Series: Real Time System Studies.

Title: Program Documentation.

Author: C.A.R. Hoare.

Date: April, 1968.

Introduction.

One of the major problems in the construction, check-out and maintenance of a large program is that of adequate documentation of the code. Aids to documentation, and means of enforcing the required disciplines on programmers, have been developed over the years. Among these are the adoption of symbolic programming languages, macro-languages, high level languages, etc. Unfortunately the use of the higher level languages has led to a decrease in efficiency in object programs, and even more seriously, to an expansion in their size.

This report makes an initial attempt to investigate alternative methods of documenting the same program, and examine the penalties involved in each technique. The program segment chosen as an example was taken from real life, but cannot necessarily be regarded as typical, since it involves mathematical computations.

report of Computing Research Division
Elliott Brothers Ltd.
The investigation is not complete, but is published partly as an illustration of the method of quantitative and factual analysis which should be applied by those engaged in research into machine and software design.

Summary.

Appendix 1 shows a flowchart for the program. Note that the boxes contain only "comments" explaining what needs doing and why. They do not contain any indication of how the tasks are to be performed.

Appendix 2 shows the same flowchart using begins, ends and indentations instead of arrows and boxes. If a "flowchart" is to be input into the computer, this is the form it should take. It may be found that the indented form is just as clear as the pictorial form, and might replace it as a standard documentation technique.

Appendix 3 shows how that data for a program might be fully annotated. For each variable name we indicate the purpose for which it is used, and list all the comments which refer primarily to it. To the right of the line, we give the actual declaration of the quantity, and the actual symbolic coding corresponding to each comment. It might be helpful also to have a cross-reference listing of all the places where the variable is used or has its value changed by assignment.

Appendix 4 shows the program written in "pure" ALGOL. All the comments have been inserted, using the PL/I comment convention /*........*/. A program like this could be produced automatically from the material of the previous two appendices, using the comments as macro-names and the coding as macro bodies. However, this would be a difficult operation on a small computer, since it is an essential prerequisite of good documentation that the comments should be written out in full. Furthermore, the resulting code would be very diffuse, since it makes no attempt to "bind" the code together by storage of temporary
results, use of registers, etc. Finally, a systematic use of program generation in this way is likely to lead to frequent failures to set variables properly before their use.

Appendix 5 shows the same program (unannotated) written in ALGOL, but using the letters A and B to stand for the accumulator and the B-register. This is very useful in the production of efficient object code. However, the values of A and B are frequently changed as a side effect of other instructions which do not mention them, and the programmer runs a constant risk of forgetting this.

Appendix 6 shows a form of infix notation which mirrors exactly the structure of the machine code. Its advantages over machine code are:

1. Bracketed and indented techniques can be used (although in this example they are not).
2. More than one related instruction can be placed on a single line.
3. Infix notations are more familiar and pleasant in general than prefix notations of machine code.

This code was the first attempt to understand the original machine code. It may be worth while to do more work on this form of low-level notation.

The actual SIR coding for this program is also available for inspection.

As an additional study of the code, the following figures were obtained:

Of 253 instructions, there were
32 b-register loads
59 b-lined instructions
50 jumps.

112 other instructions.

If a second B-register had been available 15 instructions would have been saved; if yet a third b-register had been available, a further 5 instructions would be saved. If the B-register were wholly volatile, (i.e. always had to be reloaded before use), an additional 25 instructions would be required.

Conclusion.

No firm conclusions are available at the present stage. Numeric estimates should be made of the penalties of using each of the proposed notations. The example program should be used to test the characteristics of proposed future machine designs, and software designs.

One possibility arises from the study of optimisation, that perhaps the hardware should have one entirely volatile accumulator and one entirely volatile B-register. These would be used exclusively by the translator of a high-level language. Any other registers available may be put under the control of the programmer by providing standard identifiers, as in Appendix 5.
Appendix 1. TRACKING PROGRAM: FIGHTER MODULE.
Flow chart in conventional notation.

ETRACK \rightarrow \text{set console number from keyboard}

\begin{align*}
\text{set internal track number} \\
\text{track is not in system} \quad \text{TRUE} \rightarrow \text{WRSEQ} \\
\text{FALSE} \\
\text{this is not a vertical point} \quad \text{TRUE} \rightarrow \text{start has been failed} \\
\text{FALSE} \rightarrow \text{FSEQC} \\
\text{clear fade flag} \quad \text{TRUE} \\
\text{track has been sequenced in this control period} \\
\text{FALSE} \rightarrow \text{FSEQC} \\
\text{this is a fighter} \quad \text{TRUE} \rightarrow \text{ENEWTRACK} \\
\text{FALSE} \rightarrow \text{ETRACK}
\end{align*}
T
X
Y
Z

- fade flag is set
  - TRUE
    - confidence is zero
      - TRUE
        - clear fade flag
      - FALSE
        - reduce confidence
        - ESEQ
  - FALSE
    - store injected coordinates

- target is newly initiated
  - TRUE
    - clear confidence factor
  - FALSE
    - calculate discrepancy between injected and predicted coordinates.

- target is within gate
  - TRUE
    - confidence is nonzero
      - TRUE
        - increase confidence
      - FALSE
        - ESEQ
  - FALSE
    - smooth coordinates

- pack data

- calculate new velocity

- calculate and display heading and mesh numbers

- reduce confidence factor by two
Appendix 2. TRACKING PROGRAM: FIGHTER MODULE  
Flow chart in ALGOL notation.

EFADE:  set fade flag;

ETRAK:  set console from keyboard;
         set internal track number;
         if track is not in system then go to WRSEQ  
         if this is not a vertical point then
                           begin  if this start has been faded then go to ESEQC;
                                             clear fade flag;
                                             if track is newly initiated then go to ELSFADE;
                                             else  to  to ENEWTRAK
                           end;
         
         if this track has been sequenced in this aerial period
         then
                           begin  clear fade flag;
                           go to ESEQC
         end;
if this is a fighter then
    begin if fade flag is set then
        begin if target was outside gate size
            then go to SETFREPRO
        else go to ACLEAR
    end;

store injected coordinates;
if associated target has been cancelled then
    begin if this fighter is not in recovery segment then
        begin set coordinates to injected values;
            go to ACLEAR
        end;
    end;
calculate discrepancy between injected and predicted coordinates;
if fighter is within fighter gate size then
    begin if target was outside gate size
        then go to SETFREPRO
    else go to ACLEAR
end;
set coordinates and ddt table to injected value;
pack ddt;

SETFREPRO: if reprofile flag is clear then
    begin set reprofile flag for this intercept;
        decrement count of reprofiles waiting
    end;

ACLEAR; clear flaga;
clear fade flag;
go to ESEQC
end fighter
else
begin comment this is a target;
   if fade flag is set then
      begin if confidence is zero then clear fade flag;
                  else reduce confidence;
      go to ESEQC.
   end;
store injected coordinates;
if this target is newly initiated then clear confidence factor
   else begin calculate discrepancy between injected and
           predicted coordinates;
               if target is within gate then
                  begin if confidence is non zero then increase
                      confidence;
                      go to ESEQC
               end;
   end;
smooth coordinates;
pack ddt;
calculate new velocity;
calculate and display heading and mach numbers;
reduce confidence factor by two;
if fighter is off the airfield then
   begin set flag a;
       go to ESEQC
   end;
if fighter has not been allocated then go to ESEQC;
if intercept reprofile flag is set then go to ESEQC;
set reprofile flag;
decrement count of repfiles waiting;
go to ESEQC
end;
<table>
<thead>
<tr>
<th>Elaga [y console] = 0</th>
<th>Clear Elaga</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elaga [y console] = 0</td>
<td>Target was outside gate size</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Elaga (0, 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elaga [0:2] =</td>
</tr>
</tbody>
</table>

Target was outside gate size. Indicating that target was outside table of Elaga for each console.

Appendix 3: Correlation of comments and code.

Page 13.
68/43/0
| [T̅] target 1 \text{ within gate}. & 2. |
| \text{Gate size}. & 2. |
| Target is within trigger & 2. |
| \text{Trigger is within gate}. & 1. |

\begin{tabular}{|l|}
\hline
\( +1.49/3.5, +3.5/35, +2.4/5 \) \\
\( +1.37/2.5, +5/7, +6/5, +7/5, +2/5, +1.25/5 \) \\
\( +1.7/1 \) \\
\hline
\end{tabular}
injected x,  
injected y;

integer injected x,
injected y;

store injected coordinates.

injected x : = v rollx [ v console]
injected y : = v roolly [ v console]

for a = 1
<table>
<thead>
<tr>
<th>fade flag</th>
<th>integer fade</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>set fade flag.</td>
</tr>
<tr>
<td>2</td>
<td>clear fade flag.</td>
</tr>
<tr>
<td>0</td>
<td>fade flag is set.</td>
</tr>
<tr>
<td>&lt; 0</td>
<td>fade</td>
</tr>
</tbody>
</table>

flag indicates that the point is to be faded.
<table>
<thead>
<tr>
<th>v console</th>
<th>number of console which has injected the current trace.</th>
<th>integer v console.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>set console from keyboard</td>
<td>v console: = ckbno -1</td>
</tr>
<tr>
<td>vtrackno</td>
<td>number of current track</td>
<td>integer vtrackno.</td>
</tr>
<tr>
<td>---------</td>
<td>-------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>.1</td>
<td>set internal track number</td>
<td>vtrackno: = tabtrak [ptrak [vconsole]]</td>
</tr>
<tr>
<td>.2</td>
<td>track is not in system.</td>
<td>vtrackno &lt; 0</td>
</tr>
<tr>
<td>.3</td>
<td>this is a fighter.</td>
<td>vtrackno &gt; 11</td>
</tr>
<tr>
<td>t seqc</td>
<td>table of times of last sequencing for each track.</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>t seqc</td>
<td>[0:23]</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>vtrackno</th>
<th>tseqc (vtrackno)=tud time [vtrackno]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>track has been sequenced in this aerial period.</td>
</tr>
</tbody>
</table>
tx, ty.  

<table>
<thead>
<tr>
<th>tables of x and y coordinates for each track.</th>
</tr>
</thead>
<tbody>
<tr>
<td>tx [0:23], ty [0.23]</td>
</tr>
</tbody>
</table>

1. set coordinates to injected values.
   tx[vtrackno]:= injected x;
   ty[vtrackno]:= injected y;

2. set coordinates and ddt table to injected value.
   tx[vtrackno]:= bddt3[2]:= injected x;
   ty[vtrackno]:= bddt3[3]:= injected y;

3. smooth coordinates.

   begin integer smoothing factor, B;
   B:= vtrackno; smoothing factor:= tpos[vcon[B]];
   bddt3[2]:= tx[B] = ((.5 - smoothing factor) * injected x
   + smoothing factor * tx[B]) left 1;
   bddt3[3]:= ty[B] = ((.5 - smoothing factor) * injected y
   + smoothing factor * ty[B]) left 1;

   end.
<table>
<thead>
<tr>
<th>bddt [0:3]</th>
<th>?</th>
</tr>
</thead>
<tbody>
<tr>
<td>.1 pack ddt.</td>
<td>bddt 3[1]: = vtrackno left 7 and 139944 +5;</td>
</tr>
<tr>
<td></td>
<td>PERIPH (4206)</td>
</tr>
<tr>
<td>vkon</td>
<td>table of confidence factors for non fighter targets. values range up to 7.</td>
</tr>
<tr>
<td>------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td>.1</td>
<td>this target is newly initiated.</td>
</tr>
<tr>
<td>.2</td>
<td>clear confidence factor.</td>
</tr>
<tr>
<td>.3</td>
<td>confidence is non zero.</td>
</tr>
<tr>
<td>.4</td>
<td>confidence is zero.</td>
</tr>
<tr>
<td>.5</td>
<td>reduce confidence.</td>
</tr>
<tr>
<td>.6</td>
<td>reduce confidence by two.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Appendix 4. TRACKING PROGRAM: FIGHTER MODULE.
Fully annotated in pure ALGOL.

begin integer injected x, injected y, vtrackno, v console;

EFADE:  fade := -1; /* set fade flag*/
ETRAK:  v console := ckbno -1/*set console from keyboard*/
        vtrackno := tabtrak [ptrak[v console]]; /* set internal track number */
      
if vtrackno < 0 then /* track is not in system*/
      go to WRSEQ;

if vidcount [v console] > 0 then /* this is not a vertical point. */

begin if fade > 0 then go to ESEQC;

/* this start has been faded */
fade := 0; /* clear fade flag*/

if finit [v console] > 0 then go to ENEWTRAK
/* track is newly initiated */
else go to ESLFADE

end;

if tseqc [vtrackno]=tudtime [vtrackno] then
/* track has been sequenced in this aerial period*/

begin fade := 0; /* clear fade flag*/

go to ESEQC

end;

if vtrackno > 11 then /* this is a fighter*/

begin integer injected x, injected y, discrepancy;

if fade > 0 then /* fade flag is set*/

begin if flaga [v console] > 0 then
/* target was outside gate size*/

go to SETFREPRO

else go to ACLEAR

end;
injected x := vrollx[v console];
injected y := vrolly[v console];/* store injected coordinates*/
if tabtrak[ptrak[v console]-1](0) then
   /* associated target has been cancelled */
   begin if tsegno[vtrackno - 12] ≠ 18 then
      /* this fighter is not in recovery segment */
      begin tx[vtrackno] := injected x;
      ty[vtrackno] := injected y;
      /* set coordinates to injected values */
      go to ACLEAR
   end;
end;

LDISCREP; /* calculate discrepancy between injected and predicted coordinates */
if discrepancy ≠ tgate [-1] then
   /* fighter is within fighter gate size */
   begin if flaga[v console] (0) then
      /* target was outside gate size */ go to SETFREPRO
   else go to ACLEAR;
   end;

   tx[vtrackno] := bddt3[2]; = injected x;
ty[vtrackno] := bddt3[3]; = injected y;
/* set coordinates and ddt table to injected value */
bddt3[1]; = vtrackno left 7 and 130944 + 5;

PERIPH (4206); /* pack ddt */

SETFREPRO: B; = store 2 + vtrackno;
   if frepro[B - 12] = 0 then
      /* reprofile flag is clear */
      begin increment (frepro[B - 12]);
      /* set reprofile flag for this intercept */
      frepro[store 2 - 1]; := frepro[store 2 - 1] - 1;
      /* decrement count of reporfiles waiting */
ACLEAR: flaga [v console] := 0; /* clear flaga*/
    fade := 0; /* clear fade flag*/
end fighter;
/* this is a target*/
else if fade (0 then /* fade flag is set*/
    begin if vkon [vtrackno] = 0 then /* confidence is zero*/
        fade := 0 /* clear fade flag*/
        else vkon [vtrackno] := vkon[vtrackno]-1;
        /* reduce confidence*/
    go to ESBQC
end;

injected x := vrollx [vconsole];
injected y := vroly [vconsole]; /*store injected coordinates*/
if vkon [vtrackno] (0 then /*this target is newly initiated*/
    vkon [vtrackno] := 0 /* clear confidence factor*/
else begin LDISCREP;
    /* calculate discrepancy between injected and predicted coordinates */
    if discrepancy < tgate [vkon[vtrackno]]then
        /* target is within gate */
        begin if vkon [vtrackno] ≠ 0 then
            /* confidence is non zero */
            vkon [vtrackno] := vcon[vtrackno] + 1;
            go to ESBQ
end;
begin integer smoothing factor, B;
    B := vtrackno;
    smoothing factor := tpos [vcon[B]];
end; /* smooth coordinates*/
bdtt3[1] := vtrackno left 7 and 130944 + 5;
PERIPH (4206); /* pack dtt */
begin integer dt, new unsmoothed x velocity, velocity smoothing
  factor, new unsmoothed y velocity, B;
  B := vtrackno;
  dt := tude [B] - tpctude [B];
new unsmoothed x velocity := (tprex [B] - injected x)/dt;
velocity smoothing factor := trel [vcon[B]];  
txvelo[B] := (new unsmoothed x velocity * velocity smoothing
  factor + (.5 - velocity smoothing factor) * txvelo
  [B]) left 1;
new unsmoothed y velocity := (tprey [B] - injected y)/dt;
tyvelo [B] := (new unsmoothed y velocity * velocity smoothing
  factor + (.5 velocity smoothing factor)*tyvelo
  [B] left 1;
end;  /* calculate new velocity*/;

LMHDIS; /* calculate and display heading and mach numbers */
A := vkon [vtrackno] - 2;
vkon [vtrackno] := if A > 0 then A else 0;
/* reduce confidence factor by two*/
if tabtrak [ptrak[vconsole]+1] > 0 then
/* fighter is off the airfield*/
begin flaga [vconsole] := -1; /* set flaga */
go to ESEQC
end;

if tote [tabtrak[ptrak[v console]]+12] = 0 then
/* fighter has not been allocated */
go to ESEQC
if frepro [store 2 + vtrackno] = 0 then
/* intercept reprofile flag is set */
go to ESEQC
increment (frepro [store 2 + vtrackno]);
/* set reprofile flag */
frepro [store 2 - 1] := frepro [store 2 - 1] - 1;
/* decrement count of profiles awaiting recalculation */
end.
Appendix 5. TRACKING PROGRAM: FIGHTER MODULE.
ALGOL subset with register optimisation.

begin injected x, injected y, vtrackno, v console;

EFADE: fade = -1;
ETRAK: B = v console = ckbno - 1;
vtrackno:=A = tabtrak [ptrak[B]];
if A < 0 then go to WRESEQ;

B = v console;
if vidcount [B] ≠ 0 then
begin if fade < 0 then go to ESEQC;
fade = fade + 1;
if finit [B] then go to ENEWTRAK
else go to ESLFADE
end;

A = tudtime [vtrakno] - tseqc [vtrakno];
if A = 0 then
begin fade = A;
go to ESEQC
end;

if vtrakno > ll then
begin comment FIGHTER;
integer injected x, injected y, discrepancy;
if fade < 0 then
begin if flaga [v console] then go to SETPREPRO
else go to ACLEAR
end;

end;
B: = v console;
injected x: = vrollx [B];
injected y: = vrolly [B];
if tabtrak [ptrak[B]-1] (0 then
   begin comment CANTGT;
      B: = vtrackno;
      if tsegno [B-12] ≠ 18 then
         begin tx [B]: = injected x;
            ty [B]: = injected y;
            go to ACLEAR
         end;
      end;
end;
LDISCREP;
if tgate [-1] \ discrepancy then
   begin if flaga [v console](0 then go to
   SETREPO
      else go to ACLEAR
   end
B: = vtrakno;
rx[B]: = bddt3[2]: = injected x;
ry[B]: = bddt3[3]: = injected y;
bddt3[1]: = vtrakno left 7 and 130994 + 5;
PERIPH (4206);

SETREPO:  B: = store 2 + vtrakno;
if frepro [B - 12] ≠ 0 then
   begin frepro [B - 12]: + 1;
      B: = store 2;
      frepro [B - 1]: = frepro [B - 1]-1
   end;
ACLEAR:   flaga [v console]: = fade : = 0;
go to ESEQC;
end fighter;
else
  if fade (0 then begin
    comment DBCRK;
    B: = vtrackno;
    A: = vkon [B]
    if A = 0 then begin fade: = A; go to ESEQC
    end;
    vkon (B): = A - 1;
    go to ESEQC
  end;

B: = v console;
injected x:= vrollx [B];
injected y:= vrolly [B];
B: = vtrackno;
else begin if v con [B] < 0 then v con [B]: = 0
  LDISCREP;
  A: = discrepancy - t gate [vcon[vtrackno]];
  B: = vtrackno;
  if A < 0 then begin comment INCRK; if vcon [B] /\ 0 then
    vcon [B]: = 1;
    go to ESEQC
  end;

begin integer smoothing factor, temp;
B: = vtrackno;
A: = smoothing factor : = tpos [vcon[B]];  
  temp: = (65536 - A) *tx [B];
  bddt3[2]:=tx[B]:= (smoothing factor * injected x +
    temp) left 1;
  temp: = (65536 - A) *ty [B];
  bddt[3]:=ty[B]:= (smoothing factor * injected y +
    temp) left 1;
end;
bdtt3[1] := vtrackno left 7 and 13094 + 5;
PERIPH (4206);
begin
integer dt, new unsmoothed x velocity, velocity
smoothing factor, new unsmoothed y velocity, temp;
B := vtrackno;
dt := tuddtime[B] - tpredtime[B];
new unsmoothed x velocity := (t prex[B] - injected x) / dt;
A := velocity smoothing factor := trel[vcon[B]]
B := vtrackno;
temp := (65536 - A) * txvelo[B];
ftxvelo[B] := (velocity smoothing factor * new unsmoothed
x velocity + temp) left 1;
new unsmoothed y velocity := (t prey[B] - injected y) * dt;
temp := (65536 - velocity smoothing factor) * tyvelo[B]
tyvelo[B] := (new unsmoothed y velocity * velocity
smoothing factor + temp) left 1;
end;
LMHDIS;
B := vtrackno;
A := vcon[B] - 2;
vcon[B] := if A > 0 then A else 0;
B := ptrak[v console]
if tabtrak[B + 1] > 0 then
begin
flaga[v console] := -1;
go to ESEQC
end
if tote[tabtrak[B] + 12] = 0 go to ESEQC;
B := store 2 + vtrackno;
if frepro[B] < 0 then go to ESEQC;
frepro[B] := + 1;
B := store 2;
frepro[B - 1] := frepro[B - 1] - 1;
go to ESEQC
### Unoptimised Object Code

<table>
<thead>
<tr>
<th>EFADE</th>
<th>4</th>
<th>-1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5</td>
<td>FADE</td>
</tr>
<tr>
<td>ETRAK</td>
<td>4</td>
<td>CKEYBOARD</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>VCONSOLE</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>VCONSOLE</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>PTRAK</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>TABTRAK</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>VTRACKNO</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>T1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>VCONSOLE</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>VIDCOUNT</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>VCONSOLE</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>FADE</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>ESEQC</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>FADE</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>PINIT</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>ENEWTRAK</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>ESLFADE</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>VTRACKNO</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>TSEQ</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>VTRACKNO</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>TUIDTIME</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>(</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>+ O</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>FADE</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>ESEQC</td>
</tr>
</tbody>
</table>

- set fade flag
- set console from keyboard.
- set internal track number.
- This is not a vertical point
  - FALSE
- This start has been faded.
  - FALSE
  - TRUE
- clear fade flag
- track is newly initiated.
  - TRUE
  - FALSE
- track has been sequenced in this aerial period.
  - TRUE
Appendix 6. Infix notation for machine code.

EFADE: -1 =: FADE;
ETRAK: CKN0 -1 =: VCONNO =: B;
TABTRAK(ETRAK(B)) (O go to WRSEQ =: VTRAKNO;
VIDCOUNT (VCONNO) # O (FADE ) O go to ESEQC;
FADE: + 1; FINIT (B) (O (go to ELSFADE else go to
TSEQC(VTRAKNO) O TUDTIME (B) = 0 (=:FADE go to ESEQC);

READ: VTRAKNO - 11 (O go to fighter.
FADE (O go to DECK;
VROLLX(VCONNO) =: W3S(4);
VROLLY(B) =: W3S(5);
VKON(VTRAKNO) ) O go to KIN IT
call LDISCREP;
TGATE(VKON(VTRAKNO)) O W3S [VTRAKNO =: B](O go to INCREK

KIN IT: O =: VKON(B);
(TPOS(VKON(B))=: W3S O 65536) *TX(VTRAKNO) =: W3S - (1);
W3S*W3S (4) + W3S (1) left 1 =: TX(B) =: BDDT 3(2);
(W3S O 65536) * TY (B) =: W3S (1);
W3S * W3S(5) + W3S (1) left 1 =: T7(B) =: BDDT 3(3);
VTRAKNO left 7 and 130944 + 5 =: BDDT 3(1);
PERIPH (4206);
TPRETUDT (B) O TUDTIME =: W3S (1)
W3S(4) O TPREDX (B) O (clear auxiliaric register)
O /W3S(1) =: W3S(3);
TVEL (VCON(B)) =: W3S O 65536 * TXVELO(TRAKNO) =: W3S(2);
W3S(3) *W3S + W3S (2) left 1 =: TXVELO (B);
W3S(5) O TPREDY (B) O 0 / W3S (1) =: W3S (3);
W3S O 65536 * TYVELO (B) =: W3S (2);
W3S(3) *W3S + W3S (2) left 1 =: TYVELO (B);
call LMHDIS;
VKON(VTRAKNO) -2 )0 (=: VKON (B) else 0=: VKON(B));
TABTRAK (PTRAK (VCONNO) +1) ) 0
(-1 =:FLAGA (VCONNO); go to ESEQC);

TESTOTO: TOTE (TABTRAK (B) + 12) = 0 go to ESEQC;
STORE 2 + VTRAKNO =: B;
FREPRO (B) = 0 go to ESEQC;
FREPRO (B): +1;
FREPRO (STORE 2 - -) -1 =: FREPRO (B-1);
go to ESEQC
DECRX: VKON (VTRAKNO) = O go to ACLEAR (3); -1 =: VKON (B);

FADE: +1; go to ESEQC;

INCRX: VKON (VTRAKNO) = O go to ESEQC;

VKON (B) =: +1; go to ESEQC;

FIGHTER: FADE (O go to READF);

VROLLX (VCONNO) =: W3S (4);

VROLLY (B) =: W3S (5);

TABTRAK (PTRAK (B) -1) (O go to CANTGT

FIGHTERIO: call LDISCREP;

tGATE (-1) @ W3S (O go to READF;

B: = VTRAKNO;

W3S (4) =: TX (B) =: BDDT3 (2);

W3S (5) =: TY (B) =: BDDT3 (3);

VTRAKNO left 7 and 130944 + 5 =: BDDT3 (1);

PERIPH (4206);

SETFREPRO: STORE 2 + VTRAKNO =: B;

FREPRO (B -12) = O go to ACLEAR;

FREPRO (B -12) =: +1;

B: = STOREZ;

-1 + FREPRO (B -1) =: FREPRO (B -1)

ACLEAR: B: = VCONNO;

O =: FLAGA (B) =: FADE;

go to ESEQC;

READF: FLAGA (VCONNO) (O go to SETFREPRO

else go to ACLEAR;

CANTGT: TSEGNO (VTRAKNO -12) -18 = O go to FIGHTERIO;

W3S (4) =: TX (B);

W3S (5) =: TY (B);

go to ACLEAR.