The Lasergravure Impositioner

This note was inspired by my leaving presentation from FFEI on 2016-04-12, which was given by Bob Wilson. I was taken on by Crosfield in 1984 as a Principal Electronic Engineer, though Bob said I hadn't done any electronics. I did, for the first few years, before Bob arrived. Those years were spent on the New Electronics for Lasergravure and Gravure Plus, which I designed, taking some parts right through while others were detailed by the rest of the team: Gary Stewart, Pravin Tailor, Salim Mehta and Dave Thomas. There were a set of boards for the main Lasergravure data path, some smaller boards to control peripheral parts over a serial bus, and a second set of datapath boards for the Helioklishograph interfaces.

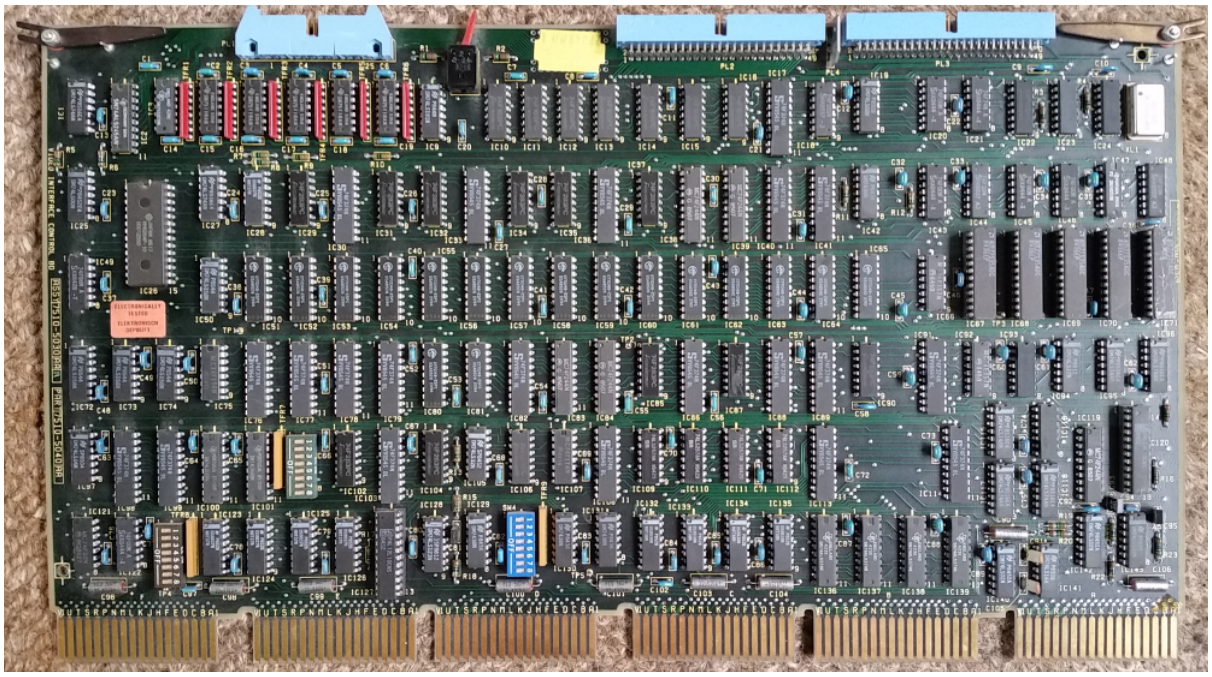
The core of the datapath for both Lasergravure and Gravure Plus was the Impositioner (also known earlier as the Video Interface).

The PCBs

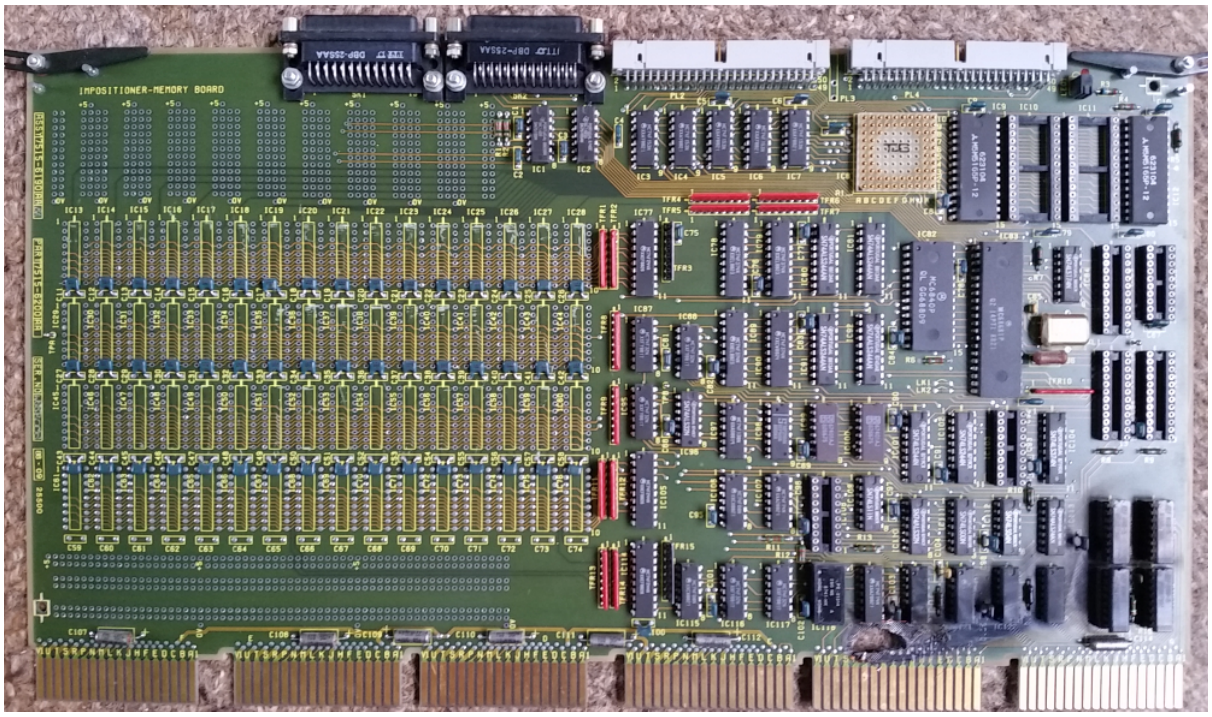
The Impositioner was the 1980s technology equivalent to the Caslon Inkjet FPE. It was two big cards in a PDP-11 backplane. Its job was to take in data from a set of images on a disk pack and to cut them up into a serial stream to feed to the engraver to produce imposed pages around the gravure cylinder, along the way adding registration marks, gutters and so on. The stream was then fed to the engraver electronics where the video output board (another of mine) did the detailed timing against the encoder and then digital-to-analogue conversion to drive the laser. I took one of the boards, the control, right through and Pravin did the detail on the other, the memory.

The Impositioner also included some of what is done by the re-order server on inkjet. It included 7M * 16-bit of DRAM and was able to take quite big chunks of images by DMA from the PDP backplane. It built the output stream using a versatile output DMA engine that followed a continually-extending linked list of control blocks, some of which referred to bits of input images and others to locally-generated data for marks and gaps. All this was controlled by an imposition plan that was downloaded at the start of engraving one cylinder.

The local processor that interpreted the plan and generated the control blocks was a 68000 which shared access to the DRAM with the DMA engine, on a devious scheme whereby the DMA got an access while the 68000 was getting its address ready, though this was very slow by modern standards with memory accessed at 1.5 MHz: 0.75 MHz for the CPU and 0.75 MHz for the DMA. The DMA engine ran a single chain of blocks for reading image parts in but up to 32 chains at once for output, in round-robin, to provide the data needed to engrave up to 8 heads simultaneously at arbitrary resolution to drive a Hell K202 mechanical engraver. The DMA engine was implemented in 74-series logic around a microcoded sequencer and a fast SRAM holding all of the counts and pointers. I think the DMA microcode ran at 9 MHz (3 states per DRAM cycle); the board has a 36 MHz crystal.



Impositioner control board (7510-5030) (this one is actually called a video interface and it was scrapped as the factory built it on the wrong issue PCB).



Impositioner memory board (7515-6190) (this one was set on fire by a faulty tantalum capacitor and had its memory chips recovered).

The microcode

The microcode was written in AMDASM, intended for AMD sequencers and bit-slice logic but easily used more generally. There were two files, one containing the definitions of the bit patterns within the control word and one setting out the actual instructions.

Definitions file

```
TITLE LASERGRAVURE VIDEO INTERFACE DEFINITIONS
WORD 40
;
; N.I.BROMLEY 85.10.23  1549
;
;
; FIELDS TO CONTROL THE RAM-BASED DMA ADDRESS GENERATOR
; =====================================================
;
; (ALL REFER TO THE CURRENT CHAIN)
;
NOPAD: DEF 8X,B#00000,27X
;
; NO OPERATION
;
NEXTAD: DEF 8X,B#01001,27X
;
; READS THE NEXT ADDRESS, WORD COUNT, ETC. OUT OF RAM AND SAVES
;  THE UPDATED ADDRESS AND DECREMENTED WORD COUNT IN THE UPDATE
;  REGISTER. DURING THE OPERATION OF THIS INSTRUCTION, THE
;  ADDRESS MAY BE LOADED INTO THE DRAM ADDRESS REGISTER USING
;  LOADWD
;
NEWAD: DEF 8X,B#10010,27X
;
; ENABLES THE CONTENTS OF THE TEMPORARY REGISTERS AND THE DATA BUS
;  TO THE DMA ADDRESS GENERATOR RAM AND WRITES IN A COMPLETE NEW
;  ENTRY
;
UPDATE: DEF 8X B#00110,27X
;
; WRITES THE CONTENTS OF THE UPDATE REGISTER INTO THE ADDRESS
;  GENERATOR RAM
;
LOADAD: DEF 8X,B#10000,27X
;
; ENABLES THE CONTENTS OF THE TEMPORARY REGISTERS AND THE DATA BUS
;  SO THAT THEY MAY BE LOADED INTO THE DRAM ADDRESS REGISTER
;  USING LOADWD
;
```

```
TESTWC: DEF 8X,B#00001,27X
;
; ENABLES THE CONTENTS OF THE ADDRESS RAM SO THAT THE OPERATION
;  COUNT UNDERFLOW CONDITION MAY BE SET UP WITHOUT CHANGING DATA
;  OR REGISTER CONTENTS
;
;
; FIELDS TO CONTROL THE DRAM ADDRESS COUNTER
; ==========================================
;
; (ALL OPERATE AFTER THE INSTRUCTION IN WHICH THEY APPEAR)
;
NOPWD: DEF 6X,B#01,32X
;
; NO OPERATION
;
LOADWD: DEF 6X,B#11,32X
;
; LOADS THE DRAM ADDRESS REGISTER FROM THE OUTPUT OF THE ADDRESS
;  GENERATOR RAM OR FROM THE TEMPORARY REGISTERS AND THE DATA BUS,
;  DEPENDING ON THE DMA ADDRESS GENERATOR CONTROL INSTRUCTION USED
;
NEXTWD: DEF 6X,B#10,32X
;
; ADVANCES THE DRAM ADDRESS TO THE NEXT WORD
;
LASTWD: DEX 6X,B#00.32X
;
; MOVES THE DRAM ADDRESS COUNTER TO THE PREVIOUS WORD
;
;
; FIELDS TO CONTROL THE OUTPUT CHANNEL POINTER
; ============================================
;
; (TAKE EFFECT AFTER THE INSTRUCTION IN WHICH THEY APPEAR)
;
CHNNOP: DEF 4X,B#1,35X
;
; NO OPERATION
;
CHNADV: DEF 4X,B#0,35X
;
; MOVE THE OUTPUT CHANNEL POINTER TO POINT TO THE NEXT OUTPUT CHANNEL
;  (COUNT LENGTH DEPENDS ON SETTING OF MOD REGISTER)
;
;
; DRAM DATA DIRECTION CONTROL
; ===========================
;
; MUST BE VALID THROUGHOUT A DRAM CYCLE
;
```

```
DRAMR: DEF 27X,B#1,12X
;
; READ DRAM
;
DRAMW: DEF 27X,B#0,12X
;
; WRITE DRAM
;
;
; DATA SOURCE FIELDS
; ==================
;
; THESE FIELDS ENABLE DATA ONTO THE DATA BUS
;
RDSCRA: DEF 14X,B#1,6X,B#00,1V,1X,B#00,13X
;
; READS FROM THE SCRATCHPAD MEMORY FOR THE CURRENT CHAIN (INPUT OR
;  CURRENT OUTPUT CHANNEL). THERE ARE TWO SCRATCHPAD LOCATIONS PER
;  CHAIN, SELECTED BY A QUALIFIER:
;
;   RDSCRA <WHICH>
;     WHERE <WHICH> IS ONE OF:
;                  UPPER
;                  LOWER
;
RDDEV: DEF 14X,B#0,6X,B#10,2X,B#00,13X
;
; READ FROM THE CURRENT INPUT DEVICE
;
RDDRAM: DEF 14X,B#0,6X,B#01,2X,B#00,13X
;
; READ FROM THE REGISTER WHICH HOLDS THE RESULT OF THE LAST
;  DRAM READ OPERATION
;
RDHPLO: DEF 14X,B#0,6X,B#00,2X,B#10,13X
RDHPHI: DEF 14X,B#0,6X,B#00,2X,B#01,13X
;
; READ THE HIGH OR LOW PART OF A FIXED CONSTANT REGISTER WHICH
;  POINTS AT THE START OF THE ABSOLUTE ADDRESS IN DRAM AT WHICH
;  THE INPUT AND OUTPUT HEADPOINTERS ARE STORED
;
NORD: DEF 14X,B#0,6X,B#00,2X,B#00,13X
;
; READ NOTHING
;
UPPER: EQU B#0
LOWER: EQU B#1
;
; VALUES FOR RDSCRA AND WRSCRA DEFINITIONS
;
;
```

```
; DATA DESTINATION FORMATS
; ========================
;
; THESE FIELDS SELECT A LOCATION TO BE WRITTEN WITH THE DATA ON THE
;  DATA BUS
;
WRSCRA: DEF 13X,B#1,1X,B#000000,2X,1V,16X
;
; WRITE TO THE SCRATCHPAD FOR THE CURRENT CHAIN. USED IN THE FORMAT:
;
;     WRSCRA <WHICH>
;     WHERE <WHICH> IS ONE OF:
;                  UPPER
;                  LOWER
;
WRDEV: DEF 13X,B#0,1X,B#100000,19X
;
; WRITE TO THE CURRENT OUTPUT DEVICE
;
WRDRAM: DEF 13X,B#0,1X,B#01000,19X
;
; WRITE TO THE REGISTER WHICH CONTAINS THE DATA TO BE USED IN THE
;  NEXT DRAM WRITE CYCLE (MAY BE WRITTEN AS LATE AS ONE INSTRUCTION
;  INTO A WRITE CYCLE)
;
WRMOD: DEF 13X,B#0,1X,B#001000,19X
;
; WRITE TO THE REGISTER CONTROLLING THE COUNT MODULUS OF THE OUTPUT
;  CHANNEL POINTER (COUNT CYCLE IS THIS VALUE + 1, MAX 32)
;
WRAHI: DEF 13X,B#0,1X,B#000100,19X
;
; WRITE TO THE TEMPORARY HOLDING REGISTER FOR ADDRESS INCREMENT (MS
;  8 BITS) AND HIGH PART OF DMA ADDRESS (LS 8 BITS)
;
WRCNT: DEF 13X,B#0,1X,B#000010,19X
;
; WRITE TO THE TEMPORARY HOLDING REGISTER FOR OPERATION COUNT
;
WRTC: DEF 13X,B#0,1X,B#000001,19X
;
; WRITE TO THE TEMPORARY HOLDING REGISTER FOR NEW-LINE (BIT 7)
;  AND TONE CURVE SELECT (LS 5 BITS)
;
NOWR: DEF 13x,B#0,1X,B#000000,19X
;
; WRITE NOTHING
;
;
; ACTION FIELDS
; =============
;
; A COLLECTION OF MISCELLANEOUS BITS, FOR WHICH ACTION SUPPLIES DEFAULT
;  INACTIVE STATES
;
; USE IN THE FORM:
;   ACTION <CBIT>,<IBIT>,<OUTPUT>,<INPUT>,<KIKRAM>
;
ACTION: DEF 1VB#1,1VB#0,1VB#1,1VB#1,20X,1VB#1,15X
;
; USE THE FOLLOWING VALUES TO CAUSE AN ACTION:
;
CBIT: EQU B#0
;
; WHEN THIS ACTION IS ENABLED, IF THE DATA BUS BIT 15 IS CLEAR THE
;  CURRENT DEVICE (INPUT OR OUTPUT) WILL BE SET "STOPPED". NORMALLY
;  THIS IS USED WHILE THE FIRST WORD OF A LINK ADDRESS IS ON THE
;  BUS (BIT 15 = CONTINUATION BIT)
;
IBIT: EQU B#1
;
; WHEN THIS ACTION IS ENABLED, IF THE DATA BUS BIT 14 IS SET, AN
;  INTERRUPT VECTORED BY THE BUS BITS 13..8 WILL BE QUEUED.
;  NORMALLY THIS IS USED WHILE THE FIRST WORD OF A LINK ADDRESS
;  IS ON THE BUS (BIT 14 = INTERRUPT BIT)
;
OUTPUT: EQU B#0
;
; WHEN THIS ACTION IS ENABLED, OUTPUT MODE IS SET AND HELD, AND ALL
;  SCRATCHPAD AND DMA ADDRESS GENERATOR REFERENCES WILL BE TO THE
;  CURRENT OUTPUT CHANNEL. THE NEW MODE IS VALID DURING THE INSTRUCTION
;  WITH THIS ACTION ENABLED
;
INPUT: EQU B#0
;
; WHEN THIS ACTION IS ENABLED, INPUT MODE IS SET AND HELD, AND ALL
;  SCRATCHPAD AND DMA ADDRESS GENERATOR REFERENCES WILL BE TO THE
;  INPUT CHAIN. THE NEW MODE IS VALID DURING THE INSTRUCTION WITH
;  THIS ACTION ENABLED
;
KIKRAM: EQU B#0
;
; WHEN THIS ACTION IS ENABLED A REQUEST IS ENTERED FOR A DRAM CYCLE.
;  THE REQUEST IS GENERATED DURING THE INSTRUCTION WITH THIS ACTION
;  ENABLED, AND THE NEXT INSTRUCTION COULD BE OVERLAPPING THE
;  CYCLE REQUESTED
;
; SEQUENCE CONTROL FIELDS
; ======================
;
JMP: DEF 28X,B#1,B#000,8V%
;
```

```
; TRANSFERS CONTROL TO THE SPECIFIED ADDRESS UNCONDITIONALLY
;     JMP <ADDRESS>
;
NEXT: DEF 28X,B#0,B#000,8X
;
; TRANSFERS CONTROL TO THE NEXT INSTRUCTION IN ADDRESS ORDER
;     NEXT
;
JMPIF: DEF 28X,B#0,3V,8V%
;
; TRANSFERS CONTROL TO THE SPECIFIED ADDRESS IF THE CONDITION
;  IS TRUE
;     JMPIF <CONDITION>,<ADDRESS>
;
JMPIFN: DEF 28X,B#1,3V,8V%
;
; TRANSFERS CONTROL TO THE SPECIFIED ADDRESS IF THE CONDITION
;  IS FALSE
;     JMPIFN <CONDITION>,<ADDRESS>
;
; VALUES FOR <CONDITION> ARE:
;
NEVER: EQU B#000
;
; ALWAYS FALSE
;
INNOTR: EQU B#001
;
; TRUE IF THE INPUT DEVICE IS NOT READY
;
OUNOTR: EQU B#010
;
; TRUE IF THE OUTPUT DEVICE IS NOT READY
;
WCNOTE: EQU B#011
;
; TRUE IF THE OPERATION COUNT IS NOT IN THE TERMINAL STATE. THE COUNT IS
;  ONLY VALID FOR THE ONE INSTRUCTION FOLLOWING A NEXTAD OR TESTWC
;  ADDRESS GENERATOR INSTRUCTION, AND IS TRUE IF THE OPERATION COUNT
;  READ OUT IS ZERO (THEREFORE THE OPERATION COUNT IS SET TO ONE
;  LESS THAN THE NUMBER OF OPERATIONS REQUIRED)
;
CHNOTE: EQU B#100
;
; TRUE IF THE CHANNEL COUNTER IS NOT IN THE TERMINAL STATE (ZERO).
:  THIS IS VALID FOR TESTING FROM THE NEXT BUT ONE INSTRUCTION AFTER
;  THE COUNTER IS CHANGED USING CHNADV
;
NOTEOC: EQU B#101
;
; TRUE IF THE DRAM HAS NOT COMPLETED A REQUESTED CYCLE. TO AVOID
;  PIPELINING DELAYS, THIS CONDITION IS VALID DURING THE LAST
;  INSTRUCTION BEFORE THE CYCLE FINISHES, AND ONLY THEN.
;
INPUTM: EQU B#110
;
; TRUE IF INPUT MODE IS SET
;
WAIT: DEF B#10111,1X,B#01000000000000000,1X,B#10010101,8V%
;
; DO NOTHING EXCEPT JUMP TO SPECIFIED ADDRESS IF MEMORY IS NOT READY.
;  USE IN FORM:   WAIT <ADDRESS>  WHERE <ADDRESS> IS USUALLY $.
;  EQUIVALENT TO NOPAD & NOPWD & CHNNOP & DRAMR & NORD & NOWR
:  & ACTION & JMPIF NOTEOC,<ADDRESS>
;
END
```

Microcode instructions

```
TITLE LASERGRAVURE VIDEO INTERFACE MICROCODE
;
; N.I.BROMLEY 85.11.27  1538
;
ORG 0
LIST B
NOLIST L
;
INIT:
;
; ENTRY POINT WHEN DMA CONTROLLER ENABLE BIT IS SET. THE CODE IS AN INFINITE
;  LOOP, THE ONLY EXIT IS VIA HARDWARE RESET WHEN THE CONTROLLER IS
;  DISABLED. WHEN THE CONTROLLER IS ENABLED ITS DATA MUST BE ALREADY
;  AVAILABLE IN DRAM, AND DMA DATA MUST BE AVAILABLE AS WELL IF THE OUTPUT
;  DEVICE IS ENABLED AND READY
;
;
      NOPAD & NOPWD & CHNNOP & DRAMR & NORD & NOWR & ACTION & JMP H#01
;
; THIS INSTRUCTION WILL BE EXECUTED REPEATEDLY AS IT WILL ALSO BE THE
;  INIT WORD IN THE PROMS, SO FOR SAFETY WE DO NOTHING IN IT. WITH THE
;  HARDWARE AS IT IS TODAY, THE COPY HERE WILL NEVER BE EXECUTED,
;  AS THE INIT WORD WILL EXECUTE UNTIL "GO" IS ASSERTED, THEN THE
;  NEXT INSTRUCTION WILL BE AT ADDRESS H#01. THIS INSTRUCTION IS KEPT
;  HERE AS A REMINDER OF INIT WORD, AND IN CASE I DECIDE THAT THE INIT
;  WORD IS TOO CUMBERSOME, AND I MODIFY THE HARDWARE TO SET ADDRESS
;  ZERO WHILE "GO" IS INACTIVE
;
;
      NOPAD & NOPWD & CHNNOP & DRAMR & RDHPHI & WRAHI
/     & ACTION ,,,INPUT & NEXT
```

```
;
; COPY THE HIGH PART OF THE ABSOLUTE ADDRESS VALUE INTO A TEMPORARY
;  REGISTER. SET INPUT MODE
;
      LOADAD & LOADWD & CHNNOP & DRAMR & RDHPLO & NOWR
/     & ACTION ,,,,KIKRAM & NEXT
;
; TRANSFER THE LOW PART OF THE ABSOLUTE ADDRESS TOGETHER WITH THE
; HIGH PART SAVED IN THE LAST INSTRUCTION TO THE DRAM ADDRESS
;  COUNTER, AND INITIATE A READ CYCLE
;
;
      WAIT $
;
; WAIT FOR READ TO COMPLETE
;
;
      NOPAD & NEXTWD & CHNNOP & DRAMR & RDDRAM & WRAHI
/     & ACTION ,,,,KIKRAM & NEXT
;
; SAVE DATA FETCHED (HIGH PART OF INPUT CHAIN HEAD POINTER) IN
;  TEMPORARY REGISTER, ADVANCE DRAM ADDRESS, AND START ANOTHER READ
;
;
      WAIT $
;
; WAIT FOR READ TO COMPLETE
;
;
      LOADAD & LOADWD & CHNNOP & DRAMR & RDDRAM & NOWR
/     & ACTION ,,,,KIKRAM & NEXT
;
; TRANSFER DATA FETCHED (LOW PART OF INPUT CHAIN HEAD POINTER) TOGETHER
;  WITH HIGH PART SAVED ABOVE INTO THE DRAM ADDRESS COUNTER. THIS NOW
;  POINTS TO THE FIRST WORD OF THE FIRST ELEMENT OF THE INPUT CHAIN
;  WHICH MUST NOW BE LOADED INTO THE DMA ADDRESS GENERATOR. INITIATE
;  READ
;
;
      WAIT $
;
; WAIT FOR READ TO COMPLETE
;
;
      NOPAD & NEXTWD & CHNNOP & DRAMR & RDDRAM & WRSCRA UPPER
/     & ACTION ,,,,KIKRAM & NEXT
;
; SAVE THE HIGH PART OF THE LINK FROM THE FIRST ELEMENT IN SCRATCHPAD
;  AREA FOR INPUT CHAIN, ADVANCE THE DRAM ADDRESS AND INITIATE A READ
;
;
```

```
      WAIT $
;
; WAIT FOR READ TO COMPLETE
;
;
      NOPAD & NEXTWD & CHNNOP & DRAMR & RDDRAM & WRSRRA LOWER
/     & ACTION ,,,,KIKRAM & NEXT
;
; SAVE THE LOW PART OF THE LINK IN SCRATCHPAD, ADVANCE THE DRAM ADDRESS
;  AND START ANOTHER READ (THIS IS GOING TO GET MONOTONOUS)
;
;
      WAIT $
;
; WAIT FOR READ TO COMPLETE
;
;
      NOPAD & NEXTWD & CHNNOP & DRAMR & RDDRAM & WRCNT
/     & ACTION ,,,,KIKRAM & NEXT
;
; SAVE THE OPERATION COUNT FOR THE FIRST INPUT ELEMENT IN A TEMPORARY
;  REGISTER, ADVANCE THE DRAM ADDRESS AND INITIATE A READ
;
;
      WAIT $
;
; WAIT FOR READ TO COMPLETE
;
;
      NOPAD & NEXTWD & CHNNOP & DRAMR & RDDRAM & WRTC
/     & ACTION ,,,,KIKRAM & NEXT
;
; SAVE THE DATA (ACTUALLY DON'T CARE FOR AN INPUT CHAIN) IN A TEMPORARY
;  REGISTER, ADVANCE THE DRAM ADDRESS AND START ANOTHER READ
;
;
      WAIT $
;
; WAIT FOR READ TO COMPLETE
;
;
      NOPAD & NEXTWD & CHNNOP & DRAMR & RDDRAM & WRAHI
/     & ACTION ,,,,KIKRAM & NEXT
;
; SAVE THE ADDRESS INCREMENT AND HIGH PART OF DMA ADDRESS FOR THE
;  FIRST INPUT ELEMENT IN A TEMPORARY REGISTER, ADVANCE THE DRAM
;  ADDRESS AND START ANOTHER READ
;
;
      WAIT $
;
```

```
; WAIT FOR READ TO COMPLETE
;
;
      NEWAD & NOPWD & CHNNOP &DRAMR & RDDRAM & NOWR
/     & ACTION & NEXT
;
; LOAD THE LOWER PART OF THE DMA ADDRESS TOGETHER WITH ALL THE DATA
;  SAVED ABOVE IN TEMPORARY REGISTERS INTO THE DMA ADDRESS
;  GENERATOR RAM. EVERYTHING IS NOW SET UP FOR INPUT DMA (PHEW!)
;  BUT HERE GOES WITH THE OUTPUT, WHICH, OF COURSE, IS MORE
;  COMPLICATED
;
;
      NOPAD & NOPWD & CHNNOP & DRAMR & RDHPHI & WRAHI
/     & ACTION ,,OUTPUT & NEXT
;
; SET OUTPUT MODE AND COPY THE HIGH PART OF THE ABSOLUTE ADDRESS
;  VALUE INTO TEMPORARY REGISTER
;
;
      LOADAD & LOADWD & CHNNOP & DRAMR & RDHPLO & NOWR
/     & ACTION & NEXT
;
; TRANSFER THE LOW PART OF THE ABSOLUTE ADDRESS, TOGETHER WITH THE
;  HIGH PART SAVED IN THE LAST INSTRUCTION, INTO THE DRAM ADDRESS
;  COUNTER, WHICH NOW POINTS AGAIN TO THE INPUT CHAIN HEAD
;  POINTER
;
;
      NOPAD & NEXTWD & CHNNOP & DRAMR & NORD & NOWR
/     & ACTION & NEXT
;
; ADVANCE THE DRAM COUNTER BY ONE WORD
;
;
      NOPAD & NEXTWD & CHNNOP & DRAMR & NORD & NOWR
/     & ACTION ,,,,KIKRAM & NEXT
;
; ADVANCE THE DRAM COUNTER ONE MORE WORD SO THAT IT NOW POINTS TO
;  THE OUTPUT CHAIN HEAD POINTER. INITIATE A READ
;
;
      WAIT $
;
; WAIT FOR READ TO COMPLETE
;
;
      NOPAD & NEXTWD & CHNNOP & DRAMR & RDDRAM & WRAHI
/     & ACTION ,,,,KIKRAM & NEXT
;
; SAVE DATA FETCHED (HIGH PART OF ADDRESS OF OUTPUT INITIALISING TABLE)
;  IN TEMPORARY REGISTER, ADVANCE DRAM ADDRESS AND START A READ
;
;
      WAIT $
;
; WAIT FOR READ TO COMPLETE
;
;
      LOADAD & LOADWD & CHNNOP & DRAMR & RDDRAM & NOWR
/      & ACTION ,,,,KIKRAM & NEXT
;
; LOAD THE COMPLETE ADDRESS OF THE OUTPUT INIT. TABLE INTO THE
;  DRAM ADDRESS COUNTER AND START A READ
;
;
      WAIT $
;
; WAIT FOR READ TO COMPLETE
;
;
      NOPAD & NEXTWD & CHNNOP & DRAMR & RDDRAM & WRMOD
/      & ACTION & NEXT
;
; READ FIRST ENTRY OF OUTPUT INIT. TABLE, WHICH IS NUMBER OF CHANNELS
;  LESS ONE, AND LOAD THIS VALUE INTO THE MODULUS CONTROL REGISTER
;  OF THE CHANNEL COUNTER. THE CHANNEL COUNTER COUNTS DOWN, VALUE=
;  MOD CORRESPONDS TO THE FIRST CHAIN IN THE TABLE, VALUE=0 TO THE
;  LAST. THE COUNTER IS AT THE MOMENT IN AN INDETERMINATE STATE.
;  ADVANCE DRAM ADDRESS TO POINT TO FIRST CHANNEL HEAD POINTER
;
;
BEER: NOPAD & NOPWD & CHNADV & DRAMR & NORD & NOWR
/      & ACTION & JMPIFN CHNOTE,CIDER
;
; ADVANCE THE CHANNEL COUNTER (DOWNWARDS!). IF IT WAS ZERO BEFORE
;  JUMP OUT (IT WILL NOW BE = MOD BECAUSE OF THE CHNADV HERE),
;  OTHERWISE WE'LL KEEP GOING
;
;
      NOPAD & NOPWD &CHNNOP & DRAMR & NORD & NOWR
/      & ACTION & JMP BEER
;
; DURING THIS INSTRUCTION THE CHANNEL COUNTER WILL BE IN A STATE
;  ONE LESS THAN DURING THE LAST. BY THE END OF THIS INSTRUCTION
;  THE END CONDITION CHNOTE WILL BE SET UP READY TO BE TESTED.
;  THIS INSTRUCTION IS JUST TO PREVENT THE PIPELINING FROM MAKING
;  US STOP ADVANCING THE CHANNEL COUNTER TOO LATE
;
;
CIDER: NOPAD & NOPWD & CHNNOP & DRAMR & NORD & NOWR
/      & ACTION ,,,,KIKRAM & NEXT
```

```
;
; INITIATE READ OF HIGH PART OF CHANNEL HEAD POINTER
;
;
      WAIT $
;
; WAIT FOR READ TO COMPLETE
;
;
      NOPAD & NEXTWD & CHNNOP & DRAMR & RDDRAM & WRSCRA UPPER
/     & ACTION ,,,,KIKRAM & NEXT
;
; SAVE THE UPPER PART OF THE HEAD POINTER FOR THE CURRENT OUTPUT
;  CHANNEL IN THE CORRESPONDING SCRATCHPAD, ADVANCE THE DRAM;
;  ADDRESS AND START A NEW READ
;
;
      WAIT $
;
; WAIT FOR READ TO COMPLETE
;
;
      NOPAD & NEXTWD & CHNADV & DRAMR & RDDRAM & WRSCRA LOWER
/     & ACTION & JMPIF CHNOTE,CIDER
;
; SAVE THE LOWER PART OF THE HEAD POINTER IN SCRATCHPAD AND ADVANCE
;  THE DRAM ADDRESS. ADVANCE THE CHANNEL COUNTER (N.B. IT CHANGES
;  AFTER THE WRSCRA IS COMPLETE) AND IF WE HAVE NOT JUST DONE THE
;  CASE COUNTER=0 REPEAT FROM CIDER. THE LOOP WILL EXIT WITH
;  COUNTER=MOD. AT EXIT WE HAVE THE HEAD POINTERS FOR ALL THE
;  OUTPUT CHAINS STORED IN THE CORRESPONDING SCRATCHPAD LOCATIONS,
;  AND WE CAN USE THEM TO FETCH THE DATA FOR THE FIRST ELEMENTS
;  IN EACH CHAIN.
;
;
SCOTCH: NOPAD & NOPWD & CHNNOP & DRAMR & RDSCRA UPPER & WRAHI
/     & ACTION & NEXT
;
; PREPARE FOR NEXT CHANNEL BY FETCHING HIGH PART OF HEAD POINTER
;  FROM SCRATCHPAD
;
;
      LOADAD & LOADWD & CHNNOP & DRAMR & RDSCRA LOWER & NOWR
/     & ACTION ,,,,KIKRAM & NEXT
;
; COMPLETE SETTING OF DRAM ADDRESS COUNTER TO POINT TO FIRST WORD
;  OF FIRST ELEMENT OF CURRENT OUTPUT CHAIN, AND START READ OF
;  FIRST WORD
;
;
      WAIT $
```

```
;
; WAIT FOR READ TO COMPLETE
;
;
      NOPAD & NEXTWD & CHNNOP & DRAMR & RDDRAM & WRSCRA UPPER
/     & ACTION ,,,,KIKRAM & NEXT
;
; SAVE LINK HIGH DATA IN SCRATCHPAD (NOW WE NO LONGER NEED IT TO
;  HOLD HEAD POINTER!), ADVANCE DRAM ADDRESS AND START ANOTHER READ
;
;
      WAIT $
;
; WAIT FOR READ TO COMPLETE
;
;
      NOPAD & NEXTWD & CHNNOP & DRAMR & RDDRAM & WRSCSA LOWER
/     & ACTION ,,,,KIKRAM & NEXT
;
: SAVE LINK LOW DATA, ADVANCE DRAM ADDRESS AND START ANOTHER READ
;
;
      WAIT $
;
; WAIT FOR READ TO COMPLETE
;
;
      NOPAD & NEXTWD & CHNNOP & DRAMR & RDDRAM & WRCNT
/     & ACTION ,,,,KIKRAM & NEXT
;
; SAVE OPERATION COUNT DATA IN TEMPORARY REGISTER, ADVANCE DRAM
;  ADDRESS AND START NEW READ
;
;
      WAIT $
;
; WAIT FOR READ TO COMPLETE
;
;
      NOPAD & NEXTWD & CHNNOP & DRAMR & RDDRAM & WRTC
/     & ACTION ,,,,KIKRAM & NEXT
;
; SAVE NEWLINE AND TONE-CURVE SELECT DATA. ADVANCE DRAM ADDRESS AND
;  START A NEW READ
;
;
      WAIT $
;
; WAIT FOR READ TO COMPLETE
;
;
```

```
      NOPAD & NEXTWD & CHNNOP & DRAMR & RDDRAM & WRAHI
/     & ACTION ,,,,KIKRAM & NEXT
;
; SAVE INCREMENT AND HIGH DMA ADDRESS DATA, ADVANCE DRAM ADDRESS AND
;  START A NEW READ
;
;
      WAIT $
;
; WAIT FOR READ TO COMPLETE
;
;
      NEWAD & NOPWD & CHNADV & DRAMR & RDDRAM & NOWR
/     & ACTION & JMPIF CHNOTE,SCOTCH
;
; LOAD COMPLETED DATA FROM FIRST ELEMENT OF CURRENT CHAIN TO DMA
;  ADDRESS GENERATOR RAM. ADVANCE TO NEXT CHANNEL, IF THE PREVIOUS
;  VALUE WAS NOT THE LAST (ZERO) THEN REPEAT FOR NEXT OUTPUT
;  CHANNEL. OTHERWISE, THAT'S ABOUT IT, WE'VE SET UP THE INPUT AND
;  ALL OF THE OUTPUT CHANNELS, THE CHANNEL COUNTER IS AT MOD READY
;  FOR THE FIRST OUTPUT DMA, ALL THE NEXT LINKS ARE IN SCRATCHPAD,
;  WE JUST SIT BACK AND WAIT FOR DMA REQUESTS
;
;
      NOPAD & NOPWD & CHNNOP & DRAMR & NORD & NOWR
/     & ACTION & JMP LOOP
;
; GO DO IT
;
;
NXLINK:
;
; FETCH NEXT LINK FOR CURRENT CHANNEL. INPUT/OUTPUT MODE AND
;  OUTPUT CHANNEL POINTER ARE STILL SET TO THE CURRENT CHAIN
;
;
      NOPAD & NOPWD & CHNNOP & DRAMR & RDSCRA UPPER & WRAHI
/     & ACTION CBIT,IBIT & NEXT
;
; FETCH UPPER PART OF THE LINK ADDRESS FOR THIS CHAIN AND SAVE IN ADDRESS
;  HIGH REGISTER. THE UPPER PART OF THIS WORD CONTAINS INTERRUPT AND
;  CONTINUE BITS, WHICH ARE ACTIONED AS THIS WORD IS TRANSMITTED. IF
;  THERE IS NO CONTINUATION, THE HARDWARE WILL ENSURE THAT THE
;  APPROPRIATE DEVICE READY CANNOT REAPPEAR, BUT MEANWHILE THE DMA
;  MACHINE WILL FETCH RUBBISH FROM ANYWHERE AS THE NEXT LINK DATA
;
;
      LOADAD & LOADWD & CHNNOP & DRAMR & RDCSRA LOWER & NOWR
/     & ACTION ,,,,KIKRAM & NEXT
;
; FETCHES LOWER PART OF LINK ADDRESS FOR THIS CHAIN, AND LOADS THIS
;  TOGETHER WITH THE SAVED HIGH PART INTO THE DRAM ADDRESS COUNTER,
;  THEN INITIATES A READ
;
;
      WAIT $
;
; WAIT FOR READ TO COMPLETE
;
;
      NOPAD & NEXTWD & CHNNOP & DRAMR & RDDRAM & WRSCRA UPPER
/     & ACTION ,,,,KIKRAM & NEXT
;
; SAVE THE DATA (HIGH PART OF NEW LINK ADDRESS) OVER THE TOP OF
;  THE LAST LINK, ADVANCE DRAM ADDRESS AND START A NEW READ
;
;
      WAIT $
;
; WAIT FOR READ TO COMPLETE
;
;
      NOPAD & NEXTWD & CHNNOP & DRAMR & RDDRAM & WRSCRA LOWER
/     & ACTION ,,,,KIKRAM & NEXT
;
; SAVE THE DATA (LOWER PART OF NEW LINK ADDRESS), ADVANCE THE DRAM
;  ADDRESS, AND START A NEW READ
;
;
      WAIT $
;
; WAIT FOR READ TO COMPLETE
;
;
      NOPAD & NEXTWD & CHNNOP & DRANR & RDDRAM & WRCNT
/     & ACTION ,,,,KIKRAM & NEXT
;
; SAVE DATA (OPERATION COUNT) IN TEMPORARY REGISTER, ADVANCE DRAM ADDRESS
;  AND START A NEW READ
;
;
      WAIT $
;
; WAIT FOR READ TO COMPLETE
;
;
      NOPAD & NEXTWD & CHNNOP & DRAMR & RDDRAM & WRTC
/     & ACTION ,,,,KIKRAM & NEXT
;
; SAVE DATA (NEWLINE AND TONE-CURVE SELECT) IN TEMPORARY REGISTER,
;   ADVANCE DRAM ADDRESS AND START A NEW READ
;
```

```
;
      WAIT $
;
; WAIT FOR READ TO COMPLETE
;
;
      NOPAD & NEXTWD & CHNNOP & DRAMR & RDDRAM & WRAHI
/      & ACTION ,,,,KIKRAM & NEXT
;
; SAVE DATA (INCREMENT AND HIGH DMA ADDRESS) IN TEMPORARY REGISTER,
;  ADVANCE DRAM ADDRESS AND START A NEW READ
;
;
      WAIT $
;
; WAIT FOR READ TO COMPLETE
;
;
      NEWAD & NOPWD & CHNNOP & DRAMR & RDDRAM & NOWR
/      & ACTION & NEXT
;
; LOAD DATA (LOW DMA ADDRESS) AND ALL PREVIOUS DATA IN TEMPORARY
;  REGISTERS INTO THE DMA ADDRESS GENERATOR RAM FOR THE CURRENT
;  CHAIN
;
;
      NOPAD & NOPWD & CHNNOP & DRAMR & NORD & NOWR
/      & ACTION & JMPIF INPUTM,LOOP
;
; THAT'S ALL FOR INPUT CHAINS
;
;
      NOPAD & NOPWD & CHNADV & DRAMR & NORD & NOWR
/      & ACTION & JMP LOOP
;
; FOR OUTPUT CHAINS ONLY ADVANCE THE CHANNEL POINTER TO THE NEXT
;  CHANNEL
;
;
; MAIN LOOP - ACTUALLY DOING THE DMA - ENTER AT LOOP
;  THIS FIRST PART IS THE ACTUAL DMA SYSTEM
;
;
I:   TESTWC & NOPWD & CHNNOP & DRAMR & NORD & NOWR
/      & ACTION & JMPIF NOTEOC,I
;
; A DRAM CYCLE IS GOING ON SO WE IS KEPT INACTIVE. THIS INSTRUCTION
;  REPEATEDLY READS OUT THE CURRENT ADDRESS AND WORDCOUNT READY FOR TEST
;  IN THE NEXT INSTRUCTION, UNTIL THE DRAM CYCLE HAS FINISHED. (N.B. EOC
;  IS SUPPLIED ONE CYCLE EARLY SO THAT NO PIPELINE DELAY IS INTRODUCED)
;
```

```
;
      NOPAD & NOPWD & CHNNOP & DRAMR & RDDRAM & WRDEV
/     & ACTION & JMPIFN WCNOTE,NXLINK
;
; AN OUTPUT (DRAM READ) CYCLE HAS JUST FINISHED. THE CURRENT WORD COUNT
;  FOR THE CURRENT OUTPUT CHANNEL IS TESTED, IF IT IS ZERO THEN BRANCH
;  OUT TO FETCH THE NEXT LINK IN THE CURRENT OUTPUT CHAIN
;
;
      UPDATE & NOPWD & CHNADV & DRAMR & NORD & NOWR
/     & ACTION & JMPIF INNOTR,Z
;
; AN OUTPUT DMA HAS BEEN COMPLETED, SO THE NEXT ADDRESS AND WORD COUNT
;  STORED IN THE UPDATE REGISTER ARE WRITTEN BACK TO THE ADDRESS RAM.
;  MEANWHILE, IF THE INPUT DEVICE IS NOT READY, THEN SKIP OUT TO
;  SEE WHETHER WE CAN DO ANOTHER OUTPUT CYCLE IMMEDIATELY. ADVANCE
;  THE OUTPUT CHANNEL POINTER TO THE NEXT CHANNEL
;
;
Y:   NEXTAD & NOPWD & CHNNOP & DRAMR & NORD & NOWR
/     & ACTION ,,,INPUT,KIKRAM & NEXT
;
; ENTRY POINT TO DO AN INPUT CYCLE
;  SET INPUT MODE AND FETCH NEXT INPUT ADDRESS TO THE DRAM ADDRESS
;  COUNTER. KICK THE DRAM TO DO A CYCLE. THE NEXT ADDRESS, COUNT, ETC
;  ARE WRITTEN TO THE UPDATE REGISTER
;
;
N:   NOPAD & NOPWD & CHNNOP & DRAMW & RDDEV & WRDRAM
/     & ACTION & NEXT
;
; HAVING REQUESTED AN INPUT CYCLE AND SET ITS ADDRESS, SET WRITE
;  ENABLE AND TRANSFER THE DATA (THIS MAY OVERLAP THE FIRST THIRD
;  OF THE DRAM WRITE CYCLE)
;
J:   TESTWC & NOPWD & CHNNOP & DRAMW & NORD & NOWR
/     & ACTION & JMPIF NOTEOC,J
;
; A DRAM WRITE CYCLE IS GOING ON, SO WE IS KEPT ACTIVE. THIS INSTRUCTION
;  REPEATEDLY READS OUT THE CURRENT ADDRESS AND WORD COUNT READY FOR
;  TEST IN THE NEXT INSTRUCTION UNTIL THE DRAM CYCLE HAS FINISHED
;
;
      NOPAD & NOPWD & CHNNOP & DRAMR & NORD & NOWR
/     & ACTION & JMPIFN WCNOTE,NXLINK
;
; AN INPUT (DRAM WRITE)    CYCLE HAS JUST FINISHED. THE CURRENT WORD COUNT
;  FOR INPUT (SET UP BY THE LAST INSTRUCTION) IS TESTED, IF IT IS ZERO
;  THEN BRANCH OUT TO FETCH THE NEXT LINK IN THE INPUT CHAIN
;
;
```

```
      UPDATE & NOPWD & CHNNOP & DRAMR & NORD & NOWR
/     & ACTION & JMPIF OUNOTR,W
;
; AN INPUT DMA HAS BEEN COMPLETED, SO THE NEXT ADDRESS AND WORD COUNT
;  SAVED IN THE UPDATE REGISTER ARE WRITTEN BACK TO THE ADDRESS RAM.
;  MEANWHILE, IF THE OUTPUT DEVICE IS NOT READY, THEN SKIP OUT TO
;  SEE WHETHER WE CAN DO ANOTHER INPUT CYCLE IMMEDIATELY.
;
X:   NEXTAD & LOADWD & CHNNOP & DRAMR & NORD & NOWR
/     & ACTION ,,OUTPUT,,KIKRAM * JMP I
;
; ENTRY POINT TO DO AN OUTPUT CYCLE
;  SET OUTPUT MODE AND FETCH NEXT OUTPUT ADDRESS FOR THE CURRENT
;  CHANNEL TO THE DRAM ADDRESS COUNTER. KICK DRAM TO REQUEST A
;  CYCLE. THE NEXT ADDRESS, WORD COUNT, ETC. ARE WRITTEN TO THE
;  UPDATE REGISTER. PROCESS CONTINUES FROM LABEL I
;
;
; MAIN LOOP BACKWATER
;
;
W;   NEXTAD & LOADWD & CHNNOP & DRAMR & NORD & NOWR
/     & ACTION ,,,INPUT,KIKRAM & JMPIFN INNOTR,N
;
; AN INPUT DMA HAS BEEN COMPLETED, BUT THE OUTPUT DEVICE WAS NOT
;  READY TO TAKE THE NEXT CYCLE. THERE IS NO TIME TO CHECK
;  FIRST, SO SET INPUT MODE AND LOAD THE NEXT INPUT ADDRESS AND
;  START A CYCLE. IF THE INPUT DEVICE WAS READY, GO DO IT
;
;
      WAIT $
;
; WASTE THE INPUT CYCLE ASKED FOR WHEN THE DEVICE WAS NOT READY
;
;
      NOPAD & NOPWD & CHNNOP & DRAMR & NORD & WOWR
/     & ACTION & JMP LOOP
;
; SEE WHAT ELSE THERE IS TO DO
;
;
Z:   NEXTAD & LOADWD & CHNNOP & DRAMR & NORD & NOWR
/     & ACTION ,,OUTPUT,,KIKRAM & JMPIFN OUNOTR,I
;
; AN OUTPUT DMA CYCLE HAS BEEN COMPLETED BUT THE INPUT DEVICE WAS NOT
;  READY TO TAKE THE NEXT CYCLE. THERE IS NO TIME TO CHECK FIRST, SO
;  SET OUTPUT MODE, LOAD THE NEXT OUTPUT ADDRESS, AND START A CYCLE.
;  IF THE OUTPUT DEVICE WAS READY, GO DO IT
;
;
      WAIT $
```

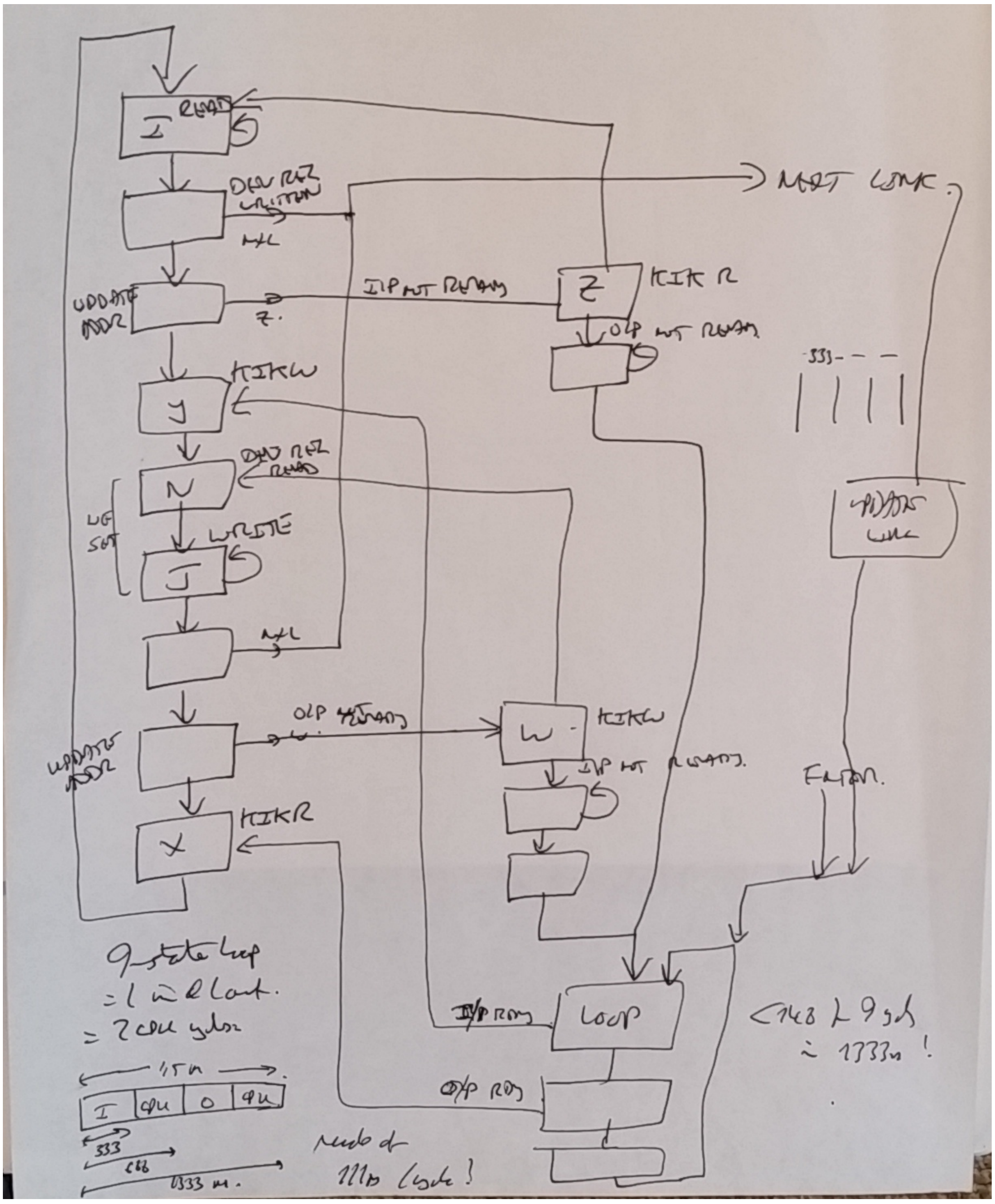
```
;
; WASTE THE OUTPUT CYCLE ASKED FOR WHEN THE DEVICE WAS NOT READY
;
;
; THIS IS IT - THE MAIN SCANNING LOOP
;
;
LOOP: NOPAD & NOPWD & CHNNOP & DRAMR & NORD & NOWR
/      & ACTION & JUMPIFN INNOTR,Y
;
; DO NOTHING UNLESS THE INPUT DEVICE IS READY, IN WHICH CASE DO AN
;  INPUT CYCLE
;
;
      NOPAD & NOPWD & CHNNOP & DRAMR & NORD & NOWR
/      & ACTION & JMPIFN OUNOTR,X
;
; DO NOTHING UNLESS THE OUTPUT DEVICE IS READY, IN WHICH CASE DO AN
;  OUTPUT CYCLE
;
;
      NOPAD & NOPWD & CHNNOP & DRAMR & NORD & NOWR
/      & ACTION & JMP LOOP
;
; DO NOTHING AT ALL, JUST LOOP BACK TO KEEP LOOKING FOR SOMETHING
;  USEFUL TO DO
;
ORG H#400
;
;
      NOPAD & NOPWD & CHNNOP & DRAMR & NORD & NOWR
/      & ACTION & NEXT
;
; INSTRUCTION FOR INIT WORD (PROM PROGRAMMER FIX MAPS IT TO WORD
;  1024)
;
END
```

Microcode word format

```
3333333333222222222211111111110000000000
9876543210987654321098765432109876543210
hhhhc.bbaaaaageggggggeefheediiiijjjjjjjjj

a dma address generator
b dram address counter
c channel counter control
d dram we
e data source
```

```
f data source parameter (read and write)
g data destination
h action bits
i conditional select
j jump field
. not used
```



Rough flow diagram of the actual DMA part of the code, a loop of 9 states for each input + output pair of DRAM accesses, a pair of short-circuits (Z and W) that allow inputs or outputs to take every available DRAM access if the other isn't ready, a waiting loop if neither are ready, and a call-out to fetch the next link when the count for this block has expired.

Note that if the "normal" next operation isn't ready it kicks off the other immediately, without checking that it is ready, and then checks. This is safe as although the unwanted DRAM cycle does go through, the DMA address and count are not updated, no write cycle is performed (WE is never set, it's only set in N and J) and the output register is not overwritten.

Other details

The linked list format was (in 16-bit memory, 24-bit addresses):

Fixed address:

  Input head pointer H
  Input head pointer L
  Output head pointer H
  Output head pointer L

Input head pointer:

  Link to next block H
  Link to next block L
  Block count
  Not used
  DMA increment and address H
  DMA address L

Output head pointer:

  Number of chains N-1
  Chain head pointer 1 H (0?)
  Chain head pointer 1 L (0?)
  ...
  Chain head pointer N-1 H
  Chain head pointer N-1 L

Chain head pointer N:

  Link to next block H
  Link to next block L
  Block count
  New-line and Tone Curve
  DMA increment and address H
  DMA address L

New-line fed out a pulse that told the output circuit to reset to a new line.
Tone-curve selected one of N mapping curves, implemented in SRAM.

The upper half of the link to next block contained a bit which if set made the DMA engine stop, and another which if set caused an interrupt to the 68000. Further bits in this half were used to vector the interrupt; interrupt and vectors were queued in a hardware FIFO. The interrupts were used to prompt the 68000 to keep extending the linked lists to keep ahead as they were consumed.

When working with Lasergravure, the Impositioner sent out data at the Crosfield native resolution of 12 pixels/mm and the screen was superimposed by the video output board. The Klischographs had to work at the mechanical pecking resolution which was unrelated to the contone image and different depending on the screen pattern chosen. To handle this the digital data were passed from the Impositioner into axial and circumferential resolution converter boards which performed cubic interpolation in both axes. This required, for the axial direction, data from four lines, which is why there were 32 output chains, 4 for each of the 8 heads. The calculation of the circumferential conversion depended on knowing or guessing how the Klisch calculated its clock dividers; we found a reliable algorithm that depended on reducing to prime factors.

Thinks. DRAM cycle was 333 ns, CPU bus cycle 666 ns ("6 MHz" chip), complete cycle of input DMA - CPU - output DMA - CPU = 1333 ns and this takes at least 9 microcode steps, so won't work unless microcode is <148 ns, so probably was 111 ns.

Related

There was also an in-line hardware USM board. The data path through resolution conversion, unsharp masking and output was designed with a common interface so boards could be used or bypassed easily.

The serial comms bus to remote units (laser power supply, modulator chassis, lathe control) was a very simple custom design with a single-chip micro on each. The bus was a single serial pair and power. The remote units were programmed in Forth.

There was also a whole rack of boards making a K202 simulator, which was to allow the factory to test Gravure Plus without having a K202. That was a fun design, out of which I did two of the boards. It provided all the signals to convince the Gravure Plus to run and collected the streamed data for analysis. The interface to the K202 was a huge 100-way T-bar screw connector which we broke into, taking the K202 scanner's data and adding our digital data so that it was possible to merge our digital images with scanned images on the same engraving run; we emulated the digital interface to the "AK Daten" board within the Klisch (Abtastkopf Daten - read head data).

Everything was got to work but was never used in earnest. Lasergravure died because the plastic would never last on the press. Gravure Plus died because Hell would not tolerate it.

nib

2016-04-15