HESSI GSE SOFTWARE

HSI_IDPU_018A

1. Introduction

Delivered with the HESSI GSE hardware are two software applications that support the hardware. One application (HESSI_GSE) is used both to directly test the parallel interface to the GSE and also to transmit test files to the GSE. In addition this application converts files from the "dat" format to the "hgf" format. The other application (HESSI Test File Maker) creates files to be used in testing the GSE hardware in the HESSI GSE format (hgf). The following is a description of this software, it's operation, and file formats.

2. "dat" format

The "dat" format files (indicated by a ".dat" extension to the file name) are formatted as follows:

The first record contains 1 longword (4 bytes) giving the number of succeeding records.

Each successive record contains:

Time	unsigned longword (4 bytes)	0-4194303	binary microseconds
Channel	unsigned byte	0-17	PHA input ID
Amplitude	unsigned word (2 bytes)	0-8191	(for Chan.s 0-8)
		0-50000	(" " 9-17)

3. "hgf" format

The "hgf" format files (indicated by a ".hgf" extension to the file name) are formatted as follows:

All records are 6 bytes long. All records except the last are as follows:

Channel	unsigned byte	0-17	PHA input ID
Time	unsigned (3 bytes)	0-4194303	binary microseconds
Amplitude	unsigned word (2 bytes)	1-65535	D/A input

Note: The amplitude is always odd – the lsb is a data present flag

The last record is as follows:

Channel	unsigned byte	85 Indicates checksum rec.
Checksum	unsigned word (2 bytes)	= sum of all bytes transmitted between byte 1 of record 1 and byte 6 of last record before this one, truncated to 16 bits.
Pad	3 bytes	0

4. HESSI_GSE Application

The HESSI_GSE (HessiGSE.exe) application serves two purposes, first to test the operation of the parallel I/F on the PC used for data transmission to the GSE hardware and then to transmit files to the GSE hardware to test the HESSI instrument operation.

4.1 Parallel I/F test

After starting the HessiGSE.exe application a window appears with two menu selections, File and Manual. Clicking with the mouse on Manual puts the program in parallel I/F test mode.

There are 3 indicator panels, Status, Control and Data. These panels indicate the condition of the corresponding signals on the parallel I/F connector. The status signals are input to the parallel I/F and cannot be changed by the user. The user can change the control signals by toggling the box corresponding to the signal to be changed. The user can change the data outputs either by toggling the individual bits or by entering a hex value. In addition a single byte transfer of the data currently on display can be initiated by hitting the handshake button with the mouse.

The Parallel I/F test mode is exit by either clicking File/Exit, which exits the application, or clicking File/Open putting the application in file transfer mode.

4.2 File Transfer

After starting the HessiGSE.exe application a window appears with two menu selections, File and Manual. Clicking with the mouse on File/Open puts the program in file transfer mode, and a file selection window will appear.

If a file with the 'dat' extension is selected the file will be converted to the 'hgf' format as described in 4.3 (below) and then transfer of the 'hgf' file will start.

If a file with the 'hgf' extension is selected the file transfer will start immediately, starting with initialization of the GSE hardware, (approx. 45 seconds). The disappearance of the file selection window and the appearance of the name of the file selected and the words "Initializing HESSI GSE" on the status bar will indicate this. If initialization is not finished after two minutes have elapsed, the words "GSE Initialization Time Out" will appear on the status bar and a message box will appear asking if the user wishes to try again or quit.

Once initialization has finished and transfer has begun this will be replaced by the word "Transferring". During transfer the progress bar will indicate how much of the file has been transferred.

After transfer has completed, the GSE hardware will compare the checksum calculated against the checksum received. If there is a timeout during checksum calculation, the words "GSE Checksum Time Out" will appear on the status bar and a message box will appear asking if the user wishes to try again or quit. . If there is a transmission error, the words "Bad Checksum" will appear on the status bar and a message box will appear asking if the user wishes to try again or quit. If there is no transmission error the words "Transfer Completed" will appear and the GSE hardware will send the stimulus to the instrument, repeating every four seconds until a new file transfer is initiated.

The software listing that follows is of the mnuFileOpen_Click routine that is called when File/Open is clicked by the mouse. After initializing variables, a file dialog box is opened to get a new file name from the user. Upon return the file name is check to see if the file is hgf or dat. If it is dat, fblnFileConvert is called to create the corresponding hgf file. Then the GSE hardware is sent an initialize command to clear memory using the "init" signal. The hardware signals receipt of this command by asserting the "busy" signal and removing it when the initialization is finished, a check is made to insure that a timeout does not occur and the system is given a chance to

catch up with external events. The file is then transferred 10000 bytes at a time, (except for the remainder at the end). After the transfer is complete the program waits for the hardware to do a checksum comparison before indicating successful completion. As before, a check is made to insure that a timeout does not occur and the system is given a chance to catch up with external events.

```
Private Sub mnuFileOpen_Click()

    Dim strInRec As String, LastGaugeVal As Long, NewGaugeVal As Long
    Dim ILoop As Long, LastByte As Long, bytOutput As Byte
    Dim strMessage1 As String, lngIOResult As Long, lngLOF1 As Long
    Dim lngFileBytePointer As Long, intMsgResult As Integer, lngBytesLeft As Long
    Dim lngLoopCount As Long, vntStartTime As Variant
    strMessage1 = "Input file length corrupt" & Chr(10) & "File Transfer Terminated"
    strInRec = String(10000, " ")       'make strInRec 10,000 bytes long so binary Get's get 10,000 bytes
    With frmFileID.cdlFIleID
       SSFrame1(0).Top = 8000
       SSFrame1(3).Top = 480
       Me.Height = 2340
       SSFrame1(3).Caption = "File Transfer - IDLE"
       ProgressBar1.Value = 0              'initialize progress status bar
       Me.Refresh
       .FileName = ""
       .ShowOpen                    'open file open dialog box to get file name
       Me.Refresh                   'redraw start up form
       If .FileName = "" Then Exit Sub 'If user hit cancel then quit
       gbytControlOut = 0             'assert "init" to signal new image coming (all other  cntls high at connector)
       Call COUpdate
       mstrFileName = .FileName
       mstrFileTitle = .FileTitle
       If Right$(mstrFileName, Len(mstrFileName) - InStr(1, mstrFileName, ".")) = "dat" Then   'User entered name of "Sc
          mstrFileName = Left$(mstrFileName, InStr(1, mstrFileName, ".")) & "hgf"
          mstrFileTitle = Left$(mstrFileTitle, InStr(1, mstrFileTitle, ".")) & "hgf"
          If Not fblnFileConvert Then Exit Sub                                    ' try to convert file, if invalid then quit
       End If
Retry:  Open mstrFileName For Binary Access Read Shared As #1
       lngLOF1 = LOF(1)
       If (lngLOF1 < = 9) Or ((lngLOF1 Mod 6) < > 0) Then                          'Check for valid input file length
          Close #1
          SSFrame1(3).Caption = mstrFileTitle & " Transfer Aborted"
          intMsgResult = MsgBox(strMessage1, vbExclamation)
          Exit Sub
       End If '(lngLOF1 < = 9) Or (((lngLOF1 - 3) Mod 6) < > 0)
       mnuManual.Enabled = False
       Timer1.Interval = 0        'Turn timer off during file transfer
       gbytControlOut = &H4       'Turn all controls high for faster pullup on status inputs
       Call COUpdate
       SSFrame1(3).Caption = "Initializing HESSI GSE"
       ProgressBar1.Value = 0              'initialize progress status bar
```

```
ProgressBar1.Max = 101
Me.Refresh
gbytControlOut = gbytControlOut And &HB                    'Assert Init
Call COUpdate
vntStartTime = Time
Do Until (Not (IO1.In(PAR_STAT)) And &H80)                 'Wait for Busy to be asserted
   DoEvents                                                'Allow OS to process other requests while looping
   If CDbl(Time - vntStartTime) * 86400# > 60# Then Exit Do         'Check for time out
Loop 'Until (Not (IO1.In(PAR_STAT)) And &H80)
If (IO1.In(PAR_STAT) And &H80) Then                        'See if we timed out
   SSFrame1(3).Caption = mstrFileTitle & " Initialization Time Out "       ' yes, deal with it
   Me.Refresh
   strMessage1 = "GSE Initilization Time Out" & Chr(10) & "Do you want to try again?"
   intMsgResult = MsgBox(strMessage1, vbRetryCancel)
   Close #1
   If intMsgResult = vbRetry Then GoTo Retry
   Exit Sub       'otherwise quit
End If '(IO1.In(PAR_STAT) And &H80)
gbytControlOut = gbytControlOut Or &H4                     'Remove Init
Call COUpdate
vntStartTime = Time
Do Until (IO1.In(PAR_STAT) And &H80)                       'Wait for Busy to be removed
   DoEvents                                                'Allow OS to process other requests while looping
   If CDbl(Time - vntStartTime) * 86400# > 60# Then Exit Do         'Check for time out
Loop 'Until (IO1.In(PAR_STAT) And &H80)
If (Not (IO1.In(PAR_STAT)) And &H80) Then                  'See if we timed out
   SSFrame1(3).Caption = mstrFileTitle & " Initialization Time Out "       ' yes, deal with it
   Me.Refresh
   strMessage1 = "GSE Initilization Time Out" & Chr(10) & "Do you want to try again?"
   intMsgResult = MsgBox(strMessage1, vbRetryCancel)
   Close #1
   If intMsgResult = vbRetry Then GoTo Retry
   Exit Sub       'otherwise quit
End If 'Not (Not (IO1.In(PAR_STAT)) And &H80)
SSFrame1(3).Caption = mstrFileTitle & " Transfering"
Me.Refresh
lngFileBytePointer = 1
Debug.Print lngLOF1
Debug.Print Now
Do While lngFileBytePointer < = (lngLOF1 - 10000)          'Output file 10000 bytes at a time
   Get #1, lngFileBytePointer, strInRec
   lngFileBytePointer = lngFileBytePointer + 10000
   ProgressBar1.Value = CInt((CSng(lngFileBytePointer) / CSng(lngLOF1)) * 100!)
```

```
        lngIOResult = IO1.WriteData(strInRec, 10000)
      Loop 'While lngFileBytePointer < = (lngLOF1 - 10000)
      Get #1, lngFileBytePointer, strInRec                    'Output remainder of file
      lngBytesLeft = lngLOF1 - lngFileBytePointer + 1
      ProgressBar1.Value = 101
      lngIOResult = IO1.WriteData(strInRec, lngBytesLeft)
      Debug.Print Now
      Close #1
      SSFrame1(3).Caption = mstrFileTitle & " Computing Checksum"
      Me.Refresh                                  'wait for checksum result
      vntStartTime = Time
      Do Until (Not (IO1.In(PAR_STAT)) And &H8) Or (Not (IO1.In(PAR_STAT)) And &H10)
        DoEvents                                  'Allow OS to process other requests while looping
        If CDbl(Time - vntStartTime) * 86400# > 5# Then Exit Do        'Check for time out
      Loop 'Until (Not (IO1.In(PAR_STAT)) And &H8) Or (Not (IO1.In(PAR_STAT)) And &H10)
      If (IO1.In(PAR_STAT) And &H18) = &H18 Then              'See if we timed out
        SSFrame1(3).Caption = mstrFileTitle & " Checksum Time Out "       ' yes, deal with it
        Me.Refresh
        strMessage1 = "GSE Checksum Time Out" & Chr(10) & "Do you want to try again?"
        intMsgResult = MsgBox(strMessage1, vbRetryCancel)
        If intMsgResult = vbRetry Then GoTo Retry
        Exit Sub      'otherwise quit
      End If 'Not (Not (IO1.In(PAR_STAT)) And &H80)
      mnuManual.Enabled = True
      If (Not (IO1.In(PAR_STAT)) And &H8) Then   'If it is a bad checksum,
        SSFrame1(3).Caption = mstrFileTitle & " Bad Checksum"   ' then deal with it
        Me.Refresh
        strMessage1 = "Checksum Error" & Chr(10) & "Do you want to try again?"
        intMsgResult = MsgBox(strMessage1, vbRetryCancel)
        If intMsgResult = vbRetry Then GoTo Retry
        Exit Sub      'otherwise quit
      End If '(Not (IO1.In(PAR_STAT)) And &H8)
      SSFrame1(3).Caption = mstrFileTitle & " Transfer Completed"
      Me.Refresh
      Timer1.Interval = 100      'Turn timer back on for manual port control
    End With 'frmFileID.cdlFIleID
End Sub
```

4.3 File Conversion

If a file with the ‘dat’ extension is selected, a new file will be created in the ‘hgf’ format with the same file name base. When converting the file the channel and time parameters are unchanged with the exception of shortening the time to 3 bytes. The amplitude is converted using the parameters contained in the file "C:\HessiCalib.asc". This file contains 18 records, one for each channel. Each record has a gain (floating point), range (floating-point) and offset (fixed-point) value in ASCII text, separated by spaces. The amplitude (hgf) is calculated from the amplitude (dat) using the following formula:

$$Amp_{hgf} = (Amp_{dat} * (gain / range)) + offset$$

The gain is nominally 65535 which represent .27 volts output by the GSE for channels 0-8 and .81 volts for channels 9-17. The range is the expected full range value in the ‘dat’ file, typically 8191 for channels 0-8 and 50000 for channels 9-17. The offset is used to correct for offsets in the electronics and is nominally 0.

If a discrepancy in the ‘dat’ file format is discovered, the words “Conversion Terminated” will appear on the status bar, and a message box will appear informing the user of the cause of termination.

If the conversion completed without error the file transfer of the ‘hfg’ file is started as described in 4.2 (above).

The software listing that follows is of the fblnFileConvert subroutine that is called by mnuFileOpen when a file with the extension ‘dat’ is selected in the file dialog box as described above. After initializing variables and obtaining the data from the HessiCalib.asc file used for data conversion, the ‘dat’ file is opened and checked for valid length. Then data is read from the ‘dat’ file 28000 bytes (4000 records) at a time, converted to ‘hgf’ format and output to the ‘hgf’ file. The remaining data is read, converted and output one record at a time. Finally a checksum record is constructed and appended to the output file.

```
Private Function fblnFileConvert() As Boolean

   Dim lngNumRecords As Long, lngInBytePointer As Long, lngOutBytePointer As Long
   Dim lngTime As Long, bytChan As Byte, lngErrCode As Long, lngCheckSum As Long
   Dim strMessage1 As String, strMessage2 As String, lngAmpl As Long
   Dim strMessage As String, intDummy As Integer, lngLoopCounter As Long
   Dim bytMidAddr As Byte, bytLSBAddr As Byte, bytMSBAddr As Byte
   Dim bytLSBAmpl As Byte, bytMSBAmpl As Byte, dblGain(0 To 17) As Double
   Dim dblRange(0 To 17) As Double, lngOffset(0 To 17) As Long
   Dim udtCheckSumRec As typCheckSumRec, lngLoopCount As Long

   strMessage1 = "The input file "                                    'Initialize error message
   strMessage2 = Chr(10) & "does not have a recognizable format " & Chr(10) & _
                 "File Conversion Terminated" & Chr(10) & "Error Code"

   Open "c:\HessiCalib.asc" For Input As #1
   For lngLoopCount = 0 To 17                                         'Get calibration parameters
      Input #1, dblGain(lngLoopCount), dblRange(lngLoopCount), lngOffset(lngLoopCount)
   Next lngLoopCount
   Close #1
   lngInBytePointer = 1                                             'Set up file pointers
   lngOutBytePointer = 1
   lngCheckSum = 0
   Open frmFileID.cdlFIleID.FileName For Binary As #1                 'Open input file
   Get #1, lngInBytePointer, lngNumRecords                            ' get number of records
   lngInBytePointer = lngInBytePointer + 4
   If (LOF(1) - 4) < > (lngNumRecords * 7) Then lngErrCode = 101: GoTo Invalid  '  and check for validity
   Open mstrFileName For Binary As #2                                 'Open output file
   SSFrame1(3).Caption = frmFileID.cdlFIleID.FileTitle & " Converting"        'initialize progress status bar
   ProgressBar1.Value = 1
   ProgressBar1.Max = 101
   Do While (LOF(1) - lngInBytePointer) > 28000                        'this loop is for doing BIG chunks of data
      Get #1, lngInBytePointer, gudtInConvert                         'Get input array
      lngInBytePointer = lngInBytePointer + 28000
      For lngLoopCounter = 1 To 4000                                  'This loop process one records at a time from array
         lngTime = gudtInConvert.Arr(lngLoopCounter).lngTime              'get one input record from array
         bytChan = gudtInConvert.Arr(lngLoopCounter).bytChan
         lngAmpl = gudtInConvert.Arr(lngLoopCounter).intAmpl
         If lngAmpl < 0 Then lngAmpl = lngAmpl + 65536                 'Convert signed to unsigned
         If lngTime < 0 Or lngTime > 4194303 Then lngErrCode = 102: GoTo Invalid 'check time   for validity
         If bytChan < 0 Or bytChan > 17 Then lngErrCode = 103: GoTo Invalid     ' "   Channel "      "
         If lngAmpl > dblRange(bytChan) Then lngErrCode = 104: GoTo Invalid
```

```
      lngAmpl = CLng(CDbl(lngAmpl) * dblGain(bytChan) / dblRange(bytChan)) + lngOffset(bytChan)
      If lngAmpl > 0 Then lngAmpl = lngAmpl Or 1                    'Turn on LSB to indicate non-zero value
      gudtOutConvert.Arr(lngLoopCounter).bytChan = bytChan          'Store output data in array
      lngCheckSum = (lngCheckSum + bytChan) And &HFFFF&                 '  and compute checksum
      bytLSBAddr = lngTime And &HFF&
      gudtOutConvert.Arr(lngLoopCounter).bytLSBAddr = bytLSBAddr
      lngCheckSum = (lngCheckSum + bytLSBAddr) And &HFFFF&
      bytMidAddr = (lngTime And &HFF00&) / &H100&
      gudtOutConvert.Arr(lngLoopCounter).bytMidAddr = bytMidAddr
      lngCheckSum = (lngCheckSum + bytMidAddr) And &HFFFF&
      bytMSBAddr = (lngTime And &HFF0000) / &H10000
      gudtOutConvert.Arr(lngLoopCounter).bytMSBAddr = bytMSBAddr
      lngCheckSum = (lngCheckSum + bytMSBAddr) And &HFFFF&
      bytLSBAmpl = lngAmpl And &HFF&
      gudtOutConvert.Arr(lngLoopCounter).bytLSBAmpl = bytLSBAmpl
      lngCheckSum = (lngCheckSum + bytLSBAmpl) And &HFFFF&
      bytMSBAmpl = (lngAmpl And &HFF00&) / &H100&
      gudtOutConvert.Arr(lngLoopCounter).bytMSBAmpl = bytMSBAmpl
      lngCheckSum = (lngCheckSum + bytMSBAmpl) And &HFFFF&
   Next lngLoopCounter
   Put #2, lngOutBytePointer, gudtOutConvert                        'Put output array
   lngOutBytePointer = lngOutBytePointer + 24000
   ProgressBar1.Value = 100 * (lngInBytePointer / LOF(1))           'Update progress bar
   Me.Refresh
Loop 'While (LOF(1) - lngInBytePointer) > 28000                     'Do another BIG chunk of records
Do While LOF(1) > lngInBytePointer                                  'Here for the remaining data after all the BIG chunks
   Get #1, lngInBytePointer, gudtInConvertRec                       'Get one input record
   lngInBytePointer = lngInBytePointer + 7
   lngTime = gudtInConvertRec.lngTime                               'get input data from record
   bytChan = gudtInConvertRec.bytChan
   lngAmpl = gudtInConvertRec.intAmpl
   If lngAmpl < 0 Then lngAmpl = lngAmpl + 65536                    'Convert signed to unsigned
   If lngTime < 0 Or lngTime > 4194303 Then lngErrCode = 102: GoTo Invalid 'check time   for validity
   If bytChan < 0 Or bytChan > 17 Then lngErrCode = 103: GoTo Invalid     ' "   Channel "      "
   If bytChan < 9 Then                                              'Scale amplitude and check for validity
      If lngAmpl > 8191 Then lngErrCode = 104: GoTo Invalid
      lngAmpl = CLng(CDbl(lngAmpl) * 65535# / 8191#)
   Else
      If lngAmpl > 50000 Then lngErrCode = 105: GoTo Invalid
      lngAmpl = CLng(CDbl(lngAmpl) * 65535# / 50000#)
   End If 'bytChan < 9
   If lngAmpl > 0 Then lngAmpl = lngAmpl Or 1                       'Turn on LSB to indicate non-zero value
```

```
        udtOutConvertRec.bytChan = bytChan                          'Store output data in output record
        lngCheckSum = (lngCheckSum + bytChan) And &HFFFF&           '  and compute checksum
        bytLSBAddr = lngTime And &HFF&
        udtOutConvertRec.bytLSBAddr = bytLSBAddr
        lngCheckSum = (lngCheckSum + bytLSBAddr) And &HFFFF&
        bytMidAddr = (lngTime And &HFF00&) / &H100&
        udtOutConvertRec.bytMidAddr = bytMidAddr
        lngCheckSum = (lngCheckSum + bytMidAddr) And &HFFFF&
        bytMSBAddr = (lngTime And &HFF0000) / &H10000
        udtOutConvertRec.bytMSBAddr = bytMSBAddr
        lngCheckSum = (lngCheckSum + bytMSBAddr) And &HFFFF&
        bytLSBAmpl = lngAmpl And &HFF&
        udtOutConvertRec.bytLSBAmpl = bytLSBAmpl
        lngCheckSum = (lngCheckSum + bytLSBAmpl) And &HFFFF&
        bytMSBAmpl = (lngAmpl And &HFF00&) / &H100&
        udtOutConvertRec.bytMSBAmpl = bytMSBAmpl
        lngCheckSum = (lngCheckSum + bytMSBAmpl) And &HFFFF&
        Put #2, lngOutBytePointer, udtOutConvertRec                 'Put output record
        lngOutBytePointer = lngOutBytePointer + 6                   'Do another records
    Loop 'While LOF(1) > lngInBytePointer
    ProgressBar1.Value = 101                                        'Update progress bar to show finished
    Me.Refresh
    udtCheckSumRec.bytChan = &H85                                   'Make checksum output record
    lngCheckSum = (lngCheckSum + &H85) And &HFFFF&
    udtCheckSumRec.bytLSBChkSum = lngCheckSum And &HFF&
    udtCheckSumRec.bytMSBChkSum = ((lngCheckSum And &HFF00&) / 256) And &HFF&
    udtCheckSumRec.bytDummy(0) = 0: udtCheckSumRec.bytDummy(1) = 0: udtCheckSumRec.bytDummy(2) = 0
    Put #2, lngOutBytePointer, udtCheckSumRec                       ' and put it in output file
    Close #1                                                        'Close files
    Close #2
    fblnFileConvert = True
    Exit Function                                                   ' and quit
Invalid:                                                            'Here when input file is invalid for some reason
    Close #1                                                        ' so close files
    Close #2
    SSFrame1(3).Caption = frmFileID.cdlFIleID.FileTitle & " Conversion Terminated" 'Change progress status bar
    strMessage = strMessage1 & frmFileID.cdlFIleID.FileTitle & strMessage2 & Format(lngErrCode, "###")
    intDummy = MsgBox(strMessage, vbExclamation)                    'Tell user what happened
    fblnFileConvert = False                                         ' and quit
End Function
```

5.0 HESSI Test File Make

This application creates test files in the 'hgf' format. It is started by running prjHessiTstFilMak. A listing of the source follows.

The first file output is named "c:\hessi1.hgf" and contains 10 records for each channel (0-17) with the following times and amplitudes:

Time	Amplitude
0x061A80	0x0001
0x0C3500	0x0011
0x124F80	0x00FF
0x186A00	0x0401
0x1E8480	0x0801
0x249F00	0x2001
0x2AB980	0x2F0F
0x30D455	0x5AA5
0x36EE80	0x6001
0x3D09AA	0x7FFF

The next file output is named "C:\HessiRamp.hgf" and contain repeating ramps on channel one. Each ramp has a data point every 1000 binary microsec.s. The first eleven points increment the amplitude by 100, the next ten increment by 1000 and the last 10 increment by 5000. There is a gap of 20 microsec.s before starting the next ramp.

The last file output contains sine waves based on parameters stored in a file called C:\HessiTest.asc, where the first line contains the name of the output file. The rest of the lines in the files contain the following parameters separated by commas.

Channel	PHA input I/D	
lngDecim	sample rate	Microsec.s
dblFreq	Frequence of the sine wave	Hz
dblPhase	Phase of the sine wave	Degrees
dblMaxAmp	Amplitude of the sine wave	Percent of corresponding Gain parameter in HessiCalib.asc)
dblOffset	DC offset of the sine wave	Percent of corresponding Gain parameter in HessiCalib.asc

```
Public Sub subMakeTestFile()
   Dim udtOutRec As typOutRec, lngAmplitude As Long
   Dim bytOutput As Byte, lngLoopCount As Long, lngOutput As Long
   Dim dblAmplitude As Double, lngDecim As Long, dblScale As Double
   Dim dblPhase As Double, dblMaxAmp As Double, dblOffSet As Double
   Dim strOutFilNam As String, dblFreq As Double, dblGain(0 To 17) As Double
   Dim lngOffset(0 To 17) As Long, dblRange(0 To 17) As Double
   Dim lngAddress As Long, lngStartLoop As Long, lngEndLoop As Long
   Const conPi = 3.14159265358979

   Open "C:\HessiCalib.asc" For Input As #1    'Get calibration parameters
   For lngLoopCount = 0 To 17
      Input #1, dblGain(lngLoopCount), dblRange(lngLoopCount), lngOffset(lngLoopCount)
   Next lngLoopCount
   Close #1

   'First put out short test file: all channels, 10 identical events each, making a ramp
   Open "c:\hessi1.hgf" For Binary As #1
   mlngBytePointer = 1
   mlngCheckSum = 0

   For lngLoopCount = 0 To 17
      bytOutput = lngLoopCount                                    'Set Channel number

      Call subPutAByteInFile(bytOutput)                           'Put out Channel
      lngOutput = &H61A80: Call subPutThreeBytesInFile(lngOutput)      'Put out First Address
      lngOutput = &H1&: Call subPutTwoBytesInFile(lngOutput)           'Put out First Amplitude
      Call subPutAByteInFile(bytOutput)                           'Put out Channel
      lngOutput = &HC3500: Call subPutThreeBytesInFile(lngOutput)      'Put out Second Address
      lngOutput = &H11&: Call subPutTwoBytesInFile(lngOutput)          'Put out Second Amplitude
      Call subPutAByteInFile(bytOutput)                           'Put out Channel
      lngOutput = &H124F80: Call subPutThreeBytesInFile(lngOutput)     'Put out Third Address
      lngOutput = &HFF&: Call subPutTwoBytesInFile(lngOutput)          'Put out Third Amplitude
      Call subPutAByteInFile(bytOutput)                           'Put out Channel
      lngOutput = &H186A00: Call subPutThreeBytesInFile(lngOutput)     'Put out Fourth Address
      lngOutput = &H401&: Call subPutTwoBytesInFile(lngOutput)         'Put out Fourth Amplitude
      Call subPutAByteInFile(bytOutput)                           'Put out Channel
      lngOutput = &H1E8480: Call subPutThreeBytesInFile(lngOutput)     'Put out Fifth Address
      lngOutput = &H801&: Call subPutTwoBytesInFile(lngOutput)         'Put out Fifth Amplitude
      Call subPutAByteInFile(bytOutput)                           'Put out Channel
      lngOutput = &H249F00: Call subPutThreeBytesInFile(lngOutput)     'Put out Sixth Address
      lngOutput = &H2001&: Call subPutTwoBytesInFile(lngOutput)        'Put out Sixth Amplitude
```

```
   Call subPutAByteInFile(bytOutput)                                  'Put out Channel
   lngOutput = &H2AB980: Call subPutThreeBytesInFile(lngOutput)       'Put out Seventh Address
   lngOutput = &H2F0F&: Call subPutTwoBytesInFile(lngOutput)          'Put out Seventh Amplitude
   Call subPutAByteInFile(bytOutput)                                  'Put out Channel
   lngOutput = &H30D455: Call subPutThreeBytesInFile(lngOutput)       'Put out Eighth Address
   lngOutput = &H5AA5&: Call subPutTwoBytesInFile(lngOutput)          'Put out Eighth Amplitude
   Call subPutAByteInFile(bytOutput)                                  'Put out Channel
   lngOutput = &H36EE80: Call subPutThreeBytesInFile(lngOutput)       'Put out Ninth Address
   lngOutput = &H6001&: Call subPutTwoBytesInFile(lngOutput)          'Put out Ninth Amplitude
   Call subPutAByteInFile(bytOutput)                                  'Put out Channel
   lngOutput = &H3D09AA: Call subPutThreeBytesInFile(lngOutput)       'Put out Tenth Address
   lngOutput = &H7FFF&: Call subPutTwoBytesInFile(lngOutput)          'Put out Tenth Amplitude
Next lngLoopCount

bytOutput = &H85: Call subPutAByteInFile(bytOutput)                   'Chan 85 indicate checksum follows
lngOutput = mlngCheckSum And &HFFFF
Call subPutTwoBytesInFile(lngOutput)                                  'Put out Checksum
bytOutput = 0: Call subPutAByteInFile(bytOutput)                      'Pad with zeroes
lngOutput = 0: Call subPutTwoBytesInFile(lngOutput)

Close #1


'Then put out a file with a different ramp on channel 1
Open "C:\HessiRamp.hgf" For Binary As #1
mlngBytePointer = 1
mlngCheckSum = 0
lngAddress = 0                                                        'Initialize address
lngAmplitude = 0
bytOutput = 1                                                         'Set Channel address to 1
Do While lngAddress < 4194304
   lngStartLoop = lngAddress: lngEndLoop = lngStartLoop + 10000
   For lngLoopCount = lngStartLoop To lngEndLoop Step 1000
      If lngLoopCount > 4194304 Then Exit For
      Call subPutAByteInFile(bytOutput)                               'Put out Channel
      Call subPutThreeBytesInFile(lngLoopCount)                       'Put out Address
      lngAmplitude = lngAmplitude + 100                               'Increment amplitude
      Call subPutTwoBytesInFile(lngAmplitude)                         'Put out Amplitude
      Debug.Print lngLoopCount, lngAmplitude
   Next lngLoopCount
   lngAddress = lngLoopCount
   If lngAddress > 4194304 Then Exit Do
```

```
    lngStartLoop = lngAddress: lngEndLoop = lngStartLoop + 9000
    For lngLoopCount = lngStartLoop To lngEndLoop Step 1000
       If lngLoopCount > 4194304 Then Exit For
       Call subPutAByteInFile(bytOutput)                          'Put out Channel
       Call subPutThreeBytesInFile(lngLoopCount)                      'Put out Address
       lngAmplitude = lngAmplitude + 1000                        'Increment amplitude
       Call subPutTwoBytesInFile(lngAmplitude)                    'Put out Amplitude
       Debug.Print lngLoopCount, lngAmplitude
    Next lngLoopCount
    lngAddress = lngLoopCount
    If lngAddress > 4194304 Then Exit Do

    lngStartLoop = lngAddress: lngEndLoop = lngStartLoop + 9000
    For lngLoopCount = lngStartLoop To lngEndLoop Step 1000
       If lngLoopCount > 4194304 Then Exit For
       Call subPutAByteInFile(bytOutput)                          'Put out Channel
       Call subPutThreeBytesInFile(lngLoopCount)                      'Put out Address
       lngAmplitude = lngAmplitude + 5000                        'Increment amplitude
       Call subPutTwoBytesInFile(lngAmplitude)                    'Put out Amplitude
       Debug.Print lngLoopCount, lngAmplitude
    Next lngLoopCount
    lngAddress = lngLoopCount + 20 - 1000
    If lngAddress > 4194304 Then Exit Do
    lngAmplitude = 0
    Call subPutAByteInFile(bytOutput)                             'put out channel
    Call subPutThreeBytesInFile(lngAddress)
    Call subPutTwoBytesInFile(lngAmplitude)
    Debug.Print lngAddress, lngAmplitude
    lngAddress = lngAddress + 1000
    If lngAddress > 4194304 Then Exit Do
Loop 'While lngAddress < 4194304

bytOutput = &H85: Call subPutAByteInFile(bytOutput)                  'Chan 85 indicate checksum follows
lngOutput = mlngCheckSum And &HFFFF
Call subPutTwoBytesInFile(lngOutput)                              'Put out Checksum
bytOutput = 0: Call subPutAByteInFile(bytOutput)                      'Pad with zeroes
lngOutput = 0: Call subPutTwoBytesInFile(lngOutput)


Close #1
```

```
'Then put out a file based on input parameters from a file called C:\HessiTest.asc
Open "c:\HessiTest.asc" For Input As #2
Input #2, strOutFilNam
Open strOutFilNam For Binary As #1      'Get rid of file if already exists
Close #1
Kill strOutFilNam
Open strOutFilNam For Binary As #1      'Now we are creating a brand new file
mlngBytePointer = 1
mlngCheckSum = 0
Do While Not EOF(2)
   Input #2, udtOutRec.bytChan, lngDecim, dblFreq, dblPhase, dblMaxAmp, dblOffSet
   For lngLoopCount = 0 To 4194304 Step lngDecim
      udtOutRec.bytAddr1 = lngLoopCount And &HFF&                    'Put out Address
      udtOutRec.bytAddr2 = (lngLoopCount And &HFF00&) / 256&
      udtOutRec.bytAddr3 = (lngLoopCount And &HFF0000) / 65536
      dblScale = dblGain(udtOutRec.bytChan) / 100#
      dblAmplitude = (((Sin(((CDbl(lngLoopCount) / 1048576#) * 2# * conPi * dblFreq) + _
         ((dblPhase / 360#) * 2# * conPi)) / 2#) + 0.5) * dblMaxAmp * dblScale) + _
         dblOffSet * dblScale 'Put out Sine Wave
      If dblAmplitude > 65535 Then dblAmplitude = 65535              'Truncate to allowed range of values
      If dblAmplitude < 0 Then dblAmplitude = 0
      lngAmplitude = CLng(dblAmplitude) + lngOffset(udtOutRec.bytChan)
      udtOutRec.bytAmpl1 = lngAmplitude And &HFF&
      udtOutRec.bytAmpl2 = (lngAmplitude And &HFF00&) / 256
      mlngCheckSum = (mlngCheckSum + udtOutRec.bytChan) And &HFFFF&      'Update checksum
      mlngCheckSum = (mlngCheckSum + udtOutRec.bytAddr1) And &HFFFF&
      mlngCheckSum = (mlngCheckSum + udtOutRec.bytAddr2) And &HFFFF&
      mlngCheckSum = (mlngCheckSum + udtOutRec.bytAddr3) And &HFFFF&
      mlngCheckSum = (mlngCheckSum + udtOutRec.bytAmpl1) And &HFFFF&
      mlngCheckSum = (mlngCheckSum + udtOutRec.bytAmpl2) And &HFFFF&
      Put #1, mlngBytePointer, udtOutRec                             'Put record out
      mlngBytePointer = mlngBytePointer + 6
   Next lngLoopCount
Loop 'While Not EOF(2)
bytOutput = &H85: Call subPutAByteInFile(bytOutput)                  'Chan 85 indicate checksum follows
lngOutput = mlngCheckSum And &HFFFF
Call subPutTwoBytesInFile(lngOutput)                                 'Put out Checksum
bytOutput = 0: Call subPutAByteInFile(bytOutput)                     'Pad with zeroes
lngOutput = 0: Call subPutTwoBytesInFile(lngOutput)

Close #1
Close #2
```

```
End Sub
Public Sub subPutAByteInFile(bytOutput As Byte)

   mlngCheckSum = (mlngCheckSum + bytOutput) And &HFFFF&
   Put #1, mlngBytePointer, bytOutput
   mlngBytePointer = mlngBytePointer + 1

End Sub
Public Sub subPutThreeBytesInFile(lngOutput As Long)

   Dim bytOutput As Byte

   bytOutput = lngOutput And &HFF&: Call subPutAByteInFile(bytOutput)
   bytOutput = (lngOutput And &HFF00&) / 256&: Call subPutAByteInFile(bytOutput)
   bytOutput = (lngOutput And &HFF0000) / 65536: Call subPutAByteInFile(bytOutput)

End Sub
Public Sub subPutTwoBytesInFile(lngOutput As Long)

   Dim bytOutput As Byte

   bytOutput = lngOutput And &HFF&: Call subPutAByteInFile(bytOutput)
   bytOutput = (lngOutput And &HFF00&) / 256: Call subPutAByteInFile(bytOutput)

End Sub
```

HESSI GSE HARDWARE

6. Introduction

The HESSI GSE hardware consists of a PC connected through a standard parallel port to a separate chassis referred to as the HESSI IMAGER which is capable of producing 18 channels of analog output. The waveform produced on each of these 18 channels is determined by the data transmitted by the PC to the HESSI IMAGER. The analog output is synchronized to the external clock provided on the 9 pin connector. Also on this connector is an output signal capable of driving an LED to indicate the start of an image.

6.1 Theory of Operation

The HESSI IMAGER consists of the following elements:

1.) An optically isolated parallel interface for the receipt of data from the host PC.
2.) A MPU which controls the transfer of data from the parallel interface to the internal DRAM.
3.) 18 8Mword memory channels
4.) A dynamic RAM memory controller which synchronously clocks data to the analog output system
5.) 6, 3 channel DAC driven analog output sections capable of ramping from –2.5V to +2.5V.

6.2 Parallel interface

The parallel interface is optically isolated within the HESSI IMAGER in order to provide isolation between chassis ground and the analog ground reference for the output channels. This 8 bit data is transmitted across this interface in the format described in section 3.

6.3 MPU Operation

The internal MPU controls the flow of data from the parallel interface to the internal dynamic RAM. This process involves the bulk erasing of all of the dynamic RAM followed by the writing of 16 bit data to address specified within the received data packets. The MPU will report through the parallel interface the completion status of the data transfer as detailed in section 3.

6.4 Dynamic RAM

The HESSI IMAGER contains 9 32bit X 8Mbyte dynamic RAM memory sticks. This memory is configured to provide 18 16 bit x 8Mword channels. Under the direction of the DRAM controller, this memory stores the complete image for all 18 channels and provides image sequentially to each of 18 DAC driven analog circuits.

6.5 Dynamic RAM controller

The DRAM controller is designed to allow writing of randomly addressed data from the MPU, and read sequentially addressed data to the 18 DAC driven analog sections. With no clock present at the external clock input on the DB-9 connector, the DRAM controller will continuously clock through 4,194,304 address at a rate of 1.25Mcps. When an external 1.048576Mcps clock is present, the output of the DRAM controller will be synchronized to this clock.

The DB-9 connector also contains an output signal, which is capable of driving an LED. This output will be TRUE for 16us starting at address 0 of the internal DRAM address. This corresponds to an event time of 0.

6.6 DAC driven analog outputs

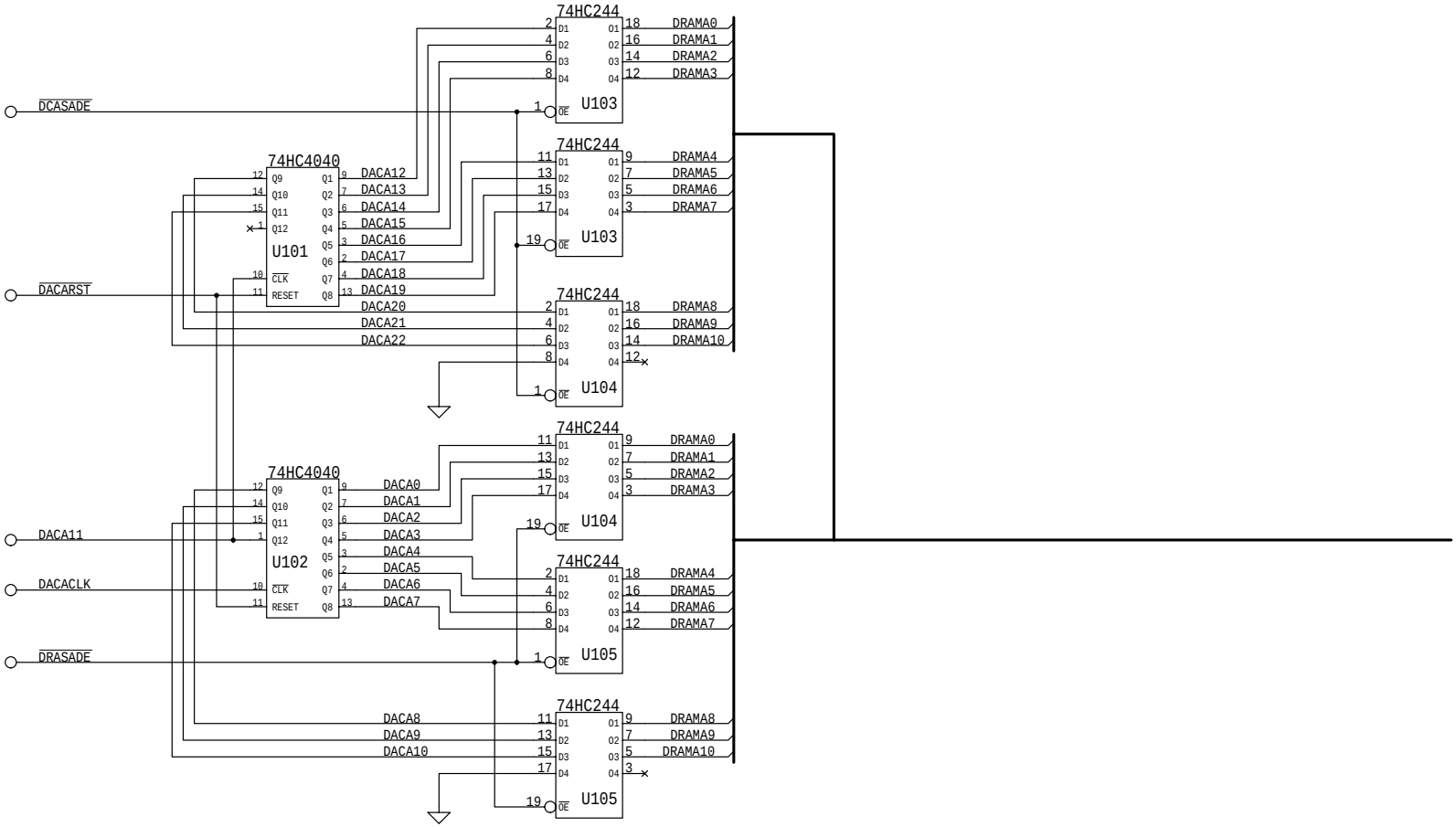
The HESSI IMAGER contains 6 3 channel analog circuit boards. Each of the 18 analog channels provides a positive and negative output. The Dynamic RAM controller continuously clocks the image data to each of the 18 DACs. If the least significant bit of the data word is 1 then the DAC is clocked an its output will update. After a delay, this output will be strobed into the analog step amplifier, which will increase its output in proportion to the amplitude of the DAC output. This will result in a step in the analog output. The analog output will continue to rise in this manner until it reaches approximately 2.5Volts. At this point the analog output will be reset to a value of approximately –2.5Volts. A slight positive bias is built into the circuit causing the analog output to rise smoothly even when no events are clock into the DACs. This is to ensure that the analog output does not drift negative.

6.7 HESSI IMAGER DRAWINGS

The following drawings detail the circuitry contained in the HESSI IMAGER along with a description of the external connector layout.

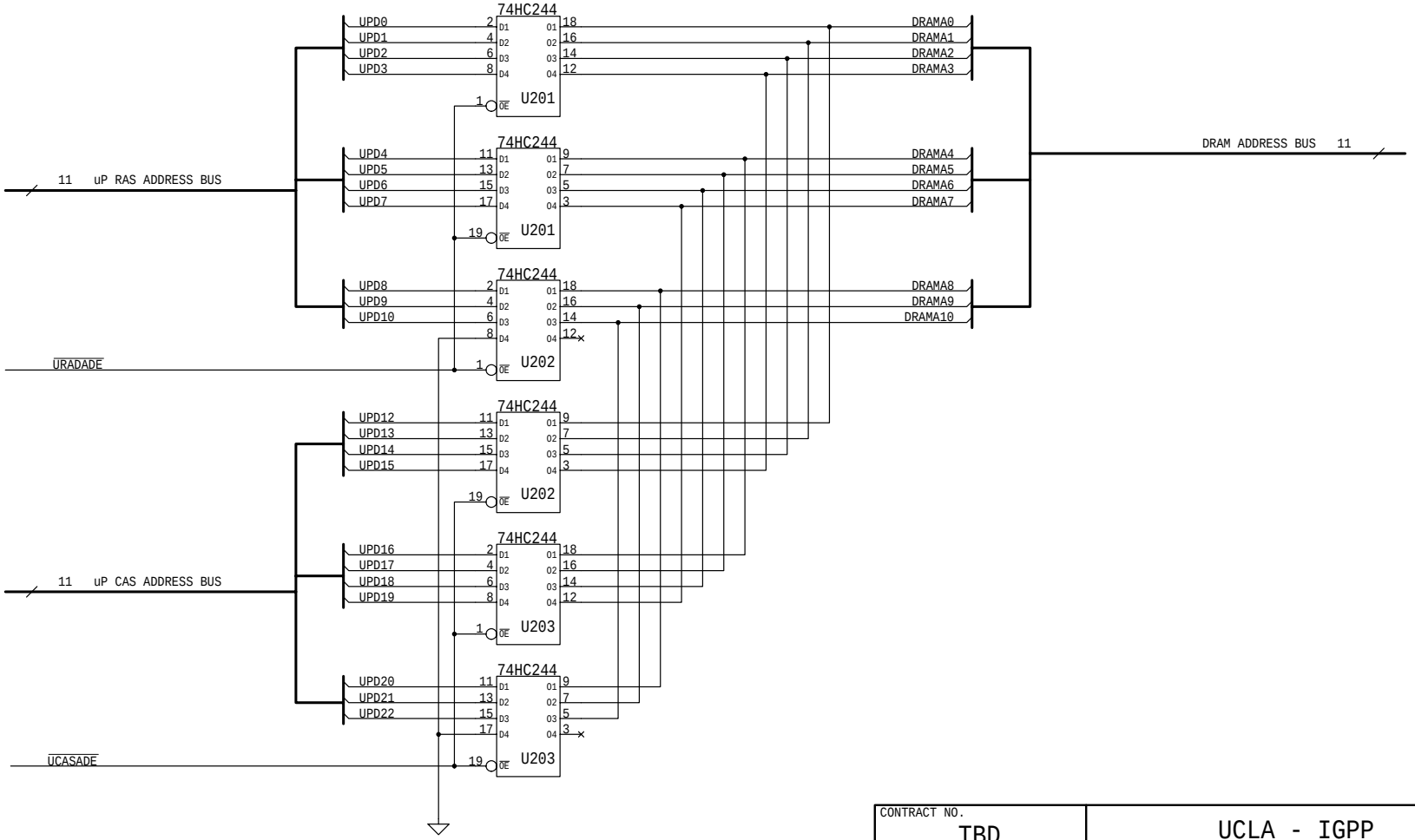
DWG. NO. HESSI10 | SH 1 | REV.

REVISIONS				
ZONE	REV	DESCRIPTION	DATE	APPROVED



CONTRACT NO. TBD		UCLA - IGPP	
APPROVALS	DATE	TITLE HESSI IMAGE GSE	
DRAWN DAVID PIERCE	9-10-98		
CHECKED		SIZE B / FSCM NO.	DWG. NO. HESSI101.SCH / REV.
		SCALE / FNAME HESSI101.SCH	SHEET 1 OF 11

REVISIONS				
ZONE	REV	DESCRIPTION	DATE	APPROVED



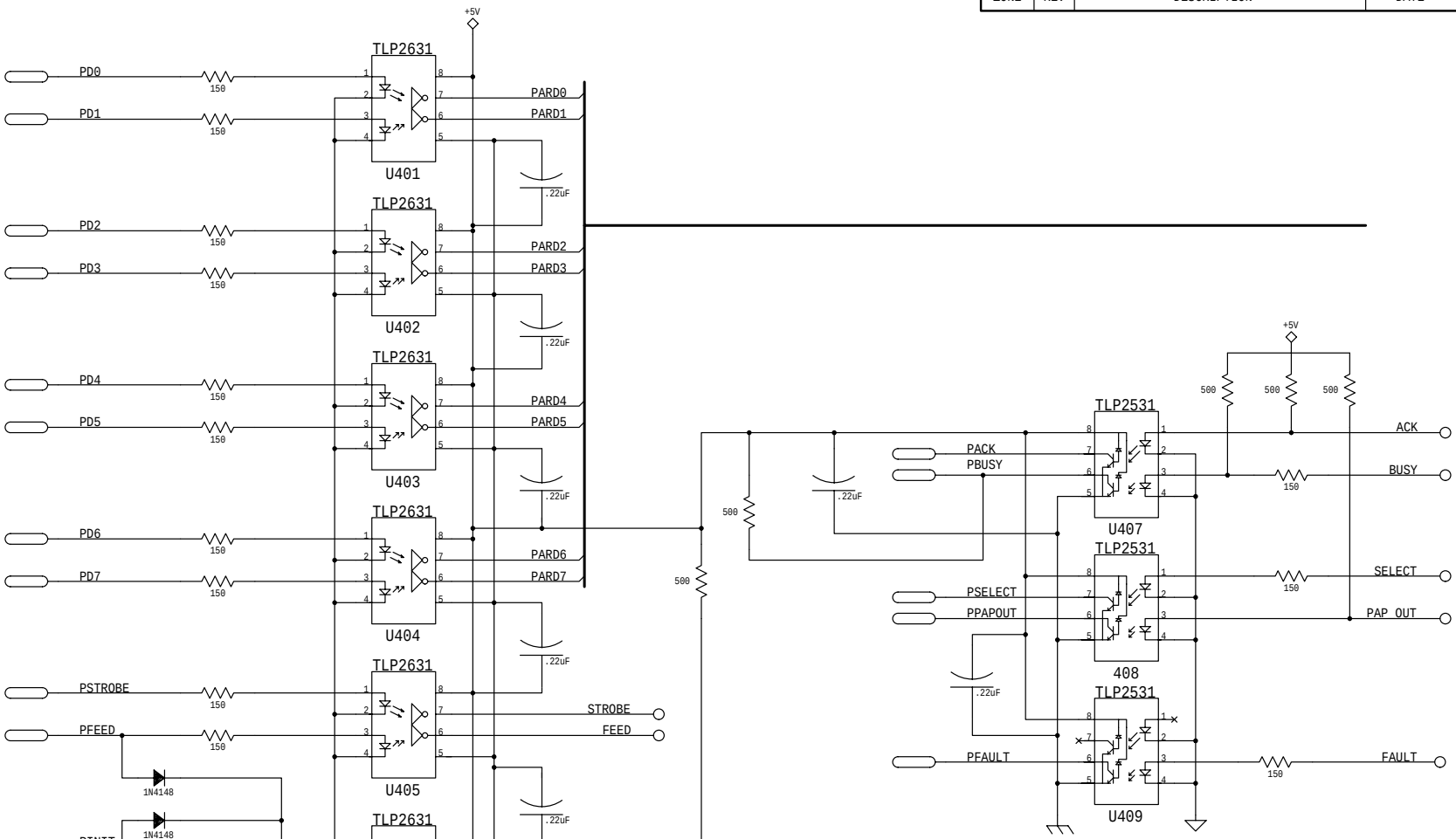
CONTRACT NO. TBD		UCLA - IGPP	
APPROVALS	DATE	TITLE	
DRAWN DAVID PIERCE	9-15-98	HESSI IMAGE GSE	
CHECKED		SIZE B / FSCM NO. / DWG. NO. HESSI102.SCH / REV.	
		SCALE / FNAME HESSI102.SCH / SHEET 2 OF 11	

REVISIONS				
ZONE	REV	DESCRIPTION	DATE	APPROVED

1 of 9

CONTRACT NO. TBD		UCLA - IGPP
APPROVALS	DATE	TITLE: HESSI IMAGE GSE
DRAWN DAVID PIERCE	9-11-98	
CHECKED		SIZE B | FSCM NO. | DWG. NO. HESSI103.SCH | REV.
		SCALE | FNAME HESSI103.SCH | SHEET 3 OF 11

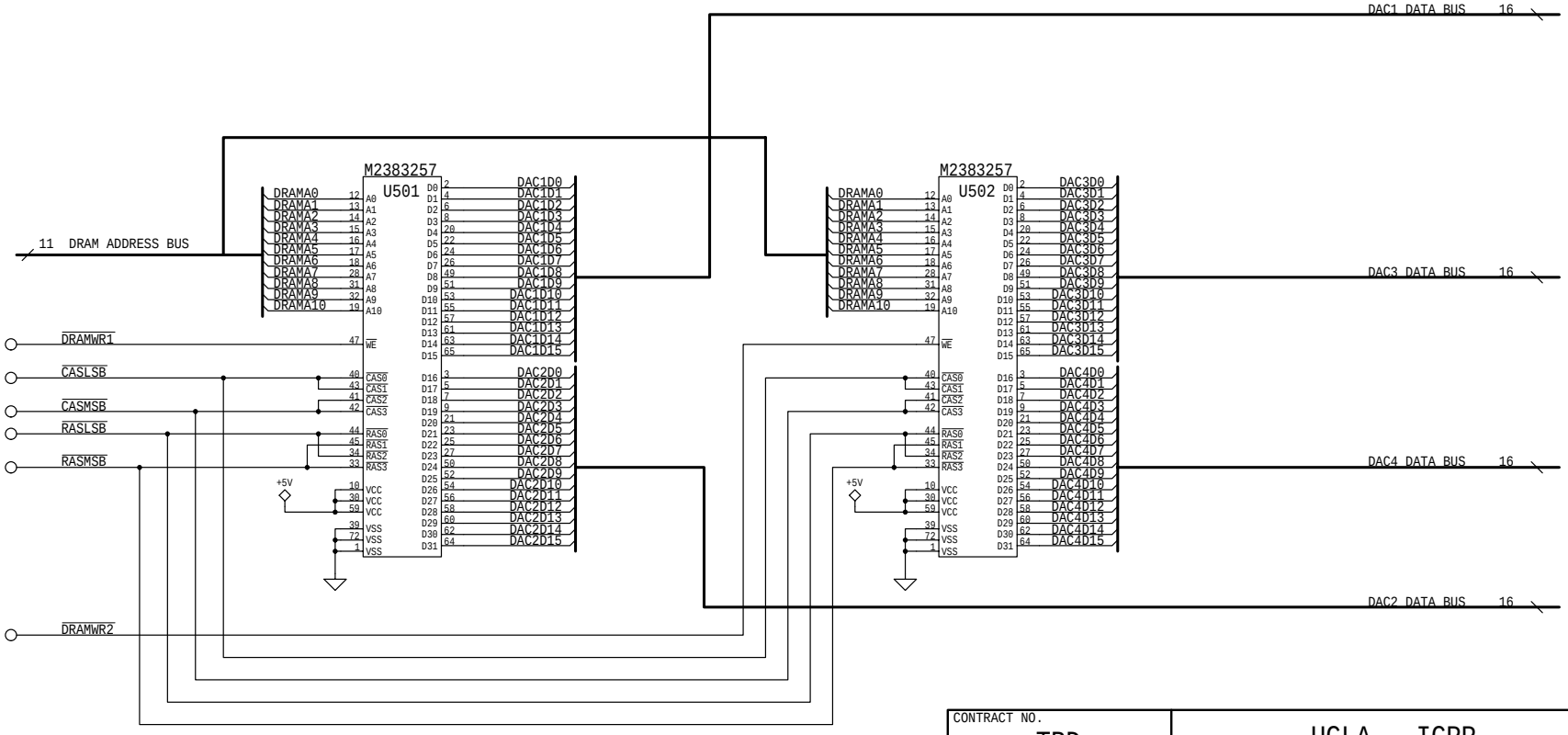
REVISIONS				
ZONE	REV	DESCRIPTION	DATE	APPROVED



CONTRACT NO. TBD		UCLA - IGPP	
APPROVALS	DATE	TITLE	HESSI IMAGE GSE
DRAWN DAVID PIERCE	4-15-99		
CHECKED		SIZE B / FSCM NO.	DWG. NO. HESSI104.SCH / REV.
		SCALE / FNAME HESSI104.SCH	SHEET 4 OF 11

DWG. NO. HESSI10 | SH 5 | REV.

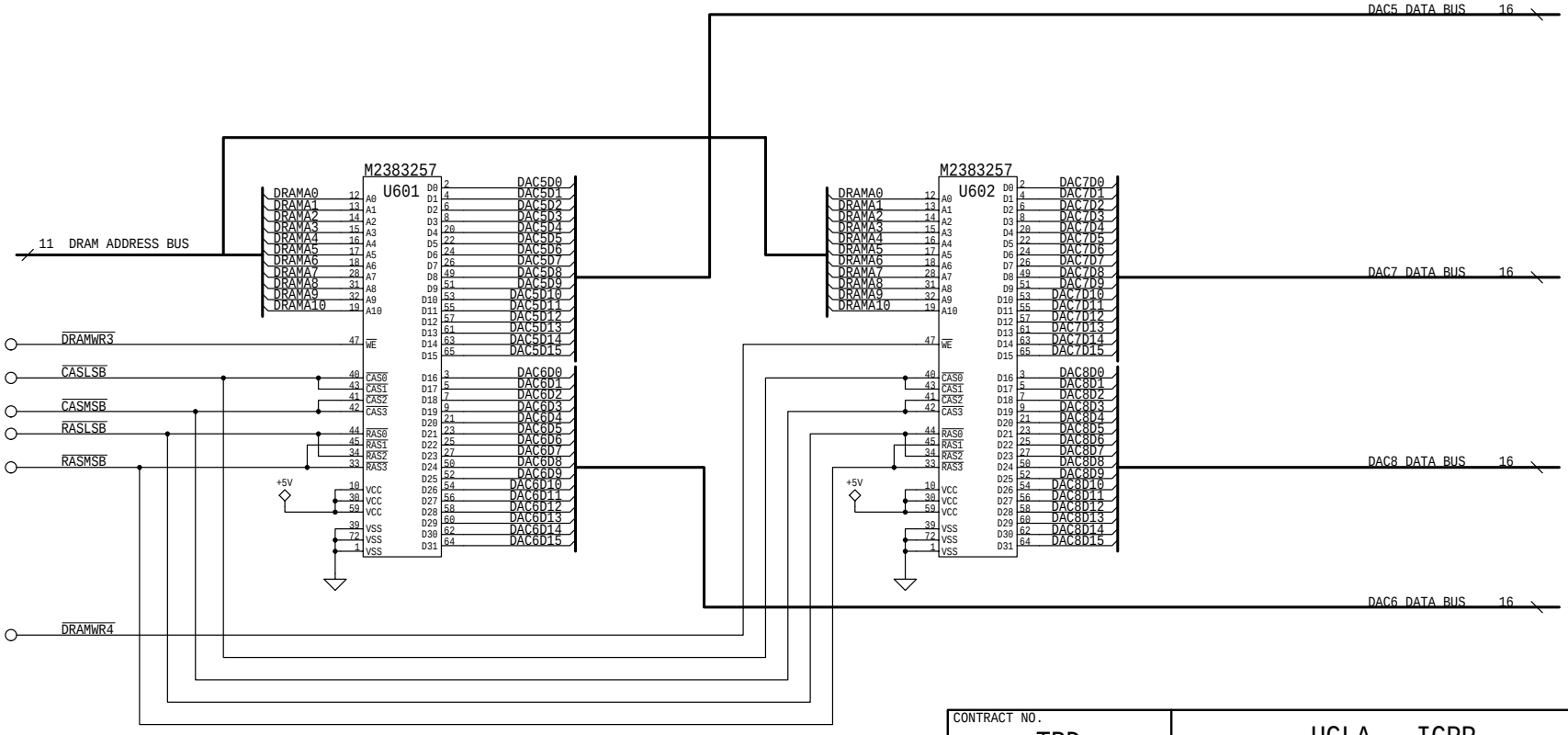
REVISIONS				
ZONE	REV	DESCRIPTION	DATE	APPROVED



CONTRACT NO. TBD		UCLA - IGPP	
APPROVALS	DATE	TITLE HESSI IMAGE GSE	
DRAWN DAVID PIERCE	9-12-98		
CHECKED		SIZE B / FSCM NO. / DWG. NO. HESSI105.SCH	REV.
		SCALE / FNAME HESSI105.SCH	SHEET 5 OF 11

DWG. NO. HESSI10 | SH 6 | REV.

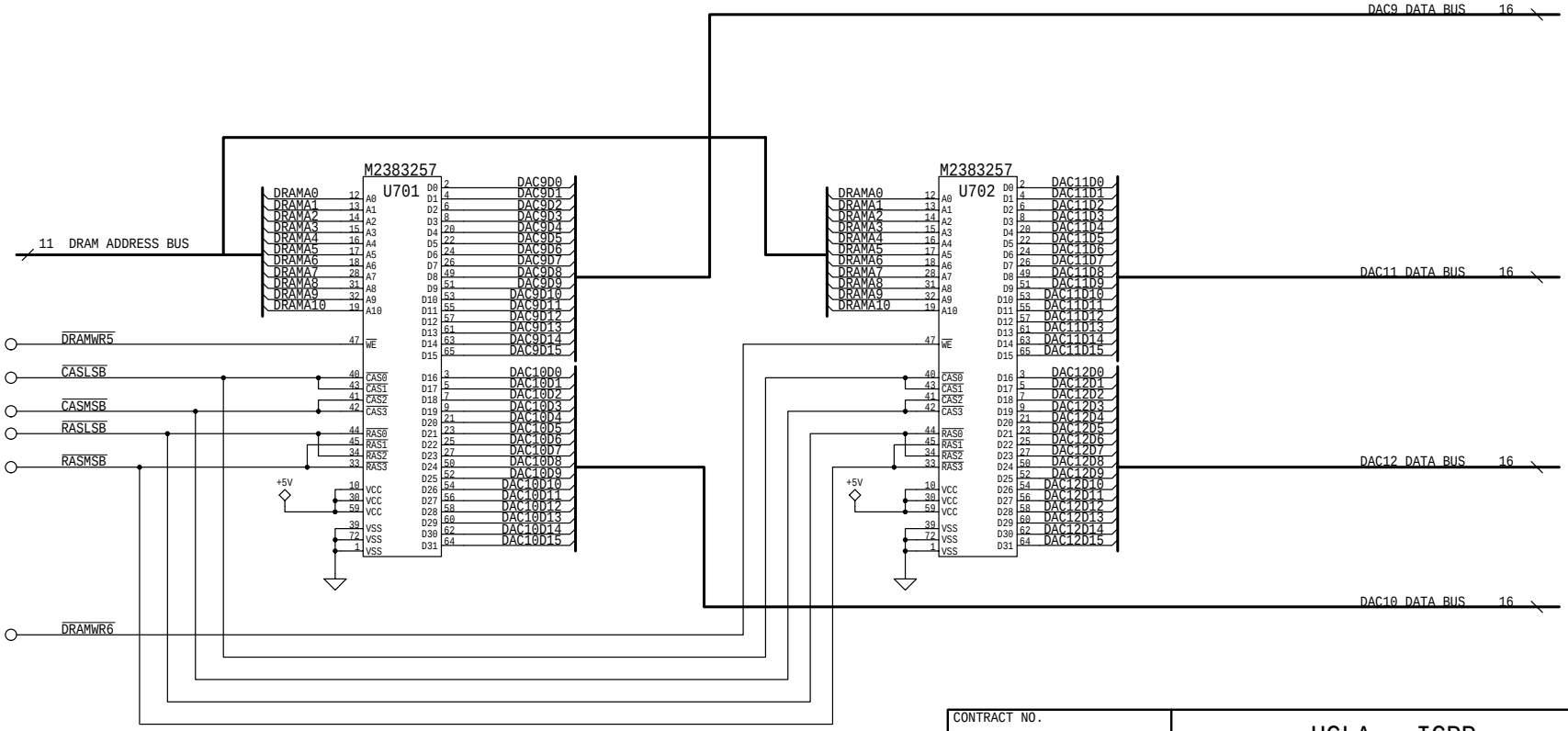
REVISIONS				
ZONE	REV	DESCRIPTION	DATE	APPROVED



CONTRACT NO. TBD		UCLA - IGPP	
APPROVALS	DATE	TITLE HESSI IMAGE GSE	
DRAWN DAVID PIERCE	9-12-98		
CHECKED		SIZE B / FSCM NO. / DWG. NO. HESSI106.SCH	REV.
		SCALE / FNAME HESSI106.SCH	SHEET 6 OF 11

DWG. NO. HESSI10 | SH 7 | REV.

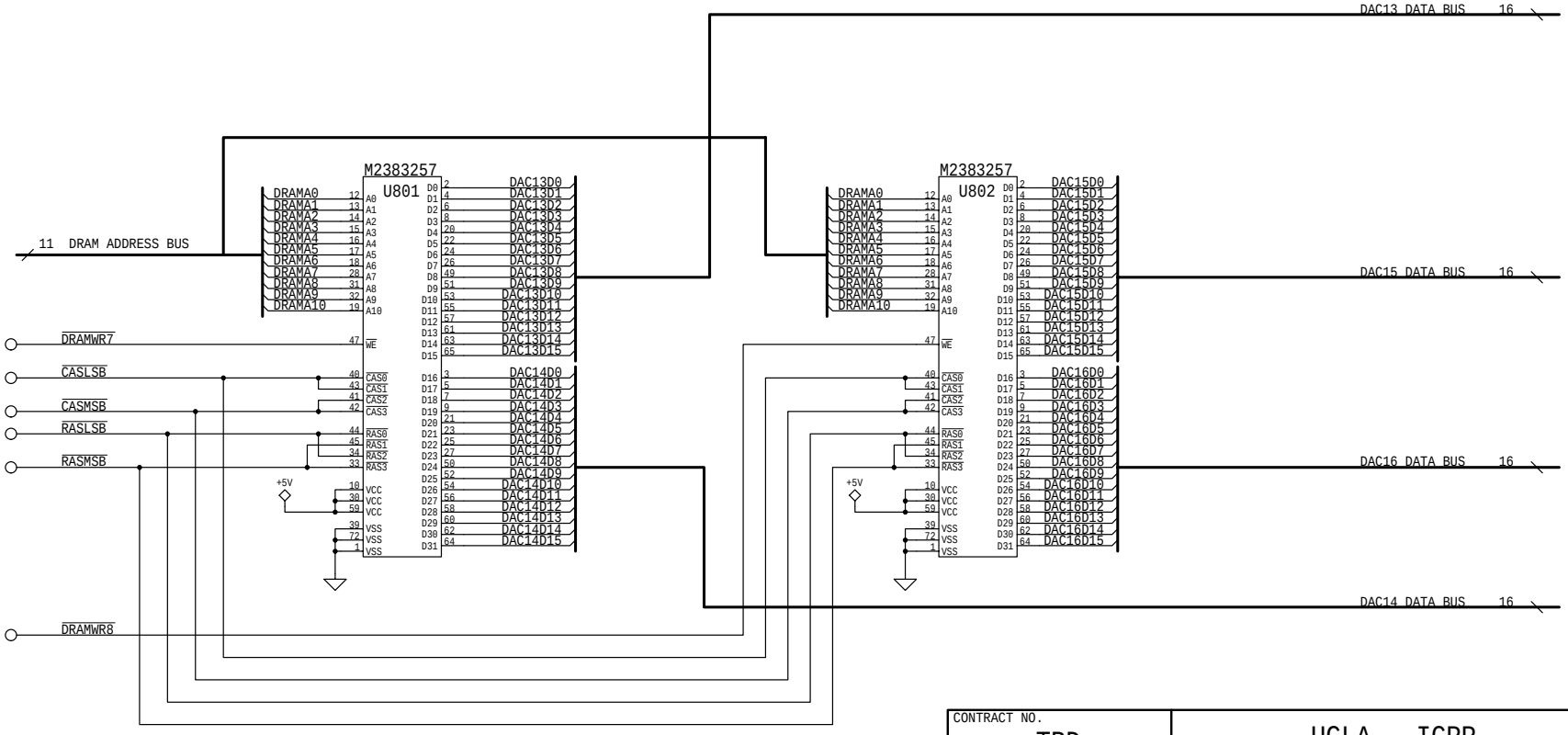
REVISIONS				
ZONE	REV	DESCRIPTION	DATE	APPROVED



CONTRACT NO. TBD		UCLA - IGPP	
APPROVALS	DATE	TITLE HESSI IMAGE GSE	
DRAWN DAVID PIERCE	9-12-98		
CHECKED		SIZE B / FSCM NO. / DWG. NO. HESSI107.SCH / REV.	
		SCALE / FNAME HESSI107.SCH / SHEET 7 OF 11	

DWG. NO. HESSI10 | SH 8 | REV.

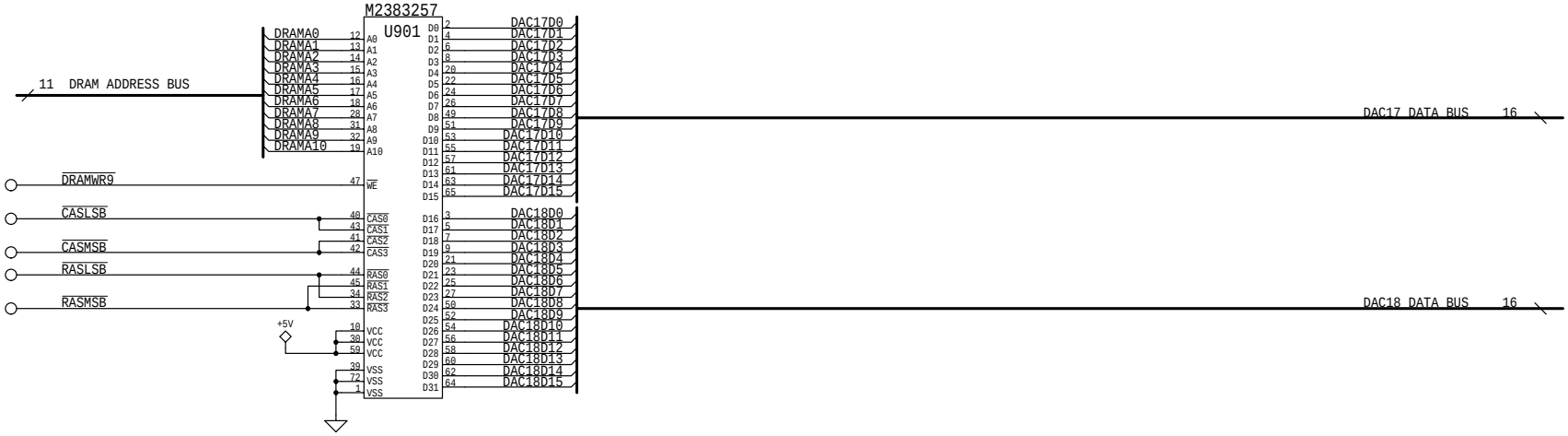
REVISIONS				
ZONE	REV	DESCRIPTION	DATE	APPROVED



CONTRACT NO. TBD		UCLA - IGPP	
APPROVALS	DATE	TITLE	
DRAWN DAVID PIERCE	9-12-98	HESSI IMAGE GSE	
CHECKED		SIZE B / FSCM NO. / DWG. NO. HESSI108.SCH / REV.	
		SCALE / FNAME HESSI108.SCH / SHEET 8 OF 11	

DWG. NO.	SH	REV.
HESSI10	9	

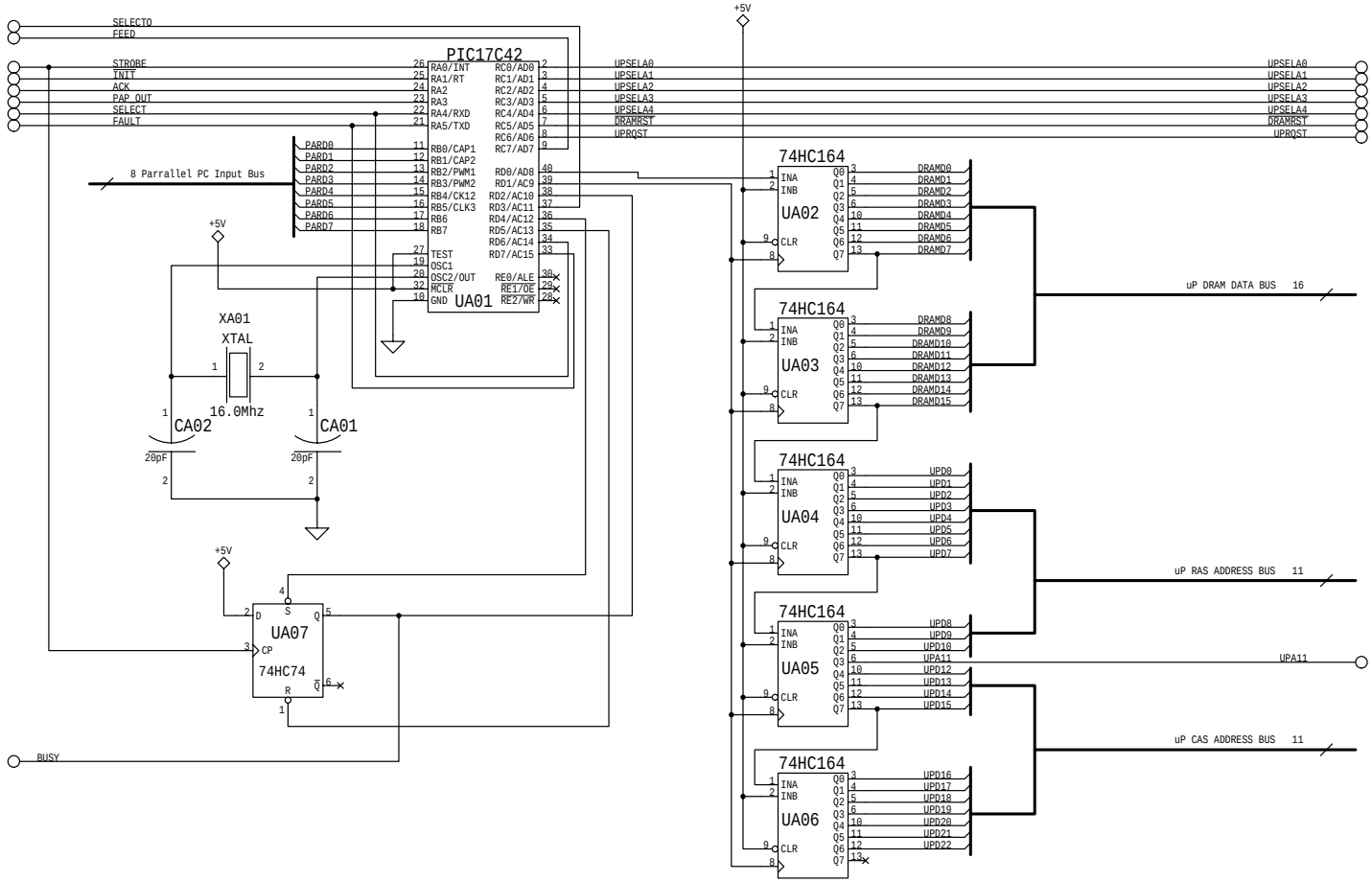
REVISIONS				
ZONE	REV	DESCRIPTION	DATE	APPROVED



CONTRACT NO. TBD	UCLA - IGPP
APPROVALS / DATE	TITLE: HESSI IMAGE GSE
DRAWN DAVID PIERCE 9-12-98	
CHECKED	SIZE B / FSCM NO. / DWG. NO. HESSI109.SCH / REV.
	SCALE / FNAME HESSI109.SCH / SHEET 9 OF 11

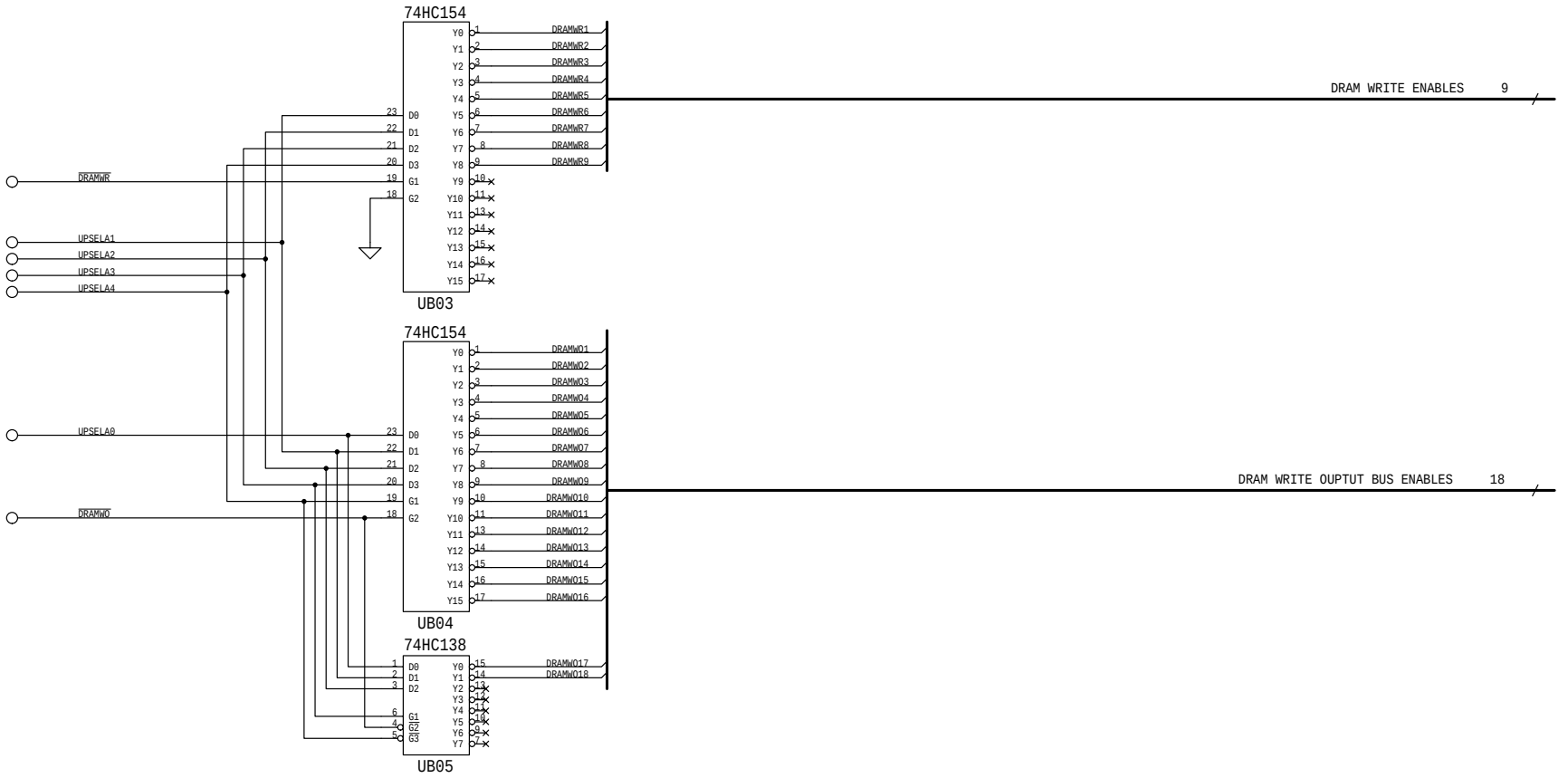
DWG. NO. HESSI10 SH 10 REV.

REVISIONS				
ZONE	REV	DESCRIPTION	DATE	APPROVED



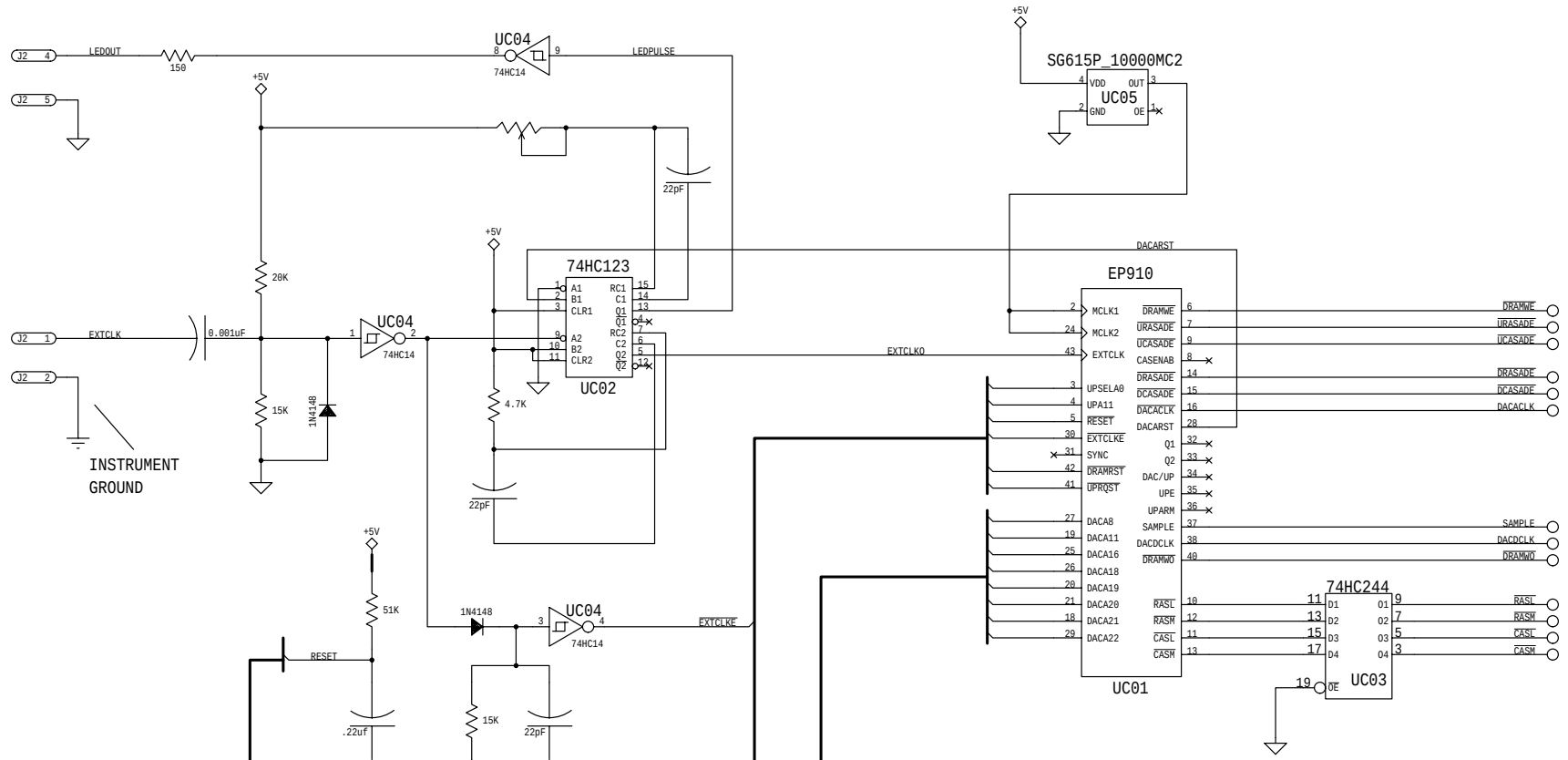
CONTRACT NO. TBD		UCLA - IGPP	
APPROVALS	DATE	TITLE	
DRAWN DAVID PIERCE	9-15-98	HESSI IMAGE GSE	
CHECKED		SIZE B / FSCM NO. / DWG. NO. HESSI110.SCH	REV.
		SCALE / FNAME HESSI110.SCH	SHEET 10 OF 11

REVISIONS				
ZONE	REV	DESCRIPTION	DATE	APPROVED



CONTRACT NO. TBD		UCLA - IGPP	
APPROVALS	DATE	TITLE HESSI IMAGE GSE	
DRAWN DAVID PIERCE	4-15-99		
CHECKED		SIZE B / FSCM NO. / DWG. NO. HESSI111.SCH	REV.
		SCALE / FNAME HESSI111.SCH	SHEET 11 OF 11

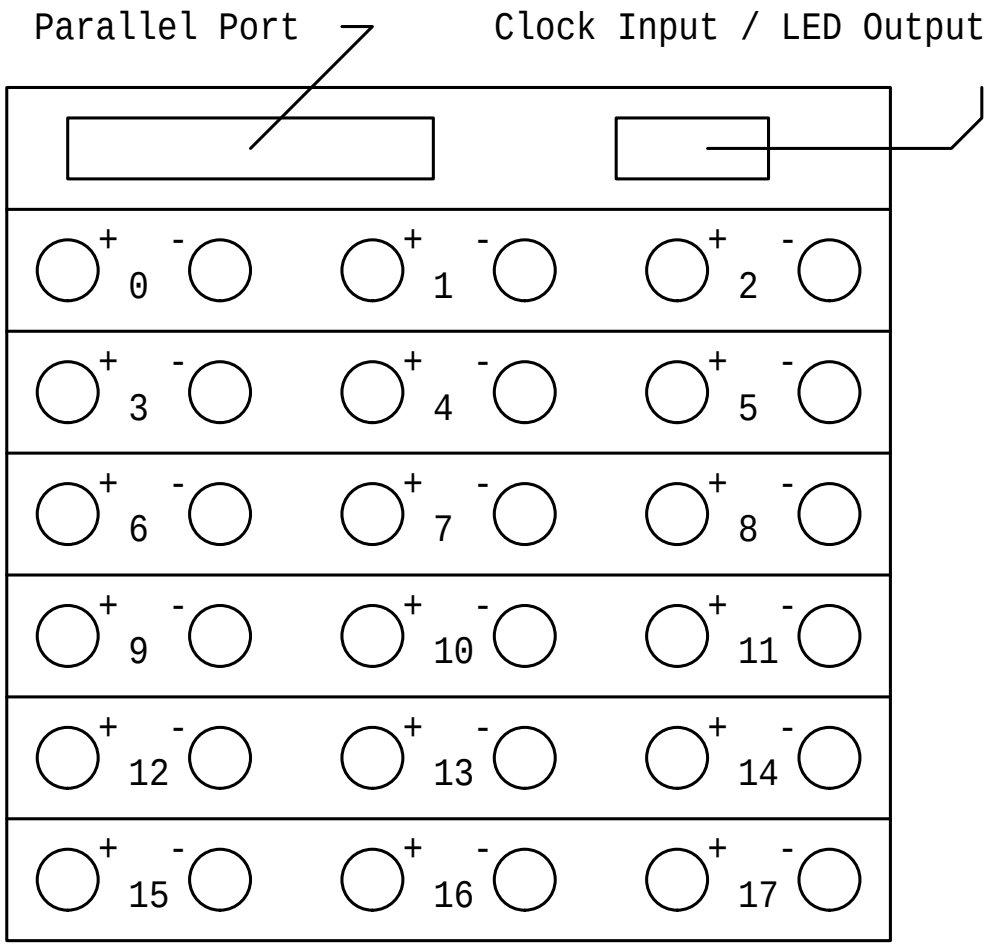
REVISIONS				
ZONE	REV	DESCRIPTION	DATE	APPROVED



CONTRACT NO. TBD	UCLA - IGPP
APPROVALS / DATE	TITLE: HESSI IMAGE GSE
DRAWN DAVID PIERCE 4-15-99	
CHECKED	
SIZE B / FSCM NO. / DWG. NO. HESSI112.SCH / REV.	
SCALE / FNAME HESSI112.SCH / SHEET 12 OF 13	

DWG. NO.	SH	REV.
	1	

REVISIONS				
ZONE	REV	DESCRIPTION	DATE	APPROVED



1.) Always use Parallel Port adaptor provided.

2.) 9Pin Port Connections:

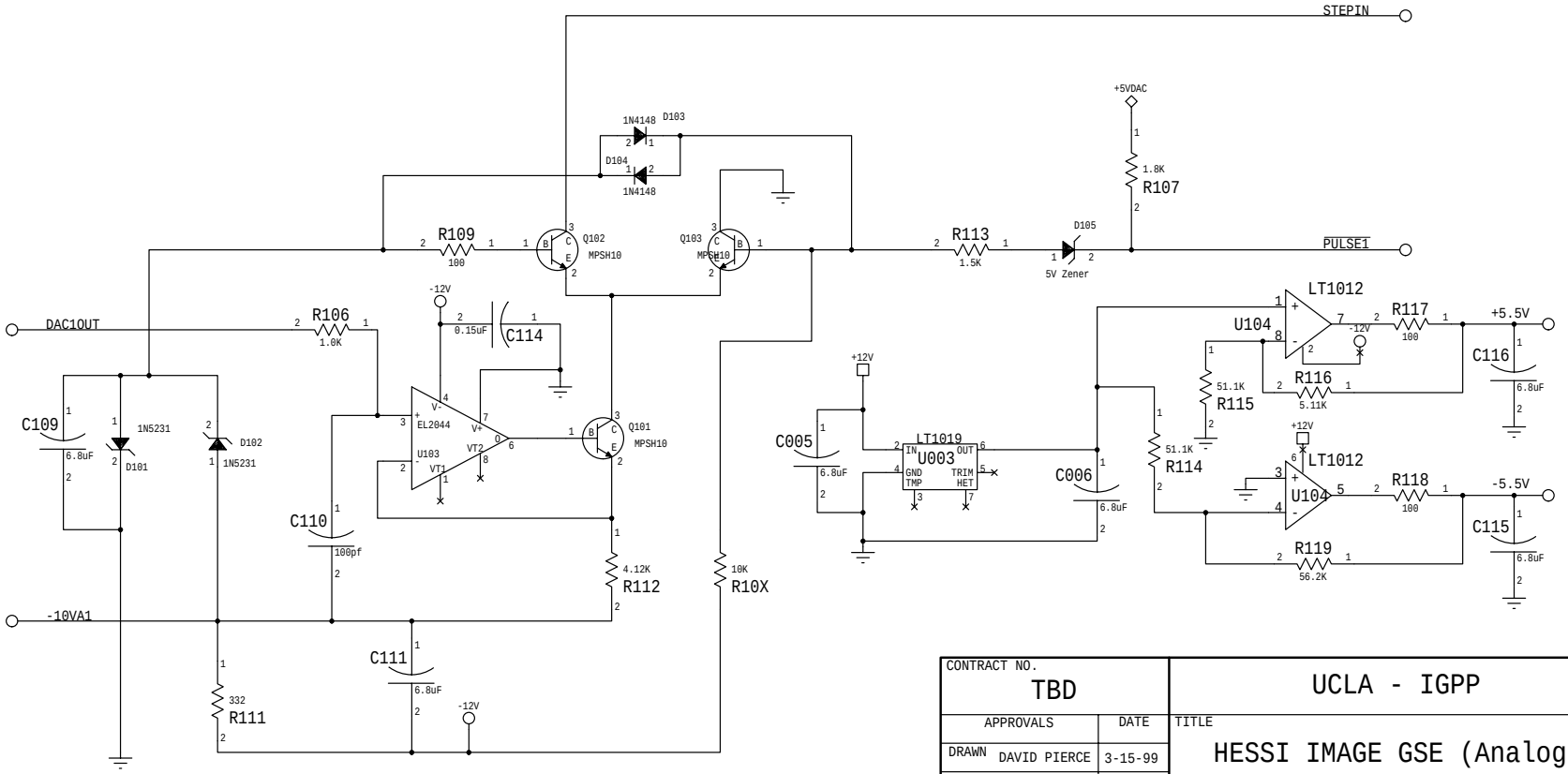
PIn 1	External Clock
Pin 7	Led Anode
Pin 9	Led Cathode

CONTRACT NO.		UCLA - IGPP
APPROVALS	DATE	TITLE: Connector Layout
DRAWN DAVID PIERCE	3/01/99	
CHECKED		SIZE B | FSCM NO. | DWG. NO. | REV.
		SCALE | FNAME | SHEET 1

DWG. NO.	HESSI20	SH	1	REV.

REVISIONS				
ZONE	REV	DESCRIPTION	DATE	APPROVED

1 of 3 sections per board.

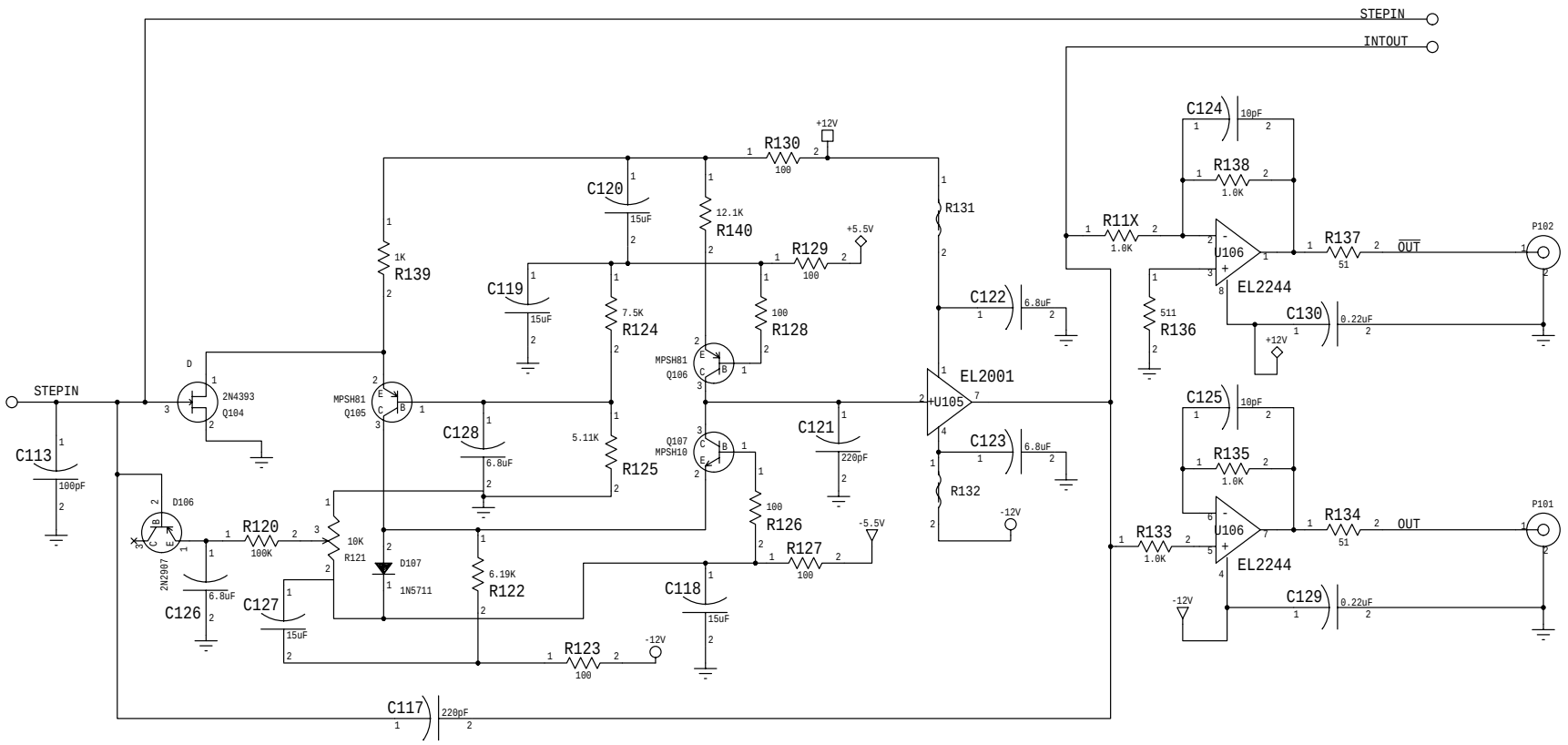


CONTRACT NO.		UCLA - IGPP	
TBD			
APPROVALS	DATE	TITLE	
DRAWN DAVID PIERCE	3-15-99	HESSI IMAGE GSE (Analog)	
CHECKED		SIZE B / FSCM NO. / DWG. NO. HESSI201.SCH	REV.
		SCALE / FNAME HESSI101.SCH	SHEET 1 OF 5

DWG. NO.	HESSI20	SH	2	REV.

1 of 3 sections per board

REVISIONS				
ZONE	REV	DESCRIPTION	DATE	APPROVED

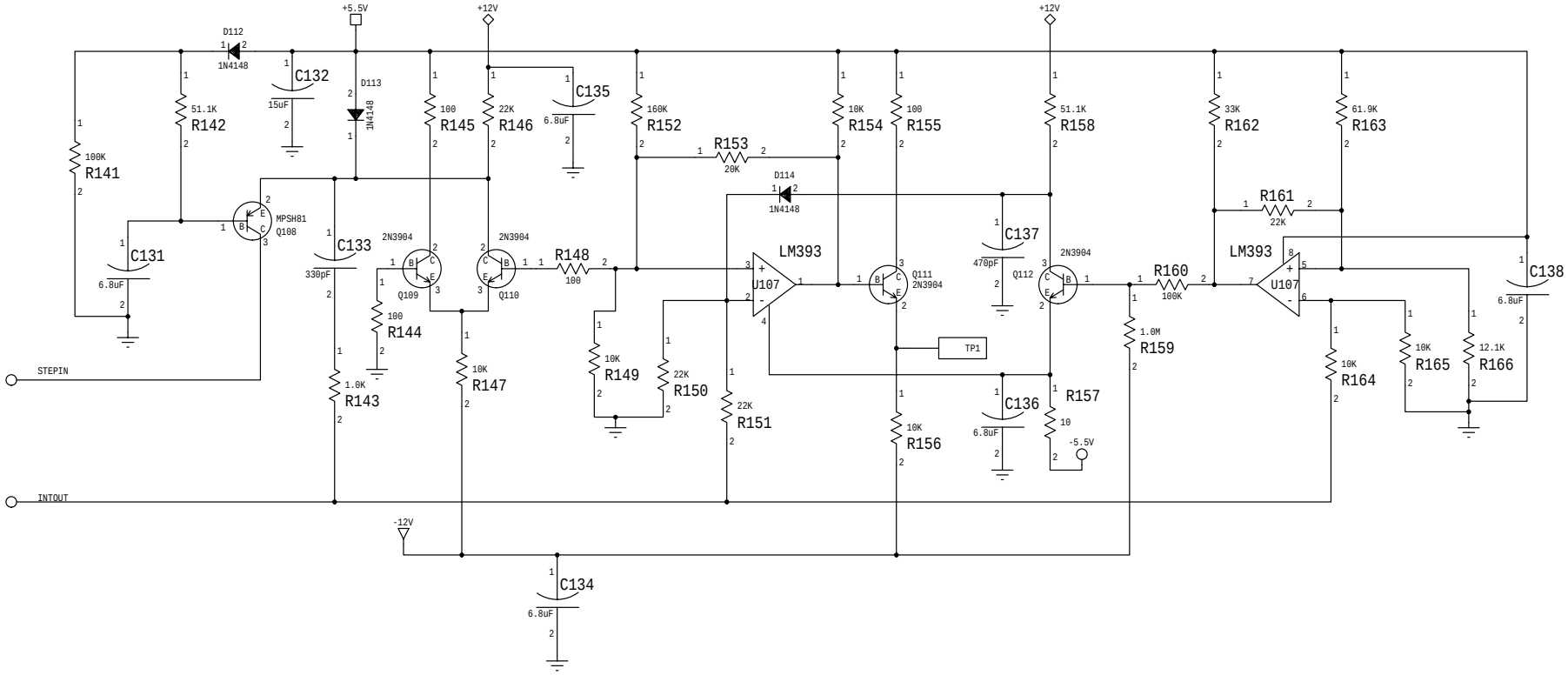


CONTRACT NO.		UCLA - IGPP	
TBD			
APPROVALS	DATE	TITLE	
DRAWN DAVID PIERCE	3-15-99	HESSI IMAGE GSE (INT)	
CHECKED		SIZE B / FSCM NO. / DWG. NO. HESSI202.SCH	REV.
		SCALE / FNAME HESSI101.SCH	SHEET 2 OF 5

DWG. NO. HESSI20 | SH 3 | REV.

REVISIONS				
ZONE	REV	DESCRIPTION	DATE	APPROVED

1 of 3 sections per board

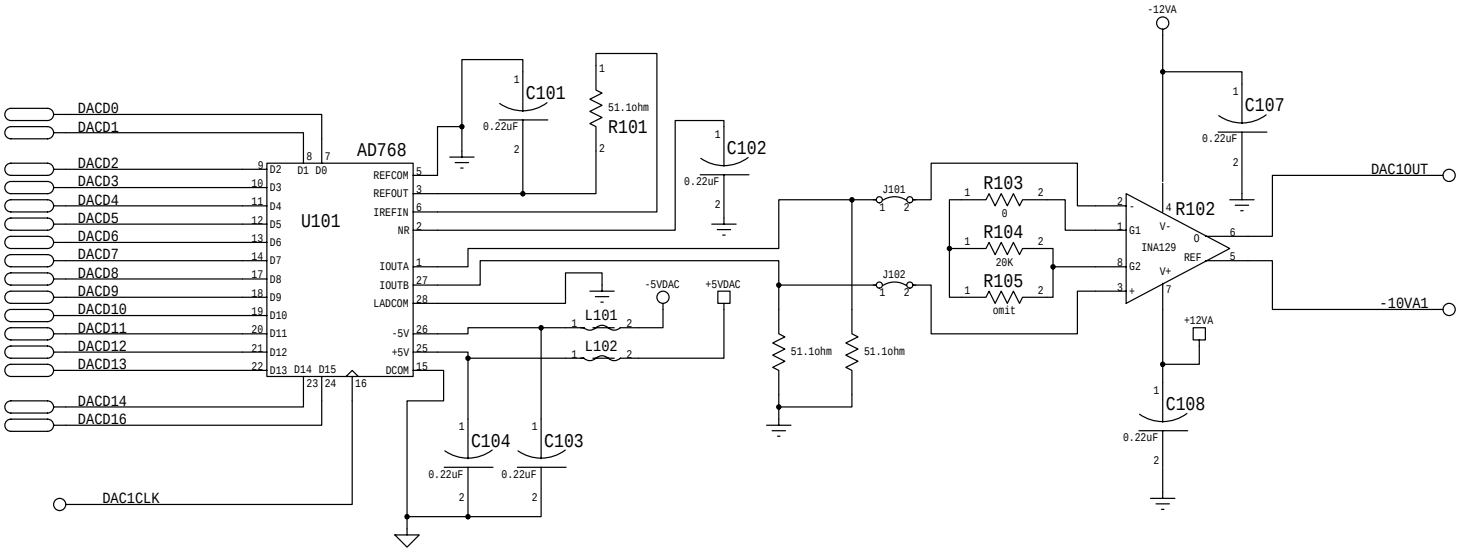


CONTRACT NO. TBD	UCLA - IGPP
APPROVALS / DATE	TITLE: HESSI IMAGE GSE (RESET)
DRAWN DAVID PIERCE 3-15-99	
CHECKED	
SIZE B / FSCM NO. / DWG. NO. HESSI203.SCH / REV.	
SCALE / FNAME HESSI103.SCH / SHEET 3 OF 5	

DWG. NO.	HESSI20	SH	3	REV.

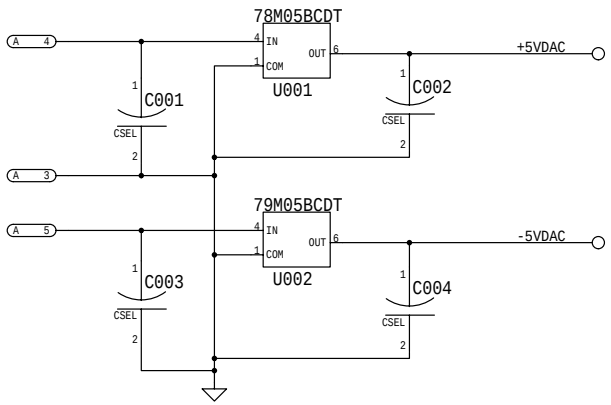
REVISIONS				
ZONE	REV	DESCRIPTION	DATE	APPROVED

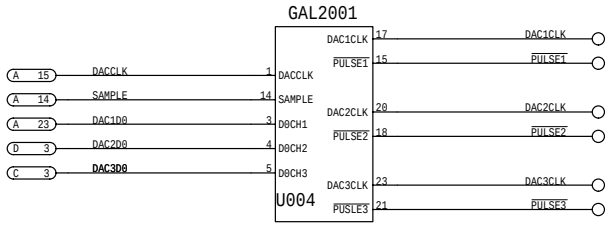
1 of 3 sections per board



CONTRACT NO.		UCLA - IGPP	
TBD			
APPROVALS	DATE	TITLE	
DRAWN DAVID PIERCE	4-15-99	HESSI IMAGE GSE (RESET)	
CHECKED		SIZE B / FSCM NO. / DWG. NO. HESSI203.SCH	REV.
		SCALE / FNAME HESSI103.SCH	SHEET 4 OF 5

REVISIONS				
ZONE	REV	DESCRIPTION	DATE	APPROVED





CONTRACT NO. TBD		UCLA - IGPP	
APPROVALS	DATE	TITLE	
DRAWN DAVID PIERCE	4-15-99	HESSI IMAGE GSE (RESET)	
CHECKED		SIZE B / FSCM NO. / DWG. NO. HESSI203.SCH	REV.
		SCALE / FNAME HESSI103.SCH	SHEET 5 OF 5