```
  CCCCCCCC    TTTTTTTTTTTT     CCCCCCCC
 CC      CC        TT         CC      CC
 CC      CC        TT         CC      CC
 CC                TT         CC
 CC                TT         CC
 CC                TT         CC
 CC                TT         CC
 CC                TT         CC
 CC      CC        TT         CC      CC
 CC      CC        TT         CC      CC
  CCCCCCCC         TT          CCCCCCCC
```

CARTRIDGE TAPE CONTROLLER

INTERNAL
SPECIFICATION FOR :

P.C. Board Rev...............	4, or B
Re-worked P.C. Board Rev....	5
CTC rev. level(s)...........	8,9,10,11, or B
DATE	July 13th, 1981

PROM SET REVISION :

#0125-0048	NO MATTER
#0125-0049	A or B
#0125-0059	Same as #0125-0049

This document should contain everything you need to know to read, write, and edit CTC micro-code.

Written by

David W. Milton

TABLE OF CONTENTS

```
          |             |                |             |
    49 48 ! 47 46 45 44 ! 43 42    41 40 ! 39 38 37 36 !
   \----------------------------/ \------------------/
                 !                          !
           Literal Field and         Branch condition
              Ram Address                 Select
              pg. 18                      pg. 22


   |                |                |
   ! 35    34 33 32 ! 31 30    29 28 ! 27 26 25
    \--/  \----------------/  \-----------------/
     !            !                  !
     RAM     Destination          Source
   Address     Select             Select
   Select      pg. 11              pg.8
pg. 10 & 14


       |              |              |
    24 ! 23 22     21 20 ! 19     18 17 16 ! 15
   \----------/   \----------/   \---------/ \--/
         !              !             !        !
       2901           2901          2901     Carry
   Destination      Function       Source      In
      pg. 17      \-----------------------------/
                                 !
                           2901 Op-Code
                              pg. 5


             |             |             |             |
    14 13 12 ! 11 10  9  8 !  7  6  5  4 !  3  2  1  0 !
   \-------------------------------------/ \-----------/
                   !                             !
             Jump Address                  Next Address
               pg. 21                         Select
                                              pg. 19
```

The CTC assembler is a two line assembler. For every two lines of source code, one line of binary code will be generated. In the listing, the binary code will appear adjacent to the second line of the assembled source code.

The CTC assembler has the following source code format :

```
LABEL;2901 OP-CODE;SOURCE;DESTINATION;2901 DESTINATION;LITERAL;COMMENT
LABEL;NEXT ADDRESS SELECT;JUMP ADDRESS;BRANCH CONDITION SELECT;COMMENT
```

In the assembled listing, the semicolons will not appear but instead the fields will be separated by spaces.

Read on in this document to see what mnemonics are defined for which field.

If you leave a field blank in the source code - that is, you type nothing between the two semicolons which delineate that field - the assembled listing will print nothing where that field belongs, but the assembled binary code will be the default op-codes.

Source code that looks like this :

```
;;;;;;First line
;NEXT;;;Second line
```

Will be assembled and listed like this :

```
                        First line
    NEXT                Second line
```

And the binary code will be the following defaults :

```
;ZERO;R0;R0;NOP;#00;
;NEXT;*+1;ALWAYS;
```

Note that in order for the assembler to keep track of what's going on, every "second line" of your source code must have some mnemonic called out in the Next Address Select field.

This is a discussion of some assembler pseudo-ops in the context of the Cartridge Tape Controller. A complete explanation of all pseudo-operations is provided in the "A2100" manual, availiable from the training department.

```
;EJECT;
            This statement will cause the listing to skip to the top of
            the next page before printing anything more.


;TITLE;Anything you want
                             This statement puts a title at the top of
                             the page. This title is "Anything you want".

       If the title specification is the first line of source code
       following an eject, the title will appear at the top of that page.
       If the title specification is the second or more line of source
       code following an eject, the title will appear beginning on the
       next page.


;ORG;<expression>
                      This statement lets you specify at what address the
                      next micro-code instruction will appear.

     The expression may look like :

     #7FF --------- any hex number not to exceed #7FF
     *+2 ---------- the address of the previous instruction plus 2
     400 ---------- a decimal number which will be converted to hex
     *-1 AND #7FE - the previuos instruction's address minus one with
                    the least significant bit masked to zero.


;BOX                 \  This format draws pretty boxes around text
 any text you like  |-  which will not be assembled.  One line of text
;EBOX                /  may contain up to 80 characters.


;LBOX          >>>>>>>>>>>>
 source code   > the source code will be assembled and high-lighted
 source code   > like this sentence is.
;ELBOX         >>>>>>>>>>>>


;INCLUDE;<filename>
                          This will insert the contents of the specified
                          file into this location and it will be
                          assembled like the rest of this source code is.


;END
         This must be the last line of the source code.
```

Format : LABEL;

A LABEL may contain up to 12 alpha-numeric characters. A decimal point is also allowed. The first character of a LABEL must be an alpha (A-Z).

Putiing a label in the first field of a line of code (the LABEL field) implicitly defines that label as equal to the address at which this line of code, once assembled, will appear. Thus a specific label may appear only once in the first field of a line of code. It may appear numerous times in other fields. If two different labels are put in the first field of both the first and second lines of the two lines of source code to be assembled into one line of object code, both of those labels will be defined as the same value.

A label's value may also be explicitly specified with the pseudo-op "EQU" :

```
LABEL;EQU;#6A3;Any comment you like
```

This will assign the value of #6A3 to the label "LABEL"

Format: OP-CODE;

This field specifies micro-code bits 15-21, which are the 2901 SOURCE operands and the 2901 FUNCTION select and the CARRY-IN bit. In the following listing of permissable OP-CODES, here is what the letters "A", "B", "S", and "Q" represent. Read this carefully, since many unique aspects of this controller's design are explained here.

"A" refers to one of 16 addressable registers internal to the 2901. If the SOURCE field contains one of the mnemonics "R0"-"RF", the 0 or the F or whichever in between will be the address of the register whose contents "A" represents. If the SOURCE field contains a mnemonic other than "R0"-"RF" then the mnemonic in the DESTINATION field determines which of the 16 addressable registers' contents "A" refers to. A little micro-code writing will make it clear why this is done. However, here is an attempted explanation:
Many of the 2901 op-codes availliable involve operations on "A" and "S", where "S" is the data input to the 2901. "S"'s value is what is on the data bus which is determined by the mnemonics in the SOURCE field. If "A" were always selected by the SOURCE field, an operation like "S.ADD.A" would have both of its operands selected by the SOURCE field. To get around this, when the SOURCE field specifies a register external to the 2901, "A" is determined by the DESTINATION field instead. See also "B" below and the SOURCE and DESTINAITON fields.

"B" refers to one of the same 16 addressable registers internal to the 2901 as "A" above did. In this case, the DESTINATION field will always specify the register whose contents "B" respresents. If the DESTINATION field contains one of the mnemonics "R0"-"RF" the 0 or the F is the address of the register inside the 2901 whose contents "B" represents. Beware, though, if the DESTINATION field specifies one of the 12 registers external to the 2901, each of these destination registers has an equivalent value associated with it which, if "B" is used anywhere in this instruction, will address a register internal to the 2901. See DESTINATION field for those equivalent values.

"Q" refers to a 17th register inside the 2901.

"S" refers to the data inputs to the 2901, which is the data bus of this controller. The source of the data to be enabled onto this data bus is the output of the 2901 unless a register external to the 2901 is called out in the SOURCE field in which case that register is the source of the data on the data bus.
This does mean that it is possible to feed the 2901's output back into itself which could produce some interesting results. Since this anomoly is of little use from a programming point of view, there is no reason to do it and less reason to worry about it.

That covers all the not-so-obvious symbols used in the 2901 OP-CODES. "A", "B", and "Q" also appear in the op-codes for the 2901 DESTINATION select.

2901 OP-CODE (cont.)

STATE OF MICRO-CODE BITS: 21	20	19	18	17	16	15	MNEMONIC	MEANING ALU OUTPUT = :
0	0	0	0	0	0	0	A.ADD.Q	A+Q
0	0	0	0	0	0	1	A.AD1.Q	A+Q+1
0	0	0	0	0	1	0	A.ADD.B	A+B
0	0	0	0	0	1	1	A.AD1.B	A+B+1
0	0	0	0	1	0	0	PASS.Q	Q
0	0	0	0	1	0	1	INC.Q	Q+1
0	0	0	0	1	1	0	PASS.B	B
0	0	0	0	1	1	1	INC.B	B+1
0	0	0	1	0	0	0	PASS.A	A
0	0	0	1	0	0	1	INC.A	A+1
0	0	0	1	0	1	0	S.ADD.A	S+A
0	0	0	1	0	1	1	S.AD1.A	S+A+1
0	0	0	1	1	0	0	S.ADD.Q	S+Q
0	0	0	1	1	0	1	S.AD1.Q	S+Q+1
0	0	0	1	1	1	0	PASS.S	S
0	0	0	1	1	1	1	INC.S	S+1
0	0	1	0	0	0	0	Q.SB1.A	Q-A-1
0	0	1	0	0	0	1	Q.SUB.A	Q-A
0	0	1	0	0	1	0	B.SB1.A	B-A-1
0	0	1	0	0	1	1	B.SUB.A	B-A
0	0	1	0	1	0	0	DEC.Q	Q-1
0	0	1	0	1	1	0	DEC.B	B-1
0	0	1	1	0	0	0	DEC.A	A-1
0	0	1	1	0	1	0	A.SB1.S	A-S-1
0	0	1	1	0	1	1	A.SUB.S	A-S
0	0	1	1	1	0	0	Q.SB1.S	Q-S-1
0	0	1	1	1	0	1	Q.SUB.S	Q-S
0	0	1	1	1	1	0	NOT.S	Compliment of S = -S-1
0	0	1	1	1	1	1	NEG.S	-S = 1+Compliment of S
0	1	0	0	0	0	0	A.SB1.Q	A-Q-1
0	1	0	0	0	0	1	A.SUB.Q	A-Q
0	1	0	0	0	1	0	A.SB1.B	A-B-1
0	1	0	0	0	1	1	A.SUB.B	A-B
0	1	0	0	1	0	0	NOT.Q	Compliment of Q = -Q-1
0	1	0	0	1	0	1	NEG.Q	-Q = 1 + Compliment of Q
0	1	0	0	1	1	0	NOT.B	Compliment of B = -B-1
0	1	0	0	1	1	1	NEG.B	-B = 1 + Compliment of B
0	1	0	1	0	0	0	NOT.A	Compliment of A = -A-1
0	1	0	1	0	0	1	NEG.A	-A = 1 + Compliment of A
0	1	0	1	0	1	0	S.SB1.A	S-A-1
0	1	0	1	0	1	1	S.SUB.A	S-A
0	1	0	1	1	0	0	S.SB1.Q	S-Q-1
0	1	0	1	1	0	1	S.SUB.Q	S-Q
0	1	0	1	1	1	0	DEC.S	S-1

STATE OF MICRO-CODE BITS:							MNEMONIC	MEANING	
21	20	19	18	17	16	15		ALU OUTPUT = :	
0	1	1	0	0	0	0	A.IOR.Q	A Inclusive ORed with Q	
0	1	1	0	0	1	0	A.IOR.B	A Inclusive ORed with B	
0	1	1	1	0	1	0	S.IOR.A	S Inclusive ORed with A	
0	1	1	1	1	0	0	S.IOR.Q	S Inclusive ORed with Q	
1	0	0	0	0	0	0	A.AND.Q	A ANDed with Q	
1	0	0	0	0	1	0	A.AND.B	A ANDed with B	
1	0	0	0	1	0	0	ZERO	ALL ZEROS	DEFAULT CONDITION
1	0	0	0	1	0	0	NOP	ALL ZEROS	
1	0	0	1	0	1	0	S.AND.A	S ANDed with A	
1	0	0	1	1	0	0	S.AND.Q	S ANDed with Q	
1	0	1	0	0	0	0	A.BCL.Q	Compliment of A ANDed with Q	*
1	0	1	0	0	1	0	A.BCL.B	Compliment of A ANDed with B	*
1	0	1	1	0	1	0	S.BCL.A	Compliment of S ANDed with A	*
1	0	1	1	1	0	0	S.BCL.Q	Compliment of S ANDed with Q	*
1	1	0	0	0	0	0	A.XOR.Q	A Exclusive ORed with Q	**
1	1	0	0	0	1	0	A.XOR.B	A Exclusive ORed with B	**
1	1	0	1	0	1	0	S.XOR.A	S Exclusive ORed with A	**
1	1	0	1	1	0	0	S.XOR.Q	S Exclusive ORed with Q	**
1	1	1	0	0	0	0	A.XNOR.Q	A Exclusive NORed with Q	***
1	1	1	0	0	1	0	A.XNOR.B	A Exclusive NORed with B	***
1	1	1	1	0	1	0	S.XNOR.A	S Exclusive NORed with A	***
1	1	1	1	1	0	0	S.XNOR.Q	S Exclusive NORed with Q	***

* BCL means "BIT CLEAR". For "A.BCL.Q", the ALU output is Q with the bits cleared which correspond to the bits in A which were set.

** The XOR function sets a bit each time the corresponding bits in the two operands were unequal.

*** The XNOR function sets a bit each time the corresponding bits in the two operands were equal.

Format: OP-CODE;

This field performs two related functions. If this field contains one of the mnemonics "R0" - "RF", then the "0" - "F" addresses the register inside the 2901 whose contents the symbol "A" represents in the 2901 OP-CODE and the 2901 DESTINATION field. The controller's data bus will be driven by the output of the 2901 which is determined by the 2901 DESTINATION field.

If this SOURCE field contains one of the mnemonics which refers to a register external to the 2901, then that register is enabled onto the data bus and whatever is in the DESTINATION field will be used to address the register whose contents "A" represents. See all other fields mentioned above for further explaination.

For the case that this field specifies "A", and the 2901 output drives the data bus :

STATE OF MICRO-CODE BITS					MNEMONICS	ADDRESS OF REGISTER INSIDE THE 2901 WHOSE CONTENTS "A" REPRESENTS	
29	28	27	26	25			
0	0	0	0	0	R0	2901 REGISTER 0	DEFAULT CONDITION
0	0	0	0	1	R1	2901 REGISTER 1	
0	0	0	1	0	R2	2901 REGISTER 2	
0	0	0	1	1	R3	2901 REGISTER 3	
0	0	1	0	0	R4	2901 REGISTER 4	
0	0	1	0	1	R5	2901 REGISTER 5	
0	0	1	1	0	R6	2901 REGISTER 6	
0	0	1	1	1	R7	2901 REGISTER 7	
0	1	0	0	0	R8	2901 REGISTER 8	
0	1	0	0	1	R9	2901 REGISTER 9	
0	1	0	1	0	RA	2901 REGISTER A	
0	1	0	1	1	RB	2901 REGISTER B	
0	1	1	0	0	RC	2901 REGISTER C	
0	1	1	0	1	RD	2901 REGISTER D	
0	1	1	1	0	RE	2901 REGISTER E	
0	1	1	1	1	RF	2901 REGISTER F	

For the case that this field enables a register external to the 2901 on to the data bus and the DESTINATION field specifies "A" :

STATE OF MICRO-CODE BITS 29 28 27 26 25	MNEMONIC (SIGNAL NAME)	MEANING
1 0 0 0 0	DIN (E0-)	DATA IN. This is the 8 bit data byte sent by the PPU. This may only be read when Data Flag-out is true and Data Command-in is false. You must check the parity in this same instruction. This clears Data Flag-out. See PPU-Controller Interface specifications for details on how these signals work. See also the Branch Condition Select field.
1 0 0 0 1	CIN (E1-)	COMMAND-IN. This is the 8 bit command-status byte sent by the PPU. This may only be read when Command Flag-out is true and Command Command-in is false. See PPU-Controller Interface specifications for details on how these signals work. See also the Branch Condition Select field.
1 0 0 1 0	DRSTS (E2-)	DRIVE STATUS. Here's what the 8 bits mean :

```
 7   6   5   4   3   2   1   0 = LSB
-----------------------------
 S   T   D   E   E   B   B   W
 E   R   R   O   O   O   O   P
 L   D   D   T   T   T   T  (-)
 D   Y   Y   W  (-)  W  (-)
(+) (+) (+) (-)     (-)
```

7)SELD+	SELECTED. The selected drive indicates it is present and alive by acknowledging that it is selcted. If SELD+ is low, the other status bits are meaningless.
6)TRDY+	TAPE READY. The installed cartridge tape position is known. Bits 4-1 are accurate. If Tape not Ready, bits 4-1 are inactive high.
5)DRDY+	DRIVE READY. A cartridge is installed and the tape position sensor light is drawing current. If Drive not Ready, bit 6 is low, bit 0 will be low.
4)EOTW-	END OF TAPE WARNING. Active low to indicate the tape is positioned within 4 feet of the end of the tape.
3)EOT-	END OF TAPE. Goes active low when forward motion finds the end. Goes high when reverse motion passes 4 feet from the end.
2)BOTW-	BEGINNING OF TAPE WARNING. Active low to indicate the tape is positioned within 4 feet of the beginning of the tape.
1)BOT-	BEGINNING OF TAPE. Works just like EOT- only for the other end of the tape.
0)WP-	WRITE PROTECED. Acitve low to indicate you should not write or erase the installed tape cartridge.

STATE OF MICRO-CODE BITS 29 28 27 26 25	MNEMONIC (SIGNAL NAME)	MEANING
1 0 0 1 1	CLRTC (E3-)	CLEAR RTC. This will let the data bus float while it clears the RTC test condition. See BRANCH CONDITION SELECT field for details.
1 0 1 0 0	LIT (E4-)	LITERAL. The 8 bit LITERAL / RAM ADDRESS field is enabled onto the data bus. See LITERAL / RAM ADDRESS field for details.
1 0 1 0 1	SCR,RAM (E5-)	SCRATCH or RAM. Either mnemonic will enable the same R.A.M. chips onto the data bus. SCR will also set micro-code bit 35. RAM will leave 35 clear. SCR uses the scratch pad address register to address the R.A.M. chips. See DESTINATION field SADRL & SADRH for details. RAM uses the 8 bits in the LITERAL / RAM ADDRESS field to address the top 256 locations in this 1024 loaction R.A.M. chips. See also DESTINATION field mnemonics SCR and RAM.
1 0 1 1 0	RDATA (E6-)	READ DATA. When branch condition "RDR" (Read Data Ready) is true, RDATA will enable onto the data bus the last 8 bits of data read off the tape. This will clear RDR.
1 0 1 1 1	RDSTS (E7-)	READ DATA STATUS. The data bus will look like :

```
  7    6    5    4    3    2    1    0 = LSB
-----------------------------------
  R    R    R    C    See below....
  D    D    D    R
  D    R    L    CERR
```

7) RDD READ DATA DETECTED. Readable data is detected on the tape now.

6) RDR READ DATA READY. It is time to read the next byte of data which the read logic has decoded.

5) RDL READ DATA LATE. This signal latches to indicate data was not taken from the read logic fast enough, a byte was lost.

4) CRCERR CRC ERROR. The CRC character did not check indicating the data has a mistake in it.

3-0) This is the value of the byte counter in the read logic. See FPLA code for details. It's main purpose is to distinguish legal from illegal GCR codes. "D", "E", or "F" here indicates RDATA has a decoded illegal GCR code in it now. "C" will not appear here when GCR density is selected.

FORMAT : OP-CODE;

The DESTINATION field performs a number of functions. If it contains a symbol from "RO"-"RF", then that "0"-"F" addresses one of the 16 registers inside the 2901 whose contents the symbol "B" represents in the 2901 OP-CODES and the 2901 DESTINATION field. If the DESTINATION field contains a mnemonic which refers to a register external to the 2901, then whatever is on the data bus, as determined by the SOURCE field, will be loaded into that register. Each of these mnemonics which refers to a destination register outside the 2901 also has an equavalent value associated with it which is the "B" address.

For instance, if "RO" is in this DESTINATION field :

If "B" is used in the 2901 OP-CODE field, it refers to the contents of "RO".

If "B" is used in the 2901 DESTINATION field, the 2901's ALU output is loaded into "RO".

If "A" is used in the 2901 OP-CODE field or the 2901 DESTINATION field, AND the SOURCE field contains a mnemonic other than "RO"-"RF", the "A" also refers to "RO" in the 2901.

If "DOUT" (whose equivalent value is RO) is used in the DESTINATION field, in addition to all the above statements being true, whatever is on the data bus, as decided by the SOURCE field, will be loaded into the register called "DOUT", which is of course, outside the 2901.

The only reason one might decide not to use these features to do two data transfers simultaneously is because it might make the coding rather obscure.

For the case that any register loading is inside the 2901 and this field specifies "B" and sometimes "A" (see SOURCE field) in an obvious way :

STATE OF MICRO-CODE BITS					MNEMONICS	ADDRESS OF REGISTER INSIDE THE 2901 WHOSE CONTENTS "B" AND SOMETIMES "A" REPRESENTS.	
34	33	32	31	30			
0	0	0	0	0	R0	2901 REGISTER 0	DEFAULT CONDITION
0	0	0	0	1	R1	2901 REGISTER 1	
0	0	0	1	0	R2	2901 REGISTER 2	
0	0	0	1	1	R3	2901 REGISTER 3	
0	0	1	0	0	R4	2901 REGISTER 4	
0	0	1	0	1	R5	2901 REGISTER 5	
0	0	1	1	0	R6	2901 REGISTER 6	
0	0	1	1	1	R7	2901 REGISTER 7	
0	1	0	0	0	R8	2901 REGISTER 8	
0	1	0	0	1	R9	2901 REGISTER 9	
0	1	0	1	0	RA	2901 REGISTER A	
0	1	0	1	1	RB	2901 REGISTER B	
0	1	1	0	0	RC	2901 REGISTER C	
0	1	1	0	1	RD	2901 REGISTER D	
0	1	1	1	0	RE	2901 REGISTER E	
0	1	1	1	1	RF	2901 REGISTER F	

For the case that a register outside the 2901 is to be loaded off the data bus and a register inside the 2901 may also be loaded if the 2901 DESTINATION field so indicates, each of the following mnemonics refers to the loadable register outside the 2901 and each mnemonic has an equivalent value which addresses the register inside the 2901 which the symbol "B" refers to, and which the symbol "A" also refers to if the SOURCE field so indicates. This equivalent value is numerically the same as the signal name associated with each nmemonic. For instance, "DOUT" activates siganl "LO-" which which loads the "DATA OUT" register and "DOUT" also sends the address "RO" to the 2901.

STATE OF MICRO-CODE BITS					MNEMONIC (SIGNAL NAME)	MEANING
34	33	32	31	30		
1	0	0	0	0	DOUT (LO-)	DATA OUT. This loads the data bus out to be sent to the PPU over the data path. This also sets Data Command-out. See PPU-Controller Interface specification for details.
1	0	0	0	1	COUT (L1-)	COMMAND OUT. Actually, this is Command-Status Data out. This loads the data bus out to be sent to the PPU over the command-status path. This also sets Command-Status Command-out. See PPU-Controller Interface specification for details.
1	0	0	1	1	SADRL (L3-)	SCRATCH PAD ADDRESS LOW. This loads the data bus into the low 8 bits of the 10 bit scratch pad address register.
1	0	1	0	0	CLDCO (L4-)	CLEAR DATA COMMAND-OUT. There is no data loading associated with this signal. It does only what its name implies. See PPU-Controller Interface specification for details.
1	0	1	0	1	CLCCO (L5-)	CLEAR COMMAND COMMAND-OUT. There is no data associated with this signal. It does only what its name implies. See PPU-Controller Interface specification for details.
1	0	1	1	0	SADRH (L6-)	SCRATCH PAD ADDRESS HIGH. This loads the low two bits of the data bus into the high two bits of the 10 bit scratch pad address register. This way you can select 1 of 0-3 "pages".

STATE OF MICRO-CODE BITS 34 33 32 31 30	MNEMONIC (SIGNAL NAME)	MEANING
1 0 1 1 1	SCR,RAM (L7-)	SCRATCH PAD or RAM. Either mnemonic will load the data bus into the same R.A.M. chips. SCR will also set bit 35. RAM will clear bit 35. SCR uses the scratch pad address register to address the location in R.A.M. to be written into. See SADRL and SADRH in the DESTINATION field. RAM uses the 8 bits in the LITERAL/RAM ADDRESS field to address the top (768-1024) 256 locations of the R.A.M.. See SOURCE field SCR, and RAM.
1 1 0 0 0	DRSEL (L8-)	DRIVE SELECT. The dat bus is loaded as follows :

7	6	5	4	3	2	1	0 = LSB
D E N	T R 2	T R 1	T R 0	N C	S E L	D R 1	D R 0

DEN selects the recording density. Low : downward compatible MFM. High : error correcting GCR.
TRO-2 selects one of 8 tracks. Currently, only 4 exist. They are 0-3.
NC no connection
SEL asserts select to the drives. Selected drive should respond. See SOURCE field, DRSTS.
DRO-1 selects one of 4 drives. If present, this is the drive which responds to SEL.

1 1 0 0 1	DRCON (L9-)	DRIVE CONTROL. The data bus loads as follows :

7	6	5	4	3	2	1	0 = LSB
N C	N C	N C	H S	W E	F W	R V	N C

NC no connection

4) HS HIGH SPEED. If positioned between BOTW and EOTW, the tape will move at 90 ips. Otherwise, it will move at 30 ips.
3) WE WRITE ENABLE. This tells the drive to erase/write onto this tape Allow 5 milli-seconds after clearing this signal for the erase circuitry to wind down.
2) FW FORWARD. This tells the drive to move forward. The drive will not move the tape if EOT is active.
1) RV REVERSE. This tells the drive to move reverse. The drive will not move the tape if BOT is active.

All signals above are active high. If both FW and RV are asserted together, no motion will result.

STATE OF MICRO-CODE BITS 34 33 32 31 30	MNEMONIC (SIGNAL NAME)	MEANING
1 1 0 1 0	WDATA (LA-)	WRITE DATA. The data bus is written into the logic that will write these 8 bits to the tape. This clears the BRANCH CONDITION "WDR".
1 1 0 1 1	RDCON (LB-)	READ CONTROL. The data bus is loaded to control the read logic :

7	6	5	4	3	2	1	0 = LSB
R I N I T	E R D H	I D L E L	C R I N I T	R V R S H	S D U M	C H C R C	R C R C 7

7) RINIT READ INITIALIZE. Active low to to hold RDR, RDL, and RDDX clear. See BRANCH CONDITION SELECT field, RDR, RDL, RDDX.

6) ERDH ENABLE READ DATA HIGH. Active high to enable RDR and RDL to work as described in the BRANCH CONDITION SELECT field.

5) IDLEL IDLE LOW. Active low to turn off (but not clear) the read logic.

4) CRINIT CRC INITIALIZE. Acitve low to clear CRCERR. Dis-assert this before or while asserting CHCRC.

3) RVRSH REVERSE. Active low to decode GCR while the tape is moving backwards and still get forwards information. This has never been used and probably never will be.

2) SDUM SET DUMMY. Active high to instruct the read logic to find a sync zone. This only applies to GCR mode. See FPLA code for details.

1) CHCRC CHECK CRC. Active high to let CRCERR flag an error if the bits read off the tape do not match the CRC which the read logic has generated. This must be asserted in response to the RDR which gives the 2nd to last byte of data proir to the CRC.

0) RCRC7 READ CRC 7. Active high to check only 7 more bits of CRC instead of 8. This should be asserted in response to RDR which gives you the last byte of data prior to the CRC. This applies only to MFM's 15 bit CRC.

To assure that BRANCH CONDITION SELECT "RDL" accurately flags the error that the read logic was not serviced in time, you must always load RDCON before loading RDATA because RDL detects that RDATA came too late.

STATE OF MICRO-CODE BITS 34 33 32 31 30	MNEMONIC (SIGNAL NAME)	MEANING
1 1 1 0 0	WRCON (LC-)	WRITE CONTROL. The data bus is loaded to control the write logic :

7	6	5	4	3	2	1	0 = LSB
RCWL	WCRCH	WINITL	ENWDH	CRC7H	ACEH	SYNCH	NC

7)	RCWL	RESET WRITE CRC LOW. Assert this low to clear the write CRC generater. This should be dis-asserted (high) in response to WDR which calls for the 2nd byte of the data stream this CRC is checking.
6)	WCRCH	WRITE CRC HIGH. Assert this high to send the bits in the CRC generater to the tape instead of the bits in the WRITE DATA register. Do it when WDR calls for the CRC.
5)	WINITL	WRITE INITIALIZE. Assert this low to clear WDR and WDL.
4)	ENWDH	ENABLE WRITE DATA HIGH. Assert this high to let WDR and WDL do their job as described in the BRANCH CONDITION SELECT field.
3)	CRC7H	CYCLIC REDUNDANCY CHARACTER 7 HIGH. Assert this with WCRCH to write the second byte of the 15 bit CRC in MFM density.
2)	ACEH	A.C. ERASE HIGH. Assert this high to tell the write logic to write an Inter-Record-Gap.
1)	SYNCH	SYNC (bytes) HIGH. In GCR mode, asserting this high will let you write the illegal GCR codes. In particular, you will want to do this to write sync zones and blockette headers.
0)	NC	no connection.

To assure that BRANCH CONDITION SELECT "WDL" accurately flags the error that the write logic was not serviced in time, you must always load wRCON before loading WDATA because WDL detects that WDATA came too late.

FORMAT : OP-CODE;

The DESTINATION field told us how to load a register external to the 2901. It also doubled as the field which defined the symbol "B" and sometimes "A", depending on what was in the SOURCE field. So far, "B" has appeared only as an operand in the 2901 OP-CODES. Well, "B" is also a loadable register inside the 2901 and here is how you tell the 2901 to write into whichever register is address by "B". "A" and "Q" also appear in these mnemonics and what they are is best found in the 2901 OP-CODE field.

The "ALU output" is the result of the operation called out in the 2901 OP-CODE.

MICRO-CODE BITS 24 23 22	MNEMONIC	MEANING
0 0 0	Q	The ALU output is written into the Q register and the ALU output is enabled onto the data bus if the SOURCE field hasn't enabled something else onto the data bus.
0 0 1	NOP	THIS IS THE DEFAULT. The ALU output is enabled onto the data bus if the SOURCE field hasn't enabled something else.
0 1 0	B.A	The ALU output is written into the register addressed by "B" as determined by the DESTINATION field. The data bus is driven by the contents of the register addresed by "A" unless the source field has enabled something else onto the data bus.
0 1 1	B	The ALU output is written into the register addressed by "B" as determined by the DESTINATION field. The data bus is driven by the ALU output unless the SOURCE field has enabled something else onto the data bus.
1 0 0	BRQR	Three things : 1) The ALU output is enabled onto the data bus unless something else is already on the data bus. 2) The ALU output is shifted right once and loaded into the register addressed by "B", as defined by the DESTINATION field. 3) Q's contents are shifted right once and loaded into Q
1 0 1	BR	This performs functions 1 and 2 of "BRQR". Q is unchanged.
1 1 0	BLQL	This is the same as "BRQR" except the shift is left once.
1 1 1	BL	This is the same as "BR" except the shift is left once.

Note that shifts in this machine are circular in nature. A bit shifted off one end is shifted into the other end.

```
FORMATS :    #3F;
             107;
           LABEL;
    LABEL AND #3;
```

This field specifies 8 micro-code bits (49-42) which may be used in one or both of two ways:

1. As a literal to be enabled onto the data bus.
2. As the low 8 bits of the 10 bit RAM address. In this case, the top 2 bits will both be 1's.

To use this as a literal field, the SOURCE field must have the mnemonic "LIT" in it. To use these 8 bits to address the 8 low bits of the RAM, either the SOURCE or DESTINATION fields must use the mnemonic "RAM".

To define this field :

Use a hex number preceeded by a "#" sign to explicitly specify the 8 bits. The hex number cannot exceed #FF.

Use a decimal number not to exceed 255 which the assembler will convert to hex.

Use a label. The label must have been defined somewhere and must not exceed #FF.

Use an expression involving any combination of defined labels and constants and operators like "AND or "+" or whatever, not to exceed #FF. See the A2100 assembler manual for more details. There was one ocassion in the self-diagnostics to use the symbol "*" in this field. It is a bit obscure, but is defined. See the JUMP ADDRESS field for details on "*".

Finally, if this field is not specified, it will default to a #00.

FORMAT : OP-CODE;

There is a bit in the hardware called the "condition bit". The BRANCH CONDITION SELECT field tells you how to set and clear this bit. Many of the op-codes which chose the next instruction to execute choose one of two next addresses based on the state of the condition bit. This process is pipe-lined one level so in a first instruction, a particular condition is specified and its state is stored in the condition bit. The second instruction then chooses what the third instruction to be executed will be, based on the state of the condition bit. With that in mind, here are the op-codes :

MICRO-CODE BITS 3 2 1 0	MNEMONIC	MEANING
0 0 0 0	RESETU	RESET UNCONDITIONAL. This is an unconditional jump to address #000. This also clears the stack.
0 0 0 1	CALLC	CALL (a subroutine). If the condition bit is : False : Go to *+1 (the next sequential instruction). True : Go to the JUMP ADDRESS. Push *+1 onto the stack (which is 5 deep).
0 0 1 0	JMPU	JUMP UNCONDITIONAL. Go to the JUMP ADDRESS.
0 0 1 0	NEXT	NEXT puts a JMPU in this field and a *+1 in the JUMP ADDRESS field if nothing else is specified.
0 0 1 1	JMPC	JUMP CONDITIONAL. Depending on the condition bit : False : Go to *+1. True : Go to the JUMP ADDRESS.
0 1 0 0	PUSHLDZ	PUSH and LOAD Z. This pushes *+1 onto the stack. It loads the JUMP ADDRESS into the "Z" register and it does a "Go to *+1".
0 1 0 1	CALLZJ	CALL (1 of 2 subroutines). If the condition bit is : False : Go to Z, push *+1 onto the stack. True : Go to the JUMP ADDRESS, push *+1 on the stack.
0 1 1 0	INDEXC	INDEX CONDITIONAL. This one replaces the 4 LS bits of the JUMP ADDRESS with the 4 LS bits of the data bus in the previous instruction. If the condition bit is : False : Go to *+1. True : Go to the indexed JUMP ADDRESS.

MICRO-CODE BITS 3 2 1 0	MNEMONIC	MEANING
0 1 1 1	JMPZJ	JUMP (to 1 of 2 addresses) If the condition bit is : False : Go to Z. True : Go to the JUMP ADDRESS.
1 0 1 0	RETURNC	RETURN (from a subroutine) If the condition bit is : False : Go to *+1. True : Pop the top of the stack off and go to it.
1 0 1 1	JMPOPC	JUMP-POP CONDITIONAL. If the condition bit is : False : Go to *+1. True : Go to the JUMP ADDRESS, pop the top of the stack off and throw it away.
1 1 0 0	LDZ	LOAD Z. This one is unconditional. It loads the JUMP ADDRESS into the "Z" register and does a "Go to *+1"
1 1 0 1	ESTKLPC	END STACK LOOP CONDITIONAL. If condition bit is : False : Go to the top of the stack (do not pop it off). True : Go to *+1, pop the top of the stack off and throw it away.

While "JMPU" is the default for the dis-assembler, in any source code this field must be called out explicitly in order for the assembler to keep track of when to generate binary code from the 2 lines of source code.

```
FORMATS :     #43F;
               107;
             LABEL;
               *+1;
      LABEL AND #3;
```

The JUMP ADDRESS field's only use is to provide an 11 bit address to the next address logic, if the NEXT ADDRESS SELECT field asks for it. There are many ways to tell the assembler what 11 bits to put here :

Use a hex number preceeded by a "#" sign to explicitly specify the 11 bits. The hex number cannot exceed #7FF.

Use a decimal number not to exceed 2047 which the assembler will convert to hex.

Use a label. The label must have been defined somewhere and must not exceed #7FF.

Use the symbol "*" to indicate this instruction's address. A "*+1" indicates this instruction's address plus one, or the next consequtive intruction.

Use an expression involving any combination of defined labels and constants and "*" and operators like "AND or "+" or whatever, provided the expression does not exceed #7FF. See the A2100 assembler manual for more details.

If this field is not specified, it will default to a #000. Remember that if the mnemonic "NEXT" is used in the NEXT ADDRESS SELECT field, a *+1 will be put into the binary code for this field, but the listing will not indiacte anything is in this field.

FORMAT : OP-CODE;

As mentioned in the NEXT ADDRESS SELECT field, during each instruction there is a bit called the condition bit which will determine which of two next addresses will be chosen. The state of that bit, true or false, is determined by the BRANCH CONDITION SELECT field in the previous instruction. The BRANCH CONDITION SELECT field takes one of 32 bits scattered throughtout the hardware and either stores it in the condition bit register or inverts it and stores it.

STATE OF MICRO-CODE BITS: 41 40 39 38 37 36	MNEMONIC	MEANING
		If not explicitly stated, the state descibed below is the one that means the condition is true.
0 0 0 0 0 0	ALWAYS	ALWAYS TRUE — DEFAULT CONDITION
0 0 0 0 0 1	RTC	REAL TIME CLOCK. This bit is set every 41 & 2/3 micro-seconds (256 instructions) if low density MFM is selected in the Drive Select Register (see DESTINATION field, DRSEL). If high density GCR is selected, this bit is set every 32.5 micro-seconds (200 instructions). CLRTC in the SOURCE field clears this bit.
0 0 0 0 1 0	DPE	DATA PARITY ERROR. You must specify this condition during the instruction which reads data from the PPU (DIN in the SOURCE field) to check if there is a parity error.
0 0 0 0 1 1	WDL	WRITE DATA LATE. If ever you are too late getting the data to the write logic which writes it out to the tape, this bit will latch true. Check it before clearing the write logic after a write to tape.
0 0 0 1 0 0	RDR	READ DATA READY. When you are reading tape, this bit will come true as often as RTC does +/-20%. It tells you to take the data read off the tape with RDATA (see SOURCE field) and this action clears RDR.
0 0 0 1 0 1	WDR	WRITE DATA READY. This bit is set exactly like RTC when you are writing to tape. It tells you to use WDATA (see DESTINATION field) to give more data to be written to the tape. WDATA clears this signal.
0 0 0 1 1 0	RDL	READ DATA LATE. If you are ever too late taking the data read off the tape, this bit will latch true to tell you you dropped a byte. You must check this bit before clearing the read logic following a read.

STATE OF MICRO-CODE BITS:						MNEMONIC		MEANING
41	40	39	38	37	36			If not explicitly stated, the state descibed below is the one that means the condition is true.
0	0	0	1	1	1	PFW		POWER FAIL WARNING. This indicates the 8000 power supplies are abput to fail and/or a RESET is comming. This need only be checked immediately before beginning a write.
0	0	1	0	0	0	BIT0	LSB \	
0	0	1	0	0	1	BIT1	|	
0	0	1	0	1	0	BIT2	|	This condition will be true if the
0	0	1	0	1	1	BIT3	| -	data bus's bit which is called out is
0	0	1	1	0	0	BIT4	|	set during this instruction.
0	0	1	1	0	1	BIT5	|	
0	0	1	1	1	0	BIT6	|	
0	0	1	1	1	1	BIT7	MSB /	
0	1	0	0	0	0	CFO		COMMAND-STATUS FLAG-OUT
0	1	0	0	0	1	CCO		COMMAND-STATUS COMMAND-OUT
0	1	0	0	1	0	CFI		COMMAND-STATUS FLAG-IN
0	1	0	0	1	1	CCI		COMMAND-STATUS COMMAND-IN
0	1	0	1	0	0	DFO		DATA FLAG-OUT
0	1	0	1	0	1	DCO		DATA COMMAND-OUT
0	1	0	1	1	0	DFI		DATA FLAG-IN
0	1	0	1	1	1	DCI		DATA COMMAND-IN
								These conditions are true if the indicated handshake control signal is active. See PPU-Controller Interface specifications for how to use these signals to do command-status and data transfers to/from the PPU.
0	1	1	0	0	0	ZERO		This is true if the ALU output is zero.
0	1	1	0	0	1	CARRY	\/	These two mnemonics set the condition bit
0	1	1	0	0	1	NBORROW	/\	if the carry-out of the ALU is active. *
0	1	1	0	1	0	SIGN		This is true if the Most Significant Bit (MSB) of the ALU output is set.
0	1	1	0	1	1	OVFLW		This is true if the OVERFLOW output of the ALU is set. *

* See the appendix A for a description of the numbering systems used in this controller and an explanation of how to use CARRY and OVFLW.

STATE OF MICRO-CODE BITS: 41 40 39 38 37 36	MNEMONIC	MEANING
		If not explicitly stated, the state descibed below is the one that means the condition is true.
0 1 1 1 0 0	NSTKFULL	True if the SUBROUTINE STACK IS NOT FULL. This will change following the 2nd instruction which pushes or pops the 5th address on/off the stack.
0 1 1 1 0 1	RDD	READ DATA DETECTED. If the read head is currently detecting flux transitions which are data, this signal is true.
0 1 1 1 1 0	CRCERR	CRC ERROR. When you are checking the CRC written onto the tape, this bit will latch true if an error is detected.
0 1 1 1 1 1	RDDX	READ DATA DETECTED CHANGED. Whenever the state of RDD changes, one or more times, this bit will be set. It can be held clear by one of the bits in the READ CONTROL REGISTER. See DESTINATION field, RDCON.
1 0 0 0 0 0	NEVER	This condition is never true.

That takes care of all 32 bits which you can inspect. So far I've described the condition which causes the condition bit to be true. You can also have the same condition cause the condition bit to be false. To do this, just put "N" in front of each mnemonic. In the case of NBORROW and NSTKFULL, remove the "N".

For example, while "BITO" is true if the data bus bit 0 in this instruction is set, "NBITO" will be true if the data bus bit 0 in this instruction is clear.

Puting "N" in front of these mnemonics actually creates another mnemonic whose micro-code bits are the same except bit 41 will be set instead of cleared.

This "N" business does not apply to "ALWAYS" and "NEVER".

Following the last semicolon which terminates the LITERAL / RAM ADDRESS field in first of two lines of source code, and following the last semicolon which terminates the BRANCH CONDITION SELECT field in the second of two lines of source code, there is room for up to 52 characters (including blanks) of comment. If this is not enough, you may write an entire line of comment by using the symbol "*" as the first character which may be followed by up to 105 characters of comment which will appear, as is, in the listing. Also, any completely blank line in the source code will appear as a blank line in the listing and will have no effect on the binary code.

Here is a description of how CARRY and OVERFLOW are used in the two arithmetic systems I've taken the time to consider.

The two number representations I will describe are unsigned positive integers which I represent by "UN" and 2's compliment signed integers which I represent by "SN". Mixing the two kinds of integers in one expression can be fatal, so avoid this. Keep in mind in the following discussion that the function "ADD" also includes "AD1" and "INC". The function "SUB" includes "SB1" and "DEC".

```
UN.ADD.UN = UN : OVERFLOW is meaningless.

                 CARRY may be viewed as a 9th MS bit, and the answer is
                 a 9 bit unsigned integer.


UN.SUB.UN = UN : OVERFLOW is meaningless.

                 Invert CARRY and it may be viewed as a 9th MS bit and the
                 answer is a 9 bit signed integer in 2's compliment form.


   ADD
SN. or.SN = SN : If OVERFLOW = 0, then CARRY is meaningless and the answer
   SUB           is correct in 2's compliment form.

                 If OVERFLOW = 1, then the answer was too large and
                 overflowed into the sign bit. Imagine CARRY as the 9th
                 MS bit and the answer is in 2's compliment form.
```

```
DRSTS : 7) SELD +
        6) TRDY +
        5) DRDY +
        4) EOTW -
        3) EOT -
        2) BOTW -
        1) BOT -
        0) WP -

RDSTS : 7) RDD +
        6) RDR +
        5) RDL +
        4) CRCERR +
        3)  \
        2)   | - The byte counter in the read logic
        1)   |
        0)  /

DRSEL : 7) DENSITY : GCR + / MFM -
        6) TR2
        5) TR1
        4) TR0
        3) nc
        2) SELECT +
        1) DR1
        0) DR0

DRCON : 7) nc
        6) nc
        5) nc
        4) HIGH SPEED +
        3) WRITE ENABLE +
        2) FORWARD +
        1) REVERSE +
        0) nc

RDCON : 7) RDR & RDL & RDDX CLEAR -
        6) ENABLE RDR & RDL +
        5) IDLE -
        4) CLEAR CRC -
        3) REVERSE +
        2) SDUM
        1) CHECK CRC +
        0) CRC7 +

WRCON : 7) CLEAR CRC -
        6) WRITE CRC +
        5) CLEAR WDR & WDL -
        4) ENABLE WDR & WDL +
        3) CRC7 +
        2) A.C. ERASE +
        1) ILLEGAL GCR CODE +
        0) nc
```