SM\_take\_tb

DAQ calls

76 #define DAQmxErrChk(functionCall) if( DAQmxFailed(error=(functionCall)) ) goto Error; else

1470

DAQmxErrChk(DAQmxCreateTask("", &taskHandle\_clk));

DAQmxErrChk(DAQmxCreateCOPulseChanTime(taskHandle\_clk, "Dev1/ctr0", "", DAQmx\_Val\_Seconds, DAQmx\_Val\_Low, 0.00, 0.50 / output\_rate, 0.50 / output\_rate));

DAQmxErrChk(DAQmxCfgImplicitTiming(taskHandle\_clk, DAQmx\_Val\_ContSamps, output\_sample));

DAQmxErrChk(DAQmxCreateTask("", &taskHandle\_out));

DAQmxErrChk(DAQmxCreateDOChan(taskHandle\_out, "Dev1/port0/line0:2", "", DAQmx\_Val\_ChanForAllLines));

DAQmxErrChk(DAQmxCfgSampClkTiming(taskHandle\_out, "/Dev1/PFI12", output\_rate, DAQmx\_Val\_Rising, DAQmx\_Val\_FiniteSamps, output\_sample)); //P

if (ExtTriggerFlag)

DAQmxErrChk(DAQmxCfgDigEdgeStartTrig(taskHandle\_clk, "/Dev1/PFI1", DAQmx\_Val\_Rising));

1491

DAQmxErrChk(DAQmxCreateTask("", &taskHandle\_AO));

DAQmxErrChk(DAQmxCreateAOVoltageChan(taskHandle\_AO, "Dev1/ao0", "", 0.0, 5.0, DAQmx\_Val\_Volts, NULL)); // for 6341 - 2 analog lines

DAQmxErrChk(DAQmxCfgSampClkTiming(taskHandle\_AO, "/Dev1/PFI0", sample\_chan\_rate, DAQmx\_Val\_Rising, DAQmx\_Val\_FiniteSamps, AO\_frames));

1545

DAQmxErrChk(DAQmxCreateTask("", &taskHandle\_AO));

if (MNC\_flag == 2)

DAQmxErrChk(DAQmxCreateAOVoltageChan(taskHandle\_AO, "Dev1/ao0:1", "", 0.0, 5.0, DAQmx\_Val\_Volts, NULL)); // for 6341 - 2 analog lines

else

DAQmxErrChk(DAQmxCreateAOVoltageChan(taskHandle\_AO, "Dev1/ao0:2", "", 0.0, 5.0, DAQmx\_Val\_Volts, NULL));

DAQmxErrChk(DAQmxCfgSampClkTiming(taskHandle\_AO, "/Dev1/PFI0", sample\_chan\_rate, DAQmx\_Val\_Rising, DAQmx\_Val\_FiniteSamps, AO\_frames));

1553

// config inputs and trigger

DAQmxErrChk(DAQmxCreateTask("", &taskHandle\_in));

if (numBNC\_chan == 8)

DAQmxErrChk(DAQmxCreateAIVoltageChan(taskHandle\_in, "Dev1/ai0:7", "", DAQmx\_Val\_RSE, -10.0, 10.0, DAQmx\_Val\_Volts, NULL));

else

DAQmxErrChk(DAQmxCreateAIVoltageChan(taskHandle\_in, "Dev1/ai0:3", "", DAQmx\_Val\_RSE, -10.0, 10.0, DAQmx\_Val\_Volts, NULL));

if (BNC\_ratio > 1) {

DAQmxErrChk(DAQmxCfgSampClkTiming(taskHandle\_in, "/Dev1/PFI12", sample\_chan\_rate, DAQmx\_Val\_Falling, DAQmx\_Val\_FiniteSamps, BNC\_num\_frames)); //frame-by-frame clock trigger

DAQmxErrChk(DAQmxCfgDigEdgeStartTrig(taskHandle\_in, "/Dev1/PFI0", DAQmx\_Val\_Rising));

}

else

DAQmxErrChk(DAQmxCfgSampClkTiming(taskHandle\_in, "/Dev1/PFI0", sample\_chan\_rate, DAQmx\_Val\_Rising, DAQmx\_Val\_FiniteSamps, BNC\_num\_frames)); //frame-by-frame clock trigger

if (NI\_flag != 1)

DAQmxErrChk(DAQmxCfgDigEdgeStartTrig(taskHandle\_in, "/Dev1/PFI2", DAQmx\_Val\_Rising));

DAQmxErrChk(DAQmxRegisterDoneEvent(taskHandle\_clk, 0, DoneCallback, NULL));

DAQmxErrChk(DAQmxWriteDigitalU32(taskHandle\_out, output\_sample, 0, 10.0, DAQmx\_Val\_GroupByChannel, output\_data, &total\_written, NULL));

if (MNC\_flag || ACell\_flag)

DAQmxErrChk(DAQmxWriteAnalogF64(taskHandle\_AO, images + 1, 0, 10.0, DAQmx\_Val\_GroupByChannel, AO\_data, NULL, NULL));

DAQmxErrChk(DAQmxStartTask(taskHandle\_in));

if (MNC\_flag || ACell\_flag)

DAQmxErrChk(DAQmxStartTask(taskHandle\_AO));

DAQmxErrChk(DAQmxStartTask(taskHandle\_out));

DAQmxErrChk(DAQmxStartTask(taskHandle\_clk));

1664 & 1715

DAQmxReadAnalogF64(taskHandle\_in, factor\*BNC\_ratio\*(j + 1) - total\_read, 5.0, DAQmx\_Val\_GroupByScanNumber, NI\_pt, (factor\*BNC\_ratio\*(j + 1) - total\_read + 1)\*numBNC\_chan, &read, NULL);

else

DAQmxReadAnalogF64(taskHandle\_in, factor\*BNC\_ratio\*(j + 1) - total\_read, 0.0, DAQmx\_Val\_GroupByScanNumber, NI\_pt, (factor\*BNC\_ratio\*(j + 1) - total\_read)\*numBNC\_chan, &read, NULL);

1740-1760

if (taskHandle\_AO) {

DAQmxStopTask(taskHandle\_AO);

DAQmxClearTask(taskHandle\_AO);

2991

int32 CVICALLBACK DoneCallback(TaskHandle taskHandle, int32 status, void \*callbackData)

3001

if (DAQmxFailed(error)) {

DAQmxGetExtendedErrorInfo(errBuff, 2048);

DAQmxClearTask(taskHandle);

Pdv calls

22 extern PdvDev \*pdv\_pt[MAXCHANNELS], \*pdv\_p, \*pcd\_p;

127 file\_width = image\_width = (frame\_width = pdv\_get\_width(pdv\_pt[0]));

128 file\_height = image\_height = (frame\_height = pdv\_get\_height(pdv\_pt[0]));

223 for (int i = 0; i < numChannels; ++i)

224 pdv\_setsize(pdv\_pt[i], config\_num\_col, config\_num\_row\*live\_factor);

403 pdv\_flush\_fifo(pdv\_pt[i]);

404 pdv\_multibuf(pdv\_pt[i], 4); //has to be 4

418 pdv\_start\_images(pdv\_pt[thread\_id], 4);

419 image\_ptr = pdv\_wait\_image\_timed\_raw(pdv\_pt[thread\_id],(unsigned int \*)&times[thread\_id][0],1);

420 pdv\_start\_image(pdv\_pt[thread\_id]);

429 image\_ptr = pdv\_wait\_image\_raw(pdv\_pt[thread\_id]);

430 pdv\_start\_image(pdv\_pt[thread\_id]);

501 pdv\_start\_images(pdv\_pt[0], loops + (factor > 1 ? 1 : 0));

502 for (int j = 0; j < loops; j++) {

503 image\_ptr = pdv\_wait\_image\_raw(pdv\_pt[0]);

504 memcpy(data\_ptr[0], image\_ptr, frame\_length);

505 data\_ptr[0] += image\_length;

940 for (i = 0; i < numChannels; ++i)

941 pdv\_start\_image(pdv\_pt[i]);

1020 timeouts[i] = pdv\_timeouts(pdv\_pt[i]);

1264 for (i = 0; i < numChannels; ++i)

1265 pdv\_start\_images(pdv\_pt[i], loops + 1);

Edt calls

386 edt\_reg\_write(pdv\_p, SSD16\_CHEN, 0x0);

edt\_flush\_channel(pdv\_p, 0);

if (edt\_configure\_ring\_buffers(pdv\_p, bufsize, numbufs, EDT\_READ, NULL) != -1)

{

edt\_start\_buffers(pdv\_p, numbufs);

edt\_reg\_write(pdv\_p, SSD16\_CHEN, 0x0001);

image\_ptr = edt\_wait\_for\_buffers(pdv\_p, 1);

memcpy(data\_ptr[0], image\_ptr, frame\_length);

data\_ptr[0] += image\_length;

396 }

398 MessageBox(main\_window\_handle, "edt\_configure\_ring\_buffers failed", "message", MB\_OK);

928 if ((overrun[thread\_id] = (edt\_reg\_read(pdv\_pt[thread\_id], PDV\_STAT) & PDV\_OVERRUN)))

++overruns[thread\_id];

1593 edt\_reg\_write(pdv\_p, SSD16\_CHEN, 0x0);

edt\_flush\_channel(pdv\_p, 0);

if (edt\_configure\_ring\_buffers(pdv\_p, bufsize, numbufs, EDT\_READ, NULL) != -1)

{

edt\_start\_buffers(pdv\_p, loops);

edt\_reg\_write(pdv\_p, SSD16\_CHEN, 0x0001);

for (j = 0; (j < loops || loops == 0); j++)

{

image\_ptr = edt\_wait\_for\_buffers(pdv\_p, 1);