.

MODIFYING of DATA: COMMAND-CODE /

After this command key has been depressed, the keys Ø - 9 and A - F modify the address contents (DATA) in the following way. The entered value is placed in the second digit and the former value is shifted to the first digit. The entered value is immediately read out and displayed again. If this address is ROM, the input causes no changes so MONITOR can be used for ROM/RAM checking.

MODE 2 : OUTPUT at SCREEN or PRINTER

This mode is only valid for the DISASSEMBLER, because the MONITOR uses only the screen. (With MONITOR the contents of the selected memory locations are permanently read out and displayed. This is useful to recognize a TIMER REGISTER or similar memory locations). At each call of the DISASSEMBLER MONITOR the SCREEN MODE is turned on.

OUTPUT to SCREEN DISPLAY: COMMAND-CODE:]

After pressing this command key the DISASSEMBLING output switched to DEVICE 3 for the SCREEN. It is not necessary to open a FILE.

OUTPUT to PRINTER: COMMAND-CODE [

After pressing this command key the DISASSEMBLING output is switched to DEVICE 4 for the printer. It is not necessary to open a FILE.

MODE 3 : DISPLAY of C-64 MEMORY or DISK MEMORY

By means of this it is possible to list the memory locations of a connected FLOPPY-DISK (ROM and RAM). Upon starting the DISASSEMBLER MONITOR, program is switched to C-64 memory.

C-64 MEMORY: COMMAND-CODE: >

After pressing this command key program is turned over to C-64 memory and the content of the C-64 memory location is displayed.

DISK MEMORY: COMMAND-CODE: <

After pressing this command key, program is turned over to DISK memory and the content of the DISK memory location is displayed. In this state memory content is only read out once, unlike the C-64 MEMORY so that the DATABUS is not obstructed.

INCREMENT ADDRESS: COMMAND-CODE: RETURN KEY

The address is incremented by one. The new address and its content do not overwrite the old line.

DECREMENT ADDRESS: COMMAND-CODE ↑

The address is decremented by one. The new address and its contents overwrite the old line.

RUN: COMMAND KEY: *

With this key a machine language program with the desired address is started. With RTS in a machine languages program you return to the MONITOR AGAIN IF DISK MEMORY MODE is on, with this command a program in the FLOPPY DISK can be started. Here MONITOR is not obstructed.

DISASSEMBLER 1 COMMAND: COMMAND-CODE: -

Typing this command key causes the disassembly of one command starting with the selected address. If PRINTER OUTPUT MODE is selected, this output is directed to DEVICE 4. If this is no command-code, then *** is displayed. All displayed

addresses and data are HEXADECIMAL and have no leading $-character for easier readability.

CONTINUED DISASSEMBLER: COMMAND-CODE: SPACE KEY

Disassembly starts at the selected address and contines until STOP key is depressed. Printing can be slowed down with CTRL (as with listing) and it can be stopped with a SHIFT key (as long as you press it). If PRINTER-OUTPUT-MODE is selected, this output is directed to DEVICE 4.

TRANSFER: COMMAND-CODE: @

With this command program or data can be moved from one memory location to another. The length of the moving program is not limited. Before the command key is pressed, the following entries have to be made:-

1. START ADDRESS of the program to be moved
   ØØ22 low-part and ØØ23 high-part
2. END ADDRESS +1 of the program to be moved
   ØØ24 low-part and ØØ25 high-part
3. BEGINNING ADDRESS - where to move the PROGRAM, to
   (selected in MONITOR)

EXAMPLE: The BASIC-ROM (AØØØ-BFFF) to be moved up to the RAM memory (starting at 2ØØØ).

1. Type the COMMAND KEY +	(ADDRESS-MODE)
2. Put the ADDRESS to ØØ22	(Entry: Ø Ø 2 2)
3. Type the COMMAND KEY /	(DATA-MODE)
4. Change the contents at 0022 to Ø Ø	(Entry: Ø Ø)
5. Press RETURN key	(INCREMENT to ØØ23)
6. Change contents of ØØ23 to AØ	(Entry: AØ)
7. Press RETURN key	(INCREMENT to ØØ24)
8. Change contents of ØØ24 to ØØ	(Entry: to Ø Ø end address + 1 = CØØØ)
9. Press RETURN key	(INCREMENT to ØØ25)
10. Change contents of ØØ25 to CØ	(Entry: to CØ)
11. Press COMMAND KEY +	(ADDRESS-MODE)
12. Put ADDRESS to 2ØØØ	(Entry: 2 Ø Ø Ø)
13. Press command key @	(Transfer follows)

RETURN to BASIC: COMMAND-CODE: =

DISASSEMBLER-MONITOR program is terminated. An interruption BASIC program can be continued by CONT after using the DISASSEMBLER-MONITOR.

3 D O S S U P P O R T

This program section is for simplifying and shortening the command input for DISK- instructions in the DIRECT-MODE. When SUPER HELP is initialized DEVICE 8 for these commands is assumed. But following commands are possible.

LOAD PROGRAM with relocating: COMMAND KEY: / PROG NAME

A program name must exist after this command key. (It can also be only a part of the name, but if so it must be ended with *). Between command code and program name no quote marks or spaces should be inserted.

LOAD PROGRAM without relocating: COMMAND KEY: % PROG NAME

The program is not loaded according to the Start of Basic pointer but as the disk Start of Program pointer.

LOAD PROGRAM with relocating + RUN: COMMAND KEY↑ PROG NAME

After loading, program is started automatically.

SAVE PROGRAM: COMMAND KEY; ← PROG NAME

With a double FLOPPY it is important to enter the DRIVE number before the program name in the following form: Ø:NAME for DRIVE Ø or 1:NAME for DRIVE 1. If a STATUS-ERROR occurs during saving, saving is stopped and the DISKSTATUS is listed. If it is a FILE EXISTS ERROR you are asked: OVERWRITE YES/NO. The program on DISKETTE is deleted by pressing Y and the new program is saved If the same error occurs once more, the disk should be verified for WRITE PROTECTION. If you enter N nothing is deleted or saved.

After termination of saving, DISKSTATUS is dispalyed.

VERIFY PROGRAM: COMMAND KEY: <PROG NAME

Program on DISK is verified with program in memory.

COMMAND CHANNEL 15 COMMANDS: COMMAND KEY: @

The command following this command key must be entered without comma or quote mark. eg.:

@NØ:NAME,ID	= a NEW of a diskette with ID
@IØ	= INITIALIZE DRIVE Ø
@SØ:NAME	= delete a program (Scratch)
@B-R:2Ø5 2	= BLOCK READ to CHANNEL 2 from DRIVE Ø TRACK 5 SECTOR 2
@C1:NAME=Ø:NAME	= COPY of a program on DRIVE Ø to 1
@D1=Ø	= DUPLICATE from DRIVE Ø to 1

M-R command cannot be used with this command. This facility is provided by the DISK MEMORY MODE OF THE MONITOR.

TABLE OF CONTENTS: COMMAND KEY:$

With this command key the table of contents (DIRECTORY) is listed on the screen. If you have a double-floppy you can select with $Ø or $1 which drive you want. If you enter CMD before this command, the directory will be displayed on the printer.

A program existing in memory will NOT be deleted.

DISK-STATUS INPUT: COMMAND KEY: @without text

If this command key (without text) pressed, DISK-STATUS is listed on the screen, a CMD is closed.

GET CHANNEL 15: COMMAND KEY:>without text

One single character is displayed from the COMMAND CHANNEL in HEXADECIMAL. It is used after the command M-R. NOTE: After M-R no INPUT with the command key @ is allowed because it does not transmit an END CHARACTER.

4 2 PASS ASSEMBLER [SUPER HELP ONLY]

COMMAND CODE: [<RETURN>

Super Help Assembler only operates with a 1541 Floppy Disk Drive.

The assembler is a full symbolic two pass assembler and can be used at any time that SUPER HELP is active, and it does not use any other external programs. To generate an assembler source program the normal BASIC editor is used. Consequently the complete SUPER HELP command set can be used when creating an assembler source program, mainly :-GENLINE,RENUMBER, APPEND, DELETE, FIND and PAGE-LIST.

The assembler is a two pass assembler this means that the source program that is stored on disk is read by the assembler twice. During the first pass all the labels in the source program are calculated and then during the second pass all the opcodes and operands are translated into machine code.

IMPORTANT:- SEE LABEL EQUATES

Before calling up the assembler a source program must be written (using the normal BASIC editor) and then stored on disk (either by SAVE "file Name", 8 or with - filename).

LINE NUMBER (LABEL) OPCODE OPERAND (REMARK)

Each line must have a line number, also each line must have a OPCODE (COMMAND). In addition to these two fields the bulk of opcodes (all addressing modes except implied) require an operand (normally an address). Both the LABEL and REMARK fields are optional. For each line only one of each of the fields is permitted and there must be at least a one space separator between each individual field.

LABEL

A Label may consist of 1 to 6 alpha-numeric characters but it must start with a letter. However, you are not allowed to use of the 56 reserved opcodes and the reserved single

letters A,X and Y of the 65xx series microprocessors. A label is used in a program for two reasons either to mark a program part or to equate an address for ease of use.

If the ASSEMBLER recognizes a LABEL while translating the source code then the present program memory address (PROGRAM POINTER) is equalled to the LABEL at that time. If you use SYMBOLIC LABEL instead of an OPERAND (ADDRESS) then the ASSEMBLER during the second pass will replace the LABEL with the ADDRESS that was allocated to that LABEL during the first pass and an error will occur. When assembling, the ASSEMBLER reserves 2 BYTES in the object code for each ADDRESS equated to a SYMBOLIC LABEL,consequently all ZERO-PAGE LABELS must be defined as OPERANDS before they occur in the source code, as ZERO-PAGE ADDRESS require that only 1 BYTE is reserved.

OPCODE

Command word for the 65xx microprocessor. It consists of 3 letters and there are 56 commands. In addition to these 56 commands the SUPER HELP C64 ASSEMBLER also knows the ASSEMBLER-DIRECTIVES (PSEUDO OPCODES).

The 65xxx-microprocessor knows the following 56 OPCODES:-

ADC = add with carry
AND = AND operation
ASL = Arithmetic shift left one bit
BCC = relative jump if C in PSR = Ø
BCS = relative jump if C in PSR = 1
BEQ = relative jump if Z in PSR = Ø
BIT = testing for BIT 7 and 6
BMI = relative jump if N in PSR = 1
BNE = relative jump if Z in PSR = 1
BPL = relative jump if N in PSR = Ø
BRK = program interupt, I and B in PSR will be 1 (NO OPERAND)
BVC = relative jump if V in PSR = Ø
BVS = relative jump if V in PSR = 1
CLC = C in PSR will be Ø (NO OPERAND)
CLD = D in PSR will be Ø (NO OPERAND)
CLI = I in PSR will be Ø (NO OPERAND)
CLV = V in PSR will be Ø (NO OPERAND)
CMP = compare memory with accumulator

CPX = compare memory with X-register
CPY = compare memory with Y-register
DEC = compare memory with Y-register
DEX = decrement X-register by one (NO OPERAND)
DEY = decrement Y-register by one (NO OPERAND)
EOR = EXCLUSIVE-OR-operation
INC = increment memory by one
INX = increment X-register by one (NO OPERAND)
INY = increment Y-register by one (NO OPERAND)
JMP = JUMP absolute or indirect
JSR = sub-routine is called
LDA = load accumulator with memory contents
LDX = load X-register with memory contents
LDY = load Y-register with memory contents
LSR = Logical shift right one bit
NOP = no operation (NO OPERAND)
ORA = OR-operation
PHA = Push accumulator on to the (NO OPERAND)
PHP = Push PSR on to the stack (NO OPERAND)
PLA = Pull accumulator off the stack (NO OPERAND)
PLP = Pull PSR off the stack (NO OPERAND)
ROL = Rotate one bit left
ROR = Rotate one bit right
RTI = return from interrupt (NO OPERAND)
RTS = return from sub-routine (NO OPERAND)
SBC = subtracting with borrow
SEC = C in PSR will be 1 (NO OPERAND)
SED = D in PSR will be 1 (NO OPERAND)
SEI = I in PSR will be 1 (NO OPERAND)
STA = Store accumulator in memory
STX = Store X-register in memory
STY = Store Y-register in memory
TAX = transfer accumulator to X-register (NO OPERAND)
TAY = transfer accumulator to Y-register (NO OPERAND)
TSX = transfer stack pointer to X-register (NO OPERAND)
TXA = transfer X-register to stack pointer (NO OPERAND)
TXS = transfer X-register to stack pointer (NO OPERAND)
TYA = transfer Y-register to accumulator (NO OPERAND)

OPERAND:

An operand is the address part of an opcode. The 65xx series of microprocessors recognise 13 different addressing modes.

With some of the 56 opcodes more than one addressing mode can be used, while some opcodes only allow one addressing mode. There are some opcodes that do not require any operand this is called implied addressing. The operand for most opcodes is normally 16 bits long (2 bytes), but certain addressing mode s require an 8 bit (1 byte) operand (namely zero page addressing modes and relative addressing). For these 8 or 16 bit addresses the following methods of decription can be used:-

Decimal number: An address which begins with a figure is interpreted as a decimal number. This number must be in the range Ø to 65535.

Hexadecimal number: An address which begins with the dollar character ($) is interpreted as a hexadecimal figure This number must be in range $ØØØØ to $FFFF. (Hexadecimal = base 16).

Octal number:- An address that begins with the at character (@) is interpreted as an octal number. This number must be in the range @ØØØØØ to @177777. (Octal = base 8).

Binary Number: An address which begins with the percent character (%) is interpreted as a binary number. The length of this number can be up to 16 digits and the individual digits may only be either Ø to 1. (Binary = base 2)

ASCII figure: A figure or character which is preceded by a ' or a " is translated as an ASCII character.

Symbolic address: The address that is equated to this symbol is used as a figure. The address may be defined in any of the previously mentioned forms:-

```
eg.   MEMTOP = 65535
      MEMTOP = $FFFF
      MEMTOP = @177777
      MEMTOP = %1111111111111111
```

The operand may be several of these figures joined together. However, between each of these single figures there must be either a + (for adding figures or a - (for subtracting figures).

If the low byte of a two byte figure or symbolic address is all that is required then the LESS THAN character (<) is placed after the figure or address.

Similarly if the high byte of a two byte address or symbolic address is required then the GREATER THAN character (>) is placed after the figure or address.

ASSEMBLER DIRECTIVES. (PSUEDO-OPCODES).

In addition to the 56 opcodes that the SUPER HELP assembler recognises there are a further 7 ASSEMBLER directives. These are the sign of equality (=), and 6 psuedo-opcodes. Each of the psuedo opcodes begins with a point (.).

. BYTE Here a figure between Ø and 255 (decimal) can be entered. The syntax of this figure in the assembler source code can be any type of operand (decimal hexadecimal, octal, binary, ASCII or symbolic) and may also be connected with either + or -. After the .BYTE psuedo-opcode there may also be more bytes, but these must be separated with comas eg .BYTE $2Ø, 'A, 22Ø, %11Ø1. TXTABLW

. WORD syntax like .BYTE, HOWEVER A FIGURE FROM Ø to 65535 can be entered. This 2 byte figure is stored in memory by the assembler in the form low byte first. followed by the high byte.
eg, .WORD $FFD2 is stored as D2 FF

. TEXT After this command there must either be the ? character or the " character. The character string must end with the same delimiter as it began. Between the delimiters there can be more characters (letters, numbers, special characters). During assembly each of these characters is translated into ASCII.

.DISP like the .TEXT psuedo-op, however the characters between the delimiters are translated into 6 bit ASCII.
eg. .TEXT 'A' returns $41 in object code
.DISP 'A' returns $Ø1 in object code

. END If this directive is in the source program then the assembler terminates translation. Normally the assembler

will write this directive into the source program itself if it gets an end of program status when reading from the disk drive.

. LOAD With this command a new program is loaded from disk for translation. The program name follows this directive. N.B. The .LOAD directive has one restriction,, that is the program name of the called program must be of exactly the same length as the name of the calling program.

SIGN OF EQUALITY= Using this sign of equality one can either equate symbols to specific memory locations or set the program pointer. When using the sign of equality there are the following possibilities, either,
1) There is a label before the =, and after it there must be an operand (decimal, hexadecimal, octal, binary, ASCII or symbolic. The value after the = is equated to the label before the sign of equality.
EG.

```
BACKG = 53280
FOREG = BACK G + 1
```

NB. ALL LABEL EQUATES MUST APPEAR AT THE START OF THE PROGRAM OR BRANCH ERRORS MAY RESULT.
2) There is an asterix (*) before the sign of equality. (* refers to the current PROGRAM POINTER). After the equality there must be an address figure in the form of an operand. The program pointer is equated to the value following the sign of equality. This command must always be at the beginning of an assembler source program otherwise the start address for the object code will be undefined.
eg. * = $7ØØØ here the object code will reside from $7ØØØ onwards.
and the current program point is equated to $7ØØØ

REMARK. The remark field within a line in the assembler source program must begin with a semi-colon (;). If the assembler recognises this character all assembly on this line is terminated. All further characters can be used for explanation of this line.

ASSEMBLING

When the source program is complete and saved onto disk it needs to be assembled. To initiate the assembler the command key [is typed, the assembler then asks for the PROGRAM NAME. The name entered here must be the same as the filename of the source program saved on the disk. Furthermore this name is used as the page heading for the print out obtained on a printer (Optional).
The assembler then asks a second question, that is ORG OFFSET ($). This question has the following meaning:
When the source program is translated by the assembler it is normally written to the appropriate location immediately, however if the considered memory areas are not free, eg. a ROM or operating system area or occupied by an assembler label table (the assembler needs 8 bytes for each label in the source code, and this data is stored from $ØØØØ onwards) then the origin of the program may be modified by answering the question with a hexadecimal number of up to four digits. Note (return = $ØØØØ). This figure is then subtracted from the origin as stated in the program.
eg. a source program starts with
10* = $8ØØØ ; ie 32768 decimal
Unfortunately the is in the SUPER HELP C64 ROM area, so by answering the question ORG OFFSET ($) with 1ØØØ ie $1ØØØ the assembled program can be written $1ØØØ bytes lower, that is starting at $7ØØØ

The final question is OUTPUT CODE the required answer to this question is based around a three bit binary number, each of the three bits represent a different form of output and whether the bit is set or reset determines what occurs.

If the first bit is set (ie = 1) then if an error occurs during assembly then an error message and appropriate line number are displayed on both the screen and the printer, If a return or a zero is input then no errors are reported during assembly. If the second bit is set then an assembly listing is displayed on both the screen and printer. If a return or zero is entered then no listing is displayed.

If the last bit is set then at the end of assembly there will be alphabetically sorted symbol table displayed on both the screen and printer. (A symbol table is a list of all symbolic labels found with in source program were their associated values). When this bit is reset (Ø or return) then a symbol table will not be displayed.

Some examples of input for this final question:-

What is typed	What is output
return (equivalent to ØØØ return)	there is no listing of errors or program and no symbol table
111 return	there is a list of all the errors occuring and a listing of the program and symbol table.
011 returns	no errors but a listing of the program and a symbol table.
1 return (equivalent to 1ØØ return)	Errors only

For each display on the screen there is a corresponding output on a printer. The printer runs a line counter with 72 lines per page and a page counter. At each new page a title is printed consisting of the program name and a page number. If output to the printer is not required than all that is required is for the printer to be turned off.

After these three questions have been answered the assembler begins to translate the source program. During the first pass nothing visible occurs, then during the second pass the object code is written into memory (and output as appropriate).

At the end of assembly there is a display of the number of errors, symbols and lines translated that exist in the source program.

ERROR MESSAGES

The SUPER HELP assembler can distinguish six different types of error during the assembly process. With the one exception of DUPLICATE SYMBOL ERROR which it indicates during the first pass, all of the remaining error messages are indicated during the second pass. These errors are displayed in accordance with the reply to the OUTPUT CODE question. An error counter counts each error and lists the result at the end of assembly. (The error counter counts up to 255 errors and then reset to Ø and starts again for more than 255 errors.

The errors

SYNTAX ERROR. Occurs if there are several (or no) opcodes occuring in one line, or if there is no space between the fields within one line.

ONE BYTE RANGE ERROR. Occurs if a figure is greater than 255, but only one byte is allowed (eg. during zero page addressing).

RELATIVE BRANCH ERROR. Occurs if a relative jump address is not in the permitted range (ie. -128 to +127).

ILLEGAL OPERAND ERROR. Occurs if the addressing mode is not valid for the opcode occuring in the same line.

UNDEFINED DIRECTIVE ERROR. Occurs if an opcode begins with a point but the following text is not one of the six recognised psuedo-opcodes.

UNDEFINED SYMBOL ERROR. Occurs if a symbolic address is referenced without having been defined as a label.

DUPLICATE SYMBOL ERROR. Occurs if a label is appointed to more than one symbolic address.

SAVING THE ASSEMBLED OBJECT CODE

When the assembler has translated the source program without any errors, one is left with the object code in RAM. Now this program can be started withe SYS command but normally it would be saved for later use. To do this, four pointers need to be altered, these are program start pointer ($2B = low byte, $2C = high byte) which is set equal to the object code start, and program end pointer ($2D = low byte, $2E = high byte) which is set equal to the object code and + 1. These pointers can be modified using the monitor section of SUPER HELP. When these pointers have been set the object code can be saved as a normal basic program. After saving these pointers they must be reset to their original values (easily done by turning the C64 off and then back on). It is important to note that this object code can only be loaded with a secondary address of 1 or with the SUPER HELP DOS-LOADING COMMAND %.

SUMMARY

1. BASIC COMMANDS

APPEND	#A	VARIABLE DUMP	#V
HELP	#H	MATRIX DUMP	#M
FIND	#F	PAGE LIST	#L n
DELETE	#Dx-y	HEX CONVERT	!# HHHH
GENLINE	#G A,B	DEC CONVERT	!$ DDDD
TRACE	#T	BASIC RENEW	#B
STEP	#S	OPENING FILES	(d
END #T#S	#E	CLOSING FILES	)
RENUMBER	#R, A,B	MODIFY DEVICE	£d
KILL	#K	COMPACTOR	#C
FRE MEMORY	*	UNDEF'D	#U

2. DISASSEMBLER-MONITOR

COMMAND KEY	Return	C64 MEMORY	>
MODIFYING ADDR	+	DISK MEMORY	<
MODIFYING DATA	/	INCREMENT ADDRESS	RETURN
OUTPUT TO SCREEN	]	DECREMENT ADDRESS	↑
OUTPUT TO PRINTER	[	RUN	*
DISASSEMBLE 1 line	—	RETURN TO BASIC	=
" CONT	SPACE BAR		
TRANSFER	@		

3. DOS SUPPORT

LOAD PROGRAM (RELOCATE)	/	VERIFY PROG	<
LOAD PROG	%	NEW DISK	@NØ:NAME,ID
		INITIALISE	@IØ
LOAD PROG RELOCATE & RUN	↑	DELETE	@SØ
SAVE PROG	←	BLOCK READ	@B-R
		COPY DO TO D1	@C1:
		DISK COPY	@D1=Ø
TABLE OF CONTENTS	:$		
DISK STATUS	@	GET CHANNEL 15	>

4. ASSEMBLER

INITIATE ASS	[return

DISCLAIMER.

Whilst every effort has been made to provide a flexible, reliable and above all low-cost product STACK COMPUTER SERVICES LTD. wish to point out that no claim is made for complete compatibility with any other equipment or program. The information given is believed to be accurate but no liability can be accepted for the consequences of any error. Ours is a policy of continued development and we therefore reserve the right to alter the design of specifications without prior notice.

REQUEST FOR INFORMATION.

In order to provide the user with as much support as is practical, we would appreciate it if any useful comments or hints could be forwarded in writing to:-

PRODUCT DEVELOPMENT
STACK COMPUTER SERVICES LTD
290/298 DERBY ROAD
BOOTLE
LIVERPOOL
L20 8LN