TANGERINE USERS GROUP

16 Iddesleigh Road Charminster
Bournemouth Dorset U.K. BH3 7JR
Telephone (0202) 294393


```
07AC  48A9058568A2FF8AA8A56F29FD8DF28F4910856F88D0FDCA    0E68
07C4  D0EDC668D0E7A901200108A56F29EF856F68AA689DDF0368    0CDC
07DC  9DDE03689DDD0320F604F838A53CE901853CB0034C050AD8    0P
07F4  68A868AA602408609004 18A90F856BA2FF88D0FDADF3BF0A    r
080C  857E2902D007CAD0F0C668D0EA608DF3BFBD61098570E8      0'
0823  BD6109857 1A0FFC8E88D6109F0049170D0F5ADF08F60A200
0838  BC5F088DF3BFB538484A4A4A4A 186930992002C868290F
0852  6930992002C8E8E008D0DEF0D60806040C 101816144488A4'
086A  4A4A4A4AAAF8A552C9348008188D8 10865388538A5394F
0881
0898
0880
08C7
08DF
08F5  B9340930D3002D01DB93409 1D300290D3002A4' length = 31
090D  90B6E678A57DD004C678C678602067 08A90'
0923  9D4002CA10FAA9008578A56F29F7856F 6' nts = 28 and 31
0938  BD0010C67AD01EC678D01AA9058578A9 length = 2147483647
0952  A9638578A91E8579A56F0908856F60'
0963  494E564153494F4E20434F4D504C4'
0977  505245535320504C4159204845 ions $60 to $63 must be
0988  535041434520494E564153494F
099F  3530001103343000510333 0' eded before using the
09B3  43542C4554492C284329544 routine
09CB  00008B02883F7400CC021'
09E3  032404000020AC05A22F ;
09F8  E8BD610900F720FF07 0  * = $1F00                    ; int PC
0A13  10FA20FF07A2028 5?
0A2B  CA10F9301DD8A90' 120 ; initialise variables
0A42  942818692090 0' 130 ;
0A58  205A05A21420'
0A6E  8DF2BF2039 0'       140 REG = $60
0A86  A9648538A'          150 TEMP = $64
0A9D  84578452
0AB4  C64AC6'             160 ;
0ACB  203E 0'             170 ; Start of Program
0AE3  38E'
0AFB  EF                  180 ;
0B12  1F00 A5 63          190 LDA REG + 3                ; Get bit 28
0B2A  1F02 29 08          200 AND #8                     ; from reg
0B41  1F04 0A             210 ASL A
0B5'  1F05 0A             220 ASL A
0B'   1F06 0A             230 ASL A
```

EDITOR ▶ ASSEMBLER.

INTRODUCTION

This Editor/Pass/Assembler is a powerful working tool for software development. The E.P.A. makes machine code programming far more meaningful for the programmer in that, the programmer is able to achieve a greater understanding of overall concept of the objects required for the software package.

An E.P.A. allows greater freedom of programming choice. Repair work to get programs running is minimal as ammendments are easy and any wasteful repair work is no longer required.

Many variations of Assemblers exist in different formats, it is not within the scope of this handbook to teach the use of Assemblers, simply bring to the users attention the facilities offered with this particular software package.

HARDWARE REQUIREMENTS:

Microtan 65 + Tanex +*7K RAM + Xbug
+ Ascii Keyboard + Lower Case Option.

*NB A minimum of 5K is required.

SOFTWARE DETAILS:

Program length 4K

Ram location $0400 - $13FF

Working space:

All Zero Page locations are used with the exception of locations 0098 - 009F & 00B8 - 00BF.

Locations 0100, 0101, 0102 are also used.

After assembly has taken place Zero Page locations 0040 - 00FF are usable.

Filename: EDITOR

Filed: CUTS (300 Baud) X 2.

Start Address: $0400

It would be prudent of the user to keep a master copy of this package in a secure place.

ELEMENTS OF THE ASSEMBLY LANGUAGE

LABELS:

Labels are used for;

(1) Identifying program points mnemonically, or by name, rather than by machine address of the location.

(2) A numerical value which is not a program point.

(3) Automatically updating all references to a program point when its machine address changes due to changes in the program.

A Label consists of one to six characters from the set A - Z & 0 - 9. ('SPACES' are counted as characters!)

The first character in a label must be a letter.
NB. A cannot be used as a label.

There is one predefined label '*' which refers to the program counter.

CONSTANTS:

The Assembler permits the use of three data formats, Ascii - Hexidecimal - Decimal;

With no prefix - decimal constant	eg	171
With '$' prefix - hexidecimal constant	eg	$20
With ''' prefix - Ascii constant	eg	'B

The colon ':' is only required if a label is used.
- The label and comment fields are optional.

```
Eg.   EXAM: LDA  TEST       ; Get Check
            STX  VALUE + 1  ; Save IX
      CONT: SEC
```

DIRECTIVE STATEMENTS:

The fields in a directive statement are:

(Label) : (Directive) (Operand) (Comment)

DEFB

DEFINE BYTE:

```
Eg    DEFB VALUE, 0,$20,'A,100,VALUE-7
```

DEFW

DEFINE WORD: (In Low, High order by byte)

```
Eg    DEFW VALUE, 0, 1000, $412, VALUE+20
```

DEFM

DEFINE MESSAGE:

```
Eg    DEFM  'This is a test'
```

Message enclosed in the quote marks "'" CHR Code $27

EQUALS (=) Directive:

Setting the Program Location Origin

```
Eg      * = $0400
        * = 1400
        * = START
        * = END - 40
```

RESERVING SPACE:

```
Eg      * = * + 20          reserves 20 places
        * = * + $20            "      32    "
        * = * + 7000           "    7000    "
```

EXPRESSIONS:

The Arithmetic operators are;

'+' Add

'-' Subtract

The format is:

(Label)(+ or -)(Constant)

Examples:

START + 1

OPCHR - 2

CHR + $20

VALUE + 'A

ASSEMBLER STATEMENTS:

Comment statements;

- provide headings and other explanatory information.
- these do not offset the machine code generated by the Assembler.

A comment statement is created by a semicolon ';' as the leftmost non blank character followed by any commentary whatever.

```
Eg.       ; ASSEMBLER
          ; **OUTPUT SUBROUTINE**
          ;
```

Instruction Statements;

The fields in an instruction statement are:

(Label) : (Opcode) (Operand) (Comment)

The spaces between each field is optional

EQUATING LABELS TO VALUES:

Eg TEN = 10
EXAM = $400

COMMANDS:

ASSEMBLY COMMANDS:

A0 - FULL ASSEMBLY LISTING DOES NOT WRITE TO RAM

From left to right for each line of sorce code;

- Program location
- Object (Machine) Code
- Source code line number
- Source code

A1 - ERRORS ONLY LISTING DOES NOT WRITE TO RAM

A2 - ASSEMBLE TO RAM WITH OFFSET

After typing A2 (CR) the program responds with
'Offset=' (Offset in HEX)
Now type in your offset value followed by (CR)

Eg If you want a M/C program starting at $1400 where the source code starts, in this case at $1400 and finishes at $1470, we can operate an A2 assembly with Offset 100. This will store the program from $1500 onwards as though it was writing into $1400 onwards where it was intended to go.
- Very useful for relocating programs for Eproms.

A3 - ASSEMBLE DIRECT TO RAM

If any Errors occur in any of the Assembly modes, the source code line is printed.

GENERAL OPERATIONS - EXAMPLES - OTHER FEATURES

As an effective demonstration of the E.P.A. facilities we shall procede through the operations in general.

Activate the program G400 (CR)

On activation the program will respond;

Initialise (Y/N)?

If you reply 'Y' the program will respond;

Source Add= Enter 1400 (CR)

Type in the Hex value of the start address of the source code which as this package resides at 0400 - 13FF must be $1400 or greater.

From this point the program sets up the required parameters for assembly and jumps into command mode and all previous data is destroyed.

WARM START:

If at any time the user wishes to re-enter the program from the Monitor, at 'Initialise (Y/N)? reply 'N', at this point the program will jump to command mode and all previous data is secure.
If by accident you answer 'Y', hit 'Reset' or 'Esc' and re-enter by G400 (CR).

COMMAND FACILITIES:

DELETE

To delete a line, type in the line number followed by (CR).

LIST

L (CR) List source code from the beginning.
L xx (CR) List source code from line number xx to end.
Only eight lines are output at one time, to continue the listing, type any key. To Abort type Cntrl C.

EXAMPLE PROGRAM USING THE E.P.A.

Type in the listing as shown on the right i.e. line 10 - 180. Check for accuracy then apply an AØ command, the completed listing will now appear as shown.

AØ Listing.

```
                        10 ;** DELAY SUBROUTINE **
                        20 ;
                        30 ; Delay = DLYCNT x K
                        40 ; K = CONSTANT
                        50 ; Int PC
                        60 * = $1F00
                        70 ; Int Variables
                        80 DLYCNT = 100
                        90 ;
                       100 ; START
                       110 ;
                       115 ;       SEQUENCE
1F00 A6 64              120 LDX DLYCNT              ; get delay value
1F02 A0 FF              130 LDY #$FF
1F04 88                 140 DLY : DEY
1F05 D0 FD              150 BNE DLY
1F07 CA                 160 DEX
1F08 D0 FA              170 BNE DLY
1F0A 60                 180 RTS
```

An added feature is the use of the 'Line Feed Key' as AØ is making its 'PASS', this facility is to assist the programmer to study the output.

EXAMPLE PROGRAM 2.

```
                    10 ; PSEUDO RANDOM NUMBER GENERATOR
                    20 ;
                    30 ; Shift register length = 31
                    40 ; Feedback points = 28 and 31
                    50 ; Sequence length = 2147483647
                    60 ;
                    70 ; Locations $60 to $63 must be
                    80 ; seeded before using the
                    90 ; routine
                   100 ;
                   110 * = $1F00                    ; int PC
                   120 ; initialise variables
                   130 ;
                   140 REG = $60
                   150 TEMP = $64
                   160 ;
                   170 ; Start of Program
                   180 ;
1F00 A5 63         190 LDA REG + 3                  ; Get bit 28
1F02 29 08         200 AND #8                       ; from reg
1F04 0A            210 ASL A
1F05 0A            220 ASL A
1F06 0A            230 ASL A
1F07 85 64         240 STA TEMP
1F09 A5 63         250 LDA REG + 3
1F0B 29 40         260 AND #$40
1F0D 45 64         270 EOR TEMP                     ; EOR with
1F0F 0A            280 ASL A                        ; bit 31
1F10 0A            290 ASL A                        ; Ans=carry
1F11 26 60         300 ROL REG                      ; Shift all
1F13 26 61         310 ROL REG + 1                  ; bits left
1F15 26 62         320 ROL REG + 2
1F17 26 63         330 ROL REG + 3
1F19 A5 60         340 LDA REG                      ; RND in A
1F1B 60            350 RTS                          ; END
```

PRINTER COMMANDS

P1 - PRINTER ON

PØ - PRINTER OFF

This facility allows the user to jump to their own printer routines which must end in an RTS ($60) command.

Modify $13FO JUMP TO PRINTER ROUTINE

The CHR code to be output is stored in location $02. Without any modification, the program P1 simply causes all print output to the screen to be delayed. A very useful facility for producing full listings.

EDIT & CURSOR CONTROL COMMANDS

CNTRL	U -	Cursor UP
"	D -	" DOWN
"	R -	" RIGHT
"	L -	" LEFT
"	T -	MOVE ALL CHARACTERS RIGHT OF THE CURSOR ONE PLACE TO THE RIGHT.
"	I -	TAB CURSOR 21 PLACES TO THE RIGHT OF THE MARGIN.
"	E -	ERASE CHARACTER INDICATED ABOVE CURSOR.

CASSETTE FILE HANDLING

When in E.P.A. command mode:

C (CR) - Sets Cuts 300 baud.
F (CR) - Sets Fast 2400 baud.
D (CR) - Sets Dump command and computer replies with
  D (Start add), (End add), ?
  Enter your filename followed by (CR).
After the dump is completed the program returns to the monitor. Fetch and Examine commands are the same as Xbug, these also return to the monitor after completion.

The Fetch command (F,Filename) does not set up the required parameters for the program to function.
To load a program follow this procedure:

F, (Filename) (CR)
Computer replies: (Filename).A
On loading: (Filename).A xxxx (address)
Re-enter the E.P.A. in the normal manner (G0400) and answer 'Y' to Initialise and enter 100 to 'Source add'.

Now type Ixxxx (address) (CR)
The procedure is now complete.

```
E.g.     F,GAME (CR)
         GAME   .A
         GAME   .A   xxxx
         G0400 (CR)
         Initialise (Y/N) Y
         Source add=100 (CR)
         Ixxxx (CR)
```

On completion the command mode is set, normal functions resume.

If required and if the program has been assembled to Ram, the normal routine under monitor commands may be used for cassette handling.