GEODÆTISK INSTITUTS INTERNE RAPPORT NR. 14
THE DANISH GEODETIC INSTITUTE
INTERNAL REPORT NO. 14

Appendix 2.

REFERENCE MANUAL
for
GPU

and

REFERENCE MANUAL
for
DPU

K. Engsager

1983

ISBN 87 7450-048-1

GEODÆTISK INSTITUTS INTERNE RAPPORT NR. 14
THE DANISH GEODETIC INSTITUTE
INTERNAL REPORT NO. 14

Appendix 2.

REFERENCE MANUAL
for
GPU

and

REFERENCE MANUAL
for
DPU

K. Engsager

1983

ISBN 87 7450-048-1

Contents

II

Reference Manual

for

GPU

K. Engsager

Geodætisk Institut

1983

REFERENCE MANUAL for GPU

General Processing Unit designed for double precision floating-point arithmetic after Wilkinson. The double precision orders are processed by the DPU (Double Processing Unit).

Basic model is the CPU811.

The instructions are extended and modified.

Modifications :

The GPU will turn off the MONITORMODE and ESCAPEMODE.
The GPU tests that the instructions are taken from an area limited by code_low and code_top given in the message to the GPU.
No registers are dumped in case of errors.

The following instructions are illegal :

illegal	: di	29	data in	illegal
illegal	: do	1	data out	illegal
illegal	: gg	28	general get	illegal
illegal	: gp	47	general put	illegal
illegal	: jd	14	jump disable	illegal
illegal	: je	15	jump enable	illegal
illegal	: re	22	return from escape	illegal
illegal	: ri	12	return from interrupt	illegal

The following instructions are changed :

changed	: jl	13	jump link	changed

The following instructions are new :

new	: gpustop	0		new
new	: gpusqrt	1		new
new	: gpucmvs	29	vector sub (vector mult constant)	new
new	: gpuadd	30		new
new	: gpusub	31		new
new	: gpucmva	47	vector add (vector mult constant)	new
new	: gpumla	58	repetive mult add	new
new	: gpuarm	59	mult accumulator	new
new	: gpuinv	60	invert	new
new	: gpustr	61	store	new
new	: gpured	62	autoreduction	new
new	: gpuinit	63		new

The GPU can not be autoloaded.

The following procedures are changed or new :

procedure get address;
escapemode is always false.

```
procedure getdwd;
-----------------
begin
  getdouble1;
  getdouble2;
end;


procedure get instruction;
--------------------------
comment instructions must be found between code_low and code_top,
        else is a jump made to codeviolation;
begin
  if addr(23)=1 then addr := addr-1;
  if code_low <= addr + base < code_top then
  begin
    aq := addr + base;
    goto ifetch;
  end
  else
    goto code_violation;

  ifetch:  instruction := word(aq);
  if buserror then goto test_bus;
  pic := aq(1:23);
  after ifetch:
  fetchedq := true;
end;

procedure normalize and round;
------------------------------
comment only the unflow/overflow routine has been changed :
if exp(0:11) <> signextended expq(12) then
begin
  if expq > 0 then
  begin
    ex(22) := 1;
    if floating point exception active then
    goto floating point exception;
  end
  else
    dregw := 0.0; <* underflow *>
end;


procedure set busexception;
---------------------------
Does not exist.
```

```
procedure exit to program;
--------------------------
Does not exist.

procedure next instruction;
---------------------------
begin
  if interrupt flag then goto external interrupt;
  if fetchedq then fetchedq := false else
  begin
    pic := pic + 2;
    if code_low <= pic < code_top then
    begin
      instruction := word(pic);
      if buserror then goto testbus;
    end
    else goto code_violation;
  end;

  F := instruction(0:5);
  W := instruction(6:7);
  M := instruction(8:9);
  X := instruction(10:11);
  D := signextended instruction(12:23);

  aswitchq := after am con M;
  goto case aswitchq of
    (  direct,   indirect,   relative,   relative indirect,
     amdirect, amindirect, amrelative, amrelative indirect);

  goto case F of
    ( gpustop, gpusqrt, bl, hl, la, lo, lx, wa,
      ws, am, wm, al, illop, jl, illop, illop,
      xl, bs, ba, bz, rl, sp, illop, rs,
      wd, rx, hs, xs, illop, gpucmvs, gpuadd, gpusub,
      ci, ac, ns, nd, as, ad, ls, ld,
      sh, sl, se, sn, so, sz, sx, gpucmva,
      fa, fs, fm, ks, fd, cf, dl, ds,
      aa, ss, gpumla, gpuarm, gpuinv, gpustr, gpured, gpuinit);

end;

procedure test interrupt;
comment interrupt during dpu working;
begin
  work2 := interrupt level;
  if work2 = 0 then goto power up;
  clear interrupt;
  if work2 >= 8 then
  begin
    wait 77 microsec; <* dpu must be finished *>
    opq := work2;
    goto external interrupt;
  end;
end;
```

```
procedure wait dpu;
comment to finish operation;
begin
 WDPU:
  if interrupt flag then test interrupt;
  if dpu not ready then goto WDPU;
  q := dpu status(22:23);
  status := status + q;
  if q <> 0 and floating point exception active then
  goto floating point exception;
end;


procedure store opq;
comment opq is stored in word pointed by addr inside writelimits;
begin
  if lowlim <= addr + base < uplim then
  begin
    word(addr + base) := opq;
    if buserror then goto operand error;
  end
  else
  goto program exception;
end;


procedure store dopq;
comment store oplq con opq in words (addr-2) con addr;
begin
  if lowlim <= addr + base < uplim then
  begin
    word(addr + base) := opq;
    if bus_error then goto operand error;
    if lowlim <= addr + base - 2 then
    begin
      word(addr + base - 2) := opq;
      if buserror then goto operand error;
    end
    else
    begin
      addr := addr - 2;
      if addr < 8 then reg(addr) := oplq
      else goto program exception;
    end;
  end;
  else
  if 0 <= addr < 8 then
  begin
    reg(addr) := opq;
    addr := addr - 2;
    reg(addr) := oplq;
  end
  else
  goto program exception;
end;
```

```
procedure square root;
comment diagonal element in cholesky reduction;
begin
  opq := SP16 and 8.2000,0000; <* no clear, normalized store *>
  gpu init1;
  <* gpu init returns to next microinstruction when called
     as procedure *>
  regw := 6;
  gpu str; <* w2 con w3 := accumulator *>
  <* gpu str returns to next micro when called as procedure *>

  work4 := sign_extended(work4(12:23));
  work2 := sign_extended(w3(12:23)) - work4;
  <* exp loss *>

  work3 := SP17 + 22; <* n0 *>
  addr := work3 - 2; <* b4 *>
  get word; <* status base *>
  work5 :=
  addr  := opq + w1 + w1; <* status addr *>

  op1q := 1; <* normal dia result *>
  if regw > 0 then
  begin
    opq := opq;
    store dopq;
    addr := work3 - 4; <* b6 *>
    get word;
    if opq > work4 <* exp_lim > exp_loss *> then
    begin
      gpu sqrt;
      <* gpu sqrt returns to next microinstruction when called
         as procedure *>
      regw := SP16 and 8.3000,0000; <* no clear *>
      gpu init1;
      <* gpu init return to next micro when called as procedure *>
      op1q := w2;
      opq  := w3;
      regw := RTC; <* addr_A = store addr *>
      gpu inv1;
      <* gpu inv returns to next micro when called as procedure *>
    end
    else
    begin
      op1q := 3; <* diaresult 3 : zero by exp_loss *>
      dia exception;
    end;
  end
  else
  begin
    op1q := if result = 0 then 4 else 3;
    opq  := 0; <* exp_loss *>
    dia exception;
  end;
  work3 := SP17 - 4;
  goto BFIC1;
end;
```

```
procedure dpu error;
begin
  status := status + q; <* q = exceptionbits *>
  if floating point exception active then
  goto floating point exception;
end;


procedure dia exception;
begin
  addr := work5; <* status addr *>
  store dopq;
  oplq := 0;
  opq  := 1 shift 11; <* 0.0 *>
  addr := RTC; <* addr_A = store addr *>
  store dopq;
end;
```

```
Instruction excecution.
**********************

aa:  see (1).
ac:   do
ad:   do
al:   do
am:   do
as:   do
cf:   do
ci:   do
di:  <* illegal *> goto illop;
dl:  see (1).
do:  <* illegal *> goto illop;
ds:  see (1).
ea:   do
el:   do
es:   do
fa:   do
fd:   do
fm:   do
fs:   do
gg:  <* illegal *> goto illop;
gp:  <* illegal *> goto illop;

gpuadd:
comment "double floating point add to accumulator"
         ----------------------------------------
If addr = 0 then (the double floatingpoint dreg(w) con 0.0) else
(the double floatingpoint dreg(w) con dreg(w+addr)) is added to
the DPU accumulator AR by the sign given in MODUS (see gpuinit).
If MODUS(1) = 0 then positive sign else negative sign.
Numeric code :    30;
begin
  work3 := 0;
  gpu_add2:
  if addr = 0 then
  begin
   opq  := 0;
   oplq := 0;
  end
  else
  begin
   addr := addr + regw;
   getdwd;
   opq(12:23) := work3(12:23);
  end;
  start dpu11;
  addr := regw;
  add2:
  getdw;
  start dpu;
  ex(22:23) := 0;
  wait dpu;
  goto next instruction;
end;
```

```
gpuarm:
comment "set accumulator equal to
         accumulator multiplied by floating point"
         ---------------------------------------
The floating point dreg(addr) is multiplied by the DPU accumulator
and the result is placed in the DPU accumulator.
Numeric code :    59;
begin
  getdwd;
  start dpuarm;
  ex(22:23) := 0;
  wait dpu;
  got next instruction;
end;


gpucmva:
comment "constant mult vector add vector and store in last vector"
         ------------------------------------------------------
  The contant given by the effective address is multiplied by the
  doubleword poointed by regpre an added to the doubleword pointed
  by regw. The doubleword result is stored in the doubleword pointed
  by regw. regpre and regw is increased by four. The register w0
  gives the number of repetitions.
Numeric code :    47;
begin
  CAUSE := 0;
  ILIM  := 8.4000_4000; <* clear  AR, blexp = -2048 *>
  goto cmv;
end;

gpucmvs:
comment "constant mult vector subtract from vector
         and store in last vector"
         ----------------------------------------
  The constant given by the effective address is multiplied by
  the doubleword pointed by regpre and subtracted from the double-
  word pointed by regw. The doubleword result is stored in the
  address given by regw. regpre and regw is increased by four.
  The register w0 gives the number of repetitions.
Numeric code :      31;
begin
  CAUSE := 8.0000_7777; <* minus *>
  ILIM  := 8.6000_4000; <* clear AR, sub, bockexp = -2048 *>

  cmv:
  status(22:23) := 0;
  work3         := 3;
  if w0 > 0 then
  begin
    get dwd;
    if op1q <> 0 and opq <> (1 shift 11) then
    begin
      SP16 := op1q;
      SP17 := opq;
```

```
while w0 > 0 do
begin

  addr := reg_pre;
  get dwd;
  if op1q <> 0 and opq <> (1 shift 11) then
  begin
    q := op1q;
    op1q := ILIM;
    gpu_init1;
    <* gpu modus1 will return in next microstep when
       called as procedure *>
    start dpus1;
    addr := reg_w;
    start dpu; <* whith c *>
    <* (q,opq) * (SP16,SP17) *>

    get dwd;
    W3:
    if interrupt flag then test_interrupt;
    if dpu_not_ready then goto W3;
    if dpu_status(22:23) <> 0 then dpu_error;

    if op1q <> 0 and opq <> (1 shift 11) then
    begin
      start dpu11; <* 0, CAUSE *>
      addr := reg_w;
      add2;
      <* gpu add2 returns to next microstep when called
         as procedure *>
    end;

    start  dpustr; <* single *>
    gpu inv3;
    <* gpu inv3 returns to next microstep when called
       as procedure *> .

  end <* multiplicand = 0.0 *>;

  reg_w   := reg_w + 4;
  reg_pre := reg_pre + 4;
  w0      := w0 - 1;

end <* w0 > 0 *>;
```

```
gpuinit:
comment "store register in DPU register MODUS"
        ------------------------------------
The word pointed by the effective address is loaded into the DPU
register MODUS. The first 12 bits is handled as modusbits and the
last 12 bits as block_exponent used in block_floatingmode.
  bits       used in              function
  b(0)   gpuinit          0 : none        1 : AR := 0.0

  b(1)   gpuadd        )     add             sub
  b(1)   gpumla        ( 0 : add         1 : sub
  b(1)   gpusub        )     sub             add

  b(2)   gpuadd        )
  b(2)   gpumla        ( 0 : normalize AR  1 : not normalize AR
  b(2)   gpusub        )

  b(2)   gpuinv       )( 0 : normalized rounded result
  b(2)   gpustr *     )( 1 : blockfloated rounded result

  *) w(2) = 1 has no effect in single store, i.e. normalized rounded
     result

  blockfloated means that the mantissa is shifted to the right or to
  the left until the mantissa is normalized or the exponent is equal
  to the block_exponent.

Numeric code :   63;
begin
  get_word;
  gpuinit1:
  op1q := opq(0:11) con (1 shift 7);
  opq  := opq(12:23);
  start dpuinit;
  W7:
  if interruptflag then testinterrupt;
  if dpu not ready then goto W7;
  goto next instruction;
end;
```

```
gpuinv:
comment "invert floatingpoint and store".
        ------------------------------
The floating point pointed by the effective address is inverted
and the result is stored in the doubleword pointed by register w.
Numeric code :  60;
begin
  getdwd;
  if -, mormalized then
  begin
    work5 := reg_w;
    work2 := sign_ext(opq(12:23));
    opq(12:23) := 0;
    normalize and round;
    opq := reg_w;
    oplq := reg_pre;
    if zero then
    begin
      q := 2;
      status(22:23) := 0;
      dpu_error;
      oplq := 0;
      opq  := 4096; <* 0.0 *>
      goto z_str_2;
    end;
  end;

  begin
    gpu_inv_1:
    start dpuinv;
    gpu_inv_2:
    addr := regw;
    gpu_inv_3:
    ex(22:23) := 0;
    W4:
    if interrupt flag then testinetrrupt;
    if dpu not ready then goto W4;
    oplq := dpuresult0
    opq  := dpuresult1
    store dopq;
    if dpu_ex(22:23) <> 0 then
    begin
      ex(22:23) := dpu_ex(22:23);
      if floating point exception active then
      goto floatingpointexception;
    end;
  end;
  goto next instruction;
end;
```

```
gpumla:
------
comment "repetitive mult and add to accumulator".
The two floating point numbers addressed by w and wpre is mul-
tiplied and added to the accumulator with the sign given in
modus(1). then the addresses are increased by 4. The repetition
count in w0 is then decreased and if notzero the procedure is
repeated.
NB : At return one addres is equal to the divisor_addr during
--   cholesky-reduction of a matrix.
Numeric code :  58;

if w0 > 0 then
begin
  gpu_mla_1:
  ex(22:23) := 0;

  while w0 > 0 do
  begin
    addr := regw;
    getdwd;
    if op1q <> 0 and opq <> (1 shift 11) <* 0.0 *> then
    begin
      work4 := op1q;
      work5 := opq;
      addr  := regpre;
      getdwd;
      if op1q <> 0 and opq <> (1 shift 11) <* 0.0 *> then
      begin
        W1:
        if interrupt flag then test_interrupt;
        if dpu not ready then goto W1;
        if dpu_status(22:23) <> 0 then
           dpu_error;
        start dpusl;
        reg_pre := reg_pre + 4;
        reg_w   := reg_w   + 4;
        i/o_addr := 0;
        start dpu; <* with work4, work5 *>
        go_to mla_c1;
      end;
    end;
    regpre := regpre + 4;
    regw   := regw   + 4;
    mla_c1:
    w0 := w0 - 1;
  end w0 > 0 loop;

  wait dpu;
  goto next instruction;
end;
```

```
gpured:
------
comment "cholesky-reduction of a datamatic block of columns in
         blockfloating mode".
The instruction must use w3 as regw and w1 as regx and the address
must point to address c5 in the working area :
          gpured  w3  x1   c5.
w1 need not be zero as it is subtracted from the address.
working area :
 b0  =  base of working area ;  CAT_I1U + nlu_base
 c4  = b0 +  2  : AUTORED
 c3  = b0 +  4  : LR = last_reduced column
 b7  = b0 +  6  : (R_max + 1) Helmert block_red_limit.
 n2  = b0 +  8  : SZU = Saved Zeroes in Unreduced coulmb
 c1  = b0 + 10  : CAT_SZR + n1r_base, base of SAved Zeroes Reduced..
 c0  = b0 + 12  : CAT_I1R + n1r_base, index of first nonzero element
                  in reduced column.
 n11 = b0 + 14  : 4 * (R_MAX + 1) = 2 * word(b7)
 b5  = b0 + 16  : tail_displacement.
 c5  = b0 + 18  : accu_mode -> addsign
 c6  = b0 + 20  : block_exp_r + n1r_base, base of catalog on exponents
                  of reduced columns to be subtracted from the expo-
                  nent of the element under reduction to make it a
                  block_floating number.
 n13 = b0 + 22  : block_exp_u , to be subtracted from the exponent
                  of the element under reduction.
 b8  = b0 + 24  : block_exp_u + n1u_base, base of catalog on exponents
                  of unreduced column to be e.t.c.
 b6  = b0 + 26  : exp_lim, max acceptable loss of binals.
 b4  = b0 + 28  : status + n1u_base, base of catalog of statusarea of
                  unreduced elements.
 n0  = b0 + 30  : U, index of unreduced column.
 b1  = b0 + 32  : cat_szu + n1u_base, base of catalog of saved zeroes
                  of unreduced columns.
 c2  = b0 + 34  : FR, first index of reduced column in datamatic block
 b3  = b0 + 36  : LU, last index of unreduced column in datamatic block
 b2  = b0 + 38  : FU, first index of unreduced column in ...

Numeric code :  62;
```

```
begin
  q       := addr - regx;
  w0      := 0;
  reg_x   := 1;
  reg_pre := 2;
  reg_w   := 3;
  if w1 <> 1 then goto illop;
  if w3 <> 3 then goto illop;

  addr := q;
  get word;
  work4 := 8.1000,0000 and op1q; <*b1_f1_bit *>
  work5 := op1q(12:23, 0:11);
  work3 := (8.2000,0000 and work5 <* sign_bit *>) or work4;
  op1q  := 8.0000,7777 and op1q;
  store opq;
  op1q  := 8.4000,0000 or work3;
  work4 := work4(12:23, 0:11);
  SP16  := work4 or op1q;
  addr  := addr - 12; <* b7 *>
  SP17  := addr + 2;
  get word;
  opq   := opq + opq;
  work5 := addr - 2; <* c3 *>
  addr  := q - 4; <* n11 *>
  store opq;
  work4 := addr + 20; <* c2 *>
  addr  := work4 + 4; <* b2 *>
  get word;
  addr  := work4 - 4; <* n0 *>
```

```
NEXT COL:
  store opq;
  work3 := opq;
  addr := work5 - 2; <* c4 *>
  get word;
  if opq <> 0 then
  begin <* AUTORED *>
    addr := work5; <* c3 *>
    opq := work3; <* U *>
    store opq;
  end;

  addr := work5 - 4; <* b0 *>
  get word;
  addr := opq + work3; <* addr I1U *>
  get word;
  CAUSE := opq; <* I1U *>

  addr := work4 - 2; <* b1 *>
  get word;
  addr := opq + work3; <* addr SZU *>
  get word;
  work2 := opq + opq;

  addr := work5 + 2; <* b7 *>
  get word;
  if work2 <= opq then
  begin <* SZU*2 <= RMAX+1 *>

    addr := SP17; <* N2 *>
    opq  := work2; <* SZU *>
    store opq;

    if (8.0000,7777 and SP16 <> 0 ) then
    begin
      work5 := addr:= work5 + 20; <* b8 *>
      get word;
      addr := opq + work3; <* addr EXP_U *>
      get word;
      addr := work5 - 2; <* n13 *>
      store opq;
    end;

    addr := work4; <* c2 *>
    get word;
    regx := opq; <* FR *>
    work3 := SP17; <* n2 *>
```

```
NEXT REDUCED:
  get word;
  if regx >= opq - 1 then
  begin <* R >= SZU - 1 *>
    work5 := opq; <* SZU *>

    addr := addr + 4; <* c0 *>
    get word;
    addr := opq + regx; <* addr I1R *>
    get word;
    regpre := opq; <* I1R *>

    addr := work3 + 2; <* c1 *>
    get word;
    addr := opq + regx; <* addr SZR *>
    get word;
    opq := opq + opq; <* 2 * SZR *>

    if opq < work5 then
    opq := work5 <* SZR := SZU *>;

    work2 := opq;
    addr  := work3 - 2; <* b7 *>
    get word;
    if work2 < opq then
    begin <* SZR < (RMAX+1) *>

      work2 := work2 + work2; <* 2*SZR *>
      regpre := regpre + work2; <* addr_B *>
      work5 := regx + regx; <* 2 * R *>
      RTC := CAUSE + work5; <* accu_addr_A *>
      work4 := CAUSE + work2; <* addr_A *>

      addr := addr + 8; <* n11 *>
      get word;
      if work5  >= (RMAX+1)*4 then
      begin
        work5 := work0; <* (RMAX+1)*4 *>
        ILIM := 0; <* full_red := false *>
      end
      else
        ILIM := -1; <* full_red := true *>

      work5 := work5 - work2; <* MAX_R - SZR *>
      if work5 < 0 then work5 := 0;
      work5 := work5 shift (-2); <* repetition_count *>

      opq := 8.7777,0000 and SP16; <* MODUS *>
      gpuinit1; <* when called as a procedure gpuinit returns to
                   the next microprogramstep  in stead of next instr *>
      regw := work4; <* addr_A *>
      w0 := work5; <* repetition_count *>
      gpumla1; <* when called as a procedure gpumla returns to the
                  next microprogramstep in stead of next instruction *>
      comment at return is w2 = wpre equal to divisor_addr;
```

```
reg_w :=
w0 := w2 - reg_w; <* if DIA then 0 else <> 0 *>

work2 :=
addr  := work4 + 12; <* c6 *>

if (8.0000,7777 and SP16) <> 0 then
begin
  get word;
  addr := opq + regx; <* addr EXP_R *>
  get word;
  work3 := opq; <* EXP_R *>
  addr := work2 + 2; <* n13 *>
  get word;
  work3 := work3 + opq; <* EXP_R + EXP_U *>
end
else work3 := 0;

addr := work2 - 2; <* c5 *>
get word;
work4 := opq; <* add_sign *>
addr := addr - 2; <* b5 *>
get word;
work5 := opq; <* tail_disp *>
addr := RTC + opq; <* tail_addr_A *>
get dwd;
work2 := signextended(opq(12:23));
if work2 <= -2048 or work2 - work3 <= -2048 then
_  work2 := -2048
else
begin
  work2 := work2 - work3;
  opq := opq(0:11) + work4; <* tail, addsign *>
  start dpu11;
  addr := RTC; <* head_addr_A *>
  get dwd;
  work4 := sign_extend(opq(12:23)) - work3; <* block exp *>
  opq := opq(0:11) con work2(12:23);
  gpuadd entrypoint gpuadd2;
  <* gpuadd returns to next microstep when called as pro-
  cedure in stead of going to next instruction *>
end;
regw := RTC; <* head_addr_A *>
```

```
        if ILIM = 0 then
        begin <* full_red = false *>
          opq := 0;
          gpu_init_1;
          addr := work5; <* tail_disp_A *>
          gpu str double;
          <* gpu str returns to next microstep when
             called as procedure *>
        end
        else
        <* full_red = true *>
        if w0 = 0 then squareroot
        else
        begin
          addr := regpre; <* div_addr *>
          gpu arm;
          <* gpu arm returns to next microstep when
             called as procedure *>
          addr := 0;
          gpu str;
          <* gpustr returns to next microstep when called
             as procedure *>
          end;
        work3 := SP17; <* n2 *>

      end SZR < (RMAX+1);

    end R > SZU - 1;

    regx := regx + 2; <* R := R + 2 *>
    addr := work3 - 4; <* c3 *>
    get word;
    if regx <= opq <* R <= LR *> then goto NEXT REDUCED;

    work5 := addr;
    work4 := work3 + 26; <* c2 *>
  end SZU <= (RMAX+1);

  addr := work4 + 2; <* b3 *>
  get word;
  q := opq;
  addr := work4 - 4; <* n0 *>
  get word;
  opq := opq + 2; <* u := u + 2 *>
  if opq <= q <* U <= LU *> then goto NEXT COL;

  goto next instruction;
end;
```

```
gpusqrt:
comment "wpre,w := sqrt( dwd(addr) )."
        ----------------------------
  The squareroot of the doubleword pointed by the effective
  address is taken and the result is stored in wpre,w.
Numeric code: 1;
begin
  procedure sqrt_exception;
  begin
    sqrt_exc :
    reg_pre := op1q;
    reg_w   := opq
    status  := overflow;
    if floating point exception active then goto
    _             floating point exception;
  end;
  end;

  get_dwd;
  if -, norm then
  begin
    prepare for normalization;
    normalize and round;
    SQRT2:
    opq := w;
    op1q := w_pre;
    if -, norm then goto sqrt_ud;
  end;

  SQRT1:
  if neg then sqrt_exception;
  work5 := opq;
  work4 := op1q;

  if work5(23) = 1 then op1q := op1q // 2;
  op1q   := work0 shift (-2);
  addr   := op1q; <* save radicand *>
  regpre := op1q; <* radicand *>

  <* linear appr. of sqrt(regpre con reg) as long *>
  op1q := 1 949 686 + addr + addr;

  CAUSE := op1q; <* iterand *>
  wd entrypoint at wdbf after get word;
  <* wd returns to next micro when called as procedure *>
  <* regw := regpre // work0 *>
  reg_w := (reg_w + CAUSE) // 2;
  CAUSE :=
  op1q  := w0; <* itterand *>
```

```
  regpre := addr; <* radicand *>
  wd entrypoint at wdbf after get word;
  <* wd returns to next micro when called as procedure *>
  <* regw := regpre // opq *>
  reg_w := reg_w + CAUSE;
  if overflow then reg_w := reg_w // 2;

  oplq   := reg_w;
  regpre := work4;
  regw   := work5; <* real radicand *>
  work2  := (sign_extended(work5(12:23) + 1) shift (-1);
  addr   :=
  opq    := work2(12:23); <* iterand *>
  CAUSE  := oplq;

  fd entrypoint after getdouble2;
  <* fd return to next micro when called as a procedure *>
  <* regpre con regw := regpre con regw / oplq con opq *>

  oplq := CAUSE;
  opq  := addr; <* iterand *>
  fa entrypoint after getdouble2;
  <* fa return to next micro when called as procedure *>
  <* regpre con regw := regpre con regw + oplq con opq *>

  work2 := sign_extended(regw(12:23)) - 1;
  reg_w(12:23) := work2(12:23);

end;



gpustop:
comment "stop insruction".
        ---------------
  the instruction stops the excecution of code and returns a
  normal answer to the sender.
Numeric code:  0;
begin
  work4 := 0;
  work5 := C_TOP - C_LOW + 2;
  send answer;
end;
```

```
gpustr:
comment "store accumulator".
        -----------------
  If addr = 0 then the accumulator is stored in addr regw (single
  precision) possibly in blockfloating mode decided by MODUS(2)
  see gpu init.
  If addr <> 0 then the accumulator is stored in the addresses regw
  and (regw+addr) in double floating point mode.
  The accumulator is unchanged.
Numeric code:  61;
begin
  if addr = 0 then
  begin
    gpu_str_single:
    oplq := opq := 0;
    goto gpu_inv_2;
  end;
  gpu_str_double:
  oplq := 0;
  opq  := 1 shift 16; <* double round constant *>
  start dpustr;
  addr  := addr + regw;
  work5 := regw;
  opq   := 8.77770000; <* mask *>
  work4 := 35;
  q     := 3;
  W6:
  if interrupt flag then test interrupt;
  if dpu not ready then goto W6;
  work3 := dpuresult1;
  q := dpustatus and q; <* exeptionbits *>
  if q <> 0 then
  begin
    dpu_error;
    goto z_str;
  end;
  oplq := dpuresult0;
  if oplq < 0 then goto z_str;
  start_dpu
  opq := opq and work3; <* frac(24:35) *>
  work3 := signextended(work3(12:23)) - 35; <* exp *>
  if <* exp = *> work3 >= -2048 then
  begin
    opq := opq + work3(12:23); <* frac(24:25) con tail_exp *>
    store dopq;
    addr := work5; <* addr_head *>
    oplq := dpuresult0;
    opq  := dpuresult1;
    store dopq;
  end
  else
```

```
    <* else if exp < -2048 then *>
    begin
      z_str:
      oplq := 0;
      z_str_1:
      opq  := 1 shift 11; <* 0.0 *>
      store dopq;
      addr := work5; <* addr head *>
      z_str_2:
      store dopq;
    end;
    goto next instruction;
end;

gpusub:
comment "subtract double floating point from accumulator".
        ------------------------------------------------
    If addr = 0 then (the double floating point dwd(regw) con 0.0)
    else (the double floating point dwd(regw) con dwd(regw+addr))
    is subtracted from the DPU accumulator AR by the sign given
    in MODUS (see gpu_init). If MODUS(0) = 0 then positive else
    negative sign.
Numeric code:  31;
begin
    work3 := 8.0000,7777;
    goto gpu_add2:
end;

hl:  see (1).
hs:   do
jd:  <* illegal *> goto illop;
je:  <* illegal *> goto illop;

jl:
comment "jump with link in register".
        -------------------------
    Transfers control to the instruction pointed by the effective
    address. If the W-field is non-zero the link, i.e. the logical
    addess of the next instruction is assign to the specified re-
    gister. References outside codelow and codetop leads to program
    exception!
Numeric code:   13;
begin
    getinstruction;
    if W <> 0 then regw := ic;
    ic := addr;
    if code_low <= ic + base < code_top then goto next instruction
    else goto code_violation;
end;
```

```
ks:
comment "key store"
         ---------
        used in codedebugging.
        see GPU users manual : gpu_exec.
Numeric code:   51;
comment NO OP;
goto next instruction;

la:  see (1).
ld:   do
lo:   do
ls:   do
lx:   do
nd:   do
ns:   do
re:  <* illegal *> goto illop;
ri:  <* illegal *> goto illop;
rl:  see (1).
rs:   do
rx:   do
se:   do
sh:   do
sl:   do
sn:   do
so:   do
sp:   do
ss:   do
sx:   do
sz:   do
wa:   do
wd:   do
wm:   do
ws:   do
xl:   do
xs:   do
zl:   do
```

Interrupt service, message answer routine and exception routines.

POWER UP will reset the internal registers and clear all interrupts.
whereupon the gpu waits on interrupts in the idle loop.

In the IDLE LOOP the GPU will respond on interrupts :

power up	level 1
input from tecnical panel	level 2
single instruction	level 3
external interrupts	level 8 - 15

The gpu has devicenumber (interruptlevel - 7) * 4.

The address of controller table is
```
ct = core(8) + devicenumber * 8;
```

The format of controller table is :
```
ct + 0 :  channel program address : ch
ct + 2 :  standard status area address : st
ct + 4 :  interrupt destination (host cpu devicenumber)
ct + 6 :  interrupt level (host)
```

The format of channel program is :
```
ch + 0 :  operation
ch + 2 :  message area base
ch + 4 :  not used

ch + 6 :  operation = stop
ch + 8 :  not used
ch + 10:  timer in 0.1 msec
```

The format of message is :
```
mess +  0 :  next buffer
mess +  2 :  previous buffer
mess +  4 :  receiver or answer type
mess +  6 :  sender
mess +  8 :  (address_code shift 12) + operation shift 8
mess + 10 :  first code (see below)
mess + 12 :  last code
```

operation = 0 shift 8 : start gpu
operation = 1 shift 8 : stop gpu

first code is also called code_low.
last code is also called code_top.

code excecution starts in word (first code + 4).
++

The following parts of the processdescription is used :

```
pr_descr +  88 :  status
pr_descr +  96 :  cpa
pr_descr +  98 :  base
pr_descr + 100 :  lower write limit
pr_descr + 102 :  upper write limit
```

The status will at return contain :

```
st + 0 :  ch + (if status ok then 12 else 0)
st + 2 :  bytes count
st + 4 :  chars count
st + 6 :  status;
```

The status bits are :

```
1 < 14  :  code address fault <* pic < code_low or code_high < pic
                or code_low < llim or ulim < code_high *>
1 < 15  :  integer underflow *)
1 < 16  :  integer overflow *)
1 < 17  :  floating point underflow **)
1 < 18  :  floating point overflow *)
1 < 19  :  read/write address fault
1 < 20  :  bus error
1 < 21  :  time out
1 < 22  :  illegal instruction
```

*) controlled by the statusbits of the senders process.
e.g. integer underflow is only active when the senders process has set the status : integer exception active.

**) in case of floating point underflow the result is set to floating point zero and no exception is set up.

++++++++++++

```
external interrupt:
comment interruptlevel 8 - 15;
begin
  opq   := ((opq - 8) shift 2 + 4) shift 3;
  work1 := word(8) + opq;
  if buserror then goto system fault;
  CT_ADDR := work1;
  opq := word(work1);
  if buserror then goto system fault;
  q := word(q+2);
  work4 := 8.7777 7776;
  if buserror then goto system fault;
  work5 := word(opq) and 8.0000_7400;
  if buserror then goto system fault;
  if work5 = 0 then goto stop;

  addr := q + 10;
  IC   := word(addr) and work4;
  if buserror then goto system fault;

  pic := word(addr+2) and work4; <* last_code *>
  if buserror then goto system fault;
  addr := word(addr-4); <* pr_descr *>
  if buserror then goto system fault;

  addr := addr + 96;
  cpa  := word(addr);
  if buserror then goto system fault;
  addr := addr + 2;
  base := word(addr);
  if buserror then goto system fault;
  addr := addr + 2;
  llim := word(addr);
  if buserror then goto system fault;
  addr := addr + 2;
  ulim := word(addr);
  if buserror then goto system fault;
  addr := addr - 14;
  cpu_status :=
  status := word(addr);
  if buserror then goto system fault;
  cpustatus := status := status and 8.0300_0000;
  <* take integer and floating point exception active bits *>
  pic := pic + base;
  code_top := pic;
  code_low := IC + base;
  if code_top > ulim or code_low < llim then goto code violation;
  ic  := ic + 4;
  pic := pic + 4;
  get instruction;
  goto next instruction;
end;
```

```
stop:
  status := 8.1000_0000;
send1:
  work5 := pic - code_low;


send answer:
comment set the result and cause in message. set the result and
  cause in status area and goto idle loop;
begin
  comment set status and goto idle;
  q := CT_ADDR;
  opq := word(q);
  if bus_error then system fault;
  q := q + 2;
  addr := word(q) + 2; <* status addr + 2 *>
  if buserror then goto system fault;
  word(addr) := work5; <* bytes count *>
  if buserror then goto system fault;
  work3 := work5;
  word(addr+4) := work4; <* status *>
  if buserror then goto system fault;
  work3 := work3 // 2;
  word(addr+2) := work3 + work5; <* chars count *>
  if buserror then system fault;
  word(addr-2) := opq + 12;
  if buserror then system fault;
  data_out := word(q+4);
  if buserror then goto system fault;
  data_in := word(q+2); <* CT + 4 : CPU_ADDR *>
  if buserror then goto system fault;
  word(data_in) := data_out; <* interrupt cpu *>
  if buserror then goto system fault;
  clear code_limits;
  goto idle loop;
end;
```

```
floating point exception:
comment set result and cause;
begin
  work4 := 8.0040_0000;
  fl_exc_1:
  if overflow then work4 := work4 * 2;
  goto sendl;
end;

integer exception:
begin
  work4 := 8.0010_0000;
  goto fl_exc_1;
end;

operand error:
comment buserror. set status;
begin
  status := status and 8.77770010;
  if busparity  then status := status + 8.00000004;
  if busnack    then status := status + 8.00000001;
  if bustimeout then status := status + 8.00000002;
  work4 := 8.0400_0000;
  goto sendl;
end;

program exception:
comment write limit violation;
begin
  work4 := 8.0200_0000;
  goto sendl;
end;

code violation:
begin
  work4 := 8.0004_0000;
  goto sendl;
end;


illop:
comment illegal operation. word(ic) is given in cause;
begin
  work4 := 8.2000_0000;
  if buserror then goto system fault;
  goto sendl;
end;


system fault:
goto idle;
```

Reference Manual

for

DPU

K. Engsager

Geodætisk Institut

1983

Basic Hardware.

The FPU 801 unit from RC COMPUTER A/S is used as basic hardware. Se 1, 2, 3.

Modifications.

Some few modifications has been made to facilitate the microprogram of doublefloatingpoint precision.

I) Bit 35 of the microinstruction is used as an addressbit in connection with the usual bits 27:31 for addressing the 16 ALU-registers. The addressing is done in 4 groups of each 4 registers. Bit 35 con bit 29 give the groupnumber (0:3). Bit 30:31 give the index 0:3 of RAMA register in the named group. Bit 27:28 give the index 0:3 of RAMB register in the named group. Se table 1.

II) The functionfield has no decrementcontrol (originally bit 35).

III) The functionfield has a changed workingarea as follows :
bit 35 work on alu0 (bit 0:4)
bit 46 work on alu 3-9 (bit 12:39)
Se table 2 and 3.

IV) Bit 40 of the sumregister in the 8 * 40 multiplier is not connected to the carrybit of the RAM-shifter when moving the bits 0:39 to the sourcebus. The hardware is used to take the carry in long shifts.

Table 1

REG no	Assembler name	bitno 35	bitno 29	Reference name	Register used bits	group
0	A0	0	0	IFRH)_(IFR : intermedi-	0 : 39	A
1	A1	0	0	IFRT) (ate fraction	0 : 39	A
2	A2	0	0	AFRH)_(AFR : accumulator	0 : 39	A
3	A3	0	0	AFRT) (fraction	0 : 39	A
4	B0	0	1	IEX intermediate exponent	0 : 12	B
5	B1	0	1	AEX accumulator exponent	0 : 12	B
6	B2	0	1	MRE multiplicatorexponent	0 : 12	B
7	B3	0	1	DIA (= IEX - AEX mostly)	0 : 12	B
8	C0	1	0	MRF multiplicator fraction	0 : 35	C
9	C1	1	0	H1 workregister	3 : 10	C
10	C2	1	0	H2 workregister	3 : 10	C
11	C3	1	0	H3 workregister	3 : 10	C
12	D0	1	1	MODUS, B16	0:7, 8:15	D
13	D1	1	1	K72, exp.dif.control	0 : 12	D
14	D2	1	1	block exponent	0 : 12	D
15	D3	1	1	WRKMODUS	0 : 6	D

DPU G.I.

MICROPROGRAM	46 I5 3-9	45 I5 0	44 I0 3-8	43 I2 3-8	42 I0 0-2,9	41 I1 0-9	40 I2 0-2,9	39 CARRY 2	38 CARRY 9	37 I3 0-9	36 I4 0-9	(36:46) FUNCTION	DESCRIPTION DPU G.I.
LOADF	0	0	1	1	1	1	1	0	0	0	0	0174	BUS (0:39)
LOADE	0	0	1	1	1	1	1	0	0	0	0	0177	BUS (Y:11), BUS(0:3,12:39) ∧ 0
SUBQA	0	0	0	0	0	0	0	1	0	0	1	7400	Q - RAM
PAS.Q	1	0	0	0	0	1	0	0	0	1	1	3040	0 ∨ Q (= Q)
NEG.Q	1	0	0	0	0	1	0	1	0	1	0	2440	0 - Q (= -Q)
ADDQA	0	0	0	0	0	0	0	0	0	0	0	0000	Q + RAM
ADD	0	0	0	0	0	0	1	0	0	0	0	0024	RAM + RAM
PAS.A	1	0	0	0	0	1	0	0	0	1	1	3110	A ∨ 0 (= A)
ADD.E	0	0	0	0	0	1	0	0	0	0	0	0130	RAM(0:11) + BUS (0:11)
PAS	1	0	0	0	0	1	1	0	0	0	0	3174	BUS (0:39)
LOADI	0	0	0	0	0	1	1	0	0	0	0	0175	BUS (0:11), BUS (12-39) ∧ 0
NEG.A	1	0	0	0	0	1	0	1	0	1	0	2570	-A RAM
SUB	1	0	0	0	0	0	1	1	0	1	0	2424	RAM_B - RAM_B
CLEAR	1	0	0	0	0	0	1	1	0	1	0	2424	0 = RAM - RAM
SUB.E	0	1	0	0	0	1	0	1	0	0	1	1530	RAM - (BUS(0:11), 0(12:35), BUS(36:39))
ISUB	0	1	0	0	0	0	1	1	0	0	1	1424	RAM_B - RAM_B
SUBAQ	1	0	0	0	0	0	0	1	0	0	1	2400	RAM - Q
INC.A	0	0	0	0	0	1	0	0	1	0	1	0311	RAM + 1B11
ISUBE	1	0	0	0	0	1	0	1	0	1	0	2530	(BUS(0:11), 0(12:35), BUS(36:39)) - RAM
INC.R	0	0	0	0	0	1	0	0	1	1	0	0510	RAM + 1B39
OR	1	1	0	0	0	0	1	0	0	0	1	3024	RAM_A ∨ RAM_B
ADD1	0	1	0	0	0	0	1	0	0	0	1	0424	RAM + RAM +1
ISUB1	0	0	0	0	0	0	1	1	0	0	1	1024	RAM_B - RAM_A -1

table 2

MICROPROGRAM FUNCTION FIELD

MIC	CARRY	SOURCE						FUNCTION									DESCRIPTION			
FNC	9	2	ALU 0-2,9			ALU 3-8			ALU 0			ALU 1-2			ALU 3-9			ALU 0	ALU 1-2	ALU 3-8	ALU 9
BIT:	38	39	40	41	42	43	41	44	45	36	37	Z	36	37	46	36	37				
LOADF	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	D + Z	D + Z	D + Z	D + Z
LOADE	0	0	1	1	1	1	1	1	1	0	0	0	0	0	1	0	0	D AND Z	D + Z	D AND Z	D AND Z
SUBQA	1	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	Q - A	Q - A	Q - A	Q - A
PAS.Q	0	0	0	1	0	0	1	0	0	1	1	0	1	1	0	1	1	Z OR Q	Z OR Q	Z OR Q	Z OR Q
ADDQA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	A + Q	A + Q	A + Q	A + Q
ADD	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	A + B	A + B	A + B	A + B
PAS.A	0	0	1	0	0	1	0	0	0	1	1	0	1	1	0	1	1	Z OR A	Z OR A	Z OR A	Z OR A
ADD.E	0	0	1	0	1	1	0	1	0	0	0	0	0	0	0	0	0	D + A	D + A	Z + A	D + A
PAS	0	0	1	1	1	1	1	1	0	1	1	0	1	1	0	1	1	D OR Z	D OR Z	D OR Z	D OR Z
LOADI	0	0	1	1	1	1	1	1	0	0	0	0	0	0	1	0	0	D + Z	D + Z	D AND Z	D AND Z
NEG.A	1	0	1	0	0	1	0	0	0	1	0	0	1	0	0	1	0	Z - A	Z - A	Z - A	Z - A
SUB	1	0	0	0	1	0	0	1	0	1	0	0	1	0	0	1	0	A - B	A - B	A - B	A - B
CLEAR	1	0	0	0	1	0	0	1	0	1	0	0	1	0	0	1	0	A - A	A - A	A - A	A - A
SUB.F	1	0	1	0	1	1	0	1	0	0	1	0	0	1	0	0	1	A - D	A - D	A - D	A - D
ISUB	1	0	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	B - A	B - A	B - A	B - A
SUBQA	1	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0	A - Q	A - Q	A - Q	A - Q
INC.A	0	1	1	0	0	1	0	0	0	0	0	0	0	0	1	0	0	A + Z	A +Z+ 1	A AND Z	A AND Z
ISUBF	1	0	1	0	1	1	0	1	0	1	0	0	1	0	0	1	0	D - A	D - A	D - A	D - A
INC.R	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	A + Z	A + Z	A + Z	A +Z+ 1
OR	0	0	0	0	1	0	0	1	0	1	1	0	1	1	0	1	1	A OR B	A OR B	A OR B	A OR B
NEG.Q	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	Z - Q	Z - Q	Z - Q	Z - Q
ADD1	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	A + B	A + B	A + B	A +B+ 1
ISUB1	0	0	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	B - A	B - A	B - A	B -A- 1

Table 3.

Microprogram function field

Desciption of the microprogram and format of the registers.

The single precision floatingpoint numbers is the usual RC 8000 floatingpoint numbers.
The double precision floatingpoint number consists of two single floatingpoint numbers, where the least significant number is positive and unnormalized with an exponent equal the most significand exponent minus thirtyfive.

Double floating = FHEAD + FTAIL,
where FHEAD and FTAIL is the usual floating point numbers,
where FTAILexp = FHEADexp - 35 and FTAILfrac >= 0.

In the DPU the double precision floatingpoint numbers are stored in IR and AR as given by example IR :

IFR(0:70) = FHEAD(0:35) con FTAIL(1:35)
IEX(0:12) = FHEADexp(0) con FHEADexp(0:11)
IFRH(0:39) = IFR(0:39)
IFRT(0:30) = IFR(40:70)

NOTE : the two signs of the exponent.
---- the leftpositioning and compact notation of fraction there is 9 guardingciffersbits IFRT(31:39), which allows to add the unnormalized product to the accumulator because normalising of the product claims 0 - 2 leftshifts.

The MODUS register contain information used during the execution of the microprogram :

MODUS bit	used in	value = 0	value = 1
0	init	AR unchanged	AR := 0
1	long load mult add	add	sub
2	long load shortload str inv	normalizing mode	blockfloating mode *

* blockfloating store only in singlestore.

Blockfloating mode.
------------------ When MODUS(2) = 1 then id the dpu operating in blockfloating mode, which is a kind of fixedpoint arithmetic. All numbers have an exponent equal to the blockexponent (or greater) and the fraction may not be normalized.

Examples: MODUS(2) = 1, blockexponent = 1

-1 is FR = 6000.0000, exp = 1, (octal)
-2 is FR = 4000.0000, exp = 1
-4 if FR = 4000.0000, exp = 2
-0.5 is FR = 7000.0000, exp = 1
+1 is FR = 2000.0000, exp = 1
+2 is FR = 2000.0000, exp = 2
+0.5 is FR = 1000.0000, exp = 1
+0.25 is FR = 0400.0000, exp = 1

In blockfloating mode there will be no normalizationshifts after an addition and all the storing operations will store a number where the exponent is equal or greater than the blockexponent and the fraction may be unnormalized.

DPU instructions

In the wait on next start operation the workmodus is loadet :
WRKMODUS(0:10) = MODUS(1:11)

INIT :
---- The modus and zeroexponent register is loaded. The MODUS register too contail the constant 1B16.
If the clearbit is set then AFR is set zero and AEX is loaded with the zeroexponent.

SHORT LOAD :
---------- The registers Y, MRF and MRE are loaded.
goto multiply.

MULTIPLY AR :
---------- AR is multiplied by the constant input and the result is stored in AR.

LONG LOAD :
--------- If the exponent of the tail is notzero then the sign of wrkMODUS is changed i.e. WRKMODUS(0) := -, WRKMODUS(0), which is the original MODUS-bit 1 the function is : add becommes sub, - and sub becommes add. IFRT is shifted fourtimes into IFRH. Then the headfraction is added to IFRH and IEX is loaded.

After this a jump is made to the alignment routine.

MULTIPLY and ADD/SUB :
-------------------- The input number is multiplied by MR and a jump is made to the alignment routine. See fig. 1.

The multiplication is described below :

```
    H1          :=
    X(0:35)     := IN(0:35);
    IEX(0:12)   := IN(36) con IN(36:47);
    MULT        := y * X(32:39);  IEX := IEX + MRE;
    MULT        := Y * X(24:31);  SUM := MULT;       MRE := IEX;
    if IEX-over/underflow then goto mla_exception;
    RTN:
    SUM         := MULT + SUM;    MRE := MRE - AEX;
    MULT        := Y * X(16:23);  H3  := 0 ext 3 con SUM(40:47);
    SUM         := MULT + SUM;
    MULT        := Y * X(8:15);   H2  := 0 ext 3 con SUM(40:47);
    SUM         := MULT + SUM;    ALU := H1 + H1; <*test -1 frac*>
    H1          := 0 ext 3 con SUM(40:47);
    if zero then goto neg_mr;
    MULT        := Y * X(0:7);    X   := 1B16;
    SUM         := MULT + SUM;    Y   := H3 * 2;
    IFRT        := SUM(37:47); con 0;
    IFRH        := SUM(1:39) con 0;
    MULT        := Y * X(8:15);   Y   := H2 * 2;
    MULT        := Y * X(8:15);   SUM := MULT;       Y := H1 * 2;
    MULT        := Y * X(8:15);   SUM := MULT + SUM;Y := IFRT * 2;
    MULT        := Y * X(8:15);   SUM := MULT + SUM;DIA := MRE*2;
    SUM         := MULT + SUM;
    IFRT        := SUM(3:39); comment SUM(34:39) == 0;
    IFRH        := IFRH(0:38) con SUM(2);
    goto alignment;

mla_exception :
    SUM         := SUM + MULT;    MRE := MRE - 2;
    MULT        := Y * X(32:39);  ALU := MRE;
    if neg then goto next; <* i.e. underflow *>
    MULT        := Y * X(24:31);  SUM := MULT;       MRE:= IEX;
    goto (if -, norm then RTN else exception);


neg_mr :
    <* change add_sign of work_modus and goto alignment *>
    IFRH        := MRF;
    WRKMODUS(0) := -, WRKMODUS(0);
    IFRT        := 0;
    DIA         := (IEX - AEX) * 2;
    goto alignment;
```

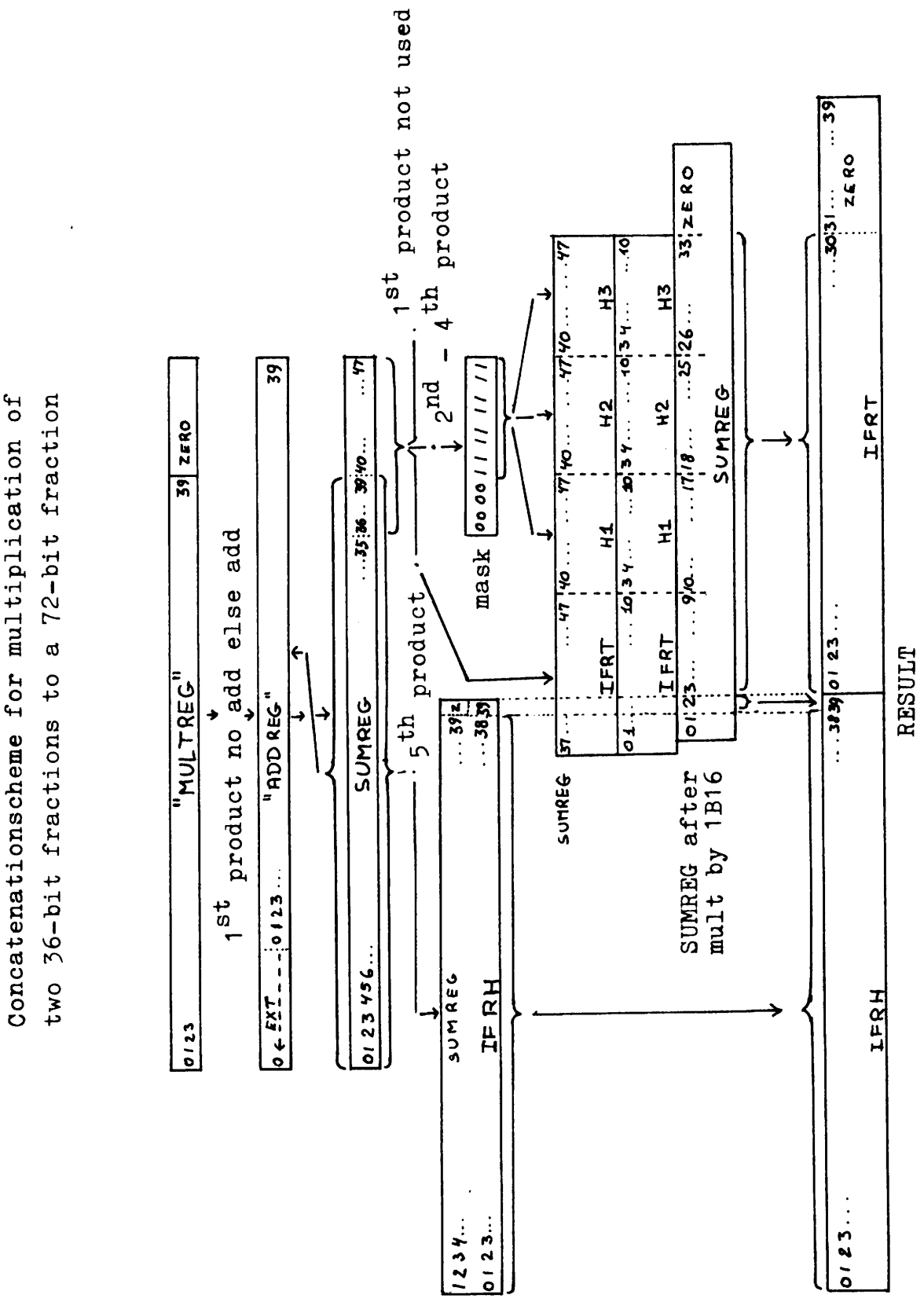

Concatenationscheme for multiplication of
two 36-bit fractions to a 72-bit fraction
"MULTREG"
ZERO
1st product no add else add
"ADD REG"
EXT
SUMREG
1st product not used
2nd - 4th product
5th product
mask
SUMREG
IFRH
IFRT
H1
H2
H3
ZERO
SUMREG after mult by 1B16
RESULT

fig. 1.

INVERSION :
--------- The input single precision floatingpint number is inverted. The single precision result is output and store in MR; the GPU has tested for zero and minus one.
AR is unchanged!

Following the description page 16 in (3) the function is :

```
Y := MRF := IN(0:35);
IEX := IN(36) con IN(36:47);      X := R(Y(0:11));
if Y < 0 then
begin
  Y := - MRF;
  X := R(Y(0:11));          IFRT := 0;
  if ALU(0) = 1 then
  begin
    IEX := IEX // 2;
    IEX := -IEX;
    goto test_exp;
  end;
end;

IFRT := 1B0;
MULT := Y * X(8:15);  comment  -0.5 > R = X >= -1 + 2**(-11);
MULT := Y * X(0.7);         SUM := MULT;
SUM  := MULT + SUM;
IFRH := SUM(1:40);  comment  -0.5 <= A1 < -0.5 + 2**(-11);
Y := IFRT - IFRH;           IFRH := -2 * IFRH;
MULT := Y * X(8:15);        IEX := IEX // 2;
MULT := Y * X(0:7);         SUM := MULT;          IEX := -IEX;
SUM  := MULT + SUM;         X   := IFRH;
IFRH := SUM(1:40);   comment  0.5 < 2*B2 <= 1+2**(-11);
MULT := Y * X(32:39);
MULT := Y * X(24:31);       SUM := MULT;
MULT := Y * X(16:23);       SUM := MULT + SUM;
MULT := Y * X(8:15);        SUM := MULT + SUM;
MULT := Y * X(0:7);         SUM := MULT + SUM;
SUM  := MULT + SUM;         X := 2 * IFRH;
IFRH := SUM(1:40);  comment 0.5-2**(-20) < A2 <= 0.5;
Y    := IFRT - IFRH;
MULT := Y * X(32:39);
MULT := Y * X(24:31);       SUM := MULT;          IFRT := 0;
MULT := Y * X(16:23);       SUM := MULT + SUM;    DIA  := 0;
MULT := Y * X(8:15);        SUM := MULT + SUM;    DIA  := 1B11;
MULT := Y * X(0:7);         SUM := MULT + SUM;    IEX  := IEX + DIA;
SUM  := MULT + SUM;         DIA := DIA // 2;
IFRH := SUM(1:40);  comment .125 < B3 < .25+2**(-12)+delta;
if MRF < 0 then IFRH := -IFRH;  comment           delta is smal;
test_exp:
normalize IFRH and adjust IEX accordingly;
round;
if expo_under/overflow then goto single_ou_flow;
MRF := Y_out := IFRH;
expoout := 2 * MRE;
goto next;
```

STORE the accumulator :
---------- The accumulator AR is unchanged ! The content of AR is moved to IR. IR is rounded and normalized.
If the input number is notzero a double floatingpoiresult is output the number input is the roundkonstant 1B31.
IF the input number is zero a single floatingpoint result is output and stored in MR.

Alignment subroutine :
---------- The routine is working on AR and IR.
The least significand number is shifted right until IEX = AEX. If the exponentdifference is greater than 72 there will be no shifts, but the least significand number is set to zero.
Dependend on the sign of WRKMODUS = MODUS(1) (see note under LONG LOAD) the IR is added to or subtracted from AR.
AR is rounded and a jump is made to wait next opeartion.

Excecution times.

SHORT LOAD : 0.64 us

INIT : 0.64 us by clear add 0.64 us

LONG LOAD : min 3.04 us (when addend << AR)
max 55.84 us
mean 11 us
or < 5.44 - 6.56 > + 0.32 * normshifts
or < 6.24 - 7.68 >
+ 0.16 * alignshifts + 0.32 * normshifts,
where 0 < alignshifts < 73 and 0 < normshifts < 81

MULTIPLY : 1.44 us + time(LONG LOAD)
min 4.48 us
max 57.28 us
mean 12.50 us

INVERSION : min 7.52 us
max 8.96 us
by negative divisor add (min 0.16 us max 0.96 us)

STORE DOUBLE : min 7.52 us
max 4.00 us
when unnormalized add 0.48 * prenormshifts us

STORE SINGLE : min 2.24 us
max 2.72 us
when unnormalized add 0.48 us * prenormshifts

Syncronizations.

LONG LOAD : The input register is free 0.80 us after the tailload.
The microprogram wil start waiting the headpart after
1.60 us (+ 0.16 by signshift on MODUS).

STORE DOUBLE : The output register is loaded by headpart when
0.48 us elapsed since the tailpart was loaded.

References :

(1) FPU 801 General Information, RCSL.no. 30 M - 252
(2) FPU 801 Tecnical Manual, RCSL.no. 30 M - 253
(3) FPU 801 Microprogram Manual, RCSL.no. 30 M - 254
All from RC Computer, Denmark

Appendix 1.

Listing of GPU microprogram :

```
0001 .MAIN DOMUS MACRO ASSEMBLER REV C2.00
01                ;*************************************************************
02                ;
03                ;MACRO FOR GENERATION OF ARITHMETIC-LOGIC MICROINSTRUCTIONS
04                ;
05                ;*************************************************************
06
07                      .XPNG
08        000001        .NOMA 1
09
10
11                ;
12                ; MACRO FILL   -
13                ;
14                ; THE MACRO IS STORING ONES UNTIL 'LOCATION'
15                ;
16                ; MACRO CALL:
17                ;
18                ; FILL 'LOCATION'
19                ;
20
21                .MACRO FILL
22                  .IFG  ./3-.1
23                    SEGMENT TOO LONG
24                  .ENDC
25                  .DO   .1*3-.
26                    -1
27                  .ENDC
28                %
29
30
```

```
10002 ;MAIN
01
02                ;
03                ;                                780706/FK.
04                ;
05                ;       THE MACRO XVFD GENERATE A NEW MACRO NAMED AS
06                ;THE FIRST ARGUMENT IN THE CALL TO XVFD. SUBSEQUENT USE
07                ;OF THE NAME GIVEN IN THE XVFD CALL GENERATES 3 16-BIT
08                ;STORAGE WORDS HAVING PRIMARY VALUES TO WHICH FIELDS
09                ;ARE ASSEMBLED AS DESCRIBED IN THE CALL TO XVFD.
10                ;THE CALL IS OF THE FORM:
11                ;
12                ;   XVFD NAME PRIMARY_VALUE_WORD0 PRIMARY_VALUE_WORD1
13                ;        PRIMARY_VALUE_WORD2 FIELD(1)_RIGHT_BIT FIELD(1)_LENGTH
14                ;        .....FIELD(I)_RIGHT_BIT FIELD(I)_LENGTH
15                ;
16                ;       THE 5TH,7TH,.. ARGUMENT SPECIFY THE RIGHTMOST BIT
17                ;POSITION OF THE 1ST,2ND,.. FIELDS. THE 6TH,8TH,.. ARGUMENT
18                ;SPECIFY THE FIELD LENGTHS FOR THE 1ST,2ND,.. FIELDS.
19                ;TO ASSEMBLE THE FIELDS IN THE PROPER BIT POSITIONS
20                ;A CALL IS MADE OF THE FORM:
21                ;
22                ;       NAME FIELD(1) FIELD(2) ... FIELD(I)
23                ;
24                ;       IF FIELD(I) IS OMITED IS IT POSSIBLE TO
```

```
10CC3 .MAIN
01                ;SPECIFY A DEFAULT AKTION IN XVFD.
02                ;       NOTE!
03                ;THE CALL XVFD NAME GENERAE A AUXILIARY VARIABLE:
04                ;NAME. WHICH CONTAIN THE NUMBER OF FIELDS. THIS
05                ;DEMAND THAT NAME CONSIST OF MAX. 4 CHARACTERS.
06                ;
07        000000  I=0
08        000001  J=1
09        000000  K=0
10        000000  MASK=0
11        000000  DATA=0
12        000000  VALU1=0
13        000000  VALU2=0
14        000000  VALU3=0
15                        .MACRO  XVFD
16                        I=5
17                                .MACRO %1
18                        .DO .PASS
19                                VALU1=%2
20                                VALU2=%3
21                                VALU3=%4
22                                J=1
23                        .ENDC
24                _%
25                        .DO .ARGCT/2-2
26                        I=I+1
27                                .MACRO %1
28                                .IFN '_%J'<>''
29                        .DO       .PASS
30                                MASK=(-1)B(15.-%I&15.)
31                                MASK=(-MASK-1)
32                        .ENDC
33                _%
34                        I=I-1
35                                .MACRO %1
36                        .DO       .PASS
37                                DATA=_%J
38                                MASK=(MASK)B(%I&15.)
39                                DATA=(DATA)B(%I&15.)
40                        .ENDC
41                _%
```

```
100C4 .MAIN
01                              .IFE %I/16.     ;IF FIELD IN 1.WORD THEN
02                              .MACRO %1
03                      .DO .PASS
04                              VALU1=(VALU1&(-MASK-1))+DATA
05              .ENDC
06              _%
07                              .ENDC           ;END
08
09                              .IFE %I/16.-1   ;IF FIELD IN 2.WORD THEN
10                              .MACRO %1
11                      .DO .PASS
12                              VALU2=(VALU2&(-MASK-1))+DATA
13              .ENDC
14              _%
15                              .ENDC           ;END
16                              .IFE %I/16.-2   ;IF FIELD IN 3.WORD THEN
17                              .MACRO %1
18                      .DO .PASS
19                              VALU3=(VALU3&(-MASK-1))+DATA
20              .ENDC
21              _%
22                              .ENDC           ;END
23
24                      K=I+1
25                      MASK=1-%K
26                      .IFL (%I&15.+MASK)      ;IF FIELD>THE REST BIT
27                                              ;   IN THE WORD THEN
28                      .MACRO %1
29                      .DO     .PASS
30                              DATA=_%J
31                              MASK=(-1)B(16.-(%K-%I&15.))
32                              MASK=-MASK-1
33                              .DC 1+%I&15.
34                              DATA=DATA/2
35                              .ENDC
36              .ENDC
37              _%
```

```
100C5 .MAIN
01                              .IFE %I/16.-1           ;IF FIELD IN 1. WORD THEN
02                              .MACRO %1
03                      .DO .PASS
04                      VALU1=(VALU1&(-MASK-1))+DATA
05              .ENDC
06              _%
07                              .ENDC                   ;END
08                              .IFE %I/16.-2           ;IF FIELD IN 2.WORD THEN
09                              .MACRO %1
10                      .DO .PASS
11                        VALU2=(VALU2&(-MASK-1))+DATA
12              .ENDC
13              _%
14                              .ENDC                   ;END
15                              .ENDC                   ;END EXTENDED FIELD.
16                              .MACRO %1
17                              .ENDC                   ;END NOT DUMMY PARM.
18              _%
19                      I=I+2
20                                      .MACRO %1
21
22                                      J=J+1
23              _%
```

```
!CCCC   AIN
01
02                      .ENDC           ;END .DO ARGCT/2-2.
03                                      .MACRO %1
04                  ++                  **.NOMAC 0
05                  ++                  VALU1
06                  ++                  VALU2
07                                      VALU3
08                  _%                  **.NOMAC 1
09                  %
10
11
12
13
```

```
!CCC7 .  -IN
C1
C2
C3
C4
C5
C6
C7
C8
C9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
```

```
;*************************************************
;
;MACRO FOR GENERATION OF JUMP MICROINSTRUCTIONS
;
;*************************************************


                                        780706/FK
;
;       USE:
;
;AS XVFD, EXCEPT THE LAST BUT ONE PARAMETER IS DIVIDED BY 3.
;

        .MACRO  AVFD
        I=5
                .MACRO %1
        .DO     .PASS
                VALU1=%2
                VALU2=%3
                VALU3=%4
                J=1
.ENDC
_%
        %1.=.ARGCT/2-2          ;%1.:=THE NUMBER OF FIELDS.

        .DO .ARGCT/2-2
        I=I+1
                .MACRO %1
                .IFN '_%J'<>''
        .DO     .PASS
                MASK=(-1)B(15.-%I&15.)
                MASK=(-MASK-1)

.ENDC
_%
```

```
100C8 .MAIN
01                      I=I-1
02                              .MACRO %1
03                      .DO     .PASS
04                              DATA=_%J
05                              .IFN J==%1.-1           ;LAST BUT ONE PARM.
06                              DATA=_%J/3              ;ADDRESS PARM.
07                              .ENDC
08                              MASK=(MASK)B(%I&15.)
09                              DATA=(DATA)B(%I&15.)
10              .ENDC
11              _%
12
13                                      .IFE %I/16.     ;IF FIELD IN 1.WORD THEN
14                                      .MACRO %1
15                      .DO .PASS
16                              VALU1=(VALU1&(-MASK-1))+DATA
17              .ENDC
18              _%
19                                      .ENDC           ;END
20
21                                      .IFE %I/16.-1   ;IF FIELD IN 2.WORD THEN
22                                      .MACRO %1
23                      .DO .PASS
24                              VALU2=(VALU2&(-MASK-1))+DATA
25              .ENDC
26              _%
27                                      .ENDC           ;END
28                                      .IFE %I/16.-2   ;IF FIELD IN 3.WORD THEN
29                                      .MACRO %1
30                      .DO .PASS
31                              VALU3=(VALU3&(-MASK-1))+DATA
32              .ENDC
33              _%
34                                      .ENDC           ;END
35
```

```
!00C9 .MAIN
01                              K=I+1
02                              MASK=1-@K
03                              .IFL (@I&15.+MASK)          ;IF FIELD>THE REST BIT
04                                                          ;   IN THE WORD THEN
05                              .MACRO @1
06                      .DO     .PASS
07                              DATA=-@J
08                              MASK=(-1)B(16.-(@K-@I&15.))
09                              MASK=-MASK-1
10                              .IFN J==@1.-1
11                              DATA=-@J/3
12                              .ENDC
13                              .DC 1+@I&15.
14                              DATA=DATA/2
15                              .ENDC
16              .ENDC
17              -%
18                              .IFE @I/16.-1               ;IF FIELD IN 1. WORD THEN
19                              .MACRO @1
20                      .DO .PASS
21                      VALU1=(VALU1&(-MASK-1))+DATA
22              .ENDC
23              -%
24                              .ENDC                       ;END
25                              .IFE @I/16.-2               ;IF FIELD IN 2.WORD THEN
26                              .MACRO @1
27                      .DO .PASS
28                        VALU2=(VALU2&(-MASK-1))+DATA
29              .ENDC
30              -%
31                              .ENDC                       ;END
32                              .ENDC                       ;END EXTENDED FIELD.
33                              .MACRO @1
34                              .ENDC
35
```

```
10010 .MAIN
01                  _%
02                          I=I+2
03                                      .MACRO %1
04
05                                      J=J+1
06                  _%
07                          .ENDC                       ;END .DO ARGCT/2-2.
08                                      .MACRO %1
09                                      **.NOMAC 0
10                  ++                  VALU1
11                  ++                  VALU2
12                  ++                  VALU3
13                                      **.NOMA 1
14                  _%
15                  %
16
17
18
```

```
!0011 .MAIN
01                ;*******************************************
02                ;DEFINITION OF MICROINSTRUCTION PARAMETERS              791129 AAJ/LBJ
03                ;*******************************************
04
05                ;ALU FUNCTION AND CARRY,MIR(13:17)
06
07        000000          ADD=0           ;R+S
08        000001          ADD0=1          ;R+S+1
09        000002          ADDC=2          ;R+S+C
10        000003          ADDX=3          ;R+S+ADDCOND
11        000004          SUN=4           ;-R+S-1
12        000005          SUN0=5          ;-R+S
13        000006          SUNC=6          ;-R+S-1+C
14        000007          SUNX=7          ;-R+S-1+ADDCOND
15        000010          SUB=10          ;R-S-1
16        000011          SUB0=11         ;R-S
17        000012          SUBC=12         ;R-S-1+C
18        000013          SUBX=13         ;R-S-1+ADDCOND
19        000014          OR=14           ;R OR S
20        000020          AND=20          ;R AND S
21        000024          CAND=24         ;,-R AND S
22        000030          EXOR=30         ;R EXOR S
23        000034          EXNOR=34        ;,-(R EXOR S)
24
25
26                ;GENERAL REGISTER ADDRESSES,MIR(28:31) AND MIR(32:35)
27
28        000000          W0=0
29        000001          W1=1
30        000002          W2=2
31        000003          W3=3
32        000004          STAT=4
33        000005          IC=5            ;INSTR. COUNTER FOR LOGICAL ADDRESS
34        000006          CAUSE=6
35        000007          SB=7
36        000010          PC=10           ;INSTR. COUNTER FOR ABSOLUTE ADDRESS
37        000011          CNTR=11         ;WORKS IN CONJUNCTION WITH CONTR. OUTP. REG.
38        000012          WRK0=12
39        000013          WRK1=13
40        000014          WRK2=14
41        000015          WRK3=15
42        000016          WRK4=16
43        000017          WRK5=17
```

```
10012 .MAIN
01              ;INDIRECT ADDRESSING OF GENERAL REGISTERS
02
03              ;       THE FIRST 4 WORDS OF THE B-PART OF THE GENERAL REGISTERS
04              ;       CAN ONLY BE INDIRECTLY ADDRESSED
05
06      000000          W=0             ;GRB ADDR=W-FIELD OF INSTR REG
07      000001          WPRE=1          ;GRB ADDR=W-PRE OF INSTR REG
08      000002          X=2             ;GRB ADDR=X-FIELD OF INSTR REG
09      000003          GRX=3           ;GRB ADDR=I/OADDR(21:22)
10
11
12              ;SCRATCHPAD ADDRESSES, MIR(24:27)
13
14      000000          CPA=0           ;COMMON PROTECTED AREA LIMIT
15      000001          BASE=1
16      000002          LLIM=2          ;LOWER LIMIT
17      000003          ULIM=3          ;UPPER LIMIT
18      000004  MESS=4   ;MESSAGEADDRESS
19      000005  CLOW=5   ;CODE LOW
20      000006  CTOP=6   ;CODE TOP
21      000007  CTADDR=7 ; CONTROLLER_TABLE ADDRESS
22      000010          RTC=10          ;REAL TIME CLOCK
23      000011          C2=11           ;CONSTANT 00000002
24      000012          M2=12           ;CONSTANT 00000003
25      000013          M8=13           ;CONSTANT 00000377
26      000014          M12=14          ;CONSTANT 00007777
27      000015          ILIM=15         ;INTERRUPT LIMIT
28      000016          SP16=16
29      000017          SP17=17
30
31
32              ;SOURCE REGISTER ADDRESSES, MIR(24:27)
33
34      000000          IMOP=0          ;IMMEDIATE OPERAND
35      000001          SEXT=1          ;SIGN EXTENSION
36      000002          HWEXC=2         ;HALF WORD EXCHANGE
37      000003          DATI=3          ;DATA IN
38      000004          ILEV=4          ;INTERRUPT LEVEL
39      000005          TPIN=5          ;TCP DATA IN
40      000014          FPUR0=14        ; FPU RESULT FRACTION(0:23)
41      000015          FPUR1=15        ;            FRACTION(24:35) CON EXP(0:11)
42      000016          FPUST=16        ;            EXCEPTION(22:23)
```

```
!0013 .MAIN
01              ;DESTINATION REGISTER ADDRESSES AND I/O CONTROL, MIR(18:23)
02
03              ;       ADDRESSING OF THE I/O ADDR REG STARTS AN I/O OPERATION
04
05                                      ;0=NO LOAD
06      000001          CPUST=1         ;CPU STATUS
07      000002          READ=2          ;I/O READ
08      000022          READP=22        ;I/O READ WITH PROTECTION
09      000042          WRT=42          ;I/O WRITE
10      000062          WRTP=62         ;I/O WRITE WITH PROTECTION
11      000003          DATO=3          ;DATA OUT
12      000004          CNTRO=4         ;CONTROL OUTPUT REGISTER
13      000005          TPOUT=5         ;TCP DATA OUT
14      000006          IR=6            ;INSTRUCTION REGISTER
15      000007          MIX=7           ;MICRO INDEX REGISTER
16      000010          INTRG=10        ;INTERRUPT REGISTER
17      000011          CBCR=11         ;CPUBUS CONTROL REGISTER
18      000014          FPOPO=14        ;FPU OPERAND(0:23)
19      000015          FPOP1=15        ;FPU OPERAND(24:47)
20      000015          FPULL   =15     ;FPU OPERAND(24:47) + COMMAND
21      000035          FPINI   =35     ; "     "                "
22      000055          FPSTR   =55     ; "     "                "
23      000075          FPUSL   =75     ; "     "                "
24      000015          FPARM   =15     ; "     "                "
25      000035          FPINV   =35     ; "     "                "
26
27
28              ;JUMP CONDITION CONTROL, MIR(18)
29
30      000000          F=0             ;THE COMPLEMENTED VALUE OF THE COND. IS SELECTED
31      000001          T=1             ;THE TRUE VALUE OF THE COND. IS SELECTED
32
33
```

```
10014 .MAIN
01                ;JUMP CONDITIONS, MIR(19:23)
02
03        000000          FALSE=0         ;THIS CONDITION IS ALWAYS FALSE
04        000001          NNEG=1          ;RESULT>=0
05        000002          NZ=2            ;RESULT<>0
06        000003          OVFL=3          ;ARITHMETIC OVERFLOW
07        000004          CARRY=4         ;ARITHMETIC CARRY
08        000005          NORM=5          ;RES(0)<>RES(1)
09                                        ;
10                                        ;
11
12        000010          MMODE=10        ;MONITOR MODE
13        000011          EMODE=11        ;ESCAPE MODE
14        000012          AFAM=12         ;AFTER AM
15        000013          AFESC=13        ;AFTER ESCAPE
16        000014          IMSK=14         ;INTEGER MASK
17        000015          FPMSK=15        ;FLOATING POINT MASK
18        000016          INDIR=16        ;INDIRECT ADDRESSING, IR(9)
19        000017          LINK=17         ;W-FIELD<>0
20        000020          NMADR=20        ;NOT MEMORY ADDRESS, ADDR<8
21        000021          NWADR=21        ;NOT W-REG ADDRESS, 0>ADDR>=8
22        000022          ODD=22          ;ADDR(23)=1
23        000023          FPUNR=23        ;NOT CPUBUS UNIT READY
24        000024          BERR=24         ;BUS ERROR
25        000025          BTIM=25         ;BUS TIME OUT
26        000026          BNACK=26        ;BUS NACK
27        000027          BPAR=27         ;BUS PARITY ERROR
28
29        000030          INTR=30         ;INTERRUPT
30        000031          NTPIN=31        ;NOT TCP INPUT READY
31        000032          TPACK=32        ;TCP OUTPUT ACK
32        000033          RSTRT=33        ;RESTART ENABLE
33        000034          SHORT=34        ;TEST MODE SWITCH IN POSITION 'SHORT'
34        000035          TSTON=35        ;TEST SWITCH IN POSITION 'ON'
35        000036          PLOW=36         ;POWER LOW (PINT)
36                                        ;
37
38
39                ;SHIFT-IN CONTROL, MIR(24:25)
40
41        000000          Z=0             ;ZERO
42        000001          LNK=1           ;SHIFT LINK
43        000002          ADC=2           ;ADDCOND, LEFT SHIFTS ONLY
44        000002          SGN=2           ;SIGN EXTENSION, RIGHT SHIFTS ONLY
45                                        ;3 IS UNUSED
```

```
!0015 .MAIN
01                ;TEST CONTROL, MIR(26:27)
02
03        000000          NL=0              ;NO LOAD
04        000001          MC=1              ;ADDCOND:=MULTIPLY CONDITION
05        000002          DC=2              ;ADDCOND:=DIVIDE CONDITION
06        000003          DVS=3             ;DIVSIGN:=RESULT(0)
07
08
09                ;SEQUENCE CONTROL, MIR(1:3)
10
11        000000          CONT=0            ;CONTINUE
12        000001          CRTN=1            ;CONDITIONAL SUBROUTINE RETURN
13        000002          PUSH=2            ;PUSH AND CONTINUE
14        000003          POP=3             ;POP AND CONTINUE
15        000004          RTN=4             ;SUBROUTINE RETURN
16        000005          LRTN=5            ;LOOP RETURN, CONDITIONAL
17        000006          ADDR=6            ;CALL ADDRESS CALCULATION
18        000007          EXEC=7            ;CALL INSTRUCTION EXECUTION
```

```
10016 .MAIN
01              ;*********************************
02              ;DEFINITION OF MICROINSTRUCTIONS
03              ;*********************************
04
05
06              ;FORMAT 0: LOAD IMMEDIATE OPERAND REGISTER
07
08              ;       LDIM    OCTAL NO.,OCTAL NO.     NEXT
09
10              XVFD LDIM C,2000,0,35.,12.,47.,12.,15.,3
11
12
13              ;FORMAT 1: ARITHMETIC-LOGIC MICROINSTRUCTIONS WITH SCRATCHPAD DESTINATION
14              ;*************************************************************************
15
16              ;LOAD SCRATCHPAD
17
18              ;       AQS     ALUFUNC GRA,SP  NEXT
19              ;       ABS     ALUFUNC GRA,GRB,SP      NEXT
20              ;       ZQS     ALUFUNC SP      NEXT
21              ;       ZBS     ALUFUNC GRB,SP  NEXT
22              ;       ZAS     ALUFUNC GRA,SP  NEXT
23
24              XVFD AQS C,22000,0,29.,5,43.,4,39.,4,15.,3
25              XVFD ABS C,22200,0,29.,5,43.,4,47.,4,39.,4,15.,3
26              XVFD ZQS C,22400,0,29.,5,39.,4,15.,3
27              XVFD ZBS C,22600,0,29.,5,47.,4,39.,4,15.,3
28              XVFD ZAS C,23000,0,29.,5,43.,4,39.,4,15.,3
29
30
31              ;LOAD GRB,SCRATCHPAD
32
33              ;       AQBS    ALUFUNC GRA,GRB,SP      NEXT
34              ;       ABBS    ALUFUNC GRA,GRB,SP      NEXT
35              ;       ZQBS    ALUFUNC GRB,SP  NEXT
36              ;       ZBBS    ALUFUNC GRB,SP  NEXT
37              ;       ZABS    ALUFUNC GRA,GRB,SP      NEXT
38
39              XVFD AQBS C,26000,0,29.,5,43.,4,47.,4,39.,4,15.,3
40              XVFD ABBS C,26200,0,29.,5,43.,4,47.,4,39.,4,15.,3
41              XVFD ZQBS C,26400,0,29.,5,47.,4,39.,4,15.,3
42              XVFD ZBBS C,26600,0,29.,5,47.,4,39.,4,15.,3
43              XVFD ZABS C,27000,0,29.,5,43.,4,47.,4,39.,4,15.,3
```

```
10017 .MAIN
01              ;LOAD Q, SCRATCHPAD
02
03              ;       AQQS    ALUFUNC GRA,SP  NEXT
04              ;       ABQS    ALUFUNC GRA,GRB,SP      NEXT
05              ;       ZQQS    ALUFUNC SP      NEXT
06              ;       ZBQS    ALUFUNC GRB,SP  NEXT
07              ;       ZAQS    ALUFUNC GRA,SP  NEXT
08
09                      XVFD AQQS C,20000,0,29.,5,43.,4,39.,4,15.,3
10                      XVFD ABQS C,20200,0,29.,5,43.,4,47.,4,39.,4,15.,3
11                      XVFD ZQQS C,20400,0,29.,5,39.,4,15.,3
12                      XVFD ZBQS C,20600,0,29.,5,47.,4,39.,4,15.,3
13                      XVFD ZAQS C,21000,0,29.,5,43.,4,39.,4,15.,3
14
15
16              ;FORMAT 2: ARITHMETIC-LOGIC MICROINSTRUCTIONS WITH SCRATCHPAD SOURCE
17              ;*****************************************************************
18
19
20              ;NO LOAD
21
22              ;       SA      ALUFUNC SP,GRA  NEXT
23              ;       SQ      ALUFUNC SP      NEXT
24              ;       SZN     ALUFUNC SP      NEXT
25
26                      XVFD SA 0,43200,0,29.,5,39.,4,43.,4,15.,3
27                      XVFD SQ 0,43400,0,29.,5,39.,4,15.,3,
28                      XVFD SZN C,43600,0,29.,5,39.,4,15.,3
29
30
31              ;LOAD GRB
32
33              ;       SAB     ALUFUNC SP,GRA,GRB      NEXT
34              ;       SQB     ALUFUNC SP,GRB  NEXT
35              ;       SZB     ALUFUNC SP,GRB  NEXT
36
37                      XVFD SAB C,47200,0,29.,5,39.,4,43.,4,47.,4,15.,3
38                      XVFD SQB C,47400,0,29.,5,39.,4,47.,4,15.,3
39                      XVFD SZB C,47600,0,29.,5,39.,4,47.,4,15.,3
```

```
!0018 .MAIN
01              ;LOAD Q
02
03              ;       SAQ     ALUFUNC SP,GRA  NEXT
04              ;       SQQ     ALUFUNC SP      NEXT
05              ;       SZQ     ALUFUNC SP      NEXT
06
07                      XVFD SAQ C,41200,0,29.,5,39.,4,43.,4,15.,3
08                      XVFD SQQ C,41400,0,29.,5,39.,4,15.,3
09                      XVFD SZQ C,41600,0,29.,5,39.,4,15.,3
10
11              ;LOAD DESTINATION REGISTER
12
13              ;       SAR     ALUFUNC SP,GRA,DR       NEXT
14              ;       SQR     ALUFUNC SP,DR   NEXT
15              ;       SZR     ALUFUNC SP,DR   NEXT
16
17                      XVFD SAR C,43200,0,29.,5,39.,4,43.,4,35.,6,15.,3
18                      XVFD SQR C,43400,0,29.,5,39.,4,35.,6,15.,3
19                      XVFD SZR C,43600,0,29.,5,39.,4,35.,6,15.,3
20
21
22              ;LOAD GRB, DESTINATION REGISTER
23
24              ;       SABR    ALUFUNC SP,GRA,GRB,DR   NEXT
25              ;       SQBR    ALUFUNC SP,GRB,DR       NEXT
26              ;       SZBR    ALUFUNC SP,GRB,DR       NEXT
27
28                      XVFD SABR C,47200,0,29.,5,39.,4,43.,4,47.,4,35.,6,15.,3
29                      XVFD SQBR C,47400,0,29.,5,39.,4,47.,4,35.,6,15.,3
30                      XVFD SZBR C,47600,0,29.,5,39.,4,47.,4,35.,6,15.,3
31
32
33              ;LOAD Q, DESTINATION REGISTER
34
35              ;       SAQR    ALUFUNC SP,GRA,DR       NEXT
36              ;       SQQR    ALUFUNC SP,DR   NEXT
37              ;       SZQR    ALUFUNC SP,DR   NEXT
38
39                      XVFD SAQR C,41200,0,29.,5,39.,4,43.,4,35.,6,15.,3
40                      XVFD SQQR C,41400,0,29.,5,39.,4,35.,6,15.,3
41                      XVFD SZQR C,41600,0,29.,5,39.,4,35.,6,15.,3
```

```
10019 .MAIN
01              ;FORMAT 3: ARITHMETIC-LOGIC MICROINSTRUCTIONS WITH EXT. SOURCE REGISTER AND EXT. DEST. REGISTER
02              ;*****************************************************************************************
03
04
05              ;NO LOAD
06
07              ;       AQ      ALUFUNC GRA     NEXT
08              ;       AB      ALUFUNC GRA,GRB NEXT
09              ;       ZQ      ALUFUNC         NEXT
10              ;       ZB      ALUFUNC GRB     NEXT
11              ;       ZA      ALUFUNC GRA     NEXT
12              ;       RA      ALUFUNC SR,GRA  NEXT
13              ;       RQ      ALUFUNC SR      NEXT
14              ;       RZ      ALUFUNC SR      NEXT
15
16                      XVFD AQ 0,62000,0,29.,5,43.,4,15.,3
17                      XVFD AB 0,62200,0,29.,5,43.,4,47.,4,15.,3
18                      XVFD ZQ 0,62400,0,29.,5,15.,3
19                      XVFD ZB 0,62600,0,29.,5,47.,4,15.,3
20                      XVFD ZA 0,63000,0,29.,5,43.,4,15.,3
21                      XVFD RA 0,63200,0,29.,5,39.,4,43.,4,15.,3
22                      XVFD RQ 0,63400,0,29.,5,39.,4,15.,3
23                      XVFD RZ 0,63600,0,29.,5,39.,4,15.,3
24
25
26              ;LOAD GRB
27
28              ;       AQB     ALUFUNC GRA,GRB NEXT
29              ;       ABB     ALUFUNC GRA,GRB NEXT
30              ;       ZQB     ALUFUNC GRB     NEXT
31              ;       ZBB     ALUFUNC GRB     NEXT
32              ;       ZAB     ALUFUNC GRA,GRB NEXT
33              ;       RAB     ALUFNC  SR,GRA,GRB      NEXT
34              ;       RQB     ALUFUNC SR,GRB  NEXT
35              ;       RZB     ALUFUNC SR,GRB  NEXT
36
37                      XVFD AQB C,66000,0,29.,5,43.,4,47.,4,15.,3
38                      XVFD ABB C,66200,0,29.,5,43.,4,47.,4,15.,3
39                      XVFD ZQB C,66400,0,29.,5,47.,4,15.,3
40                      XVFD ZBB C,66600,0,29.,5,47.,4,15.,3
41                      XVFD ZAB C,67000,0,29.,5,43.,4,47.,4,15.,3
42                      XVFD RAB C,67200,0,29.,5,39.,4,43.,4,47.,4,15.,3
43                      XVFD RQB C,67400,0,29.,5,39.,4,47.,4,15.,3
44                      XVFD RZB C,67600,0,29.,5,39.,4,47.,4,15.,3
```

```
10C2C .MAIN
01                ;LOAD Q
02
03                ;       AQQ     ALUFUNC GRA      NEXT
04                ;       ABQ     ALUFUNC GRA,GRB  NEXT
05                ;       ZQQ     ALUFUNC          NEXT
06                ;       ZBQ     ALUFUNC GRB      NEXT
07                ;       ZAQ     ALUFUNC GRA      NEXT
08                ;       RAQ     ALUFUNC SR,GRA   NEXT
09                ;       RQQ     ALUFUNC SR       NEXT
10                ;       RZQ     ALUFUNC SR       NEXT
11
12                        XVFD AQQ C,60CC0,0,29.,5,43.,4,15.,3
13                        XVFD ABQ C,60200,0,29.,5,43.,4,47.,4,15.,3
14                        XVFD ZQQ C,60400,0,29.,5,15.,3
15                        XVFD ZBQ C,60600,0,29.,5,47.,4,15.,3
16                        XVFD ZAQ C,61000,0,29.,5,43.,4,15.,3
17                        XVFD RAQ C,61200,0,29.,5,39.,4,43.,4,15.,3
18                        XVFD RQQ C,61400,0,29.,5,39.,4,15.,3
19                        XVFD RZQ C,61600,0,29.,5,39.,4,15.,3
20
21
22                ;LOAD DEST. REG.
23
24                ;       AQR     ALUFUNC GRA,DR   NEXT
25                ;       ABR     ALUFUNC GRA,GRB,DR       NEXT
26                ;       ZQR     ALUFUNC DR       NEXT
27                ;       ZBR     ALUFUNC GRB,DR   NEXT
28                ;       ZAR     ALUFUNC GRA,DR   NEXT
29                ;       RAR     ALUFUNC SR,GRA,DR        NEXT
30                ;       RQR     ALUFUNC SR,DR    NEXT
31                ;       RZR     ALUFUNC SR,DR    NEXT
32
33                        XVFD AQR C,62000,0,29.,5,43.,4,35.,6,15.,3
34                        XVFD ABR C,62200,0,29.,5,43.,4,47.,4,35.,6,15.,3
35                        XVFD ZQR C,62400,0,29.,5,35.,6,15.,3
36                        XVFD ZBR C,62600,0,29.,5,47.,4,35.,6,15.,3
37                        XVFD ZAR C,63000,0,29.,5,43.,4,35.,6,15.,3
38                        XVFD RAR C,63200,0,29.,5,39.,4,43.,4,35.,6,15.,3
39                        XVFD RQR C,63400,0,29.,5,39.,4,35.,6,15.,3
40                        XVFD RZR C,63600,0,29.,5,39.,4,35.,6,15.,3
```

```
10021 .MAIN
01              ;LOAD GRB, DEST. REG.
02
03              ;       AQBR    ALUFUNC GRA,GRB,DR      NEXT
04              ;       ABBR    ALUFUNC GRA,GRB,DR      NEXT
05              ;       ZQBR    ALUFUNC GRB,DR  NEXT
06              ;       ZBBR    ALUFUNC GRB,DR  NEXT
07              ;       ZABR    ALUFUNC GRA,GRB,DR      NEXT
08              ;       RABR    ALUFUNC SR,GRA,GRB,DR   NEXT
09              ;       RQBR    ALUFUNC SR,GRB,DR       NEXT
10              ;       RZBR    ALUFUNC SR,GRB,DR       NEXT
11
12                      XVFD AQBR C,66000,0,29.,5,43.,4,47.,4,35.,6,15.,3
13                      XVFD ABBR C,66200,0,29.,5,43.,4,47.,4,35.,6,15.,3
14                      XVFD ZQBR C,66400,0,29.,5,47.,4,35.,6,15.,3
15                      XVFD ZBBR C,66600,0,29.,5,47.,4,35.,6,15.,3
16                      XVFD ZABR C,67000,0,29.,5,43.,4,47.,4,35.,6,15.,3
17                      XVFD RABR C,67200,0,29.,5,39.,4,43.,4,47.,4,35.,6,15.,3
18                      XVFD RQBR C,67400,0,29.,5,39.,4,47.,4,35.,6,15.,3
19                      XVFD RZBR C,67600,0,29.,5,39.,4,47.,4,35.,6,15.,3
20
21
22              ;LOAD Q, DEST. REG.
23
24              ;       AQQR    ALUFUNC GRA,DR  NEXT
25              ;       ABQR    ALUFUNC GRA,GRB,DR      NEXT
26              ;       ZQQR    ALUFUNC DR      NEXT
27              ;       ZBQR    ALUFUNC GRB,DR  NEXT
28              ;       ZAQR    ALUFUNC GRA,DR  NEXT
29              ;       RAQR    ALUFUNC SR,GRA,DR       NEXT
30              ;       RQQR    ALUFUNC SR,DR   NEXT
31              ;       RZQR    ALUFUNC SR,DR   NEXT
32
33                      XVFD AQQR C,60000,0,29.,5,43.,4,35.,6,15.,3
34                      XVFD ABQR C,60200,0,29.,5,43.,4,47.,4,35.,6,15.,3
35                      XVFD ZQQR C,60400,0,29.,5,35.,6,15.,3
36                      XVFD ZBQR C,60600,0,29.,5,47.,4,35.,6,15.,3
37                      XVFD ZAQR C,61000,0,29.,5,43.,4,35.,6,15.,3
38                      XVFD RAQR C,61200,0,29.,5,39.,4,43.,4,35.,6,15.,3
39                      XVFD RQQR C,61400,0,29.,5,39.,4,35.,6,15.,3
40                      XVFD RZQR C,61600,0,29.,5,39.,4,35.,6,15.,3
```

```
!0022 .MAIN
01              ;SIGN EXTENSION
02
03              ;       EXT     GRA,GRB NEXT            ;GRB:=12 EXT GRA(12) CON GRA(12:23)
04
05                      XVFD EXT C,65660,400,43.,4,47.,4,15.,3
06
07
08              ;LOAD DEST. REG. & EXTEND SIGN
09
10              ;       EXTR    GRA,GRB,DR      NEXT    ;DR:=GRA
11              ;                                       ;GRB:=12 EXT GRA(12) CON GRA(12:23)
12
13                      XVFD EXTR C,65660,400,43.,4,47.,4,35.,6,15.,3
14
15
16              ;EXCHANGE HALF-WORDS
17
18              ;       SWAP    GRA,GRB NEXT            ;GRB:=GRA(12:23) CON GRA(0:11)
19
20                      XVFD SWAP C,65660,1000,43.,4,47.,4,15.,3
21
22
23              ;CLEAR INTERRUPT BIT
24
25              ;       CLIN    GRA     NEXT            ;INTRG(GRA):=0
26
27                      XVFD CLIN C,63060,100000,43.,4,15.,3
```

```
!0023 .MAIN
01              ;FORMAT 4: SHIFT MICROINSTRUCTIONS
02              ;**********************************
03
04
05              ;SHIFT LEFT GRB
06
07              ;       SSLA    ALUFUNC,GRA,GRB,SI,TST  NEXT
08              ;       SSLB    ALUFUNC,GRB,SI,TST      NEXT
09
10                      XVFD SSLA C,116200,0,29.,5,43.,4,47.,4,37.,2,39.,2,15.,3
11                      XVFD SSLB C,116600,0,29.,5,47.,4,37.,2,39.,2,15.,3
12
13
14              ;SHIFT LEFT GRB CON Q
15
16              ;       DSLA    ALUFUNC,GRA,GRB,SI,TST  NEXT
17              ;       DSLB    ALUFUNC,GRB,SI,TST      NEXT
18
19                      XVFD DSLA C,114200,0,29.,5,43.,4,47.,4,37.,2,39.,2,15.,3
20                      XVFD DSLB C,114600,0,29.,5,47.,4,37.,2,39.,2,15.,3
21
22
23              ;SHIFT RIGHT GRB
24
25              ;       SSRA    ALUFUNC,GRA,GRB,SI,TST  NEXT
26              ;       SSRB    ALUFUNC,GRB,SI,TST      NEXT
27
28                      XVFD SSRA C,112200,0,29.,5,43.,4,47.,4,37.,2,39.,2,15.,3
29                      XVFD SSRB C,112600,0,29.,5,47.,4,37.,2,39.,2,15.,3
30
31
32              ;SHIFT RIGHT GRB CON Q
33
34              ;       DSRA    ALUFUNC,GRA,GRB,SI,TST  NEXT
35              ;       DSRB    ALUFUNC,GRB,SI,TST      NEXT
36
37                      XVFD DSRA C,110200,0,29.,5,43.,4,47.,4,37.,2,39.,2,15.,3
38                      XVFD DSRB C,110600,0,29.,5,47.,4,37.,2,39.,2,15.,3,
```

```
10024 .MAIN
01              ;SHIFT LEFT GRB, CONDITION TEST
02
03              ;       SLAB    ALUFUNC,GRA,GRB,SI,TST,T/F,COND NEXT
04              ;       SLZB    ALUFUNC,GRB,SI,TST,T/F,COND     NEXT
05              ;       SLZA    ALUFUNC,GRA,GRB,SI,TST,T/F,COND NEXT
06
07                      XVFD SLAB C,116200,0,29.,5,43.,4,47.,4,37.,2,39.,2,30.,1,35.,5,15.,3
08                      XVFD SLZB C,116600,0,29.,5,47.,4,37.,2,39.,2,30.,1,35.,5,15.,3
09                      XVFD SLZA C,117000,0,29.,5,43.,4,47.,4,37.,2,39.,2,30.,1,35.,5,15.,3
10
11
12              ;SHIFT LEFT GRB CON Q, CODITION TEST
13
14              ;       DLAB    ALUFUNC,GRA,GRB,SI,TST,T/F,COND NEXT
15              ;       DLZB    ALUFUNC,GRB,SI,TST,T/F,COND     NEXT
16              ;       DLZA    ALUFUNC,GRA,GRB,SI,TST,T/F,COND NEXT
17
18                      XVFD DLAB C,114200,0,29.,5,43.,4,47.,4,37.,2,39.,2,30.,1,35.,5,15.,3
19                      XVFD DLZB C,114600,0,29.,5,47.,4,37.,2,39.,2,30.,1,35.,5,15.,3
20                      XVFD DLZA C,115000,0,29.,5,43.,4,47.,4,37.,2,39.,2,30.,1,35.,5,15.,3
21
22
23              ;SHIFT RIGHT GRB, CONDITION TEST
24
25              ;       SRAB    ALUFUNC,GRA,GRB,SI,TST,T/F,COND NEXT
26              ;       SRZB    ALUFUNC,GRB,SI,TST,T/F,COND,    NEXT
27              ;       SRZA    ALUFUNC,GRA,GRB,SI,TST,T/F,COND NEXT
28
29                      XVFD SRAB C,112200,0,29.,5,43.,4,47.,4,37.,2,39.,2,30.,1,35.,5,15.,3
30                      XVFD SRZB C,112600,0,29.,5,47.,4,37.,2,39.,2,30.,1,35.,5,15.,3
31                      XVFD SRZA C,113000,0,29.,5,43.,4,47.,4,37.,2,39.,2,30.,1,35.,5,15.,3
32
33
34              ;SHIFT RIGHT GRB CON Q, CONDITION TEST
35
36              ;       DRAB    ALUFUNC,GRA,GRB,SI,TST,T/F,COND NEXT
37              ;       DRZB    ALUFUNC,GRB,SI,TST,T/F,COND     NEXT
38              ;       DRZA    ALUFUNC,GRA,GRB,SI,TST,T/F,COND NEXT
39
40                      XVFD DRAB C,110200,0,29.,5,43.,4,47.,4,37.,2,39.,2,30.,1,35.,5,15.,3
41                      XVFD DRZB C,110600,0,29.,5,47.,4,37.,2,39.,2,30.,1,35.,5,15.,3
42                      XVFD DRZA C,111000,0,29.,5,43.,4,47.,4,37.,2,39.,2,30.,1,35.,5,15.,3
43
44
```

```
10025 .MAIN
01              ;NO SHIFT, LOAD GRB, CONDITION TEST
02
03              ;       NSAB    ALUFUNC,GRA,GRB,TST,T/F,COND    NEXT
04              ;       NSZB    ALUFUNC,GRB,TST,T/F,COND        NEXT
05              ;       NSZA    ALUFUNC,GRA,GRB,TST,T/F,COND    NEXT
06              ;       NSAQ    ALUFUNC,GRA,GRB,TST,T/F,COND    NEXT
07              ;       NSZQ    ALUFUNC,GRB,TST,T/F,COND        NEXT
08
09                      XVFD NSAB C,106200,0,29.,5,43.,4,47.,4,39.,2,30.,1,35.,5,15.,3
10                      XVFD NSZB C,106600,0,29.,5,47.,4,39.,2,30.,1,35.,5,15.,3
11                      XVFD NSZA C,107000,0,29.,5,43.,4,47.,4,39.,2,30.,1,35.,5,15.,3
12                      XVFD NSAQ C,106000,0,29.,5,43.,4,47.,4,39.,2,30.,1,35.,5,15.,3
13                      XVFD NSZQ C,106400,0,29.,5,47.,4,39.,2,30.,1,35.,5,15.,3
```

```
10C26 .MAIN
01              ;FORMAT 5: MULTIPLY MICROINSTRUCTIONS
02              ;**********************************
03
04
05              ;       MULD    ALUFUNC,GRA,GRB,SI,TST,T/F,COND NEXT    ;MULTIPLY, DOUBLE SHIFT
06              ;       MULS    ALUFUNC,GRA,GRB,SI,TST,T/F,COND NEXT    ;MULTIPLY, SINGLE SHIFT
07              ;       MULN    ALUFUNC,GRA,GRB,TST,T/F,COND    NEXT    ;MULTIPLY, NO SHIFT
08              ;       MULQ    ALUFUNC,GRA,TST,T/F,COND        NEXT    ;MULTIPLY WITH Q-REGISTER, NO SHIFT
09
10                      XVFD MULD C,130200,0,29.,5,43.,4,47.,4,37.,2,39.,2,30.,1,35.,5,15.,3
11                      XVFD MULS C,132200,0,29.,5,43.,4,47.,4,37.,2,39.,2,30.,1,35.,5,15.,3
12                      XVFD MULN C,126200,0,29.,5,43.,4,47.,4,39.,2,30.,1,35.,5,15.,3
13                      XVFD MULQ C,120000,0,29.,5,43.,4,39.,2,30.,1,35.,5,15.,3
```

```
!0027 .MAIN
01              ;FORMAT 6: DIVIDE MICROINSTRUCTIONS
02              ;**********************************
03
04
05              ;       DIVD    ALUFUNC,GRA,GRB,SI,TST,T/F,COND NEXT    ;DIVIDE, DOUBLE SHIFT
06              ;       DIVS    ALUFUNC,GRA,GRB,SI,TST,T/F,COND NEXT    ;DIVIDE, SINGLE SHIFT
07              ;       DIVN    ALUFUNC,GRA,GRB,TST,T/F,COND    NEXT    ;DIVIDE, NO SHIFT
08              ;       DIVR    ALUFUNC,GRA,GRB,SI,TST,T/F,COND NEXT    ;DIVIDE, RIGHT SHIFT
09
10                      XVFD DIVD C,154200,0,29.,5,43.,4,47.,4,37.,2,39.,2,30.,1,35.,5,15.,3
11                      XVFD DIVS C,156200,0,29.,5,43.,4,47.,4,37.,2,39.,2,30.,1,35.,5,15.,3
12                      XVFD DIVN C,146200,0,29.,5,43.,4,47.,4,39.,2,30.,1,35.,5,15.,3
13                      XVFD DIVR C,152200,0,29.,5,43.,4,47.,4,37.,2,39.,2,30.,1,35.,5,15.,3
```

```
10C2E MAIN
01              ;FORMAT 7: JUMP MICROINSTRUCTIONS
02              ;**********************************
03
04
05              ;CONDITIONAL JUMPS
06
07              ;       JMP     T/F,COND,ADDR   NEXT
08              ;       JMX     T/F,COND        NEXT
09
10                      AVFD JMP C,162000,0,30.,1,35.,5,47.,12.,15.,3
11                      XVFD JMX C,162004,0,30.,1,35.,5,15.,3
12
13
14              ;CONDITIONAL SUBRCUTINE CALLS
15
16              ;       CAL     T/F,COND,ADDR   NEXT
17              ;       CAX     T/F,COND        NEXT
18
19                      AVFD CAL C,162010,0,30.,1,35.,5,47.,12.,15.,3
20                      XVFD CAX C,162014,0,30.,1,35.,5,15.,3
21
22
23              ;CONDITIONAL JUMPS WITH I/O SYNCHRONIZATION
24
25              ;       WJMP    T/F,COND,ADDR   NEXT
26              ;       WJMX    T/F,COND        NEXT
27
28                      AVFD WJMP C,162020,0,30.,1,35.,5,47.,12.,15.,3
29                      XVFD WJMX C,162024,0,30.,1,35.,5,15.,3
30
31
32              ;CONDITIONAL SUBRCUTINE CALLS WITH I/O SYNCHRONIZATION
33
34              ;       WCAL    T/F,COND,ADDR   NEXT
35              ;       WCAX    T/F,COND        NEXT
36
37                      AVFD WCAL C,162030,0,30.,1,35.,5,47.,12.,15.,3
38                      XVFD WCAX C,162034,0,30.,1,35.,5,15.,3
```

```
10029 .MAIN
01              .EOT
```

```
 0030 .MAIN
```

```
CCOC SOURCE LINES IN ERROR
```

```
 0001 GPU MICROPROGRAM  GI APR 1982
01                000001          .NOMA   1
02                        ;INTERRUPT JUMP TABLE
03                        ;*********************
04
05                        ;       THE FIRST 3 LOCATIONS CONTAIN A JUMP TABLE
06                        ;       WITH JUMPS TO THE INTERRUPT SERVICE ROUTINES
07
08 C0000 434400000003           JMP     F,F,POWUP                       ;POWER ON
09 C0001 434400002676           JMP     F,F,TCPIN                       ;INPUT FROM TCP
10 C0002 034400003156           JMP     F,F,SINGL                       ;SINGLE INSTRUCTION
```



```
!0002 GPU MICROPROGRAM  GI APR 1982
01                        ;POWER UP
02                        ;********
03
04 C0003 415560040011 POWUP:   ZBBR    AND,CNTR,CNTRO                  ;CNTR:=CNTRO:=0
05 C0004 034402000210          CAL     F,F,INITR                       ;BASE:=0
06                                                                     ;C2:=2, M2:=3, M8:=377, M12:=7777
07                                                                     ;CBCR:=0,7070
08 C0005 415760010004          RZBR    AND,IMOP,STAT,CPUST             ;STAT:=CPUST:=00000000 (MMODE:=C)
09
10 C0006 034400760006 WL1:     JMP     T,PLOW,WL1                      ;WAIT UNTIL NOT POWER LOW
11 C0007 034402000231          CAL     F,F,CINTR                       ;CLEAR ALL INTERRUPTS
```

```
;CCC3 GPU MICROPROGRAM  GI APR 1982
)1                      ;IDLE LOOP AND SYSTEM FAULT
)2                      ;**************************
)3
)4                      SYFLT:
)5 CC010 415560040011 IDLE:   ZBBR    AND,CNTR,CNTRO                    ;CNTR:=CNTRO:=0, (RUN:=0)
)6 CC011 434400300011 ILOOP:  JMP     F,INTR,ILOOP                      ;WAIT FOR INTERRUPT
)7 CC012 415754002012         RZB     OR,ILEV,WRK0                      ;WRK0:=ILEV
)8 CC013 000400000017         LDIM    0000,0017
)9 CC014 415660070252         RABR    AND,IMOP,WRK0,WRK0,MIX            ;MIX:=WRK0:=20 EXT 0 CON WRK0(20:23)
10 CC015 00040000000C3        LDIM    0,3
11 CC016 014645000240         RA      SUN0,IMOP,WRK0                    ;-3+WRK0
12 CC017 434401010CCC   JMX   F,NNEG      ; IF ILEV < 3 THEN GOTO MIX
13 CC020 014614100240   CLIN  WRK0        ; CLEAR INTERRUPT
14 CC021 400400000010   LDIM  0,10        ;
15 CC022 014645000240   RA    SUN0,IMOP,WRK0 ; ALU := ILEV - 8
16 CC023 034400410026   JMP   T,NNEG,CPUIN   ; IF ILEV >= 8 THEN GOTO CPUIN
17 CC024 434400000011   JMP   F,F,ILOOP   ; GOTO ILOOP
```

```
!0004 GPU MICROPROGRAM  GI APR 1982
01                      ;AUTOLOAD SEQUENCES  ARE NOT USED
02                      ;*******************
03
04
05                      ; EXTERNAL INTERRUPT
06                      ; *******************
07                      ;
08                      ; ENTRY CONDITIONS FOR EXTERNAL INTERRUPT:
09                      ; WORK0 = ILEV ; 8 <= ILEV < 16
10                      ;
11                      ; DEVICE_NO = ( ILEV - 7 ) * 4
12                      ;
13                      ; THE ADDRESS OF CONTROLLER TABLE IS :
14                      ;     CTADDR = CORE(8) + DEVICE_NO SHIFT 3
15                      ;
16                      ; FORMAT OF A CONTROLLER TABLE:
17                      ;
18                      ;       CT + C :  CHANNEL PROGRAM ADDRESS
19                      ;       CT + 2 :  STANDARD STATUS AREA ADDRESS
20                      ;       CT + 4 :  INTERRUPT DESTINATION (HOST CPU DEVICENUMBER)
21                      ;       CT + 6 :  INTERRUPT LEVEL
22                      ;
23
24 CC025 434404000026 CPUI1: WJMP F,F,CPUIN          ; WAIT BUS READY
25                      CPUIN:
26 CC026 41075422540C         SZR  CR,M8,READP       ; IOADDR := 8; READP
27 CC027 411645005652         SAB  SUN0,M8,WRK0,WRK0 ; WORK0 := ILEV - 8
28 CC030 023440000252         SSLA ADD,WRK0,WRK0,Z,NL ; WORK0 :=
29 CC031 411641005252         SAB  ADD0,M2,WRK0,WRK0  ; WORK0 SHIFT 2  +  4
30 CC032 023440000252         SSLA ADD,WRK0,WRK0,Z,NL ;
31 CC033 015440000252         ABB  ADD,WRK0,WRK0      ;       SHIFT 3
32 CC034 03440464001C         WJMP T,BERR,SYFLT      ; IF BUSERROR THEN GOTO SYFLT
33 CC035 015640221653         RABR ADD,DATI,WRK0,WRK1,READP ; I/O-ADDR := WORK1 := C(8) + DEV
34 CC036 40461000366C         ZAS  SUB,WRK1,CTADDR   ; CT_ADDR := WORK1
35 CC037 03440464001C         WJMP T,BERR,SYFLT      ; IF BUSERROR THEN GOTO SYFLT
36 CC040 015754001412         RZB  CR,DATI,WRK0      ; WORK0 := CH
37 CC041 410640224640         SAR  ADD,C2,WRK0,READP ; I/O-ADDR := CH + 2, READP
38 CC042 011745004416         SZB  SUN0,C2,WRK4      ; WORK4 := 7777,7776 <* MASK *>
```

```
!00C5 GPU MICROPROGRAM  GI APR 1982
)1
)2                      ;
)3                      ; FORMAT OF CHANNEL PROGRAM :
)4                      ;       CH + C :  OPERATION
)5                      ;       CH + 2 :  MESSAGE AREA BASE
)6                      ;       CH + 4 :  NOT_USED
)7                      ;
)8                      ;       OPERATION = 1 SHIFT 8 :  START GPU
)9                      ;                 = 0 SHIFT 8 :  STOP GPU
10
11 CC043 034404640010           WJMP T,BERR,SYFLT       ; WAIT, IF BUSERROR THEN GOTO SYFLT
12 CC044 014354001400           RZQ  OR,DATI            ; Q := MESSAGE_ADDR
13 CC045 414614220240           ZAR  OR,WRK0,READP      ; I/O-ADDR := CH_BASE, READP
14 CC046 000400007400           LDIM C00C,7400          ; MASK FOR OPERATION
15 CC047 015754000017           RZB  OR,IMOP,WRK5       ; WORK5 := MASK
16 CC050 034404640010           WJMP T,BERR,SYFLT       ; WAIT, IF BUSERROR THEN GOTO SYFLT
17 CC051 015660001777           RAB  AND,DATI,WRK5,WRK5 ; WORK5 := OPERATION SHIFT 8
18
19 CC052 034400020131           JMP  F,NZ,STOP          ; IF OP = 0 THEN GOTO STOP
20
21
```

```
!0006 GPU MICROPROGRAM  GI APR 1982
01
02
03
04                      ;
05                      ; FORMAT OF MESSAGE
06                      ;    MESS +  0 :  NEXT BUFFER
07                      ;         +  2 :  PREVIOUS BUFFER
08                      ;         +  4 :  RECEIVER OR ANSWER TYPE
09                      ;         +  6 :  SENDER
10                      ;         +  8 :  COMMAND, MODE
11                      ;         + 10 :  FIRST CODE (SEE BELOW)
12                      ;         + 12 :  LAST CODE
13                      ;
14                      ; CODE EXECUTION STARTS IN ADDRESS : FIRST_CODE + 4
15                      ;
16
17 CC053 000400000012          LDIM COOC,0012          ; IMOP := 10
18 CC054 015700220007          RQBR ADD,IMOP,SB,READP ; SB := IMOP := ADDR_MESS(5), READP
19 CC055 014614000160          ZA   CR,SB              ; NO OP
20 CC056 034404640010          WJMP T,BERR,SYFLT       ; WAIT, IF BUSERROR THEN GOTO SYFLT
21 CC057 415660001745          RAB  AND,DATI,WRK4,IC ; IC := DATA_IN(0:22) CON 0
22
23 CC060 010640224560          SAR  ADD,C2,SB,READP  ; I/O-ADDR := MESS(6), READP
24 CC061 411644005167          SAB  SUN,M2,SB,SB       ; SB := ADDR_MESS(3);
25 CC062 034404640010          WJMP T,BERR,SYFLT       ; WAIT, IF BUSERROR THEN GOTO SYFLT
26 CC063 415754001410          RZB  CR,DATI,PC         ; PC := LAST_CODE
27 CC064 014614220160          ZAR  CR,SB,READP        ; I/O-ADDR := SB; READP;
28 CC065 000400000140          LDIM COOC,0140          ; IMOP := 96;
29 CC066 034404640010          WJMP T,BERR,SYFLT       ; WAIT, IF BUSSERROR THEN GOTO SYFLT
30 CC067 415754001407          RZB  CR,DATI,SB         ; SB := INT_PROCESS_DESCR_BASE
```

```
00C7 GPU MICROPROGRAM  GI APR 1982
01                     ;
02                     ; FORMAT OF PROCESSDESCRIPTION (IN PARTS)
03                     ;      PR_DESCR +  88 :  STATUS
04                     ;               +  96 :  CPA
05                     ;               +  98 :  BASE
06                     ;               + 100 :  LOWER WRITE LIMIT
07                     ;               + 102 :  UPPER WRITE LIMIT
08                     ;
09 CC070 415640220167        RABR ADD,IMOP,SB,SB,READP ; SB := I/O-ADDR := SB + 96, READP
10 CC071 400400000016        LDIM C000,0016           ; IMOP := 14
11 CC072 03440464001C        WJMP T,BERR,SYFLT        ; WAIT, IF BUSERROR THEN GOTO SYFLT
12 CC073 0143540014CC        RZQ  CR,DATI             ; Q := CPA
13 CC074 011640224567        SABR ADD,C2,SB,SB,READP ; SB := I/O-ADDR := SB + 2; READP
14 CC075 40451000000C        ZQS  SUB,CPA             ; CPA := Q
15 CC076 03440464001C        WJMP T,BERR,SYFLT        ; WAIT, IF BUSERROR THEN GOTO SYFL
16 CC077 0143540014CC        RZQ  CR,DATI             ; Q := BASE
17 CC100 011640224567        SABR ADD,C2,SB,SB,READP ; SB := I/O-ADDR := SB + 2, READP
18 CC101 00451000040C        ZQS  SUB,BASE            ; BASE := Q;
19 CC102 03440464001C        WJMP T,BERR,SYFLT        ; WAIT, IF BUSERROR THEN GOTO SYFLT
20 CC103 0143540014CC        RZQ  CR,DATI             ; Q := LLIM
21 CC104 011640224567        SABR ADD,C2,SB,SB,READP ; SB := I/O-ADDR := SB + 2, READP
22 CC105 00451000100C        ZQS  SUB,LLIM            ; LLIM := Q
23 CC106 03440464001C        WJMP T,BERR,SYFLT        ; WAIT, IF BUSERROR THEN GOTO SYFLT
24 CC107 0143540014CC        RZQ  CR,DATI             ; Q := ULIM
25 CC110 415645220167        RABR SUNO,IMOP,SB,SB,READP ; SB := I/O_ADDR := SB - 14, READP
26 CC111 40451000140C        ZQS  SUB,ULIM            ; ULIM := Q
27 CC112 03440464001C        WJMP T,BERR,SYFLT        ; WAIT, IF BUSERROR THEN GOTO SYFLT
28 CC113 015754011404        RZBR CR,DATI,STAT,CPUST ; STAT := CPUSTATUS := DATA_IN
29
30
31
```

```
1008 GPU MICROPROGRAM  GI APR 1982
01
02
03
04 C0114 00040300000C          LDIM  0300,0000        ; LOAD INTEGER AND FLOATINGPOINT EXC ACTIVE
05 C0115 415660010104          RABR AND,IMOP,STAT,STAT,CPUST ; FROM PROCESSDESCR.STATUS
06 C0116 411640000618          SAB  ADD,BASE,PC,PC    ; PC := PC + BASE
07 C0117 0046100032CC          ZAS  SUB,PC,CTOP       ; CODE_TOP := PC
08 C0120 411640000532          SAB  ADD,BASE,IC,WRK0  ; WORK0 := IC + BASE
09 C0121 404610002648          ZAS  SUB,WRK0,CLOW     ; CODE_LOW := WORK0
10 C0122 414405000208          AQ   SUN0,PC           ; IF CODE_TOP > ULIM
11 C0123 034400012670          JMP  F,NNEG,INT2       ; THEN GOTO INT2;
12 C0124 010645001248          SA   SUN0,LLIM,WRK0    ; IF CODE_LOW < LLIM
13 C0125 034400012670          JMP  F,NNEG,INT2       ; THEN GOTO INT2
14 C0126 411641005125          SAB  ADD0,M2,IC,IC     ; IC := IC + 4
15 C0127 011640000538          SAB  ADD,BASE,IC,PC    ; PC := IC + BASE
16 C0130 034400000241          JMP  F,F,MLOOP         ; GOTO MLOOP
17
18                     ; STOP
19                     ; ****
20 C0131 4C0410000C0C STOP:  LDIM 1000,0000          ; TIME_OUT
21 C0132 415754000016 SEND1: RZB  OR,IMOP,WRK4       ; WORK4 := STATUS
22 C0133 411645002617 SEND2: SAB  SUN0,CLOW,PC,WRK5 ; WORK5 := INSTR REL TO CODESTART
23 C0134 434400000143          JMP  F,F,SEND          ; GOTO SEND
24
25                     ; ILLEGAL OPERATION
26                     ; ******************
27 C0135 40042000000C ILLOP: LDIM 200C,0000          ; ILLEGAL INSTRUCTION
28 C0136 434400000132          JMP  F,F,SEND1         ; GOTO SEND1
29
30                     ; CODE STOP CS
31                     ; ************
32
33 C0137 415560000016 CS:    ZBB  AND,WRK4           ; WORK4 := 0
34 C0140 411754003017         SZB  OR,CTOP,WRK5       ; WORK5 :=
35 C0141 011645002777         SAB  SUN0,CLOW,WRK5,WRK5 ;  CTOP - CLOW
36 C0142 011640004777         SAB  ADD,C2,WRK5,WRK5    ;   + 2;
37
38
```

```
!00C9 GPU MICROPROGRAM  GI APR 1982
01
02
03
04                      ; STATUS AT RETURN :
05                      ;   ST + 0 : CHPG + 12
06                      ;   ST + 2 : BYTES COUNT
07                      ;   ST + 4 : CHARS COUNT
08                      ;   ST + 6 : STATUS
09                      ;            STATUS :
10                      ;                1<14 : CODE ADDRESSING FAULT
11                      ;                1<15 : INTEGER UNDERFLOW
12                      ;                1<16 : INTEGER OVERFLOW
13                      ;                1<17 : FLOATING POINT UNDERFLOW (*** SEE NOTE ***)
14                      ;                1<18 : FLOATING POINT OVERFLOW
15                      ;                1<19 : READ/WRITE ADDRESS FAULT
16                      ;                1<20 : BUSERROR
17                      ;                1<21 : TIME OUT
18                      ;                1<22 : ILLEGAL INSTRUCTION
19                      ;
20                      ; NOTE: IN CASE OF FLOATING POINT UNDERFLOW THE RESULT IS SET TO
21                      ; ****  FLOATING POINT ZERO AND NO EXCEPTION IS SET UP
22                      ;
23                      ; SEND STATUS
24                      ; ***********
25 CC143 01035422340C SEND:   SZQR CR,CTADDR,READP  ; Q := I/O-ADDR := CT, READP
26 CC144 000400000014         LDIM C00C,0014        ; IMOP := 12
27 CC145 03440464001C         WJMP T,BERR,SYFLT     ; WAIT, IF BUSERROR THEN GOTO SYFLT
28 CC146 415754001413         RZB  CR,DATI,WRK1     ; WORK1 := CHPG
29 CC147 01030022440C         SQQR ADD,C2,READP     ; Q := I/O-ADDR := CT + 2, READP
30 CC150 414614030360         ZAR  CR,WRK5,DATO     ; DATA_OUT := BYTES COUNT
31 CC151 011754004407         SZB  CR,C2,SB         ; SB := 2
32 CC152 03440464001C         WJMP T,BERR,SYFLT     ; WAIT, IF BUSERROR GOTO SYFLT
33 CC153 015640621567         RABR ADD,DATI,SB,SB,WRTP ; SB :=I/O-ADDR := STATUS + 2, WRITEP
34 CC154 415614000375         ZAB  CR,WRK5,WRK3     ; WORK3 := BYTES;
35 CC155 03440464001C         WJMP T,BERR,SYFLT     ; WAIT, IF BUSERROR THEN GOTO SYFLT
36 CC156 014614030340         ZAR  CR,WRK4,DATO     ; DATA_OUT := STATUS
37 CC157 010641625160         SAR  ADDO,M2,SB,WRTP  ; I/O-ADDR := STATUS + 6, WRITEP
38 CC160 022540004015         SSRB ADD,WRK3,SGN,NL  ; WORK3 := BYTES // 2;
39 CC161 03440464001C         WJMP T,BERR,SYFLT     ; WAIT, IF BUSERROR THEN GOTO SYFLT
40 CC162 014440030375         ABR  ADD,WRK5,WRK3,DATO ; DATA_OUT := CHARS COUNT
41 CC163 410640624560         SAR  ADD,C2,SB,WRTP   ; I/O-ADDR := STATUS + 4, WRITEP
42 CC164 414514000000         ZQ   OR               ; NO OP
43 CC165 03440464001C         WJMP T,BERR,SYFLT     ; WAIT, IF BUSERROR THEN GOTO SYFLT
```

```
!001C GPU MICROPROGRAM  GI APR 1982
01
02
03
04
05 CC166 414640003026C          RAR   ADD,IMOP,WRK1,DATO ; DATA_OUT := CHPG + 12;
06 CC167 410645624560           SAR   SUNO,C2,SB,WRTP   ; I/O-ADDR := ST, WRITEP
07 CC170 414514000CCC           ZQ    CR                ; NO OP
08 CC171 034404640010           WJMP  T,BERR,SYFLT      ; WAIT, IF BUSERROR THEN GOTO SYFLT
09
10                        ; SEND RETURN INTERRUPT
11 CC172 010701225CCC           SQR   ADDO,M2,READP     ; I/O-ADDR := CT +6, READP
12 CC173 414514000CCC           ZQ    CR                ; NO OP
13 CC174 034404640010           WJMP  T,BERR,SYFLT      ; WAIT, IF BUSERROR THEN GOTO SYFLT
14 CC175 414754031400           RZR   CR,DATI,DATO      ; DATA_OUT := INTERRUPT LEVEL
15 CC176 410700224400           SQR   ADD,C2,READP      ; I/O-ADDR := CT + 4, READP
16 CC177 414514000CCC           ZQ    CR                ; NO OP
17 CC200 034404640010           WJMP  T,BERR,SYFLT      ; WAIT, IF BUSERROR THEN GOTO SYFLT
18 CC201 414754421400           RZR   CR,DATI,WRT       ; I/O-ADDR := CPU_ADDR, WRITE
19 CC202 414514000CCC           ZQ    CR                ; NO OP
20 CC203 034404640010           WJMP  T,BERR,SYFLT      ; WAIT, IF BUSERROR THEN GOTO SYFLT
21 CC204 434402000227           CAL   F,F,CLIM          ; CLEAR CODE_LIMITS
22 CC205 034400000010           JMP   F,F,IDLE          ; GOTO IDLE
23
24                         FILL 210
```



```
!0011 GPU MICROPROGRAM  GI APR 1982
01                        ;REAL TIME CLOCK INTERRUPT   IS NOT USED
02                        ;**************************
03
04                        ;       RTC IS INCREMENTED BY 1 EVERY 0.1 MILLISECOND.
05                        ;       EVERY 25.6 MILLISECONDS THE INTERVAL TIMER INTERRUPT IS SET.
06                        ;       AT ENTRY: WRK0=ILEV
```

```
!C012 GPU MICROPROGRAM  GI APR 1982
01                      ;SUBROUTINE INITIALIZE REGISTERS AND CONSTANTS
02
03 CC210 404050000652 INITR:  ABQS    SUB,WRK0,WRK0,BASE                ;BASE:=0, Q:=-1
04 CC211 414104000000         ZQQ     SUN                               ;Q:=-2
05 CC212 004104004400         ZQQS    SUN,C2                            ;C2:=2, Q:=-3
06 CC213 404504005000         ZQS     SUN,M2                            ;M2:=3
07
08 CC214 400400000010         LDIM    0000,0010
09 CC215 014354000000         RZQ     OR,IMOP
10 CC216 004510005400         ZQS     SUB,M8                            ;M8:=10
11 CC217 400400004000         LDIM    0000,4000
12 CC220 014354000000         RZQ     OR,IMOP
13 CC221 004510002000         ZQS     SUB,MESS                          ; MESS := 0,-2048
14
15 CC222 000400007777         LDIM    0,7777
16 CC223 014354000000         RZQ     OR,IMOP
17 CC224 404510006000         ZQS     SUB,M12                           ;M12:=7777
18 CC225 000400007070         LDIM    0,7070                            ;
19 CC226 414754110000         RZR     OR,IMOP,CBCR                      ;CBCR:=0,7070
20 CC227 404450002567 CLIM:   ABS   SUB,SB,SB,CLOW       ; CODE_LOW := 0;
21 CC230 604451003167         ABS   SUB0,SB,SB,CTOP RTN ; CODE_TOP := -1;
22
23
24
25
26                      ;SUBROUTINE CLEAR INTERRUPTS
27
28 CC231 000400000017 CINTR:  LDIM    0000,0017
29 CC232 015754000012         RZB     OR,IMOP,WRK0                      ;WRK0:=15.
30 CC233 014614100240 CLEAR:  CLIN    WRK0                              ;CLEAR INTERRUPT BIT(WRK0)
31 CC234 015544000012         ZBB     SUN,WRK0                          ;WRK0:=WRK0-1
32 CC235 634400420233         JMP     T,NZ,CLEAR      RTN               ;IF WRK0<>0 THEN GOTO CLEAR ELSE RETURN
33
34
35                      ;SUBROUTINE DELAY
36
37 CC236 015754000012 DELAY:  RZB     OR,IMOP,WRK0                      ;WRK0:=IMOP
38 CC237 015544000012 DELLP:  ZBR     SUN,WRK0                          ;WRK0:=WRK0-1
39 CC240 234400420237         JMP     T,NZ,DELLP      RTN               ;IF WRK0<>0 THEN GOTO DELLP ELSE RETURN
```

```
10013 GPU MICROPROGRAM  GI APR 1982
01
02                      ;MAINLOOP
03                      ;********
04
05                      ;       THE MAINLOOP FETCHES AN INSTRUCTION, TESTS FOR INTERRUPT AND BUSERRORS
06                      ;       DURING FETCH, AND CALLS SUBROUTINES FOR ADDRESS CALCULATION AND
07                      ;       INSTRUCTION EXECUTION
08 CC241 01064500260C MLOOP: SA    SUNO,CLOW,PC     ; ALU := PC - CODE_LOW
09 CC242 03440001267C        JMP   F,NNEG,INT2      ; IF NEG THEN GOTO INT2
10 CC243 01461422C2CC         ZAR     OR,PC,READP                     ; I/O ADDR:=PC, READP
11
12 CC244 0106510032CC MLOP1: SA    SUBO,CTOP,PC   ; ALU := CODE_TOP - PC
13 CC245 03440001267C        JMP   F,NNEG,INT2    ; IF NEG THEN GOTO INT2
14 CC246 034404642656         WJMP    T,BERR,FINT                     ; WAIT, IF BUSERROR THEN GOTO FINT
15 CC247 015754061412         RZBR    OR,DATI,WRKO,IR                 ; WRKO:= DATAIN; IR:=DATAIN(0:11)
16
17 CC250 315354000652 AFTCH:  EXT     WRKO,WRKO    ADDR               ;WRKO:=12 EXT WRKO(12) CON WRKO(12:23)
18                                                                    ; CALL ADDR CALCULATION
19 CC251 751640004525         SAB     ADD,C2,IC,IC    EXEC            ; IC:=IC+2, CALL INSTR EXECUTION
20 CC252 011640224610         SABR    ADD,C2,PC,PC,READP              ; PC:= PC + 2, READP
21 CC253 034400300244         JMP     F,INTR,MLOP1                    ; IF NOT INTR THEN GOTO MLOP1
22
23
24                      ;INTERRUPT
25
26 CC254 415754002012 INTT:   RZB     OR,ILEV,WRKO                    ; WRKO:= ILEV, BITS(0:19) IS UNDEFINED
27 CC255 000400000017         LDIM    0000,0017
28 CC256 415660070252         RABR    AND,IMOP,WRKO,WRKO,MIX          ; MIX:= WRKO:= 20 EXT 0 CON WRKO(20:23)
29 CC257 410645005240         SA      SUNO,M2,WRKO                    ; IF ILEV<3 THEN
30 CC260 434401010000         JMX     F,NNEG                          ; GOTO (MIX)
31 CC261 014614100240   CLIN WRKO                ; CLEAR INTERRUPT
32 CC262 010645005640   SA   SUNO,M8,WRKO        ;
33 CC263 03440041CC25   JMP  T,NNEG,CPUI1        ; IF ILEV >= 8 THEN GOTO CPUI1
34 CC264 034400000244   JMP  F,F,MLOP1           ; GOTO MLOP1
```

```
!0014 GPU MICROPROGRAM  GI APR 1982
01                      ;NOT MEMORY ADDRESS
02
03
04                      ;ADDRESS CALCULATION SUBROUTINES
05                      ;**********************************
06
07                      ;       THE ADDRESS CALCULATIONS ARE PERFORMED BY SUBROUTINES
08                      ;       WHICH ARE CALLED VIA A SEPARATE ADDRESS TABLE
09                      ;       INPUTS TO THE TABLE IS: AFTER ESCAPE, AFTERAM, IR(8), IR(10,11)
10                      ;       WRKC = EXTENDED DISPLACEMENT
11
12                      ;AFTER ESCAPE = 1       MUST NOT BE USED
13                      ;*****************
14
```

```
!0015 GPU MICROPROGRAM  GI APR 1982
01
02                      ;AFTER ESCAPE = 1, AFTER AM = 1     MUST NOT BE USED
03                      ;**********************************
04
05                      ;AFTER AM = C
06                      ;*************
07
08 CC265 061614160247 DIR:     NSZA    OR,WRKO,SB,NL,F,INDIR     CRTN    ; SB:=WRKO; IF NOT INDIRECT THEN RETURN
09 CC266 034400000313          JMP     F,F,INADR                         ; ELSE GOTO INADR;
10
11 CC267 415614000247 REL:     ZAB     OR,WRKO,SB                        ; SB:= WRKO
12 CC270 061440160127          NSAB    ADD,IC,SB,NL,F,INDIR      CRTN    ; SB:=SB+IC; IF NOT INDIRECT THEN RETURN
13 CC271 034400000313          JMP     F,F,INADR                         ; ELSE GOTO INADR;
14
15 CC272 414040000242 INDEX:   ABQ     ADD,WRKO,X                        ; Q:= WRKO + X
16 CC273 061514160007          NSZQ    OR,SB,NL,F,INDIR          CRTN    ; SB:= Q, IF NOT INDIRECT THEN RETURN
17 CC274 034400000313          JMP     F,F,INADR                         ; ELSE GOTO INADR;
18
19 CC275 414040000242 RELX:    ABQ     ADD,WRKO,X                        ; Q:= WRKO + X
20 CC276 461400160127          NSAQ    ADD,IC,SB,NL,F,INDIR      CRTN    ; SB:= IC+Q, IF NOT INDIRECT THEN RETURN
21 CC277 034400000313          JMP     F,F,INADR                         ; ELSE GOTO INADR
22
23
```

```
10016 GPU MICROPROGRAM  GI APR 1982
01                      ;AFTER AM = 1
02                      ;*************
03
04 CC300 415440000127 AMR:     ABB     ADD,IC,SB                          ; SB:= SB + IC
05 CC301 415440000247 AMD:     ABB     ADD,WRK0,SB                        ; SB:= SB + WRK0
06 CC302 400410000000          LDIM    1000,0000                          ; IMOP:= 10000000
07 CC303 015664010104          RABR    CAND,IMOP,STAT,STAT,CPUST          ; AFTER AM:= 0
08 CC304 634400560313          JMP     T,INDIR,INADR              RTN     ; IF INDIRECT THEN GOTO INADR
09                                                                        ; ELSE RETURN;
10
11 CC305 415440000127 AMRX:    ABB     ADD,IC,SB                          ; SB:=SB+IC
12 CC306 414040000242 AMX:     ABQ     ADD,WRK0,X                         ; Q:=X+WRK0
13 CC307 415400000167          AQB     ADD,SB,SB                          ; SB:=Q+SB
14 CC310 400410000000          LDIM    1000,0
15 CC311 015664010104          RABR    CAND,IMOP,STAT,STAT,CPUST          ;AFTER AM:=0
16 CC312 634400560313          JMP     T,INDIR,INADR              RTN     ; IF INDIRECT THEN GOTO INADR
17                                                                        ; ELSE RETURN;
18
19                      ;INDIRECT ADDRESSING
20                      ;*******************
21
22 CC313 011640004525 INADR:   SAB     ADD,C2,IC,IC                       ;IC:=IC+2;
23
24 CC314 034402000321 INAD1:   CAL     F,F,GETOP                          ;WRK0:=ADDRESS
25 CC315 355614000247          ZAB     OR,WRK0,SB                 EXEC    ;SB:=ADDRESS, CALL INSTR EXECUTION
26 CC316 011640224610          SABR    ADD,C2,PC,PC,READP                 ;I/O ADDR:=PC:=PC+2, READP
27 CC317 034400700254          JMP     T,INTR,INTT                        ;IF INTERRUPT THEN GOTO INTT
28 CC320 034400000244          JMP     F,F,MLOP1                          ;GOTO MLOP1  (MAIN LOOP)
```

```
!0017 GPU MICROPROGRAM  GI APR 1982
01                      ;SUBROUTINE GET OPERAND
02                      ;**********************
03
04                      ;       FETCHES AN OPERAND FROM THE MEMORY LOCATION ADDRESSED BY
05                      ;       SB + BASE, IF ADDRESS IS WITHIN UPPER AND LOWER LIMIT OR FROM
06                      ;       LOCATION ADDRESSED BY SB, IF ADDRESS IS WITHIN CPA AREA.
07                      ;       RETURNS WITH WRK0 = FETCHED OPERAND
08                      ;                    WRK1 = PHYSICAL ADDRESS
09                      ;                    SB = LOGICAL ADDRESS
10
11 C0321 411640220573 GETOP:  SABR    ADD,BASE,SB,WRK1,READP          ; I/O ADDR:=WRK1:=BASE + SB, READP
12 C0322 034400600332         JMP     T,NMADR,NOTM                    ; IF I/O ADDR < 8 THEN GOTO NOTM
13 C0323 410645001260         SA      SUNO,LLIM,WRK1                  ; IF ADDRESSES < LOWER LIMIT
14 C0324 434400010331         JMP     F,NNEG,GTCPA                    ; THEN GOTO GTCPA
15 C0325 410650001660         SA      SUB,ULIM,WRK1                   ; IF ADDRESS >= UPPER LIMIT
16 C0326 434400010331         JMP     F,NNEG,GTCPA                    ; THEN GOTO GTCPA
17 C0327 034404642656         WJMP    T,BERR,INTB                     ; WAIT, IF BUSERROR THEN GOTO INTB
18 C0330 615754001412         RZB     OR,DATI,WRK0    RTN             ; WRK0:= DATAIN, RETURN
19
20 C0331 434404000332 GTCPA:  WJMP    F,FALSE,NOTM                    ; WAIT, GOTO NOTM
21 C0332 015614220173 NOTM:   ZABR    OR,SB,WRK1,READP                ; I/O ADDR:=WRK1:=SB, READP
22 C0333 434400210341         JMP     F,NWADR,GTREG                   ; IF 8 > I/O ADDR >= 0 THEN GOTO GTREG
23 C0334 034400602672         JMP     T,NMADR,INT0                    ; IF I/O ADDR < 0 THEN GOTO INT0
24 C0335 410650000160         SA      SUB,CPA,SB                      ; IF SB >= CPA
25 C0336 434400012672         JMP     F,NNEG,INT0                     ; THEN GOTO INT0
26 C0337 034404642656         WJMP    T,BERR,INTB                     ; IF BUSERROR THEN GOTO INTB
27 C0340 615754001412         RZB     OR,DATI,WRK0    RTN             ; WRK0:= DATAIN, RETURN
28
29 C0341 414154000003 GTREG:  ZBQ     OR,GRX                          ; Q:= W(I/O ADDR)
30 C0342 615514000012         ZQB     OR,WRK0 RTN                     ; WRK0:= Q, RETURN
31
```

```
!0018 GPU MICROPROGRAM  GI APR 1982
01                      ;INSTRUCTION EXECUTION SUBROUTINES
02                      ;**********************************
03                      ;
04                      ;       THE INSTRUCTION EXECUTION SUBROUTINES ARE CALLED
05                      ;       VIA A SEPARATE ADDRESS TABLE.
06                      ;       INPUTS TO THE TABLE IS: ESCAPE MODE, IR(0:5)
07                      ;       SB = EFFECTIVE ADDRESS
08
```

```
0019 GPU MICROPROGRAM  GI APR 1982
)1                      ;ADDRESS HANDLING
)2                      ;*****************
)3
)4
)5                      ;MODIFY NEXT ADDRESS, AM, OPCODE = 11
)6
)7 CC343 400410000CCC AM:       LDIM    1000,0000
)8 CC344 615654010104           RABR    OR,IMOP,STAT,STAT,CPUST RTN     ; AFTER AM:= 1, RETURN
)9
10
11                      ;LOAD ADDRESS, AL, OPCODE = 13
12
13 CC345 215614000160 AL:       ZAB     OR,SB,W RTN                     ; W(IR):= SB, RETURN
14
15
16                      ;LOAD ADDRESS COMPLEMENTED, AC, OPCODE = 41
17                      ;
18
19 CC346 011664005104 AC:       SAB     CAND,M2,STAT,STAT               ; EX(22):= EX(23):= 0
20 CC347 415611000160           ZAB     SUB0,SB,W                       ; W(IR):= - SB
21 CC350 434400430352           JMP     T,OVFL,INOFL                    ; IF OVERFLOW THEN GOTO INOFL
22 CC351 215542000004           ZBB     ADDC,STAT       RTN             ; EX(23):= CARRY
23
24
25                      ;INTEGER OVERFLOW
26
27 CC352 411642004504 INOFL:    SAB     ADDC,C2,STAT,STAT               ; EX(22):= 1, EX(23):= CARRY
28 CC353 234400542674           JMP     T,IMSK,INTA     RTN             ; IF INT MASK=1 THEN GOTO INTA ELSE RTN
```

```
!0C2C GPU MICROPROGRAM  GI APR 1982
01                      ;WORD INSTRUCTIONS WITH MEMORY REFERENCE
02                      ;**************************************
03
04                      ;LOAD REGISTER, RL, OPCODE = 24
05
06 CC354 034402000321 RL:       CAL     F,FALSE,GETOP                   ; CALL GETOP, WRK0:=OPERAND
07 CC355 61561400024C           ZAB     OR,WRK0,W       RTN             ; W(IR):= WRK0, RETURN
08
09
10                      ;LOGICAL AND, LA, OPCODE = 4
11
12 CC356 034402000321 LA:       CAL     F,FALSE,GETOP                   ; CALL GETOP, WRK0:= OPERAND
13 CC357 21546000024C           ABB     AND,WRK0,W      RTN             ; W(IR):= W(IR) AND WRK0, RETURN
14
15
16                      ;LOGICAL OR, LO, OPCODE = 5
17
18 CC360 034402000321 LO:       CAL     F,FALSE,GETOP                   ; CALL GETOP, WRK0:= OPERAND
19 CC361 61545400024C           ABB     OR,WRK0,W       RTN             ; W(IR):= W(IR) OR WRK0, RETURN
20
21
22                      ;LOGICAL EXCLUSIVE OR, LX, OPCODE = 6
23
24 CC362 034402000321 LX:       CAL     F,FALSE,GETOP                   ; CALL GETOP, WRK0:= OPERAND
25 CC363 61547000024C           ABB     EXOR,WRK0,W     RTN             ; W(IR):= W(IR) EXOR WRK0
```

```
!0021 GPU MICROPROGRAM  GI APR 1982
01                      ;ADD INTEGER WORD, WA, OPCODE = 7
02
03 CC364 034402000321 WA:       CAL     F,FALSE,GETOP                   ; CALL GETOP, WRK0:= OPERAND
04 CC365 011664005104           SAB     CAND,M2,STAT,STAT               ; EX(22):= EX(23):= 0
05 CC366 015440000240           ABB     ADD,WRK0,W                      ; W(IR):= W(IR) + WRK0
06 CC367 434400430352           JMP     T,OVFL,INOFL                    ; IF OVERFLOW THEN GOTO INOFL
07 CC370 215542000004           ZBB     ADDC,STAT       RTN             ; EX(23):= CARRY
08
09
10                      ;SUBTRACT INTEGER WORD, WS, OPCODE = 10
11
12 CC371 034402000321 WS:       CAL     F,FALSE,GETOP                   ; CALL GETOP,WRK0:= OPERAND
13 CC372 011664005104           SAB     CAND,M2,STAT,STAT               ; EX(22):= EX(23):= 0
14 CC373 015445000240           ABB     SUNO,WRK0,W                     ; W(IR):= W(IR) - WRK0
15 CC374 434400430352           JMP     T,OVFL,INOFL                    ; IF OVERFLOW THEN GOTO INOFL
16 CC375 215542000004           ZBB     ADDC,STAT       RTN             ; EX(23):= CARRY
```

```
!0022 GPU MICROPROGRAM  GI APR 1982
01                    ;MULTIPLY INTEGER WORD, WM, OPCODE=12
02
03 CC376 034402000321 WM:     CAL     F,F,GETOP                       ;WRK0:=OPERAND
04 CC377 41415400000C         ZBQ     OR,W                            ;Q:=W
05 CC400 415560000001         ZBB     AND,WPRE                        ;WPRE:=0
06 CC401 42215400040C1        DRZB    OR,WPRE,Z,MC,F,F                ;ADCND:=IF Q(23)=1 THEN 0 ELSE 1
07                                                                    ;WPRE CON Q:=0 CON WPRE CON Q(0:22)
08 CC402 000400000027         LDIM    0,27
09 CC403 115754000013         RZB     OR,IMOP,WRK1     PUSH           ;FOR WRK1:=23
10 CC404 415544000013         ZBB     SUN,WRK1                        ;STEP -1 UNTIL 0 DO
11                                                                    ;BEGIN
12 CC405 666040024641         MULD    ADD,WRK0,WPRE,SGN,MC,F,NZ LRTN  ;WPRE:=IF ADCND=0 THEN WPRE+WRK0
13                                                                    ;                 ELSE WPRE+0
14                                                                    ;ADCND:=IF Q(23)=1 THEN 0 ELSE 1
15                                                                    ;WPRE CON Q:=WPRE(0) CON WPRE CON Q(0:22)
16                                                                    ;END
17 CC406 426045004241         MULD    SUN0,WRK0,WPRE,SGN,NL,F,F       ;WPRE:=IF ADCND=0 THEN WPRE-WRK0
18                                                                    ;                 ELSE WPRE-0
19                                                                    ;WPRE CON Q:=WPRE(0) CON WPRE CON Q(0:22)
20 CC407 61551400000C         ZQB     OR,W             RTN            ;W:=Q, RTN
```

```
!0023 GPU MICROPROGRAM  GI APR 1982
01                      ;DIVIDE INTEGER WORD, WD, OPCODE=30
02
03 CC410 034402000321 WD:      CAL     F,F,GETOP                       ;WRK0:=OPERAND, (=DIVISOR)
04 CC411 011664005104 WDBF:    SAB     CAND,M2,STAT,STAT               ;EX(22):=EX(23):=0
05 CC412 021554001412          NSZB    OR,WRK0,DVS,F,F                 ;DVS:=WRK0(0), (=DIVISOR SIGN)
06 CC413 034400020452          JMP     F,NZ,WDOFL                      ;IF DIVISOR=0 THEN GOTO WDOFL
07 CC414 014154000001          ZBQ     OR,WPRE
08 CC415 415514000013          ZQB     OR,WRK1                         ;WRK1:=WPRE, (=DIVIDEND(0:23))
09 CC416 414154000000          ZBQ     OR,W                            ;Q:=W, (=DIVIDEND(24:47))
10
11 CC417 023154001013          DLZB    OR,WRK1,Z,DC,F,F                ;ADCND:=IF WRK1(0)=DVS THEN 1 ELSE 0
12                                                                     ;WRK1 CON Q:=WRK1(1:23) CON Q CON 0
13
14 CC420 433043005253          DIVD    ADDX,WRK0,WRK1,ADC,DC,F,F       ;WRK1:=IF ADCND=1 THEN WRK1-WRK0
15                                                                     ;                 ELSE WRK1+WRK0
16                                                                     ;ADCND:=IF WRK1(0)=DVS THEN 1 ELSE 0
17                                                                     ;WRK1 CON Q:=WRK1(1:23) CON Q CON ADCND
18
19                      ;       AFTER THE FIRST STEP OF DIVISION,QUOTIENT OVERFLOW IS ANTICIPATED BY
20                      ;       COMPARING THE CARRY FROM BIT(0) OF THE ALU WITH THE SIGN OF THE DIVIDEND
21
22 CC421 434400440425          JMP     T,CARRY,WD2                     ;IF CARRY=1 THEN GOTO WD2
23 CC422 414554000001          ZB      OR,WPRE                         ;IF CARRY=0 AND DIVIDEND>=0
24 CC423 034400410427          JMP     T,NNEG,WD1                      ;THEN GOTO WD1
25 CC424 434400000452          JMP     F,F,WDOFL                       ;ELSE GOTO WDOFL
26
27 CC425 015554000001 WD2:     ZBB     OR,WPRE                         ;IF CARRY=1 AND DIVIDEND>=0
28 CC426 434400410452          JMP     T,NNEG,WDOFL                    ;THEN GOTO WDOFL
```

```
0C24 GPU MICROPROGRAM  GI APR 1982
01 CC427 400400000026 WD1:    LDIM    0,26
02 CC430 515754000014         RZB     OR,IMOP,WRK2     PUSH            ;FOR WRK2:=22
03 CC431 015544000014         ZBB     SUN,WRK2                         ;STEP -1 UNTIL 0 DO
04                                                                     ;BEGIN
05 CC432 273043025253         DIVD    ADDX,WRK0,WRK1,ADC,DC,F,NZ LRTN  ;WRK1:=IF ADCND=1 THEN WRK1-WRK0
06                                                                     ;                  ELSE WRK1+WRK0
07                                                                     ;ADCND:=IF WRK1(0)=DVS THEN 1 ELSE 0
08                                                                     ;WRK1 CON Q:=WRK1(1:23) CON Q CON ADCND
09                                                                     ;END
10 CC433 023154004015         DLZP    OR,WRK3,ADC,NL,F,F               ;Q:=Q(1:23) CON ADCND
11 CC434 031443001253         DIVN    ADDX,WRK0,WRK1,DC,F,F            ;WRK1:=IF ADCND=1 THEN WRK1-WRK0
12                                                                     ;                  ELSE WRK1+WRK0
13                                                                     ;ADCND:=IF WRK1(0)=DVS THEN 1 ELSE 0
14 CC435 023154004015         DLZB    OR,WRK3,ADC,NL,F,F               ;Q:=Q(1:23) CON ADCND
15 CC436 425440000253         MULN    ADD,WRK0,WRK1,NL,F,F             ;WRK1:=IF ADCND=0 THEN WRK1+WRK0
16                                                                     ;                  ELSE WRK1+0
17
18                    ;       AT THIS POINT Q=QUOTIENT AND WRK1=REMAINDER
19                    ;       THE EQUATION: DIVIDEND=QUOTIENT*DIVISOR+REMAINDER, IS FULFILLED
20
21 CC437 014451000272         AB      SUB0,WRK1,WRK0                   ;IF REMAINDER=DIVISOR
22 CC440 034400020445         JMP     F,NZ,WDCOR                       ;THEN GOTO WDCOR
23 CC441 014614000260         ZA      OR,WRK1                          ;IF REMAINDER=0
24 CC442 434400020450         JMP     F,NZ,WDEND                       ;THEN GOTO WDEND
25 CC443 414470000261         AB      EXOR,WRK1,WPRE                   ;IF SIGN OF DIVIDEND=SIGN OF REMAINDER
26 CC444 034400410450         JMP     T,NNEG,WDEND                     ;THEN GOTO WDEND
27 CC445 415445000253 WDCOR:  ABB     SUN0,WRK0,WRK1                   ;REMAINDER:=REMAINDER-DIVISOR
28 CC446 414101000000         ZQQ     ADD0                             ;QUOTIENT:=QUOTIENT+1
29 CC447 034400430452         JMP     T,OVFL,WDOFL                     ;IF OVERFLOW THEN GOTO WDOFL
30 CC450 015614000261 WDEND:  ZAB     OR,WRK1,WPRE                     ;WPRE:=REMAINDER
31 CC451 615514000000         ZQB     OR,W             RTN             ;W:=QUOTIENT, RTN
32
33 CC452 011654004504 WDOFL:  SAB     OR,C2,STAT,STAT                  ;EX(22):=1
34 CC453 234400542674         JMP     T,IMSK,INTA      RTN             ;IF INT MASK=1 THEN GOTO INTA ELSE RTN
```

```
10025 GPU MICROPROGRAM  GI APR 1982
01
02                      ;HALF-WORD INSTRUCTIONS WITH MEMORY REFERENCE
03                      ;*****************************************
04
05
06                      ;LOAD HALF REGISTER, HL, OPCODE=3
07
08 CC454 034402000321 HL:      CAL     F,F,GETOP                       ; WRK0:= OPERAND
09 CC455 034400620457          JMP     T,ODD,HLRGT                     ; IF ODD ADDR THEN GOTO HLRGT
10 CC456 015354001252          SWAP    WRK0,WRK0                       ; WRK0:= WRK0(12:23) CON WRK0(0:11)
11 CC457 411660006252 HLRGT:   SAB     AND,M12,WRK0,WRK0               ; WRK0:= 12 EXT 0 CON WRK0(12:23)
12 CC460 414154000CCC          ZBQ     OR,W                            ; Q:= W
13 CC461 011724006CCC          SQB     CAND,M12,W                      ; W:=W(0:11) CON 12 EXT 0
14 CC462 615454000240          ABB     OR,WRK0,W       RTN             ; W:=W OR WRK0, RTN
15
16
17                      ;LOAD INTEGER HALF-WORD (BYTE), ZERO EXTENSION, BZ, OPCODE=23
18
19 CC463 034402000321 BZ:      CAL     F,F,GETOP                       ; WRK0:=OPERAND
20 CC464 434400620466          JMP     T,ODD,BZRGT                     ; IF ODD ADDR THEN GOTO BZRGT
21 CC465 015354001252          SWAP    WRK0,WRK0                       ; WRK0:=WRK0(12:23) CON WRK0(0:11)
22 CC466 211660006240 BZRGT:   SAB     AND,M12,WRK0,W  RTN             ; W:=12 EXT 0 CON WRK0(12:23), RTN
23
24
25                      ;LOAD INTEGER HALF-WORD (BYTE), SIGN EXTENSION, BL, OPCODE=2
26
27 CC467 034402000321 BL:      CAL     F,F,GETOP                       ; WRK0:= OPERAND
28 CC470 434400620472          JMP     T,ODD,BLRGT                     ; IF ODD ADDR THEN GOTO BLRGT
29 CC471 015354001252          SWAP    WRK0,WRK0                       ; WRK0:= WRK0(12:23) CON WRK0(0:11)
30 CC472 615354000640 BLRGT:   EXT     WRK0,W  RTN                     ; W:= 12 EXT WRK(12) CON WRK0(12:23), RTN
```

```
10026 GPU MICROPROGRAM  GI APR 1982
01                      ;ADD INTEGER HALF-WORD (BYTE), BA, OPCODE = 22
02
03 CC473 034402000321 BA:       CAL     F,FALSE,GETOP                   ; CALL GETOP, WRK0:= OPERAND
04 CC474 011664005104           SAB     CAND,M2,STAT,STAT               ; EX(22):=EX(23):=0
05 CC475 434400620477           JMP     T,ODD,BARGT                     ; IF I/O ADDR(23) = 1 THEN GOTO BARGT
06 CC476 015354001252           SWAP    WRK0,WRK0                       ; WRK0:= WRK0(12:23) CON WRK0(0:11)
07 CC477 015354000652 BARGT:    EXT     WRK0,WRK0                       ; WRK0:= 12EXT WRK0(12) CON WRK0(12:22)
08 CC500 015440000240           ABB     ADD,WRK0,W                      ; W:= W + WRK0
09 CC501 434400430352           JMP     T,OVFL,INOFL                    ; IF OVERFLOW THEN GOTO INOFL
10 CC502 215542000004           ZBB     ADDC,STAT       RTN             ; EX(23) := CARRY, RETURN
11
12
13                      ;SUBTRACT INTEGER HALF-WORD (BYTE), BS, OPCODE = 21
14
15 CC503 034402000321 BS:       CAL     F,FALSE,GETOP                   ; CALL GETOP, WRK0:= OPERAND
16 CC504 011664005104           SAB     CAND,M2,STAT,STAT               ; EX(22):=EX(23):=0
17 CC505 434400620507           JMP     T,ODD,BSRGT                     ; IF ODD ADDR THEN GOTO BSRGT
18 CC506 015354001252           SWAP    WRK0,WRK0                       ; WRK0:= WRK0(12:23) CON WRK0(0:11)
19 CC507 015354000652 BSRGT:    EXT     WRK0,WRK0                       ; WRK0:= 12EXT WRK0(12) CON WRK(12:23)
20 CC510 015445000240           ABB     SUN0,WRK0,W                     ; W:= W - WRK0
21 CC511 434400430352           JMP     T,OVFL,INOFL                    ; IF OVERFLOW THEN GOTO INOFL
22 CC512 215542000004           ZBB     ADDC,STAT       RTN             ; EX(23):= CARRY
23
24
25                      ;LOAD EXCEPTION REGISTER, XL, OPCODE = 20
26
27 CC513 034402000321 XL:       CAL     F,FALSE,GETOP                   ; CALL GETOP, WRK0:= OPERAND
28 CC514 434400620516           JMP     T,ODD,XLRGT                     ; IF ODD ADDR THEN GOTO XLRGT
29 CC515 015354001252           SWAP    WRK0,WRK0                       ; WRK0:=WRK0(12:23) CON WRK0(0:11)
30 CC516 400400000007 XLRGT:    LDIM    0000,0007                       ; IMOP:= 0000007
31 CC517 015660000252           RAB     AND,IMOP,WRK0,WRK0              ; WRK0:= 21EXT0 CON WRK0(21:23)
32 CC520 415664000104           RAB     CAND,IMOP,STAT,STAT             ; STAT:= STAT(0:20) CON 3 EXT 0
33 CC521 215454000244           ABB     OR,WRK0,STAT    RTN             ; STAT:= STAT(0:20) CON WRK0(21:23), RETURN
```

```
!0027 GPU MICROPROGRAM  GI APR 1982
01                      ;STORE INSTRUCTIONS
02                      ;******************
03
04
05                      ;SUBROUTINE STORE WORD
06
07                      ;STORES W(IR) IN THE ADDRESSED MEMORY LOCATION
08                      ;       RETURNS WITH: I/O ADDRESS=ADDRESS OF MEMORY LOCATION
09                      ;                   : WRK1=SB+BASE
10
11 CC522 411640000573 STWD:   SAB     ADD,BASE,SB,WRK1                ;WRK1:=SB+BASE
12 CC523 41064500126C         SA      SUNO,LLIM,WRK1                  ;IF WRK1<LOWER LIMIT
13 CC524 434400010532         JMP     F,NNEG,STWD1                    ;THEN GOTO STWD1
14 CC525 41065000166C         SA      SUB,ULIM,WRK1                   ;IF WRK1>=UPPER LIMIT
15 CC526 434400012672         JMP     F,NNEG,INTO                     ;THEN GOTO INTO
16 CC527 014554030000         ZBR     OR,W,DATO                       ;DATAOUT:=W(IR)
17 CC530 014554620013         ZBR     OR,WRK1,WRTP                    ;I/O ADDR:=WRK1, WRITEP
18 CC531 634400602672         JMP     T,NMADR,INTO    RTN             ;IF I/O ADDR<8 THEN GOTOINTO ELSE RTN
19 CC532 41415400000C STWD1:  ZBQ     OR,W                            ;Q:=W(IR)
20 CC533 41064500556C STWD2:  SA      SUNO,M8,SB                      ;IF SB>=8
21 CC534 034400412672         JMP     T,NNEG,INTO                     ;THEN GOTO INTO
22 CC535 414554220007         ZBR     OR,SB,READP                     ;I/O ADDR:=SB, READP
23 CC536 434400612672         JMP     T,NWADR,INTO                    ;IF 0=<I/O ADDR<8 THEN GOTO INTO
24 CC537 615514000003         ZQB     OR,GRX          RTN             ;W(I/O ADDR):=Q, RTN
```

```
CC28 GPU MICROPROGRAM  G1 APR 1982
01                      ;STORE REGISTER, RS, OPCODE=27
02
03 CC540 034402000522 RS:       CAL     F,F,STWD                        ;MEM(ADDR):=W(IR)
04 CC541 634404642656           WJMP    T,BERR,INTB     RTN             ;WAIT, IF BUSERROR THEN GOTO INTB ELSE RTN
05
06
07                      ;STORE DOUBLE REGISTER, DS, OPCODE=67
08
09 CC542 034402000522 DS:       CAL     F,F,STWD                        ;MEM(ADDR):=W(IR)
10 CC543 014154000001           ZBQ     OR,WPRE                         ;Q:=WPRE(IR)
11 CC544 434400210555           JMP     F,NWADR,DS1                     ;IF 0=<I/O ADDR<8 THEN GOTO DS1
12 CC545 011645004673           SAB     SUNO,C2,WRK1,WRK1               ;WRK1:=WRK1-2
13 CC546 410645001260           SA      SUNO,LLIM,WRK1                  ;IF WRK1<LOWER LIMIT
14 CC547 434400010557           JMP     F,NNEG,DS2                      ;THEN GOTO DS2
15 CC550 034404642656           WJMP    T,BERR,INTB                     ;IF BUSERROR THEN GOTO INTB
16 CC551 414554030001           ZBR     OR,WPRE,DATO                    ;DATAOUT:=WPRE(IR)
17 CC552 414614620260           ZAR     OR,WRK1,WRTP                    ;I/O ADDR:=WRK1, WRITEP
18 CC553 034400602672           JMP     T,NMADR,INTO                    ;IF I/O ADDR<8 THEN GOTOINTO
19 CC554 634404642656           WJMP    T,BERR,INTB     RTN             ;WAIT,
20                                                                      ;IF BUSERROR THEN GOTO INTB ELSE RTN
21
22 CC555 011645224567 DS1:      SABR    SUNO,C2,SB,SB,READP             ;I/O ADDR:=SB, READP
23 CC556 615514000003           ZQB     OR,GRX          RTN             ;W(I/O ADDR):=Q, RTN
24
25 CC557 011645004567 DS2:      SAB     SUNO,C2,SB,SB                   ;SB:=SB-2
26 CC560 434400000533           JMP     F,F,STWD2                       ;GOTO STWD2
```

```
!0029 GPU MICROPROGRAM  GI APR 1982
01                      ;SUBROUTINE GET OPERAND FOR WRITE MODIFICATION
02
03                      ;       FETCHES AN OPERAND FROM THE MEMORY LOCATION ADDRESSED BY SB+BASE
04                      ;       OR FROM W(SB) IF 0=<SB<8.
05                      ;       RETURNS WITH WRK0=FETCHED OPERAND
06
07 CC561 41164022C575 GTW0P:   SABR    ADD,BASE,SB,WRK3,READP          ; I/O ADDR:=WRK3:=SB+BASE, READP
08 CC562 434400600572          JMP     T,NMADR,GTW01                   ; IF I/O ADDR < 8 THEN GOTO GTW01
09 CC563 41064500132C          SA      SUN0,LLIM,WRK3                  ; IF WRK3 < LOWER LIMIT
10 CC564 034400010571          JMP     F,NNEG,GTW02                    ; THEN GOTO GTW02
11 CC565 41065000172C          SA      SUB,ULIM,WRK3                   ; IF WRK3 <= UPPER LIMIT
12 CC566 434400012672          JMP     F,NNEG,INT0                     ; THEN GOTO INT0
13 CC567 034404642656          WJMP    T,BERR,INTB                     ; WAIT, IF BUSERROR THEN GOTO INTB
14 CC570 615754001412          RZB     OR,DATI,WRK0    RTN             ; WRK0:= DATAIN, RTN
15
16 CC571 034404000572 GTW02:   WJMP    F,F,GTW01,                      ; WAIT, GOTO GTW01
17 CC572 01461422016C GTW01:   ZAR     OR,SB,READP                     ; I/O ADDR:= SB, READP
18 CC573 034400210575          JMP     F,NWADR,GTW03                   ; IF 0 =< I/O ADDR < 8 THEN GOTO GTW03
19 CC574 434404002672          WJMP    F,F,INT0                        ; WAIT, GOTO INT0
20
21 CC575 414154000003 GTW03:   ZBQ     OR,GRX                          ; Q:= W(I/O ADDR)
22 CC576 615514000012          ZQB     OR,WRK0 RTN                     ; WRK0:= Q, RTN
```

```
                        ;STORE HALF REGISTER, HS, OPCODE = 32

C0577 434402000561 HS:      CAL     F,F,GTWOP                      ; WRKO:= MEMORY WORD
C0600 41415400000C          ZBG     OR,W                           ; Q:= W
C0601 011720006013          SOB     AND,M12,WRK1                   ; WRK1:= 12EXTO CON Q(12:23)
C0602 434400620613          JMP     T,ODD,HSRGT                    ; IF I/O ADDR(23) = 1 THEN GOTO HSRGT
C0603 015354001273 HSLFT:   SWAP    WRK1,WRK1                      ; WRK1:= WRK1(12:23) CON WRK1(0:11)
C0604 411660006252          SAB     AND,M12,WRKO,WRKO              ; WRKO:= 12EXT 0 CON WRKO(12:23)

C0605 415454030272 HSTOR:   ABBR    OR,WRK1,WRKO,DATO              ; DATAOUT:= WRKO:= WRKO OR WRK1
C0606 434400210612          JMP     F,NWADR,HSREG                  ; IF 0 =< I/O ADDR < 8 THEN GOTO HSREG
C0607 01064062056C          SAR     ADD,BASE,SB,WRTP               ; I/O ADDR:= SB + BASE, WRTP
C0610 034400602672          JMP     T,NMADR,INTO                   ; IF I/O ADDR < 8 THEN GOTO INTO
C0611 634404642656          WJMP    T,BERR,INTB        RTN         ; WAIT, IF BUSERROR THEN GOTO INTB ELSE RTN
C0612 615614000243 HSREG:   ZAB     OR,WRKO,GRX        RTN         ; W(I/O ADDR):= WRKO, RTN

C0613 011664006252 HSRGT:   SAB     CAND,M12,WRKO,WRKO             ; WRKO:= WRKO(0:11) CON 12EXTO
C0614 4344000006C5          JMP     F,F,HSTOR                      ; GOTO HSTOR


                        ;STORE EXCEPTION REGISTER, XS, OPCODE = 33

C0615 434402000561 XS:      CAL     F,F,GTWOP                      ; WRKO:= MEMORY WORD
C0616 400400000007          LDIM    0,7
C0617 015660000113          RAB     AND,IMOP,STAT,WRK1             ; WRK1:= 23 EXT 0 CON EX
C0620 434400620613          JMP     T,ODD,HSRGT                    ; IF I/O ADDR(23) = 1 THEN GOTO HSRGT
C0621 4344000006C3          JMP     F,F,HSLFT                      ; GOTO HSLFT
```

```
!0031 GPU MICROPROGRAM  GI APR 1982
01                    ;EXCHANGE REGISTER AND MEMORY WORD, RX, OPCODE = 31
02
03 C0622 434402000561 RX:      CAL     F,F,GTWOP                       ; WRKO:= MEMORY WORD
04 C0623 41415400000C          ZBQ     OR,W                            ; Q:= W(IR)
05 C0624 41451403000C          ZQR     OR,DATO                         ; DATAOUT:= Q
06 C0625 034400210631          JMP     F,NWADR,RXREG                   ; IF 0 =< I/O ADDR < 8 THEN GOTO RXREG
07 C0626 010640620560          SAR     ADD,BASE,SB,WRTP                ; I/O ADDR:= SB + BASE, WRITEP
08 C0627 015614000240          ZAB     OR,WRKO,W                       ; W(IR):= WRKO
09 C0630 634404642656          WJMP    T,BERR,INTB     RTN             ; IF BUSERROR THEN GOTO INTB ELSE RTN
10 C0631 015514000003 RXREG:   ZQB     OR,GRX                          ; W(I/O ADDR):= Q
11 C0632 615614000240          ZAB     OR,WRKO,W       RTN             ; W(IR):= WRKO, RTN
```

```
!CC32 GPU MICROPROGRAM  GI APR 1982
01                      ;DOUBLE-WORD INSTRUCTIONS WITH MEMORY REFERENCE
02                      ;*********************************************
03
04
05                      ;SUBROUTINE GET DOUBLE-WORD
06
07                      ;       FETCHES A DOUBLE-WORD FROM THE ADDRESSED MEMORY LOCATION
08                      ;       AND THE PRECEDING MEMORY LOCATION.
09                      ;       RETURNS WITH: WRK1 = WORD(ADDRESS)
10                      ;                     WRK0 = WORD(ADDRESS-2)
11
12 CC633 011640220574 GTDWD:  SABR    ADD,BASE,SB,WRK2,READP          ; I/O ADDR:= WRK2:= SB + BASE, READP
13 CC634 034400600652         JMP     T,NMADR,GTDW1                   ; IF I/O ADDR < 8 THEN GOTO GTDW1
14 CC635 010645001300         SA      SUNO,LLIM,WRK2                  ; IF WRK2 < LOWER LIMIT
15 CC636 434400010651         JMP     F,NNEG,GTDW2                    ; THEN GOTO GTDW2
16 CC637 010650001700         SA      SUB,ULIM,WRK2                   ; IF WRK2 >= UPPER LIMIT
17 CC640 434400010651         JMP     F,NNEG,GTDW2                    ; THEN GOTO GTDW2
18 CC641 034404642656         WJMP    T,BERR,INTB                     ; WAIT, IF BUSERROR THEN GOTO INTB
19 CC642 415754001413         RZB     OR,DATI,WRK1                    ; WRK1:= DATAIN
20 CC643 011645224714         SABR    SUNO,C2,WRK2,WRK2,READP         ; I/O ADDR:= WRK2:= WRK2 - 2, READP
21 CC644 011645004567         SAB     SUNO,C2,SB,SB                   ; SB:= SB - 2
22 CC645 010645001300         SA      SUNO,LLIM,WRK2                  ; IF WRK2 < LOWER LIMIT
23 CC646 434400010661         JMP     F,NNEG,GTDW3                    ; THEN GOTO GTDW3
24 CC647 034404642656         WJMP    T,BERR,INTB                     ; WAIT, IF BOSERROR THEN GOTO INTB
25 CC650 615754001412         RZB     OR,DATI,WRK0    RTN             ; WRK0:= DATAIN, RETURN
26
27 CC651 434404000652 GTDW2:  WJMP    F,F,GTDW1                       ; WAIT, GOTO GTDW1
28 CC652 014614220160 GTDW1:  ZAR     OR,SB,READP                     ; I/O ADDR:= SB, READP
29 CC653 034400610665         JMP     T,NWADR,GTDW6                   ; IF 0 > I/O ADDR >= 8 THEN GOTO GTDW6
30 CC654 414154000003         ZBQ     OR,GRX                          ; Q := W(I/O ADDR)
31 CC655 415514000013         ZQB     OR,WRK1                         ; WRK1:= Q
32 CC656 011645224567         SABR    SUNO,C2,SB,SB,READP             ; I/O ADDR:= SB:= SB - 2
33 CC657 414154000003 GTDW5:  ZBQ     OR,GRX                          ; Q:= W(I/O ADDR)
34 CC660 615514000012         ZQB     OR,WRK0 RTN                     ; WRK0:= Q, RETURN
35
36 CC661 434404000662 GTDW3:  WJMP    F,F,GTDW4                       ; WAIT, GOTO GTDW4
37 CC662 014614220160 GTDW4:  ZAR     OR,SB,READP                     ; I/O ADDR:= SB, READP
38 CC663 034400210657         JMP     F,NWADR,GTDW5                   ; IF 0 =< I/O ADDR < 8 THEN GOTO GTDW5
39 CC664 034400002672         JMP     F,F,INTO                        ; GOTO INTO
```

```
10033 GPU MICROPROGRAM  GI APR 1982
01 CC665 034400602672 GTDW6:  JMP     T,NMADR,INTO                    ; IF I/O ADDR < 8 THEN GOTO INTO
02 CC666 41065000016C         SA      SUB,CPA,SB                      ; IF SB >= CPA
03 CC667 434400012672         JMP     F,NNEG,INTO                     ; THEN GOTO INTO
04 CC670 034404642656         WJMP    T,BERR,INTB                     ; WAIT, IF BUSERROR THEN GOTO INTB
05 CC671 415754001413         RZB     OR,DATI,WRK1                    ; WRK1:= DATAIN
06 CC672 011645224567         SABR    SUNO,C2,SB,SB,READP             ; I/O ADDR:= SB := SB - 2
07 CC673 034400602672         JMP     T,NMADR,INTO                    ; IF I/O ADDR < 8 THEN GOTO INTO
08 CC674 034404642656         WJMP    T,BERR,INTB                     ; WAIT, IF BUSERROR THEN GOTO INTB
09 CC675 615754001412         RZB     OR,DATI,WRK0    RTN             ; WRK0:= DATAIN, RETURN
```

```
!0034 GPU MICROPROGRAM  GI APR 1982
01                      ;LOAD DOUBLE REGISTER, DL, OPCODE = 66
02
03 CC676 034402000633 DL:       CAL     F,F,GTDWD                       ; WRK1:= WORD(ADDR), WRK0:= WORD(ADDR-2)
04 CC677 41561400026C           ZAB     OR,WRK1,W                       ; W(IR):= WRK1
05 CC700 434400600702           JMP     T,NMADR,DL1                     ; IF I/O ADDR<8 THEN GOTO DL1
06 CC701 215614000241           ZAB     OR,WRK0,WPRE    RTN             ; WPRE(IR):= WRK0, RETURN
07
08 CC702 414154000003 DL1:      ZBQ     OR,GRX
09 CC703 215514000001           ZQB     OR,WPRE         RTN             ; WPRE(IR):=W(I/O ADDR), RTN
10
11
12                      ;ADD INTEGER DOUBLE WORD, AA, OPCODE = 70
13
14 CC704 034402000633 AA:       CAL     F,F,GTDWD                       ; WRK1:= WORD(ADDR), WRK0:= WORD(ADDR-2)
15 CC705 011664005104           SAB     CAND,M2,STAT,STAT               ; EX(22):= EX(23):= 0
16 CC706 415440000260           ABB     ADD,WRK1,W                      ; W(IR):= W(IR) + WRK1
17 CC707 014052000377           ABQ     SUBC,WRK5,WRK5                  ; Q:=IF CARRY THEN 0 ELSE -1
18 CC710 022154000417           DSRB    OR,WRK5,Z,MC                    ; ADCND:=CARRY
19 CC711 434402600657           CAL     T,NMADR,GTDW5                   ; IF I/O ADDR<8 THEN
20                                                                      ; WRK0:=W(I/O ADDR)
21 CC712 415443000241           ABB     ADDX,WRK0,WPRE                  ; WPRE(IR):= WPRE(IR)+WRK0+ADCND
22 CC713 434400430352           JMP     T,OVFL,INOFL                    ; IF OVERFLOW THEN GOTO INOFL
23 CC714 215542000004           ZBB     ADDC,STAT       RTN             ; EX(23):= CARRY, RTN
24
25
```

```
10035 GPU MICROPROGRAM  GI APR 1982
01                    ;SUBTRACT INTEGER DOUBLE WORD, SS, OPCODE = 71
02
03 00715 034402000633 SS:     CAL     F,F,GTDWD                       ; WRK1:= WORD(ADDR), WRK0:= WORD(ADDR-2)
04 00716 011664005104         SAB     CAND,M2,STAT,STAT               ; EX(22):= EX(23):= 0
05 00717 415445000260         ABB     SUNO,WRK1,W                     ; W(IR):= W(IR) - WRK1
06 00720 014052000377         ABQ     SUBC,WRK5,WRK5                  ; Q:=IF CARRY THEN 0 ELSE -1
07 00721 022154000417         DSRP    OR,WRK5,Z,MC                    ; ADCND:=CARRY
08 00722 434402600657         CAL     T,NMADR,GTDW5                   ; IF I/O ADDR<8 THEN
09                                                                    ; WRK0:=W(I/O ADDR)
10 00723 015447000241         ABB     SUNX,WRK0,WPRE                  ; WPRE(IR):= WPRE(IR)-WRK0-1+ADCND
11 00724 434400430352         JMP     T,OVFL,INOFL                    ; IF OVERFLOW THEN GOTO INOFL
12 00725 215542000004         ZBB     ADDC,STAT       RTN             ; EX(23):= CARRY, RTN
13
```

```
!CC36 GPU MICROPROGRAM  GI APR 1982
01                      ;SHIFT INSTRUCTIONS
02                      ;******************
03
04
05                      ;SUBROUTINE SET SHIFT COUNT
06
07                      ;       WRKC:= IF 0 < WRK0 =< 48 THEN WRK0
08                      ;              IF -48 =< WRK0 < 0 THEN -WRK0 ELSE 48
09
10 CC726 415551000012 SCNTR:  ZBB     SUB0,WRK0                          ; WRK0:= -WRK0
11 CC727 00040000006C SCNTL:  LDIM    0,60                               ; IMOP:= 48.
12 CC730 414644000240         RA      SUN,IMOP,WRK0                      ; IF WRK0 > 48
13 CC731 234400410732         JMP     T,NNEG,SCNT1    RTN                ; THEN GOTO LDCNT ELSE RETURN
14 CC732 615754000012 SCNT1:  RZB     OR,IMOP,WRK0    RTN                ; WRK0:= 48, RETURN
```

```
!0037 GPU MICROPROGRAM  GI APR 1982
01                    ;ARITHMETICALLY SHIFT SINGLE, AS, OPCODE = 44
02
03 CC733 415614000172 AS:      ZAB     OR,SB,WRK0                          ; WRK0:= SB
04 CC734 034400010751          JMP     F,NNEG,ASR                          ; IF SB < 0 THEN GOTO ASR
05 CC735 434400020755          JMP     F,NZ,ASR1                           ; IF SB = 0 THEN GOTO ASR1
06 CC736 434402000727 ASL:     CAL     F,F,SCNTL                           ; WRK0:= IF WRK0 > 48 THEN 48 ELSE WRK0
07 CC737 011664005104          SAB     CAND,M2,STAT,STAT                   ; EX(22):=EX(23):=0
08 CC740 115554000000          ZBB     OR,W     PUSH                       ; W:= W, PUSH
09
10 CC741 034402450750          CAL     T,NORM,ASOF1                        ; IF W(0) <> W(1) THEN EX(22):=1
11 CC742 015544000012          ZBB     SUN,WRK0                            ; WRK0:= WRK0 - 1
12 CC743 263554020000          SLZB    OR,W,Z,NL,F,NZ  LRTN                ; W:= W(1:23) CON 0
13                                                                         ; IF WRK0 = 0 THEN POP ELSE REPEAT
14 CC744 034402450750          CAL     T,NORM,ASOF1                        ; IF W(0)<>W(1) THEN EX(22):=1
15 CC745 410660004500 ASLOF:   SA      AND,C2,STAT                         ; IF EX(22)=1
16 CC746 634400420747          JMP     T,NZ,ASOF2              RTN         ; THEN GOTO ASOF2 ELSE RTN
17 CC747 234400542674 ASOF2:   JMP     T,IMSK,INTA             RTN         ; IF INT MASK=1 THEN GOTO INTA ELSE RTN
18
19 CC750 611654004504 ASOF1:   SAB     OR,C2,STAT,STAT         RTN         ; EX(22):=1, RTN
20
21
22 CC751 034402000726 ASR:     CAL     F,F,SCNTR                           ; WRK0:= IF WRK0 >= -48 THEN -WRK0 ELSE 48
23 CC752 115554000000          ZBB     OR,W     PUSH                       ; W:= W, PUSH
24
25 CC753 015544000012          ZBB     SUN,WRK0                            ; WRK0:= WRK0 - 1
26 CC754 262540024000          SRZB    ADD,W,SGN,NL,F,NZ       LRTN        ; W:= W(0) CON W(0:22)
27                                                                         ; IF WRK0 = 0 THEN POP ELSE REPEAT
28 CC755 611664005104 ASR1:    SAB     CAND,M2,STAT,STAT       RTN         ; EX(22):= EX(23):= 0, RETURN
```

```
!0038 GPU MICROPROGRAM  GI APR 1982
01                    ;ARITHMETICALLY SHIFT DOUBLE, AD, OPCODE = 45
02
03 CC756 011664005104 AD:     SAB     CAND,M2,S AT,STAT                    ; EX(22):= EX(23) := 0
04 CC757 414154000CCC         ZBQ     OR,W                                 ; Q:= W
05 CC760 415614000172         ZAB     OR,SB,WRK0                           ; WRK0 := SB
06 CC761 034400010773         JMP     F,NNEG,ADR                           ; IF SB < 0 THEN GOTO ADR
07 CC762 634400420763         JMP     T,NZ,ADL          RTN                ; IF SB <> 0 THEN GOTO ADL ELSE RTN
08 CC763 434402000727 ADL:    CAL     F,F,SCNTL                            ; WRK0:= IF WRK0 =< 48 THEN WRK0 ELSE 48
09 CC764 515554000001         ZBB     OR,WPRE PUSH                         ; WPRE:= WPRE, PUSH
10
11 CC765 034402450750         CAL     T,NORM,ASOF1                         ; IF WPRE(0) <> WPRE(1) THEN EX(22):=1
12 CC766 015544000012         ZBB     SUN,WRK0                             ; WRK0:= WRK0 - 1
13 CC767 263154020001         DLZB    OR,WPRE,Z,NL,F,NZ         LRTN       ; WPRE CON Q:= WPRE(1:22) CON Q CON 0
14                                                                         ; IF WRK0 = 0 THEN POP ELSE REPEAT
15
16 CC770 034402450750         CAL     T,NORM,ASOF1                         ; IF WPRE(0)<>WPRE(1) THEN EX(22):=1
17 CC771 015514000CCC         ZQB     OR,W                                 ; W:= Q
18 CC772 434400000745         JMP     F,F,ASLOF                            ; GOTO ASLOF
19
20
21 CC773 034402000726 ADR:    CAL     F,F,SCNTR                            ; WRK0:= IF WRK0 <= -48 THEN -WRK0 ELSE 48
22 CC774 515554000001         ZBB     OR,WPRE PUSH                         ; WPRE:= WPRE, PUSH
23
24 CC775 015544000012         ZBB     SUN,WRK0                             ; WRK0:= WRK0 - 1
25 CC776 262140024001         DRZB    ADD,WPRE,SGN,NL,F,NZ      LRTN       ; WPRE CON Q:= WPRE(0) CON WPRE CON Q(0:22)
26                                                                         ; IF WRK0 = 0 THEN POP ELSE REPEAT
27 CC777 615514000CCC         ZQB     OR,W     RTN                         ; W:= Q, RTN
```

```
10039 GPU MICROPROGRAM  GI APR 1982
01                      ;LOGICALLY SHIFT SINGLE, LS, OPCODE = 46
02
03 C1000 415614000172 LS:       ZAB     OR,SB,WRK0                      ; WRK0:= SB
04 C1001 034400011010           JMP     F,NNEG,LSR                      ; IF SB < 0 THEN GOTO LSR
05 C1002 634400421003           JMP     T,NZ,LSL        RTN             ; IF SB <> 0 THEN GOTO LSL ELSE RTN
06 C1003 434402000727 LSL:      CAL     F,F,SCNTL                       ; WRK0:= IF WRK0 =< 48 THEN WRK0 ELSE 48
07 C1004 115554000000           ZBB     OR,W    PUSH                    ; W:= W, PUSH
08
09 C1005 015544000012           ZBB     SUN,WRK0                        ; WRK0:= WRK0 - 1
10 C1006 263554020000           SLZB    OR,W,Z,NL,F,NZ  LRTN            ; W:= W(1:23) CON 0
11                                                                      ; IF WRK0 = 0 THEN POP ELSE REPEAT
12
13 C1007 215554000000           ZBB     OR,W    RTN                     ; W:= W, RTN
14
15 C1010 034402000726 LSR:      CAL     F,F,SCNTR                       ; WRK0:= IF WRK0 >= -48 THEN -WRK0 ELSE 48
16 C1011 115554000000           ZBB     OR,W    PUSH                    ; W:= W, PUSH
17
18 C1012 015544000012           ZBB     SUN,WRK0                        ; WRK0:= WRK0 - 1
19 C1013 662554020000           SRZB    OR,W,Z,NL,F,NZ  LRTN            ; W:= 0 CON W(0:22)
20                                                                      ; IF WRK0 = 0 THEN POP ELSE REPEAT
21 01014 215554000000           ZBB     OR,W    RTN                     ; W:= W, RTN
22                      ;LOGICALLY SHIFT DOUBLE, LD, OPCODE = 47
23
24 C1015 415614000172 LD:       ZAB     OR,SB,WRK0                      ; WRK0:= SB
25 C1016 034400011025           JMP     F,NNEG,LDR                      ; IF SB < 0 THEN GOTO LDR
26 C1017 234400421020           JMP     T,NZ,LDL        RTN             ; IF SB <> 0 THEN GOTO LDL ELSE RTN
27 C1020 434402000727 LDL:      CAL     F,F,SCNTL                       ; WRK0:= IF WRK0 =< 48 THEN WRK0 ELSE 48
28 C1021 114154000000           ZBQ     OR,W    PUSH                    ; Q:= W, PUSH
29
30 C1022 015544000012           ZBB     SUN,WRK0                        ; WRK0:= WRK0 - 1
31 C1023 263154020001           DLZB    OR,WPRE,Z,NL,F,NZ       LRTN    ; WPRE CON Q:= WPRE(1:22) CON Q CON 0
32                                                                      ; IF WRK0 = 0 THEN POP ELSE RTN
33 C1024 615514000000           ZQB     OR,W    RTN                     ; W:= Q, RTN
34
35 C1025 034402000726 LDR:      CAL     F,F,SCNTR                       ; WRK0:= IF WRK0 >= -48 THEN -WRK0 ELSE 42
36 C1026 114154000000           ZBQ     OR,W    PUSH                    ; Q:= W, PUSH
37
38 C1027 015544000012           ZBB     SUN,WRK0                        ; WRK0:= WRK0 - 1
39 C1030 662154020001           DRZB    OR,WPRE,Z,NL,F,NZ       LRTN    ; WRPE CON Q:= 0 CON WPRE(0:23) CON Q(0:22)
40                                                                      ; IF WRK0 = 0 THEN POP ELSE REPEAT
41 C1031 615514000000           ZQB     OR,W    RTN                     ; W:= Q, RTN
```

```
!0C4C GPU MICROPROGRAM  GI APR 1982
01                    ;NORMALIZE SINGLE, NS, OPCODE = 42
02
03 C1032 01455400000C NS:     ZB      OR,W                                ; IF W=0
04 C1033 034400021045         JMP     F,NZ,NS1                            ; THEN GOTO NS1
05 C1034 111750004413         SZB     SUB,C2,WRK1      PUSH               ; WRK1:= 1, PUSH
06
07 C1035 415544000013         ZBB     SUN,WRK1                            ; WRK1:=WRK1-1
08 C1036 01455400000C         ZB      OR,W                                ; NORM:=IF W(0)<>W(1) THEN 1 ELSE 0
09 C1037 26355445000C         SLZB    OR,W,Z,NL,T,NORM         LRTN       ; SH LINK CON W:= W(0:23) CON 0
10                                                                        ; IF SH LINK <> W(0) THEN POP ELSE REPEAT
11 C1040 42255400200C         SSRB    OR,W,LNK,NL                         ; W:= SHLINK CON W(0:22)
12 C1041 411660006273 NS3:    SAB     AND,M12,W,1,WRK1                    ; WRK1:= 12 EXT 0 CON WRK1(12:23)
13 C1042 434402000561 NS2:    CAL     F,F,GTWOP                           ; WRK0:= WORD(ADDR)
14 C1043 434400620613         JMP     T,ODD,HSRGT                         ; WORD(ADDR):= WORD(0:11) CON WRK1(12:23),
15                                                                        ; RTN
16 C1044 434400000603         JMP     F,F,HSLFT                           ; WORD(ADDR):= WRK1(12:23) CON WORD(12:23),
17                                                                        ; RTN
18
19 C1045 411754002013 NS1:    SZB     OR,MESS,WRK1                        ; WRK1:= 12 EXT 0 CON -2048
20 C1046 034400001042         JMP     F,F,NS2                             ; GOTO NS2
21
22
```

```
10041 GPU MICROPROGRAM  GI APR 1982
01                    ;NORMALIZE DOUBLE, ND, OPCODE = 43
02
03 C1047 414554000001 ND:     ZB      OR,WPRE                              ; 0 OR WPRE
04 C1050 034400421053         JMP     T,NZ,ND1                             ; IF WPRE <> 0 THEN GOTO ND1
05 C1051 014554000000         ZB      OR,W
06 C1052 034400021045         JMP     F,NZ,NS1                             ; IF WPRE CON W = 0 THEN GOTO NS1
07 C1053 414154000000 ND1:    ZBQ     OR,W                                 ; Q:=W
08 C1054 111750004413         SZB     SUB,C2,WRK1     PUSH                 ; WRK1:= 1, PUSH
09
10 C1055 415544000013         ZBB     SUN,WRK1                             ; WRK1:=WRK1-1
11 C1056 414554000001         ZB      OR,WPRE                              ; NORM:=IF WPRE(0)<>WPRE(1) THEN 1 ELSE 0
12 C1057 263154450001         DLZB    OR,WPRE,Z,NL,T,NORM     LRTN         ; SHLINK CON WPRE CON Q:= WPRE CON Q CON 0
13                                                                         ; IF SHLINK <> WPRE(0) THEN POP ELSE REPEAT
14 C1060 422154002001         DSRB    OR,WPRE,LNK,NL                       ; WPRE CON Q := SHLINK CON WPRE CON Q(0:22)
15 C1061 015514000000         ZQB     OR,W                                 ; W:= Q
16 C1062 034400001041         JMP     F,F,NS3                              ; GOTO NS3
```

```
!CC42 GPU MICROPROGRAM  GI APR 1982
01                      ;SKIP INSTRUCTIONS
02                      ;******************
03
04
05                      ;SKIP INSTRUCTIONS WHICH SKIPS JUMPS TO THIS CODE
06
07 C1063 011640004525 SKIP:   SAB     ADD,C2,IC,IC                    ; IC:= IC + 2
08 C1064 611640004610         SAB     ADD,C2,PC,PC    RTN             ; PC:= PC + 2
09
10
11                      ;SKIP IF REGISTER HIGH, SH, OPCODE  = 50
12
13 C1065 014451000160 SH:     AB      SUB0,SB,W                       ; SB - W
14 C1066 034400431070         JMP     T,OVFL,SH1                      ; IF OVERFLOW THEN GOTO SH1
15 C1067 234400011063         JMP     F,NNEG,SKIP     RTN             ; IF (SB-W) < 0 THEN GOTO SKIP ELSE RTN
16 C1070 634400411063 SH1:    JMP     T,NNEG,SKIP     RTN             ; IF (SB-W) >= 0 THEN GOTO SKIP ELSE RTN
17
18
19                      ;SKIP IF REGISTER LOW, SL, OPCODE = 51
20
21 C1071 014445000160 SL:     AB      SUN0,SB,W                       ; W - SB
22 C1072 434400431074         JMP     T,OVFL,SL1                      ; IF OVERFLOW THEN GOTO SL1
23 C1073 234400011063         JMP     F,NNEG,SKIP     RTN             ; IF (W-SB) < 0 THEN GOTO SKIP ELSE RTN
24 C1074 634400411063 SL1:    JMP     T,NNEG,SKIP     RTN             ; IF (W-SB) >= 0 THEN GOTO SKIP ELSE RTN
25
26
27                      ;SKIP IF REGISTER EQUAL, SE, OPCODE = 52
28
29 C1075 014451000160 SE:     AB      SUB0,SB,W                       ; IF (SB-W) = 0
30 C1076 234400021063         JMP     F,NZ,SKIP       RTN             ; THEN GOTO SKIP ELSE RTN
```

```
!0043 GPU MICROPROGRAM  GI APR 1982
01                      ;SKIP IF REGISTER NOT EQUAL, SN, OPCODE = 53
02
03 C1077 014451000160 SN:       AB      SUBO,SB,W                       ; IF (SB-W) <> 0
04 C1100 634400421063           JMP     T,NZ,SKIP       RTN             ; THEN GOTO SKIP ELSE RTN
05
06
07                      ;SKIP IF REGISTER BITS ONE, SO, OPCODE = 54
08
09 C1101 014060000160 SO:       ABQ     AND,SB,W                        ; Q:= SB AND W
10 C1102 414411000160           AQ      SUBO,SB                         ; IF SB = SB AND W
11 C1103 234400021063           JMP     F,NZ,SKIP       RTN             ; THEN GOTO SKIP ELSE RTN
12
13
14                      ;SKIP IF REGISTER BITS ZERO, SZ, OPCODE = 55
15
16 C1104 414460000160 SZ:       AB      AND,SB,W                        ; IF SB AND W = 0
17 C1105 234400021063           JMP     F,NZ,SKIP       RTN             ; THEN GOTO SKIP ELSE RTN
18
19
20                      ;SKIP IF NO EXCEPTIONS, SX, OPCODE = 56
21
22 C1106 400400000007 SX:       LDIM    C,7
23 C1107 414260000100           RAQ     AND,IMOP,STAT                   ; Q:= 21 EXT 0 CON EX(21:23)
24 C1110 014420000160           AQ      AND,SB                          ; IF (21 EXT 0 CON EX) = 0
25 C1111 234400021063           JMP     F,NZ,SKIP       RTN             ; THEN GOTO SKIP ELSE RTN
```

```
!0044 GPU MICROPROGRAM  GI APR 1982
01                      ;SKIP IF NO WRITE PROTECTION, SP, OPCODE = 25
02
03 C1112 414554000007 SP:      ZB       OR,SB                              ; IF SB < 0
04 C1113 234400411114          JMP      T,NNEG,SKPP1     RTN               ; THEN RTN
05                      SKPP1:
06 C1114 411640000573          SAB      ADD,BASE,SB,WRK1                   ; WRK1:= SB + BASE
07 C1115 410645001260          SA       SUNO,LLIM,WRK1                     ; IF (SB+BASE) < LOWER LIMIT
08 C1116 034400011121          JMP      F,NNEG,SKPP2                       ; THEN GOTO SKPP2
09 C1117 010645001660          SA       SUNO,ULIM,WRK1                     ; IF (SB + BASE) ≤ UPPER LIMIT
10 C1120 234400011063          JMP      F,NNEG,SKIP      RTN               ; THEN GOTO SKIP ELSE RTN
11
12 C1121 410645005560 SKPP2:   SA       SUNO,M8,SB                         ; IF SB < 8.
13 C1122 234400011063          JMP      F,NNEG,SKIP      RTN               ; THEN GOTO SKIP ELSE RTN
```

```
10045 GPU MICROPROGRAM  GI APR 1982
01                      ;JUMP INSTRUCTIONS
02                      ;******************
03
04
05                      ;JUMP WITH REGISTER LINK, JL, OPCODE = 15
06
07 C1123 034402000321 JL:     CAL     F,F,GETOP                       ; WRK0:= NEXT INSTRUCTION
08                                                                    ; WRK1:= PHYSICAL ADDRESS
09                                                                    ; SB:= LOGICAL ADDRESS
10 C1124 034400171126 JL1:    JMP     F,LINK,JL2                      ; IF W-FIELD=0 THEN GOTO JL2
11 C1125 015614000120         ZAB     OR,IC,W                         ; W(IR):=IC
12 C1126 400400000001 JL2:    LDIM    0,1
13 C1127 015664000165         RAB     CAND,IMOP,SB,IC                 ; IC:= SB(0:22) CON 0
14 C1130 415664000270         RAB     CAND,IMOP,WRK1,PC               ; PC:= WRK1(0:22) CON 0
15 C1131 010645002600     SA    SUBO,CLOW,PC      ; ALU := PC - CODE_LOW
16 C1132 034400012670     JMP   F,NNEG,INT2       ; IF NEG THEN GOTO INT2
17 C1133 010651003200     SA    SUBO,CTOP,PC      ; ALU := CODE_TOP - PC
18 C1134 034400012670     JMP   F,NNEG,INT2       ; IF NEG THEN GOTO INT2
19 C1135 034400700254         JMP     T,INTR,INTT                     ; IF INTERRUPT THEN GOTO INTT
20 C1136 014554060012         ZBR     OR,WRK0,IR                      ; IR:=WRK0(0:11)
21 C1137 034400000250         JMP     F,F,AFTCH                       ; GOTO AFTCH (MAIN LOOP)
22                      ;JUMP WITH INTERRUPT DISABLED, JD, OPCODE = 16
23
24
25                      ;JUMP WITH INTERRUPT ENABLED, JE, OPCODE = 17
26
27
28
29                      ;GET AND PUT GENERAL REGISTER INSTRUCTIONS
30                      ;*****************************************
31
32                      ;GET GENERAL REGISTER, GG, OPCODE = 34
33
34                      ;PUT GENERAL REGISTER, GP, OPCODE = 57, PRIVILEGED INSTRUCTION
35
36
37
38                      ;INPUT-OUTPUT INSTRUCTIONS
39                      ;*************************
40
41                      ;DATA IN, DI, OPCODE = 35, PRIVILEGED INSTRUCTION
42
43                      ;DATA OUT, DC, OPCODE = 1, PRIVILEGED INSTRUCTION
```

```
!0046 GPU MICROPROGRAM  GI APR 1982
01
02
03                      ;FLOATING POINT INSTRUCTIONS
04                      ;***************************
05
06                      ;CONVERT INTEGER TO FLOATING, CI, OPCODE = 40
07                                                                          ; CI:
08 C1140 41415400000C CI:       ZBQ     OR,W                                ;   Q:=W
09                                                                          ;   WRK0:=Q
10 C1141 015514000012           ZQB     OR,WRK0                             ;   WRK1:=0
11 C1142 015451000273           ABB     SUB0,WRK1,WRK1                      ;
12 C1143 000400000027           LDIM    0,27                                ;   WRK2:=SB+23
13 C1144 015640000174           RAB     ADD,IMOP,SB,WRK2                    ; NORMALIZE:
14                      NORMA:                                              ;   WRK0,WPK1=FRACTION
15                                                                          ;   WRK2=EXPONENT
16                                                                          ;   EX(22:23):=0
17 C1145 011664005104           SAB     CAND,M2,STAT,STAT                   ;   Q:=WRK1
18 C1146 41421400026C           ZAQ     OR,WRK1                             ;   IF WRK0=0 AND Q=0 THEN
19 01147 014554000012           ZB      OR,WRK0                             ;   BEGIN
20 C1150 434400421155           JMP     T,NZ,CINZ                           ;     W:=12EXT0-2048
21 C1151 41451400000C           ZQ      OR                                  ;     WPRE:=0
22 C1152 434400421155           JMP     T,NZ,CINZ                           ;     RETURN
23 C1153 415560000001 CIZER:    ZBB     AND,WPRE                            ;
24 C1154 611754002000           SZB     OR,MESS,W           RTN             ;
25 C1155 114614000240 CINZ:     ZA      OR,WRK0             PUSH            ;   WHILE SHLINK <> WRK0(0) DO
26 C1156 015544000014           ZBB     SUN,WRK2                            ;   BEGIN
27 C1157 021554000012           NSZB    OR,WRK0,NL,F,F                      ;
28 C1160 663154450012           DLZB    OR,WRK0,Z,NL,T,NORM LRTN            ;     SH LINK CON WRK0 CON Q :=
29 C1161 022154002012           DSRB    OR,WRK0,LNK,NL                      ;     WRK0 CON Q CON 1EXT0
30                                                                          ;     WRK2:=WRK2-1
31                                                                          ;   END
32                                                                          ;   WRK0 CON Q:= SHLINK CON WRK0 CON Q(0:2
33 C1162 410300002000           SQQ     ADD,MESS                            ;   Q:=Q + 8'4000
```

```
!0047 GPU MICROPROGRAM  GI APR 1982
01                    ROUND:                                                    ; ROUND:
02                                                                              ;   WRK0,Q=FRACTION
03                                                                              ;   WRK2=EXPONENT
04 C1163 122142004012          DSRB    ADDC,WRK0,SGN,NL          PUSH         ;   WRK0:=WRK0+CARRY; WRK0 CON Q:=
05                                                                              ;   1 EXT WRK0(0) CON WRK0 CON Q(0:22);
06 C1164 015544000014          ZBB     SUN,WRK2                                ;   WHILE SHLINK<>WRK0(0) DO
07 C1165 021554000012          NSZB    OR,WRK0,NL,F,F                          ;   BEGIN
08                                                                              ;     WRK2:=WRK2-1;
09                                                                              ;     SHLINK CON WRK0 COM Q :=
10 C1166 663154450012          DLZB    OR,WRK0,Z,NL,T,NORM         LRTN        ;     WRK0 CON Q CON 1EXT0
11                                                                              ;   END
12
```

```
!0048 GPU MICROPROGRAM  GI APR 1982
01 C1167 022154002012          DSRB    OR,WRK0,LNK,NL                    ;   WRK0:= SH LINK CON WRK0 CON Q(0:22)
02 C1170 415614000241          ZAB     OR,WRK0,WPRE                      ;   WPRE:=WRK0
03 C1171 400400004003          LDIM    0,4003                            ;   WRK2:=WRK2+8'4003
04 C1172 415640000314          RAB     ADD,IMOP,WRK2,WRK2                ;
05 C1173 411670002314          SAB     EXOR,MESS,WRK2,WRK2               ;   WRK2:=WRK2 EXOR 8'4000
06 C1174 411660006312          SAB     AND,M12,WRK2,WRKC                 ;   WRK0:=12EXT0 CON WRK2(12:23)
07 C1175 010324006000          SQQ     CAND,M12                          ;   Q:=Q(0:11) CON 12 EXT 0;
08 C1176 415400000240          AQB     ADD,WRK0,W                        ;   W:=WRK0+Q;
09 C1177 410664006300          SA      CAND,M12,WRK2                     ;   IF WRK2(0:11) CON 12EXT0 <> THEN
10 C1200 634400421201          JMP     T,NZ,CI1                  RTN     ;   BEGIN
11 C1201 034400411204 CI1:     JMP     T,NNEG,CIOFL                      ;     IF WRK2>0 THEN GOTO CIOFL
12                    ; UNDERFLOW:
13 C1202 415560000001          ZBB  AND,WPRE          ; WPRE := 0
14 C1203 611754002000          SZB  OR,MESS,W    RTN  ; W := 0,-2048 RETURN
15 C1204 415541000004 CIOFL:   ZBB     ADD0,STAT                         ;
16 C1205 061601150104          NSZA    ADD0,STAT,STAT,NL,F,FPMSK  CRTN   ;     IF -,FLOATING POINT MASK THEN RETURN
17                                                                       ;   END ELSE RETURN;
18                    INTF:                                              ; FLOATINGPOINT EXCEPTION:
19 C1206 400400400000          LDIM C04C,0000         ; FLOATING POINT UNDERFLOW
20 C1207 415754000016 INTFA:   RZB  OR,IMOP,WRK4      ; WORK4 := STATUS
21 C1210 410660004500          SA   AND,C2,STAT       ; IF OVERFLOW
22 C1211 434400020133          JMP  F,NZ,SEND2        ; THEN
23 C1212 015440000356          ABB  ADD,WRK4,WRK4     ; WORK4 := WORK4 SHIFT 1;
24 C1213 034400000133          JMP  F,F,SEND2         ; GOTO SEND2;
25
26
27                    ;CONVERT FLOATING TO INTEGER, CF , OPCODE = 65
28 C1214 011664005104 CF:      SAB     CAND,M2,STAT,STAT                 ; CF:
29 C1215 414154000000          ZBQ     OR,W                              ;   EX(22:23):=0
30                                                                       ;   Q:=W
31 C1216 415514000013          ZQB     OR,WRK1                           ;   WRK1:=Q
32 C1217 015354000673          EXT     WRK1,WRK1                         ;   WRK1:= 12 EXT WRK1(12) CON WRK1(12:23)
33 C1220 015440000173          ABB     ADD,SB,WRK1                       ;   WRK1:=SB+WRK1
34 C1221 034400411223          JMP     T,NNEG,CFNZR                      ;   IF WRK1<0 THEN
35 C1222 615560000000 CFZER:   ZBB     AND,W                     RTN     ;   BEGIN
```

```
10049 GPU MICROPROGRAM  GI APR 1982
01 01223 000400000027 CFNZR:  LDIM    0,27                                ;     W:=0
02                                                                        ;     RETURN
03                                                                        ;   END
04 01224 415645000273         RAB     SUNO,IMOP,WRK1,WRK1                 ;   WRK1:=WRK1-23
05 01225 434400011232         JMP     F,NNEG,CFSHI                        ;   IF WRK1<=0 THEN
06 01226 434400021232         JMP     F,NZ,CFSHI                          ;
07 01227 414554000001         ZB      OR,WPRE                             ;     GOTO CFSHI
08                                                                        ;   IF WPRE=0 THEN
09 01230 034400021222         JMP     F,NZ,CFZER                          ;     GOTO CFZER
10 01231 034400001246         JMP     F,F,CFOVF                           ;   ELSE
11                                                                        ;     GOTO CFOVF
12 01232 011724006012 CFSHI:  SQB     CAND,M12,WRK0                       ;   WRK0:=Q(0:11) CON 12EXT0
13 01233 014154000001         ZBQ     OR,WPRE                             ;   Q:=WPRE
14 01234 015514000000         ZQB     OR,W                                ;   W:=Q
15 01235 414154000012         ZBQ     OR,WRK0                             ;   Q:=WRK0
16 01236 414554000013         ZB      OR,WRK1                             ;   IF WRK1=0 THEN
17 01237 534400021242         JMP     F,NZ,CF0                  PUSH      ;     GOTO CF0
18 01240 415541000013         ZBB     ADDO,WRK1                           ;   FOR WRK1:=WRK1 STEP 1 UNTIL 0 DO
19 01241 662140024000         DRZB    ADD,W,SGN,NL,F,NZ         LRTN      ;     W CON Q:= W(0) CON W CON Q(0:22)
20 01242 414500000000 CF0:    ZQ      ADD                                 ;   IF Q<0 THEN
21 01243 634400011244         JMP     F,NNEG,CF1                RTN       ;     W:=W+1    C. ROUND
22 01244 015541000000 CF1:    ZBB     ADDO,W                              ;   IF OVERFLOW AND INTEGER MASK THEN
23 01245 234400431246         JMP     T,OVFL,CFOVF              RTN       ;     GOTO INTA
24 01246 011640004504 CFOVF:  SAB     ADD,C2,STAT,STAT                    ;
25 01247 234400542674         JMP     T,IMSK,INTA               RTN       ;   ELSE
26                                                                        ;     RETURN
27
28
```

```
!0050 GPU MICROPROGRAM  GI APR 1982
01                    ;MULTIPLY FLOATING, FM, OPCODE =62
02
03 01250 034402000633 FM:     CAL     F,F,GTDWD                          ; FM:
04                                                                       ;   GET DOUBLEWORD(WRK0,WRK1)
05 01251 415354000674         EXT     WRK1,WRK2                          ;   WRK2:=12EXT WRK1(12) CON WRK1(12:23)
06 01252 011664006273         SAB     CAND,M12,WRK1,WRK1                 ;   WRK1:=WRK1(0:11) CON 12 EXT 0;
07 01253 41415400000C         ZBQ     OR,W                               ;   Q:=W;
08 01254 015514000017         ZQB     OR,WRK5                            ;   WRK5:=Q;
09 01255 415354000776         EXT     WRK5,WRK4                          ;   WRK4:=12 EXT WRK5(12) CON WRK5(12:23);
10 01256 015441000354         ABB     ADD0,WRK4,WRK2                     ;   WRK2:=WRK2+WRK4+1;
11 01257 022154000001         DSRB    OR,WPRE,Z,NL                       ;   WPRE CON Q:= 1 EXT 0 CON WPRE CON Q(0:2
12 01260 415514000016         ZQB     OR,WRK4                            ;   WRK4:=Q;
13 01261 015354001377         SWAP    WRK5,WRK5                          ;   WRK5:=WRK5(12:23) CON WRK5(0:11);
14 01262 414154000017         ZBQ     OR,WRK5                            ;   Q:=WRK5;
15 01263 422045000735         DSRA    SUN0,WRK3,WRK3,Z,MC                ;   ADDC:=Q(23); WRK3:=0;
16 01264 01415400000C         ZBQ     OR,WPRE                            ;   Q:=WPRE;
17 01265 015514000017         ZQB     OR,WRK5                            ;   WRK5:=Q;
18 01266 015354001356         SWAP    WRK4,WRK4                          ;   WRK4:=WRK4(12:23) CON WRK4(0:11);
19 01267 01026000634C         SAQ     AND,M12,WRK4                       ;   Q:=12 EXT 0 CON WRK4(12:23);
20 01270 400400000013         LDIM    0,13                               ;   COUNT:=12;
21 01271 415754000016         RZB     OR,IMOP,WRK4                       ;MULTIPLY:
22 01272 52400000026C         MULQ    ADD,WRK1,NL,F,F          PUSH      ;   REPEAT
23 01273 426042004655         MULD    ADDC,WRK0,WRK3,SGN,MC,F,F          ;     IF ADDC THEN
24 01274 415544000016         ZBB     SUN,WRK4                           ;     BEGIN
25 01275 66400002026C         MULQ    ADD,WRK1,NL,F,NZ         LRTN      ;       Q:=Q+WRK1;
26 01276 426042004655         MULD    ADDC,WRK0,WRK3,SGN,MC,F,F          ;       WRK3:=WRK3+WRK0+C;
27 01277 01032400600C         SQQ     CAND,M12                           ;     END;
28 01300 011660006376         SAB     AND,M12,WRK5,WRK4                  ;     ADDC:=Q(23);
29 01301 014000000340         AQQ     ADD,WRK4                           ;     WRK3 CON Q:=1 EXT WRK3(0) CON
30                                                                       ;     WRK3 CON Q(0:22);
31 01302 415754000016         RZB     OR,IMOP,WRK4                       ;     COUNT:=COUNT-1;
32 01303 52400000026C         MULQ    ADD,WRK1,NL,F,F          PUSH      ;   UNTIL COUNT=0;
33 01304 426042004655         MULD    ADDC,WRK0,WRK3,SGN,MC,F,F          ;END MULTIPLY;
34 01305 415544000016         ZBB     SUN,WRK4                           ;   Q:=Q(0:11) CON 12 EXT 0;
35 01306 66400002026C         MULQ    ADD,WRK1,NL,F,NZ         LRTN      ;   WRK4:=12 EXT 0 CON WRK5(12:23);
```

```
!0051 GPU MICROPROGRAM  GI APR 1982
01
02 C1307 426042004655          MULD    ADDC,WRK0,WRK3,SGN,MC,F,F          ;   Q:=Q+WRK4;
03 C1310 01032400600C          SQQ     CAND,M12                           ;   COUNT:=12;
04 C1311 015354001377          SWAP    WRK5,WRK5                          ;   DO MULTIPLY
05 C1312 411660006377          SAB     AND,M12,WRK5,WRK5                  ;   Q:=Q(0:11) CON 12 EXT 0;
06 C1313 414000000360          AQQ     ADD,WRK5                           ;   WRK5:=WRK5(12:23) CON WRK5(0:11);
07 C1314 015750000016          RZB     SUB,IMOP,WRK4                      ;   WRK5:=12 EXT C CON WRK5(12:23);
08 C1315 524000000260          MULQ    ADD,WRK1,NL,F,F          PUSH      ;   Q:=Q+WRK5;
09 C1316 426042004655          MULD    ADDC,WRK0,WRK3,SGN,MC,F,F          ;   COUNT:=11;
10 C1317 415544000016          ZBB     SUN,WRK4                           ;   DO MULTIPLY
11 C1320 664000002260          MULQ    ADD,WRK1,NL,F,NZ         LRTN      ;   IF ADDC THEN
12 C1321 426042004655          MULD    ADDC,WRK0,WRK3,SGN,MC,F,F          ;   BEGIN
13 C1322 024005000260          MULQ    SUN0,WRK1,NL,F,F                   ;     Q:=Q-WRK1;
14 C1323 426046004255          MULD    SUNC,WRK0,WRK3,SGN,NL,F,F          ;     WRK3:=WRK3-WRK0-1+C;
15 C1324 415514000013          ZQB     OR,WRK1                            ;   END;
16 C1325 415614000332          ZAB     OR,WRK3,WRK0                       ;   WRK1:=Q;
17 C1326 034400001145          JMP     F,F,NORMA                          ;   WRK0:=WRK3;
                                                                          ;   GOTO NORMALIZE;
18
```

```
!0052 GPU MICROPROGRAM  GI APR 1982
01                      ; FLOATING ADD, FA, OPCODE = 60
02                      FA:                                                 ;FA:
03 C1327 41035400050CC          SZQ     OR,M2                               ;   Q:=3 C. ADD
04 C1330 434400001332           JMP     F,F,FSALI                           ;   GOTO FSALI
05
06                      ; FLOATING SUBTRACT, FS, OPCODE = 61
07 C1331 41035400440C FS:       SZQ     OR,C2                               ;   Q:=2 C. SUB
08
09 C1332 022154000417 FSALI:    DSRB    OR,WRK5,Z,MC                        ; FSALI:
10 C1333 034402000633           CAL     F,F,GTDWD                           ;   ADDCONDITION:=Q(23)
11 C1334 415354000674 FABF:     EXT     WRK1,WRK2                           ;   GET DOUBLEWORD(WRK0,WRK1)
12                                                                          ;   WRK2:=12EXT WRK1(12) CON WRK1(12:23)
13 C1335 411664006277           SAB     CAND,M12,WRK1,WRK5                  ;   WRK5:=WRK1(0:11) CON 12EXT0
14 C1336 41415400000C           ZBQ     OR,W                                ;   Q:=W
15 C1337 415514000013           ZQB     OR,WRK1                             ;   WRK1:=Q
16 C1340 015354000675           EXT     WRK1,WRK3                           ;   WRK3:=12EXT WRK1(12) CON WRK1(12:23)
17 C1341 411724006013           SQB     CAND,M12,WRK1                       ;   WRK1:=Q(0:11) CON 12EXT0
18 C1342 415614000256           ZAB     OR,WRK0,WRK4                        ;   WRK4:=WRK0
19 C1343 014154000001           ZBQ     OR,WPRE                             ;   Q:=WPRE
20 C1344 015514000012           ZQB     OR,WRK0                             ;   WRK0:=Q
21 01345 415445000315           ABB     SUN0,WRK2,WRK3                      ;   WRK3:=WRK3-WRK2
22 C1346 034400011361           JMP     F,NNEG,FSASM                        ;   IF WRK3>0 THEN
23 C1347 03440002137C           JMP     F,NZ,FSAEQ                          ;   BEGIN
24 C1350 415440000334           ABB     ADD,WRK3,WRK2                       ;     WRK2:=WRK3+WRK2
25 C1351 400400000046           LDIM    0,46                                ;     IF WRK3>= 38 THEN
26 C1352 414645000320           RA      SUN0,IMOP,WRK3                      ;       RETURN
27 C1353 634400011354           JMP     F,NNEG,FSASH            RTN         ;     ELSE
28 C1354 514214000360 FSASH:    ZAQ     OR,WRK5                 PUSH        ;     BEGIN
29 C1355 415544000015           ZBB     SUN,WRK3                            ;       Q:=WRK5
30 C1356 262140024016           DRZB    ADD,WRK4,SGN,NL,F,NZ    LRTN        ;       FOR WRK3:=WRK3 STEP -1 UNTIL 0 DO
```

```
10053 GPU MICROPROGRAM  GI APR 1982
01                                                                                ;          WRK0 CON Q :=
02                                                                                ;          WRK4(0) CON WRK4 CON Q(0:22)
03 C1357 015514000017          ZQB      OR,WRK5                                   ;        WRK5:=Q
04 C1360 434400001370          JMP      F,F,FSAEQ                                 ;      END
05                                                                                ;    END ELSE
06                                                                                ;    IF WRK3<0 THEN
07                                                                                ;    BEGIN
08 C1361 400400000045 FSASM:   LDIM     0,45                                      ;      IF WRK3<=-38 THEN
09 C1362 414640000320          RA       ADD,IMOP,WRK3                             ;
10 C1363 034400011375          JMP      F,NNEG,FSASL                              ;        GOTO FSASL
11 C1364 514154000013          ZBQ      OR,WRK1                          PUSH     ;      Q:=WRK1
12 C1365 415541000015          ZBB      ADDO,WRK3                                 ;      FOR WRK3:=WRK3 STEP 1 UNTIL 0 DO
13 C1366 662140024012          DRZB     ADD,WRK0,SGN,NL,F,NZ             LRTN     ;        WRK0 CON Q:=
14                                                                                ;        WRK0(0) CON WRK0 CON Q(0:22)
15 C1367 415514000013          ZQB      OR,WRK1                                   ;      WRK1:=Q
16                                                                                ;    END
17                     FSAEQ:                                                     ; FSAEQ:
18 C1370 431443000373          DIVN     ADDX,WRK5,WRK1,NL,F,F                     ;    IF ADDCONDITION=1 THEN
19 C1371 432442004352          DIVR     ADDC,WRK4,WRK0,SGN,NL,F,F                 ;    BEGIN
20 C1372 022540002013          SSRB     ADD,WRK1,LNK,NL                           ;      WRK1:=WRK1-WRK5
21 C1373 015541000014          ZBB      ADDO,WRK2                                 ;      WRK0:=WRK0-WRK4-1+CARRY
22                                                                                ;    END ELSE
23                                                                                ;    BEGIN
24                                                                                ;      WRK1:=WRK1+WRK5
25                                                                                ;      WRK0:=WRK0+WRK4+CARRY
26 C1374 034400001145          JMP      F,F,NORMA                                 ;    END
27                                                                                ;    GOTO NORMALIZE
28 C1375 415560000013 FSASL:   ZBB      AND,WRK1                                  ; FSASL:
29 C1376 015560000012          ZBB      AND,WRK0                                  ;    WRK1:=0
30 C1377 434400001370          JMP      F,F,FSAEQ                                 ;    WRK0:=0
31                                                                                ;    GOTO FSAEQ
32
```

```
!CC54 GPU MICROPROGRAM  GI APR 1982
01                      ; FLOATING DIVIDE, FD, OPCODE = 64
02
03                      FD:                                                     ; FD:
04 C1400 034402000633           CAL     F,F,GTDWD                               ;   GET DOUBLEWORD(WRK0,WRK1);
05 C1401 011664005104           SAB     CAND,M2,STAT,STAT                       ;   EX(22:23):=0;
06 C1402 415354000674           EXT     WRK1,WRK2                               ;   WRK2:=12EXT WRK1(12) CON WRK1(12:23)
07 C1403 411664006277 FDBF:     SAB     CAND,M12,WRK1,WRK5                      ;   WRK5:=WRK1)0:11) CON 12EXT0
08 C1404 415614000256           ZAB     OR,WRK0,WRK4                            ;   WRK4:=WRK0
09 C1405 034400421410           JMP     T,NZ,FD0                                ;   IF WRK4=0 THEN
10 C1406 011640004504           SAB     ADD,C2,STAT,STAT                        ;     EX(22:23):=2
11 C1407 234400551206           JMP     T,FPMSK,INTF                    RTN     ;     IF FLOATING POINT MASK THEN GOTO INTF
12                                                                              ;     ELSE RETURN
13 C1410 414554000001 FD0:      ZB      OR,WPRE                                 ;   IF WPRE=0 THEN
14 C1411 034400021153           JMP     F,NZ,CIZER                              ;     GOTO CIZER
15 C1412 414154000000           ZBQ     OR,W                                    ;   Q:=W
16 C1413 011724006000           SQB     CAND,M12,W                              ;   W:=Q(0:11) CON 12EXT0
17 C1414 415514000013           ZQB     OR,WRK1                                 ;   WRK1:=Q
18 C1415 015354000673           EXT     WRK1,WRK1                               ;   WRK1:=12 EXT WRK1(12) CON WRK1(12:23)
19 C1416 415451000274           ABB     SUB0,WRK1,WRK2                          ;   WRK2:=WRK1-WRK2
20 C1417 421554001416           NSZB    OR,WRK4,DVS,F,F                         ;   DIVSIGN:=WRK4(0)
21 C1420 021554001001           NSZB    OR,WPRE,DC,F,F                          ;   ADC:=WPRE(0) EXOR .-DIVSIGN
22 C1421 414120000000           ZQQ     AND                                     ;   Q:=0
23 C1422 000400000027           LDIM    0,27                                    ;   FOR WRK3:=24 STEP -1 UNTIL 0 DO
24 C1423 415740000015           RZB     ADD,IMOP,WRK3                           ;   BEGIN
25 C1424 133043004360           DIVD    ADDX,WRK5,W,ADC,NL,F,F          PUSH    ;     IF ADC=1 THEN
26 C1425 033442003341           DIVS    ADDC,WRK4,WPRE,LNK,DC,F,F               ;     BEGIN
```

```
!0055 GPU MICROPROGRAM  GI APR 1982
01 C1426 415544000015          ZBB     SUN,WRK3                              ;      W:=W-WRK5
02 C1427 27304302436C          DIVD    ADDX,WRK5,W,ADC,NL,F,NZ    LRTN       ;        WPRE:=WPRE-WRK4-1+CARRY
03                                                                           ;      END ELSE
04                                                                           ;      BEGIN
05                                                                           ;        W:=W+WRK5
06                                                                           ;        WPRE:=WPRE+WRK4+CARRY
07                                                                           ;      END
08                                                                           ;      SHLINK CON WPRE CON W CON Q:=
09                                                                           ;      WPRE CON W CON Q CON 1EXTADC
10 C1430 033442003341          DIVS    ADDC,WRK4,WPRE,LNK,DC,F,F             ;      ADC:=WPRE(0) EXOR -.DIVSIGN
11 C1431 40044000000C          LDIM    4000,0                                ;    END
12 C1432 015514000012          ZQB     OR,WRK0                               ;    WRK0:=Q
13 C1433 415670000252          RAB     EXOR,IMOP,WRK0,WRK0                   ;    WRK0:=WRK0 EXOR 4000,0
14 C1434 034400451441          JMP     T,NORM,FD1                            ;    IF WRK0 NOT NORMALISED THEN
15 C1435 023554004012          SSLB    OR,WRK0,ADC,NL                        ;    BEGIN
16 C1436 43344300036C          DIVS    ADDX,WRK5,W,Z,NL,F,F                  ;    SH LINK CON WRK0:=WRK0 CON ADC
17 C1437 033442003341          DIVS    ADDC,WRK4,WPRE,LNK,DC,F,F             ;      'DIVIDE'
18 C1440 015544000014          ZBB     SUN,WRK2                              ;      WRK2:=WRK2-1
19                       FD1:                                                ;    END
20 C1441 4103540060CC          SZQ     OR,M12                                ;    Q:=8'7777
21 C1442 400400000013          LDIM    0,13                                  ;    FOR WRK3:=12 STEP -1 UNTIL C DO
22 C1443 415740000015          RZB     ADD,IMOP,WRK3                         ;      'DIVIDE'
23 C1444 13304300436C          DIVD    ADDX,WRK5,W,ADC,NL,F,F     PUSH       ;
24 C1445 033442003341          DIVS    ADDC,WRK4,WPRE,LNK,DC,F,F             ;
25 C1446 415544000015          ZBB     SUN,WRK3                              ;
26 C1447 27304302436C          DIVD    ADDX,WRK5,W,ADC,NL,F,NZ    LRTN       ;
27 C1450 033442003341          DIVS    ADDC,WRK4,WPRE,LNK,DC,F,F             ;
28 C1451 415514000013          ZQB     OR,WRK1                               ;    WRK1:=Q
29 C1452 015354001273          SWAP    WRK1,WRK1                             ;    WRK1:=WRK1(0:11) CON WRK1(12:23)
30 C1453 414203000260          ZAQ     ADDX,WRK1                             ;    Q:=WRK1+ADC
31 C1454 434400001163          JMP     F,F,ROUND                             ;    GOTO ROUND
32
```

```
!0056 GPU MICROPROGRAM  GI APR 1982
01
02
03
04                      ; IADD
05
06                      ;
07                      ; IADD       W(=ADDR_HEAD)   SB(=DISP)  OPPOSITE SIGN AS MODUS
08                      ;            SB <> 0 DOUBLE FLOAT ADD  DWD(W) CON DWD(W+SB)
09                      ;            SB =  0 SINGLE FLOAT ADD  DWD(W)
10                      ;
11
12 C1455 011754006015 DADDI: SZB    OR,M12,WRK3          ; WRK3 := M12
13 C1456 43440000146C        JMP    F,F,ADD1             ; GOTO ADD1
```

```
10057 GPU MICROPROGRAM  GI APR 1982
01
02                      ; ADD
03
04                      ; ADD       W(=ADDR_HEAD)   SB(=DISP)  SAME SIGN AS MODUS
05                      ;           SB <> 0 DOUBLE FLOAT ADD  DWD(W) CON DWD(W+SB)
06                      ;           SB =  0 SINGLE FLOAT ADD  DWD(W)
07                      ;
08
09 C1457 415560000015 DADD:  ZBB   AND,WRK3            ; WRK3 := 0
10 C1460 01461400016C ADD1:  ZA    OR,SB               ; ALU := DISP
11 C1461 434400421465        JMP   T,NZ,ADD3           ; IF DISP <> 0 THEN GOTO ADD3
12 C1462 015560000012        ZBB   AND,WRK0            ; WORK0 := 0
13 C1463 415560000013        ZBB   AND,WRK1            ; WORK1 := 0
14 C1464 434400001471        JMP   F,F,ADD4            ; GOTO ADD4
15 C1465 41415400000C ADD3:  ZBQ   OR,W                ; Q := W
16 C1466 415400000167        AQB   ADD,SB,SB           ; SB := SB + Q
17 C1467 034402000633        CAL   F,F,GTDWD           ; WRK0 CON WRK1 := W(SB-2) CON W(SB)
18
19                      ; START FPU LONG LOAD
20 C1470 011664006273        SAB   CAND,M12,WRK1,WRK1  ; WRK1 := FRAC(24:35) CON 0 EXT 12
21 C1471 014454150275 ADD4:  ABR   OR,WRK1,WRK3,FPULL  ; FPUOP1 := WRK1 OR SIGNSHIFT;
22                                                     ; UNITFNC := LL AND ADD
23 C1472 014554140012        ZBR   OR,WRK0,FPOP0       ; FPUOP0 := WRK0
24                      ; DPU READY AFTER 0.80 MYS = 4 GPU-MICROS
25
26                      ; GET HEAD
27 C1473 41415400000C        ZBQ   OR,W                ; Q := W
28 C1474 41551400000C7       ZQB   OR,SB               ; SB := Q <* HEAD_ADDR *>
29 C1475 034402000633        CAL   F,F,GTDWD           ; WRK0 CON WRK1 := W(SB-2) CON W(SB)
30
31                      ; LONG LOAD MUST NOW BE FINISHED!!!
32                      ; LOAD HEAD
33 C1476 014554150013 ADD2:  ZBR   OR,WRK1,FPOP1       ; FPUOP1 := WRK1
34 C1477 014554140012        ZBR   OR,WRK0,FPOP0       ; FPUOP0 := WRK0
35                      ; DPU USES C.48 - 16.96 MYS  =  3 - 85 GPU MICROS
36
37                      ; CLEAR STATUS AND GOTO WAIT FPU READY
38 C1500 011754005015        SZB   OR,M2,WRK3          ; WRK3 := 3
39 C1501 011664005104        SAB   CAND,M2,STAT,STAT   ; STATUS(22:23) := 0
40 C1502 434400001573        JMP   F,F,WFPUR           ; GOTO WAIT DPU READY
41
42
```

```
01
02
03
04                      ; INIT
05
06                      ;
07                      ; INIT        W (DUMMY)  SB (=MODUS<12 CON BLOCK-EXPONENT)
08                      ;             MODUS: 1<11 : CLEAR AR
09                      ;                    1<10 : NEGATIVE ACCUMULATION
10                      ;                    1< 9 : BLOCK FLOATING MODE
11                      ;
12
13                      MODUS:
14
15                      ; TAKE EXP FROM MODUS
16 C1503 034402000321           CAL   F,F,GETOP             ; WORK0 := MODUS;
17 C1504 010660356240 MODU1:    SAR   AND,M12,WRK0,FPINI    ; FPUOP1 := 12 EXT 0 CON EXP0
18                                                          ; UNITFNC := INIT
19                      ; LOAD MODUS AND 1B16
20 C1505 011664006252           SAB   CAND,M12,WRK0,WRK0    ; WORK0 := MODUS;
21 C1506 40040000020C           LDIM  0000,0200             ; IMOP := 1B16
22 C1507 014654140240           RAR   OR,IMOP,WRK0,FPOP0    ; FPUOP0 := MODUS + 1B16
23
24                      ; WAIT DPU READY
25 C1510 434402701513 FPUW7:    CAL   T,INTR,DPUIN          ; IF INTERRUPT GOTO DPUIN
26 C1511 414754006000           RZ    OR,FPUR0              ; IF FPU_NOT_READY
27 C1512 234400631510           JMP   T,FPUNR,FPUW7  RTN    ; GOTO FPUW7 ELSE RETURN
28
29                      ; INTERRUPT WHILE WAITING ON DPU
30 C1513 000400000017 DPUIN:    LDIM  0000,0017             ; MASK
31 C1514 015754000014           RZB   OR,IMOP,WRK2          ; WORK2 := INTERRUPT_LEVEL
32 C1515 015660072314           RABR  AND,ILEV,WRK2,WRK2,MIX ; MIR := WORK2 := ILEV EXTRACT 8
33 C1516 434401020000           JMX   F,NZ                  ; IF ILEV = 0 THEN GOTO (MIX)
34 C1517 014614100300           CLIN  WRK2                  ; CLEAR INTERRUPT
35 C1520 010645005700           SA    SUN0,M8,WRK2          ; IF ILEV >= 8 THEN
36 C1521 634400411522           JMP   T,NNEG,DPUI1   RTN    ; GOTO DPUI1 ELSE RETURN
37 C1522 000400000300 DPUI1:    LDIM  0000,0300             ; 192 * 0.40 = 77 MICROSEC
38 C1523 434402000236           CAL   F,F,DELAY             ; WAIT UNTIL DPU MUST BE FINISHED
39 C1524 015614000312           ZAB   OR,WRK2,WRK0          ; WORK0 := ILEV
40 C1525 034400000026           JMP   F,F,CPUIN             ; GOTO CPUIN
41
42
43
```

```
10059 GPU MICROPROGRAM  GI APR 1982
01
02
03
04                      ; MLA
05
06                      ;
07                      ; MLA       W( ADDR_A)   WPRE( ADDR_B)   WO ( REPETITIONS)
08                      ; AT RETURN : W = W + 4 * WO;  WPRE = WPRE + 4 * WO; WO = 0;
09                      ;           ELSE EXCEPTION
10                      ;           WORK3 = 3;
11                      ; MLA ADDRESSING A D_WORD REGISTER MAY ONLY REPEAT ONCE (WO=1)
12                      ; ELSE THE RESULT IS UNPREDICTABLE AS THE ADDR LOOPS INSIDE
13                      ; THE REGISTERS W0,W1,W2,W3
14
15                      ; THIS VERSION LOADS BOTH OPERANDS BEFORE CALLING DPU
16
17                      ; TEST ZERO REPETITION
18 C1526 414611000CCC MLA:    ZA     SUB0,W0              ; IF WO <= 0 THEN RETURN
19 C1527 234400011530         JMP    F,NNEG,MLA1     RTN ; ELSE GOTO MLA1
20
21                      ; CLEAR STATUS AND SET ZEROTEST IN IMOP
22 C1530 011740005015 MLA1:   SZB    ADD,M2,WRK3          ; WRK3 := 3
23 C1531 011664005104         SAB    CAND,M2,STAT,STAT    ; STATUS(22:23) := 0
```

```
!0060 GPU MICROPROGRAM  GI APR 1982
01
02                          ; MLA - LOOP
03 C1532 41415400000C MLAL:   ZBQ    OR,W              ; Q := ADDR_A
04 C1533 41551400000C         ZQB    OR,SB             ; SB := ADDR_A
05 C1534 034402000633         CAL    F,F,GTDWD         ; WRK0 CON WRK1 := W(SB-2) CON W(SB)
06
07                          ; TEST MULTIPLICAND = 0
08 C1535 434400421540         JMP    T,NZ,MLAL1        ; IF WORK0 <> 0 THEN GOTO MLAL1;
09 C1536 410670002260         SA     EXOR,MESS,WRK1    ; IF WRK1 = MESS
10 C1537 434400021566         JMP    F,NZ,MLAC         ; THEN GOTO MLA-CONTROL
11
12                          ; SAVE MULTIPLICAND
13 C1540 415614000277 MLAL1:  ZAB    OR,WRK1,WRK5      ; WRK5 := WRK1
14 C1541 415614000256         ZAB    OR,WRK0,WRK4      ; WRK4 := WRK0
15
16                          ; LOAD ADDR_B
17 C1542 014154000001         ZBQ    OR,WPRE           ; Q := ADDR_B
18 C1543 41551400000C         ZQB    OR,SB             ; SB := Q
19
20                          ; GET MULTIPLICATOR
21 C1544 034402000633         CAL    F,F,GTDWD         ; WRK0 CON WRK1 := W(SB-2) CON W(SB)
22
23                          ; TEST MULTIPLICATOR = 0
24 C1545 034400421550         JMP    T,NZ,FPUW1        ; IF WORK0 <> 0 THEN GOTO FPUW1;
25 C1546 410670002260         SA     EXOR,MESS,WRK1    ; IF WRK1 = MESS
26 C1547 434400021566         JMP    F,NZ,MLAC         ; THEN GOTO MLA-CONTROL
```

```
!0061 GPU MICROPROGRAM  GI APR 1982
01
02                      ; WAIT DPU
03 C1550 434402701513 FPUW1: CAL    T,INTR,DPUIN        ; IF INTERRUPT GOTO DPUIN
04 C1551 41475400600C        RZ     OR,FPURC            ; IF FPU_NOT_READY
05 C1552 03440063155C        JMP    T,FPUNR,FPUW1       ; THEN GOTO FPUWAITPOINT1
06
07                      ; TEST STATUS
08 C1553 014260007320        RAQ    AND,FPUST,WRK3      ; Q := FPUSTATUS(22:23)
09 C1554 034402421601        CAL    T,NZ,DPUER          ; IF Q <> 0 THEN GOTO DPUERROR
10
11                      ; START DPU SHORT LOAD
12 C1555 014554750013        ZBR    OR,WRK1,FPUSL       ; FPUOP1 := WRK1 ; UNITFNC := SL
13 C1556 014554140012        ZBR    OR,WRK0,FPOP0       ; FPUOP0 := WRK0
14                      ;      DPU READY AFTER 0.64 MYS  =  GPU 4 MICROS
15
16 C1557 41451400000C        ZQ     OR                  ; NO OP
17 C1560 415441000321        ABB    ADD0,WRK3,WPRE      ; WPRE := WPRE + 4
18 C1561 015441000320        ABB    ADD0,WRK3,W         ; W := W + 4
19 C1562 014620220240        ZAR    AND,WRK0,READP      ; I/O ADDR := 0, READP
20
21 C1563 414554150017        ZBR    OR,WRK5,FPOP1       ; FPUOP1 := WRK5 <* SYNCRO BY SL *>
22 C1564 414554140016        ZBR    OR,WRK4,FPOP0       ; FPUOP0 := WRK4
23                      ;      DPU USES 4.64 - 20.96 MYS  =  3 - 105 GPU MICROS
24 C1565 034400001571        JMP    F,F,MLAC1           ; GOTO MLAC1
25
26                      ; MLA - CONTROL
27                      MLAC:
28                      ; INCREASE ADDRESSES BY 4
29 C1566 415441000321        ABB    ADD0,WRK3,WPRE      ; WPRE := WPRE + 3+1
30 C1567 015441000320        ABB    ADD0,WRK3,W         ; W := W + 3+1
31 C1570 014620220240        ZAR    AND,WRK0,READP      ; I/O ADDR := 0, READP
32 C1571 015544000003 MLAC1: ZBB    SUN,GRX             ; W0 := W0 - 1
33 C1572 434400421532        JMP    T,NZ,MLAL           ;  IF W0 <> 0 THEN GOTO MLA - LOOP
```

```
01
02                      ; WAIT DPU READY AFTER OPERATION
03
04                      ;
05                      ; WAIT DPU READY AFTER OPERATION
06 C1573 434402701513 WFPUR: CAL    T,INTR,DPUIN          ; IF INTERRUPT GOTO DPUIN
07 C1574 414754006CCC        RZ     OR,FPUR0              ; IF FPU_NOT_READY
08 C1575 434400631573        JMP    T,FPUNR,WFPUR         ; THEN GOTO WAIT_FPU_READY
09 C1576 01426000732C        RAQ    AND,FPUST,WRK3        ; Q := FPUSTATUS(22:23)
10 C1577 06140002C1C4        NSAQ   ADD,STAT,STAT,NL,F,NZ  CRTN; STATUS := STATUS + Q
11                                                         ; IF Q = 0 THEN RETURN
12 C1600 23440055120C6       JMP    T,FPMSK,INTF    RTN ; IF FPMASK THEN GOTO INTF ELSE RETURN
13
14
15                      ; DPU ERROR
16
17                      ;
18                      ; FPU ERROR
19 C1601 415400000C1C4 DPUER: AQR    ADD,STAT,STAT         ; STAT := STAT + Q;
20 C1602 23440055120C6       JMP    T,FPMSK,INTF    RTN ; IF FPMASK THEN GOTO INTF ELSE RETURN
21
22
23
```

```
10063 GPU MICROPROGRAM  GI APR 1982
01
02
03
04                      ; ARM      ACCUMULATOR REGISTER MULT
05                      ;          AR := AR * DWD(SR)
06
07 C1603 034402000633 ARM:    CAL    F,F,GTDWD           ; WRK0,WRK1 := MULTIPLICATOR
08 C1604 400400047070         LDIM   0004,7070           ; CBCR :=
09 C1605 414754110000         RZR    OR,IMOP,CBCR        ;           4,7070
10 C1606 014554150013         ZBR    OR,WRK1,FPARM       ; FPUOP1 := WRK1; UNITFNC := ARM
11 C1607 014554140012         ZBR    OR,WRK0,FPOP0       ; FPUOP0 := WRK0
12 C1610 011664005104         SAB    CAND,M2,STAT,STAT   ; STATUS(22:23) := 0;
13 C1611 011754005015         SZB    OR,M2,WRK3          ; WRK3 := 3
14 C1612 000400007070         LDIM   0000,7070           ; CBCR :=
15 C1613 414754110000         RZR    OR,IMOP,CBCR        ;           0,7070
16 C1614 434400001573         JMP    F,F,WFPUR           ; GOTO WFPUR
17
18
```

```
01
02
03
04                      ; INV
05
06                      ; INV          W(=STORE_ADDR)    SB(=LOAD_ADDR)
07                      ;              WPRE IS USED DURING NORMALIZATION WHEN OP UNNORMALIZED
08                      ;
09
10 C1615 034402000633 INV:    CAL    F,F,GTDWD           ; WRK0 CON WRK1 := W(SB-2) CON W(SB)
11
12                      ; TEST ZERO (I.E. EXP == -2048)
13 C1616 034400451632         JMP    T,NORM,INV1         ; IF NORMALIZED GOTO INV1;
14 C1617 41415400000C         ZBQ    OR,W                ; Q := W
15 C1620 015514000017         ZQB    OR,WRK5             ; WORK5 := Q;
16 C1621 415354000674         EXT    WRK1,WRK2           ; WORK2 := EXP;
17 C1622 011664006273         SAB    CAND,M12,WRK1,WRK1  ; WORK1 := FRAC(24:35) CON 0;
18 C1623 434402001145         CAL    F,F,NORMA           ; GOTO NORMALIZE
19 C1624 41415400000C         ZBQ    OR,W                ; Q := W
20 C1625 415514000013         ZQB    OR,WRK1             ; WORK1 := Q;
21 C1626 015614000C3C         ZAB    OR,WRK3,W           ; W := WORK3
22 C1627 014154000001         ZBQ    OR,WPRE             ; Q := WPRE
23 C1630 015514000012         ZQB    OR,WRK0             ; WORK0 := WPRE;
24 C1631 034400021657         JMP    F,NZ,INVER          ; IF ZERO THEN GOTO INVER;
25
26                      ; START DPU INV
27 C1632 400400047070 INV1:   LDIM   0004,7070           ; CBCR :=
28 C1633 414754110CCC         RZR    OR,IMOP,CBCR        ;          4,7070
29 C1634 414554350013         ZBR    OR,WRK1,FPINV       ; FPUOP1 := WRK1 ; UNITFNC := INV
30 C1635 014554140012         ZBR    OR,WRK0,FPOP0       ; FPUOP0 := WRK0
31
32                      ; GET STORE_ADDRESS TO SB AND CLEAR STATUS
33 C1636 41415400000C INV2:   ZBQ    OR,W                ; Q := W
34 C1637 415514000007         ZQB    OR,SB               ; SB := Q
35 C1640 011754005015 INV3:   SZB    OR,M2,WRK3          ; WRK3 := 3;
36 C1641 011664005104         SAB    CAND,M2,STAT,STAT   ; STATUS(22:23) := 0
37 C1642 000400007070         LDIM   0000,7070           ; CBCR :=
38 C1643 414754110CCC         RZR    OR,IMOP,CBCR        ;          0,7070
```

```
10065 GPU MICROPROGRAM  GI APR 1982
01
02                      ; WAIT DPU READY
03 C1644 434402701513 FPUW4: CAL   T,INTR,DPUIN        ; IF INTERRUPT GOTO DPUIN
04 C1645 41475400600C        RZ    OR,FPUR0           ; IF FPU_NOT_READY
05 C1646 034400631644        JMP   T,FPUNR,FPUW4      ; THEN GOTO FPUW4
06 C1647 015754006413        RZB   OR,FPUR1,WRK1      ; WRK1 := FPUR1
07 C1650 015754006012        RZB   OR,FPUR0,WRK0      ; WRK0 := FPUR0
08 C1651 434402002063        CAL   F,F,GPUST          ; SUBROUTINE GPU STORE
09 C1652 41451400000C        ZQ    OR                 ; NO OP
10 C1653 034404642656        WJMP  T,BERR,INTB        ; WAIT, IF BUSERROR GOTO INTB
11 C1654 014260007320        RAQ   AND,FPUST,WRK3     ; Q := FPUSTATUS(22:23)
12 C1655 061400020104        NSAQ  ADD,STAT,STAT,NL,F,NZ  CRTN ; STATUS := STATUS + Q
13                                                     ; IF Q = 0 THEN RETURN
14 C1656 234400551657        JMP   T,FPMSK,INVER  RTN ; IF FPMASK THEN GOTO INVER ELSE RETURN
15
16                      ; INV ERROR
17 C1657 4103540044CC INVER:  SZQ   OR,C2              ; Q(22) := 1 (=OVERFLOW)
18 C1660 011664005104         SAB   CAND,M2,STAT,STAT  ; STATUS(22:23) := 0;
19 C1661 034402001601         CAL   F,F,DPUER          ; GOTO DPU ERROR
20 C1662 015560000012         ZBB   AND,WRK0           ; WORK0 := 0
21 C1663 411754002013         SZB   OR,MESS,WRK1       ; WORK1 := 0,-2048
22 C1664 434400002054         JMP   F,F,ZSTR2          ; GOTO ZSTR2
23
24
25
26
```

```
10066 CPU MICROPROGRAM  GI APR 1982
01
02                      ; CMV   WPRE(=ADDR_A)   W(=ADDR_B)  SB(=ADDR_C)   WO(=REP_COUNT)
03                      ;
04                      ; CMV : B(I) := B(I) + A(I) * C , FOT I = 0, 1, ... , WO-1
05                      ;
06                      ; CMVI: B(I) := B(I) - A(I) * C , FOR I = 0, 1, ... , WO-1
07                      ;
08 01665 015560000006 CMV:    ZBB   AND,CAUSE            ; ADD_SIGN := +
09 01666 000440004000         LDIM  4000,4000            ; MODUS
10 01667 034400001672         JMP   F,F,CMV1             ; GOTO CMV1;
11
12 01670 011754006006 CMVI:   SZB   OR,M12,CAUSE         ; ADD_SIGN := -
13 01671 000460004000         LDIM  6000,4000            ; MODUS
14 01672 015754000012 CMV1:   RZB   OR,IMOP,WRK0         ; MODUS := CLEAR OR ADDSIGN OR
15 01673 004610006640         ZAS   SUB,WRK0,ILIM        ;   ZERO_EXP
16
17                      ; PREPARE THE LOOP
18 01674 011664005104         SAB   CAND,M2,STAT,STAT    ; STATUS(22:23) := 0;
19 01675 011754005015         SZB   OR,M2,WRK3           ; WORK3 := 3;
20 01676 014611000000         ZA    SUB0,WO              ; ALU := - REP_COUNT;
21 01677 234402010633         CAL   F,NNEG,GTDWD     RTN ; IF NEG GOTO GTDWD ELSE RETURN;
22 01700 434400421703         JMP   T,NZ,CMVLO           ; IF <> 0 THEN GOTO CMVLO
23 01701 010670002260         SA    EXOR,MESS,WRK1       ; IF WORK1 <> MESS
24 01702 234400421703         JMP   T,NZ,CMVLO       RTN ; THEN GOTO CMVLO ELSE RETURN;
25 01703 004610007660 CMVLO:  ZAS   SUB,WRK1,SP17        ; SP16, SP17 :=
26 01704 004610007240         ZAS   SUB,WRK0,SP16        ;               C;
```

```
!0067 GPU MICROPROGRAM  GI APR 1982
01
02                          ; CMV LOOP
03 C1705 014154000001 CMVL:   ZBQ    OR,WPRE              ; Q := ADDR_A;
04 C1706 415514000007         ZQB    OR,SB                ; SB := Q;
05 C1707 034402000633         CAL    F,F,GTDWD            ; WORK0,WORK1 := A;
06 C1710 034400421713         JMP    T,NZ,CMVL1           ; IF <> 0 THEN GOTO CMVL1;
07 C1711 410670002260         SA     EXOR,MESS,WRK1       ; IF WORK1 = MESS
08 C1712 434400021746         JMP    F,NZ,CMVC            ; THEN GOTO CMV CONTROL;
09 C1713 014214000240 CMVL1:  ZAQ    OR,WRK0              ; SAVE WORK0
10 C1714 011754006412         SZB    OR,ILIM,WRK0         ; WORK0 := MODUS
11 C1715 03440200154          CAL    F,F,MODU1            ; GOTO MODU1;
12
13                          ; MULT
14 C1716 014554750013         ZBR    OR,WRK1,FPUSL        ; FPUOP1 := WORK1; UNITFNC := SL;
15 C1717 414514140000         ZQR    OR,FPOP0             ; FPUOP0 := Q;
16 C1720 414154000000         ZBQ    OR,W                 ; Q := W
17 C1721 415514000007         ZQB    OR,SB                ; SB := Q
18 C1722 414514000000         ZQ     OR                   ; NO OP
19 C1723 414514000000         ZQ     OR                   ; NO OP
20                          ; LOAD C
21 C1724 410754157400         SZR    OR,SP17,FPOP1        ; FPUOP1 := SP17
22 C1725 410754147000         SZR    OR,SP16,FPOP0        ; FPUOP0 := SP16;
23
24                          ; GET B
25 C1726 034402000633         CAL    F,F,GTDWD            ; WORK0,WORK1 := B;
26
27                          ; WAIT
28 C1727 434402701513 FPUW3:  CAL    T,INTR,DPUIN         ; IF INTERRUPT THEN GOTO DPUIN;
29 C1730 414754006000         RZ     OR,FPUR0             ; IF DPU NOT_READY
30 C1731 434400631727         JMP    T,FPUNR,FPUW3        ; THEN GOTO FPUW3;
31 C1732 014260007320         RAQ    AND,FPUST,WRK3       ; Q := FPUSTATUS;
32 C1733 034402421601         CAL    T,NZ,DPUER           ; IF NOTZERO GOTO DPUER;
```

```
01
02
03                      ; ADD B;
04 01734 014614150140           ZAR    OR,CAUSE,FPULL        ; LOAD ADDSIGN AND ZEROTAIL
05 01735 014620140140           ZAR    AND,CAUSE,FPOPO       ;
06 01736 414154000000           ZBQ    OR,W                  ; Q := ADDR_B;
07 01737 415514000007           ZQB    OR,SB                 ; SB := Q;
08 01740 414514000000           ZQ     OR                    ; NO OP
09 01741 414514000000           ZQ     OR                    ; NO OP WAIT DPU READY
10 01742 434402001476           CAL    F,F,ADD2              ; GOTO ADD2;
11
12                      ; STORE
13 01743 014520550000           ZQR    AND,FPSTR             ; FPUOP1 := 0, UNITFNC := STORE
14 01744 014520140000           ZQR    AND,FPOPO             ; FPUOPO := 0
15 01745 034402001640           CAL    F,F,INV3              ; GOTO INV3
16
17                      ; CMV CONTROL
18 01746 415441000321 CMVC:     ABB    ADDO,WRK3,WPRE        ; ADDR_B := ADDR_B + 4;
19 01747 015441000320           ABB    ADDO,WRK3,W           ; ADDR_A := ADDR_A + 4;
20 01750 014520220000           ZQR    AND,READP             ; I/O ADDR := 0;
21 01751 015544000003           ZBB    SUN,GRX               ; WO := WO - 1;
22 01752 234400421705           JMP    T,NZ,CMVL       RTN   ; IF NOT_ZERO GOTO CMV LOOP
23                                                           ; ELSE RETURN
24
25
26                       FILL 2000
27
```

```
10069 GPU MICROPROGRAM  GI APR 1982
01
02
03
04                      ; STR
05
06                      ; STR          W(=HEADADDR)          SB(=DISP)
07                      ;              WORK5 = W     +++IT IS POSSIBLE TO STORE IN W+++
08                      ;              SB = 0 GIVE SINGLE PRECISION ROUNDED RESULT
09                      ;              SB <> 0 GIVE DOUBLE PRECISION ROUNDED RESULT
10
11
12                      STR:
13 C2000 414554000007           ZB     OR,SB                ; IF SB = 0
14 C2001 034400022057           JMP    F,NZ,STRS            ; THEN GOTO STORE_SINGLE
15
16                      ; START STORE DPU
17 C2002 400400200CCC STRD:     LDIM   0020,0000            ; DOUBLE ROUND CONSTANT =  1B31
18 C2003 414754550CCC           RZR    OR,IMOP,FPSTR        ; FPUOP1 := IMOP , UNITFNC := STORE
19 C2004 414560140012           ZBR    AND,WRK0,FPOP0       ; FPUOP0 := 0 (I.E. DOUBLE STORE)
20
21                      ; GET STOREADDRESSES
22 C2005 414154000CCC           ZBQ    OR,W                 ; Q := W
23 C2006 415400000167           AQB    ADD,SB,SB            ; SB := SB + Q
24 C2007 015514000017           ZQB    OP,WRK5              ; WRK5 := Q
25                      ; PREPARE WRK1 MASK FOR FRAC(24:35)
26 C2010 411774006013           SZB    EXNOR,M12,WRK1       ; WRK1 := 1 EXT 12 CON 0 EXT 12
27 C2011 400400000043           LDIM   00CC,0043            ; IMOP := 35
28 C2012 415754000016           RZB    OR,IMOP,WRK4         ; WRK4 := 35
29 C2013 410354005CCC           SZQ    OR,M2                ; Q := 3;
30 C2014 011664005104           SAB    CAND,M2,STAT,STAT    ; STATUS(22:23) := 0;
31
32                      ; WAIT DPU
33 C2015 434402701513 FPUW6:    CAL    T,INTR,DPUIN         ; IF INTERRUPT GOTO DPUIN
34 C2016 414754006CCC           RZ     OR,FPUR0             ; IF FPU_NOT_READY
35 C2017 434400632015           JMP    T,FPUNR,FPUW6        ; THEN GOTO FPUW6
36 C2020 015754006415           RZB    OR,FPUR1,WRK3        ; WORK3 := TAILFRAC(24:35) CON HEADEXPO
37 C2021 414320007CCC           RQQ    AND,FPUST            ; Q(22:23) := DPUSTATUS(22:23)
38 C2022 434400422046           JMP    T,NZ,STRER           ; IF <>0 THEN GOTO STR ERROR;
39 C2023 015754006012           RZB    OR,FPUR0,WRK0        ; WORK0 := TAILFRAC(0:23);
40 C2024 434400012047           JMP    F,NNEG,ZSTR          ; IF NEG THEN GOTO ZSTR;
41 C2025 414520150CCC           ZQR    AND,FPOP1            ; START DPU
42 C2026 014520140CCC           ZQR    AND,FPOP0            ; START DPU
43 C2027 415460000333           ABB    AND,WRK3,WRK1        ; WRK1 := FRAC(24:35) CON 12 EXT 0
```

```
00070 GPU MICROPROGRAM  GI APR 1982
01
02                      ; SET TAIL EXPO
03 02030 015354000735           EXT   WRK3,WRK3           ; WRK3 := SIGNEXTENDED HEAD_EXPO
04 02031 015445000355           ABB   SUNO,WRK4,WRK3      ; WORK3 := WORK3 - 35 <*TAIL EXP*>
05 02032 410640002320           SA    ADD,MESS,WRK3       ; IF WORK3 < -2048
06 02033 434400012047           JMP   F,NNEG,ZSTR         ; THEN GOTO ZERO STORE
07 02034 411660006335           SAB   AND,M12,WRK3,WRK3   ; WRK3 := TAIL EXPO
08 02035 015454000333           ABB   OR,WRK3,WRK1        ; WRK1 := TAIL(FRAC(24:35) CON EXPO)
09 02036 434402002063           CAL   F,F,GPUST           ; STORE TAILPART
10
11                      ; LOAD HEADADDRESS
12 02037 415614000367           ZAB   OR,WRK5,SB          ; SB := WRK5
13
14                      ; NOW DPU MUST BE READY
15 02040 015754006413 STR1:     RZB   OR,FPUR1,WRK1       ; WRK1 := TAIL(FRAC(24:35) CON EXPO)
16 02041 015754006012           RZB   OR,FPUR0,WRK0       ; WRK0 := FRAC(0:23)
17 02042 034404642656           WJMP  T,BERR,INTB         ; WAIT, IF BUSERROR GOTO INTB
18 02043 434402002063           CAL   F,F,GPUST           ; GOTO GPUSTORE
19 02044 414514000000           ZQ    OR                  ; NO OP
20 02045 634404642656           WJMP  T,BERR,INTB    RTN  ; WAIT, IF BUSERROR GOTO INTB
21                                                        ; ELSE RETURN
22
23 02046 034402001601 STRER:    CAL   F,F,DPUER           ; GOTO DPUER
24
25                      ; ZERO STORE
26 02047 415620000252 ZSTR:     ZAB   AND,WRK0,WRK0       ; WRK0 := 0
27 02050 411754002013 ZSTR1:    SZB   OR,MESS,WRK1        ; WORK1 := 0,-2048;
28 02051 434402002063           CAL   F,F,GPUST           ; CAL GPUSTORE
29 02052 415614000367           ZAB   OR,WRK5,SB          ; SB := WRK5
30 02053 034404642656           WJMP  T,BERR,INTB         ; WAIT, IF BUSERROR GOTO INTB
31 02054 434402002063 ZSTR2:    CAL   F,F,GPUST           ; GOTO GPUSTORE
32 02055 414514000000           ZQ    OR                  ; NO OP
33 02056 634404642656           WJMP  T,BERR,INTB    RTN  ; WAIT, IF BUSERROR GOTO INTB
34                                                        ; ELSE RETURN
35                      ; STORE SINGLE
36 02057 014520550000 STRS:     ZQR   AND,FPSTR           ; FPUOP1 := 0 , UNITFNC := STORE
37 02060 014520140000           ZQR   AND,FPOP0           ; FPUOP0 := 0
38 02061 034400001636           JMP   F,F,INV2            ; GOTO INV2
39
40                      ; KS = KEY STORE : 51
41 02062 214514000000 KS:       ZQ    OR             RTN  ; NO_OP RETURN
42
43
```

```
!0071 GPU MICROPROGRAM  GI APR 1982
01
02
03
04                      ; SUBROUTINE GPU STORE
05                      ; ********************
06                      ; GPU STORE STORE WORK0 CON WORK1 IN THE MEMORY
07                      ;      PRELOCATION AND LOCATION
08
09                      ; WORK2 = SB + BASE
10 C2063 011640000574 GPUST: SAB    ADD,BASE,SB,WRK2    ; WRK2 := SB + BASE
11 C2064 0106450013CC        SA     SUN0,LLIM,WRK2      ; IF WRK2 < LOWER_LIMIT
12 C2065 03440001210C        JMP    F,NNEG,GPUS1        ; THEN GOTO GPUS1
13
14 C2066 0106500017CC        SA     SUB,ULIM,WRK2       ; IF WRK2 >= UPPR_LIMIT
15 C2067 434400012672        JMP    F,NNEG,INT0         ; THEN GOTO INT0
16
17 C2070 01461403026C        ZAR    OR,WRK1,DAT0        ; DATAOUT := WORK1
18 C2071 0146146203CC        ZAR    OR,WRK2,WRTP        ; I/O ADDR := WRK2, WRITEP
19 C2072 034400602672        JMP    T,NMADR,INT0        ; IF NOT_M_ADDR GOTO INT0
20
21 C2073 011645004714        SAB    SUN0,C2,WRK2,WRK2   ; WRK2 := WRK2 - 2
22 C2074 0106450013CC        SA     SUN0,LLIM,WRK2      ; IF WRK2 < LOWER_LIMIT
23 C2075 434400012101        JMP    F,NNEG,GPUS2        ; THEN GOTO GPUS2
24
25                      ; WAIT
26 C2076 034404642656        WJMP   T,BERR,INTB         ; IF BUSERROR GOTO INTB
27
28 C2077 41461403024C        ZAR    OR,WRK0,DAT0        ; DATAOUT := WRK0
29 C21C0 6146146203CC        ZAR    OR,WRK2,WRTP   RTN  ; I/O ADDR := WRK2, WRITEP
30                                                      ; RETURN
```

```
10072 GPU MICROPROGRAM  GI APR 1982
01
02                      ; IT IS POSSIBLE TO STORE IN MEMORY(8) AND W3
03 02101 011645004567 GPUS2: SAB    SUNO,C2,SB,SB        ; SB := SB - 2
04 02102 415614000253        ZAB    OR,WRK0,WRK1         ; WRK1 := WRK0
05 02103 415620000314        ZAB    AND,WRK2,WRK2        ; WRK2 := 0
06 02104 034404642656        WJMP   T,BERR,INTB          ; WAIT, IF BUSERROR GOTO INTB
07 02105 034400002107        JMP    F,F,GPUS3            ; GOTO GPUS3
08
09 02106 411754006014 GPUS1: SZB    OR,M12,WRK2          ; WRK2 := 0 EXT 12 CON 1 EXT 12
10 02107 410645005560 GPUS3: SA     SUNO,M8,SB           ; IF SB >= 8
11 02110 034400412672        JMP    T,NNEG,INTO          ; THEN GOTO INTO
12 02111 414554220007        ZBR    OR,SB,READP          ; I/O ADDR := SB, READP
13 02112 434400612672        JMP    T,NWADR,INTO         ; IF NOT_W-REG GOTO INTO
14 02113 415614000263        ZAB    OR,WRK1,GRX          ; W(I/O) := WRK1
15
16 02114 414614000300        ZA     OR,WRK2              ; IF WRK2 <> 0
17 02115 634400422116        JMP    T,NZ,GPUS4     RTN ; THEN GOTO GPUS4 ELSE RETURN
18
19 02116 011645224567 GPUS4: SABR   SUNO,C2,SB,SB,READP ; I/OADDR := SB := SB - 2, READP
20 02117 615614000243        ZAB    OR,WRK0,GRX    RTN ; W(I/O) := WRK0;
21
22
23
```

```
!0C73 GPU MICROPROGRAM  GI APR 1982
01
02
03
04                      ; SQUAREROOT  WPRE,W := SQRT( DWD(SB) )
05                      ;
06                      ; THE SQUAREROOT OF THE FLOATING POINT DWD(SB) IS TAKEN
07                      ; AND THE RESULT IS STORED IN WPRE,W
08                      ; NEGATIVE RADICANDS GIVE OVERFLOW
09
10 C2120 034402000633 SQRT:  CAL   F,F,GTDWD              ; LOAD RADICAND
11 C2121 434400452132         JMP   T,NORM,SQRT1          ; IF NORMALIZED GOTO SQRT1
12
13 C2122 415354000674         EXT   WRK1,WRK2             ; WORK2 := EXP
14 C2123 011664006273         SAB   CAND,M12,WRK1,WRK1    ; WORK1 := FRAC(24:35) CON 0
15 C2124 434402001145         CAL   F,F,NORMA             ; GOTO NORMALIZE
16 C2125 414154000CCC SQRT2:  ZBQ   CR,W                  ; WORK1:=
17 C2126 415514000013         ZQB   CR,WRK1               ;      W;
18 C2127 014154000001         ZBQ   CR,WPRE               ; WORK0 :=
19 C2130 015514000012         ZQB   CR,WRK0               ;        WPRE;
20 C2131 234400452132         JMP   T,NORM,SQRT1    RTN ; IF NORMALIZED GOTO SQRT1 ELSE RETURN
21
22 C2132 034400012211 SQRT1: JMP    F,NNEG,SQRTE         ; IF NEGATIVE THEN GOTO SQRT EXCEPTION
23                      ; SAVE RADICAND
24 C2133 415614000277         ZAB    OR,WRK1,WRK5        ; WORK5 := WORK1
25 C2134 415614000256         ZAB    OR,WRK0,WRK4        ; WORK4 := WORK0
26
27                      ; SAVE INTEGER RADICAND
28 C2135 411750004413         SZB    SUB,C2,WRK1         ; WORK1 := 2 - 1
29 C2136 414460000373         AB     AND,WRK5,WRK1       ; ALU := EXP EXTRACT 1
30 C2137 434400422141         JMP    T,NZ,BFSQ4          ; IF EVEN EXPO
31 C2140 022554000012         SSRB   OR,WRK0,Z,NL        ; THEN WORK0 := WORK0 // 2
32 C2141 022554000012 BFSQ4:  SSRB   OR,WRK0,Z,NL        ; WORK0 :=
33 C2142 022554000012         SSRB   OR,WRK0,Z,NL        ;        WORK0 // 4
34 C2143 415614000247         ZAB    OR,WRK0,SB          ; SB := RADICAND
35 C2144 415614000241         ZAB    OR,WRK0,WPRE        ; WPRE := RADICAND
```

```
10074 GPU MICROPROGRAM  GI APR 1982
01
02                      ; LINEAR APPR OF SQRT(WPRE,W) (WPRE,W AS LONG)
03 02145 400407337766         LDIM   0733,7766              ; IMOP := 1 949 686
04 02146 015640000172         RAB    ADD,IMOP,SB,WRK0       ; WORK0 := 1 949 686
05 02147 415440000172         ABB    ADD,SB,WRK0            ;         + 2 * RADICAND;
06 02150 015614000246         ZAB    OR,WRK0,CAUSE          ; CAUSE := ITERAND
07
08                      ; NEWTON INTEGER
09 02151 434402000411         CAL    F,F,WDBF               ; W := WPRE // WORK0;
10                                                           ; WDBF ADDR FIRST AFTER WD
11 02152 422440000140         SSRA   ADD,CAUSE,W,Z,NL       ; W := (W+CAUSE) // 2
12 02153 414154000000         ZBQ    OR,W                   ; WORK0 :=
13 02154 015514000012         ZQR    OR,WRK0                ; CAUSE :=
14 02155 015514000006         ZQB    OR,CAUSE               ;          ITERAND;
15 02156 015614000161         ZAB    OR,SB,WPRE             ; WPRE := RADICAND
16 02157 434402000411         CAL    F,F,WDBF               ; W := WPRE // WORK0
17 02160 015440000140         ABB    ADD,CAUSE,W            ; W := W + CAUSE
18 02161 434400032163         JMP    F,OVFL,BFSQ5           ; IF -, OVERFLOW GOTO BFSQ5
19 02162 022554000000         SSRB   OR,W,Z,NL              ; ELSE W := W // 2;
20
```

```
!0075 GPU MICROPROGRAM  GI APR 1982
01
02
03
04                      ; PREARE NEWTON REAL
05 C2163 414154000CCC BFSQ5: ZBQ    OR,W                    ; WORK0 :=
06 C2164 015514000012        ZQB    OR,WRK0                 ;         W;
07 C2165 015614000341        ZAB    OR,WRK4,WPRE            ; WPRE := FRAC(0:23)
08 C2166 01561400036C        ZAB    OR,WRK5,W               ; W := FRAC(24:35) CON EXP;
09 C2167 015354000774        EXT    WRK5,WRK2               ; WORK2 := EXP
10 C2170 022541004014        SSRB   ADD0,WRK2,SGN,NL        ; WORK2 := (EXP + 1) // 2;
11 C2171 011660006313        SAB    AND,M12,WRK2,WRK1       ; WORK1 := WORK2 EXTRACT 12
12 C2172 015614000246        ZAB    OR,WRK0,CAUSE           ; CAUSE := WORK0
13 C2173 015614000267        ZAB    OR,WRK1,SB              ; SB := WORK1
14                      ; DIVIDE
15 C2174 034402001403        CAL    F,F,FDBF                ; WPRE,W := WPRE,W / WORK0WORK1
16                                                           ; FDBF AFTER GTDWD IN FRONT OF
17                                                           ; SAB CAND,M12,WRK1,WRK5
18 C2175 01561400C152        ZAB    OR,CAUSE,WRK0           ; WORK0 := CAUSE
19 C2176 015614000173        ZAB    OR,SB,WRK1              ; WORK1 := SB <* REAL ITERAND *>
20                      ; SET ADDCONDITION AND ADD
21 C2177 410354005CCC        SZQ    OR,M2                   ; Q := 3
22 C2200 022154000417        DSRB   OR,WRK5,Z,MC            ; ADDCOND := Q(23)
23 C2201 034402001334        CAL    F,F,FABF                ; WPRE,W := WPRE,W + WORK0,WORK1
24                                                           ; FABF AFTER FSALI IN FRONT OF
25                                                           ; EXT WRK1,WRK2
26
27                      ; DIVIDE BY 2
28 C2202 414154000CCC        ZBQ    OR,W                    ; WORK2 :=
29 C2203 015514000014        ZQB    OR,WRK2                 ;
30 C2204 015354000714        EXT    WRK2,WRK2               ;         EXP;
31 C2205 010324006CCC        SQQ    CAND,M12                ; Q := FRAC(24:35)
32 C2206 015544000014        ZBB    SUN,WRK2                ; WORK2 := EXP - 1
33 C2207 411660006314        SAB    AND,M12,WRK2,WRK2       ; WORK2 := EXP EXTRACT 12
34 C2210 215414000CCC        AQB    OR,WRK2,W          RTN  ; W := FRAC CON EXP
35
36
37 C2211 415614000241 SQRTE: ZAB    OR,WRK0,WPRE            ; WPRE := RES;
38 C2212 415614000260        ZAB    OR,WRK1,W               ; W := WORK1
39 C2213 011654004504        SAB    OR,C2,STAT,STAT         ; STATUS := OVERFLOW;
40 C2214 234400551206        JMP    T,FPMSK,INTF       RTN  ; IF FPMASK THEN GOTO INTF ELSE RETURN
41
```

```
10076 GPU MICROPROGRAM  GI APR 1982
01
02
03                      ;   AUTO REDUTION OF A DATAMATIC BLOCK
04
05                      ;
06                      ; CALL          BF     W3      X1       C5.
07                      ;
08                      ; W1 MAY BE ZERO ATT CALL ( NOT CLAIMED ).
09                      ;
10                      ; W0 W1 W2 W3 IS UNDEFINED AT RETURN.
11
12                      ;
13                      ; WORKLOCATIONS USED IN BLOCK FLOATING AUTO REDUCTION RESERVED BY THE
14                      ; CALLING PROCESS:
15                      ;
16
17                      ; B0   = BASE OF WORKLOCATION  ; CAT_I1U + NLU_BASE
18                      ; C4   =    BC +  2  ; AUTORED
19                      ; C3   =    BC +  4  ; LR
20                      ; B7   =    BC +  6  ; (R_MAX + 1)
21                      ; N2   =    BC +  8  ; SZU
22                      ; C1   =    BC + 10  ; CAT_SZR + NLR_BASE
23                      ; C0   =    BC + 12  ; CAT_I1R
24                      ; N11  =    BC + 14  ; 4 * (R_MAX + 1)
25                      ; B5   =    BC + 16  ; TAIL_DISP
26                      ; C5   =    BC + 18  ; BLOCK_FLOATING-, ACU-MODE / ADDSIGN
27                      ; C6   =    BC + 20  ; BLOCK_EXP_R_BASE
28                      ; N13  =    BC + 22  ; BLOCK_EXP_U TO BE SUBTRACTED FROM EXP
29                      ; B8   =    BC + 24  ; BLOCK_EXP_U_BASE
30                      ; B6   =    BC + 26  ; EXP_LIM
31                      ; B4   =    BC + 28  ; STATUS_BASE
32                      ; N0   =    BC + 30  ; U
33                      ; B1   =    BC + 32  ; CAT_SZU + NLU_BASE
34                      ; C2   =    BC + 34  ; FR
35                      ; B3   =    BC + 36  ; LU
36                      ; B2   =    BC + 38  ; FU
```

```
!0077 GPU MICROPROGRAM  GI APR 1982
01
02                      ; WORKLOCATIONS IN GPU REGISTERS :
03                      ;
04                      ; N1  =  CAUSE      ; I1U
05                      ; N3  =  W1 ,  X    ; R
06                      ; N4  =  WORK4      ; I1R
07                      ; N5  =  WORK2      ; SZR
08                      ; N6  =  W3 ,  W    ; ADDR_A <* IN W *>
09                      ; N7  =  W2 , WPRE  ; ADDR_B <* IN WPRE *>
10                      ; N8  =  ILIM       ; FULL_RED
11                      ; N9  =  W0         ; REPETITIONS
12                      ; N10 =  RTC        ; ACCU_ADDR = STORE_ADDR
13                      ;        SP17       ; ADDR N2 IN WORKAREA
14                      ;        SP16       ; MODUS, BLOCKFLOATING MODE
15
16                      ; AFTER EACH CALL HAS THE CONTENT BEEN CHANGED IN :
17                      ;
18                      ; C5   :  ACU_MODE
19
20                      ; TEST OP
21 C2215 014051000162 BF:    ABQ   SUB0,SB,X          ; Q := SB - W1
22 C2216 014520220CCC        ZQR   AND,READP          ; I/O-ADDR := 0
23 C2217 415520000CC3        ZQB   AND,GRX            ; W0 := 0
24 C2220 41175000440C2       SZB   SUB,C2,X           ; X := 1
25 C2221 01175400440C1       SZB   OR,C2,WPRE         ; WPRE := 2
26 C2222 4117540050CC        SZB   OR,M2,W            ; W := 3
27 C2223 410650004420        SA    SUB,C2,W1          ; IF W1 <> 1
28 C2224 034400420135        JMP   T,NZ,ILLOP         ; THEN GOTO ILLOP
29 C2225 41065100506C        SA    SUB0,M2,W3         ; IF W3 <> 3
30 C2226 034400420135        JMP   T,NZ,ILLOP         ; THEN GOTO ILLOP
31
32
```

```
!0078 GPU MICROPROGRAM  GI APR 1982
01
02
03
04                          ; SET MODUS AND ADD SIGNBIT
05 C2227 415514000007               ZQB   OR,SB                 ; SB := ADDR C5
06 C2230 034402000321               CAL   F,F,GETOP             ; WORK0 := ACU_MODE
07
08
09 C2231 400410000000               LDIM  1000,0000             ; IMOP := BLOCK_FL MASK
10 C2232 415660000256               RAB   AND,IMOP,WRK0,WRK4    ; WORK4 := BL_FL - BIT
11 C2233 015354001257               SWAP  WRK0,WRK5             ; WORK5 := WORK0(12:23, 0:11);
12 C2234 400420000000               LDIM  2000,0000             ; IMOP := SIGNBIT_MASK
13 C2235 415660000375               RAB   AND,IMOP,WRK5,WRK3    ; WORK3 := SIGNBIT;
14 C2236 015454000355               ABB   OR,WRK4,WRK3          ; WORK3 := SIGNBIT OR BLFL
15 C2237 411660006252               SAB   AND,M12,WRK0,WRK0     ; WORK0 := ADD_SIGN
16 C2240 034402002554               CAL   F,F,STB               ; STORE IN C5.
17 C2241 400440000000               LDIM  4000,0000             ; MODUS := CLEAR_R
18 C2242 015654000332               RAB   OR,IMOP,WRK3,WRK0     ;        OR SIGN
19 C2243 015354001356               SWAP  WRK4,WRK4             ;
20 C2244 415454000352               ABB   OR,WRK4,WRK0          ; SP16 := 00000, BLFL
21 C2245 404550007012               ZBS   SUB,WRK0,SP16         ;      OR MODUS, 0000
22
23                          ; SET 4*(R_MAX+1)
24 C2246 000400000014               LDIM  0000,0014             ; IMOP := 12
25 C2247 415645000167               RAB   SUN0,IMOP,SB,SB       ; SB := C5 - 12 = B7
26 C2250 011640004573               SAB   ADD,C2,SB,WRK1        ; WORK1 := B7 + 2 = N2
27 C2251 404550007413               ZBS   SUB,WRK1,SP17         ; SP17 := ADDR N2
28 C2252 034404642656               WJMP  T,BERR,INTB           ; WAIT, IF BUSERROR GOTO INTB
29 C2253 034402000321               CAL   F,F,GETOP             ; WORK0 := R_MAX + 1
30 C2254 015440000252               ABB   ADD,WRK0,WRK0         ;          * 2;
31 C2255 411645004577               SAB   SUN0,C2,SB,WRK5       ; WORK5 := B7 - 2 = C3
32 C2256 011704005007               SQB   SUN,M2,SB             ; SB := C5 - 4 = N11 <* Q = C5 *>
33 C2257 034402002554               CAL   F,F,STB               ; STORE WORK0 IN N11
34
35                          ; GET FU
36 C2260 000400000024               LDIM  0000,0024             ; IMOP := 20
37 C2261 415640000176               RAB   ADD,IMOP,SB,WRK4      ; WORK4 := N11 + 20 = C2
38 C2262 411641005347               SAB   ADD0,M2,WRK4,SB       ; SB := C2 + 4 = B2
39 C2263 034404642656               WJMP  T,BERR,INTB           ; WAIT E.T.C
40 C2264 034402000321               CAL   F,F,GETOP             ; WORK0 := FU
41 C2265 411644005347               SAB   SUN,M2,WRK4,SB        ; SB := C2 - 4 = N0
```

```
10079 GPU MICROPROGRAM  GI APR 1982
01
02                      ; NEXT COL. LOOP
03                      ; CLAIMS :  WORK0 = U; WORK4 = C2; WORK5 = C3; SB = N0;
04
05 C2266 034402002554 BFNC:  CAL    F,F,STB                 ; STORE COL IN N0
06                      ; TEST AUTORED
07 C2267 415614000255        ZAB    OR,WRK0,WRK3            ; WORK3 := U
08 C2270 411645004767        SAB    SUNC,C2,WRK5,SB         ; SB := C3 - 2 = C4
09 C2271 034404642656        WJMP   T,BERR,INTB             ; WAIT E.T.C.
10 C2272 034402000321        CAL    F,F,GETOP               ; WORK0 := AUTORED
11 C2273 03440002301         JMP    F,NZ,BFNC1              ; IF NOT_AUTORED GOTO BFNC1
12
13                      ; IF AUTORED STORE U AS LR
14 C2274 415614000367        ZAB    OR,WRK5,SB              ; SB := C3
15 C2275 415614000332        ZAB    OR,WRK3,WRK0            ; WORK0 := U
16 C2276 034402002554        CAL    F,F,STB                 ; STORE U IN C3
17 C2277 41451400000C        ZQ     OR                      ; NO OP
18 C2300 034404642656        WJMP   T,BERR,INTB             ; WAIT E.T.C.
19
20                      ; LOAD I1U AND STORE IN CAUSE
21 C2301 011644005367 BFNC1: SAB    SUN,M2,WRK5,SB          ; SB := C3 - 4 = B0
22 C2302 034402000321        CAL    F,F,GETOP               ; WORK0 := BASE I1U
23 C2303 415440000332        ABB    ADD,WRK3,WRK0           ; WORK0 := ADDR I1U
24 C2304 415614000247        ZAB    OR,WRK0,SB              ; SB := WORK0
25 C2305 034402000321        CAL    F,F,GETOP               ; WORK0 := I1U
26 C2306 015614000246        ZAB    OR,WRK0,CAUSE           ; CAUSE := I1U
27
```

```
!0080 GPU MICROPROGRAM  GI APR 1982
01
02
03
04                          ; LOAD SZU
05 C2307 011645004747               SAB   SUNO,C2,WRK4,SB        ; SB := C2 - 2 = B1
06 C2310 034404642656               WJMP  T,BERR,INTB            ; WAIT E.T.C.
07 C2311 034402000321               CAL   F,F,GETOP              ; WORK0 := BASE SZU
08 C2312 415440000332               ABB   ADD,WRK3,WRK0          ; WORK0 := ADDR SZU
09 C2313 415614000247               ZAB   OR,WRK0,SB             ; SB := WORK0
10 C2314 034402000321               CAL   F,F,GETOP              ; WORK0 := SZU
11 C2315 015614000254               ZAB   OR,WRK0,WRK2           ; WORK2 :=
12 C2316 015440000314               ABB   ADD,WRK2,WRK2          ;           2 * SZU;
13
14                          ; TEST SZU > RMAX + 1
15 C2317 411640004767               SAB   ADD,C2,WRK5,SB         ; SB := C3 + 2 = B7
16 C2320 034402000321               CAL   F,F,GETOP              ; WORK0 := RAMX + 1;
17 C2321 414445000254               AB    SUNO,WRK0,WRK2         ; ALU := SZU - (RMAX+1)
18 C2322 034400412535               JMP   T,NNEG,BFIC            ; IF SZU > (RMAX+1) GOTO BFIC
19
20                          ; STORE SZU
21 C2323 011754007407               SZB   OR,SP17,SB             ; SB := N2
22 C2324 015614000312               ZAB   OR,WRK2,WRK0           ; WRK0 := WRK2 <* SZU *>
23 C2325 034402002554               CAL   F,F,STB                ; STORE IN N2
24
25 C2326 411754006012               SZB   OR,M12,WRK0            ; WORK0 := 12 EXT 0, 12 EXT 1
26 C2327 011660007252               SAB   AND,SP16,WRK0,WRK0     ; EXTRACT BLFL?
27 C2330 034400022343               JMP   F,NZ,NBLF1             ; IF ZERO GOTO NBLF1
28                           ; TAKE EXPU <* BLOCKFLOATING EXPONENT FOR COL U *>
29 C2331 000400000024               LDIM  0000,0024              ; IMOP := 20
30 C2332 015640000367               RAB   ADD,IMOP,WRK5,SB       ; SB := C3 + 20 = B8
31 C2333 415614000177               ZAB   OR,SB,WRK5             ; WORK5 := B8
32 C2334 034404642656               WJMP  T,BERR,INTB            ; WAIT E.T.C.
33 C2335 034402000321               CAL   F,F,GETOP              ; WORK0 := EXPU BASE
34 C2336 415440000332               ABB   ADD,WRK3,WRK0          ; WORK0 := ADDR EXPU
35 C2337 415614000247               ZAB   OR,WRK0,SB             ; SB := WORK0
36 C2340 034402000321               CAL   F,F,GETOP              ; WORK0 := EXPU
37 C2341 411645004767               SAB   SUNO,C2,WRK5,SB        ; SB := B8 - 2 = N13
38 C2342 034402002554               CAL   F,F,STB                ; STORE EXPU IN N13
39
40                          ; TAKE FR
41 C2343 015614000347 NBLF1: ZAB   OR,WRK4,SB                   ; SB := C2
42 C2344 034404642656               WJMP  T,BERR,INTB            ; WAIT E.T.C.
43 C2345 034402000321               CAL   F,F,GETOP              ; WORK0 := FR
44 C2346 415614000242               ZAB   OR,WRK0,X              ; W1 := FR
45 C2347 011754007415               SZB   OR,SP17,WRK3           ; WORK3 := N2
```

```
!0081 GPU MICROPROGRAM  GI APR 1982
01
02                      ; NEXT ROW LOOP
03                      ; CLAIMS :  WORK3 = N2; SP17 = N2;
04 C2350 011754007407 BFNR:  SZB   OR,SP17,SB          ; SB := N2
05                      ; TEST R < SZU -1
06 C2351 034402000321        CAL   F,F,GETOP           ; WORK0 := SZU
07 C2352 414045000242        ABQ   SUNC,WRK0,X         ; Q := R - SZU
08 C2353 01450100000C        ZQ    ADD0                ; ALU := R - SZU + 1
09 C2354 034400012525        JMP   F,NNEG,BFIR         ; IF R < SZU -1 GOTO BFIR
10 C2355 015614000257        ZAB   OR,WRK0,WRK5        ; WORK5 := SZU
11
12                      ; LOAD I1R
13 C2356 411641005167        SAB   ADD0,M2,SB,SB       ; SB := N2 + 4 = C0
14 C2357 034402000321        CAL   F,F,GETOP           ; WORK0 := CAT_I1R_BASE
15 C2360 414040000242        ABQ   ADD,WRK0,X          ; Q := WORK0 + R <* ADDR I1R *>
16 C2361 41551400000C7       ZQB   OR,SB               ; SB := Q
17 C2362 034402000321        CAL   F,F,GETOP           ; WORK0 := I1R
18 C2363 415614000241        ZAB   OR,WRK0,WPRE        ; W2 := I1R
19
20                      ; GET SZR
21 C2364 011640004727        SAB   ADD,C2,WRK3,SB      ; SB := N2 + 2 = C1
22 C2365 034402000321        CAL   F,F,GETOP           ; WORK0 := CAT_SZR
23 C2366 414040000242        ABQ   ADD,WRK0,X          ; Q := CAT_SZR + R <* ADDR SZR *>
24 C2367 41551400000C7       ZQB   OR,SB               ; SB := ADDR SZR
25 C2370 034402000321        CAL   F,F,GETOP           ; WORK0 := SZR;
26 C2371 015440000252        ABB   ADD,WRK0,WRK0       ; WORK0 := 2 * SZR;
27
```

```
!0082 GPU MICROPROGRAM  GI APR 1982
01
02
03
04                          ; TEST SZR < SZU
05 C2372 414445000372               AB    SUNO,WRK5,WRK0          ; ALU := SZR - SZU
06 C2373 434400412375               JMP   T,NNEG,BFNR1            ; IF SZU < SZR THEN GOTO BFNR1
07 C2374 015614000372               ZAB   OR,WRK5,WRK0            ; ELSE SZR := SZU;
08
09                          ; TEST SZR > (RMAX+1)
10 C2375 015614000254 BFNR1: ZAB    OR,WRK0,WRK2            ; WRK2 := SZR;
11 C2376 011645004727               SAB   SUNO,C2,WRK3,SB         ; SB := N2 - 2 = B7
12 C2377 034402000321               CAL   F,F,GETOP               ; WORK0 := RMAX + 1
13 C2400 414445000254               AB    SUNO,WRK0,WRK2          ; ALU := SZR - (RMAX+1)
14 C2401 434400412525               JMP   T,NNEG,BFIR             ; IF SZR >= (RMAX+1) GOTO BFIR
15
16                          ; CALCULATE CP_ADDR AND STORE_ADDR
17 C2402 015440000314               ABB   ADD,WRK2,WRK2           ; SZR := SZR * 2;
18 C2403 415440000301               ABB   ADD,WRK2,WPRE           ; W2 := I1R+SZR <* ADDR_B *>
19 C2404 014214000140               ZAQ   OR,CAUSE                ; Q := I1U
20 C2405 415614000037               ZAB   OR,W1,WRK5              ; WORK5 := R
21 C2406 015440000377               ABB   ADD,WRK5,WRK5           ; WORK5 := 2 * R
22 C2407 415400000372               AQB   ADD,WRK5,WRK0           ; WORK0 := I1U + 2 * R <* STORE ADDR *>
23 C2410 004610004240               ZAS   SUB,WRK0,RTC            ; RTC := ACCU_ADDR
24 C2411 015400000316               AQB   ADD,WRK2,WRK4           ; WORK4 := I1U + SZR; <* ADDR_A *>
25
26                          ; TEST FOR FULL_RED
27 C2412 411640005567               SAB   ADD,M8,SB,SB            ; SB := B7 + 8 = N11
28 C2413 034402000321               CAL   F,F,GETOP               ; WORK0 := (RMAX+1)*4
29 C2414 414445000257               AB    SUNO,WRK0,WRK5          ; ALU := R - (RMAX+1)
30 C2415 034400412420               JMP   T,NNEG,BFI0             ; IF R >= (RMAX+1) GOTO BFI0
31 C2416 004520006400               ZQS   AND,ILIM                ; FULL_RED := -1 <* TRUE *>
32 C2417 434400002422               JMP   F,F,BFI1                ; GOTO BFI1
33                          ; NOT FULLRED
34 C2420 015614000257 BFI0:  ZAB    OR,WRK0,WRK5            ; WRK5 := (RMAX+1)
35 C2421 004444006652               ABS   SUN,WRK0,WRK0,ILIM      ; FULL_RED := 0 <* FALSE *>
36
37                          ; SET REPITION_COUNT
38 C2422 015445000317 BFI1:  ABB    SUNO,WRK2,WRK5          ; WORK5 := MAX_R - SZR;
39 C2423 034400412425               JMP   T,NNEG,BFI2             ; IF REP_COUNT * 4 < 0 THEN
40 C2424 415620000377               ZAB   AND,WRK5,WRK5           ; REP_COUNT := 0;
41 C2425 022554000017 BFI2:  SSRB   OR,WRK5,Z,NL            ; WORK5 := 4* REP_COUNT
42 C2426 022554000017               SSRB  OR,WRK5,Z,NL            ;           // 4;
```

```
!0083 GPU MICROPROGRAM  GI APR 1982
01
02                      ; CLEAR AR, BLOCKFL., ADD_SIGN, BLOCKFL_STR
03 C2427 011754007012           SZB   OR,SP16,WRK0          ; WORK0 := MODUS, ZEROEXP = 0
04 C2430 011664006252           SAB   CAND,M12,WRK0,WRK0    ;
05 C2431 034402001504           CAL   F,F,MODU1             ;
06
07                      ; MLA
08 C2432 415614000340           ZAB   OR,WRK4,W             ; W3 := ADDR_A
09 C2433 014620220240           ZAR   AND,WRK0,READP        ; I/O_ADDR := 0
10 C2434 015614000363           ZAB   OR,WRK5,GRX           ; W0 := REPCOUNT
11 C2435 434402421530           CAL   T,NZ,MLA1             ; IF W0 <> 0 GOTO MLA1
12                                                          ; AT RETURN WPRE = DIV_ADDR;
13
14                      ; SAVE DIA?
15 C2436 415451000040           ABB   SUB0,W2,W             ; W3 := ADDR_B - ADDR_A
16 C2437 014520220000           ZQR   AND,READP             ; I/O_ADDR := 0, READP
17 C2440 015614000063           ZAB   OR,W3,GRX             ; W0 := IF DIA THEN 0 ELSE W3;
18
19                      ; ADD UNRED ELEM
20
21 C2441 011754007416           SZB   OR,SP17,WRK4          ; WORK4 := N2
22                      ; LOAD EXP FOR BLOCKFLOATING NORMALIZATION
23 C2442 000400000014           LDIM  0000,0014             ; IMOP := 12
24 C2443 415640000347           RAB   ADD,IMOP,WRK4,SB      ; SB := N2 + 12 = C6
25 C2444 415614000174           ZAB   OR,SB,WRK2            ; WORK2 := C6
26 C2445 411754006012           SZB   OR,M12,WRK0           ; WORK0 := BLFL?
27 C2446 011660007252           SAB   AND,SP16,WRK0,WRK0    ;
28 C2447 434400022461           JMP   F,NZ,NBLF2            ; IF ZERO GOTO NBLF2
29 C2450 034402000321           CAL   F,F,GETOP             ; WORK0 := EXP_R_BASE
30 C2451 014154000002           ZBQ   OR,X                  ; Q := R
31 C2452 015400000247           AQB   ADD,WRK0,SB           ; SB := EXP_R ADDR
32 C2453 034402000321           CAL   F,F,GETOP             ; WORK0 := EXPR
33 C2454 415614000255           ZAB   OR,WRK0,WRK3          ; WORK3 := EXP_R
34 C2455 411640004707           SAB   ADD,C2,WRK2,SB        ; SB := C6 + 2 = N13
35 C2456 034402000321           CAL   F,F,GETOP             ; WORK0 := EXP_U
36 C2457 415440000255           ABB   ADD,WRK0,WRK3         ; WORK3 := EXP_R + EXP_U
37 C2460 034400002462           JMP   F,F,LOADT             ; GOTO LOADT
38 C2461 415620000335 NBLF2:    ZAB   AND,WRK3,WRK3         ; WORK3 := 0;
```

```
10084 GPU MICROPROGRAM  GI APR 1982
01
02
03                      ; LOAD TAILPART AND SIGN OFF ADD
04 02462 411645004707 LOADT: SAB    SUNO,C2,WRK2,SB        ; SB := C6 - 2 = C5
05 02463 034402000321        CAL    F,F,GETOP              ; WORK0 := ADDSIGN
06 02464 415614000256        ZAB    OR,WRK0,WRK4           ; WORK4 := ADDSIGN
07 02465 011645004567        SAB    SUNO,C2,SB,SB          ; SB := C5 - 2 = B5
08 02466 034402000321        CAL    F,F,GETOP              ; WORK0 := TAILDISP
09 02467 015614000257        ZAB    OR,WRK0,WRK5           ; WORK5 := TAILDISP
10 02470 011640004247        SAB    ADD,RTC,WRK0,SB        ; SB := TAIL_ADDR
11 02471 034402000633        CAL    F,F,GTDWD              ; LOAD TAILPART TO WORK0,WORK1
12 02472 415354000674        EXT    WRK1,WRK2              ; WORK2 := EXP_TAIL
13 02473 400477774000        LDIM   7777,4000              ; IMOP := -2048;
14 02474 414644000300        RA     SUN,IMOP,WRK2          ; ALU := TAIL_EXP + 2048 - 1;
15 02475 034400012545        JMP    F,NNEG,BFZA            ; IF TAIL_EXP <= -2048 GOTO BFZA;
16 02476 415445000334        ABB    SUNO,WRK3,WRK2         ; WORK2 := EXP - REDEXP;
17 02477 414644003CC         RA     SUN,IMOP,WRK2          ; ALU := TAIL_EXP +2048 -1;
18 02500 034400012545        JMP    F,NNEG,BFZA            ; IF TAILEXP <= -2048 THEN GOTO BFZA
19 02501 011664006273        SAB    CAND,M12,WRK1,WRK1     ; WORK1 := WORK1(0:11) CON 0 EXT 12
20 02502 014454150276        ABR    OR,WRK1,WRK4,FPULL     ; FPUOP1 := WRK1 CON ADDSIGN; OP = LL
21 02503 014554140012        ZBR    OR,WRK0,FPOPO          ; FPUOPU := WORK0
22                      ; LOAD HEADPART AND SUBTRACT RED_EXP (=EXP_R + EXP_U)
23 02504 411754004007        SZB    OR,RTC,SB              ; SB := HEAD_ADDR
24 02505 034402000633        CAL    F,F,GTDWD              ; WORK0,WORK1 := HEAD;
25 02506 015354000676        EXT    WRK1,WRK4              ; WORK4 := EXP
26 02507 015445000336        ABB    SUNO,WRK3,WRK4         ; WORK4 := EXP - RED_EXP
27 02510 411660006356        SAB    AND,M12,WRK4,WRK4      ; WORK4 := EXP EXTRACT 12
28 02511 011664006273        SAB    CAND,M12,WRK1,WRK1     ; WORK1 := FRAC(24:35) CON 0 EXT 12
29 02512 015454000353        ABB    OR,WRK4,WRK1           ;         + EXP
30 02513 434402001476        CAL    F,F,ADD2               ; GOTO ADD2 HEAD
31 02514 01175400400C BFZA1: SZB    OR,RTC,W               ; W3 := HEAD_ADDR_A;
32
33                      ; TEST FULL_RED
34 02515 41075400640C        SZN    OR,ILIM                ; ALU := FULL_RED
35 02516 434400022547        JMP    F,NZ,BFSTD             ; IF NOT_FULL_RED GOTO BFSTOREDOUBLE
36
37                      ; TEST DIA
38 02517 41461400000C        ZA     OR,W0                  ; ALU := DIA
39 02520 434400022563        JMP    F,NZ,BFSQ              ; IF ZERO GOTO SQUAREROOT
40
```

```
01
02
03
04                          ; DIVIDE AR
05 C2521 015614000047               ZAB   OR,W2,SB                ; SB := DIV_ADDR
06 C2522 434402001603               CAL   F,F,ARM                 ; GOTO MULT AR
07                          ; STORE IN BLOCKFLOATING MODE REDUCED ELEMENT
08 C2523 034402002057               CAL   F,F,STRS                ; GOTO STRS
09
10                          ; INCREASE R  <* ONLY CLAIM : SP17 = N2 *>
11 C2524 011754007415 BFIR0: SZB    OR,SP17,WRK3                  ; WORK3 := N2
12 C2525 011640004422 BFIR:  SAB    ADD,C2,W1,X                   ; W1 := W1 + 2
13 C2526 411644005327               SAB   SUN,M2,WRK3,SB          ; SB := N2 - 4 = C3
14 C2527 034402000321               CAL   F,F,GETOP               ; WRK0 := LR
15 C2530 014451000242               AB    SUB0,WRK0,X             ; ALU := LR - R
16 C2531 034400412350               JMP   T,NNEG,BFNR             ; IF R <= LR THEN GOTO BFNR
17
18                          ; INCREASE CCL
19                          ; CLAIMS : SB = C3; SP17 = N2;
20 C2532 415614000177 BFIC1: ZAB    OR,SB,WRK5                    ; WORK5 := C3
21 C2533 400400000032               LDIM  0000,0032               ; IMOP := 26
22 C2534 415640000336               RAB   ADD,IMOP,WRK3,WRK4      ; WORK4 := N2 + 26 = C2
23                          ; CLAIMS : WORK4 = C2; WORK5 = C3; SP17 = N2;
24 C2535 011640004747 BFIC:  SAB    ADD,C2,WRK4,SB                ; SB := C2 + 2 = B3
25 C2536 034402000321               CAL   F,F,GETOP               ; WORK0 := LU
26 C2537 014214000240               ZAQ   OR,WRK0                 ; Q := LU
27 C2540 411644005347               SAB   SUN,M2,WRK4,SB          ; SB := C2 - 4 = N0
28 C2541 034402000321               CAL   F,F,GETOP               ; WORK0 := U
29 C2542 011640004652               SAB   ADD,C2,WRK0,WRK0        ;          + 2;
30 C2543 014405000240               AQ    SUN0,WRK0               ; ALU := LU - U
31 C2544 234400412266               JMP   T,NNEG,BFNC       RTN   ; IF U <= LU GOTO BFNC ELSE RETURN
32
33
```

```
!0086 GPU MICROPROGRAM  GI APR 1982
01                      ; ZERO BLOCKNORMALIZE
02 C2545 011754002014 BFZA:  SZB   OR,MESS,WRK2          ; WORK2 := -2048;
03 C2546 034400002514        JMP   F,F,BFZA1             ; GOTO BFZA1;
04
05                      ; BF STORE DOUBLE NORMAL MODE
06 C2547 015560000012 BFSTD: ZBB   AND,WRK0              ; WORK0 := 0; <* MODUS *>
07 C2550 034402001504        CAL   F,F,MODU1             ; GOTO MODU1
08 C2551 415614000367        ZAB   OR,WRK5,SB            ; SB := TAIL_DISP
09 C2552 034402002002        CAL   F,F,STRD              ; GOTO STORE DOUBLE
10 C2553 034400002524        JMP   F,F,BFIR0             ; GOTO BFIR0
11
12                      ; STORE WORD <* STWD STORES FROM W-REG *>
13 C2554 411640000573 STB:   SAB   ADD,BASE,SB,WRK1      ; WORK1 := SB + BASE
14 C2555 41064500126C        SA    SUNC,LLIM,WRK1        ; IF WRK1 < LOWERLIMIT
15 C2556 434400012672        JMP   F,NNEG,INTO           ; THEN GOTO INTO
16 C2557 41065000166C        SA    SUB,ULIM,WRK1         ; IF WRK1 >= UPPERLIMIT
17 C2560 434400012672        JMP   F,NNEG,INTO           ; THEN GOTO INTO
18 C2561 414614030240        ZAR   OR,WRK0,DATO          ; DATA_OUT := WRK0
19 C2562 214614620260        ZAR   OR,WRK1,WRTP      RTN ; I/O_ADDR := WORK1, WRITEP, RETURN
```

```
01
02
03
04                          ; DIA
05                          ; +++
06
07                          ; X      = W1 : N3 ; NEXT ROW
08
09                          ; WORK4 = 0 EXT 12 CON URED_EXP
10                          ; SP17  = ADDR N2
11                          ; SP16  = MODUS
12                          ; W      = W3 : HEADADDR AND IN RTC
13
14                          ; THE STORE OP CALL MUST BE NOT_BLOCK_FLOAING I.E. NORMALIZING
15                          ; THE STORE OP IS CALLED WITH W = 0 AND SB = 0 (STORE IN W3W0)
16                          ; THE STORE USES WORK0, WORK1, WORK3, WORK4
17
18                          ; SET MODUS TO NORMALIZING_STORE
19 C2563 400420000CCC BFSQ:  LDIM  20C0,0000              ; MASK FOR NORMALIZED STORE
20 C2564 011754007012        SZB   OR,SP16,WRK0           ; WORK0 := MODUS
21 C2565 015660000252        RAB   AND,IMOP,WRK0,WRK0     ;       AND IMOP
22 C2566 034402001504        CAL   F,F,MODU1              ; GOTO MODUS
23
24                          ; W2W3 := AR
25 C2567 0C0400000CC6        LDIM  0000,0006              ;
26 C2570 015754000CCC        RZB   OR,IMOP,W              ; W := 6;
27 C2571 034402002057        CAL   F,F,STRS               ; GOTO STORE SINGLE
28
29                          ; SET EXP_LOSS
30 C2572 015354000756        EXT   WRK4,WRK4              ; WORK4 := UNRED_EXP
31 C2573 015354000474        EXT   W3,WRK2                ; WORK2 := RED_EXP
32 C2574 415445000316        ABB   SUNC,WRK2,WRK4         ; WORK4 := EXPLOSS
33
34                          ; SET ADDR OF STATUS
35 C2575 011754007415        SZB   OR,SP17,WRK3           ; WORK3 := N2
36 C2576 400400000026        LDIM  00C0,0026              ; IMOP := 22;
37 C2577 415640000335        RAB   ADD,IMOP,WRK3,WRK3     ; WORK3 := N2 + 22 = N0
38 C2600 011645004727        SAB   SUN0,C2,WRK3,SB        ; SB := N0 - 2 = B4
39 C2601 034402000321        CAL   F,F,GETOP              ; WORK0 := STATUSBASE
40 C2602 414040000032        ABQ   ADD,W1,WRK0            ; Q := WORK0 + ROW
41 C2603 415400000027        AQB   ADD,W1,SB              ; SB := Q + ROW
42 C2604 415614000177        ZAB   OR,SB,WRK5             ; WORK5 := STATUS_ADDR
43
```

```
10088 GPU MICROPROGRAM  GI APR 1982
01
02
03
04                          ; SET DIA_RESULT
05 C2605 011750004412               SZB   SUB,C2,WRK0          ; WORK0 := 2 -1 <* DIA_RESULT *>
06 C2606 414551000001               ZB    SUB0,WPRE            ; ALU := -RESULT
07 C2607 034400412633               JMP   T,NNEG,BFI5          ; IF -, NEG THEN GOTO BFI5
08
09                          ; STORE STATUS
10 C2610 015614000353 BFSQ1: ZAB    OR,WRK4,WRK1               ; WORK1 := EXPLOSS
11 C2611 434402002063               CAL   F,F,GPUST            ; GOTO STORE STATUS
12 C2612 411644005327               SAB   SUN,M2,WRK3,SB       ; SB := N0 - 4 = B6
13 C2613 034404642656               WJMP  T,BERR,INTB          ; WAIT E.T.C.
14
15                          ; TEST EXPLOSS
16 C2614 034402000321               CAL   F,F,GETOP            ; WORK0 := EXPLIM
17 C2615 014445000256               AB    SUN0,WRK0,WRK4       ; ALU := EXPLOSS - EXPLIM
18 C2616 034400412641               JMP   T,NNEG,BFI6          ; IF EXPLOSS >= EXPLIM GOTO BFI6
19
20 C2617 434402002125               CAL   F,F,SQRT2            ; GOTO SQRT2;
21
22                          ; SET BLOCKFLOATING MODE TO INV.
23 C2620 000430000000               LDIM  3000,0000            ; IMOP := MASK;
24 C2621 011754007012               SZB   OR,SP16,WRK0         ; WORK0 := MODUS
25 C2622 015660000252               RAB   AND,IMOP,WRK0,WRK0   ;   AND MASK
26 C2623 034402001504               CAL   F,F,MODU1            ; GOTO MODU1
27                          ; INVERT DIA AND STORE
28 C2624 415614000052               ZAB   OR,W2,WRK0           ; WORK0 := W2;
29 C2625 415614000073               ZAB   OR,W3,WRK1           ; WORK1 := W3;
30 C2626 01175400400C               SZB   OR,RTC,W             ; W3 := STORE_ADDR
31 C2627 034402001632               CAL   F,F,INV1             ; GOTO INV1 IN INV
32
33                          ; RETURN TO COL INCREASE
34 C2630 011754007415 BFSQ7: SZB    OR,SP17,WRK3               ; WORK3 := N2;
35 C2631 411644005327               SAB   SUN,M2,WRK3,SB       ; SB := N2 - 3-1 = C3;
36 C2632 434400002532               JMP   F,F,BFIC1            ; GOTO BFIC1
37
```

```
10089 GPU MICROPROGRAM  GI APR 1982
01
02
03
04                      ; SINGULAR BY LOSS OF DIGITS ; WORK5 = STATUSADDR
05                      ; SET STATUS
06 C2633 434400022636 BFI5:  JMP    F,NZ,BFSQ2          ; IF RESULT < 0 THEN
07 C2634 411754005012        SZB    OR,M2,WRK0          ; WORK0 := 3
08 C2635 434400002637        JMP    F,F,BFSQ9           ; ELSE
09 C2636 011741005012 BFSQ2: SZB    ADD0,M2,WRK0        ; WORK0 := 4;
10 C2637 415560000013 BFSQ9: ZBB    AND,WRK1            ; EXPLOSS := 0
11 C2640 034400002642        JMP    F,F,BFSQ8           ; GOTO BFSQ8
12
13                      ; EXPLOSS
14 C2641 411754005012 BFI6:  SZB    OR,M2,WRK0           ; WORK0 := 3
15
16                      ; EXCEPTIONS, CLEAR DIA_ELEM
17 C2642 415614000367 BFSQ8: ZAB    OR,WRK5,SB           ; SB := STATUS_ADDR
18 C2643 434402002063        CAL    F,F,GPUST            ; STORE STATUS
19 C2644 411754004007        SZB    OR,RTC,SB            ; SB := ADDR_A
20 C2645 015560000012        ZBB    AND,WRK0             ; WORK0 := 0
21 C2646 411754002013        SZB    OR,MESS,WRK1         ; WORK1 := 0,-2048;
22 C2647 034404642656        WJMP   T,BERR,INTB          ; WAIT E.T.C.
23 C2650 434402002063        CAL    F,F,GPUST            ; STORE IN A
24 C2651 414514000000        ZQ     OR                   ; NO OP
25 C2652 034404642656        WJMP   T,BERR,INTB          ; WAIT E.T.C.
26 C2653 034400002630        JMP    F,F,BFSQ7            ; GOTO BFSQ7
27
```

```
!0090 GPU MICROPROGRAM  GI APR 1982
01
02
03                      FILL 2656
04
05                      ;BUS INTERRUPTS
06                      ;***************
07
08                      FINT:
09                      INTB:
10 C2656 400477770010           LDIM    7777,10
11 C2657 015660000104           RAB     AND,IMOP,STAT,STAT
12 C2660 034402672665           CAL     T,BPAR,PARIT
13 C2661 434402662666           CAL     T,BNACK,REJCT
14 C2662 034402652667           CAL     T,BTIM,TIMUT
15 C2663 400404000000           LDIM    0400,0000
16 C2664 434400000132           JMP     F,F,SEND1
```

```
; FINT:   C. BUS ERROR DURING FETCH
;
;   CLEAR EX BUT BIT 20
;   IF BUSPARITY THEN PARITY
;   IF BUS NACK THEN REJECT
;   IF BUS TIMEOUT THEN TIMEOUT
; BUSERROR
;   GOTO SEND1
```

```
10091 GPU MICROPROGRAM  GI APR 1982
01                      PARIT:                                                  ; PARITY:
02 C2665 211641005104           SAB     ADDO,M2,STAT,STAT       RTN     ;   EX(21):=1   , RETURN
03                      REJCT:                                                  ; REJECT:
04 C2666 215601000104           ZAB     ADDO,STAT,STAT          RTN     ;   EX(23):=1   , RETURN
05                      TIMUT:                                                  ; TIMEOUT:
06 C2667 611640004504           SAB     ADD,C2,STAT,STAT            RTN ;   EX(22):=1
07
08 C2670 400400040000 INT2:   LDIM C004,0000          ; CODE ERROR
09 C2671 034404000132         WJMP F,F,SEND1          ; GOTO SEND1
10                      INTO:
11 C2672 400402000000         LDIM C200,0000          ; READ/WRITE OUTSIDE LIMITS
12 C2673 434400000132         JMP  F,F,SEND1          ; GOTO SEND1
13
14                      INTA:
15 C2674 400400100000         LDIM C010,0000          ; INTEGER EXCEPTION
16 C2675 034400001207         JMP  F,F,INTFA          ; GOTO INTFA;
```

```
10092 GPU MICROPROGRAM  GI APR 1982
01
02
03                      ;************************************************************
04                      ;
05                      ;       CPU 811 TCP ROUTINE
06                      ;
07                      ;************************************************************
08                      ;
09                      ;
10 C2676 034402002747 TCPIN:  CAL     F,FALSE         GCHAR                   ;WRK0:=NEXTCHAR
11 C2677 400400000103         LDIM    0,"C
12 C2700 014651000240         RA      SUBO    IMOP,WRK0                       ;IF CHAR = 'C' THEN
13 C2701 434400422705         JMP     T,NZ            TCP1
14 C2702 400400000001         LDIM    0,1
15 C2703 015754000012         RZB     OR      IMOP,WRK0                       ;WRK0:=1
16 C2704 434400002706         JMP     F,FALSE ,.+6                            ;ELSE
17 C2705 434402002720 TCP1:   CAL     F,FALSE         FORMAT                  ;WRK0:=(WRK0+5)&17;
18 C2706 115614000255         ZAB     OR      WRK0,WRK3       PUSH            ;WRK3:=WRK0
19 C2707 034402002747 TCP2:   CAL     F,FALSE         GCHAR                   ;WHILE CHAR<>'SP' AND CHAR<>'CR' DO
20 C2710 400400000040         LDIM    0,40                                    ;BEGIN
21 C2711 014651000240         RA      SUBO    IMOP,WRK0                       ;IF CHAR ='SP' THEN
22 C2712 034400022724         JMP     F,NZ    ,EMUL                           ;GOTO EMUL
23 C2713 400400000015         LDIM    0,15
24 C2714 014651000240         RA      SUBO    IMOP,WRK0                       ;IF CHAR ='CR' THEN GOTO EMUL;
25 C2715 034400022724         JMP     F,NZ            EMUL
26 C2716 434402002720         CAL     F,FALSE         FORMAT
27 C2717 655471400255         ABBR    31      WRK0,WRK3,40    LRTN            ;WRK3:=WRK3 EXOR WRK0;
28                                                                            ;END;
29
30 C2720 000400000005 FORMAT: LDIM    0,5
31 C2721 415640000252         RAB     ADD     IMOP,WRK0,WRK0                  ;WRK0:=WRK0+5;
32 C2722 000400000017         LDIM    0,17
33 C2723 615660000252         RAB     AND     IMOP,WRK0,WRK0          RTN     ;WREK0:=WRK0 AND17, RETURN.
34
35 C2724 140400002727 EMUL:   LDIM    0,CASEB/3                       POP     ;GOTO THE PROPER
36 C2725 015640070335         RABR    ADD     IMOP,WRK3,WRK3,MIX              ;INTERPRETER ROUTINE.
37 C2726 034401000000         JMX     F,FALSE                                 ;MIX:=CASEBASE+WRK1, JMP &MIX;
```

```
1CC93 GPU MICROPROGRAM  GI APR 1982
01 C2727 434400003021 CASEB:  JMP     F,F             TCPER                   ;?, GOTO TCPER
02 C2730 034400003155         JMP     F,F             C                       ;C, CONTINUE.
03 C2731 434400003115         JMP     F,F             LN                      ;LN
04 C2732 034400003111         JMP     F,F             XLM                     ;LM
05 C2733 034400003334         JMP     F,F             T.O                     ;O
06 C2734 03440000315C5        JMP     F,F             XSUBR                   ;XS, EXECUTE SUBROUTINE
07 C2735 034400003073         JMP     F,F             LR                      ;LR
08 C2736 43440000302          JMP     F,F             R                       ;R
09 C2737 034400003153         JMP     F,F             S                       ;S, SINGLE INSTRUCTION
10 C2740 03440000334C         JMP     F,F             TX                      ;T
11 C2741 034400003064         JMP     F,F             XR                      ;XR
12 C2742 434400003021         JMP     F,F             TCPER                   ;?, GOTO TCPER
13 C2743 434400003021         JMP     F,F             TCPER                   ;?, GOTO TCPER
14 C2744 434400003021         JMP     F,F             TCPER                   ;?, GOTO TCPER
15 C2745 034400003133         JMP     F,F             XN                      ;XN
16 C2746 034400003127         JMP     F,F             XM                      ;XM
17
18                    ;CHARACTOR INPUT ROUTINE.
19                    ;
20                    ;THE ROUTINE READ A 8 BIT CHARACTOR FROM TCP AND MASK
21                    ;OUT THE 7 LEAST SIGNIFICANT BIT. THE RESULT IS LOADED
22                    ;TO WRKO AND RETURNED TO TCPOUT.
23                    ; AUX. REG:  WRKO,1
24                    ;
25 C2747 400400000004 GCHAR:  LDIM    O,4
26 C2750 434400712750         JMP     T,NTPIN ,.                              ;IF NOT TCPIN READY THEN WAIT.
27 C2751 015654040231         RABR    OR      IMOP,CNTR,CNTR,CNTRO            ;SET TCPINACK
28 C2752 400400000177         LDIM    O,177
29 C2753 034400312753         JMP     F,NTPIN,.                               ;IF TCPINRDY THEN WAIT;
30 C2754 015754002412         RZB     OR      TPIN,WRKO                       ;WRKO:=XX,TCPIN, THE CHAR RDY IN RE
31 C2755 015660000252         RAB     AND     IMOP,WRKO,WRKO                  ;WRKO:=WRKO&177
32 C2756 400400000004         LDIM    O,4
33 C2757 015664040231         RABR    CAND    IMOP,CNTR,CNTR,CNTRO            ;CLEAR TCPIN _ACK.
34 C2760 000400000033         LDIM    O,33
35 C2761 01465100024C         RA      SUBO    IMOP,WRKO                       ;TEST FOR ESC
36 C2762 034400023021         JMP     F,NZ            TCPER                   ;IF ESC THEN GOTO TCPERROR;
37 C2763 415614000253         ZAB     OR      WRKO,WRK1                       ;WRK1:=CHAR
38 C2764 434400002765         JMP     F,FALSE         TYPE                    ;TYPE THE CHAR AT TCP, RETURN
39
```

```
10094 GPU MICROPROGRAM  GI APR 1982
01                      ;TYPE ROUTINE.
02                      ;
03                      ;THE ROUTINE TYPE THE 7 BIT CHAR IN WRK1.
04                      ;CR (<15>) IS TYPED AS CR LF .(<15><12>).
05                      ;
06 C2765 034400322765 TYPE:   JMP     F,TPACK,.                                   ;IF -,TPOUTACK THEN WAIT;
07 C2766 000400000377         LDIM    0,377
08 C2767 015660000273         RAB     AND     IMOP,WRK1,WRK1                      ;MASK OUT BIT (16:23)
09 C2770 01461405026C         ZAR     OR      WRK1,TPOUT                          ;TPOUT REG:=CHAR
10 C2771 400400000002         LDIM    0,2
11 C2772 015654040231         RABR    OR      IMOP,CNTR,CNTR,CNTRO                ;SET TPOUT_READY
12 C2773 434400402773         JMP     T,FALSE ,.                                  ;NO_OP.
13 C2774 015664040231         RABR    CAND    IMOP,CNTR,CNTR,CNTRO                ;CLEAR TPOUT_RDY.
14 C2775 400400000015         LDIM    0,15
15 C2776 41465100026C         RA      SUBO    IMOP,WRK1                           ;TEST FOR CR
16 C2777 6344000230CC         JMP     F,NZ            ,.+3            RTN         ;IF NON_ZERO THEN RETURN
17 C3000 000400000012         LDIM    0,12
18 C3001 415754000013         RZB     OR      IMOP,WRK1                           ;WRK1:=LF.
19 C3002 434400002765         JMP     F,FALSE         TYPE                        ;GOTO TYPE;
20
21
22                      ;OCTAL NUMBER INPUT ROUTINE.
23                      ;
24                      ;THE ROUTINE READ A OCTAL NUMBER FROM TCP TO WRK2.
25                      ;THE NUMBER IS TERMINATED BY 'CR' OR ':'.
26                      ;THE ROUTINE TEST FOR OVERFLOW.
27                      ;AUX REG: WRK0,1,2
28
29 C3003 415620000314 GTNUM:  ZAB     AND     WRK2,WRK2                           ;WRK2:=0
30 C3004 034402002747 GTNM1:  CAL     F,FALSE         GCHAR                       ;WRK0:=NEXTCHAR
31 C3005 400400000015         LDIM    0,15
32 C3006 01461100024C         RA      SUBO    IMOP,WRKC                           ;TEST FOR 'CR'
33 C3007 634400423010         JMP     T,NZ            ,.+3        RTN             ;IF CR THEN RETURN;
34 C3010 000400000072         LDIM    C,":
35 03011 01461100024C         RA      SUBO    IMOP,WRKC                           ;TEST FOR ':'
36 C3012 634400423013         JMP     T,NZ            .+3         RTN             ;IF ':' THEN RETURN;
37 C3013 4C0400000C07         LDIM    0,7
38 C3014 015660000252         RAB     AND     IMOP,WRKC,WRK0                      ;WRK0:=WRKC&7;
39 C3015 023440000314         SSLA    ADD     WRK2,WRK2,Z,NL                      ;WRK2:=WRK2 SHIFT 2;
40 C3016 423554000014         SSLB    OR      WRK2,Z,NL                           ;WRK2:=WRK2 SHIFT 1;
41 C3017 015454000254         ABB     OR      WRK0,WRK2                           ;WRK2:=WRK2 OR WRK0
42 C3020 034400003004         JMP     F,FALSE         GTNM1                       ;REPEAT.
43
```

```
!0095 GPU MICROPROGRAM  GI APR 1982
01 C3021 4C0417600015 TCPER:  LDIM    "?*16.,15                 CONT          ;COMMAND ERROR.
02 C3022 034402003027         CAL     F,FALSE         TPTXT                   ;TYPE '?<15>', RETURN.
03 C3023 03440000001C         JMP     F,F             IDLE                    ;RETURN TO IDLELOOP
04
05                    ;TYPE CR-LF.
06                    ;AUX REG: WRK1
07 C3024 400400000015 CRLF:   LDIM    0,15                                    ;ROUTINE TO TYPE CR_LF.
08 C3025 415754000013         RZB     OR      IMOP,WRK1
09 C3026 434400002765         JMP     F,FALSE         TYPE                    ;TYPE <15><12>, RETURN.
10
11                    ;TYPE TEXT.
12                    ;THE ROUTINE TYPE THE 3 CHARACTORS IN IMOP REG.
13                    ;AUX REG: WRK0,1,2 AND Q.
14
15 C3027 01435400000C TPTXT:  RZQ     OR      IMOP                            ;Q:=IMOP.
16 C3030 000400000003         LDIM    0,3
17 C3031 015754000012         RZB     OR      IMOP,WRKC                       ;WRK0:=3, COUNTER.
18 C3032 400400000010 TPTX1:  LDIM    C,8.
19 C3033 015754000014         RZB     OR      IMOP,WRK2                       ;WRK2:=8., COUNTER.
20 C3034 423154000013 TPTX2:  DSLB    OR      WRK1,Z,NL                       ;(WRK1,Q):=(WRK1,Q) SHIFT 1;
21 C3035 415604000314         ZAB     SUN     WRK2,WRK2                       ;WRK2:=WRK2-1;
22 C3036 034400423034         JMP     T,NZ            TPTX2                   ;IF NON ZERO THE REPEAT;
23 C3037 034402002765         CAL     F,FALSE         TYPE                    ;TYPE THE CHAR
24 C3040 415604000252         ZAB     SUN     WRK0,WRKC                       ;WRK0:=WRK0-1
25 C3041 634400423032         JMP     T,NZ            TPTX1   RTN             ;IF ZERO THEN RETURN ELSE REPEAT;
26
27
28                    ;TYPE BINARY NUMBER.
29
30                    ;THE ROUTINE TYPE THE BINARY NUMBER IN Q - REG AS 8
31                    ;OCTAL DIGITS, TERMINATED BY CRLF.
32                    ;AUX REG: WRK0,1 AND Q.
33
34 C3042 034402003047 TPNUM:  CAL     F,F,TPOCT                       ;TYPE THE CONTENTS OF Q AS 8 OCTAL DIGITS
35 C3043 434400003024         JMP     F,FALSE         CRLF            ;TYPE CRLF, RETURN;
36
```

```
10096 GPU MICROPROGRAM  GI APR 1982
01                      ;TYPE ADDRESS
02
03                      ;       THE ROUTINE TYPES THE BINARY NUMBER IN THE Q-REGISTER
04                      ;       AS 8 OCTAL DIGITS, TERMINATED BY : SP SP
05                      ;       AUX REG: WRK0, WRK1, WRK2, Q, IMOP
06
07 03044 034402003047 TYPAD:  CAL     F,F,TPOCT                       ;TYPE (Q) AS 8 OCTAL DIGITS
08 03045 000416420040         LDIM    1642,0040
09 03046 434400003027         JMP     F,F,TPTXT                       ;TYPE <:><SP><SP> ON TCP, RTN
10
11
12                      ;TYPE OCTAL NUMBER SUBROUTINE
13
14                      ;       THE ROUTINE TYPES THE CONTENTS
15                      ;       OF THE Q-REGISTER AS 8 OCTAL DIGITS
16
17 03047 400400000010 TPOCT:  LDIM    0,10
18 03050 015754000012         RZB     OR,IMOP,WRK0                    ;WRK0:=8., COUNTER
19 03051 415620000273 TPOC1:  ZAB     AND,WRK1,WRK1                   ;WRK1:=0
20 03052 423154000013         DSLB    OR,WRK1,Z,NL                    ;(WRK1,Q):=(WRK1,Q) SHIFT 1
21 03053 423154000013         DSLB    OR,WRK1,Z,NL
22 03054 423154000013         DSLB    OR,WRK1,Z,NL
23 03055 000400000060         LDIM    0,60
24 03056 415640000273         RAB     ADD,IMOP,WRK1,WRK1              ;WRK1:=60+WRK1, ASCII VALUE
25 03057 034402002765         CAL     F,F,TYPE                        ;TYPE THE CHAR AT TCP
26 03060 415604000252         ZAB     SUN,WRK0,WRK0                   ;WRK0:=WRK0-1
27 03061 634400423051         JMP     T,NZ,TPOC1      RTN             ;IF ZERO THEN RTN ELSE REPEAT
28
29
30                      ;TYPE <SP><SP> SUBROUTINE
31
32                      ;       AUX. REG: WRK0, WRK1, WRK2, Q, IMOP
33
34 03062 000410020000 SPSP:   LDIM    1002,0                          ;IMOP:=<SP><SP>
35 03063 434400003027         JMP     F,F,TPTXT                       ;TYPE <SP><SP>, RETURN
```

```
!0097 GPU MICROPROGRAM  GI APR 1982
01                      ;EXAMINE REGISTER
02
03                      ;       DISPLAYS THE CONTENTS OF THE ADDRESSED REGISTER AS 8
04                      ;       OCTAL DIGITS ON THE TCP DISPLAY
05
06 C3064 034402003003 XR:       CAL     F,F,GTNUM                       ;WRK2:=ADDRESS OF REGISTER
07 C3065 400440000000           LDIM    4000,0
08 C3066 015664000315           RAB     CAND,IMOP,WRK2,WRK3             ;WRK3:=WRK2 & 37777777
09 C3067 434402003062           CAL     F,F,SPSP                        ;TYPE <SP><SP>
10 C3070 434402003205           CAL     F,F,GETRG                       ;Q:=CONTENTS OF REGISTER
11 C3071 034402003042           CAL     F,F,TPNUM                       ;DISPLAY Q ON TCP
12 C3072 034400000010           JMP     F,F,IDLE                        ;GOTO IDLE LOOP
13
14
15                      ;LOAD REGISTER
16
17                      ;       THE ADDRESSED REGISTER IS LOADED WITH THE OCTAL NUMBER
18                      ;       TYPED ON THE TCP
19
20 C3073 034402003003 LR:       CAL     F,F,GTNUM                       ;WRK2:=ADDRESS OF REGISTER
21 C3074 415614000315           ZAB     OR,WRK2,WRK3                    ;WRK3:=WRK2
22 C3075 434402003062           CAL     F,F,SPSP                        ;TYPE <SP><SP>
23 C3076 034402003003           CAL     F,F,GTNUM                       ;WRK2:=NEW CONTENTS OF REGISTER
24 C3077 014214003000           ZAQ     OR,WRK2                         ;Q:=WRK2
25 C3100 034402003252           CAL     F,F,PUTRG                       ;REGISTER(WRK3):=Q
26 C3101 034400000010           JMP     F,F,IDLE                        ;GOTO IDLE LOOP
27
28
29                      ;RUN
30
31                      ;       STARTS MICROINSTRUCTION EXECUTION FROM THE ADDRESSED
32                      ;       CONTROL STORE LOCATION
33
34 C3102 034402003003 R:        CAL     F,F,GTNUM                       ;WRK2:=ADDRESS
35 C3103 014614070300           ZAR     OR,WRK2,MIX                     ;MIX:=ADDRESS
36 C3104 034401000000           JMX     F,F                             ;GOTO (MIX)
37
38
```

```
10098 GPU MICROPROGRAM  GI APR 1982
01                      ;EXECUTE SUBROUTINE, XS
02
03                      ;       EXECUTES THE ADDRESSED SUBROUTINE
04                      ;       AND RETURNS TO IDLE LOOP
05
06 C3105 034402003003 XSUBR:  CAL     F,F,GTNUM                       ;WRK2:=ADDRESS
07 C3106 014614070300         ZAR     OR,WRK2,MIX                     ;MIX:=ADDRESS
08 C3107 43440300000C         CAX     F,F                             ;CALL (MIX)
09 C3110 034400000010         JMP     F,F,IDLE                        ;GOTO IDLE LOOP
10
11
12                      ;LOAD MEMORY
13
14                      ;       LOADS THE ADDRESSED MEMORY LOCATION WITH THE OCTAL NUMBER
15                      ;       TYPED ON THE TCP
16                      ;       THE ADDRESS IS SAVED IN WRK5
17
18 C3111 034402003003 XLM:    CAL     F,F,GTNUM                       ;WRK2:=ADDRESS
19 C3112 015614000317         ZAB     OR,WRK2,WRK5                    ;WRK5:=ADDRESS
20 C3113 434402003062         CAL     F,F,SPSP                        ;TYPE <SP><SP>
21 C3114 034400003121         JMP     F,F,LDMEM                       ;GOTO LDMEM
22
23
24                      ;LOAD NEXT MEMORY
25
26                      ;       LOADS THE MEMORY LOCATION ADDRESSED BY WRK5+2
27                      ;       WITH THE OCTAL NUMBER TYPED ONTHE TCP
28                      ;       THE ADDRESS IS SAVED IN WRK5
29
30 C3115 400400000002 LN:     LDIM    0,2
31 C3116 415640000377         RAB     ADD,IMOP,WRK5,WRK5              ;WRK5:=WRK5+2
32 C3117 014214000360         ZAQ     OR,WRK5                         ;Q:=WRK5
33 C3120 034402003044         CAL     F,F,TYPAD                       ;DISPLAY ADDRESS
34 C3121 034402003003 LDMEM:  CAL     F,F,GTNUM                       ;WRK2:=DATA
35 C3122 414614030300         ZAR     OR,WRK2,DATO                    ;DATA OUT:=DATA
36 C3123 414614420360         ZAR     OR,WRK5,WRT                     ;I/O ADDR:=WRK5, WRITE
37 C3124 014114000000         ZQQ     OR                              ;NO-OPERATION
38 C3125 434404643021         WJMP    T,BERR,TCPER                    ;IF BUS ERROR THEN GOTO TCPER
39 C3126 034400000010         JMP     F,F,IDLE                        ;GOTO IDLE LOOP
```

```
!0099 GPU MICROPROGRAM  GI APR 1982
01                      ;EXAMINE MEMORY
02
03                      ;       DISPLAYS THE CONTENTS OF THE ADDRESSED MEMORY LOCATION
04                      ;       NOTE THAT ADDRESSES 0,2,4,6 REFERS TO MEMORY LOCATIONS 0,2,4,6
05                      ;       THE ADDRESS IS SAVED IN WRK5
06
07
08 C3127 034402003003 XM:       CAL     F,F,GTNUM                       ;WRK2:=ADDRESS
09 C3130 015614000317           ZAB     OR,WRK2,WRK5                    ;WRK5:=ADDRESS
10 C3131 034402003146           CAL     F,F,XMEM                        ;DISPLAY ADDRESS AND CONTENTS
11 C3132 034400000010           JMP     F,F,IDLE                        ;GOTO IDLE LOOP
12
13
14                      ;EXAMINE NEXT
15
16                      ;       DISPLAYS THE CONTENTS OF A NUMBER OF CONSECUTIVE MEMORY LOCATIONS
17                      ;       STARTING AT THE LOCATION ADDRESSED BY (WRK5)+2.
18                      ;       THE LAST ADDRESS IS SAVED IN WRK5.
19
20 C3133 034402003003 XN:       CAL     F,F,GTNUM                       ;WRK2:=NO. OF LOCATIONS
21 C3134 400400000002           LDIM    0,2
22 C3135 415754000015           RZB     OR,IMOP,WRK3                    ;WRK3:=+2
23 C3136 415614000316           ZAB     OR,WRK2,WRK4                    ;WRK4:=WRK2
24 C3137 434400020010           JMP     F,NZ,IDLE                       ;IF WRK2=0 THEN GOTO IDLE
25 C3140 415440000337 XNEXT:    ABB     ADD,WRK3,WRK5                   ;WRK5:=WRK5+WRK3
26 C3141 034402003146           CAL     F,F,XMEM                        ;DISPLAY ADDRESS AND CONTENTS
27 C3142 415544000016           ZBB     SUN,WRK4                        ;WRK4:=WRK4-1
28 C3143 434400020010           JMP     F,NZ,IDLE                       ;IF WRK4=0 THEN GOTO IDLE
29 C3144 034400312676           JMP     F,NTPIN,TCPIN                   ;IF TCP INP. RDY. THEN GOTO TCPIN
30 C3145 434400003140           JMP     F,F,XNEXT                       ;GOTO XNEXT
```

```
!01CC GPU MICROPROGRAM  GI APR 1982
01                      ;DISPLAY MEMORY ADDRESS AND CONTENTS SUBROUTINE
02
03                      ;       DISPLAYS THE ADDRESS AND THE CONTENTS OF THE MEMORY
04                      ;       LOCATION ADDRESSED BY WRK5
05
06 03146 414214020360 XMEM:   ZAQR    OR,WRK5,READ                    ;I/O ADDR:=Q:=WRK5, READ
07 03147 034402003044         CAL     F,F,TYPAD                       ;DISPLAY ADDRESS
08 03150 434404643021         WJMP    T,BERR,TCPER                    ;IF BUS ERROR THEN GOTO TCPER
09 03151 014354001400         RZQ     OR,DATI                         ;Q:=DATA IN
10 03152 434400003042         JMP     F,F,TPNUM                       ;DISPLAY CONTENTS OF MEM. LOCATION, RTN
11
12
13                      ;SINGLE INSTRUCTION
14
15                      ;       EXECUTES THE INSTRUCTION ADDRESSED BY PC AND RETURNS TO IDLE LOOP.
16                      ;       THE CONTENTS OF W0, W1, W2, W3, STAT, IC, CAUSE, AND SB
17                      ;       IS DISPLAYED
18
19 03153 400400000020 S:      LDIM    0,20
20 03154 015654040231         RABR    OR,IMOP,CNTR,CNTR,CNTRO         ;SINGLE INSTR:=1
21
22
23                      ;CONTINUE
24
25                      ;       STARTS INSTRUCTION EXECUTION IN THE LOCATION ADDRESSED BY PC
26
27 03155 434404000241 C:      WJMP    F,F,MLOOP                       ;IF I/O READY THEN GOTO MAIN LOOP
```

```
!01C1 GPU MICROPROGRAM  GI APR 1982
01                      ;SINGLE INSTRUCTION INTERRUPT
02
03 C3156 000400000120 SINGL:  LDIM    0,120
04 C3157 015664040231         RABR    CAND,IMOP,CNTR,CNTR,CNTRO       ;RUN:=SINGLE INSTR:=0
05 C3160 000400003175         LDIM    0,RGTXT/3
06 C3161 415754000016         RZB     OR,IMOP,WRK4                    ;WRK4:=RGTXT
07 C3162 115560000015         ZBB     AND,WRK3          PUSH          ;WRK3:=0, PUSH
08 C3163 014440070355         ABR     ADD,WRK4,WRK3,MIX               ;MIX:=WRK3+WRK4
09 C3164 434030000CC0         CAX     F,F                             ;IMOP:=REG. NAME
10 C3165 034402003027         CAL     F,F,TPTXT                       ;TYPE <REG NAME><:>
11 C3166 434402003062         CAL     F,F,SPSP                        ;TYPE <SP><SP>
12 C3167 434402003205         CAL     F,F,GETRG                       ;Q:=CONTENTS OF REG
13 C3170 034402003042         CAL     F,F,TPNUM                       ;TYPE CONTENTS OF REG
14 C3171 415541000015         ZBB     ADDO,WRK3                       ;WRK3:=WRK3+1
15 C3172 400400000007         LDIM    0,7
16 C3173 414651000320         RA      SUBO,IMOP,WRK3                  ;IF WRK3>7
17 C3174 674400010C1C         JMP     F,NNEG,IDLE       LRTN          ;THEN GOTO IDLE ELSE DISPLAY NEXT
18
19
20 C3175 200425630072 RGTXT:  LDIM    2563,0072         RTN           ;IMOP:= <W0:>
21 C3176 600425630472         LDIM    2563,0472         RTN           ;IMOP:= <W1:>
22 C3177 600425631072         LDIM    2563,1072         RTN           ;IMOP:= <W2:>
23 C3200 200425631472         LDIM    2563,1472         RTN           ;IMOP:= <W3:>
24 C3201 200424652072         LDIM    2465,2072         RTN           ;IMOP:= <ST:>
25 C3202 600422241472         LDIM    2224,1472         RTN           ;IMOP:= <IC:>
26 C3203 200420651472         LDIM    2065,1472         RTN           ;IMOP:= <CS:>
27 C3204 600424641072         LDIM    2464,1072         RTN           ;IMOP:= <SB:>
```

```
!01C2 GPU MICROPROGRAM  GI APR 1982
01                    ;GET REGISTER SUBROUTINE
02
03                    ;       THE SUBROUTINE LOADS Q WITH THE CONTENTS OF THE REGISTER
04                    ;       ADDRESSED BY WRK3. FOR NON EXISTING REGISTERS Q:=-1
05
06 C3205 400400000062 GETRG:  LDIM    0,62
07 C3206 414645000320         RA      SUNO,IMOP,WRK3                  ;WRK3-62
08 C3207 034400023246         JMP     F,NZ,GTRTC                      ;IF WRK3=62 THEN GOTO GTRTC
09 C3210 000400000060         LDIM    0,60
10 C3211 414645000320         RA      SUNO,IMOP,WRK3                  ;IF WRK3=60
11 C3212 434400023244         JMP     F,NZ,GSP16                      ;THEN GOTO GSP16
12 C3213 014644000320         RA      SUN,IMOP,WRK3                   ;IF WRK3=61
13 C3214 034400023245         JMP     F,NZ,GSP17                      ;THEN GOTO GSP17
14 C3215 400400000020         LDIM    0,20
15 C3216 014644000320         RA      SUN,IMOP,WRK3                   ;WRK3-20-1
16 C3217 434400413251         JMP     T,NNEG,DEFLT                    ;IF WRK3>20 THEN GOTO DEFLT
17 C3220 000400003223         LDIM    0,GRTAB/3
18 C3221 014640070320         RAR     ADD,IMOP,WRK3,MIX               ;MIX:=WRK3+GET REG TABLE BASE
19 C3222 034401000000         JMX     F,F                             ;GOTO (MIX)
20
21 C3223 614214000000 GRTAB:  ZAQ     OR,W0           RTN             ;ADDR= 0, Q:=W0
22 C3224 214214000020         ZAQ     OR,W1           RTN             ;      1, Q:=W1
23 C3225 214214000040         ZAQ     OR,W2           RTN             ;      2, Q:=W2
24 C3226 614214000060         ZAQ     OR,W3           RTN             ;      3, Q:=W3
25 C3227 214214000100         ZAQ     OR,STAT         RTN             ;      4, Q:=STAT
26 C3230 614214000120         ZAQ     OR,IC           RTN             ;      5, Q:=IC
27 C3231 614214000140         ZAQ     OR,CAUSE        RTN             ;      6, Q:=CAUSE
28 C3232 214214000160         ZAQ     OR,SB           RTN             ;      7, Q:=SB
29
30 C3233 210354000000         SZQ     OR,CPA          RTN             ;ADDR=10, Q:=CPA
31 C3234 610354000400         SZQ     OR,BASE         RTN             ;     11, Q:=BASE
32 C3235 610354001000         SZQ     OR,LLIM         RTN             ;     12, Q:=LLIM
33 C3236 210354001400         SZQ     OR,ULIM         RTN             ;     13, Q:=ULIM
34 C3237 610354002000         SZQ     OR,MESS         RTN             ;     14, Q:=MESS
35 C3240 210354002400         SZQ     OR,CLOW         RTN             ;     15, Q:=CLOW
36 C3241 214214000200         ZAQ     OR,PC           RTN             ;     16, Q:=PC
37 C3242 210354003000         SZQ     OR,CTOP         RTN             ;     17, Q:=CTOP
38 C3243 610354003400         SZQ     OR,CTADDR               RTN             ;     20, Q:=CTADDR
39
```

```
!01C3 GPU MICROPROGRAM  GI APR 1982
01 C3244 6103540070CC GSP16:  SZQ     OR,SP16          RTN            ;ADDR=60, Q:=SP16
02 C3245 2103540074CC GSP17:  SZQ     OR,SP17          RTN            ;ADDR=61, Q:=SP17
03
04 C3246 0103540040CC GTRTC:  SZQ     OR,RTC                          ;ADDR=62, Q:=RTC
05 C3247 0C0400177777         LDIM    17,7777
06 C3250 61432000000C         RQQ     AND,IMOP         RTN            ;Q:=8 EXT 0 CON RTC(8:23)
07
08 C3251 214050000252 DEFLT:  ABQ     SUB,WRK0,WRK0    RTN            ;UNKNOWN ADDR, Q:=-1
```

```
!0104 GPU MICROPROGRAM  GI APR 1982
01                    ;PUT REGISTER SUBROUTINE
02
03                    ;       THE SUBROUTINE LOADS THE REGISTER ADDRESSED BY WRK3
04                    ;       WITH THE CONTENTS OF THE Q-REGISTER
05
06 03252 400400000057 PUTRG:  LDIM    0,57
07 03253 414645000320         RA      SUN0,IMOP,WRK3                  ;WRK3-57
08 03254 034400023312         JMP     F,NZ,PUT57                      ;IF WRK3=57 THEN GOTO PUT57
09 03255 014644000320         RA      SUN,IMOP,WRK3                   ;IF WRK3=60
10 03256 434400023313         JMP     F,NZ,PUT60                      ;THEN GOTO PUT60
11 03257 000400000022         LDIM    0,22
12 03260 014644000320         RA      SUN,IMOP,WRK3                   ;WRK3-22-1
13 03261 634400013262         JMP     F,NNEG,.+3      RTN             ;IF WRK3>22 THEN RETURN
14 03262 415354001334         SWAP    WRK3,WRK2                       ;WRK2:=WRK3(12:23) CON WRK3(0:11)
15 03263 414614060300         ZAR     OR,WRK2,IR                      ;PREPARE INDIRECT ADDRESSING
16                                                                    ;OF W(0:3) VIA X-FIELD OF IR
17 03264 000400003267         LDIM    0,PRTAB/3
18 03265 014640070320         RAR     ADD,IMOP,WRK3,MIX               ;MIX:=WRK3+PUT REG TABLE BASE
19 03266 034401000000         JMX     F,F                             ;GOTO (MIX)
20
21 03267 215514000002 PRTAB:  ZQB     OR,X            RTN             ;ADDR= 0, W0:=Q
22 03270 215514000002         ZQB     OR,X            RTN             ;       1, W1:=Q
23 03271 215514000002         ZQB     OR,X            RTN             ;       2, W2:=Q
24 03272 215514000002         ZQB     OR,X            RTN             ;       3, W3:=Q
25 03273 215514000004         ZQB     OR,STAT         RTN             ;       4, STAT:=Q
26 03274 615514000005         ZQB     OR,IC           RTN             ;       5, IC:=Q
27 03275 615514000006         ZQB     OR,CAUSE        RTN             ;       6, CAUSE:=Q
28 03276 215514000007         ZQB     OR,SB           RTN             ;       7, SB:=Q
29
30 03277 204510000300         ZQS     SUB,CPA         RTN             ;ADDR=10, CPA:=Q
31 03300 604510000400         ZQS     SUB,BASE        RTN             ;      11, BASE:=Q
32 03301 604510001000         ZQS     SUB,LLIM        RTN             ;      12, LLIM:=Q
33 03302 204510001400         ZQS     SUB,ULIM        RTN             ;      13, ULIM:=Q
34 03303 604510002000         ZQS     SUB,MESS        RTN             ;      14, MESS:=Q
35 03304 204510002400         ZQS     SUB,CLOW                RTN             ;      15, CLOW:=Q
36 03305 215514000010         ZQB     OR,PC           RTN             ;      16, PC:=Q
37 03306 204510003000         ZQS     SUB,CTOP        RTN             ;      17, CTOP:=Q
38 03307 604510003400         ZQS     SUB,CTADDR      RTN             ;      20, CTADDR:=Q
39 03310 604510007000         ZQS     SUB,SP16        RTN             ;      21, SP16:=Q
40 03311 204510007400         ZQS     SUB,SP17        RTN             ;      22, SP17:=Q
41
```

```
!01C5 GPU MICROPROGRAM  GI APR 1982
01 C3312 614514100CCC PUT57:  ZQR     OR,INTRG        RTN             ;CLEAR INTERRUPT BIT(Q)
02
03 C3313 434400003042 PUT60:  JMP     F,F,TPNUM                       ;DISPLAY Q ON TCP
04
```

```
01
02                      ;*************************************************************
03                      ;
04                      ;       CPU 811 MICRO TEST - 791018/FK
05                      ;
06                      ;*************************************************************
07
08
09 C3314 034402003024 T1.1:   CAL   F,F,CRLF   ; TYPE NL
10 C3315 000461062566         LDIM  6106,2566 ;
11 C3316 034402003027         CAL   F,F,TPTXT ; TYPE  DEV
12 C3317 000463467440         LDIM  6346,7440 ; TYPE  NO_
13 C3320 034402003027         CAL   F,F,TPTXT ;
14 C3321 000416420040         LDIM  1642,0040 ; TYPE  :__
15 C3322 034402003027         CAL   F,F,TPTXT ;
16 C3323 034402003003         CAL   F,F,GTNUM ; WRK2 := DEV_NO INPUT ON TCP
17 C3324 400440000000         LDIM  4000,0000 ;
18 C3325 415654000314         RAB   CR,IMOP,WRK2,WRK2 ; WORK2 := CPUADDR (AS SWITCHES IN POS XXX)
19 C3326 004610004300         ZAS   SUB,WRK2,10 ; SP10 := CPU_ADDR
20
21 C3327 000450000000 T1:     LDIM    5000,0                          ;RUN CPU TEST
22 C3330 015754010004         RZBR    OR      IMOP,STAT,CPUST         ;SET FULL MEM TEST FLAG
23 C3331 034400003472         JMP     F,FALSE         CPUTEST
24
25 C3332 034402003347 T4:     CAL     F,FALSE         INITA           ;CLEAR INTERRUPTS, INIT REGISTERS.
26 C3333 434400002676         JMP     F,FALSE         TCPIN           ;GOTO TCP INPUT ROUTINE
27
28 C3334 034402003003 T.0:    CAL     F,FALSE         GTNUM           ;WRK2:=NUMBER
29 C3335 011654016504         SABR    OR      15,STAT,STAT,CPUST      ;START TEST IN &WRK2,LOOP MODE
30 C3336 014614070300         ZAR     OR      WRK2,MIX
31 C3337 034401000000         JMX     F,FALSE                         ;JMP TO LOOP
32
33 C3340 034402003003 TX:     CAL     F,F     GTNUM                   ;START TEST IN WRK2
34 C3341 014604000300       ZA    SUN,WRK2          ; ALU := WRK2 - 1
35 C3342 034400023314       JMP   F,NZ,T1.1         ; IF ZERO GOTO T1.1
36 C3343 400400000004       LDIM  C000,0004         ;
37 C3344 014651000300       RA    SUB0,IMOP,WRK2    ; ALU := WRK2 - 4
38 C3345 434400023332       JMP   F,NZ,T4           ; IF ZERO GOTO T4
39 C3346 434400003021       JMP   F,F,TCPER         ; GOTO TCP ERROR
40
```

```
!01C7 GPU MICROPROGRAM  GI APR 1982
01 C3347 034402000231 INITA:  CAL     F,F,CINTR                          ;CLEAR ALL INTERRUPTS
02 C3350 400400000040 INIT:   LDIM    0,40                               ;SUBROUTINT TO INITIALIZE REG
03 C3351 415754040011         RZBR    OR      IMOP,CNTR,CNTRO            ;DIG.OUT REG INITIALIZED, AUTOL-LAMP SET.
04 C3352 400404000CCC         LDIM    400,0
05 C3353 014354000CCC         RZQ     OR      IMOP                       ;Q:=IMOP
06 C3354 404510005CCC         ZQS     SUB     12                         ;SP12:=ERROR FLAG - AFESC FF
07 C3355 000452525252         LDIM    5252,5252
08 C3356 014354000CCC         RZQ     OR      IMOP
09 C3357 004510005400         ZQS     SUB     13                         ;SP13:=DATA PATTERN,SHORT MEM TEST
10 C3360 414050000252         ABQ     SUB     WRK0,WRKC                  ;Q:=-1, WRK0 DUMMY.
11 C3361 404504006CCC         ZQS     SUN     14                         ;SP14:=1,TEST SYNC FLAG
12 C3362 400402000CCC         LDIM    200,0
13 C3363 014354000CCC         RZQ     OR      IMOP
14 C3364 004510006400         ZQS     SUB     15                         ;SP15:=LOOP FLAG - IMSK FF
15 C3365 400400000010         LDIM    0,10
16 C3366 014354000CCC         RZQ     OR      IMOP
17 C3367 004510007CCC         ZQS     SUB     SP16                       ;SP16:=8., START PC
18 C3370 400400000002         LDIM    0,2
19 C3371 014354000CCC         RZQ     OR      IMOP                       ;Q:= 2
20 C3372 404510004400         ZQS     SUB     C2                         ;C2:=2
21 C3373 215620000377         ZAB     AND     WRK5,WRK5       RTN        ;WRK5:= 0, RETURN
22
23
24
25 C3374 411664046231 LOOP:   SABR    CAND    14,CNTR,CNTR,CNTRO         ;CLEAR TEST CYNC
26 C3375 574400143377         JMP     F,IMSK          LOOP1   POP        ;IF -,LOOP THEN GOTO LOOP1 ELSE POP
27 C3376 674400703400         JMP     T,INTR          TINTR   LRTN       ;IF INTERRUPT THEN GOTO INTERRUPT ELSE LOOP
28 C3377 234400703400 LOOP1:  JMP     T,INTR          TINTR   RTN        ;IF INTERRUPT THEN GOTO INTERRUPT ELSE SUB-
29
```

```
!01C8 GPU MICROPROGRAM  GI APR 1982
01                      TINTR:                                          ;INTERRUPT ROUTINE
02 C3400 415754102013           RZBR    OR      ILEV,WRK1,INTRG         ;WRK1:=INTERRUPT LEVEL, CLEAR INTERRUPT.
03 C3401 000400000017           LDIM    0000,0017
04 C3402 015660000273           RAB     AND     IMOP,WRK1,WRK1          ;WRK1:=WRK1&17
05 C3403 000400000005           LDIM    0,5
06 C3404 414651000260           RA      SUBO    IMOP,WRK1               ;TEST FOR RTC INTERRUPT
07 C3405 434400423410           JMP     T,NZ    TINT1                   ;IF -,RTC THEN GOTO TINT1;
08 C3406 634400543407           JMP     T,IMSK          .+3     RTN     ;IF LOOPMODE THEN CONT ELSE RTN;
09 C3407 274400403407           JMP     T,F     .               LRTN    ;LOOP_RETURN
10 C3410 434402003456 TINT1:    CAL     F,F             SAVE            ;SAVE WRK0,1,2 AND Q IN SP 0-3;
11 C3411 011664016504           SABR    CAND    15,STAT,STAT,CPUST      ;CLEAR LOOP FLAG
12 C3412 034400312676           JMP     F,NTPIN         TCPIN           ;IF TCPIN THEN GOTO TCP ROUTINE
13 C3413 434400330003           JMP     F,RSTRT         POWUP           ;IF AUTOL THEN GOTO AUTOLENTRY
14 C3414 415450000252           ABB     SUB     WRK0,WRK0               ;ILLEGAL INTERRUPT OCCURED, WRK0:=-1
15 C3415 411664015104           SABR    CAND    12,STAT,STAT,CPUST      ;CLEAR ERROR FLAG
16                                                                      ;GOTO MAIN ERROR ROUTINE.
17
18
19                      TERROR:         ;MAIN ERROR ROUTINE
20
21 C3416 234400133417           JMP     F,AFESC         .+3     RTN     ;IF ERROR_FLAG THEN RETURN;
22 C3417 434402003456           CAL     F,F             SAVE            ;SAVE WRK0,1,2 AND Q IN SP 0-3;
23 C3420 400421251122           LDIM    2125,1122                       ;TYPE AT TCP: ERR<15><12>
24 C3421 034402003027           CAL     F,FALSE         TPTXT
25 C3422 034402003024           CAL     F,F             CRLF
26 C3423 410354000000           SZQ     OR      CPA
27 C3424 034402003047           CAL     F,FALSE         TPOCT           ;WRK0
28 C3425 034402003024           CAL     F,F             CRLF
29 C3426 010354000400           SZQ     OR      BASE
30 C3427 034402003047           CAL     F,FALSE         TPOCT           ;WRK1
31 C3430 034402003024           CAL     F,F             CRLF
32 C3431 010354001000           SZQ     OR      LLIM
33 C3432 034402003047           CAL     F,FALSE         TPOCT           ;WRK2
34 C3433 034402003024           CAL     F,F             CRLF
35 C3434 414214000200           ZAQ     OR      PC
36 C3435 034402003047           CAL     F,FALSE         TPOCT           ;PC
37 C3436 034402003024           CAL     F,FALSE         CRLF            ;
38
```

```
!C1C9 GPU MICROPROGRAM  GI APR 1982
01 C3437 C34402002747          CAL     F,FALSE         GCHAR           ;WAIT FOR TCP INPUT
02 C3440 0C0400000120          LDIM    C,"P
03 C3441 41425100024C          RAQ     SUBO    IMOP,WRKC               ;TEST FOR 'P', PROCEED
04 C3442 034400023451          JMP     F,NZ            PROC            ;IF ZERO THEN GOTO PROC.
05 C3443 0C0400000116          LDIM    C,"N
06 C3444 41425100024C          RAQ     SUBO    IMOP,WRKC               ;TEST FOR 'N', LOOPING
07 C3445 574400023447          JMP     F,NZ            NLOOP   POP     ;IF ZERO THEN GOTO NLOOP
08 C3446 034400002677          JMP     F,FALSE         TCPIN+3         ;GOTO MAIN TCP ROUTINE, 1.CHAR IN WRK0 !
09
10 C3447 01165401650C4 NLOOP:  SABR    OR      15,STAT,STAT,CPUST      ;SET LOOP FLAG
11 C3450 41165401510C4         SABR    OR      12,STAT,STAT,CPUST      ;SET ERROR FLAG.
12 C3451 034402003024  PROC:   CAL     F,FALSE         CRLF            ;TYPE CRLF
13 C3452 411754000012          SZB     OR      CPA,WRKC                ;RESTORE WRK0,1,2 AND Q REG
14 C3453 411754000413          SZB     OR      BASE,WRK1
15 C3454 011754001014          SZB     OR      LLIM,WRK2
16 C3455 21035400140C          SZQ     OR      ULIM            RTN     ;RETURN
17
18
19 C3456 404610000240  SAVE:   ZAS     SUB     WRK0,CPA                ;SAVE WRK0,1,2 AND Q
20 C3457 404610000660          ZAS     SUB     WRK1,BASE               ;WRK0 -> CPA
21 C3460 004610001300          ZAS     SUB     WRK2,LLIM               ;WRK1 -> BASE
22 C3461 204510001400          ZQS     SUB     ULIM            RTN     ;WRK2 -> LLIM
23                                                                     ;Q    -> ULIM
24
25
```

```
10110 GPU MICROPROGRAM  GI APR 1982
01 C3462 400477777776 TBERR:  LDIM    7777,7776                       ;BUSERROR
02 C3463 015754000012         RZB     OR       IMOP,WRKC              ;WRK0:=-2
03 C3464 415620000273         ZAB     AND      WRK1,WRK1              ;WRK1:=0
04 C3465 434400653416         JMP     T,BTIM            TERROR       ;IF BUSTIMEOUT THEN 0 ELSE
05 C3466 415601000273         ZAB     ADDO     WRK1,WRK1              ;IF BUSNACK THEN 1 ELSE
06 C3467 434400663416         JMP     T,BNACK           TERROR       ;IF BUSPARITY THEN 2;
07 C3470 415601000273         ZAB     ADDO     WRK1,WRK1
08 C3471 634400673416         JMP     T,BPAR            TERROR   RTN
```

```
!0111 GPU MICROPROGRAM  GI APR 1982
01
02                      CPUTEST:                ;START OF CPU TEST
03 C3472 034402003347           CAL     F,F             INITA           ;CLEAR INTERRUPT, INIT REGS
04 C3473 400400023420           LDIM    2,3420
05 C3474 415754000015           RZB     OR      IMOP,WRK3
06 C3475 004610007720           ZAS     SUB     WRK3,SP17               ;SP17:=10000. CPU TEST RUN COUNTER.
07 C3476 400400000007           LDIM    0,7
08 C3477 434402153027           CAL     F,FPMSK         TPTXT           ;IF TCP ON THEN RING THE BELL AT TCP;
09
10                      B10:
11
12                      ;TEST OF IMOP.
13                      ;--------------
14                      ;THE LOOP INCLUDE TEST OF:
15                      ;STUCK_AT_ZERO, STUCK_AT_ONE AND BRIDGE FAILURES IN
16                      ;THE IMMEDIATE OPERAND REGISTER.
17                      ;
18 C3500 011754007012 B20:      SZB     OR      SP16,WRK0               ;WRK0:= 10 , STATUS
19 C3501 111664015104           SABR    CAND    12,STAT,STAT,CPUST  PUSH;SETUP, CLEAR ERROR FLAG
20 C3502 411654046231           SABR    OR      14,CNTR,CNTR,CNTRO      ;SET TEST SYNC
21 C3503 000452525252           LDIM    5252,5252
22 C3504 415754000013           RZB     OR      IMOP,WRK1               ;WRK1:= 1010...10
23 C3505 000425252525           LDIM    2525,2525
24 C3506 415754000010           RZB     OR      IMOP,PC                 ;PC:=010101...01
25 C3507 415450000314           ABB     SUB     WRK2,WRK2               ;WRK2:=-1, EXPECTED DATA
26 C3510 004474003670           ABS     EXNOR   WRK1,PC,CTADDR          ;CTADDR:= -,(WRK1 EXOR PC)
27 C3511 010651003700           SA      SUB0    CTADDR,WRK2             ;IF RESULT <> -1 THEN ERROR;
28 C3512 034402423416           CAL     T,NZ            TERROR          ;EXAMINE MONT (REG 8'20) FOR THE RESULT
29 C3513 034402003374           CAL     F,F             LOOP            ;CALL LOOP ADM
```

```
!0112 GPU MICROPROGRAM  GI APR 1982
01                      ;
02                      ;    TEST OF THE ALU REGISTER AND SCRATCHPAD FILES.
03                      ;-------------------------------------------------
04                      ;DATA PATTERN: A ONE IS SHIFTED THROUGH BOTH REGISTER FILES.
05                      ;THE ALU REG STACK IS READ BOTH FROM THE A- AND THE B FILES.
06                      ;NOTE:  WRKC AND STAT (SEE TEST OF COND GRUP 1) IS NOT TESTED.
07                      ;
08                      ;ERROR INFORMATION:
09                      ;   WRK0:   100-236      ;SEE THE REGBASE TABLE
10                      ;   WRK1:   THE SELECTED REG'S CONTENTS
11                      ;   WRK2:   EXPECTED DATA PATTERN
12                      ;
13 C3514 400400000CC            LDIM    0,100
14 C3515 015754000012           RZB     OR      IMOP,WRKC               ;WRK0:=100, STATUS AND POINTER IN REG TABLE
15
16 C3516 400400000C1 B100:      LDIM    0,1
17 C3517 015754000014           RZB     OR      IMOP,WRK2               ;WRK2:= 1, THE START DATA PATTRN.
18 C3520 400400003444 B110:     LDIM    0,REGBASE/3-100
19 C3521 414640070240           RAR     ADD     IMOP,WRKC,MIX           ;MIX -> SELECTED REG.
20 C3522 1116640151C4           SABR    CAND    12,STAT,STAT,CPUST  PUSH;SETUP, CLEAR ERROR FLAG
21 C3523 411654046231           SABR    OR      14,CNTR,CNTR,CNTRO      ;SET TEST SYNC
22 C3524 400440010004           LDIM    4001,0004                       ;PREPARE INDIRECT ADDRESSING OF W0,W1,W3 AN
23 C3525 414754060CCC           RZR     OR      IMOP,IR                 ;VIA W-FIELD,X-FIELD, WPRE AND BIT(21:22)
24 C3526 014754620CCC           RZR     OR      IMOP,WRTP               ;OR INSTR. REG AND I/O ADDR. REG;
25 C3527 434403000CCC           CAX     F,F                             ;SEL.REG:=WRK2, Q:=SEL.REG
26 C3530 415514000013           ZQB     OR      WRK1                    ;WRK1:=READ DATA
27 C3531 014451000313           AB      SUBO    WRK2,WRK1               ;CHECK
28 C3532 034402423542           CAL     T,NZ            REGERR          ;IF RESULT<> ZERO THEN ERROR;
29 C3533 03440200335C           CAL     F,F             INIT            ;INIT THE REG.
30 C3534 034402003374           CAL     F,F             LOOP
31 C3535 015440000314           ABB     ADD     WRK2,WRK2               ;WRK2:= WRK2 SHIFT 1;
32 C3536 03440042352C           JMP     T,NZ            B110            ;IF WRK2 <> 0 THEN GOTO B110;
33 C3537 C15541000012 B120:     ZBB     ADDO    WRK0
34 C3540 015541000012           ZBB     ADDO    WRK0                    ;WRK0:=WRK0+2
35 C3541 034400003516           JMP     F,F             B100            ;REPEAT WITH A NEW REGISTER
36
37 C3542 03440200335C REGERR:   CAL     F,F             INIT            ;REGISTER ERROR ROUTINE, INIT REG
38 C3543 434400003416           JMP     F,F             TERROR          ;GOTO MAIN ERROR ROUTINE.
```

```
!0113 GPU MICROPROGRAM  GI APR 1982
01                      REGBASE:                                        ;   WRK0
02                                                                      ;-------------
03 C3544 61421400003CC          ZAQ     OR      WRK2            RTN     ;100
04 C3545 41451400000CC          ZQ      OR                              ;(NOP)
05 C3546 01561400003CC          ZAB     OR      WRK2,W                  ;102
06 C3547 61421400000CC          ZAQ     OR      W0              RTN
07 C3550 01561400003CC          ZAB     OR      WRK2,W                  ;104
08 C3551 21415400000CC          ZBQ     OR      W0              RTN
09 C3552 41561400003C2          ZAB     OR      WRK2,X                  ;106
10 C3553 21421400002C           ZAQ     OR      W1              RTN
11 C3554 41561400003C2          ZAB     OR      WRK2,X                  ;110
12 C3555 61415400000C2          ZBQ     OR      X               RTN
13 C3556 01561400003C3          ZAB     OR      WRK2,GRX                ;112
14 C3557 21421400004C           ZAQ     OR      W2              RTN
15 C3560 01561400003C3          ZAB     OR      WRK2,GRX                ;114
16 C3561 21415400000C3          ZBQ     OR      GRX             RTN
17 C3562 41561400003C1          ZAB     OR      WRK2,WPRE               ;116
18 C3563 61421400006C           ZAQ     OR      W3              RTN
19 C3564 41561400003C1          ZAB     OR      WRK2,WPRE               ;120
20 C3565 61415400000C1          ZBQ     OR      WPRE            RTN
21 C3566 01561400003C5          ZAB     OR      WRK2,IC                 ;122
22 C3567 61421400012C           ZAQ     OR      IC              RTN
23 C3570 01561400003C5          ZAB     OR      WRK2,IC                 ;124
24 C3571 21415400000C5          ZBQ     OR      IC              RTN
25 C3572 01561400003C6          ZAB     OR      WRK2,CAUSE              ;126
26 C3573 61421400014C           ZAQ     OR      CAUSE           RTN
27 C3574 01561400003C6          ZAB     OR      WRK2,CAUSE              ;130
28 C3575 21415400000C6          ZBQ     OR      CAUSE           RTN
29 C3576 41561400003C7          ZAB     OR      WRK2,SB                 ;132
30 C3577 21421400016C           ZAQ     OR      SB              RTN
31 C3600 41561400003C7          ZAB     OR      WRK2,SB                 ;134
32 C3601 61415400000C7          ZBQ     OR      SB              RTN
33 C3602 41561400031C           ZAB     OR      WRK2,PC                 ;136
34 C3603 21421400020C           ZAQ     OR      PC              RTN
35 C3604 41561400031C           ZAB     OR      WRK2,PC                 ;140
36 C3605 61415400001C           ZBQ     OR      PC              RTN
37 C3606 01561400C311           ZAB     OR      WRK2,CNTR               ;142
38 C3607 61421400022C           ZAQ     OR      CNTR            RTN
```

```
10114 GPU MICROPROGRAM  GI APR 1982
01 C3610 015614000311          ZAB     OR      WRK2,CNTR               ;144
02 C3611 214154000011          ZBQ     OR      CNTR            RTN
03 C3612 415614000313          ZAB     OR      WRK2,WRK1               ;146
04 C3613 21421400026C          ZAQ     OR      WRK1            RTN
05 C3614 415614000313          ZAB     OR      WRK2,WRK1               ;150
06 C3615 614154000013          ZBQ     OR      WRK1            RTN
07 C3616 015614000314          ZAB     OR      WRK2,14                 ;152
08 C3617 61421400030C          ZAQ     OR      14              RTN
09 C3620 015614000314          ZAB     OR      WRK2,14                 ;154
10 C3621 214154000014          ZBQ     OR      14              RTN
11 C3622 415614000315          ZAB     OR      WRK2,15                 ;156
12 C3623 21421400032C          ZAQ     OR      15              RTN
13 C3624 415614000315          ZAB     OR      WRK2,15                 ;160
14 C3625 614154000015          ZBQ     OR      15              RTN
15 C3626 415614000316          ZAB     OR      WRK2,16                 ;162
16 C3627 21421400034C          ZAQ     OR      16              RTN
17 C3630 415614000316          ZAB     OR      WRK2,16                 ;164
18 C3631 614154000016          ZBQ     OR      16              RTN
19 C3632 015614000317          ZAB     OR      WRK2,17                 ;164
20 C3633 61421400036C          ZAQ     OR      17              RTN
21 C3634 015614000317          ZAB     OR      WRK2,17                 ;170
22 C3635 214154000017          ZBQ     OR      17              RTN
23 C3636 15175400400C          SZB     OR,10,6   POP ; SAVE DEVNO IN RAM6
24 C3637 034400003537          JMP     F,F,B120      ; RETURN
25 C3640 551754007405          SZB     OR      SP17,IC         POP     ;SAVE RUN COUNTER IN IC
26 C3641 034400003537          JMP     F,F             B120            ;RETURN
27 C3642 40461000030C          ZAS     SUB     WRK2,0                  ;176
28 C3643 21035400000C          SZQ     OR      0               RTN
29 C3644 00461000070C          ZAS     SUB     WRK2,1                  ;200
30 C3645 61035400040C          SZQ     OR      1               RTN
31 C3646 00461000130C          ZAS     SUB     WRK2,2                  ;202
32 C3647 61035400100C          SZQ     OR      2               RTN
33 C3650 40461000170C          ZAS     SUB     WRK2,3                  ;204
34 C3651 21035400140C          SZQ     OR      3               RTN
35 C3652 00461000230C          ZAS     SUB     WRK2,4                  ;206
36 C3653 61035400200C          SZQ     OR      4               RTN
37 C3654 40461000270C          ZAS     SUB     WRK2,5                  ;210
38 C3655 21035400240C          SZQ     OR      5               RTN
39 C3656 40461000330C          ZAS     SUB     WRK2,6                  ;212
40 C3657 21035400300C          SZQ     OR      6               RTN
41 C3660 00461000370C          ZAS     SUB     WRK2,7                  ;214
42 C3661 61035400340C          SZQ     OR      7               RTN
```

```
!0115 GPU MICROPROGRAM  GI APR 1982
01 C3662 CC461000430C          ZAS     SUB     WRK2,10                 ;216
02 C3663 610354004CCC          SZQ     OR      10              RTN
03 C3664 4C461000470C          ZAS     SUB     WRK2,11                 ;220
04 C3665 210354004C0C          SZQ     OR      11              RTN
05 C3666 4C461000530C          ZAS     SUB     WRK2,12                 ;222
06 C3667 210354005CCC          SZQ     OR      12              RTN
07 C3670 0C461000570C          ZAS     SUB     WRK2,13                 ;224
08 C3671 610354005CCC          SZQ     OR      13              RTN
09 C3672 4C461000630C          ZAS     SUB     WRK2,14                 ;226
10 C3673 210354006CCC          SZQ     OR      14              RTN
11 C3674 0C461000670C          ZAS     SUB     WRK2,15                 ;230
12 C3675 610354006CCC          SZQ     OR      15              RTN
13 C3676 0C461000730C          ZAS     SUB     WRK2,16                 ;232
14 C3677 610354007CCC          SZQ     OR      16              RTN
15 C3700 4C461000770C          ZAS     SUB     WRK2,17                 ;234
16 C3701 210354007CCC          SZQ     OR      17              RTN
17 C3702 144610004140          ZAS     SUB,6,10  POP ;  RESTORE DEV_NO
18 C3703 034400003537          JMP     F,F,B120      ; RETURN
19 C3704 544610007520          ZAS     SUB     IC,SP17       POP       ;RESTORE RUN COUNTER
20 C3705 034402003350          CAL     F,F             INIT            ;INIT REG.
21                                                                     ;FINIS GOTO NEXT TEST:
```

```
!0116 GPU MICROPROGRAM  GI APR 1982
01
02                      ;TEST OF EXTERNT INTERRUPT.
03                      ;-----------------------------------------
04                      ;THE TEST CHECK THAT ILEV 0-7 NOT SET INTERRUPT.
05                      ;IT IS VERIFYED THAT ILEVEL 8-15 SET THE PROPER INTERRUPT
06                      ;LEVEL.THE TEST ALSO CHECK THAT THE PROPER ILEVEL IS ABLE TO
07                      ;CLEAR THE INTERRUPT.
08                      ;
09 C3706 411754004010 B200:   SZB OR,10,PC ; PC := CPU_ADDR
10                                                          ;THE LOOP CHECK THAT EXTERNT INTERRUPT
11                                                          ;LEVEL 0-7 NOT IS ABLE TO SET THE
12                                                          ;INTERRUPT FLAG.
13 C3707 415450000314         ABB     SUB     WRK2,WRK2             ;WRK2:=-1, ILEVEL COUNTER
14 C3710 415601030314 B250:   ZABR    ADDO    WRK2,WRK2,DATO        ;DATO:=0,1,..7,WRK2:=WRK2+1;
15 C3711 111664015104         SABR    CAND    12,STAT,STAT,CPUST  PUSH;SETUP, CLEAR ERROR FLAG.
16 C3712 411654046231         SABR    OR      14,CNTR,CNTR,CNTRO    ;SET TEST SYNC
17 C3713 014614420200         ZAR     OR      PC,WRT                ;SIMULATE EXTERNT INTERRUPT
18 C3714 400400000301         LDIM    0,301
19 C3715 015754000012         RZB     OR      IMOP,WRK0             ;WRK0:= 301, STATUS
20 C3716 434406663462         WCAL    T,BNACK         TBERR         ;IF BNACK THEN GOTO TBERR
21 C3717 034402703771         CAL     T,INTR          CHINT         ;IF ILLEGALT INTERRUPT THEN GOTO ERROR;
22 C3720 034402003374         CAL     F,F             LOOP
23
24 C3721 550650007300         SA      SUB     SP16,WRK2       POP   ;RESULT:=8.-WRK2-1
25 C3722 434400423710         JMP     T,NZ            B250          ;IF WRK2 <> 7 THEN GOTO B250;
26
```

```
!0117 GPU MICROPROGRAM  GI APR 1982
01                    B260:                                                  ;TEST THAT ILEVEL 8-15 CAUSE INTERRUPT.
02                                                                           ;THE ILEV-REG IS ALSO TESTED, AND IT IS CHEC
03                                                                           ;THE PROPER LEVEL CLEAR THE INTERRUPT;
04 C3723 415601030314          ZABR    ADDO    WRK2,WRK2,DATO                ;DATO:=8,9,..15
05 C3724 111664015104          SABR    CAND    12,STAT,STAT,CPUST  PUSH;SETUP, CLEAR ERROR FLAG;
06 C3725 411654046231          SABR    OR      14,CNTR,CNTR,CNTRO            ;SET TEST SYNC
07 C3726 014614420200          ZAR     OR      PC,WRT                        ;SIMULATE EXTERNT INTERRUPT, LEVEL IN WRK2
08 C3727 400400000310          LDIM    C,310
09 C3730 015754000012          RZB     OR      IMOP,WRKC                     ;WRKO:= 310, STATUS.
10 C3731 434406653416          WCAL    T,BTIM          TERROR                ;WAIT, IF BUSTIMEOUT THEN ERROR;
11 C3732 434402643462          CAL     T,BERR          TBERR                 ;IF BUSERROR THE GOTO BUSERROR;
12 C3733 015541000012          ZBB     ADDO    WRKO                          ;WRKO:= 311
13 C3734 000400000005          LDIM    C,5
14 C3735 014754100000          RZR     OR      IMOP,INTRG                    ;CLEAR RTC INTERRUPT.
15 C3736 014114000000          ZQQ     OR                                    ;NO-OPERATION
16 C3737 014114000000          ZQQ     OR                                    ;NO-OPERATION
17 C3740 014114000000          ZQQ     OR                                    ;NO-OPERATION
18 C3741 034402303416          CAL     F,INTR          TERROR                ;IF -,INTERRUPT THEN ERROR;
19 C3742 000400000005          LDIM    C,5
20 C3743 014754100000          RZR     OR      IMOP,INTRG                    ;CLEAR RTC INTERRUPT.
21 C3744 000400000377          LDIM    C,377                                 ;NO-OPERATION/LDIM:= 377
22 C3745 015541000012          ZBB     ADDO    WRKO                          ;WRKO:=312, STATUS.
23 C3746 415754000013          RZB     OR      IMOP,WRK1                     ;WRK1:=INTERRUPT LEVEL
24 C3747 415660002273          RAB     AND     ILEV,WRK1,WRK1                ;WRK1:= WRK1 EXTRACT 8
25 C3750 014451000274          AB      SUBO    WRK1,WRK2
26 C3751 034402423416          CAL     T,NZ            TERROR                ;IF OCCURED<>EXPECTED INTERRUPT LEVEL THEN E
27 C3752 015541000012          ZBB     ADDO    WRKO                          ;WRKO:= 313, STATUS
28 C3753 014614100300          ZAR     OR      WRK2,INTRG                    ;CLEAR INTERRUPT
29 C3754 014114000000          ZQQ     OR                                    ;NO-OPERATION
30 C3755 014114000000          ZQQ     OR                                    ;NO-OPERATION
31 C3756 014114000000          ZQQ     OR                                    ;NO-OPERATION.
32 C3757 034402703771          CAL     T,INTR          CHINT                 ;IF ILLEGALT INTERRUPT THEN ERROR;
33 C3760 034402003374          CAL     F,F             LOOP
34
35 C3761 000400000017          LDIM    C,15.
36 C3762 014651000300          RA      SUBO    IMOP,WRK2                     ;IF WRK2 <> 15. THEN GOTO B260;
37 C3763 434400423723          JMP     T,NZ            B260
38
39
```

```
!0118 GPU MICROPROGRAM  GI APR 1982
01                      ;PASS ADMINI.
02                      ;--------------------
03 C3764 011750007414           SZB     SUB     SP17,WRK2               ;IF PASS COUNTER = 0 THEN TERMINATE CPU TEST
04 C3765 434400423767           JMP     T,NZ            B400
05 C3766 434400003327           JMP     F,F             T1              ; ONLY CPU TEST
06 C3767 40461000770C B400:     ZAS     SUB     WRK2,SP17               ;SP17:=SP17-1;
07 C3770 43440000350C           JMP     F,F             B10             ;START A NEW PASS
08
09
10                      ;
11                      ;   SUBROUTINER
12                      ;
13
14 C3771 015754002013 CHINT:    RZB     OR      ILEV,WRK1               ;IF ILEVEL < 8. THEN RTN ELSE
15 C3772 000400000017           LDIM    0000,0017
16 C3773 015660000273           RAB     AND     IMOP,WRK1,WRK1          ;GOTO ERROR;
17 C3774 01065000726C           SA      SUB     SP16,WRK1
18 C3775 634400013416           JMP     F,NNEG          TERROR  RTN     ;
19
```

```
!0119 GPU MICROPROGRAM  GI APR 1982
01              000002          .DO 2048.-(./3)
02                              .END
03
0002 SOURCE LINES IN ERROR
```

```
 00C1 .MAIN DOMUS MACRO ASSEMBLER REV 02.00
01          000000 .LOC 0
02                 ;ADDRESS TABLE
03                 ;**************
04
05
06                 ;       ADDRESS CALCULATION
07                 ;       INPUT: AFTER ESCAPE, AFTER AM, REL, INDIR, X-FIELD(0:1)
08
09          000004 .DO 4                                    !00C3 .MAIN
10  CC000 000265 DIR/3                                     01 CC100 000265 DIR/3
11  CC001 000272 INDEX/3                                   02 CC101 000272 INDEX/3
12  CC002 000272 INDEX/3                                   03 CC102 000272 INDEX/3
13  CC003 000272 INDEX/3                                   04 CC103 000272 INDEX/3
14                                                         05
15  CC004 000265 DIR/3                                     06 CC104 000265 DIR/3
16  CC005 000272 INDEX/3                                   07 CC105 000272 INDEX/3
17  CC006 000272 INDEX/3                                   08 CC106 000272 INDEX/3
18  CC007 000272 INDEX/3                                   09 CC107 000272 INDEX/3
19                                                         10
20  CC010 000267 REL/3              ;REL                   11 CC110 000267 REL/3              ;REL
21  CC011 000275 RELX/3                                    12 CC111 000275 RELX/3
22  CC012 000275 RELX/3                                    13 CC112 000275 RELX/3
23  CC013 000275 RELX/3                                    14 CC113 000275 RELX/3
24                                                         15
25  CC014 000267 REL/3                                     16 CC114 000267 REL/3
26  CC015 000275 RELX/3                                    17 CC115 000275 RELX/3
27  CC016 000275 RELX/3                                    18 CC116 000275 RELX/3
28  CC017 000275 RELX/3                                    19 CC117 000275 RELX/3
29                                                         20
30  CC020 000301 AMD/3              ;    AFTER AM          21 CC120 000301 AMD/3              ;    AFTER AM
31  CC021 000306 AMX/3                                     22 CC121 000306 AMX/3
32  CC022 000306 AMX/3                                     23 CC122 000306 AMX/3
33  CC023 000306 AMX/3                                     24 CC123 000306 AMX/3
34  CC024 000301 AMD/3                                     25 CC124 000301 AMD/3
35  CC025 000306 AMX/3                                     26 CC125 000306 AMX/3
36  CC026 000306 AMX/3                                     27 CC126 000306 AMX/3
37  CC027 000306 AMX/3                                     28 CC127 000306 AMX/3
38                                                         29
39  CC030 000300 AMR/3              ;REL, AFTER AM         30 CC130 000300 AMR/3              ;REL, AFTER AM
40  CC031 000305 AMRX/3                                    31 CC131 000305 AMRX/3
41  CC032 000305 AMRX/3                                    32 CC132 000305 AMRX/3
42  CC033 000305 AMRX/3                                    33 CC133 000305 AMRX/3
43  CC034 000300 AMR/3                                     34 CC134 000300 AMR/3
44  CC035 000305 AMRX/3                                    35 CC135 000305 AMRX/3
45  CC036 000305 AMRX/3                                    36 CC136 000305 AMRX/3
```

```
!OCC2 .MAIN
01 CC040 000265 DIR/3
02 CC041 000272 INDEX/3
03 CC042 000272 INDEX/3
04 CC043 000272 INDEX/3
05
06 CC044 000265 DIR/3
07 CC045 000272 INDEX/3
08 CC046 000272 INDEX/3
09 CC047 000272 INDEX/3
10
11 CC050 000267 REL/3          ;REL
12 CC051 000275 RELX/3
13 CC052 000275 RELX/3
14 CC053 000275 RELX/3
15
16 CC054 000267 REL/3
17 CC055 000275 RELX/3
18 CC056 000275 RELX/3
19 CC057 000275 RELX/3
20
21 CC060 000301 AMD/3          ;    AFTER AM
22 CC061 000306 AMX/3
23 CC062 000306 AMX/3
24 CC063 000306 AMX/3
25 CC064 000301 AMD/3
26 CC065 000306 AMX/3
27 CC066 000306 AMX/3
28 CC067 000306 AMX/3
29
30 CC070 000300 AMR/3          ;REL, AFTER AM
31 CC071 000305 AMRX/3
32 CC072 000305 AMRX/3
33 CC073 000305 AMRX/3
34 CC074 000300 AMR/3
35 CC075 000305 AMRX/3
36 CC076 000305 AMRX/3
37 CC077 000305 AMRX/3
```

```
!OCC4 .MAIN
01 CC140 000265 DIR/3
02 CC141 000272 INDEX/3
03 CC142 000272 INDEX/3
04 CC143 000272 INDEX/3
05
06 CC144 000265 DIR/3
07 CC145 000272 INDEX/3
08 CC146 000272 INDEX/3
09 CC147 000272 INDEX/3
10
11 CC150 000267 REL/3          ;REL
12 CC151 000275 RELX/3
13 CC152 000275 RELX/3
14 CC153 000275 RELX/3
15
16 CC154 000267 REL/3
17 CC155 000275 RELX/3
18 CC156 000275 RELX/3
19 CC157 000275 RELX/3
20
21 CC160 000301 AMD/3          ;    AFTER AM
22 CC161 000306 AMX/3
23 CC162 000306 AMX/3
24 CC163 000306 AMX/3
25 CC164 000301 AMD/3
26 CC165 000306 AMX/3
27 CC166 000306 AMX/3
28 CC167 000306 AMX/3
29
30 CC170 000300 AMR/3          ;REL, AFTER AM
31 CC171 000305 AMRX/3
32 CC172 000305 AMRX/3
33 CC173 000305 AMRX/3
34 CC174 000300 AMR/3
35 CC175 000305 AMRX/3
36 CC176 000305 AMRX/3
37 CC177 000305 AMRX/3
```

```
!0CC5 .MAIN
01                ;INSTRUCTION EXECUTIONS
02                ;       INPUT:ESCAPE MODE, F FIELD(0:5)
03
04          000002 .DO 2
05
06 CC200 000137 CS/3
07 CC201 002120 SQRT/3
08 CC202 000467 BL/3
09 CC203 000454 HL/3
10 CC204 000356 LA/3
11 CC205 000360 LO/3
12 CC206 000362 LX/3
13 CC207 000364 WA/3
14
15 CC210 000371 WS/3
16 CC211 000343 AM/3
17 CC212 000376 WM/3
18 CC213 000345 AL/3
19 CC214 000135 ILLOP/3
20 CC215 001123 JL/3
21 CC216 000135 ILLOP/3
22 CC217 000135 ILLOP/3
23
24 CC220 000513 XL/3
25 CC221 000503 BS/3
26 CC222 000473 BA/3
27 CC223 000463 BZ/3
28 CC224 000354 RL/3
29 CC225 001112 SP/3
30 CC226 000135 ILLOP/3
31 CC227 000540 RS/3
32
33 CC230 000410 WD/3
34 CC231 000622 RX/3
35 CC232 000577 HS/3
36 CC233 000615 XS/3
37 CC234 000135 ILLOP/3
38 CC235 001670 CMVI/3
39 CC236 001457 DADD/3
40 CC237 001455 DADDI/3
```

```
37
38 CC300 000137 CS/3
39 CC301 002120 SQRT/3
40 CC302 000467 BL/3
41 CC303 000454 HL/3
42 CC304 000356 LA/3
43 CC305 000360 LO/3
44 CC306 000362 LX/3
45 CC307 000364 WA/3
46
47 CC310 000371 WS/3
48 CC311 000343 AM/3
49 CC312 000376 WM/3
50 CC313 000345 AL/3
51 CC314 000135 ILLOP/3
52 CC315 001123 JL/3
53 CC316 000135 ILLOP/3
54 CC317 000135 ILLOP/3
55
56 CC320 000513 XL/3
57 CC321 000503 BS/3
58 CC322 000473 BA/3
59 CC323 000463 BZ/3
60 CC324 000354 RL/3
```

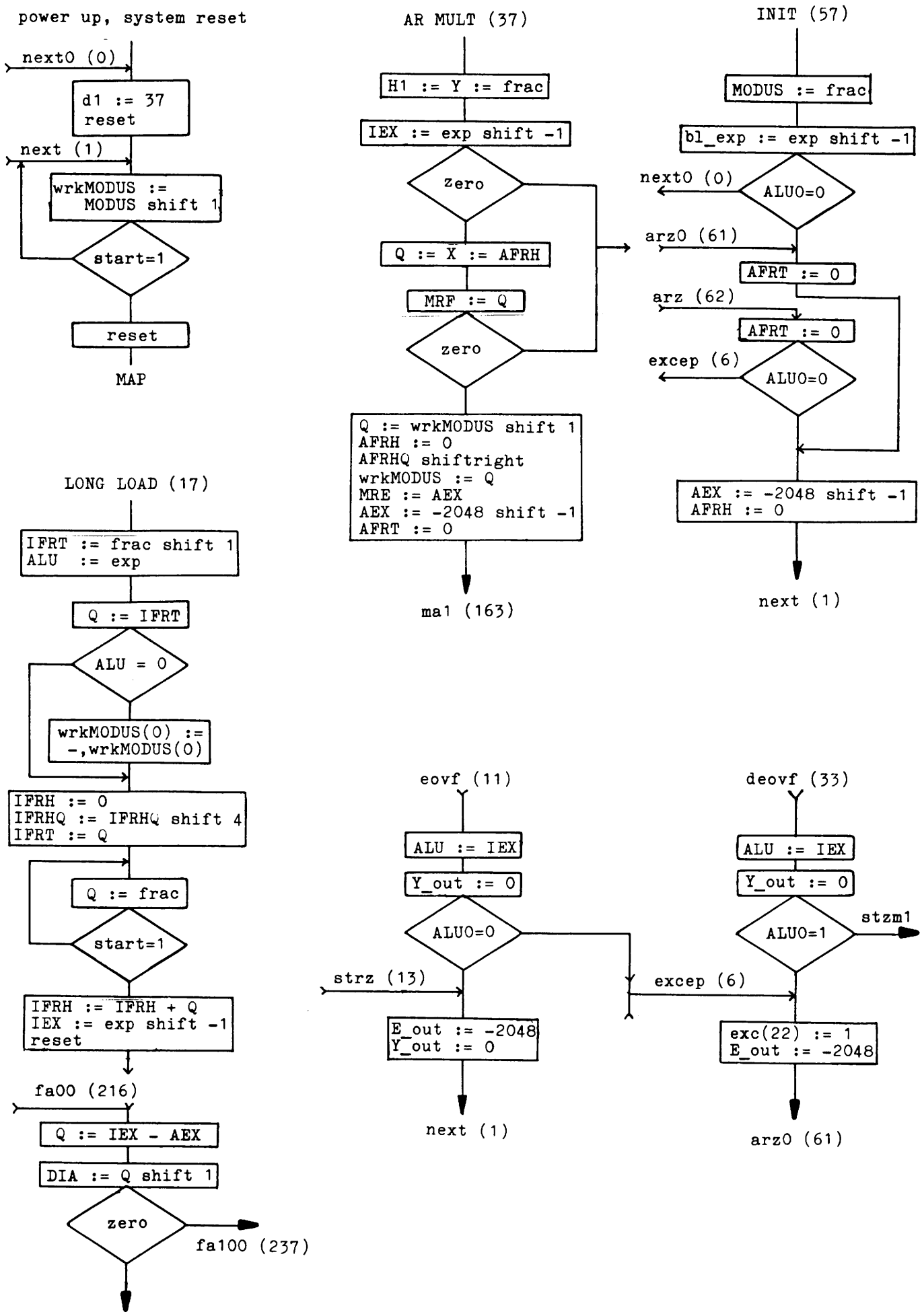
```
 0CC7 .MAIN
01 CC325 001112 SP/3
02 CC326 000135 ILLOP/3
03 CC327 000540 RS/3
04
05 CC330 000410 WD/3
06 CC331 000622 RX/3
07 CC332 000577 HS/3
08 CC333 000615 XS/3
09 CC334 000135 ILLOP/3
10 CC335 001670 CMVI/3
11 CC336 001457 DADD/3
12 CC337 001455 DADDI/3
```

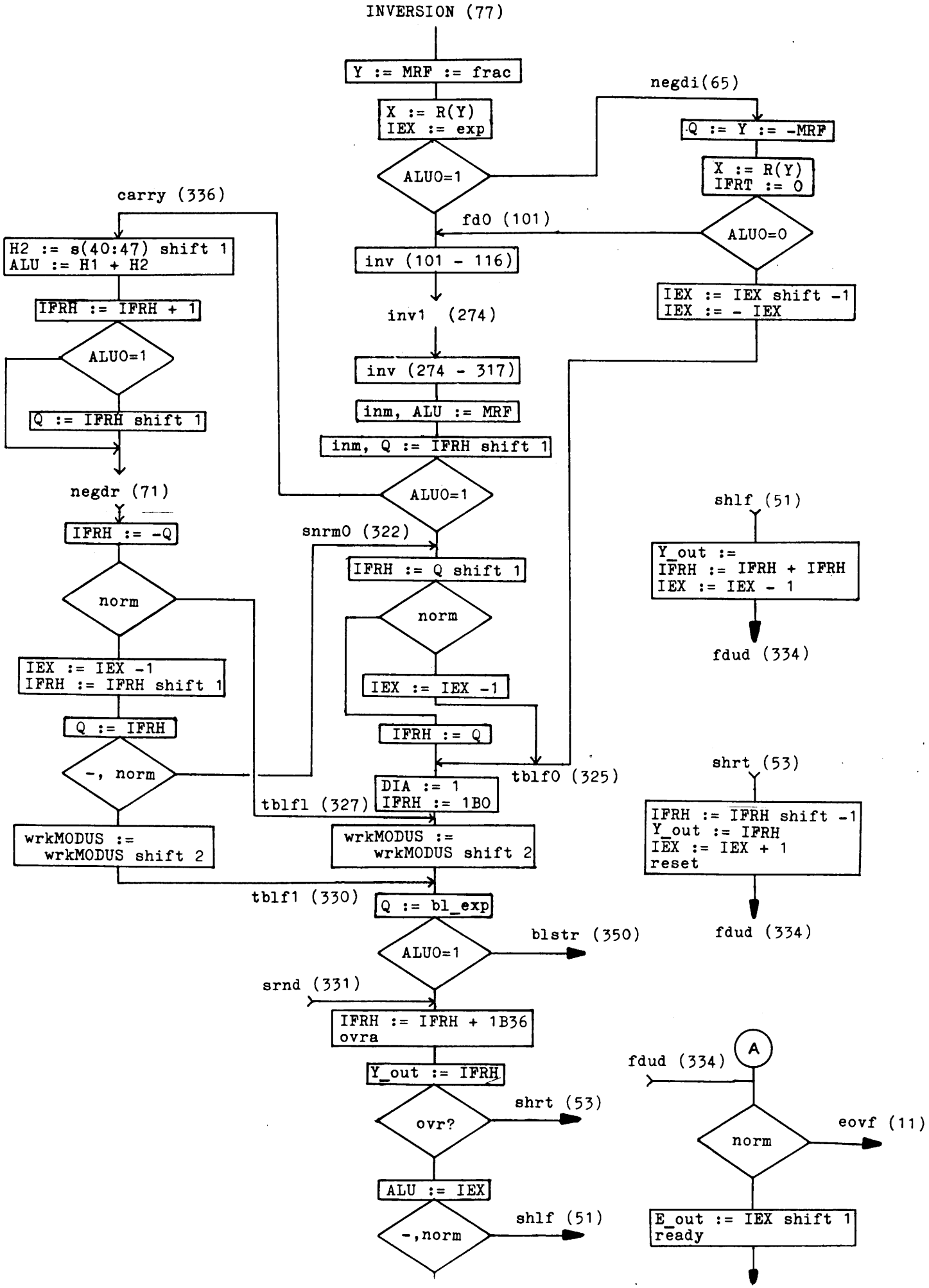
```
!0CC6 .MAIN
01 CC240 001140 CI/3
02 CC241 000346 AC/3
03 CC242 001032 NS/3
04 CC243 001047 ND/3
05 CC244 000733 AS/3
06 CC245 000756 AD/3
07 CC246 001000 LS/3
08 CC247 001015 LD/3
09
10 CC250 001065 SH/3
11 CC251 001071 SL/3
12 CC252 001075 SE/3
13 CC253 001077 SN/3
14 CC254 001101 SO/3
15 CC255 001104 SZ/3
16 CC256 001106 SX/3
17 CC257 001665 CMV/3
18
19 CC260 001327 FA/3
20 CC261 001331 FS/3
21 CC262 001250 FM/3
22 CC263 002062 KS/3
23 CC264 001400 FD/3
24 CC265 001214 CF/3
25 CC266 000676 DL/3
26 CC267 000542 DS/3
27
28 CC270 000704 AA/3
29 CC271 000715 SS/3
30 CC272 001526 MLA/3
31 CC273 001603 ARM/3
32 CC274 001615 INV/3
33 CC275 002000 STR/3
34 CC276 002215 BF/3
35 CC277 001503 MODUS/3
36
```

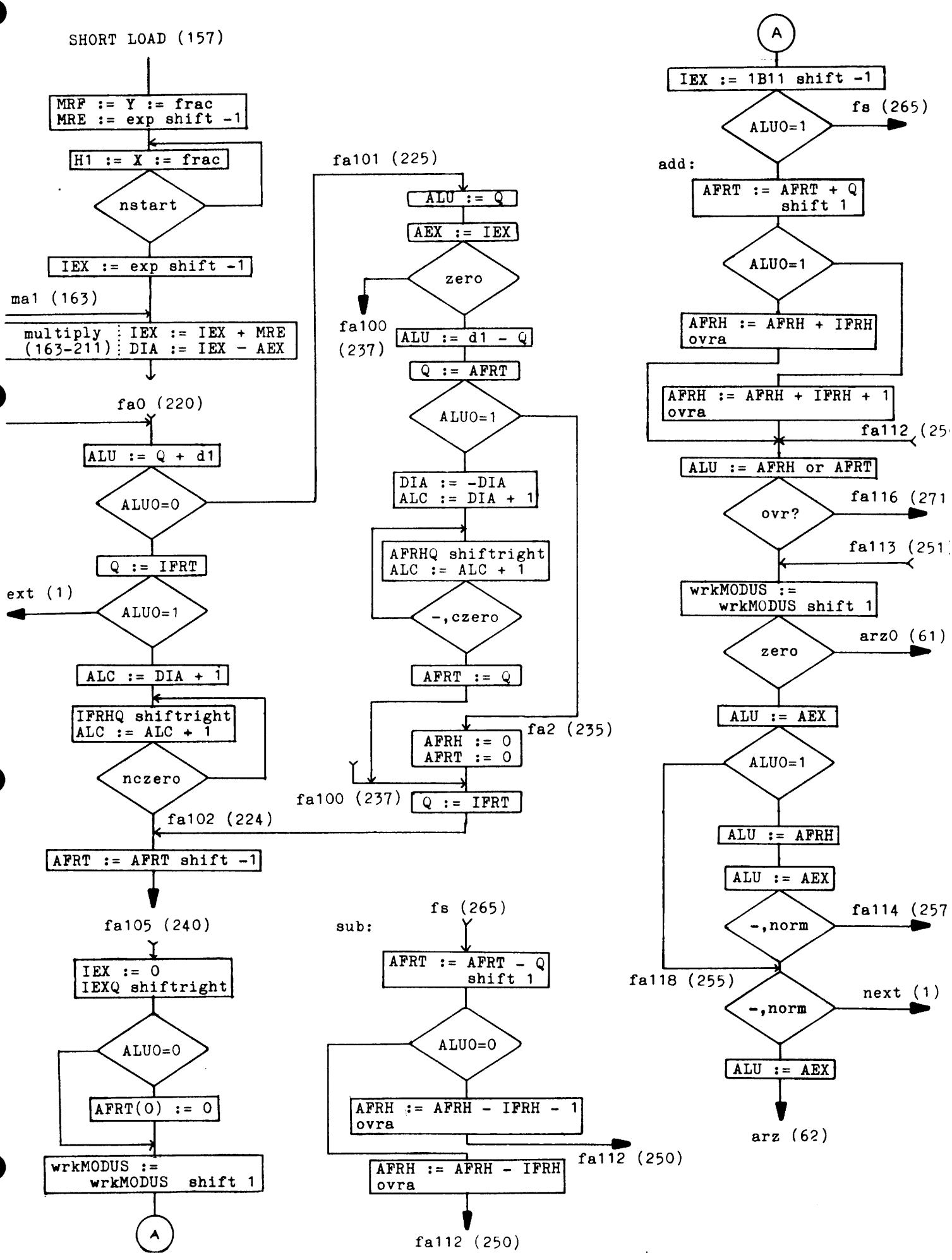
```
!0CC8 .MAIN
01 CC340 001140 CI/3
02 CC341 000346 AC/3
03 CC342 001032 NS/3
04 CC343 001047 ND/3
05 CC344 000733 AS/3
06 CC345 000756 AD/3
07 CC346 001000 LS/3
08 CC347 001015 LD/3
09
10 CC350 001065 SH/3
11 CC351 001071 SL/3
12 CC352 001075 SE/3
13 CC353 001077 SN/3
14 CC354 001101 SO/3
15 CC355 001104 SZ/3
16 CC356 001106 SX/3
17 CC357 001665 CMV/3
18
19 CC360 001327 FA/3
20 CC361 001331 FS/3
21 CC362 001250 FM/3
22 CC363 002062 KS/3
23 CC364 001400 FD/3
24 CC365 001214 CF/3
25 CC366 000676 DL/3
26 CC367 000542 DS/3
27
28 CC370 000704 AA/3
29 CC371 000715 SS/3
30 CC372 001526 MLA/3
31 CC373 001603 ARM/3
32 CC374 001615 INV/3
33 CC375 002000 STR/3
34 CC376 002215 BF/3
35 CC377 001503 MODUS/3
36
37
38
39                .END

0CCC SOURCE LINES IN ERROR
```

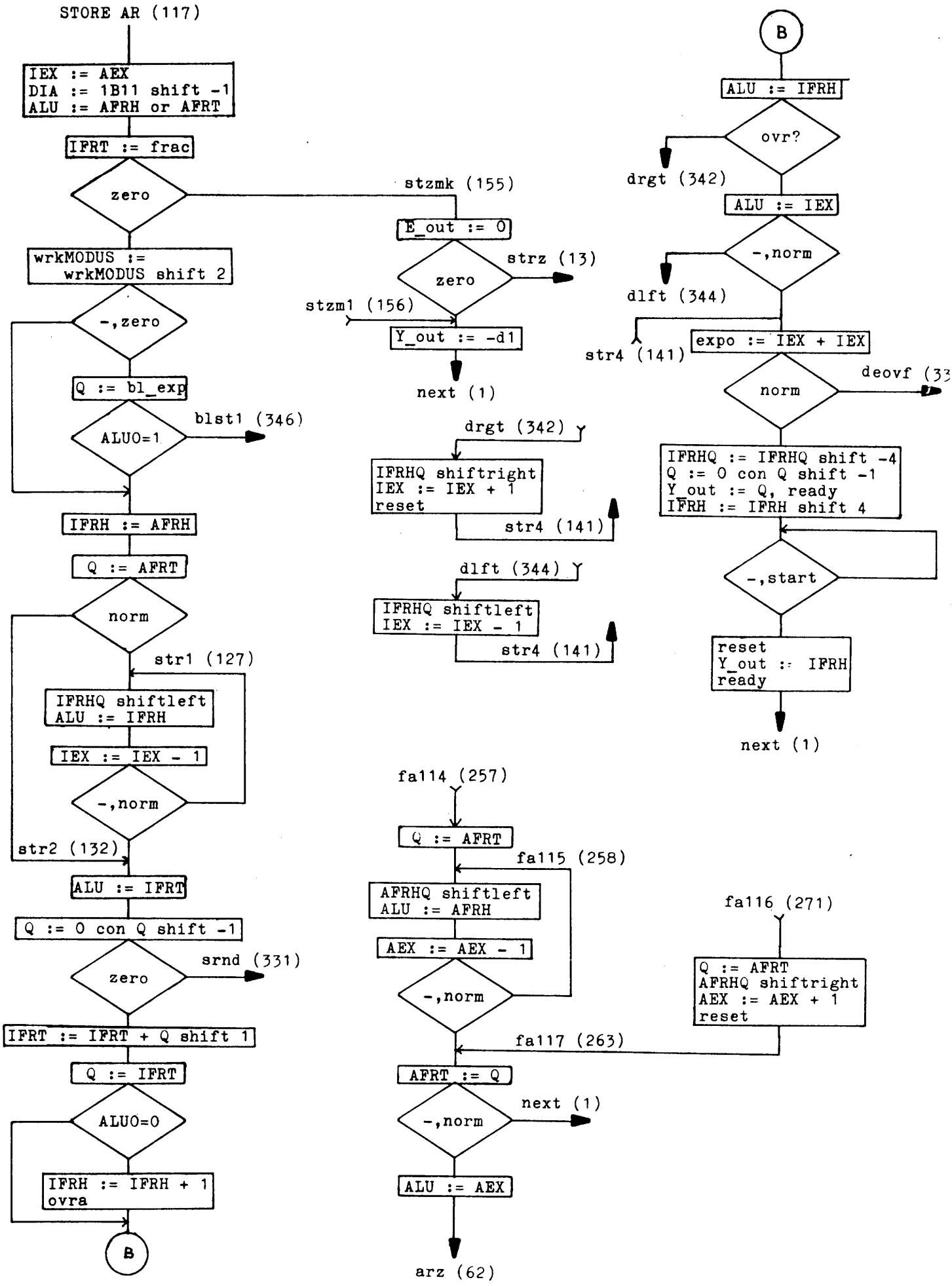
Appendix 2.

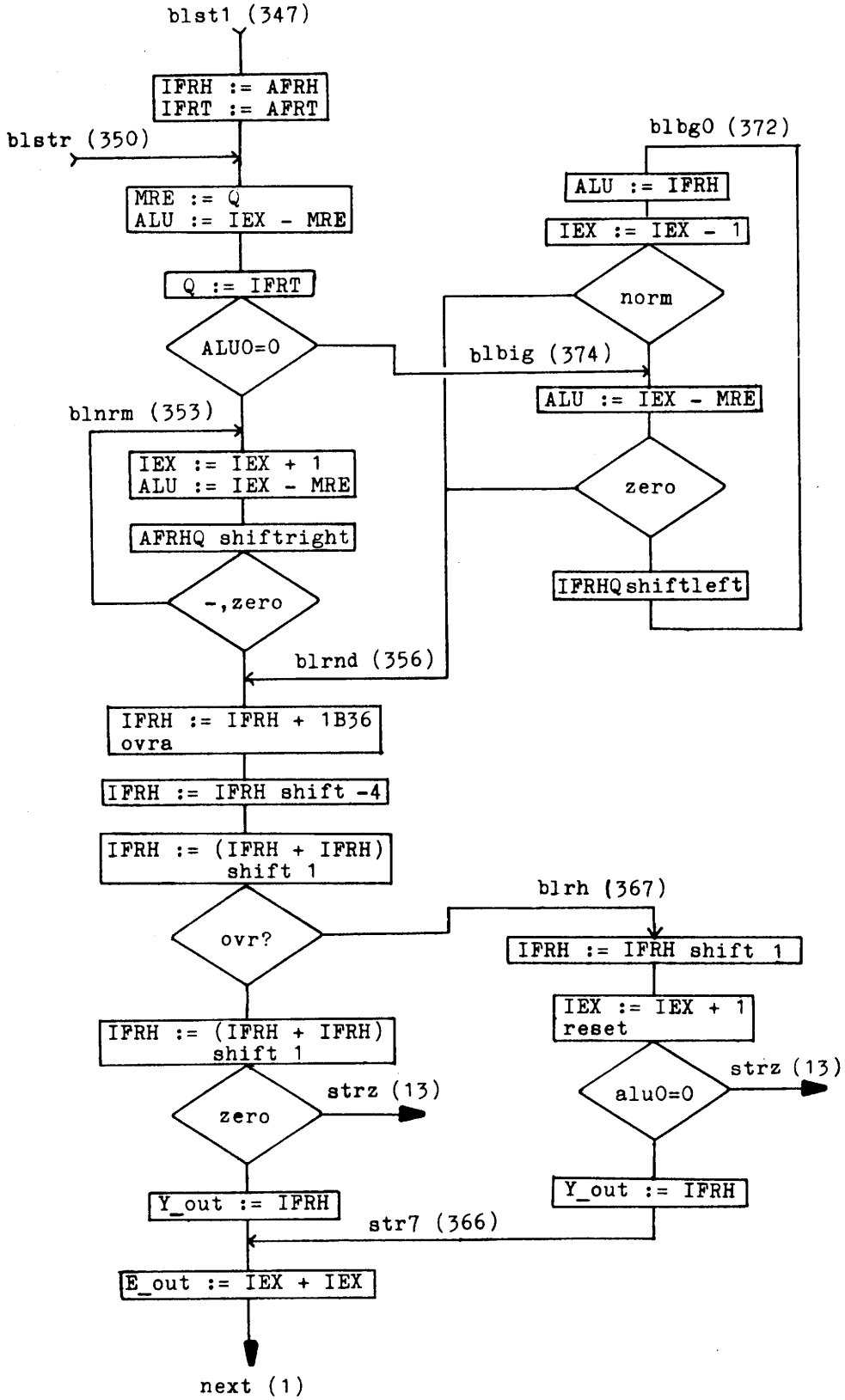
Listing of DPU microprogram :

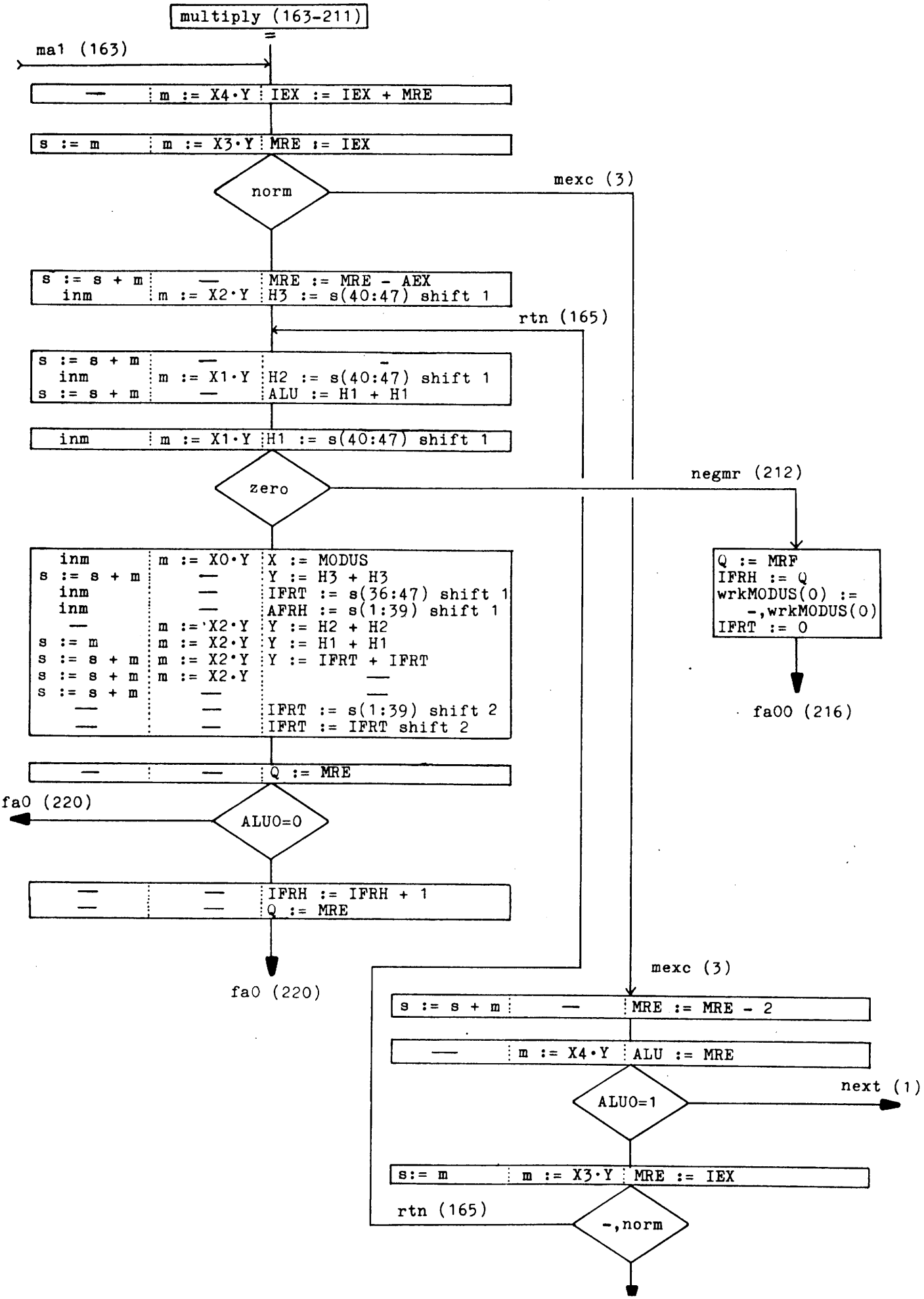

power up, system reset
next0 (0)
d1 := 37
reset
next (1)
wrkMODUS :=
MODUS shift 1
start=1
reset
MAP
AR MULT (37)
H1 := Y := frac
IEX := exp shift -1
zero
Q := X := AFRH
MRF := Q
zero
Q := wrkMODUS shift 1
AFRH := 0
AFRHQ shiftright
wrkMODUS := Q
MRE := AEX
AEX := -2048 shift -1
AFRT := 0
ma1 (163)
INIT (57)
MODUS := frac
bl_exp := exp shift -1
next0 (0)
ALU0=0
arz0 (61)
AFRT := 0
arz (62)
AFRT := 0
excep (6)
ALU0=0
AEX := -2048 shift -1
AFRH := 0
next (1)
LONG LOAD (17)
IFRT := frac shift 1
ALU := exp
Q := IFRT
ALU = 0
wrkMODUS(0) :=
-,wrkMODUS(0)
IFRH := 0
IFRHQ := IFRHQ shift 4
IFRT := Q
Q := frac
start=1
IFRH := IFRH + Q
IEX := exp shift -1
reset
fa00 (216)
Q := IEX - AEX
DIA := Q shift 1
zero
fa100 (237)
fa0 (220)
eovf (11)
ALU := IEX
Y_out := 0
ALU0=0
strz (13)
E_out := -2048
Y_out := 0
next (1)
deovf (33)
ALU := IEX
Y_out := 0
ALU0=1
stzm1 (156
excep (6)
exc(22) := 1
E_out := -2048
arz0 (61)

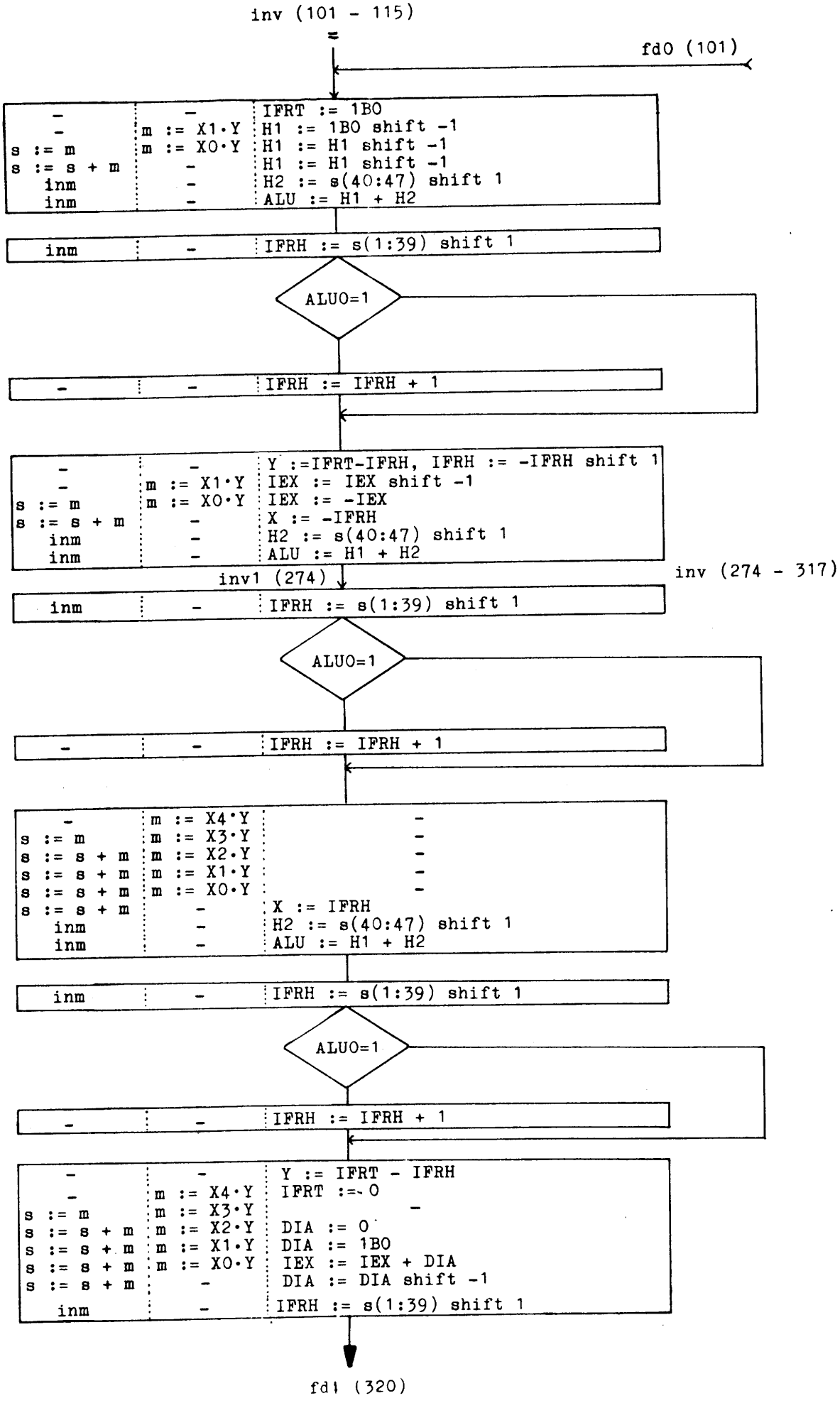

INVERSION (77)
Y := MRF := frac
X := R(Y)
IEX := exp
ALUO=1
negdi(65)
Q := Y := -MRF
X := R(Y)
IFRT := 0
ALUO=0
fd0 (101)
inv (101 - 116)
IEX := IEX shift -1
IEX := - IEX
inv1 (274)
inv (274 - 317)
inm, ALU := MRF
inm, Q := IFRH shift 1
ALUO=1
carry (336)
H2 := s(40:47) shift 1
ALU := H1 + H2
IFRH := IFRH + 1
ALUO=1
Q := IFRH shift 1
negdr (71)
IFRH := -Q
norm
IEX := IEX -1
IFRH := IFRH shift 1
Q := IFRH
-, norm
wrkMODUS :=
wrkMODUS shift 2
snrm0 (322)
IFRH := Q shift 1
norm
IEX := IEX -1
IFRH := Q
tblf0 (325)
DIA := 1
IFRH := 1B0
tblf1 (327)
wrkMODUS :=
wrkMODUS shift 2
tblf1 (330)
Q := bl_exp
ALUO=1
blstr (350)
srnd (331)
IFRH := IFRH + 1B36
ovra
Y_out := IFRH
ovr?
shrt (53)
ALU := IEX
-,norm
shlf (51)
shlf (51)
Y_out :=
IFRH := IFRH + IFRH
IEX := IEX - 1
fdud (334)
shrt (53)
IFRH := IFRH shift -1
Y_out := IFRH
IEX := IEX + 1
reset
fdud (334)
fdud (334)
A
norm
eovf (11)
E_out := IEX shift 1
ready


SHORT LOAD (157)
MRF := Y := frac
MRE := exp shift -1
H1 := X := frac
nstart
IEX := exp shift -1
ma1 (163)
multiply (163-211)
IEX := IEX + MRE
DIA := IEX - AEX
fa0 (220)
ALU := Q + d1
ALUO=0
Q := IFRT
ext (1)
ALUO=1
ALC := DIA + 1
IFRHQ shiftright
ALC := ALC + 1
nczero
fa102 (224)
AFRT := AFRT shift -1
fa105 (240)
IEX := 0
IEXQ shiftright
ALUO=0
AFRT(0) := 0
wrkMODUS := wrkMODUS shift 1
A
fa101 (225)
ALU := Q
AEX := IEX
zero
fa100 (237)
ALU := d1 - Q
Q := AFRT
ALUO=1
DIA := -DIA
ALC := DIA + 1
AFRHQ shiftright
ALC := ALC + 1
-,czero
AFRT := Q
fa2 (235)
AFRH := 0
AFRT := 0
fa100 (237)
Q := IFRT
fs (265)
sub:
AFRT := AFRT - Q shift 1
ALUO=0
AFRH := AFRH - IFRH - 1 ovra
fa112 (250)
AFRH := AFRH - IFRH ovra
fa112 (250)
A
IEX := 1B11 shift -1
ALUO=1
fs (265)
add:
AFRT := AFRT + Q shift 1
ALUO=1
AFRH := AFRH + IFRH ovra
AFRH := AFRH + IFRH + 1 ovra
fa112 (25
ALU := AFRH or AFRT
ovr?
fa116 (271
fa113 (251
wrkMODUS := wrkMODUS shift 1
zero
arz0 (61)
ALU := AEX
ALUO=1
ALU := AFRH
ALU := AEX
-,norm
fa114 (257
fa118 (255)
-,norm
next (1)
ALU := AEX
arz (62)


STORE AR (117)
IEX := AEX
DIA := 1B11 shift -1
ALU := AFRH or AFRT
IFRT := frac
zero
stzmk (155)
E_out := 0
zero
strz (13)
stzm1 (156)
Y_out := -d1
next (1)
wrkMODUS :=
wrkMODUS shift 2
-,zero
Q := bl_exp
ALU0=1
blst1 (346)
IFRH := AFRH
Q := AFRT
norm
str1 (127)
IFRHQ shiftleft
ALU := IFRH
IEX := IEX - 1
-,norm
str2 (132)
ALU := IFRT
Q := 0 con Q shift -1
zero
srnd (331)
IFRT := IFRT + Q shift 1
Q := IFRT
ALU0=0
IFRH := IFRH + 1
ovra
B
drgt (342)
IFRHQ shiftright
IEX := IEX + 1
reset
str4 (141)
dlft (344)
IFRHQ shiftleft
IEX := IEX - 1
str4 (141)
B
ALU := IFRH
ovr?
drgt (342)
ALU := IEX
-,norm
dlft (344)
str4 (141)
expo := IEX + IEX
norm
deovf (33
IFRHQ := IFRHQ shift -4
Q := 0 con Q shift -1
Y_out := Q, ready
IFRH := IFRH shift 4
-,start
reset
Y_out := IFRH
ready
next (1)
fa114 (257)
Q := AFRT
fa115 (258)
AFRHQ shiftleft
ALU := AFRH
AEX := AEX - 1
-,norm
fa116 (271)
Q := AFRT
AFRHQ shiftright
AEX := AEX + 1
reset
fa117 (263)
AFRT := Q
-,norm
next (1)
ALU := AEX
arz (62)


blst1 (347)
IFRH := AFRH
IFRT := AFRT
blstr (350)
MRE := Q
ALU := IEX - MRE
Q := IFRT
ALU0=0
blnrm (353)
IEX := IEX + 1
ALU := IEX - MRE
AFRHQ shiftright
-,zero
blbg0 (372)
ALU := IFRH
IEX := IEX - 1
norm
blbig (374)
ALU := IEX - MRE
zero
IFRHQ shiftleft
blrnd (356)
IFRH := IFRH + 1B36
ovra
IFRH := IFRH shift -4
IFRH := (IFRH + IFRH)
shift 1
ovr?
blrh (367)
IFRH := IFRH shift 1
IEX := IEX + 1
reset
alu0=0
strz (13)
IFRH := (IFRH + IFRH)
shift 1
zero
strz (13)
Y_out := IFRH
Y_out := IFRH
str7 (366)
E_out := IEX + IEX
next (1)


multiply (163-211)
ma1 (163)
— | m := X4·Y | IEX := IEX + MRE
s := m | m := X3·Y | MRE := IEX
norm
mexc (3)
s := s + m | — | MRE := MRE - AEX
inm | m := X2·Y | H3 := s(40:47) shift 1
rtn (165)
s := s + m | — | —
inm | m := X1·Y | H2 := s(40:47) shift 1
s := s + m | — | ALU := H1 + H1
inm | m := X1·Y | H1 := s(40:47) shift 1
zero
negmr (212)
inm | m := X0·Y | X := MODUS
s := s + m | — | Y := H3 + H3
inm | — | IFRT := s(36:47) shift 1
inm | — | AFRH := s(1:39) shift 1
— | m := X2·Y | Y := H2 + H2
s := m | m := X2·Y | Y := H1 + H1
s := s + m | m := X2·Y | Y := IFRT + IFRT
s := s + m | m := X2·Y | —
s := s + m | — | —
— | — | IFRT := s(1:39) shift 2
— | — | IFRT := IFRT shift 2
Q := MRF
IFRH := Q
wrkMODUS(0) := -,wrkMODUS(0)
IFRT := 0
fa00 (216)
— | — | Q := MRE
fa0 (220)
ALU0=0
— | — | IFRH := IFRH + 1
— | — | Q := MRE
fa0 (220)
mexc (3)
s := s + m | — | MRE := MRE - 2
— | m := X4·Y | ALU := MRE
ALU0=1
next (1)
s:= m | m := X3·Y | MRE := IEX
rtn (165)
-,norm


inv (101 - 115)
fd0 (101)
IFRT := 1B0
H1 := 1B0 shift -1
H1 := H1 shift -1
H1 := H1 shift -1
H2 := s(40:47) shift 1
ALU := H1 + H2
s := m
s := s + m
m := X1·Y
m := X0·Y
inm
IFRH := s(1:39) shift 1
ALU0=1
IFRH := IFRH + 1
Y :=IFRT-IFRH, IFRH := -IFRH shift 1
IEX := IEX shift -1
IEX := -IEX
X := -IFRH
inv1 (274)
inv (274 - 317)
m := X4·Y
m := X3·Y
m := X2·Y
m := X1·Y
m := X0·Y
X := IFRH
Y := IFRT - IFRH
IFRT := 0
DIA := 0
DIA := 1B0
IEX := IEX + DIA
DIA := DIA shift -1
fd1 (320)

```
 0001 0000
0001                    ; **** MICRO PROGRAM FOR DPU 4.8.81 GI ****
0002                    ; ** IT IS CLAIMED THAT THE MASTER CPU IS HAVING A CYCLUS OF **
0003                    ; ** AT LEAST 0.16 MICROSEC.                                 **
0004                            .XPNG
0005
0006
0007
0008                    ;
0009                    ; MACRO FILL
0010                    ;
0011                    ; THE MACRO IS STORING ONES UNTIL 'LOCATION'
0012                    ;
0013                    ; MACRO CALL:
0014                    ;
0015                    ; FILL 'LOCATION'
0016                    ;
0017
0018                    .MACRO FILL
0019                    .LIST 0
0020                      .IFG  ./3-.1
0021                        SEGMENT TOO LONG
0022                      .ENDC
0023                      .DO   .1*3-.
0024                        -1
0025                      .ENDC
0026                    .LIST 1
0027                    %
0028
0029                    ; DEFINITIONS OF PARAMETERS:
```

```
!00C2 DPU
01                      ;
02                      ; CONDITION PARAMETERS:
03              CC0007          MAP     =7      ; JUMP MAP UNCONDITIONAL.
04              CC0020          ZERO    =20     ; RESULT EQUAL ZERO.
05              CC0021          NCZERO  =21     ; COUNTER NOT EQUAL ZERO.
06              CC0022          NOVR    =22     ; NO OVERFLOW.
07              CC0023          OVRQ?   =23     ; OVERFLOW IN Q-REG.
08              CC0024          START   =24     ; START COMMAND.
09              CC0025          ALUC    =25     ; RESULT BIT 0.
10              CC0026          ALU1C   =26     ; RESULT BIT 0 NOT EQUAL BIT 1.
11              CC0027          JMP     =27     ; JUMP UNCONDITIONAL.
12              CC0030          NZERO   =30     ; RESULT NOT EQUAL ZERO.
13              CC0031          CZERO   =31     ; COUNTER EQUAL ZERO.
14              CC0032          OVR?    =32     ; OVERFLOW.
15              CC0033          NOVRQ   =33     ; NO OVERFLOW IN Q-REG.
16              CC0034          NSTART  =34     ; NO START COMMAND.
17              CC0035          NALU0   =35     ; RESULT BIT 0 EQUAL ZERO.
18              CC0036          NALU10  =36     ; RESULT BIT 0 EQUAL BIT 1.
19              CC0037          NJMP    =37     ; UNCONDITIONAL NO JUMP.
20                      ; DO PARAMETERS (DEFAULT: 6):
21              CC0007          INM     =7      ; INHIBIT MULTIPLY.
22              CC0004          IVT     =4      ; INVERSE TABEL.
23              CC0002          READY   =2      ; SET FPU READY FLAG.
24              CC0006 DODF=6;
25
26                      ; BUS PARAMETERS (DEFAULT: 17):
27              CC0015          NACC    =15     ; NO ACCUMULATION.
28              CC0035          ACC     =35     ; ACCUMULATION.
29              CC0007          FRAC    =7      ; BUS(0:35) := INPUT(0:35), BUS(36:39) := 0.
30              CC0013          EXP     =13     ; BUS(0:11) := INPUT(36:47), BUS(36:39) := 0.
31              CC0015          SUM     =15     ; BUS(0:39) := SUM(0:39).
32              CC0016          INT     =16     ; BUS(0:11) := SUM(36:47).
33              CC0017 BUSDF=17;
34
35                      ; DESTINATION PARAMETERS (DEFAULT 7):
36              CC0000          X       =0      ; X REG.
37              CC0001          Y       =1      ; Y REG, AND OUTPUT(0:35) := ALU(0:35).
38              CC0002          EXPO    =2      ; OUTPUT(36:47) := ALU(0:11).
39              CC0003          EXC     =3      ; EXCEPTION(16:23) := BUS(4:11).
40              CC0004          OVRQ    =4      ; OVERFLOW Q.
41              CC0005          OVRA    =5      ; OVERFLOW RAM.
42              CC0006          RESET   =6      ; RESET READY, START AND OVERFLOW.
43              CC0007 DESDF=7;
```

```
!OOC7 CPU
01
02                      ; MULTIPLY PARAMETERS (DEFAULT 7):
03              CC0003          X4      =3      ; SELECT X4, X3.
04              CC0005          X2      =5      ;   -    X2.
05              CC0006          X1      =6      ;   -    X1, X0.
06              CC0007 MULDF=7;
07
08                      ; CONSTANTS:
09              CC0200          K2048   =200
10              CC0001          K36     =1
11              CC0002          K1      =2
12              CC0004          K2      =4
13              CC0010          K4      =10
14              CC0112 K74=112
15
16                          ; JUMPLABEL DEFAULT  3 * 377 = 1375
17              CC1375 JMPDF=1375
18
```

```
!0CC4 DPU
01
02                      ; RAM PARAMETERS:
03                      ;       REGISTER GROUPS A,B,C,D INDEX 0:3
04                      ;       FIRST MENTIONED INDEX IS LOADED
05                              ;  RAMB   RAMA
06                  CC0000 A00=00   ;  IFRH   IFRH
07                  CC0020 A01=20   ;  IFRH   IFRT
08                  CC0040 A02=40   ;  IFRH   AFRH
09                  CC0022 A11=22   ;  IFRT   IFRT
10                  CC0062 A13=62   ;  IFRT   AFRT
11                  CC0004 A20=04   ;  AFRH   IFRH
12                  CC0044 A22=44   ;  ARFH   AFRH
13                  CC0064 A23=64   ;  AFRH   AFRT
14                  CC0066 A33=66   ;  AFRT   AFRT
15                  CC0010 B00=10   ;  IEX    IEX
16                  CC0030 B01=30   ;  IEX    AEX
17                  CC0050 B02=50   ;  IEX    MRE
18                  CC0070 B03=70   ;  IEX    DIA
19                  CC0012 B10=12   ;  AEX    IEX
20                  CC0032 B11=32   ;  AEX    AEX
21                  CC0014 B20=14   ;  MRE    IEX
22                  CC0034 B21=34   ;  MRE    AEX
23                  CC0054 B22=54   ;  MRE    MRE
24                  CC0016 B30=16   ;  DIA    IEX
25                  CC0036 B31=36   ;  DIA    AEX
26                  CC0056 B32=56   ;  DIA    MRE
27                  CC0076 B33=76   ;  DIA    DIA
28                  CC0001 C00=01   ;  MRF    MRF
29                  CC0023 C11=23   ;  H1     H1
30                  CC0043 C12=43   ;  H1     H2
31                  CC0045 C22=45   ;  H2     H2
32                  CC0067 C33=67   ;  H3     H3
33                  CC0011 D00=11   ;  MODUS  MODUS
34                  CC0033 D11=33   ;  K74    K74
35                  CC0055 D22=55   ;  BLEX   BLEX
36                  CC0017 D30=17   ;  WMODU  MODUS
37                  CC0077 D33=77   ;  WMODU  WMODU
38                      ; ALU DESTINATION PARAMETERS:
39                  CC0000          LOADQ   =0      ; LOAD Q-REG.
40                  CC0001          NLOAD   =1      ; NO LOAD.
41                  CC0003          LOADA   =3      ; LOAD RAM.
42                  CC0005          SHAD    =5      ; SHIFT RAM DOWN BEFORE LOAD.
43                  CC0007          SHAU    =7      ; SHIFT RAM UP BEFORE LOAD.
44                  CC0004 LSHAD=4; SHIFT RAM CON Q LEFT AND LOAD RAM
45                  CC0006 LSHAU=6; SHIFT RAM CON Q RIGHT AND LOAD RAM
```

```
10005 MCU
01
02                      ; FUNCTION PARAMETERS:
03              CC0174          LOADF   =0174   ; D(C:39) + 0.
04              CC0177          LOADE   =0177   ; D(0:11) + 0, D(12:39) AND 0.
05              CC1400          SUBQA   =1400   ; Q - RAM.
06              CC3040          PAS.Q   =3040   ; Q OR 0.
07              CC2440          NEG.Q   =2440   ; 0 - Q.
08              CC0000          ADDQA   =0000   ; Q + RAM.
09              CC0024          ADD     =0024   ; RAM + RAM.
10              CC3110          PAS.A   =3110   ; RAM OR 0.
11              CC0130          ADD.E   =0130   ; RAM + (D(0:11),Z(12:35),D(36:39))
12              CC3174          PAS     =3174   ; D(0:39).
13              CC0175          LOADI   =0175   ; D(C:11) + 0, D(12:35) AND 0, D(36:39) + 0.
14              CC2510          NEG.A   =2510   ; 0 - RAM.
15              CC2424          SUB     =2424   ; RAMA - RAMB.
16              CC2424          CLEAR   =2424   ; ZERO.
17              CC1530          SUB.E   =1530   ; RAM -(D(0:11),Z(12:35),D(36:39))
18              CC1424          ISUB=1424   ; RAMB - RAMA.
19              CC0424 ADD1 =0424; RAM + RAM + 1
20              CC1024 ISUB1=1024; RAM - RAM - 1
21              CC2400 SUBAQ=2400; RAM - Q
22              CC0311 INC.A=0311; RAM + 1B11
23              CC2530 ISUBE=2530; (D(0:11),Z(12:35),D(36:39)) - RAM
24              CC0510 INC.R=0510; RAM + 1B39
25              CC3024 OR=3024; RAM OR RAM
```

```
10006 DPU
01                      ; DEFINITIONS OF MACRO'S USED IN FPU 801 MICRO PROGRAM.
02
03                      ; EXEC (JUMPADDRESS,CONDITION,DO,BUS,DEST,X,RAM,ALUDEST,FUNCTION)
04                              .MACRO EXEC
05                      ++      ((%1)/3)B7+(%2)B12+(%3)B15
06                      ++      (%4)B4+(%5)B7+(%6)B10+((%7)/2)B15
07                      ++      (%8)B2+((%7)&1)B3+(%9)B14+1B15
08                      %
09
10                      ; IMID (CONSTANT,DEST,RAM,ALUDEST,FUNCTION)
11                              .MACRO IMID
12                      ++      (%1)B7+376B15
13                      ++      17B4+(%2)B7+7B10+((%3)/2)B15
14                      ++      (%4)B2+((%3)&1)B3+(%5)B14
15                      %
16
17                      ; MULT (BUS,X)
18                              .MACRO MULT
19                      ++      377B7+376B15
20                      ++      (%1)B4+7B7+(%2)B10+37B15
21                      ++      77B7+377B15
22                      %
23
24                      ; JUMP (ADDRESS,CONDITION)
25                              .MACRO JUMP
26                      ++      ((%1)/3)B7+(%2)B12+6B15
27                      ++      17B4+7B7+7B10+37B15
28                      ++      77B7+377B15
29                      %
30
```

```
!CCC7 U
01
02                      ; BUSALU(DO,BUS,DEST,RAM,ALUDEST,FUNCTION)
03                              .MACRO BUSALU
04                      ++      377B7+37B12+(%1)B15
05                      ++      (%2)B4+(%3)B7+7B10+((%4)/2)B15
06                      ++      (%5)B2+((%4)&1)B3+(%6)B14+1B15
07                      %
08                      ; ALUJMP (ADDRESS,CONDITION,BUS,DEST,RAM,ALUDEST,FUNCTION)
09                              .MACRO ALUJMP
10                      ++      ((%1)/3)B7+(%2)B12+6B15
11                      ++      (%3)B4+(%4)B7+7B10+((%5)/2)B15
12                      ++      (%6)B2+((%5)&1)B3+(%7)B14+1B15
13                      %
14
15                      ; ALU (RAM,ALUDEST,FUNCTION)
16                              .MACRO ALU
17                      ++      377B7+37B12+6B15
18                      ++      17B4+7B7+7B10+((%1)/2)B15
19                      ++      (%2)B2+((%1)&1)B3+(%3)B14+1B15
20                      %
21
22                      ; IEXEC (CONSTANT,DO,BUS,DEST,X,RAM,ALUDEST,FUNCTION)
23                              .MACRO IEXEC
24                      ++      (%1)B7+37B12+(%2)B15
25                      ++      (%3)B4+(%4)B7+(%5)B10+((%6)/2)B15
26                      ++      (%7)B2+((%6)&1)B3+(%8)B14
27                      %
```

```
!0CC8 DPU
01
02
03                    CC0000        .LCC C   ; LOCATION 0 IS POWER UP START
04
05                              ;
06                              ;  NEXT INSTRUCTION
07 CC000 2257717673270372 NEXTO: IMID K74,RESET,D11,LOADA,LOADI ; D11 := K74
08 CC001 0036117771776221 NEXT: EXEC NEXT,NSTART,READY,BUSDF,DESDF,MULDF,D30,SHAU,PAS.A ;
09                              ; WRKMODUS := MODUS SHIFT 1; WAIT ON START;
10 CC002 7763717670C15051 ALUJMP JMPDF,MAP,BUSDF,RESET,C00,LOADQ,CLEAR ; Q := 0; GOTO FUNCTION;
11                              ;
12                              ; EXCEPTION FOR MULT
13                              ; ******************
14 CC003 0057717775463260 MEXC: IEXEC K1,DODF,BUSDF,DESDF,MULDF,B22,LOADA,SUB.E ; MRE:=MRE-2; SUM:=SUM+MULT;
15 CC004 0032717735426221 EXEC NEXT,ALU0,DODF,BUSDF,DESDF,X4,B22,NLOAD,PAS.A ; ALU:=MRE; MULT:=X4*Y;
16                              ; IF UNDERFLOW THEN GOTO NEXT;
17 CC005 3537315771466221 ALUJMP RTN,NALU10,NACC,DESDF,B20,LOADA,PAS.A ; SUM:=MULT; MULT:=X3*Y; MRE:=IEX;
18                              ;               IF IEX - 2 < 2048 THEN GOTO RTN;
19                              ;
20                              ;
21                              ; EXCEPTION
22                              ; *********
23                              ; THE EXEPTIONREGISTER IS CLEAR BY A RESET.
24 CC006 0117517370C26370 EXCEP: IEXEC K2,READY,BUSDF,EXC,MULDF,A00,NLOAD,PAS ; EX(22) := 1; OVERFLOW;
25 CC007 4017717270C20372 IMID K2C48,EXP0,A00,NLOAD,LOADI ; E_OUT := -2048;
26 CC010 1433717777637777 JUMP ARZC,JMP ; GOTO ARZ0;
27                              ;
28                              ; TEST EXP_OVERFLOW
29 CC011 7777717771C26221 EOVF: ALL BCC,NLOAD,PAS.A ; ALU := IEX;
30 CC012 0156717170C65051 ALUJMP EXCEP,NALU0,BUSDF,Y,AC0,LOADA,CLEAR ; Y_OUT:=IFRH:=0; IF POS GOTO EXCEPTION;
31                              ;
32                              ; STORE ZERO
33                              ; **********
34 CC013 4C1771727CC20372 STRZ: IMID K2048,EXP0,A00,NLOAD,LOADI ; E_OUT := -2048;
35 CC014 0033517170C25051 EXEC NEXT,JMP,READY,BUSDF,Y,MULDF,A00,NLOAD,CLEAR ; Y := 0; GOTO NEXT;
```

```
!0009 DRU
01                              FILL
02                    CC0001 .LIST 1
03                    CC0055        .LOC 55    ;   MAP ADDRESS  17
04                           ;
05                           ; LONG LOAD (LOAD PRODUCT)  AND  ADD
06                           ;
07                           ; INPUT      TAILPART  EXP = 0 SIGN AS MODUS/ <> 0 OPPOSITE SIGN AS MODUS
08                           ; INPUT      HEADPART  EXPO_HEAD
09 CC017 7777707672360371 LL: BUSALU DODF,FRAC,RESET,A11,SHAU,LOADF ; CLEAR SIGNBIT; IFRT := FRACTION;
10 CC020 7777513770C20373 BUSALU  READY,EXP,DESDF,A00,NLOAD,LOADI ; IF EXPO = 0
11 CC021 0470317772206221 ALUJMP LL1,ZERO,BUSDF,DESDF,A11,LOADQ,PAS.A ; THEN GOTO LL1  (Q := IFRT)
12                           ;
13                           ; CHANGE SIGNBIT OF WRKMODUS
14 CC022 4017717777673260 IMID K2C48,DESDF,D33,LOADA,SUB.E ; ELSE WRKMODUS := WRKMODUS + 1B0
15                           ;
16                           ; IFRH(C:35) := 0 , IFRH(36:39) := Q(0:3) , IFRT(0:35) := Q(4:39) , IFRT(36:39) := 0
17 CC023 7777717770145051 LL1: ALU,A0C,LSHAU,CLEAR ;
18 CC024 7777717770146221 ALU A00,LSHAU,PAS.A ;
19 CC025 7777717770146221 ALU A00,LSHAU,PAS.A ;
20 CC026 7777717770146221 ALU A00,LSHAU,PAS.A ;
21 CC027 7777717772266101 ALU A11,LOADA,PAS.Q ; IFRT := Q;
22 CC030 0616107770CC0371 LL2: EXEC LL2,NSTART,READY,FRAC,DESDF,MULDF,A00,LOADQ,LOADF ; Q := FRAC(0:35)
23 CC031 7777717777C60001 ALU A00,LOADA,ADDQA ; IFRH := IFRH + Q;
24 CC032 4353713671120373 ALUJMP FA00,JMP,EXP,RESET,BCC,SHAD,LOADI ; GOTO FA00; IEX := EXPO_HEAD SHIFT (-1);
25                           ;
26                           ; DOUBLE STORE EXP OVER/UNDERFLOW
27 CC033 7777717771C26221 DEOVF: ALU B00,NLOAD,PAS.A ; ALU := IEX;
28 CC034 3352717173235051 ALUJMP STZM1,ALU0,BUSDF,Y,D11,NLOAD,CLEAR ; Y_OUT := 0;
29                           ; IF NEG THEN GOTO STORE ZERO MARK;
30 CC035 0153717777637777 JUMP EXCEP,JMP ; GOTO EXCEPTION;
```

```
1001C DPU
01                                  FILL 37
02                      CC0001 .LIST 1
03                      CC0135         .LOC 135    ;  MAP ADDRESS  37
04                             ;
05                             ; ARM           ARF := ARF * FRAC;      AEX := AEX + EXP;
06                             ;
07 CC037 7777707172270371 ARM: BUSALU DODF,FRAC,Y,C11,LOADA,LOADF ; H1 := Y := FRAC;
08 CC040 1430313671120373 ALUJMP ARZO,ZERO,EXP,RESET,BCO,SHAD,LOADI ; IEX := EXP // 2; IF ZERO GOTO AZERO;
09 CC041 7777717074C6221 BUSALU DODF,BUSDF,X,A22,LOADQ,PAS.A ; Q := X := AFRH;
10 CC042 1430317770C76101 ALUJMP ARZO,ZERO,BUSDF,DESDF,COO,LOADA,PAS.Q ; MRF := Q; IF ZERO GOTO AZERO;
11                             ;
12                             ; CLEAR AR AND SIGN OF WRKMODUS
13 CC043 7777717777610051 ALU D33,LOADQ,ADD ; Q := WRKMODUS SHIFT 1;
14 CC044 77771777745C5051 ALU A22,LSHAD,CLEAR ; AFRH := 0; Q := 0 CON Q SHIFT -1;
15 CC045 7777717777676101 ALU D33,LOADA,PAS.Q ; WRKMODUS := Q;
16 CC046 7777717773466221 ALU B21,LOADA,PAS.A ; MRE := AEX;
17 CC047 4017717773320372 IMID K2C48,DESDF,B11,SHAD,LOADI ; AEX := -2048 // 2;
18 CC050 3473717776665051 ALUJMP MA1,JMP,BUSDF,DESDF,A33,LOADA,CLEAR ; AFRT := 0;
19                             ;
20                             ; SHORTLEFTSHIFT :
21 CC051 7777717170C60051 SHLF: BUSALU DODF,BUSDF,Y,ACC,LOADA,ADD ; Y_OUT := IFRH := IFRH * 2;
22 CC052 6713717777C63051 ALUJMP FDUD,JMP,BUSDF,DESDF,BO3,LOADA,ISUB ; IEX := IEX -1; GOTO FDUD;
23                             ;
24                             ; SHORTRIGHTSHIFT :
25 CC053 7777717770126221 SHRT: ALU ACO,SHAD,PAS.A ; IFRH := IFRH // 2; SIGNSHIFT;
26 CC054 7777717170C26221 BUSALU DODF,BUSDF,Y,AOO,NLOAD,PAS.A ; Y_OUT := IFRH;
27 CC055 6713717677C60051 ALUJMP FDUD,JMP,BUSDF,RESET,BO3,LOADA,ADD ; IEX := IEX + 1; GOTO FDUD;
```

```
!0011 ALU
01                          FILL 57
02                  CC0001 .LIST 1
03                  CC0215     .LOC 215    ;   MAP ADDRESS  57
04
05                         ;
06                         ; INPUT      BIT  0:7  MODUS
07                         ;            BIT 16:23  1B16
08                         ;            BIT 36:47  ZERO EXPONENT
09 CC057 7777707671C70371 INIT: BUSALU DODF,FRAC,RESET,D00,LOADA,LOADF
10 CC060 0016513775530373 EXEC NEXT0,NALU0,READY,EXP,DESDF,MULDF,D22,SHAD,LOADI ; LOAD BLOCKFL EXP;
11                         ; IF MODUS(C) = 0 THEN GOTO NEXT0;
12                         ; UNCONDITIONALLY ZEROSET OF AR
13 CC061 1473717776665051 ARZ0: ALUJMP,ARZ1,JMP,BUSDF,DESDF,A33,LOADA,CLEAR ; AFRT := 0; GOTO ARZ1;
14                         ;
15                         ; IF -, EXPOVERFLOW THEN ZEROSET AR
16 CC062 0156717776665051 ARZ: ALUJMP,EXCEP,NALU0,BUSDF,DESDF,A33,LOADA,CLEAR ; AFRT := 0;
17                                  ; IF EXP_OVERFLOW GOTO EXCEP;
18 CC063 4017517773320372 ARZ1: IEXEC K2048,READY,BUSDF,DESDF,MULDF,B11,SHAD,LOADI ; AEX := -2048;
19 CC064 0033717774465051 ALUJMP NEXT,JMP,BUSDF,DESDF,A22,LOADA,CLEAR ; AFRH := 0; GOTO NEXT
20                         ;
21                         ; NEGATIVE DIVISOR INPUT :
22 CC065 7777717170C35221 NEGDI: BUSALU DODF,BUSDF,Y,CC0,NLOAD,NEG.A ; Y := -DIVISOR;
23 CC066 2036617072265051 EXEC FDC,NALU0,IVT,BUSDF,X,MULDF,A11,LOADA,CLEAR ; X:=R(Y(0:11)); IFRT:=0;
24                                  ; IF -DIVISOR>0 THEN GOTO FD0;
25 CC067 7777717771126221 ALU B00,SHAD,PAS.A ; IEX := IEX // 2;;
26 CC070 6533717771C65221 ALUJMP TBLFC,JMP,BUSDF,DESDF,B00,LOADA,NEG.A ; IEX := - IEX;
27                         ;
28                         ; NEGATIVE DIVISOR RESULT :
29 CC071 7777717770C65101 NEGDR: ALU AC0,LOADA,NEG.Q ; IFRH := -Q;
30 CC072 6573317770C26221 ALUJMP TBLFL,ALU10,BUSDF,DESDF,A00,NLOAD,PAS.A ; IF NORMALIZED GOTO TEST BLFL
31 CC073 7777717777C63051 ALU B03,LOADA,ISUB ; IEX := IEX - 1;
32 CC074 7777717770C60051 ALU A00,LOADA,ADD ; IFRH := IFRH * 2;
33 CC075 6457317770CC6221 ALUJMP SNRMC,NALU10,BUSDF,DESDF,A00,LOADQ,PAS.A ; IF -,NORMALIZED GOTO SNRMC;
34 CC076 6613717777770051 ALUJMP TBLF1,JMP,BUSDF,DESDF,D33,SHAU,ADD ; WRKMODUS SHIFT 2; GOTO TBLF1;
```

```
!0012 DPU
01                          FILL 77
02                  CC0001 .LIST 1
03                  CC0275     .LOC 275  ;   MAP ADDRESS  77
04
05
06                          ;
07                          ; INV                        INVERTION                    X -> 1/X
08                          ;
09                          ; CPU HAS TESTED FOR X = 0
10                          ;
11                          ; INPUT     DIVISOR
12                          ;           EXPONENT
13                          ;
14                          ; OUTPUT    36 BIT ROUNDED FRACTION TO  Y-OUT
15                          ;           CORRESPONDING EXPONENT  TO EXP-OUT
16                          ;
17
18 CC077 7777707170C70371 INV: BUSALU DODF,FRAC,Y,C00,LOADA,LOADF ; Y := MRF := DIVISOR;
19                          ;
20                          ; IF NEGATIV DIVISOR GOTO NEGDI
21 CC1C0 1532613071C60373 EXEC NEGDI,ALU0,IVT,EXP,X,MULDF,B00,LOADA,LOADI ; IEX:=DIVEXP; X:=R(Y(0:11));
22                          ;
23 CC101 4C17717772260372 FD0: IMID K2048,DESDF,A11,LOADA,LOADI ; IFRT := 1B0;
24                          ;
25                          ; PRODUCE MASK 1111000...0 IN H1
26                          ; 0.5<= A0 < 1,   -1<= R < -C.5
27                          ; FIRST PRODUCT : A1 = A0 * R;  -0.5<= A1 < -0.5 + 2 **(-11)
28 CC1C2 4C17717662330372 IEXEC K2C48,DODF,BUSDF,RESET,X1,C11,SHAD,LOADI ; MULT:=R(8:15)*Y; H1:=11000;
29 CC103 7777715772336221 BUSALU DODF,NACC,DESDF,C11,SHAD,PAS.A ; MULT:=R(0:7)*Y; H1:=11100; SUM:=MULT;
30 CC104 7777735772336221 BUSALU DODF,ACC,DESDF,C11,SHAD,PAS.A ; SUM :=MULT+SUM;   H1:= 11110...;
31 CC105 7777756774570377 BUSALU INM,INT,DESDF,C22,SHAU,LOADE ; H2 := (SUM(36:47) EXTRACT 8)*2;
32 CC1C6 7777757774230051 BUSALU INM,BUSDF,DESDF,C12,NLOAD,ADD ; ALU0 := -, CARRY;
33 CC107 2232755770160371 EXEC FD1,ALUC,INM,SUM,DESDF,MULDF,A00,SHAU,LOADF ; IFRH := SUM(1:39);
34                                    ; IF NOCARRY GOTO FD1;
35 CC110 7777717770C61221 ALU A00,LOADA,INC.R ; IFRH := IFRH + CARRY;
36                          ;
37                          ; M0 = -1 - A1;   -0.5-2**(-11)< M0 <= -0.5
38 CC111 7777717172165051 FD1: BUSALU DODF,BUSDF,Y,AC1,SHAU,SUB ; Y := 1B0 - IFRH; IFRH := (-2 * IFRH);
39                          ;
40                          ; SECOND PRODUCT : B2 = R * M0; 1/4< B2 <= 1/2+2**(-11)
41 CC112 7777717761126221 EXEC JMPDF,NJMP,DODF,BUSDF,DESDF,X1,B00,SHAD,PAS.A ; IEX:=IEX/2; MULT:=R(8:15)*Y;
42 CC113 7777715771C65221 BUSALU DODF,NACC,DESDF,B00,LOADA,NEG.A ; IEX:=-IEX; MULT:=R(0:7)*Y; SUM:=MULT;
43 CC114 7777735070C25221 BUSALU DODF,ACC,X,A00,NLOAD,NEG.A ; SUM := MULT + SUM; X := -IFRH;
44 CC115 7777756774570377 BUSALU INM,INT,DESDF,C22,SHAU,LOADE ; H2 := (SUM(36:47) EXTRACT 8)*2;
45 CC116 5713757774230051 EXEC INV1,JMP,INM,BUSDF,DESDF,MULDF,C12,NLOAD,ADD ; ALU0:=-,CARRY; GOTO INV1;
```

```
!0013 DRU
01                          FILL 117
02                  CC0001 .LIST 1
03                  CC0355     .LOC 355    ;   MAP ADDRESS 117
04
05                         ;
06                         ; STR
07                         ;
08                         ;        AFR AND AEX IS UNCHANGED !!!!!!!
09                         ;
10                         ;              INPUT                FUNCTION       OUTPUT
11                         ;       FRAC(0:11) = 0              DOUBLESTORE     Y-OUT = TAILPART  EXPO = HEAD_EXPO
12                         ;       FRAC(0:35) = DOUBLE ROUNDCONSTANT 1B31      Y-OUT = HEADPART  EXPO = HEAD_EXPO
13                         ;
14                         ;       FRAC(0:11) = -1           SINGLE STORE      Y-OUT = HEADPART 35 BIT ROUNDED
15                         ;       FRAC(12:35) = 0                             EXPO = HEAD_EXPO
16
17 CC117 7777717773C66221 STR: ALU B01,LOADA,PAS.A ; IEX := AEX;
18 CC120 0057717677720372 IMID,K1,RESET,B33,SHAD,LOADI ; DIA := (1B11) SHIFT (-1)
19 CC121 7777717776426051 ALU A23,NLOAD,OR ; ALU := AFR;
20 CC122 3330307772320371 ALUJMP STZMK,ZERO,FRAC,DESDF,A11,SHAD,LOADF ; IF ZERO GOTO STORE ZEROMARK;
21                         ;           IFRT := (DOUBLE_ROUND_CONSTANT OR ZERO) // 2;
22 CC123 2534317777770051 ALUJMP STR6,NZERO,BUSDF,DESDF,D33,SHAU,ADD ; WRKMODUS := WRKMODUS SHIFT 2;
23                         ;           IF DOUBLESTORE THEN GOTO STR6;
24 CC124 7152717775416221 ALUJMP BLST1,ALU0,BUSDF,DESDF,D22,LOADQ,PAS.A ; Q := BL_EXP;
25                                     ; IF BLOCKFL THEN GOTO BLSTR;
26 CC125 7777717774C66221 STR6: ALU AC2,LOADA,PAS.A ; IFRH := AFRH;
27 CC126 2653317776C06221 ALUJMP STR2,ALU10,BUSDF,DESDF,A33,LOADQ,PAS.A ; Q := AFRT;
28                                     ; IF NORMALIZED GOTO STR2
29                         ;
30                         ; PRENORMALIZE
31 CC127 7777717770146221 STR1: ALU ACC,LSHAU,PAS.A ; SHIFTLEFT IFR
32 CC130 7777717770C26221 ALU A00,NLOAD,PAS.A ; TEST NORMALIZED
33 CC131 2577317777C63051 ALUJMP STR1,NALU10,BUSDF,DESDF,B03,LOADA,ISUB ; IEX := IEX -1;
34                                     ; IF UNNORMALIZED GOTO STR1
35                         ;
36                         ; DECIDE SINGLE OR DOUBLE STORE
37 CC132 7777717772226221 STR2: ALU A11,NLOAD,PAS.A ; ALU := IFRT <* DOUBLE/SINGLE *>
38 CC133 6630317770115051 ALUJMP SRND,ZERO,BUSDF,DESDF,C00,LSHAD,CLEAR ; Q := 0 CON Q SHIFT (-1);
39                                     ; IF ZERO GOTO SINGLESTORE;
40 CC134 7777717772360001 ALU A11,SHAU,ADDQA ; IFRT := (Q + ROUND) * 2;
```

```
10014 DPU
01                          ;
02                          ; TEST CARRY
03 CC135 2776717772206221 ALUJMP STR3,NALU0,BUSDF,DESDF,A11,LOADQ,PAS.A ; Q := IFRT;
04 CC136 7777717570061221 BUSALU DODF,BUSDF,OVRA,A00,LOADA,INC.R ; IFRH := IFRH + CARRY;
05 CC137 7055317770026221 STR3: ALUJMP DRGT,OVR?,BUSDF,DESDF,A00,NLOAD,PAS.A ; IF OVERFLOW GOTO DRGT;
06 CC140 7117317771026221 ALUJMP DLFT,NALU10,BUSDF,DESDF,B00,NLOAD,PAS.A ; IF NOT_NORMALIZED GOTO DLFT;
07 CC141 C6733172710C0051 STR4: ALUJMP,DEOVF,ALU10,BUSDF,EXP0,B00,NLOAD,ADD ; OUTEXP0 := IEX * 2;
08                                  ; IF EXP_OVF GOTOEOVF;
09                          ;
10                          ; SET TAILPART
11 CC142 7777717770106221 ALU A00,LSHAD,PAS.A ; TAILPART := IFRH(36:39)
12 CC143 7777717770106221 ALU A00,LSHAD,PAS.A ;           CON IFRT(0:35);
13 CC144 7777717770106221 ALU A00,LSHAD,PAS.A ; IFRH := IFRH SHIFT (-4);
14 CC145 7777717770106221 ALU A00,LSHAD,PAS.A ;
15 CC146 7777717772305051 ALU A11,LSHAD,CLEAR ; TAILPART := 0 CON TAILPART SHIFT (-1);
16 CC147 7777517172236101 BUSALU READY,BUSDF,Y,C11,NLOAD,PAS.Q ; Y_OUT := Q;
17                          ;
18                          ; SET HEADPART
19 CC150 7777717770160051 ALU A00,SHAU,ADD ;
20 CC151 7777717770160051 ALU A00,SHAU,ADD ; HEADPART := IFRH SHIFT 4;
21 CC152 3256117770C35051 STRW: EXEC STRW,NSTART,READY,BUSDF,DESDF,MULDF,C00,NLOAD,CLEAR ; WAIT
22 CC153 7777717670C35051 BUSALU DCDF,BUSDF,RESET,C00,NLOAD,CLEAR ; RESET
23 CC154 0033517170C26221 EXEC NEXT,JMP,READY,BUSDF,Y,MULDF,A00,NLOAD,PAS.A ; Y-OUT := HEADPART; GOTO NEXT;
24
25                          ; STORE ZERO MARK
26 CC155 0270317270C25051 STZMK: ALUJMP STRZ,ZERO,BUSDF,EXP0,A00,NLOAD,CLEAR ; EXP_OUT := 0;
27                          ; IF SINGLE STORE GOTO STRZ;
28 CC156 0033517173235221 STZM1: EXEC NEXT,JMP,READY,BUSDF,Y,MULDF,D11,NLOAD,NEG.A ; Y_OUT := -74;
29                          ; GOTO NEXT
```

```
!0C15 GPU
01                          FILL 157
02                  CC0001 .LIST 1
03                  CC0515     .LOC 515    ;   MAP ADDRESS 157
04                         ;
05                         ; SHORT LOAD (LOAD MULTIPLIKATOR)
06 CC157 7777707170070371 SL: BUSALU DODF,FRAC,Y,C00,LOADA,LOADF ; LOAD MRF;
07 CC160 3433513775520373 EXEC MA,JMP,READY,EXP,DESDF,MULDF,B22,SHAD,LOADI ; LOAD MRE
08                         ;
09                         ; MULTIPLY AND ADD/SUB                              MA    FUNCTION   03
10                         ;
11                         ; INPUT           FRAC(0:35),EXP(36:47)  TIL REG A0, B0
12                         ; MULTIPLICATOR   FRAC(0:35),EXP(36:47)   I  REG C0, B2
13                         ;
14                         ; ZERO MULTIPLICATION MUST BE AVOIDED BY TEST IN GPU
15                         ; PRODUCT      A0 CON A1 = CC * A0
16                         ; EXP PRODUCT         B0 = BC + B2
17                         ;
18                         ; A0 IS NORMALIZED
19                         ; A1 IS PLACED LEFT
20                         ;
21                         ; EXPONENTS ARE HALF THE USUAL SIZE I.E. (0:12)
22                         ;
23 CC161 3436107072270371 MA: EXEC MA,NSTART,READY,FRAC,X,MULDF,C11,LOADA,LOADF ; X := H1 := FRAC
24 CC162 7777713671120373 BUSALU DODF,EXP,RESET,B00,SHAD,LOADI ; IEX := EXP // 2
25 CC163 7777717735C60051 MA1: EXEC JMPDF,NJMP,DODF,BUSDF,DESDF,X4,B02,LOADA,ADD ; IEX:=IEX+MRE; MULT:=X4*Y;
26 CC164 0073315771466221 ALUJMP MEXC,ALU10,NACC,DESDF,B20,LOADA,PAS.A ; MRE:=IEX; MULT:=X3*Y; SUM:=MULT;
27                         ;
28                         ; IF EXP_OVERFLOW GOTO EXCEPTION
29 CC165 7777735773463051 RTN: BUSALU DODF,ACC,DESDF,B21,LOADA,ISUB ; MRE:=MRE-AEX; SUM:=SUM+MULT;
30 CC166 7777756756770377 EXEC JMPDF,NJMP,INM,INT,DESDF,X2,C33,SHAU,LOADE ; H3:=SUM(40:47)*2; MULT:=X2*Y;
31 CC167 7777735770C25051 BUSALU DODF,ACC,DESDF,A00,NLOAD,CLEAR ; SUM := SUM + MULT;
32 CC170 7777756764570377 EXEC JMPDF,NJMP,INM,INT,DESDF,X1,C22,SHAU,LOADE ; H2:=SUM(40:47)*2; MULT:=X1*Y;
33 CC171 7777735772230051 EXEC JMPDF,NJMP,DODF,ACC,DESDF,MULDF,C11,NLOAD,ADD ; MULT:=X0*Y; SUM:=SUM+MULT;
34                         ;
35                         ; TEST FRAC == -1 <*FLOATING NOTATION*> IF FRAC SHIFT 1 = 0 THEN GOTO NEGMR;
36 CC172 4250356762370377 EXEC NEGMR,ZERO,INM,INT,DESDF,X1,C11,SHAU,LOADE ; H1:=SUM(40:47)*2; MULT:=X1*Y;
37 CC173 7777757071C36221 BUSALU INM,BUSDF,X,D00,NLOAD,PAS.A ; MULT := X0 * Y; X := 1B16;
38 CC174 7777735176630051 BUSALU DODF,ACC,Y,C33,NLOAD,ADD ; SUM := SUM + MULT; Y := H3 * 2;
39 CC175 7777756772360373 BUSALU INM,INT,DESDF,A11,SHAU,LOADI ; IFRT:= SUM(36:47)*2;
40 CC176 7777755770160371 BUSALU INM,SUM,DESDF,A00,SHAU,LOADF ; IFRH(0:39) := SUM(1:39) * 2;
41                         ;
```

```
!0016 DPU
01                          ; REARRANGING PRODUCT :
02                          ; A0 := A0(1:39) CON A1(3);
03                          ; A1 := A1(4:10) CON H1(3:10) CON H2(3:10) CON H3(3:10);
04                          ;
05 CC177 7777717154430051 EXEC JMPDF,NJMP,DODF,BUSDF,Y,X2,C22,NLOAD,ADD ; Y := H2*2; MULT := X2 * H3;
06 CC200 7777715152230051 EXEC JMPDF,NJMP,DODF,NACC,Y,X2,C11,NLOAD,ADD ; Y:=H1*2; MULT:=X2*H2; SUM:=MULT;
07 CC201 7777735152220051 EXEC JMPDF,NJMP,DODF,ACC,Y,X2,A11,NLOAD,ADD ; Y := IFRT*2; MULT := X2 * H1;
08                                      ; SUM := SUM + MULT;
09 CC202 7777735755766221 EXEC JMPDF,NJMP,DODF,ACC,DESDF,X2,B32,SHAU,PAS.A ; MULT := X2*IFRT
10                                        ;  SUM := SUM + MULT; DIA := 2 * MRE;
11 CC203 7777735777637777 MULT ACC,MULDF ; SUM := SUM + MULT;
12 CC204 7777715772360371 BUSALU DCDF,SUM,DESDF,A11,SHAU,LOADF ; IFRT(0:33) := SUM(1:34);
13 CC205 7777717772366221 ALU A11,SHAU,PAS.A ; IFRT(0:32) := SUM(2:34);
14 CC206 7777717772360051 ALU A11,SHAU,ADD ; IFRT(0:30) := SUM(4:34);
15 CC207 4416717775406221 ALUJMP FA0,NALU0,BUSDF,DESDF,B22,LOADQ,PAS.A ; Q := MRE ; IF NO_CARY GOTO FAC;
16 CC210 7777717770061221 ALU A00,LOADA,INC.R ; IFRH := IFRH + 1;
17 CC211 4413717775406221 ALUJMP FA0,JMP,BUSDF,DESDF,B22,LOADQ,PAS.A ; Q := MRE; GOTO FAC;
18                          ;
19                          ; PRODUCT UNNORMALIZED : 1 < ABS(IFR) <= 2**(-2) , NEED ONE LSHIFT IN
20                          ; 38.6 PCT OF ALL CASES
21
22                          ;
23                          ; MULT BY FRAC = -1; DIA AND IEX HAS BEEN CALCULATED;
24 CC212 7777717770016221 NEGMR: ALU C00,LOADQ,PAS.A ; Q := MRF
25 CC213 7777717770C66101 ALU A00,LOADA,PAS.Q ; IFRH := MRF
26 CC214 4017717777673260 IMID K2C48,DESDF,D33,LOADA,SUB.E ; CHANGE SIGN IN WRKMODUS
27 CC215 7777717772265051 ALU A11,LOADA,CLEAR ; IFRT := 0;
28 CC216 7777717773CC3051 FA00: ALU BC1,LOADQ,ISUB ; Q := IEX - AEX;
29 CC217 4770317777766101 ALUJMP FA10C,ZERO,BUSDF,DESDF,B33,SHAU,PAS.Q ; DIA := Q*2; IF ZERO GOTO FA1CC;
```

```
10017 FAU
01                          ;
02                          ;   ALIGNMENT
03                          ;
04                          ; INPUT :     IFR     A0A1(0:71)
05                          ;   -         AFR     A2A3(0:71)
06                          ;
07 CC220 4536717773230001 FA0: ALUJMP FA101,NALU0,BUSDF,DESDF,D11,NLOAD,ADDQA ; ALU := DIA + 74;
08                                  ; IF DIA >= 0 THEN GOTO FA101;
09 CC221 0032717772206221 ALUJMP NEXT,ALU0,BUSDF,DESDF,A11,LOADQ,PAS.A ; Q := IFRT; IF NEG GOTO NEXT;
10                          ;
11                          ; AEX > IEX,  BUT NOT MUCH
12 CC222 7777717777620623 ALU B33,NLOAD,INC.A ; ALC := DIA + 1
13 CC223 4470717770106221 FA7: ALUJMP FA7,NCZERO,BUSDF,DESDF,A00,LSHAD,PAS.A ; SHIFT IFR RIGHT UNTIL ALC = 0
14 CC224 5013717776726221 FA102: ALUJMP FA105,JMP,BUSDF,DESDF,A33,SHAD,PAS.A ; GOTO FA105;  AFRT:=AFRT/2;
15                          ;
16                          ; IEX >= AEX   OR   IEX >> AEX  (DIA >= 0)
17 CC225 7777717777626101 FA101: ALU B33,NLOAD,PAS.Q ; ALU := Q;
18 CC226 4770317771266221 ALUJMP FA100,ZERO,BUSDF,DESDF,B10,LOADA,PAS.A ; IF Q=0 GOTO FA100; AEX := IEX
19 CC227 7777717773235001 ALU D11,NLOAD,SUBAQ ; 74 - DIA (SYMMETRY IN ALIGNMENT)
20 CC230 4732717776606221 ALUJMP FA2,ALU0,BUSDF,DESDF,A33,LOADQ,PAS.A ; IF IEX >> AEX GOTO FA2  Q := AFRT;
21                          ; IEX > AEX, BUT NOT MUCH
22 CC231 7777717777665221 ALU B33,LOADA,NEG.A
23 CC232 7777717777620623 ALU B33,NLOAD,INC.A ; ALC := - DIA + 1
24 CC233 4670717774506221 FA5: ALUJMP FA5,NCZERO,BUSDF,DESDF,A22,LSHAD,PAS.A ; SHIFT AFR RIGHT UNTIL ALC = 0
25 CC234 4773717776666101 ALUJMP FA100,JMP,BUSDF,DESDF,A33,LOADA,PAS.Q ;  AFRT := Q  GOTO +/- ADD
26 CC235 7777717776665051 FA2: ALU A33,LOADA,CLEAR ; AFR := 0
27 CC236 4773717774465051 ALUJMP FA100,JMP,BUSDF,DESDF,A22,LOADA,CLEAR ;  -   -
```

```
10018 DPU
01
02                          ;
03                          ; ADD / SUB
04                          ;
05                          ; CLAIM : OVERFLOWA = 0 AND Q = AFRT
06                          ;
07                          ; INPUT  FRACTIONS AFR (A2A3)  AND IFR (A0A1)  ARE ALLIGNED AND IN COMPACKED NOTATION
08                          ;                  WITH ONLY ONE SIGN.
09                          ; INPUT  EXPONENT  AEX (B1)
10                          ;
11                          ; OUTPUT  FRACTION AFR (A2A3) = AFR +/- IFR IS NORMALIZED AND ROUNDED
12                          ; OUTPUT  EXPONENT AEX (B1) IS ADJUSTED ACCORDINGLY
13                          ;
14                          ; SHIFT TAILPART ONE TO THE RIGHT
15 CC237 4513717772206221 FA100: ALUJMP FA102,JMP,BUSDF,DESDF,A11,LOADQ,PAS.A ; Q := IFRT; GOTO FA102;
16 CC240 5056717771105051 FA105: ALUJMP FA110,NALU0,BUSDF,DESDF,B00,LSHAD,CLEAR ; IEX := 0;
17                                   ; Q(0:39) := 0 CON Q(0:38);
18 CC241 4017717776663260 IMID K2C48,DESDF,A33,LOADA,SUB.E ; AFRT(0) := 0;
19                          ;
20                          ; DECIDE + OR - ON MODUS, WHICH IS SHIFTED ONCE UNTIL NOW
21                          ; MODUS(1) = 0 GIVE ADD
22                          ; MODUS(1) = 1 GIVE SUB
23 CC242 7777717777776221 FA110: ALU D33,SHAU,PAS.A ; WRKMODUS := WRKMODUS * 2;
24 CC243 5532717771120623 ALUJMP FS,ALU0,BUSDF,DESDF,BC0,SHAD,INC.A ; IEX := 1B11 SHIFT (-1);
25                          ;
26                          ;  ADD:
27 CC244 7777717776760001 ALU A33,SHAU,ADDQA ; AFRT := (AFRT+IFRT) * 2;
28 CC245 5172717777637777 JUMP FA111,ALU0 ; IF CARRY THEN GOTO FA111;
29 CC246 5213717570460051 ALUJMP FA112,JMP,BUSDF,OVRA,A20,LOADA,ADD ; AFRH := AFRH+IFRH; GOTO FA112;
30 CC247 7777717570461051 FA111: BUSALU,DODF,BUSDF,OVRA,A20,LOADA,ADD1 ; AFRH := AFRH+IFRH+1;
31                          ;
32                          ; TEST OVERFLOW. IF OVERFLOW THEN GOTO SHIFT RIGHT AND INCREASE EXP;
33 CC250 5635317776426051 FA112: ALUJMP FA116,OVR?,BUSDF,DESDF,A23,NLOAD,OR ; AFRH OR AFRT; IF OVR GOTO FA116;
34 CC251 1430317777776221 FA113: ALUJMP ARZ0,ZERO,BUSDF,DESDF,D33,SHAU,PAS.A ; WRKMODUS SHIFT 1;
35                                   ; IF AFR = 0 GOTO SETZERO;
36                          ;
37                          ; DECIDE BLOCKFLOATING OR NOT ON MODUS, WHICH IS SHIFTED TWICE UNTIL NOW
38                          ; MODUS(2) = 0 GIVE NORMALIZATION
39                          ; MODUS(2) = 1 GIVE BLOCKFLOATING (I.E. GOTO FA118)
40 CC252 5332717773226221 ALUJMP FA118,ALU0,BUSDF,DESDF,B11,NLOAD,PAS.A ; ALU := AEX;
41 CC253 7777717774426221 ALU A22,NLOAD,PAS.A ; ALU := AFRH;
```

```
10019
01                          ;
02                          ; IF NOT NORMALIZED THEN GOTO NORMALIZE ELSE
03                          ; IF NOT_EXPOVERFLOW THEN GOTO NEXT ELSE GOTO ZZERO;
04 CC254 5377317773226221 ALUJMP FA114,NALU10,BUSDF,DESDF,B11,NLOAD,PAS.A ; ALU := AEX;  TEST_NORMALIZED;
05 CC255 0037317773226221 FA118: ALUJMP NEXT,NALU10,BUSDF,DESDF,B11,NLOAD,PAS.A ; ALU:=AEX; TEST EXPOVERFLOW
06 CC256 1453717773226221 ALUJMP ARZ,JMP,BUSDF,DESDF,B11,NLOAD,PAS.A ; GOTO ARZ; ALU := AEX;
07                          ;
08                          ; NORMALIZE
09 CC257 7777717776C6221 FA114: ALU A33,LOADQ,PAS.A ; Q := AFRT;
10 CC260 7777717774546221 FA115: ALU  A22,LSHAU,PAS.A ; AFRQ := AFRQ SHIFT 1
11 CC261 7777717774426221 ALU A22,NLOAD,PAS.A ; ALU := AFRH
12 CC262 5417317771263051 ALUJMP FA115,NALU10,BUSDF,DESDF,B10,LOADA,ISUB ; AEX := AEX - 1;
13                                              ; IF UNNORMALIZED GOTO FA115;
14 CC263 0037317776666101 FA117: ALUJMP NEXT,NALU10,BUSDF,DESDF,A33,LOADA,PAS.Q ; AFRT := Q;
15                                              ; IF NOT_EXPOVERFLOW GOTO NEXT
16                          ;
17                          ; TEST EXP OVF OR UNDERFLOW
18 CC264 1453717773226221 ALUJMP ARZ,JMP,BUSDF,DESDF,B11,NLOAD,PAS.A ; ALU := AEX;
19                          ;
20                          ; SUB:
21 CC265 7777717776765001 FS: ALU A33,SHAU,SUBAQ ; AFRT := (AFRT - Q) * 2;
22 CC266 5616717777637777 JUMP FS1,NALU0 ; IF CARRY THEN GOTO FS1;
23 CC267 5213717570462051 ALUJMP FA112,JMP,BUSDF,OVRA,A20,LOADA,ISUB1 ; AFRH := AFRH-IFRH-1; GOTO FA112;
24 CC270 5213717570463051 FS1: ALUJMP FA112,JMP,BUSDF,OVRA,A20,LOADA,ISUB ; AFRH := AFRH-IFRH; GOTO FA112;
25 CC271 7777717776C6221 FA116: ALU A33,LOADQ,PAS.A ; Q := AFRT;
26 CC272 7777717774506221 ALU A22,LSHAD,PAS.A ; AFRHQ := -,AFRH(0) CON AFRHQ SHIFT (-1)
27 CC273 5473717671260051 ALUJMP FA117,JMP,BUSDF,RESET,B10,LOADA,ADD ; AEX := AEX + 1; GOTO FA117;
```

```
10020 DPU
01                          ;
02                          ; INV CONTINUED
03                          ; **************
04 CC274 5752755770160371 INV1: EXEC FD2,ALU0,INM,SUM,DESDF,MULDF,A00,SHAU,LOADF ; IFRH := SUM(1:39)*2;
05                                          ; IF NOCARRY THEN GOTO FD2;
06 CC275 7777717770061221 ALU A00,LOADA,INC.R ; IFRH := IFRH + CARRY;
07                          ;
08                          ; THIRD PRODUCT : A2 = (2*A1) * M0; 0.5-2**(-20)< A2 <= 0.5;
09 CC276 7777717737637777 FD2: MULT BUSDF,X4 ; MULT := X4*Y;
10 CC277 7777715777637777 MULT NACC,MULDF ; MULT := X3*Y; SUM := MULT;
11 CC300 7777735757637777 MULT ACC,X2 ; MULT := X2*Y; SUM := MULT + SUM;
12 CC301 7777735767637777 MULT ACC,X1 ; MULT := X1*Y; SUM := MULT + SUM;
13 CC302 7777735777637777 MULT ACC,MULDF ; MULT := X0*Y; SUM := MULT + SUM;
14 CC303 7777735070C26221 BUSALU DCDF,ACC,X,A00,NLOAD,PAS.A ; SUM := MULT + SUM; X := B2;
15 CC304 7777756774570377 BUSALU INM,INT,DESDF,C22,SHAU,LOADE ; H2 :=E.T.C.
16 CC305 7777757774230051 BUSALU INM,BUSDF,DESDF,C12,NLOAD,ADD ;
17 CC306 6212755770160371 EXEC FD3,ALU0,INM,SUM,DESDF,MULDF,A00,SHAU,LOADF ;
18 CC307 7777717770061221 ALU A00,LOADA,INC.R ;
19 CC310 7777717172C25051 FD3: BUSALU DODF,BUSDF,Y,A01,NLOAD,SUB ; M1 := 1B0 - A2;
20                          ;
21                          ; M1 = 1 - A2;  0.5 <= M1 < 0.5+2**(-20)
22                          ; FOURTH PRODUCT : B3 = B2 * M1;  1/8 < B3 < 1/4+2**(-12)+2**(-21)-2**(-41)
23 CC311 7777717732265051 EXEC JMPDF,NJMP,DODF,BUSDF,DESDF,X4,A11,LOADA,CLEAR ; MULT := X4*Y; IFRT := C;
24 CC312 7777715777637777 MULT NACC,MULDF ; MULT := X3*Y; SUM := MULT;
25 CC313 7777735757665051 EXEC JMPDF,NJMP,DODF,ACC,DESDF,X2,B33,LOADA,CLEAR ; MULT:=X2*Y; SUM:=MULT+SUM; DIA:=0;
26 CC314 7777735767660623 EXEC JMPDF,NJMP,DODF,ACC,DESDF,X1,B33,LOADA,INC.A ; MULT:=X1*Y; SUM:=MULT+SUM; DIA:=2;
27 CC315 7777735777C60051 BUSALU DCDF,ACC,DESDF,B03,LOADA,ADD ; MULT:=X0*Y; SUM:=MULT+SUM; IEX:=IEX+2;
28 CC316 7777735777726221 BUSALU DCDF,ACC,DESDF,B33,SHAD,PAS.A ; SUM := MULT + SUM; DIA := DIA/2;
29 CC317 7777755770160371 BUSALU INM,SUM,DESDF,A00,SHAU,LOADF ; IFRH := SUM;
30 CC320 7777757770C36221 BUSALU INM,BUSDF,DESDF,C00,NLOAD,PAS.A ; TEST NEGATIVE DIVISOR;
31 CC321 6752757770CC0051 EXEC CARRY,ALU0,INM,BUSDF,DESDF,MULDF,A00,LOADQ,ADD ; Q := 1/DIVISOR;
32 CC322 6513317770166101 SNRM0: ALUJMP SRND0,ALU10,BUSDF,DESDF,A00,SHAU,PAS.Q ; IF NORMALIZED GOTO SHORTROUND;
```

```
10021
01                          ;
02                          ; NORMALIZE IFRH.  ONLY ONE SHIFT IS NEEDED !!! AND WAS DONE IN LAST MICRO
03 CC323 6573717777C63051 ALUJMP,TBLFL,JMP,BUSDF,DESDF,B03,LOADA,ISUB ; IEX := IEX -1
04                          ;
05                          ; RESET MRF WHEN NO NORMALIZATIONSHIFTS
06 CC324 6573717770C66101 SRNDO: ALUJMP TBLFL,JMP,BUSDF,DESDF,A00,LOADA,PAS.Q ; IFRH := Q; GOTO BLFL;
07                          ;
08 CC325 0057717777720372 TBLFO: IMID K1,DESDF,B33,SHAD,LOADI ; DIA := 1;
09 CC326 4017717770C60372 IMID K2C48,DESDF,A00,LOADA,LOADI ; A00 := 1B0;
10                          ;
11                          ; TEST BLOCKFLOATING MODE
12 CC327 7777717777770051 TBLFL: ALU D33,SHAU,ADD ; ALU := MUDUS(2); WRKMODUS := WRKMODUS SHIFT 2;
13 CC330 7212717775416221 TBLF1: ALUJMP BLSTR,ALU0,BUSDF,DESDF,D22,LOADQ,PAS.A ; Q := BL_EXP;
14                                                ; IF BLOCKFL GOTO BLSTR;
15                          ;
16                          ; ROUND
17 CC331 0037717570060260 SRND: IMID,K36,OVRA,A00,LOADA,ADD.E ; IFRH := IFRH + 1B36;
18 CC332 1275317170C26221 ALUJMP SHRT,OVR?,BUSDF,Y,A00,NLOAD,PAS.A ; Y := IFRH; IF OVERFLOW GOTO SHRT;
19 CC333 1237317771C26221 ALUJMP SHLF,NALU10,BUSDF,DESDF,B00,NLOAD,PAS.A ; IF ALU0 = ALU1 GOTO SHLF;
20 CC334 0233317770C25051 FDUD: ALUJMP,EOVF,ALU10 BUSDF,DESDF,A00,NLOAD,CLEAR ; IF EXPOOVERFLOW GOTO EOVF;
21 CC335 0033517271C20051 EXEC NEXT,JMP,READY,BUSDF,EXPO,MULDF,B00,NLOAD,ADD ; EXPO_OUT := 2 * IEX;
22
23                          ; CARRY BY NEG DIVISOR
24 CC336 7777756774570377 CARRY: BLSALU INM,INT,DESDF,C22,SHAU,LOADE ; H2 := E.T.C.
25 CC337 7777717774230051 ALU C12,NLOAD,ADD ; CARRY
26 CC340 1632717770C61221 ALUJMP NEGDR,ALU0,BUSDF,DESDF,A00,LOADA,INC.R ; IF NOCARRY GOTO NEGDR;
27                                                ; IFRH := IFRH + 1;
28 CC341 1633717770CC0051 ALUJMP NEGDR,JMP,BUSDF,DESDF,A00,LOADQ,ADD ; Q := 1/DIVISOR; GOTO NEGDR;
```

```
!0022 DPU
01                          ;
02                          ; STR SUBROUTINE
03                          ; ***
04                          ; DOUBLE RIGHTSHIFT WHEN OVERFLOW IN IFRH
05 CC342 7777717770106221 DRGT: ALU ACC,LSHAD,PAS.A ;
06 CC343 3033717677C60051 ALUJMP STR4,JMP,BUSDF,RESET,B03,LOADA,ADD ; IEX := IEX + 1; GOTO STR4;
07                          ;
08                          ; DOUBLE LEFTSHIFT WHEN UNNORMALIZED
09 CC344 7777717770146221 DLFT: ALU ACC,LSHAU,PAS.A ;
10 CC345 3033717777C63051 ALUJMP STR4,JMP,BUSDF,DESDF,B03,LOADA,ISUB ; IEX := IEX -1; GOTO STR4;
11                          ;
12                          ;  BLOCKFLOATING STR
13                          ;  ******************
14 CC346 7777717774C66221 BLST1: ALU A02,LOADA,PAS.A ; IFRH := AFRH;
15 CC347 7777717776266221 ALU A13,LOADA,PAS.A ; IFRT := AFRT;
16 CC350 7777717775466101 BLSTR: ALU B22,LOADA,PAS.Q ; MRE := BL_EXP;
17 CC351 7777717775C23051 ALU B02,NLOAD,ISUB ; ALU := IEX - BL_EXP;
18 CC352 7716717772206221 ALUJMP BLBIG,NALUO,BUSDF,DESDF,A11,LOADQ,PAS.A ; Q := IFRT;
19                                                 ; IF EXP >= BL_EXP GOTO BLBIG
20                          ;
21                          ; BLOCKNORMALIZE (I.E. IEX := MRE)
22 CC353 7777717777C60051 BLNRM: ALU B03,LOADA,ADD ; REPEAT  :  IEX := IEX + 1;
23 CC354 7777717775C23051 ALU B02,NLOAD,ISUB ; ALU := IEX - BL_EXP;
24 CC355 7274317770106221 ALUJMP BLNRM,NZERO,BUSDF,DESDF,A00,LSHAD,PAS.A ; SHIFTRIGHT AFRHQ; UNTIL IEX=BLEXP;
25                          ;
26                          ; ROUND
27 CC356 0037717570C60260 BLRND: IMID K36,OVRA,A00,LOADA,ADD.E ; IFRH := IFRH + 1B36;
28                          ;
29                          ; CLEAR LAST FOUR BITS
30 CC357 7777717770126221 ALU A00,SHAD,PAS.A ;
31 CC360 7777717770126221 ALU A00,SHAD,PAS.A ; IFRH := IFRH SHIFT (-4);
32 CC361 7777717770126221 ALU A00,SHAD,PAS.A ;
33 CC362 7777717770126221 ALU A00,SHAD,PAS.A ;
34 CC363 7575317770160051 ALUJMP RLRH,OVR?,BUSDF,DESDF,A00,SHAU,ADD ;
35 CC364 0270317770160051 ALUJMP STRZ,ZERO,BUSDF,DESDF,A00,SHAU,ADD ; IF ZERO GOTO STRZ;
36 CC365 7777517170C26221 BUSALU READY,BUSDF,Y,A00,NLOAD,PAS.A ; Y_OUT := IFRH;
37 CC366 0033517271C20051 STR7: EXEC NEXT,JMP,READY,BUSDF,EXPO,MULDF,B00,NLOAD,ADD ; E_OUT:=IEX; GOTO NEXT;
```

```
!0023 DRU
01                          ;
02                          ; BLOCKRIGHT
03 CC367 7777717770166221 BLRH: ALU ACC,SHAU,PAS.A ; IFRH := IFRH SHIFT 1;
04 CC370 0276717677C60051 ALUJMP STRZ,NALU0,BUSDF,RESET,B03,LOADA,ADD ; IEX := IEX + 1;
05                                              ; IF ALU(C) = 0 THEN GOTO STRZ;
06 CC371 7553517170C26221 EXEC STR7,JMP,READY,BUSDF,Y,MULDF,A00,NLOAD,PAS.A ; Y := IFRH; GOTO STR7;
07                          ;
08                          ; BLOCK EXP BIG
09 CC372 7777717770C26221 BLBG0: ALU ACC,NLOAD,PAS.A ; ALU := IFRH;
10 CC373 7353317777C63051 ALUJMP BLRND,ALU10,BUSDF,DESDF,B03,LOADA,ISUB ; IEX := IEX - 1;
11                                              ; IF NORMALIZED GOTO BLRND;
12 CC374 7777717775C23051 BLBIG: ALU BC2,NLOAD,ISUB ; ALU := IEX - BL_EXP;
13 CC375 7350317772235051 ALUJMP BLRND,ZERO,BUSDF,DESDF,C11,NLOAD,CLEAR ; IF IEX = BL_EXP GOTO BLRND;
14 CC376 7653717770146221 ALUJMP BLBGC,JMP,BUSDF,DESDF,A00,LSHAU,PAS.A ; SHIFTLEFT IFRHQ; GOTO BLBG0;
15                          ;
16
17                            FILL .4C0
18                  CC0001 .LIST 1
19
20                            .END
21
00CC SOURCE LINES IN ERROR
```