

inputCmd = cell(13,2);

inputCmd{1,1}=''; inputCmd{1,2}=''; % Geo FileName

inputCmd{2,1}=''; inputCmd{2,2}=''; % Tiss FileName

inputCmd{3,1}=''; inputCmd{3,2}=''; % dB0 FileName

inputCmd{4,1}=''; inputCmd{4,2}=''; % Sequence FileName

inputCmd{5,1}=''; inputCmd{5,2}=''; % B1+ FileName

inputCmd{6,1}=''; inputCmd{6,2}=''; % E1+ FileName

inputCmd{7,1}=''; inputCmd{7,2}=''; % B1- FileName

inputCmd{8,1}=''; inputCmd{8,2}=''; % E1- FileName

inputCmd{9,1}=''; inputCmd{9,2}=''; % Gx FileName

inputCmd{10,1}=''; inputCmd{10,2}=''; % Gy FileName

inputCmd{11,1}=''; inputCmd{11,2}=''; % Gz FileName

inputCmd{12,1}=''; inputCmd{12,2}=''; % KSpace FileName

inputCmd{13,1}=''; inputCmd{13,2}=''; % KMap FileName

inputCmd{14,1}=''; inputCmd{14,2}=''; % Noise FileName

inputCmd{15,1}=''; inputCmd{15,2}=''; % SAR FileName

Tx.Num = 1;

Tx.MagMat = ones(Tx.Num,1);

Tx.PhsMat = zeros(Tx.Num,1);

Rx.Num = 1;

function StudyNew\_Callback(source,eventdata,flag)

inputCmd{12,1}=[addr,'\Protocols\',tempStr,'\']; inputCmd{12,2}=[tempStr,'.ksig']; % KSpace FileName

inputCmd{13,1}=[addr,'\Protocols\',tempStr,'\']; inputCmd{13,2}=[tempStr,'.ktrj']; % KMap FileName

inputCmd{14,1}=[addr,'\Protocols\',tempStr,'\']; inputCmd{14,2}=[tempStr,'.nois']; % Noise FileName

inputCmd{15,1}=[addr,'\Protocols\',tempStr,'\']; inputCmd{15,2}=[tempStr,'.sar']; % SAR FileName

function StudyLoad\_Callback(source,eventdata)

inputCmd{12,1}=[addr,'\Protocols\',tempStr,'\']; inputCmd{12,2}=[tempStr,'.ksig']; % KSpace FileName

inputCmd{13,1}=[addr,'\Protocols\',tempStr,'\']; inputCmd{13,2}=[tempStr,'.ktrj']; % KMap FileName

inputCmd{14,1}=[addr,'\Protocols\',tempStr,'\']; inputCmd{14,2}=[tempStr,'.nois']; % Noise FileName

inputCmd{15,1}=[addr,'\Protocols\',tempStr,'\']; inputCmd{15,2}=[tempStr,'.sar']; % SAR FileName

Tx.Num %Number of Transmit Arrays

%% --- Executes on hSeqType\_Menu.

function SeqType\_Callback(source,eventdata,flag)

if strcmp(flag,'')==1

updateSeq();

else

tempStr = get(hSeqType,'String');

temp = find(strcmp(tempStr,flag));

set(hSeqType,'Value',temp);

clear tempStr temp;

end;

end

%% --- Executes on hSeqGen\_Button.

function SeqGen\_Callback(source,eventdata) %#ok<INUSD,INUSD>

global interSeqData inputInfo;

%cd(addr);

% tempAddr = pwd;

% path(path,[pwd,'\SeqDev\Functions\']);

% path(path,[pwd,'\SeqDev\Bin\']);

RF.FA = str2double(get(hRF\_FA,'String')); % Degrees

[MagMat, PhsMat,index] = TxMagPhsGUI(Tx.Num,Tx.MagMat,Tx.PhsMat,RF.FA);

if index ~= 0

Tx.MagMat = MagMat;

Tx.PhsMat = PhsMat;

set(hTx,'Enable','off');

set(hRx,'Enable','off');

%cd([tempAddr '\SeqDev\User']);

TE = str2double(get(hTE,'String')); %ms

TR = str2double(get(hTR,'String')); %ms

TI.duration = str2double(get(hTI,'string')); %ms

MatSize.value = get(hMatSize,'value');

FOV.value = get(hFOV,'value'); %mm

RF.Duration = str2double(get(hRF\_Duration,'String')); %ms

RF.Num = str2double(get(hRF\_Num,'String'));

RF.Shape = get(hRF\_Shape,'Value');

ADC.BW = str2double(get(hADC\_Freq,'String')); %kHz

GradOn.ss = get(hSS,'Value'); %gradients are specified in mT/m, but here it is a on/off flag.

GradOn.fe = get(hFE,'Value');

GradOn.pe = get(hPE,'Value');

temp = get(hPEDirectMenu,'String'); peDirectoin = temp(get(hPEDirectMenu,'Value')); clear temp;

dummy = str2double(get(hDummyNum,'String'));

slice.TH = str2double(get(hGeoSliceTH,'String'))/1000;

switch get(hOrientationMenu,'Value')

case 1

slice.offset = (inputInfo.currentZ - inputInfo.ctrZ)\*inputInfo.widZ/1000;

case 2

slice.offset = (inputInfo.currentX - inputInfo.ctrX)\*inputInfo.widX/1000;

case 3

slice.offset = (inputInfo.currentY - inputInfo.ctrY)\*inputInfo.widY/1000;

end;

tempNum = get(hSeqType,'Value');

tempStr = get(hSeqType,'String');

MatSize.handle = hMatSize;

FOV.handle = hFOV;

interSeqData = SeqInterfaceOut(cell2mat(tempStr(tempNum)),...

tempNum,TE,TR,MatSize,hMatSize\_Pop,FOV,hFOV\_Pop,RF,ADC,GradOn,dummy,slice,TI);

set(hSeqPlot,'Enable','on');

%cd([tempAddr '\SeqDev\Bin']);

interSeqData = checkSeqProt(interSeqData);

finalSeqData = combineSeq(interSeqData,get(hOrientationMenu,'Value'),peDirectoin,dummy); %

expSeq(finalSeqData,dummy,Tx,Rx); %

% inputCmd{5,1}=''; inputCmd{5,2}=''; % B1+ FileName

% inputCmd{6,1}=''; inputCmd{6,2}=''; % E1+ FileName

% inputCmd{7,1}=''; inputCmd{7,2}=''; % B1- FileName

% inputCmd{8,1}=''; inputCmd{8,2}=''; % E1- FileName

if ~isempty(inputCmd{1,1}) && ~isempty(inputCmd{2,1})

set(hRun,'Enable','on');

end;

set(hExpSeqEdit,'String',get(hStudyName,'String'),'FontAngle','normal');

inputCmd{4,2} = [get(hStudyName,'String'),'.seqn'];

inputCmd{4,1} = [addr,'\Protocols\',get(hStudyName,'String'),'\']; %%---

set(hSeqPlot,'Enable','on');

set(hMode,'Enable','on');

tempMode = get(hMode,'Value');

switch tempMode

case {1,2}

set(hB0Edit,'Enable','on');

set(hExpDB0,'Enable','on');

set(hExpB1m,'Enable','on');

% tempStr = get(hExpB1mEdit,'String');

% set(hExpB1mEdit,'String','Homogeneous or B1- Filename(s)','FontAngle','italic');

% clear inputCmd{7,1} inputCmd{7,2};

set(hExpGx,'Enable','on');

set(hExpGy,'Enable','on');

set(hExpGz,'Enable','on');

set(hExpB1p,'Enable','on');

set(hExpB1m,'Enable','on');

% set(hExpB1pEdit,'String','Homogeneous or B1+ Filename(s)','FontAngle','italic');

% clear inputCmd{5,1} inputCmd{5,2};

set(hExpE1p,'Enable','off');

% set(hExpE1pEdit,'String','E1+ Filename(s)','FontAngle','italic');

% clear inputCmd{6,1} inputCmd{6,2};

set(hExpE1m,'Enable','off');

% set(hExpE1mEdit,'String','E1- Filename(s)','FontAngle','italic');

% clear inputCmd{6,1} inputCmd{6,2};

set(hNumIso,'Enable','on');

set(hThread,'Enable','on');

case 3

set(hB0Edit,'Enable','off');

set(hExpDB0,'Enable','off');

set(hExpGx,'Enable','off');

set(hExpGy,'Enable','off');

set(hExpGz,'Enable','off');

set(hExpB1p,'Enable','on');

set(hExpB1m,'Enable','off');

set(hExpE1p,'Enable','on');

% set(hExpE1pEdit,'String','E1+ Filename(s)','FontAngle','italic');

% clear inputCmd{6,1} inputCmd{6,2};

set(hExpE1m,'Enable','off');

% set(hExpE1mEdit,'String','E1- Filename(s)','FontAngle','italic');

% clear inputCmd{8,1} inputCmd{8,2};

set(hNumIso,'Enable','off');

set(hThread,'Enable','on');

case 4

set(hB0Edit,'Enable','off');

set(hExpDB0,'Enable','off');

set(hExpB1m,'Enable','off');

% set(hExpB1mEdit,'String','Homogeneous or B1- Filename(s)','FontAngle','italic');

% clear inputCmd{7,1} inputCmd{7,2};

set(hExpGx,'Enable','off');

set(hExpGy,'Enable','off');

set(hExpGz,'Enable','off');

set(hExpB1p,'Enable','off');

% set(hExpB1pEdit,'String','Homogeneous or B1+ Filename(s)','FontAngle','italic');

% clear inputCmd{5,1} inputCmd{5,2};

set(hExpE1p,'Enable','off');

% set(hExpE1pEdit,'String','E1+ Filename(s)','FontAngle','italic');

% clear inputCmd{6,1} inputCmd{6,2};

set(hExpE1m,'Enable','on');

% set(hExpE1mEdit,'string','E1- Filename(s)','FontAngle','italic');

% clear inputCmd{8,1} inputCmd{8,2};

set(hNumIso,'Enable','off');

set(hThread,'Enable','off');

end;

if ~isempty(inputCmd{1,1}) && ~isempty(inputCmd{2,1})

set(hRun,'Enable','on');

end;

end

end

' "NumIsoX=',num2str(inputInfo.NumIso(1)),'"',...

' "NumIsoY=',num2str(inputInfo.NumIso(2)),'"',...

' "NumIsoZ=',num2str(inputInfo.NumIso(3)),'"',...

' "Thread=',get(hThread,'String'),'"',...

' "B0=',get(hB0Edit,'String'),'"',...

' "GeometryFile=',[inputCmd{1,1},inputCmd{1,2}],'"',...

' "TissueTypeFile=',[inputCmd{2,1},inputCmd{2,2}],'"',...

' "SequenceFile=',[inputCmd{4,1},inputCmd{4,2}],'"'];

if ~isempty(inputCmd{3,1})

runScriptSignal = [runScriptSignal, ' "DelB0File=',[inputCmd{3,1},inputCmd{3,2}],'"'];

end;

if ~isempty(inputCmd{5,1})

temp1 = size(inputCmd{5,2});

for n=1:temp1(2)

runScriptSignal = [runScriptSignal, ' "B1PlusFile=',[inputCmd{5,1}{n},inputCmd{5,2}{n}],'"']; %#ok<AGROW>

end;

clear temp1;

end;

if ~isempty(inputCmd{7,1})

temp1 = size(inputCmd{7,2});

for n=1:temp1(2)

runScriptSignal = [runScriptSignal, ' "B1MinsFile=',[inputCmd{7,1}{n},inputCmd{7,2}{n}],'"']; %#ok<AGROW>

end;

clear temp1;

end;

if ~isempty(inputCmd{9,1})

runScriptSignal = [runScriptSignal, ' "DelGxFile=',[inputCmd{9,1},inputCmd{9,2}],'"'];

end;

if ~isempty(inputCmd{10,1})

runScriptSignal = [runScriptSignal, ' "DelGyFile=',[inputCmd{10,1},inputCmd{10,2}],'"'];

end;

if ~isempty(inputCmd{11,1})

runScriptSignal = [runScriptSignal, ' "DelGzFile=',[inputCmd{11,1},inputCmd{11,2}],'"'];

end;

if ~isempty(inputCmd{12,1})

runScriptSignal = [runScriptSignal, ' "KSpaceFile=',[inputCmd{12,1},inputCmd{12,2}],'"'];

else

errordlg('Please Specify kSpace Filename.','modal');

return

end;

if ~isempty(inputCmd{13,1})

runScriptSignal = [runScriptSignal, ' "KMapFile=',[inputCmd{13,1},inputCmd{13,2}],'"'];

else

errordlg('Please Specify kMap Filename.','modal');

return

end;

runScriptSignal = [runScriptSignal, ...

' "xMin=',get(hXMin,'String'),'"',...

' "xMax=',get(hXMax,'String'),'"',...

' "yMin=',get(hYMin,'String'),'"',...

' "yMax=',get(hYMax,'String'),'"',...

' "zMin=',get(hZMin,'String'),'"',...

' "zMax=',get(hZMax,'String'),'"',...

' "xCtr=',get(hXCtr,'String'),'"',...

' "yCtr=',get(hYCtr,'String'),'"',...

' "zCtr=',get(hZCtr,'String'),'"',...

];

case 3

runScriptSAR = ['"',addr,'\Engine\NutateSAR"',...

' "Thread=',get(hThread,'String'),'"',...

' "GeometryFile=',[inputCmd{1,1},inputCmd{1,2}],'"',...

' "TissueTypeFile=',[inputCmd{2,1},inputCmd{2,2}],'"',...

' "SequenceFile=',[inputCmd{4,1},inputCmd{4,2}],'"'];

if ~isempty(inputCmd{5,1})

temp1 = size(inputCmd{5,2});

for n=1:temp1(2)

runScriptSAR = [runScriptSAR, ' "B1PlusFile=',[inputCmd{5,1}{n},inputCmd{5,2}{n}],'"']; %#ok<AGROW>

end;

clear temp1;

else

errordlg('Please Specify B1+ Filename(s).','modal');

return

end;

if ~isempty(inputCmd{6,1})

temp1 = size(inputCmd{6,2});

for n=1:temp1(2)

runScriptSAR = [runScriptSAR, ' "E1PlusFile=',[inputCmd{6,1}{n},inputCmd{6,2}{n}],'"']; %#ok<AGROW>

end;

clear temp1;

else

errordlg('Please Specify E1+ Filename(s).','modal');

return

end;

if ~isempty(inputCmd{7,1})

temp1 = size(inputCmd{7,2});

for n=1:temp1(2)

runScriptSAR = [runScriptSAR, ' "B1MinsFile=',[inputCmd{7,1}{n},inputCmd{7,2}{n}],'"']; %#ok<AGROW>

end;

clear temp1;

end;

if ~isempty(inputCmd{15,1})

runScriptSAR = [runScriptSAR, ' "SARFile=',[inputCmd{15,1},inputCmd{15,2}],'"'];

else

errordlg('Please Specify SAR Filename.','modal');

return

end;

runScriptSAR = [runScriptSAR, ...

' "xMin=',get(hXMin,'String'),'"',...

' "xMax=',get(hXMax,'String'),'"',...

' "yMin=',get(hYMin,'String'),'"',...

' "yMax=',get(hYMax,'String'),'"',...

' "zMin=',get(hZMin,'String'),'"',...

' "zMax=',get(hZMax,'String'),'"',...

];

case 4

runScriptNoise = ['"',addr,'\Engine\NutateNoise"',...

' "NEX=1"',...

' "GeometryFile=',[inputCmd{1,1},inputCmd{1,2}],'"',...

' "TissueTypeFile=',[inputCmd{2,1},inputCmd{2,2}],'"',...

' "SequenceFile=',[inputCmd{4,1},inputCmd{4,2}],'"'];

if ~isempty(inputCmd{8,1})

temp1 = size(inputCmd{8,2});

for n=1:temp1(2)

runScriptNoise = [runScriptNoise, ' "E1MinsFile=',[inputCmd{8,1}{n},inputCmd{8,2}{n}],'"']; %#ok<AGROW>

end;

clear temp1;

else

errordlg('Please Specify E1- Filename(s).','modal');

return

end;

if ~isempty(inputCmd{14,1})

runScriptNoise = [runScriptNoise, ' "KNoiseFile=',[inputCmd{14,1},inputCmd{14,2}],'"'];

else

errordlg('Please Specify Noise Filename.','modal');

return

end;

runScriptNoise = [runScriptNoise, ...

' "xMin=',get(hXMin,'String'),'"',...

' "xMax=',get(hXMax,'String'),'"',...

' "yMin=',get(hYMin,'String'),'"',...

' "yMax=',get(hYMax,'String'),'"',...

' "zMin=',get(hZMin,'String'),'"',...

' "zMax=',get(hZMax,'String'),'"',...

];

TE = str2double(get(hTE,'String'));

TR = str2double(get(hTR,'String'));

MatSize.value = get(hMatSize,'value');

FOV.value = get(hFOV,'value');

RF.Duration = str2double(get(hRF\_Duration,'String'));

if ((str2double(get(hRF\_Num,'String')) == 1) && ( get(hRF\_Shape,'Value') == 1))

set(hRF\_Num,'String','2');

end;

RF.Num = str2double(get(hRF\_Num,'String'));

RF.Shape = get(hRF\_Shape,'Value');

RF.FA = str2double(get(hRF\_FA,'String'));

ADC.BW = str2double(get(hADC\_Freq,'String'));

GradOn.ss = get(hSS,'Value');

GradOn.fe = get(hFE,'Value');

GradOn.pe = get(hPE,'Value');

dummy = str2double(get(hDummyNum,'String'));

%cd([tempAddr '\SeqDev\User']);

tempCell = SeqInterfaceIn;

set(hSeqType,'String',tempCell);

clear tempN tempM tempCell;

slice.TH = str2double(get(hGeoSliceTH,'String'))/1000;

switch get(hOrientationMenu,'Value')

case 1

slice.offset = (inputInfo.currentZ - inputInfo.ctrZ)\*inputInfo.widZ;

case 2

slice.offset = (inputInfo.currentX - inputInfo.ctrX)\*inputInfo.widX;

case 3

slice.offset = (inputInfo.currentY - inputInfo.ctrY)\*inputInfo.widY;

end;

tempNum = get(hSeqType,'Value');

tempStr = get(hSeqType,'String');

SeqInterfaceIn(tempNum,hTI);

TI.duration = str2double(get(hTI,'string'));

TI.handle = hTI;

MatSize.handle = hMatSize;

FOV.handle = hFOV;

[minTE,minTR] = SeqInterfaceOut(tempStr{tempNum},tempNum,TE,TR,MatSize,hMatSize\_Pop,FOV,hFOV\_Pop,RF,ADC,GradOn,dummy,slice,TI);

% displayImages(hImages,inputInfo,displayInfo,get(hOrientationMenu,'Value'),geoData);

if str2double(get(hTE,'String')) < minTE

set(hTE,'String',num2str(round(minTE\*10000)/10000));

end;

if str2double(get(hTR,'String')) < minTR

set(hTR,'String',num2str(round(minTR\*10000)/10000));

end;