```
  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

EXTERNAL
SPECIFICATION FOR :

MICRO-CODE REVISION	B
MICRO-CODE LISTING NAME ...	CTCC1
DEVICE I.D.	#0A 20 00 00
DATE	October 15th, 1981

PROM SET REVISION :

#0125-0048	B
#0125-0049	B
#0125-0050	B

Written by

David W. Milton

TABLE OF CONTENTS

A GENERAL DISCUSSION OF WHAT THE CTC DOES

Following power-up, a hard reset from the PPU, or in response to a store to pseudo-address #0x or #8x, the CTC spends up to 2 seconds performing a self-test whose results are stored in the device I.D. and then awaits instructions from the PPU, taking time out to check the state of the drives every 50 milli-seconds. If Power Fail Warning goes active at any time while the CTC is waiting, the CTC will lock up. A power cycle or a hard reset from the PPU is required to get out of this state. If, for some reason, the PPU sends data bytes to the CTC while the CTC is idle, waiting for activity on the command-status port, the CTC will gobble up those bytes at a speed of up to one byte every 800 nano-seconds. change to "at nominal data rate"

After every operation, the CTC eventually ends up in this same idle loop. If the 8000 has instructed the CTC to interrupt on change of state of drive and one or more of the drives do change state, within 50 milli-seconds the change will be noticed and the CTC will send an interrupt to the system. Next, the CTC will review the state of the drives again and then go back to the idle loop.

When the 8000 wants something of the CTC, the PPU will send a command string to the CTC over the command-status port. The CTC will decode the first byte of this string. If it is a store command, the CTC will wait for 7 more bytes. If it is a load command, the CTC will wait for 3 more bytes. If it is neither and not a NACK, the CTC will NACK it. If it is a NACK, the CTC will ignore it.

Sending a NACK to the PPU means the CTC will gobble up as many bytes as are necessary untill the PPU accepts the NACK. Then the CTC will review the state of the drives as it will do at the end of EVERY thing this CTC knows how to do, and return to the idle loop to await instructions from the PPU.

If the first byte of the command string indicates a load or a store, the CTC will wait for 3 or 7 more bytes. After they have been received, their vertical parity will be checked. If bad, a NACK is sent. If good, an ACK is sent, after which the indicated operation is performed. If for some reason the PPU will not accept that ACK, then the CTC just ignores the whole thing, checks the state of the drives, and goes to its idle loop.

If the command string was a load command, the CTC will send six bytes to the system over the command-status path (the middle four of which are the requested status), check the drives and enter the idle loop.

If the command string was a store command, the CTC has a number of things to do. First, the error codes, motion codes, error bits, word count valid and file mark bits, are all zeroed out. Specifically, that's primary status bits 2, and 8-22.

Second, the CTC checks to see if included with this store command are instructions to select a new drive. If so, and if the drive to be selected is number 0,1,2, or 3, the CTC looks up its record of which track on that drive is to be selected and what recording density on that drive is to be selected and then it puts all this information in the proper places to actually select that drive. If the drive to be selected is not 0,1,2, or 3, then an illegal drive error is flagged, the rest of the store command is not performed, and the store command is considered completed.

Third, the store command is decoded. If it is a legal and defined store, the specified operation is performed. If not legal, an illegal store error is flagged, no operation takes place, and the store command is considered completed.

The terminology "store" to "pseudo-address" implies you are writing data to a memory location, which is exactly how the software types like to think of it. Actually, what you are doing is instructing the CTC to perform an operation. Sometimes there is not data associated with that operation so the 32 bit operand which is received with the "pseudo-address" is ignored. Other times, there is data associated with that operation, such as the byte count when the store command is to write a data record. Then that data is in the operand. See the store command descriptions for details.

For most of the store commands, the CTC just ignores the world and executes the command. For those commands which take a while to do, like rewinding a tape that's at the end, this means that the CTC will not respond to commands from the system over the command status path for (are you ready for this) up to 10 minutes! See the descriptions of the store commands for details on just how long each store may take. The CTC does have timers on the tape, though, so unless there is a CTC hardware problem, it will (eventually) finish executing its store command and its primary status will indicate the success of that operation. There is one exception to this and that is the diagnostic store command to pseudo-address #7x or #Fx which will run forever if you let it and will stop on any activity on the command status port. See the description of the stores for details. Of course, at any time the PPU may assert a hard reset to the CTC causing it to halt, abort any operation and erase any trace of everything. When this hard reset is released, the CTC will start up again with self-test and then the idle loop.

Certain store commands require further communication with the PPU over both the data and command status paths. These are the read and write commands (stores to pseudo-addresses #4x, #5x, #Cx, and #Dx). Since the CTC does not timeout the command status path, it can get hung up indefinitely waiting for the PPU to respond. It does timeout the data path and flags the appropriate error if data is not moved fast enough.

For the write data store command, the CTC will send a GO byte over the command status path when it is ready to accept data over the data path. If the PPU doesn't accept the GO byte, the write is never started and the store command is considered completed. If the PPU accepts the GO byte, then the write will proceed. When the write is finished, and the data transfer is complete, the CTC will send an EOR byte over the command status path. Note that the GO and EOR bytes are always sent for every data write. It does not matter whether there are actual data bytes to be written. (a zero length data record write will have no data bytes transfered unless for some strange reason the operating system tells me to throw away some bytes before beginning the write)

For the read commands (there actually are 3 of them), the data is sent over the data path and when it is all sent, an EOR is sent over the command status path. Note that if the record read was a file mark or a zero length data record, there will be no data sent but the EOR is sent anyway.

One more little detail - if the store command is a write data or file mark, then just before the tape is started (in the case of writing data, just after the GO byte is sent), PFW is checked and if it is active, the CTC just locks up waiting for power to cycle or for a hard reset from the PPU.

Eventually, the CTC will have completed executing the store command or will have detected a FATAL error and given up. Then the CTC will assemble the status of the operation and pack it into the primary status. It will review the state of the drives (noteing any changes), and decide whether to interrupt on completion or not.

The most significant bit of the pseudo-address is the flag to the CTC to interrupt on completion of a store command. For example, "writing" to pseudo-address #40 is an instruction to read the next record on the tape (assuming the operand is zero but don't worry about that here). Writing to #C0 is an instruction to read the next record on the tape and when done, interrupt on completion to tell the PPU you are done.

If no interrupt on completion is requested, when a store command is finished and the drives have been reviewed and the primary status has been updated, the CTC does its normal ending to each operation by reviewing the drives (a second time) and returning to the idle loop.

If an interrupt on completion is requested, after the store command is finished and the drives have been reviewed and the primary status updated, the CTC checks to see if the secondary status has a valid byte count in it. If so, it sends the secondary status to the PPU, keeping the PPU's record of the CTC's status up to date. If there is any problem encountered while sending the secondary status to the PPU, then the CTC will give up and review the state of the drives and then idle. If the PPU accepts the secondary status or if there was no reason to update the secondary status, then the CTC updates the PPU's record of the CTC's primary status and requests the PPU to interrupt the system. Whether or not the PPU accepts the primary status, the CTC is finished with the operation, so it reviews the state of the drives and goes back to the idle loop to await new commands.

When the CTC sends status to the PPU, whether it be in response to a load, an interrupt, secondary status update, a GO byte or an End-of-Record byte (EOR), it always checks to see if for some strange reason the PPU is sending something to the CTC. If so, the CTC's send is aborted, the drives are checked, and CTC enters its idle loop. In the idle loop, the CTC will notice a command is coming.

If the PPU does not handle the command-status handshake properly, the CTC will wait indefinitely for the signals it is expecting. There are no timers on the command-status handshake. This is because if anything goes wrong, eventually the PPU will try to send another command to the CTC. This will cause the CTC to check the drives and return to the idle loop where this new command string will be noticed.

After status has been sent to the PPU, the CTC waits for an ACK or a NACK from the PPU. If NACK, the CTC will attempt to send that string again up to 4 more times, after which it will give up. If ACK, the CTC will continue on doing whatever is next. If neither, the CTC assumes it is a NACK. If it happened to be the first byte of a next command string, the CTC assumes it is a NACK and eventually will receive the second byte of this command string, mistakenly think it is a first byte, not recognize it, NACK it and then receive this next command string properly when the PPU does a re-try on sending this command string.

There are a number of times the CTC may fail while trying to send information over the command status path. Here's what happens with each.

If status sent in response to a load command cannot be sent, then the CTC just gives up, checks the drives, and idles.

If a GO byte cannot be sent, the error is flagged in the primary status, the write is never started, and an interrupt on completion send is attempted if requested. If not requested, it's off to check the drives and idle.

If an EOR byte cannot be sent, the CTC just continues on as if the EOR was sent. Presumably, the PPU will notice this error.

If a secondary status update prior to interrupting on completion or the primary status interrupt on completion cannot be sent, the CTC gives up and checks its drives and idles.

GAP DISCUSSION

A GENERAL DESCRIPTION OF TAPE MOVEMENT AND GAP SPACING

Formatting on the CTC tapes creates tapes with approximately 1.2 inch Inter-Record-Gaps (I.R.G.) between each record, whether that record be a data record, a file mark, written in the low density Modified Frequency Modulation (MFM) format or written in the high density Group Code Recording (GCR) format.

The MFM format records at 6,400 flux reversals per inch and 6,400 bits per inch which is 800 bytes per inch. The tape speed is 30 inches per second. The data transfer rate is 192,000 bits per second or 24,000 bytes per second. A record's length is 30 bytes longer than the number of bytes of data. Those 30 bytes are sync zones, I.D. zones, word count and check characters. For all practical purposes, a record is (# of bytes/800) inches long.

The GCR format records at 10,200 flux reversals per inch and effectively 8,200 bits per inch. The tape speed is 30 inches per second. The data transfer rate is 246,000 bits per second or 30,750 bytes per second. A record's length is considerably more complicated to calculate. Where B = number of data bytes in a record, the length is exactly :

```
        (562 + 268*(INT(B/256 + .999)))/1024    for B>0
```

That's approximately .26 inches for each 256 bytes of data plus .52 inches of error correcting information.

The I.R.G. is approximately 1.2 inches for both recording formats, because that's how big the gap has to be so that all three heads (the read, write and erase heads) can come to a stop in the gap.

There are two little markings on the tape, 48 inches away from either end of the tape. If drive status BOTW is active, you are between the beginning of tape and that little hole. Likewise for EOTW. Since the ends of the tape are the parts of tape most exposed to abuse, it is adviseable not to write or read these sections of tape. The CTC takes care of not writing or reading the 48 inches at the beginning of the tape. It is up to the software driver to prevent or minimize writing near the end of the tape.

The fast forward, rewind, and cycle commands all ignore these warning holes near the ends of the tape. They deal with the real (physical) beginnings and ends of tapes. The CTC always ignores the warning hole near the end of the tape. If you are positioned at or near the beginning of the tape, here's what the CTC does about it.

If you request a read operation (read/skip/search forward), the CTC will time a wait for the LOAD POINT hole which is 48 inches from the beginning of the tape. If it is not found, FATAL error #08 (load point late) will be flagged and the operation aborted, as well as "tape lost" flagged in the motion code. When the load point is found, then the CTC begins looking for records to read off the tape.

If you request a write when the CTC is located at or near BOT, the CTC will erase up to the load point (again while timing its arrival, flagging load point late if it is late) and then erase one more inch before beginning to write on to the tape.
In the reverse direction, the CTC ignores the load point hole.

In the following discussion of moving in and out of gaps, keep in mind that what I am saying applies to store commands that move the tape backward or forward while paying attention to the records on the tape. For instance, the fast forward, rewind, and cycle commands all stop at an end of the tape so while the motion code in the primary status will be in states that I discuss below, concepts of gap spacing simply do not apply. Likewise, the erase command extends the gap but leaves the heads positioned, effectively in the same place with respect to the records on either end of this extended I.R.G., so gap position has not changed.

There are lots of ways that you can enter a gap, and there are lots of things you can do to get out of that gap. Some combinations work, some don't. Here are all the rules you need to know as well as other helpful little tid-bits in order to never do any thing wrong as well as answer any questions I could.

There are two ways the heads can be positioned in a gap. They are called in this text "properly" and "improperly" positioned in a gap. Improperly positioned is positioned 1/3 inch beyond the properly positioned point. If a gap is 1/3 inch or more larger than the nominal gap size of 1.2 inches, while you can still be improperly positioned in the gap, it will not matter. Unfortunately, there is no way to know how big a gap is, so if you are improperly positioned in a gap, you must assume it is a normal sized gap which will mean you cannot write out of it. In this and all future documentation, if there is a reason to be wondering if you are properly or improperly positioned in the gap, being positioned before the load point is synonomous with properly positioned. Remember this because it is not mentioned again anywhere.

After any operation, first look at the tape motion bits 9 & 10 in the primary status. If they are 00 (no tape motion or in same I.R.G.) then you are still positioned in the gap the same way you were before. If they are 11 (tape position lost), it doesn't really matter where you are in what gap because you're lost.

If the tape motion bits are 10, you have moved backwards and are unconditionally properly positioned in the previous I.R.G. Backwards tape motion is like that. It doesn't make mistakes with respect to gap position.

If the tape motion bits are 01, you have moved forward. Presumably, you began this forward motion from a gap, or the beginning of tape. If you did not begin in a gap, that means you moved forward after an error during a write which you did not fix and you will get all sorts of trouble next time you try and read this tape.

If the operations requesting this tape motion are file mark searches, then you must keep in mind that the motion codes 01 or 10 mean you have moved one or more gaps forward or reverse, and you have no way of knowing how many gaps you have moved. If there are no errors flagged which in this case is synonomous with "If the file mark bit is set", then you know you are in the I.R.G. just beyond or before the file mark you have found. If the file mark bit is not set, then there was an error and you have stopped just beyond or before the record which you could not identify as a file mark or as a data record, unless of course, the motion code is 11 - lost. Then you are lost.

If there are no FATAL errors flagged in the primary status bits 11-15, then you are properly positioned in the next I.R.G. It does not matter if there are hardware/micro-code errors flagged in bits 16-20, you will still be properly positioned in the gap. The error bit 2 cannot be relied on to determine if you are properly positioned in a gap or not because there are certain FATAL errors which do not set this bit, and hardware/micro-code errors in the primary status bits 16-20 do set error bit 2 but do not improperly position you in a gap.

If there is an error flagged in primary status bits 11-15, then where you are depends on what operation was just performed. If the operation was not a write data or a write filemark, you will be impropery positioned in the gap. That is, if you did a read, forward skip, or forward file mark search, when an error is detected, the micro-code assumes it cannot read the record, looks for 1/3 inch of I.R.G. and stops in it. This is why improperly positioned in an I.R.G. means being 1/3 inch past where you ought to be.

If the error in bits 11-15 occurs after a write data or a write filemark, the code just stops everything as soon as the error is detected. Now for short data records and filemarks, by the time the tape has stopped, it might be in the right place in the I.R.G., but this is really, logically, not the case. You really should go by the rule that any time an error occurs during a write, you are crapped out somewhere in the middle of what you are doing and you must do a backspace followed by a re-try or an erase. Anything else will leave crap on the tape which will cause problems later.

That takes care of all the ways you can get into a gap. Now there are three ways you can get out of a gap. You can read/skip/search forward, skip/search reverse, or write/erase forward.

If the previous operation was a write with a FATAL error detected, you must do a skip reverse. This skip reverse will probably, though not necessarily, have some FATAL error also, but nevertheless, you will be positioned in the I.R.G. preceeding the bad record just written. You must either erase it or write over it until no error is detected. You must never leave an erroneous record on the tape.

Except for the write error case above, the moving out of a gap is easy. You can always read/skip/search forward or reverse out of any gap and expect to end up in the next gap with the primary status indicating the success of reading the record in between. It does not matter whether you were properly or improperly positioned in the gap, you can still read/skip/search forward or reverse out of it.

You must never write out of a gap in which you are improperly positionec Doing so will chance leaving crap in the gap, which will make this tape unreadable in the future, even though the read-after-write will not detect the error. This is true for writing data, writing a file mark, and erasing.

If you switch tracks while in the middle of a tape, you will have no idea where you are. Writing from this point would be fatal. Reads/skips/searches will probably find an error right away, and after that be okay, because you will now be in a gap. In general, you should pair track selects with rewinds so you can read the entire track. It does not matter what order you do them in.

To summarize :

If a FATAL error occurs during a write, you must do a skip reverse anc a write re-try or an erase.

If a FATAL error occurs during a forward read/skip/search, you will be improperly positioned in the gap following the record with an error in it.

Reverse skip/searches always leave you properly positioed in the gap.

Switching tracks while in the middle of the tape will probably leave you improperly positioned in a gap or in the middle of a record.

You must never write out of a gap in which you are improperly positioned.

You may read/skip/search forward or reverse out of any gap, whether you are properly positioned in it or not.

There is a routine in the micro-code called "GAPFINDER". As soon as a data record is detected, the GAPFINDER routine is turned on. Its function is to monitor the tape to look for the absence of data for 1/3 inch. It is used to find the end of a record in which an error has been detected. If a big dropout occurs in the middle of a long record, this routine will find it and stop the tape in it, thinking it's an I.R.G. Basically, an I.R.G. is defined by this controller as any 1/3 inch section of tape that has no data on it. This GAPFINDER routine always monitors what's going on so that if a big dropout does occur, it is always the same routine which finds it, whether you are skipping reverse, reading, searching for a file mark or whatever. This is how I manage to know where I am on this tape, even when records are all fouled up.

LOADS FROM THE CARTRIDGE TAPE CONTROLLER

The CTC recieves a 4 byte string from the PPU. The first byte is a #62 indicating this is a load. The second byte is the preudo-address from which the load is taken. The third byte is undefined and the fourth byte guarantees vertical parity. There are six pseudo-addresses from which information may be loaded into the 8000 computer system. They are : #00, #10, #20, #30, #40 and #50. In that 8 bit pseudo-address, the CTC only looks at three of those bits (xaaa xxxx).

BRIEFLY, HERE ARE THE LOADS :

```
#0x or #8x : DEVICE I.D. = #0A 20 00 00
                            |  |  |  |
                            |  |  |  |--not used
                            |  |  |
                            |  |  |-----self-test results (#00 = passed)
                            |  |
                            |  |--------micro-code revision level
                            |
                            |-----------device identifier (#0A = a CTC)
```

more on next page

#1x or #9x : PRIMARY STATUS, listed by bit (0=most significant bit)

0 : drive(s) changed state since I last interrupted
1 : interrupt enable on drive change state-
2 : error detected, re-try required.
3-5 : selected track number
6-7 : selected drive number

8 : file mark found
9-10 : motion code : 00 - in same gap
01 - moved forward out of gap
10 - moved backwards out of gap
11 - lost
11-15 : FATAL error codes, shown with possible motion codes

#00 :	00,01,10	00000	- no error detected
#01 :	00	00001	- illegal drive
#02 :	00	00010	- illegal store
#03 :	00	00011	- illegal track
#04 :	00	00100	- illegal tape motion
#05 :	00	00101	- no GO byte
#06 :	11	00110	- timeout waiting for an end of tape
#07 :	11	00111	- offline (tape no longer installed)
#08 :	11	01000	- laod point hole late
#09 :	11	01001	- sync zone late
#0A :	01	01010	- unexpected I.R.G. found
#0B :	01	01011	- drop-out during a write
#0C :	01,10	01100	- I.D. zone late
#0D :	01,10	01101	- I.D. zone bad
#0E :	01,10	01110	- byte count check bad
#0F :	01	01111	- MFM data CRC bad
#10 :	01	10000	- GCR reverse byte count bad
#11 :	01	10001	- trailing I.D. zone bad
#12 :	01	10010	- trailing sync zone bad
#13 :	01	10011	- GCR blockette number wrong
#14 :	01	10100	- GCR crammed byte count not found
#15 :	01	10101	- GCR read bad but recoverable
#16 :	01	10110	- GCR read unrecoverable
#17 :	11	10111	- unexpected EOT found
#18 :	11	11000	- unexpected BOT found
#1F :	11	11111	- last operation was a write repeat

16 : parity error on data coming from PPU
17 : overflow error. All data not successfully sent to PPU
18 : underflow error. All data not successfully sent to CTC
19 : WDL error - CTC's write logic goofed up
20 : RDL error - CTC's read logic goofed up
21 : last operation was a crammed byte count read
22 : byte count in secondary status is valid
23 : GCR densiy (1) / MFM density (0)

24 : EOT
25 : EOT Warning
26 : tape installed
27 : BOT Warning
28 : write protected
29 : tape position NOT known
30 : drive selectable
31 : BOT

#2x or #Ax : SECONDARY STATUS (byte count)

#3x or #Bx : TERTIARY STATUS (GCR error correcting information)

#4x or #Cx : DRIVE'S STATUS'ES (drive 0 is the most significant status)

#5x or #Dx : QUINTARY STATUS (reports nature of unrecoverable GCR error)

That, briefly, is all the defined loads. In fact, the CTC only inspects bits 1-3 of the pseudo-address, so if you do a load from #44, #CF, and so on, you'll get the status of all four drives. If you do a load from #60, or #70, you will get no response from the CTC except that it will "ACK" the PPU.

Requesting a load from the CTC is an entirely passive operation and has no effect on the state of the CTC, except that when the requested status has been sent, the CTC will review the state of all four drives as it does at the end of EVERY operation, successful or not.

The initial state (following power-up, a hard reset from the PPU, or completion of self-test in response to a store to #0x or #8x) of all the loadable addresses is as follows :

```
#00 : #0A 20 00 00
#10 : #80 00 00 xx   (xx depends on the state of drive 0)
#20 : #00 00 00 00
#30 : #00 00 00 00
#40 : #xx xx xx xx   (xx depends on the state of the drives)
#50 : #00 00 00 00
```

Following is a detailed discussion of everything you will ever need to know about the bits in the status words described briefly above.

Load #00 : Concerning the device I.D.

```
            It now looks like "#0A 20 00 00"
            #0A is the identifier that this is a CTC
            Here's where that "#20" comes from
              In a number system where you count 0,1,2,....D,E,F
              the 2x means this code runs on the third P.C. board to
              work (not to be confused with rev. 3)
              the x0 means it is the first version of the micro-code to
              run on that P.C. board.

            Concerning the self test :

             #0A 20 00 00
                    ||
                    ||---results of self-test

              The exact meanings of the bits in a failed self test are :

              bit 16 : Read logic cannot reset
                  17 : WDR/WDL signals don't work right
                  18 : RTC signal doesn't work right
                  19 : 2901 register address test failed
                  20 : Push/pop error detected
                  21 : RAM test failed
                  22 : Micro-processor test failed
                  23 : Data bus test failed

            The last byte (bits 24-31) has no meaning.
```

Load #10 : PRIMARY STATUS

bit 0 : The DRIVE HAS CHANGED STATE bit is set whenever the CTC finds the 8 bits of drive status do not exactly match its record of what that drive's status was. This may happen whenever the tape is moved, when some operator inserts or removes a cartridge, and other times. This bit is cleared after an "ACK" is received from the PPU in response to the CTC sending an interrupt to the system which may be an interrupt on completion or on change of state of drive.
The CTC reviews the state of the drives every 50 milli-seconds when it is otherwise idle, and at the completion of any tape motion (prior to interrupting on completion) and after the completion of any store command (after the interrupt on completion is sent or after deciding not to send an interrupt on completion) and after responding to a load command from the system and after sending an interrupt on drive state change to the system. This bit really means one or more of the drives have changed state.

bit 1 : An INTERRUPT ON DRIVE CHANGE STATE is ENABLED (set) by storing a #xx xx xx 09 to pseudo-address #3x or #Dx to request an interrupt on completion. The bit may be cleared by storing a #xx xx xx 08 to pseudo-address #3x or #Dx to request an interrupt on completion. This bit will be cleared in two other ways. Whenever the CTC interrupts the system on a change of state of drive, this bit will be cleared after that interrupt is sent, whether or not the PPU accepts the interrupt request. Whenever the CTC interrupts on completion of a store command, this bit will be cleared before that interrupt is sent whether or not a drive has changed state. It will be futile to request an interrupt on completion of the store command that sets this bit.
When both bits 0 and 1 of the primary status are set, the CTC will attempt to send an interrupt on change of state of drive, which will clear bit 1 and if successful, will clear bit 0.

bit 2 : An ERROR REQUIRING A RE-TRY has occurred. This is a smart bit. This bit is cleared at the beginning of each store command, and it is set at the end, if appropriate. In general, this bit will be set if the system tells the CTC to do something which does not make any sense, and when the intended operation can not be completed successfully. See the specific store command descriptions for details. Here are some examples :

ERROR BIT is SET if :
- you attempt to select track 5
- you attempt to select drive 7
- you attempt to write to a write protected tape
- a data crc error occurs during a read or write operation
- you do a store to loaction address #1x or #9x.

ERROR BIT is NOT SET if :
- a data crc error occurs during a skip forward
- the trailing ID zone is bad during a read/skip/search
- recoverable GCR error which is confined to the XOR blocks
- the leading ID is bad during a skip forward

bit 8 : FILE MARK. The CTC is stopped in the I.R.G. immediately following (if last store was a forward read/write/skip/search file mark) or immediately preceeding (if last store was a skip reverse or search file mark reverse) a file_mark.

bits 9-10 : MOTION CODE. The motion code is designed to tell you what the last operation did to the position of the tape. It tells you that you moved forward into the next I.R.G., or you moved backwards into the next I.R.G., or you are lost, or you didn't move. That's all pretty clear, but here are the funnies.
Any time you do an erase (store to #6x or #Ex), while the tape will have moved (provided you requested an erase of more than zero (0) inches), you will still be in the same I.R.G., so the motion code will be 00. In the case of a file mark search, while you will probably have moved many I.R.G.'s, you aren't lost so a 01 or a 10 will be here, depending on if this was a forward or reverse search. The fast forward, rewind, and cycle commands do not have anything to do with I.R.G.'s. Still, providing a timeout or offline error does not happen (if one does, the motion code will be 11 - lost), you'll get 01 in response to a successful fast forward, and 10 in response to a successful cycle or rewind.

bits 11-15 : ERROR CODES
These error codes are FATAL errors. When one of these errors is detected, the requested operation is aborted. Depending on the particular FATAL error, this may mean nothing was done (as in the case of an illegal store), something was half done (a byte count check bad error), or it was all done (a trailing sync zone error). If tape motion is involved in the command which gets a FATAL error, the motion code will tell what happened to the tape.
When the FATAL error code is #14 (recoverable GCR error), the operation will run to completion, but as with all FATAL error codes, when the tape motion bits indicate you moved forward out of a gap (01), you will be improperly posiotioned in the gap following the record which had this recoverable error.
The first thing any store command does is zero out all error records. Then it tries selecting the new drive if requested. If no error is detected then, the new drive will be selected and a FATAL error which occurs later will abort the rest of the store command, but the new drive will still be selected.
The error code is zeroed at the beginning of each store.

Error #01 : ILLEGAL DRIVE - There are 3 bits with which to specify what drive you want to select. If you select drives numbered #4, #5, #6, or #7, you will get the illegal drive code. If you select drive #3 when only drives numbered #0-#2 are installed, that error will be apparent in the status of the drive which isn't there. If you chose drive #5 but do not assert the bit indicating the CTC should select that drive, no error will occur. If an illegal drive select is attempted, the store command will not be executed, the selected drive will not change, and an interrupt on completion will be sent if requested. This error always sets the error bit 2 in the primary status.

Error #02 : ILLEGAL STORE - writing to pseudo-addresses #1x and #2x is illegal. If you are writing to #3x, bits 0-23 of the operand don't matter. However, it will be an illegal store if bits 24-31 are : #0A, #0B, #0E, #0F. If bits 24-31 are greater than #18, that also will be an illegal store.
If you write to #4x with bit 7 set (a crammed byte count read), you'll get an illegal store if GCR is not selceted.
If you are writing to #4x (a Read command) and bits 2,3,4,5,6, or 7 of the operand are set, that also will be an illegal store. No action is taken on an illegal store, except if the store included selecting a legal drive, that drive will be selected. If requested, an interrupt on completion of this illegal store will be sent. Note also that what is said about pseudo-addresses #1x, #2x, #3x, and #4x also applies to #9x, #Ax, #Bx, and #Cx respectively. (#9x is a store to #1x while requesting an interrupt on completion, and so on.)
This error always sets the error bit 2 in the primary status.

Error #03 : ILLEGAL TRACK - writing a #14, #15, #16, or #17 to address #3x or #Dx would be selecting tracks #4, #5, #6, or #7 if they existed, which they don't so you will get an illegal track error and an interrupt on completion if requested.
This error always sets the error bit 2 in the primary status.

Error #04 : ILLEGAL TAPE MOTION. This one happens when you attempt to do an operation to a tape which either doesn't make sense or is not allowed :
- Writing to a write protected tape
- Requesting forward tape motion of a tape at End-of-Tape (except you can Fast Forward and Erase-to-EOT when already at EOT, it just won't take very long !)
- Requesting reverse tape motion of a tape at Beginning-of-Tape (except for REWIND, which will respond by telling you you are now at BOT)
- Attempting a write, read, skip, or file mark search of a tape whose position is not known
- Attempting any operation on a drive with no cartridge installed or on a drive that is not installed.

Of course, the tape will not be moved and you'll get your interrupt on completion if requested.
This error always sets the error bit 2 in the primary status.
In the above discussion, "operation" implies there will be a tape motion. for instance, you can select a track on a drive, regardless of whether the drive is there with a tape installed or not.

Error #05 : NO GO BYTE. If the PPU does not "ACK" the GO byte, the CTC sends just prior to a write to tape, the entire operation is aborted (no tape motion results) and an interrupt on completion is sent if requested.
This error always sets the error bit 2 in the primary status.

Error #06 : TIMEOUT. During fast forwards, rewinds, cycles, and Erase-to-EOT, 7 minutes are allowed to find the appropriate end of the tape. If not found, this error is flagged, the operation is halted, and an interrupt on completion is sent if it was requested.
This error always sets the error bit 2 in the primary status.
This error will also flag "tape position lost".

Error #07 : OFFLINE. If during any operation to tape the drive goes offline (malfunctions or someone pulls out the cartridge), this error is flagged and the interrupt on completion is sent if requested.
This error will usually set error bit 2 in the primary status.
This error will always flag "tape position lost".
See each store's description for details.

Error #08 : LOAD POINT LATE. The load point is a little hole 48 inches from the physical beginning of tape. If it is not found shortly after a tape motion starts at the beginning of tape, this error is flagged, the operation is halted, and an interrupt on completion is sent if requested.
This does not apply to rewind, cycle, fast forward or Erase-to-EOT commands. They ignore the Load Pont.
This error always sets the error bit 2 in the primary status.
This error will also flag "tape position lost".

Error #09 : SYNC ZONE LATE. During reads and skips, if a sync zone is not found within about 25 inches (from starting position or load point if at BOT) or if during file mark searches an I.R.G. lasts much more than 25 inches or if during a write the sync zone is not within 10% of where it is supposed to be, this error is flagged, the operation is halted, and an interrupt on completion is sent if requested.
This error always sets the error bit 2 in the primary status.
This error will also flag "tape position lost".
This is because it is possible to pass over a record while the tape is stopping and not know it, thus being "lost".

Error #0A : UNEXPECTED I.R.G. While skipping forward over or while reading a data record, if this record ends prematurely (due to a goof-up when it was written) or a big drop-out is in the middle of this record, this error is flagged, the tape is stopped, and an interrupt on completion is sent if requested.
The way this error is detected is by a routine which monitors the data detected signal from the drive and if it is gone for 1/3 inch while the code otherwise expects data to be present, it is assumed to be an I.R.G. so it stops improperly positioned in it.
This error usually sets the error bit 2 in the primary status.
See each store's description for details.

Error #0B : WRITE DROP-OUT. If any dropout occurs during a write, the tape is stopped imediately, the operation halted, this error flagged, and an interrupt on completion is sent if requested.
A backspace and re-try or erase is mandatory after this error.
This error always sets the error bit 2 in the primary status.

Error #0C : I.D. ZONE LATE. When the I.D. zone is not found shortly after the sync zone ends, this error is flagged. If it is a write, the operation is halted and a backspace/re-try/erase is required. If it is a read, skip, file mark search, the tape advances to the next I.R.G. and stops improperly positioned in it. This means that you must not do a write operation out of this gap but you will be able to read in either direction out of this gap. As always, an interrupt on completion will be sent if requested.
This error may set the error bit 2 in the primary status. See each store's description for details.

Error #0D : LEADING I.D. ZONE BAD. This occurs when the leading I.D. zone is not recognized or is the wrong one during a write operation. The result and recovery is the same as error #0C.

Error #0E : BYTE COUNT CHECK BAD. This occurs when the check characters on the byte count flag an error. The result and recovery is the same as error #0C.

Error #0F : DATA CRC BAD. When a data error is indicated, the result and recovery is the same as error #0C. Note this is only true of the MFM recording density. In the GCR mode, the error may not be an unrecoverable one so a GCR data error is flagged in the GCR errors which are numbers #13,14,15, and #16.

Error #10 : REVERSE BYTE COUNT CHECK BAD. This error only applies to the GCR recording density. It is self-explanatory, but there are some funnies associated with it. If this error occurs during a write, as with all write errors, the write is halted immediately and a back-space/re-try/erase is required as will be indicated by the error bit 2 in the primary status. If this occurs during a read or skip forward, (it will not occur during a file mark search) the error is not a fatal one since the data has already been recovered by now. Therefor, the error bit 2 is not set. However, the error indicates the tape is not being read accurately so rather than trust the code's byte counter to know when the end of the record occurs, the GAPFINDER will do the job of finding the end of the record. This means that the heads will be improperly positioned in the I.R.G. And this means that you cannot write out of this gap but you will be able to read/skip/search in either direction out of this gap. This problem only occurs when a user wants to write beyond the last record on the tape and that record is readable but not 100% right. The only way to do that cleanly is to skip back over the last record, and re-write it and then continue writing as had originally been planned. It does not matter if there is an error on the skip reverse because reverse tape reads and skips always stop properly in the I.R.G.
In any case, after this error is detected and the tape is stopped improperly positioned in the I.R.G., an interrupt on completion will be sent if requested.

Error #11 : TRAILING I.D. ZONE BAD. The error is self-explanatory. It can happen in both MFM and GCR modes, and it can happen during a forward file mark search. The result/recovery/funnies are the same as error #10.

Error #12 : TRAILING SYNC ZONE BAD. The error is self-explanatory. The result/recovery/funnies are the same as error #11. Note also that the last 8-10 flux transitions of the trailing sync zone are not checked. This is for a few reasons. One is that due to the levels of pipelining in the hardware, by the time those flux transitions are decoded, the record has ended so it looks to the code like there's been a drop-out in the trailing sync zone. More importantly, though, the STC 1&2 controllers on the 4000/5000/6000 computer systems BTI has sold for years do not write all of the last few flux transitions and I didn't want to generate lots of relatively unimportant errors when reading the tapes made on the earlier controllers. Also, the prestress cannot be guaranteed for the last flux transition written by this CTC, so it's best not to check it.

Error #13 : BLOCKETTE NUMBER WRONG. During forward read/skip/write GCR operations, each blockette begins with a blockette number and ends with a CRC which checks that number, as well as the data in the blockette. Once the CRC ckecks okay, if the blockette number is not the one expected (ie. one less than the last blockette) something is wrong, so this error is flagged and the operation is aborted.
The result and recovery is the same as error #0C.

Error #14 : GCR CRAMMED BYTE COUNT CANNOT BE FOUND. During a cramed byte byte count operation, the CTC has to find a particular blockette of data and read from there. If that blockette cannot be found, this error is flagged, the error bit is set and the tape stops improperly positioned in the gap following the record with the error in it, and an interrupt on completion is sent if requested.

Error #15 : RECOVERABLE GCR READ. During skip forward and read GCR, this error indicates the record is not perfect but recoverable. In the case of a read, up to 512 bytes of data may be wrong. You may backspace and retry the read, or you may request the tertiary status and the XOR blockettes and that will give you the information to correct the bad data. See the tertiary status for more details. You will be improperly positioned in the gap following this record, and an interrupt on completion will be sent if requested.
This error may set the error bit 2 in the primary status. See each store's description for details.

Error #16 : UNRECOVERABLE GCR ERROR. During a write GCR, this error indicates a data error has been detected. During a read or skip forward, this indicates too many data errors have been detected, or a recoverable error and an overflow have occurred. The quintary status contains the information which tells where the "one too many" error occurred. You will be improperly positioned in the gap following this record, and an interrupt on completion will be sent if requested. This error may set the error bit 2 in the primary status. See each store's description for details.

Error #17 : UNEXPECTED EOT. If End-of-Tape is bumped into while reading, writing, skipping, or searching for a file mark, this error is flagged, the operation is halted, "tape position lost" is flagged, and an interrupt on completion is sent if requested. This error may set the error bit 2 in the primary status. See each store's description for details.

Error #18 : UNEXPECTED BOT. If Beginning-of-Tape is bumped into while doing a reverse skip or file mark search, this error is flagged and the same is done as in error #17.

Notes on error #7, #17, #18 :

These errors are "offline", "unexpected EOT", and "unexpected BOT" respectively.

During rewind, fast forward, cycle, and erase, "offline" is flagged if it occurs before the completion of the designated command, and it will always set the error bit. If the tape goes offline after the operation is completed, there will be no error reported.

During write data and write file mark commands, after the operation is over and the tape is stopped, the drive status is checked and if offline or EOT is found, that will be flagged as a FATAL error, overwriting any previous errors, and setting the error bit.

During all other tape operations which involve tape motion, when the operation is over and the tape is stopped, offline and EOT or BOT are checked and overwrite any FATAL error that may have occurred. The difference is that in not-write cases, the error bit is only set if the offline, BOT or EOT error overwrites an error that would have set the error bit. Thus, if the tape goes offline during a read after all the data has been sent and while the trailing I.D. zone is being checked, then the error #7 (offline) will be flagged in the FATAL error code, but the error bit will not be set because the read did successfully complete.

In all cases, there is a small but finite time between the "end of the operation" and when the primary status update occurs. In that time, it is possible for the status of the drive to change. In this case, it is possible for there to be no offline, EOT, or BOT error when the drive status indicates one of these conditions does exist, or visa-versa.

REMEMBER WE ARE STILL DOCUMENTING THE BITS IN THE PRIMARY STATUS

bit 16 : PARITY ERROR. If a parity error is detected on the data coming from the PPU during a write, this bit will be set, the error bit 2 in the primary status will be set, the write will continue to fill out the record as its byte count says it will be, the data byte with the bad parity will be written onto the tape, thereafter the record will be filled with zeroes, no more data bytes will be fetched from the PPU, and everything else will continue to happen as if no error had occured. If this record is not re-written or erased, there will be no indication on the tape that there is anything funny about this record.
Note if some FATAL error occurs, that will be handled normally and also recorded in the 5 bits of FATAL errors.
This bit is cleared by the next store command.

bit 17 : OVERFLOW. During a read, if all the data does not get sent to the PPU successfully, this bit will be set and no more data will be sent to the PPU. (the PPU should also indicate a byte count problem) The CTC will continue to read the tape normally, flagging any FATAL errors that may or may not occur. This bit is also set during sending the XOR bytes and during a crammed byte count read if there is an overflow. In all these cases, the error bit is set. This bit can, under certain tape error situations, get set during a skip forward, but in that case the error bit will not be set. This bit is cleared by the next store command.

bit 18 : UNDERFLOW. If the PPU does not send data fast enough during a write to tape, this bit will be set. Except that this is bit 18 instead of bit 16, the CTC's response to this error is the same as its response to a parity error.
This bit is cleared at the beginning of each store command.

bit 19 : WDL. That's Write Data Late which is a flag in the hardware to tell the micro-code that it is not feeding data to the tape fast enough. It is a bonafide micro-code goof-up and should never happen, but if it does, the record which was being written at the time will probably have a detectable error in it, but it might not. So the only thing to do is a re-try, which isn't so bad because when this bit is set, error bit 2 in the primary status is also.
This bit is cleared by the next store command.

bit 20 : RDL. That's Read Data Late which is a flag in the hardware to tell the micro-code that it did not take the data being read off the tape fast enough. This is another bonafide micro-code goof-up and should never happen and may or may not have some other error detected with it.
This bit is cleared by the next store command.
This error may set the error bit 2 in the primary status.
The only cases in which this bit will not set the error bit are when this occurs during a forward or reverse skip, in which case it doesn't matter because the skip did occur properly.

bit 21 : LAST OPERATION WAS A CRAMMED BYTE COUNT. This is self-explanatory, and is only here to remind us all that the byte count in the secondary status is not one read off the tape, but rather the very same one that was crammed down the CTC's throat. This bit is cleared bye the next store command.

bit 22 : BYTE COUNT VALID. This bit will be set when a read or forward skip or a write or a backward skip (in the GCR mode only) passes over a data record whose byte count was readable. This bit indicates the secondary status has this valid byte count in it. This bit will be set any time no errors occur as well as most other times, because most tape FATAL errors will occur in the larger data field.
This bit will aslo be set during a crammed byte count read, but in that case the byte count is the same one as was crammed down the CTC's throat by the operating system.
This bit will be cleared by the next store command.

bit 23 : 0 : Indicates MFM recording/reading density is selected
1 : Indicates GCR recording/reading density is selected
If you attempt to read a tape with the wrong density selected, you can bet you'll get all sorts of FATAL errors like SYNC ZONE LATE or I.D. ZONE LATE or I.D. ZONE BAD. This bit is set by storing a #xx xx xx 0D to pseudo-address #3x or #Dx to request an interrupt on completion. This bit is cleared by storing a #xx xx xx 0C to either of those two pseudo-addresses above.

bit 24-31 : These are the status bits of the selected drive. See the description of Load #40 for a complete explanation of these bits. Briefly :

```
24. EOT +
25. EOTW +                    (+ = active state = 1)
26. Tape Installed +          (- = active stats = 0)
27. BOTW +
28. Write Protected +
29. Tape Position Known -
30. Drive Selectable +
31. BOT +
```

Load #20 : BYTE COUNT

When bit 22 in the primary status is set, the secondary status will have the byte count which was read off the tape during the last store command which must have been a read, write, skip forward, or a skip reverse (for GCR only).
In the case of a crammed byte count read, this secondary status just echos back the crammed byte count. Since only 16 bits are allotted for the byte count, the format for the 32 bits which may be loaded into the 8000 will be #00 00 xx xx. Since the 4000/5000/6000 computer systems thought in 16 bit words, and the 8000 thinks in 8 bit bytes, you may wonder how the byte/word count is handled. It's like this. It was noted that the 4000/5000/6000 never wrote anything with the most significant bit of the word count set. So the 8000 byte count is rotated around right once before being written to tape. This allows interchangeability between the two systems. As for the day someone writes an odd length record on the 8000 and tries to read it on a 4000/5000/6000, well, that will cause all sorts of problems.
This bit is cleared by the next store command.
The secondary status is zeroed out prior to each store which causes an action which might involve reading a byte count. (skip, read, cramed byte count, write data)

Load #30 : RECOVERABLE ERROR STATUS

When a recoverable or unrecoverable error is flagged during a GCR read, skip, or crammed byte count, This status contains information about the error(s).

In general, the data blockettes in the GCR format contain 256 bytes of data and are numbered from

INT [(byte count - 1)/256] to 0.

followed by 2 check blockettes numbered 1, and 0.

If no more than one even numbered blockette and no more than one odd numbered blockette are in error, then the record is recoeverable. Two even or two odd errors makes the record unrecoverable.

This tertiary status has in it information about the first odd blockette error (if one occurred) and the first even blockette error (if one occurred). Here's the format :

```
#AB CD EF GH :
 || || || ||
 || || || ||--THE BLOCKETTE NUMBER (00-FF) OF THE FIRST EVEN BLOCKETTE
 || || ||                                                     ERROR
 || || ||-----0 = NO EVEN BLOCKETTE ERROR
 || || |      1 = THIS EVEN BLOCKETTE ERROR IS IN A DATA BLOCKETTE
 || || |      2 = THIS EVEN BLOCKETTE ERROR IS IN AN XOR BLOCKETTE
 || || |
 || || |------THE ERROR CODE (0-9) FOR THE NATURE OF THIS EVEN BLOCKETTE
 || ||                                                          ERROR
 || ||--------THE BLOCKETTE NUMBER (00-FF) OF THE FIRST ODD BLOCKETTE
 ||                                                           ERROR
 ||-----------0 = NO ODD BLOCKETTE ERROR
 |            1 = THIS ODD BLOCKETTE ERROR IS IN A DATA BLOCKETTE
 |            2 = THIS ODD BLOCKETTE ERROR IS IN AN XOR BLOCKETTE
 |
 |------------THE ERROR CODE (0-9) FOR THE NATURE OF THIS ODD BLOCKETTE
                                                               ERROR
```

The "nature" of the blockette errors is as follows :

0 : This is a forced error because the crammed byte count algorythm assumes the first data blockette is bad.
1 : This blockette's I.D. header was late.
2 : This blockette's I.D. header was late and caused an overflow which will aslo be flagged in primary status bit 17.
3 : This blockette's I.D. header was a legal GCR code (the I.D. is supposed to be an illegal GCR code)
4 : This blockette's I.D. header was not recognizeable.
5 : This blockette's blockette number was an illegal GCR code.
6 : This blockette had a data byte which was illegal GCR code.
7 : The first byte of this blockette's CRC was illegal GCR code.
8 : The second byte of this blockette's CRC was illegal GCR code.
9 : This blockette's CRC did not check.

This status is zeroed out prior to an operation which might record errors in it. (read, crammed byte count, skip forward, and write data, all only when GCR is selected)

Load #40 : STATUS OF THE DRIVES
Standing in front of the CTC, the left most drive is number 0, the next number 1, and so on up to 3. When you request the status of all four drives, bits 0-7 of the response are the status of drive 0, 8-15 are drive 1, 16-23 are drive 2, 24-31 are drive 3. Since the bits in the primary status 24-31 are also the selected drive's status, I'll describe what drive 3's status looks like. The others are, of course, the same, just displaced by a multiple of 8 bits.

bit 24 : EOT. When the tape is moving forward, this bit will be asserted when the physical end of the tape is found. When backing away from the physical end of tape, this bit will go inactive simultaneously with EOTW going inactive when you haved backed 48 inches away from the end of tape.

bit 25 : EOTW. This bit is set if the tape is located within 48 inches of the physical end of the tape.

bit 26 : TAPE INSTALLED. This bit will be set if the selected drive has a cartridge installed in it. If there is no tape installed, all indicators of tape position will be inactive (clear). That's bits 24, 25, 27, and 31. Bit 29 (Tape position not known) will be set.

bit 27 : BOTW. This bit is set if the tape is located within 48 inches of the physical beginning of the tape. That is, located before the load point hole.

bit 28 : WRITE PROTECTED. This bit is active if the installed cartridge is not to be written upon. The CTC will not let you write on it. This bit is also active if no tape is installed.

bit 29 : TAPE POSITION NOT KNOWN. When a cartridge is first installed, this bit will be active cuz its position is not known. Using the rewind, fast forward or cycle commands, you can move the tape till a marking hole is found, and this bit will be cleared. When this bit is active, all tape position bits (24, 25, 27, and 31) are inactive (clear).

bit 30 : SELECTABLE. This bit indicates a drive is present and well. If this bit is not active (set), then all other bits are rendered meaningless. This bit is how to tell how many drives are cabled up to a CTC.

bit 31 : BOT. When the tape is moving backwards, this bit will be asserted when the physical beginning of the tape is found. When advancing from the physical beginning of tape, this bit will go inactive simultaneously with BOTW going inactive when you have advanced 48 inches away from the beginning of the tape.

Load #50 : UNRECOVERABLE ERROR STATUS

There are 3 ways an unrecoverable GCR error can occur. This status reports on how the unrecoverable error came to pass. It was useful during debugging, so I left it in, but I don't expect it to be of any use to the system.

During a write data operation, any error is a FATAL error. If that error happens to be in a data or XOR blockette such that it's flavor can be described as the format for the tertiary status allows, then that error will be recorded here in exactly the same format as it would be in if it were in the tertiary status.

During a read, crammed byte count, or skip forward operation, if the error count gets too high (either 2 odd blockette or two even blockette errors) then that second error is recorded here in the quintary status.

During a read, crammed byte count, or forward skip operation, if a recoverable error occurs and an overflow occurs, then, while the record was recoverable, the system does not have all the data necessary to recunsruct the data so it becomes an unrecoverable error. In this case, the quintary status will be zero and an unrecoverable error will be flagged in the promary status.

This status is zeroed prior to every operation which might put some information here. (read, crammed byte count, skip forward, and write data, all only when GCR is selected)

STORES TO THE CARTRIDGE TAPE CONTROLLER

The CTC receives an 8 byte string over the command status path from the PPU. The first byte is a #96 indicating this is a store. The second byte is the pseudo-address into which to store the following 4 bytes of operand. The seventh byte is a don't care byte and the eighth byte guarantees vertical parity.

A store command is how the 8000 tells the CTC to perform numerous operations. Depending on what the store command is, different bits in the 8 byte string can take on different meanings. I can find no generalization of what all the bits are for, so I'll just document all the things the CTC can do and how to do them.

First, a definition of bits. Here's the received string

```
  #96 AA BB BB BB BB DD CC
                        CC = veritical parity byte
                     DD = don't care byte
         BB BB BB BB = 32 bit operand
      AA = pseudo-address
   96 = this is a store command
```

Any time a store is done to the CTC, the 8000 has the option of instructing the CTC to select a new drive. This information is in the pseudo-address "AA". If we pull it apart :

```
  iaaa sddd
       |\-/
       | |---These three bits callout which drive to select
       |
       |-----This bit indicates "select the drive"
```

If the "s" bit is not set, the three "d" bits will be ignored, and the selected drive will remain unchanged.
If the "s" bit is set, the CTC will look at the "d" bits. If they say to select drive number 4,5,6, or 7, this is an illegal drive because on this controller, those drives do not exist. That error is flagged and the store command is considered finished. If the drive to be selected is 0,1,2, or 3, the CTC will look up its record of what track and density are selected for that drive and adjust ALL its status appropriately. The rest of this store command will be performed on the newly selected drive number, and the status updates at the end will indicate this new drive is selected and will tell what track and density are selected.

Next, the CTC asks the selected (which may be newly selected) drive for its status. It will use this latest status to determine if any requested tape motion is legal or not.

Next, the CTC decodes more of the pseudo-address.

```
iaaa sddd
|\-/
| |---These bits are decoded next
|
|-----This bit indicates an interrupt on completion
      of this store command is requested. An interrupt
      on completion means that when the CTC is done executing
      this store command, or when an error that aborts the
      command is encountered, the CTC will send its secondary
      status to the PPU if it is valid and the CTC will send
      its primary status to the PPU and indicate that the PPU
      should interuupt the 8000 system.
```

On the following pages are the possible ways that the terms "aaa" can be decoded, and what is done in each case. In all cases, the definition of the store command is preceeded by the pseudo-address(es) which denote which store command is to be executed.

```
#0x or #8x : RESET / DO SELF-TEST

#1x or #9x : ILLEGAL

#2x or #Ax : ILLEGAL

#3x or #Bx : NO DATA OPERATION.  Inspect the least significant
                 byte of the operand for which operation.
            #00 : NOP
            #01 : CYCLE
            #02 : REWIND
            #03 : FAST-FORWARD
            #04 : SEARCH FILE MARK REVERSE
            #05 : SEARCH FILE MARK FORWARD
            #06 : SKIP REVERSE
            #07 : SKIP FORWARD
            #08 : DISABLE INTERRUPT ON DRIVE STATE CHANGE
            #09 : ENABLE  INTERRUPT ON DRIVE STATE CHANGE
            #0A : ILLEGAL STORE
            #0B : ILLEGAL STORE
            #0C : SELECT  MFM
            #0D : SELECT  GCR
            #0E : ILLEGAL STORE
            #0F : ILLEGAL STORE
            #10 : SELECT  TRACK 0
            #11 : SELECT  TRACK 1
            #12 : SELECT  TRACK 2
            #13 : SELECT  TRACK 3
            #14 : ILLEGAL TRACK
            #15 : ILLEGAL TRACK
            #16 : ILLEGAL TRACK
            #17 : ILLEGAL TRACK
            #18 : WRITE FILE MARK
     All others : ILLEGAL STORE

#4x or #Cx : READ COMMANDS - Inspect the most significant byte
                 of the operand for which read.
            #00 : READ NEXT RECORD
            #40 : SEND GCR ERROR CORRECTING INFORMATION
            #80 : READ GCR WITH THIS BYTE COUNT (in 2 LS bytes)
     All others : ILLEGAL STORE

#5x or #Dx : WRITE DATA (byte count in 2 LS bytes)

#6x or #Ex : ERASE TAPE (inch count in 2 LS bytes)

#7x or #Fx : DIAGNOSITC WRITE REPEAT (byte count in 2 LS bytes)

             Here's how "x" above is decoded :

                  x < #8 : Selected drive unchanged
                  x = #8 : Select   drive 0
                  x = #9 :   ..       ..   1
                  x = #A :   ..       ..   2
                  x = #B :   ..       ..   3
                  x > #B : Illegal drive - cancell operation
```

IN DETAIL, HERE'S WHAT THE STORES DO :

Store #0x or #8x

This is a command to reset the CTC. The CTC will imitate a power-up sequence, resetting its ports to the PPU and to its drives, perform its self-test and end up in its idle loop awaiting commands from the PPU. This self test may take up to 2 seconds, which means the CTC will not respond to the system again for up to 2 seconds. Note that if an interrupt on completion of this command is requested, by the time the self-test in done that information has been lost so no interrupt will be sent.
The only possible error with this store is error #1, which will set the error bit.

Store #1x or #9x

This is one of the undefined stores and will flag the illegal store error. This store takes virtually no time to execute. Possible errors are #1, and if not, definitely #2. In either case, the error bit will be set.

Store #2x or #Ax

This is one of the undefined stores and will flag the illegal store error. This store takes virtually no time to execute. Possible errors are #1, and if not, definitely #2. In either case, the error bit will be set.

Store #3x or #Bx

This store is of the group that has no data transfers over the data path associated with it. There are many such commands. The least significant 8 bits of the operand are used to specify what operation is intended. Here thay are, listed according to what's in the least significant 8 bits. What is in the other bits is of no concern and is never checked.

#00 : NOP - Nothing is done.
Possible errors are #1, which will set the error bit.
This store takes virtually no time.

#01 : CYCLE - The tape is advanced to EOT and then rewound to BOT. If the tape position is not known prior to execution of this command, the tape will first rewind 24 inches to look for a marking hole before advancing to EOT. If the tape is already at EOT, it will only rewind to BOT. This commands is useful to fix a tape whose rubber-band in the drive mechanism is out of whack. The symptom of this out of whack is excessive drop-outs, although this is not the only problem that can cause excessive drop-outs. Normally, this command will take about 1.5 minutes to work. Under totally healthy conditions, it may take up to 7 minutes. If it has not completed in 10 minutes, the timeout error will be flagged. In order to not get an illegal tape motion error, the drive must be selectable with a tape installed. possible errors are #1,4,6, or #7, all which will set the error bit. If no errors are detected, the tape position will be known at the end of this operation.

#02 : REWIND - The tape is rewound to BOT. If the tape position is not known prior to execution of this command, the tape will first advance 24 inches to look for a marking hole. If the tape is already at BOT, nothing will happen. This command will not take more than 5 minutes to complete. In order to not get an illegal tape motion error, the drive must be selectable with a tape installed. Possible errors are #1,4,6,or #7, all of which will set the error bit.
If no errors are detected, the tape position will be known at the end of this operation.

#03 : FAST FORWARD - The tape is advanced to EOT. If the tape position is not known prior to execution of this command, the tape will first rewind 24 inches to look for a marking hole. If the tape is already at EOT, nothing will happen. This command will not take more than 5 minutes to complete. In order to not get an illegal tape motion error, the drive must be selectable with a tape installed. Possible errors are #1,4,6,or #7, all of which will set the error bit.
If no errors are detected, the tape position will be known at the end of this operation.

#04 : SEARCH FILE MARK REVERSE - This command will move the tape backwards and stop in the I.R.G. preceeding the first file mark that it finds. This command will not take more than 5 minutes to complete. In order to not get an illegal tape motion error, the drive must be selectable with a tape installed with the tape position known and not at BOT. If a record that cannot be recognized is found, the tape will stop in the I.R.G. preceeding that record and report the error. Possible errors are #1,4,7,9,C,D, or #18.
The error bit will be set on errors #1,4,9,C, or #D. The error bit may or may not be set on error #7 or #18,

#05 : SEARCH FILE MARK FORWARD - This command will move the tape forwards and stop in the I.R.G. following the first file mark that it finds. This command will not take more than 5 minutes to complete. In order to not get an illegal tape motion error, the drive must be selectable with a tape installed with the tape postion known and not at EOT. If a record that cannot be recognized is found, the tape will stop in the I.R.G. following that record and report the error. Possible errors are #1,4,7,8,9,C,D,11,12, or #17.
They all set the error bit except #11,12,7, and #17.
#11 and #12 do not set the error bit.
#7 and #17 may or may not set the error bit.

#06 : SKIP REVERSE - This command will move the tape backwards, try to recognize the first record it finds and stop in the gap before that record. This command will not take more than 25 seconds to complete. In order to not get an illegal tape motion error, the drive must be selectable with a tape installed with the tape position known and not at BOT.
Possible tape errors are #1,4,7,9,C,D, or #18.
They all set the error bit except #C,D,7, and #17.
#C and #D do not set the error bit.
#7 and #18 may or may not set the error bit.

#07 : SKIP FORWARD - This command will move the tape forward and try to recognize the first record it finds. If it is a data record, it will read the byte count if it can, and then it will stop in the I.R.G. following that record. This command will not take more than 30 seconds to complete. In order to not get an illegal tape motion error, the drive must be selectable with a tape installed with the tape position known and not at EOT. Possible tape errors are : #1,4,7,8,9,A,C,D,E,F,10,11,12,13,15,16, or #17. Only error #1,4,8, and #9 will set the error bit. #7 and #17 may or may not set the error bit.

#08 : DISABLE INTERRUPT ON DRIVE CHANGE STATE - It does just that. It takes virtually no time, and the only possible error is error #1 which will set the error bit.

#09 : ENABLE INTERRUPT ON DRIVE CHANGE STATE - It does just that. It takes virtually no time, and the only possible error is error #1 which will set the error bit.
Note that if you turn on this interrupt while requesting an interrupt on completion, when that interrupt on completion is sent, this interrupt enable will be cleared, rendering the whole operation useless.
If primary status bit 0 (drive(s) changed state) is set when this command sets bit 1, an interrupt on drive change state will be sent immediately.

#0A : ILLEGAL STORE - This is one of the undefined stores and will flag the illegal store error. This store takes virtually no time to execute. Possible errors are #1, and if not, definitely #2, both which set the error bit.

#0B : ILLEGAL STORE - This is one of the undefined stores and will flag the illegal store error. This store takes virtually no time to execute. Possible errors are #1, and if not, definitely #2, both which set the error bit.

#0C : SELECT MFM - This tells the CTC that the currently selected drive will read and write the MFM recording density. This store takes virtually no time to execute. Possible errors are #1, which sets the error bit.

#0D : SELECT GCR - This tells the CTC that the currently selected drive will read and write the GCR recording density. This store takes virtually no time to execute. Possible errors are #1, which sets the error bit.

#0E : ILLEGAL STORE - This is one of the undefined stores and will flag the illegal store error. This store takes virtually no time to execute. Possible errors are #1, and if not, definitely #2, both which set the error bit.

#0F : ILLEGAL STORE - This is one of the undefined stores and will flag the illegal store error. This store takes virtually no time to execute. Possible errors are #1, and if not, definitely #2, both which set the error bit.

#10 : SELECT TRACK 0 - Track number 0 will be the track that reading and writing will be done to for the currently selected drive. This store takes virtually no time to execute. Possible errors are #1, which sets the error bit. If you change tracks when not at an end of the tape as indicated by the drive status, you may no longer be in a gap so a read is likely to fail and a write must not be done until proper position in a gap is aquired.

#11 : SELECT TRACK 1 - Track number 1 will be the track that reading and writing will be done to for the currently selected drive. This store takes virtually no time to execute. Possible errors are #1, which sets the error bit. If you change tracks when not at an end of the tape as indicated by the drive status, you may no longer be in a gap so a read is likely to fail and a write must not be done until proper position in a gap is aquired.

#12 : SELECT TRACK 2 - Track number 2 will be the track that reading and writing will be done to for the currently selected drive. This store takes virtually no time to execute. Possible errors are #1, which sets the error bit. If you change tracks when not at an end of the tape as indicated by the drive status, you may no longer be in a gap so a read is likely to fail and a write must not be done until proper position in a gap is aquired.

#13 : SELECT TRACK 3 - Track number 3 will be the track that reading and writing will be done to for the currently selected drive. This store takes virtually no time to execute. Possible errors are #1, which sets the error bit. If you change tracks when not at an end of the tape as indicated by the drive status, you may no longer be in a gap so a read is likely to fail and a write must not be done until proper position in a gap is aquired.

#14 : SELECT TRACK 4 - Track number 4 does not exist on this controller so you will get the illegal track error. This store takes virtually no time to execute. Possible errors are #1, and if not, definitely #3. The error bit will be set.

#15 : SELECT TRACK 5 - Track number 5 does not exist on this controller so you will get the illegal track error. This store takes virtually no time to execute. Possible errors are #1, and if not, definitely #3. The error bit will be set.

#16 : SELECT TRACK 6 - Track number 6 does not exist on this controller so you will get the illegal track error. This store takes virtually no time to execute. Possible errors are #1, and if not, definitely #3. The error bit will be set.

#17 : SELECT TRACK 7 - Track number 7 does not exist on this controller so you will get the illegal track error. This store takes virtually no time to execute. Possible errors are #1, and if not, definitely #3. The error bit will be set.

#18 : WRITE FILE MARK - It does just that - writes a file mark on the selected track of the selected drive. Remember that you must be properly positioned in an I.R.G. to do this, otherwise, you may leave garbage on the tape. In order to not get an illegal tape motion error, the drive must be selectable, a tape installed, tape position known, not at EOT, and not write protected. If the tape is positioned before the load point, the CTC will erase tape up to 1 inch past the load point and then write the file mark. If PFW is active just before the tape motion is started, the CTC will not begin the write and will hang up waiting for a hard reset from the PPU or a power cycle. This store will not take more than 4 seconds to complete. Possible errors are :
#1,4,7,8,9,B,C,D,11,12, or #17.
Any one of these errors will set the error bit.

All others : An illegal store error will be flagged. This store will take virtually no time to execute. Possible errors are #1, and if not, definitely #2, and the error bit will be set.

Store #4x or #Cx : READ DATA - actually, there are three ways to read data from the CTC. The most significant byte of the 32 bit operand (bits 0-7) determine which of these three reads is requested.

#00 : READ RECORD - This is the normal read command. The CTC will move forward and attempt to read the next record on the tape. If any FATAL errors occur, the read will be aborted and the CTC will space forward to the I.R.G. following the bad record and stop improperly positioned in it. If no FATAL errors occur, the CTC will stop properly positioned in the next gap. If the record was a file mark, bit 8 in the primary status will so indicate. If the record was a non-zero length record, the data bytes will have been sent to the PPU over the data path unless there was an overflow error flagged in bit 17 of the primary status. In this case, the error bit will be set. In all cases, when the read or attempted read is finished, the CTC will attempt to send an End-of-Record byte to the PPU over the command status path, although the CTC will not care if the PPU accepts it or not. In order to not get an illegal tape motion error code, the drive must be selectable with a tape installed whose position is known and not at EOT.
This store will not take more than 30 seconds to complete.
Possible errors are :
#1,4,7,8,9,A,C,D,E,F,10,11,12,13,15,16, or #17.

#1,4,8,9,A,C,D,E,F,13, and #16 will set the error bit

#10,11, and #12 will not set the error bit.

#7 and #17 may or may not set the error bit.

#15 will set the error bit unless the recoverable error is confined to the check blockettes in which case all the data was successfully recovered so the error bit is not set.

#40 : SEND GCR XOR BYTES - This one sends the information necessary to reconstruct the data after a recoverable read error in GCR mode. This sends 512 bytes to the PPU over the data path, followed by an EOR bte over the command-status path.

The data written onto the tape is divided into blockettes of 256 bytes each. The first byte written is in blockette number "INT [(byte count - 1)/256]". The last byte written is in blockette number "0". If the record is not an even multiple of 256 bytes, the last blockette (number 0) will be filled out to 256 bytes with zeroes. All the odd numbered blockettes are XORed together and the result stored in XOR blockette number 1 in such a way that if all the data was zeroes, the XOR blockette will be all zeroes. Same with the even blockettes, except that result is stored in XOR blockette 0.

When you request the XOR blockettes be sent to the system, you will get 512 bytes : XOR blockette 1 followed by blockette 0. The first XOR byte you get will be the XOR of the first byte written in the first odd numbered blockette and all subsequent bytes which occur at 512 byte intervals. Depending on the lengh of the record, that may be bytes 1,513,1024..., or it may be bytes 257,769,1281... The 257 byte you get will check the other of the two sets of bytes above.

Remember that the last blockette is filled with zeroes on the tape and that information is included in the XOR blockettes. The zeroes in the data field on the tape are never seen by the system, but the XOR information will have those zeroes in it. For instance, if you write a record that is one byte long, the XOR blockettes will be 512 bytes long, only the 257th of which has any information in it.

This command will take not more than 25 mili-seconds to execute. If all the data is not taken by the PPU, then an overflow will be flagged in primary status bit 17 and the error bit will be set. The only possible FATAL error is #1 if you attempted to select an illegal drive, which also will set the error bit and inhibit the sending of the XOR bytes.

The XOR bytes are zeroed out prior to an operation which is likely to change them. (read, crammed byte count, skip forward, and write data, all only when GCR is is selected)

#80 : READ THIS BYTE COUNT - This one works a lot like a normal READ command, with a few significant diffences. It is only useful to attemt to read a GCR data record. If you use it to attemt to read anything else, you'll get error #14, "GCR crammed byte count cannot be found". This command is used to recover a GCR data record which has an error in the very beginning which results in the byte count not being recoverable. Those errors which this command facilitates recovery from are error #C, D, and #E. All of these erros indicate the record is findable but the byte count cannot be read. In these cases, the CTC will space to the end of the record and flag the error. The system driver must instruct the CTC to do a skip reverse. Assuming the reverse byte count is recoverable, The CTC will provide the system with the byte count of this record after completing the skip reverse. If the reverse byte count is bad, then two errors have occurred in one record, and it is not recoverable. But assuming the reverse byte count is good, here's what happens next.

The system must tell the CTC to do a crammed byte count read. It does this by storing the following command in pseudo-address #40 or #C0 to request an interrupt on completion : #80 00 XX XX

where that "XX XX" is the same byte count which the CTC just read during the skip reverse. What you have done is told the CTC to read this record substituting this byte count for the one that is at the beginning of the record, whether that leading byte count is readable or not.

The CTC examines this byte count and figures out what the number on the first blockette should be. It assumes the first blockette is unreadable, flags error #0, a data error, and the appropriate blockette number and loads this into the tertiary satus. Thus a crammed byte count read will always, at best, contain one error which is recoverable and will be error #15, unless other errors bump it up to unrecoverable error #16. The number of bytes which should have been recovered from that first blockette will be sent to the PPU. If that number is 256, those 256 bytes will be page 3 of the CTC's RAM, which is where most of the CTC's status is in case you ever want to look at it for trouble-shooting purposes. Ram addresses #300, 301, 302...#3FF are sent. You'll have to look at the micro-code for what these status'es are.

Next, the CTC starts the tape and looks for the second blockette in the record. If it cannot be found, for whatever reason, error #14 (crammed byte count cannot be found) is flagged. Otherwise, when it is found, the CTC just reads the record from there on as if nothing were wrong.

#80 : READ THIS BYTE COUNT - (continiued)

At the end of the read, during the status update, bit 21 in the priamry status will be set, as well as bit 22. Bit 22 says the byte count in the secondary status is valid and bit 21 reminds you that it is valid because it is the same one you sent the CTC.

With the exception of bit 22 and the posibility of error #14, a crammed byte count read will look to the system exactly like a normal GCR read which, at best, has one recoverable bad blockette and, at worst, is unrecoverable. With a little cleverness in the system driver, this feature can be used to read a record from any point in the middle to the end, should anyone ever want to do that. A normal read will read from the beginning to the 3rd error. Using these facts, and examining the primary, tertiary and quintary status, one can recover most of a record that has enough of the middle of it destroyed that the whole record is simply not recoverable.
In all cases, when this crammed byte count read is finished, the CTC will attempt to send an End-of-Record byte to the PPU over the command status path, although the CTC will not care if the PPU accepts it or not. In order to not get an illegal tape motion error code, the drive must be selectable with a tape installed whose position is known and not at EOT.
A crammed byte count of #00 00 will look like a space forward with error #13 (blockette cannot be found).
In order to not get an illegal store error (#2), the GCR density must be selceted for this drive.
This store will not take more than 30 seconds to complete.
Possible errors are :
#1,4,7,8,9,A,13,14,15,16, or #17.
One of the above errors will occur, and the error bit will be set.

All others : An illegal store error will be flagged. This store will will take virtually no time to execute. Rossible errors are #1, and if not, definitely #2, both which will set the error bit.

Store #5x or #Dx : WRITE DATA. This store is how data records are actually written to the tape. The 32 bit operand written to location #5x or #Dx to request an interrupt on completion has lots of interesting information in it : #AA BB CC DD

#AA : not used and not looked at by the CTC.
#BB : the CTC will fetch this many bytes from the PPU before starting the tape or beginning to write. The CTC throws away these data bytes. This is a patch to fix a system design flaw that data buffers may begin on byte boundaries in memory but data transfers to the CTC must begin on word (32 bits) boundaries in memory.
#CC & DD are the 16 bit count of how many bytes are to be written in this data record. Note if this is zero, the record will be written with all the headers and trailers, there will just be no data, as well as no data check characters or error correcting information.

In order to not get a tape motion error (#4), the selected drive must be selecteable with a tape installed whose position is known and is not EOT, and the tape must not be write protected. The CTC will send a GO byte to the PPU, and if the PPU does not accept it, the write will be aborted, an EOR byte will be sent, and an interrupt on completion if requested. If the GO byte is accepted, the CTC will fetch and throw away as many bytes as it is told to, and will check PFW. If active, the CTC will hang, waiting for a power cycle or a hard reset from the PPU. If PFW is inactive, the CTC will begin the write. If the tape is positioned before load point, the CTC will time the arrival of load point, flagging an error and aborting the operation if it is late (also flagging tape position lost). The CTC will erase all tape prior to load point and one inch beyond load point and then begin the data record. If any FATAL erros are detected during the read-after-write, the operation will be aborted and a backspace is mandatory, followed by either a re-try or an erase.

The CTC has been fetching bytes (unless byte count = 0) from the PPU all along. If the PPU does not deliver the bytes fast enough, the CTC will flag an underflow error in the primary status bit 18 and write zeroes to fill the record, fetching no more bytes from the PPU. If a parity error is detected, the CTC will write the byte with bad parity and thereafter, fill the record with zeroes and fetch no more bytes from the PPU. It will set bit 16 in the primary status to flag the parity error. Both underflow and parity error will set the error bit. Otherwise, the write will continue along as before. When the record is complete, the CTC will send the EOR byte and interrupt on completion if requested. This command will not take more than 30 seconds to complete. Possible errors are :
#1,4,5,7,8,9,B,C,D,E,F,10,11,12,13,14, or #17.
Any one of these errors will set the error bit.

Store #6x or #Ex : ERASE TAPE. This is an explicit command to erase tape. The 32 bit operand which is stored into loaction #6x or #Ex to request an interrupt on completion is how many inches of tape to erase, or a flag to erase to the end of the tape. The operand is #xx xx EE EE. The CTC does not look at the "xx" bytes.

If the most significant bit of the "#EE EE" is set, the CTC will erase from where it is currently positioned on the tape to the End-Of-Tape. The load point is ignored. In this case, possible errors are :
#1,4,6, and #7, all of which will set the error bit.

If the most significant bit of "#EE EE" is clear, then the CTC uses #EE EE as an inch count of how many inches to erase. If the tape is positioned before the load point, the CTC will erase untill it is one inch past the load pont, and then begin counting out the inches of tape to erase. Thus the count of inches to erase begins at the same point as a write begins. If the tape is positioned past the load point, then the CTC will begin counting out the inches from where it is. If zero inches of tape are requested to be erased, no tape motion or erasure will result. Possible errors are #1,4,7,8, and #17, all of which will set the error bit.

This operation will not take more than 5 minutes to complete. In order to not get a tape motion error (#4), the selected drive must be selecteable with a tape installed whose position is known and is not EOT, and the tape must not be write protected.
As with the writes, you must never do an erase unless you are properly positioned in an I.R.G.

Store #7x or #Fx : WRITE REPEAT : This store is strictly for diagnostic purposes. The 8000 driver should NEVER send this to a CTC . It is exactly identical to a write data in the beginning. When the write is completed and it is time to stop the tape, though, the CTC will write a 2/3 inch I.R.G. if no FATAL errors are detected, or a 1/3 inch gap if a FATAL error is detected. Then it will write the same record over again, and again, and again forever. Even EOT or offline do not stop this. The only things that will stop this are a power cycle, a hard reset from the PPU, or the PPU sending a new command string to the CTC over the command status path. This way the Technician can get the write and read logic in the CTC periodically repeating the complete read and write cycles so that an oscilloscope probe can look around to see what's going on. When EOT is reached, the Tech. can send a rewind command to the CTC, which it will do and so on. Actually, when a new command comes from the PPU, or in this case the PPU emulater, the CTC will write one more complete data record before acknowledging the new command.

The only way to get the priamry status resulting from this write repeat is to abort the write repeat with a load of the primary status. It will have the error bit set, "tape position lost" and FATAL error #1F flagged.

DATA TRANSFER RATE. The CTC will accept data bytes at a rate of up to one byte every 1.6 micro-seconds. Typically, though, it will accept data bytes at a rate of one every 32 micro-seconds in GCR mode, one every 41 micro-seconds in MFM mode.
The CTC will send bytes at a rate up to one byte every 1.6 micro-seconds, but typically the transfer rate will be about the same as above (32 and 41 micro-seconds). Under and over flow errors occur the when transfer rate falls much below the typical values given above.

Under and over flow errors are also flagged when the first few bytes to be sent/received are late. In these cases, the CTC allows about 10 milli-seconds for the intitail transfers to complete. These initail transfers are :

write : timed from the reciept of the go byte by the PPU to the reciept of the last of those first few bytes to be thrown away to fix the system flaw that data buffers may begin on byte boundries in memory but data transfers must begin on word boundries.

read : timed from the reciept of the read command by the CTC to the reciept of the first 256 bytes by the PPU.

These are the fastest cases. There are times when the CTC won't care about byte transfers for up to 10 seconds.

MAXIMUM TIME FOR TAPE MOTIONS : For the skip and read commands, I have said the maximium time to complete the command is 25 or 30 seconds. That assumes things in the drive work properly and the tape has an inter-record-gap every once in a while. If not, the time to completion could be forever. This also applies to the search commands, for which 5 minutes is specified. For search/skip/reads, if the drive is working properly and the tape is 600 feet long, the maximum time will be about 5 minutes.
Fact is, while the CTC may still have things under control for up to 10 minutes (during a cycle command), ANYTHING that takes more than 7 minutes has somethng wrong with it.

LOADS BRIEFLY

```
#1x or #9x : PRIMARY STATUS, listed by bit (0=most significant bit)

        0 : drive(s) changed state since I last interrupted
        1 : interrupt enable on drive change state
        2 : error detected, re-try required.
      3-5 : selected track number
      6-7 : selected drive number

        8 : file mark found
     9-10 : motion code :       00 - in same gap
                                01 - moved forward out of gap
                                10 - moved backwards out of gap
                                11 - lost
    11-15 : FATAL error codes, shown with possible motion codes
            #00 : 00,01,10  00000 - no error detected
            #01 :    00     00001 - illegal drive
            #02 :    00     00010 - illegal store
            #03 :    00     00011 - illegal track
            #04 :    00     00100 - illegal tape motion
            #05 :    00     00101 - no GO byte
            #06 :    11     00110 - timeout waiting for an end of tape
            #07 :    11     00111 - offline (tape no longer installed)
            #08 :    11     01000 - laod point hole late
            #09 :    11     01001 - sync zone late
            #0A :    01     01010 - unexpected I.R.G. found
            #0B :    01     01011 - drop-out during a write
            #0C :  01,10    01100 - I.D. zone late
            #0D :  01,10    01101 - I.D. zone bad
            #0E :  01,10    01110 - byte count check bad
            #0F :    01     01111 - MFM data CRC bad
            #10 :    01     10000 - GCR reverse byte count bad
            #11 :    01     10001 - trailing I.D. zone bad
            #12 :    01     10010 - trailing sync zone bad
            #13 :    01     10011 - GCR blockette number wrong
            #14 :    01     10100 - GCR crammed byte count not found
            #15 :    01     10101 - GCR read bad but recoverable
            #16 :    01     10110 - GCR read unrecoverable
            #17 :    11     10111 - unexpected EOT found
            #18 :    11     11000 - unexpected BOT found
            #1F :    11     11111 - last operation was a write repeat

       16 : parity error on data coming from PPU
       17 : overflow error. All data not successfully sent to PPU
       18 : underflow error. All data not successfully sent to CTC
       19 : WDL error - CTC's write logic goofed up
       20 : RDL error - CTC's read logic goofed up
       21 : last operation was a crammed byte count read
       22 : byte count in secondary status is valid
       23 : GCR densiy (1) / MFM density (0)

       24 : EOT
       25 : EOT Warning
       26 : tape installed
       27 : BOT Warning
       28 : write protected
       29 : tape position NOT known
       30 : drive selectable
       31 : BOT
```

STORES IN BRIEF

```
#0x or #8x : RESET / DO SELF-TEST

#1x or #9x : ILLEGAL

#2x or #Ax : ILLEGAL

#3x or #Bx : NO DATA OPERATION.  Inspect the least significant
                  byte of the operand for which operation.
             #00 : NOP
             #01 : CYCLE
             #02 : REWIND
             #03 : FAST-FORWARD
             #04 : SEARCH FILE MARK REVERSE
             #05 : SEARCH FILE MARK FORWARD
             #06 : SKIP REVERSE
             #07 : SKIP FORWARD
             #08 : DISABLE INTERRUPT ON DRIVE STATE CHANGE
             #09 : ENABLE  INTERRUPT ON DRIVE STATE CHANGE
             #0A : ILLEGAL STORE
             #0B : ILLEGAL STORE
             #0C : SELECT  MFM
             #0D : SELECT  GCR
             #0E : ILLEGAL STORE
             #0F : ILLEGAL STORE
             #10 : SELECT  TRACK 0
             #11 : SELECT  TRACK 1
             #12 : SELECT  TRACK 2
             #13 : SELECT  TRACK 3
             #14 : ILLEGAL TRACK
             #15 : ILLEGAL TRACK
             #16 : ILLEGAL TRACK
             #17 : ILLEGAL TRACK
             #18 : WRITE FILE MARK
      All others : ILLEGAL STORE

#4x or #Cx : READ COMMANDS - Inspect the most significant byte
                  of the operand for which read.
             #00 : READ NEXT RECORD
             #40 : SEND GCR ERROR CORRECTING INFORMATION
             #80 : READ GCR WITH THIS BYTE COUNT (in 2 LS bytes)
      All others : ILLEGAL STORE

#5x or #Dx : WRITE DATA (byte count in 2 LS bytes)

#6x or #Ex : ERASE TAPE (inch count in 2 LS bytes)

#7x or #Fx : DIAGNOSITC WRITE REPEAT (byte count in 2 LS bytes)

             Here's how "x" above is decoded :

                x < #8 : Selected drive unchanged
                x = #8 : Select   drive 0
                x = #9 :   ..       ..   1
                x = #A :   ..       ..   2
                x = #B :   ..       ..   3
                x > #B : Illegal drive - cancell operation
```