MatlabTNG Software Environment for FACET-II HLA

7/27/2022

Glen White, SLAC

U.S. DEPARTMENT OF ENERGY | Stanford University

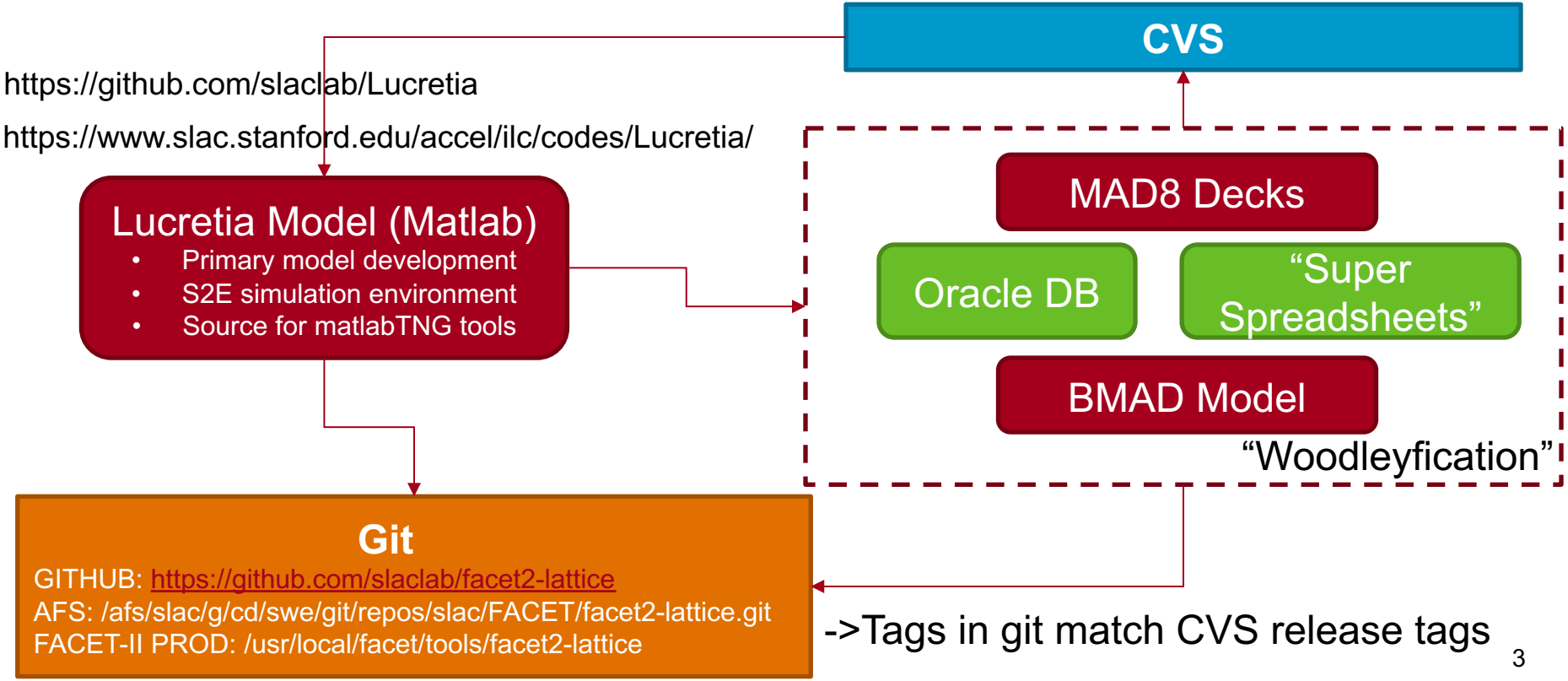
SLAC NATIONAL ACCELERATOR LABORATORY

Overview

(NB: The purpose of these slides is to provide an overview of "what is there" and give clues on how the software is used, this is not a comprehensive howto guide)

- FACET-II modeling workflow overview
- What is available?
  - Model and controls software, CVS & git repositories
  - New "matlabTNG" and legacy matlab s/w directories
- Summarize philosophy for developing & deploying new (FACET-II) apps and GUIs
- Examples using Lucretia model in live environment
- Overview of various "helper" tools that have been built up and may be of use

FACET-II Model Development Workflow



Model Git Repository

AT	e+ DR model
Data	Beam stay-clear files
GPT	e- Injector model
Lucretia	All e- & e+ lattices other than DR + S2E sim environment
MAD	All e- & e+ lattices
bmad	All e- & e+ lattices + e+ DR beam dynamics
distgen/models/f2e_inj	e- Injector beam definition files
impact/models/f2e_inj	e- Injector model

Publicly clone from:
https://github.com/slaclab/facet2-lattice

-> Also available on afs and FACET-II PROD

FACET-II Controls Software

On FACET-II prod (e.g. facet-srv01):

- **New matlab HLA environment (for matlab2020a+)**
  - /usr/local/facet/tools/matlabTNG
    - https://github.com/slaclab/facet-matlabTNG
- **Legacy matlab (2012a) environment**
  - /usr/local/facet/tools/matlab/toolbox
  - emittance_gui, wirescanner_gui, profile_monitor etc
- **FACET-specific EPICS tools**
  - /usr/local/facet/tools/epics
    - https://github.com/slaclab/facet-epics
  - s/w IOC for Laser cleaning watcher
  - s/w IOC for support tools for new matlab watcher

MatlabTNG

- MatlabTNG contains all new matlab code for accelerator and experimental HLAs & watchers
- Requires Matlab v.2020a+

To develop:

- Clone personal copy, e.g. from facet-srv01:

```
$ git clone ssh:///afs/slac/g/cd/swe/git/repos/slac/FACET/matlabTNG.git
```

- Develop & test locally, then push changes to repo & propagate to production location

```
$ git add <list of new and edited files>
$ git commit –m “commit comment”
$ git push
$ cd /usr/local/facet/tools/matlabTNG
$ git pull
```

- More help available on confluence wiki

https://confluence.slac.stanford.edu/display/FACET/FACET-II+Home

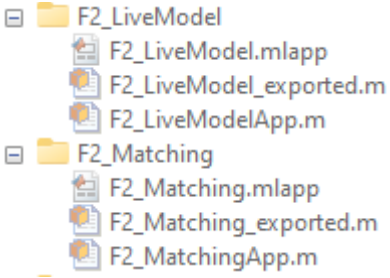
/usr/local/facet/tools/matlabTNG

- common -> useful common tools in here
- comms
- ExampleApps -> examples of how to make GUIs
- F2_CathodeServices -> laser cleaning
- F2_DAN
- F2_DAQ
- F2_Defender
- F2_Feedback -> long. feedbacks
- F2_IN10GunWatcher -> scope waveforms
- F2_LAA
- F2_LAME -> Gun energy calc
- F2_LaserArchiverUtil
- F2_LaserMultiProfmon
- F2_LEM -> FLEM (basis of live model)
- F2_LiveModel -> Lucretia Live Model
- F2_Matching -> Matching & emit calc
- F2_Orbit -> Orbit steering, disp. meas
- F2_S20Config -> Configures Sector 20 optics
- F2_S20LaserTools
- F2_SchottkyScan -> operating gun phase calc
- F2_SimControls
- F2_TuneBC20 -> W-chicane sext tuning
- F2sim -> docker epics
- python -> Xopt
- web -> documentation

+ F2_MDLFF, F2_Wirescan, F2_MultiWire

HLA Development

- General philosophy is to start Matlab in this directory, generate new sub-directory which contains your application code
  - Application code should then run from that sub-directory



- Each app directory has a main entry class file
  - Name has to be <appName>App.m
- Optionally also an “App Designer” generated GUI
  - Name should match app directory name
  - Also must have <appName>_exported.m
    - Generated by using “Save As mfile” option from within App Designer
  - Avoids occasional memory leaks by running this instead of directly using app designer file
- From command line from MatlabTNG:
  - ./runapp.sh <appName> (run app in console mode)
  - ./rungui.sh <appName> (run app in GUI mode)
  - ./runappw.sh <appName> (run app in watcher mode, typically on facet-srv02)

Example App & GUI Class Files

```
classdef F2_MatchingApp < handle & F2_common
  properties
    guihan
    QuadScanData
    TwissFitSource string {mustBeMember(TwissFitSource,["Model","Analytic"])} = "Model"
    ProfFitMethod string {mustBeMember(ProfFitMethod,["Gaussian","Asymmetric"])} = "Asymmetric"
    LiveModel
    Optimizer string {mustBeMember(Optimizer,["fminsearch","lsqnonlin"])} = "lsqnonlin"
    DimSelect string {mustBeMember(DimSelect,["X" "Y" "XY"])} = "XY"
    LM
    ShowPlotLegend logical = true
    UseMatchQuad logical % Which matching quads to use
  end
  properties(SetAccess=private) ...
  properties(SetAccess=private,SetObservable) ...
  properties(SetAccess=private,Hidden) ...
  properties(SetObservable,AbortSet) ...
  properties(SetObservable) ...
  properties(Dependent) ...
  properties(Constant) ...

  methods
    function obj = F2_MatchingApp(ghan) ...
    function DoMatch(obj) ...
    function msg = RestoreMatchingQuads(obj) ...
    function msg = WriteMatchingQuads(obj) ...
    function didload=LoadQuadScanData(obj) ...
    function WriteEmitData(obj) ...
    function ReadEmitData(obj) ...
    function FitQuadScanData(obj) ...
    function PlotQuadScanData(obj) ...
    function PlotTwiss(obj) ...
    function tab = TwissTable(obj) ...
    function tab = MagnetTable(obj) ...
    % Get/Set
    function twiss=get.TwissFit(obj) ...
    function set.ModelDate(obj,val) ...
    function set.UndoAvailable(obj,val) ...
    function set.goodmatch(obj,val) ...
    function set.ModelSource(obj,src) ...
    function kdes = get.quadscan_k(obj) ...
    function set.ProfName(obj,name) ...
    function set.NumMatchQuads(obj,num) ...
  end
  methods(Static,Hidden)
    function opt =  ModelTwissFitFn(x,dims,Rscan,sigma,sigma_err) ...
  end
end
```

Constructor used for passing app/GUI data

```
properties (Access = public)
  aobj % Accompanying application object F2_MatchingApp
end

methods (Access = public)

  function message(app,txt,iserr) ...
end
```

```
% Callbacks that handle component events
methods (Access = private)

  % Code that executes after component creation
  function startupFcn(app)
    app.message("Loading and initializing model...");
    app.DropDown.Enable=false;
    app.GetDatafromCorrPlotorEmitGUIButton.Enable=false;
    app.DoMatchingButton.Enable=false;
    drawnow
    try
      app.aobj = F2_MatchingApp(app) ;
    catch ME
      app.message(["Error initializing model...";string(ME.message)],true);
      return
    end
    app.DropDown.Enable=true;
    app.GetDatafromCorrPlotorEmitGUIButton.Enable=true;
    app.DoMatchingButton.Enable=true;
    app.DropDownValueChanged ; % populates table
    app.message(["Loading and initializing model...","Done."]);
  end
```

Lucretia Design Model

master | facet2-lattice / Lucretia / models / FACET2e / | Go to file

whitegr changed BC11 skew quad to be a skew quad | 4d5da00 | 18 days ago | History

..

FACET2e.mat	changed BC11 skew quad to be a skew quad	18 days ago
FACET2e_altL2Match.mat	alternate L2 match	25 days ago

- Matlab data files in facet2-lattice directory contains design model details
  - **BEAMLINE** cell array contains lattice (cathode to e- dump)
  - **Initial** structure contains twiss parameters at gun
- Alternative designs also kept here for reference
- Matches Oracle DB & MAD8 decks
- Once loaded into Matlab memory, operate on data files with normal Lucretia commands
  - Get 6x6 response matrix from a horizontal corrector to BC14 collimator: ->
  - >> BeamlineViewer % Look at beamline elements
  - >> TwissPlot(1,length(BEAMLINE),Initial,[1 1 0]) % plot beta and eta_x functions
  - >> [stat,Twiss] = GetTwiss(1,length(BEAMLINE),Initial.x.Twiss,Initial.y.Twiss)

- Full Lucretia documentation online:
  https://www.slac.stanford.edu/accel/ilc/codes/Lucretia/
- Lucretia on FACET PROD:
  /usr/local/facet/tools/Lucretia

```
>> findcells(BEAMLINE,'Name','XC14702')

ans =

   678

>> findcells(BEAMLINE,'Name','CE14815')

ans =

   720

>> [stat,R]=RmatAtoB(678,720);
>> disp(R)
    0.6567   11.2198         0         0         0   -0.4213
   -0.0535    0.6092         0         0         0    0.0419
         0         0   -0.2888   19.5087         0         0
         0         0   -0.0687    1.1793         0         0
    0.0050    0.7272         0         0    1.0000   -0.0362
         0         0         0         0         0    1.0000
```

Lucretia Live Model

- From MatlabTNG/ :

>> LM = F2_LiveModelApp ;

- In-memory Lucretia model is now “Live” (synced to control system variables)
- Interrogate model through BEAMLINE, PS, KLYSTRON global variables
- Then normal Lucretia functions work on live model (RmatAtoB etc)

>> LM.UpdateModel % Re-sync live model (or live re-sync with LM.autoupdate=1)

```
>> global BEAMLINE
>> global BEAMLINE PS KLYSTRON
>> findcells(BEAMLINE,'Name','Q12401')

ans =

   432   434

>> GetTrueStrength(432,1)

ans =

    0.2578

>> BEAMLINE{432}.B * PS(BEAMLINE{432}.PS).Ampl * 2

ans =

    0.2578
```

NB:
PS & KLYSTRON globals contain “live” values

```
>> findcells(BEAMLINE,'Name','K12_1*')

ans =

   398   399   400   401

>> BEAMLINE{398}.Klystron

ans =

    10

>> KLYSTRON(10)

ans =

  struct with fields:

         Ampl: 207.8251
     AmplSetPt: 207.8251
      AmplStep: 0
         Phase: -35
     PhaseSetPt: -35
      PhaseStep: 0
        Element: [398 399 400 401]
          dAmpl: 0
         dPhase: 0
           Stat: 'ON'
```

Lucretia Archive Model

```
>> LM = F2_LiveModelApp
>> LM.ArchiveData = [2021,7,1,12,1,1] %
[yr,mnth,day,hr,min,sec]
>> LM.ModelSource = “Archive”
```

- In-memory model now matches set date/time (model sync’d with data from EPICS archiver)
- All usual Lucretia functions can now be used on model which now matches archive date

Helper Classes: LucretiaModel

```
>> LM = LucretiaModel

LM =

  LucretiaModel with properties:

                 Initial: [1x1 struct]
            UseMissingEle: 0
                   istart: 1
                     iend: 1654
                       P0: 0.0060
          ModelClassList: [1422x1 string]
                  ModelP: [1422x1 double]
                  ModelZ: [1422x1 double]
               ModelBDES: [348x1 double]
             ModelBDES_Z: [348x1 double]
             ModelBDES_L: [348x1 double]
           ModelRegionID: [11x2 uint32]
            ModelRegionE: [11x2 single]
            ControlNames: [1422x1 string]
              ModelNames: [1422x1 string]
             ModelID_all: [1654x1 double]
                 ModelID: [1422x1 double]
           MissingEleInd: []
                    PSid: [1422x1 double]
    LucretiaModelVersion: 1
             MissingEle: ["YC57145"    "YC57146"]
          ModelRegionName: [11x1 string]
                  GEV2KGM: 33.3564
                   GEV2TM: 3.3356
             ModelKlysID: [8x10 uint8]
              ModelKlysZ: [8x10 double]
          ModelDesignFile: "FACET2e"
              DesignTwiss: [1x1 struct]
           DesignBeamline: {1654x1 cell}
                 RefTwiss: []
                 Initial1: []
             ModelClasses: "All"
                UseRegion: [11x1 logical]
```

- Simplifies common tasks addressing model
- Operates on unique elements (de-splits model)
- Operate on sub-regions:

>> LM.UseRegion(*reg*), where *reg* is logical vector addressing regions LM.ModelRegionName:

"INJ" "L0" "DL1" "L1" "BC11" "L2" "BC14" "L3" "BC20" "FFS" "SPECTDUMP"

- Operate on specific classes:

>> LM.ModelClasses = [“QUAD” “XCOR” “YCOR”]

(set to one or more supported Lucretia Class type)

Helper Classes: F2_common

```
>> fc=F2_common

fc =

  F2_common with properties:

            confdir: "/u1/facet/matlab/config"
           modeldir: "/usr/local/facet/tools/facet2-lattice/Lucretia/models"
     LucretiaLattice: "/usr/local/facet/tools/facet2-lattice/Lucretia/models/FACET2e/FACET2e.mat"
          UseArchive: 0
         ArchiveDate: [2021 7 1 12 1 1]
            datadir: "/u1/facet/matlab/data/2022/2022-03/2022-03-27"
           beamrate: 30
```

- Use LucretiaLattice property to reference lattice source
  - Used to change root lattice source for all apps
- Also some other common interfaces properties and methods which can be inherited or used by other app classes

Helper Classes: BPMs, Magnets, Klystrons

```
>> B = F2_bpms
Setting severity REJECTION level to 4

B =

  F2_bpms with properties:

       plotscale: 0
             dim: "xy"
            xdat: [111x10 double]
            ydat: [111x10 double]
            tmit: [111x10 double]
          pulseid: 1
           nread: []
              LM: [1x1 LucretiaModel]
    nepicsbuffer: 1.5000
        bpmnames: [111x1 string]
       modelnames: [111x1 string]
           modelZ: [111x1 double]
          modelID: [111x1 double]
       epicsnames: [111x1 string]
        stuckbpms: []
      UpdateTimer: []
          beamrate: 30
            xave: [111x1 double]
            yave: [111x1 double]
            xrms: [111x1 double]
            yrms: [111x1 double]
          tmitave: [111x1 double]
          tmitrms: [111x1 double]
        UseRegion: [1 1 1 1 1 1 1 1 1 1 1]
        BufferLen: 10
       autoupdate: 0
          badbpms: ["BPM10781"    "BPM19851"]
             edef: 3
              f2c: [1x1 F2_common]
        epicsonly: [111x1 logical]
```

```
>> M=F2_mags(LM)

M =

  F2_mags with properties:

        WriteEnable: 0
        WriteAction: "TRIM"
          WriteDest: "BDES"
         RelTolBDES: 1.0000e-03
         RelTolBACT: 0.1000
         AbsTolBDES: 1.0000e-03
         AbsTolBACT: 0.1000
           UseFudge: 0
         UpdateRate: 1
               BDES: []
         autoupdate: 0
          UseSector: [1 1 1 1 1]
         MagClasses: ["QUAD"    "SEXT"    "
            Initial: []
                 LM: [1x1 LucretiaModel]
           BDES_err: []
           BACT_err: []
         BDES_cntrl: []
         BACT_cntrl: []
               BMIN: []
               BMAX: []
           BfudName: ["QM11393"    "Q11401"
               Bfud: [1.0379 0.6997 -0.0905
            version: 1.1000
            confdir: "/u1/facet/matlab/conf
           modeldir: "/usr/local/facet/tool
     LucretiaLattice: "/usr/local/facet/tool
         UseArchive: 0
        ArchiveDate: [2021 7 1 12 1 1]
            datadir: "/u1/facet/matlab/data
           beamrate: 30
```

```
>> K=F2_klys(LM)

K =

  F2_klys with properties:

     KlysPhaseOverride: [8x10 single]
      KlysAmplOverride: [8x10 single]
    KlysForceZeroPhase: 0
         KlysUseSector: [1 1 1 1]
             KlysInUse: [8x10 logical]
            UseArchive: 0
            UpdateRate: 0
           ArchiveDate: [2021 7 1 12 1 1]
              KlysStat: [8x10 uint8]
          KlysSectorMap: [8x10 uint8]
             KlysPhase: [8x10 single]
              KlysAmpl: [8x10 single]
                    LM: [1x1 LucretiaModel]
           KlysControl: [8x10 uint8]
          KlysBeamcode: 10
               version: 1
          KlysStatName: "IGNORE"
           SectorPhase: [54.9207 1.0000e-10 -
```

- Read, readbuffer
- ReadB, WriteBDES
- GetAmpl, GetPhase, GetStat

Helper Classes: AIDA-PVA

- AIDA-PVA allows access to SCP data
- https://www.slac.stanford.edu/grp/cd/soft/aida/aida-pva/index.html
- aidaget('QUAD:LI12:401:BACT')
  - Generic access command (in general avoid, use EPICS CA instead)

```
>> MK = SCP_MKB('l2_phase')
```

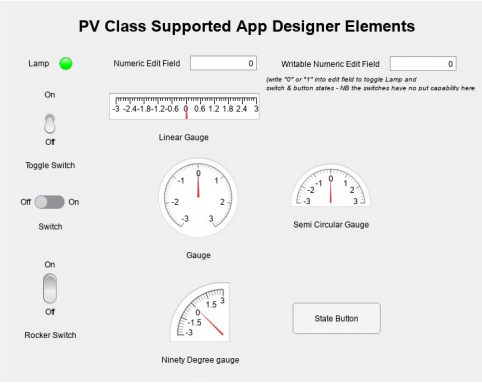
-> Use with any SCP multiknob

```
MK =

  SCP_MKB with properties:

    DeviceNames: ["SBST:LI11:1:PDES"    "SBST:LI12:1:PDES"    "SBST:LI13:1:PDES"    "SBST:LI14:1:PDES"]
     DeviceVals: [1.0000e-10 -35 -35 -35]
            val: 0
           Name: "mkb:l2_phase.mkb"
```

>> MK.set(val)

Helper Classes: PV & GUI Integration

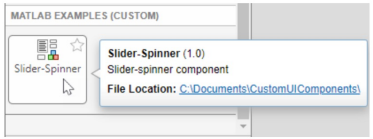
- Two EPICS channel access clients provided in helper “PV” class (in matlabTNG/common)
  - labca & “ca” (java client)
- PV class can use either
- Extensions over regular labCA:
  - Asynchronous gets
  - Simple switch between live and archive gets
  - Builtin continuous updating for single PVs or lists
  - Builtin GUI interface:



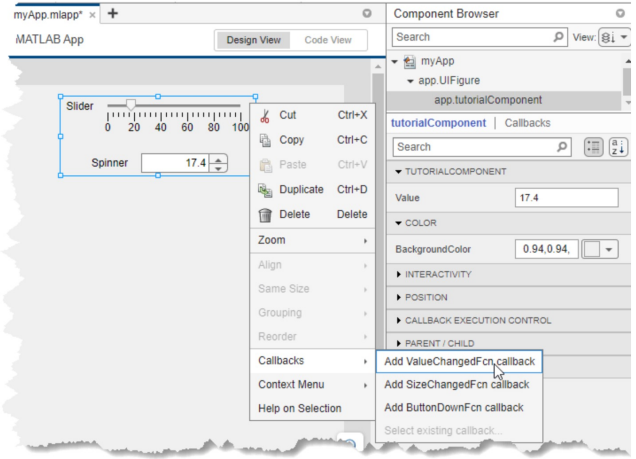

```
% Launch app and capture application object containing component fields
app = Example2 ;

% generate a (java) context object, required by the PV class to perform read/write operations to EPICS PV channels using a java CA client
% (NB: this should be called only once per Matlab session)
context = PV.Initialize(PVtype.EPICS) ;

% Generate list of PV objects and associate with app components
%  'name' field is user-supplied name to refer to this PV channel locally
%  'pvname' field should be EPICS PV name
%  'monitor' field should be set to true to automatically update the local PV values (starts when you call the run() method)
%  'guihan' field is the App Designer component to associate with a given PV channel
%  'mode' field should be set to "rw" if you want to be able to write to this PV
pvlist = [ PV(context,'name',"LampPV",'pvname',"SI0C:SYS1:MLOO:A0956",'monitor',true,'guihan',app.Lamp);
  PV(context,'name',"ToggleSwitchPV",'pvname',"SI0C:SYS1:MLOO:A0956",'monitor',true,'guihan',app.ToggleSwitch);
  PV(context,'name',"SwitchPV",'pvname',"SI0C:SYS1:MLOO:A0956",'monitor',true,'guihan',app.Switch);
  PV(context,'name',"RockerSwitchPV",'pvname',"SI0C:SYS1:MLOO:A0956",'monitor',true,'guihan',app.RockerSwitch);
  PV(context,'name',"NumericEditFieldPV",'pvname',"SI0C:SYS1:MLOO:A0956",'monitor',true,'guihan',app.NumericEditField);
  PV(context,'name',"WriteableNumericEditFieldPV",'pvname',"SI0C:SYS1:MLOO:A0956",'monitor',true,'guihan',app.WritableNumericEditField,'mode',"rw");
  PV(context,'name',"LinearGaugePV",'pvname',"SI0C:SYS1:MLOO:A0953",'monitor',true,'guihan',app.LinearGauge);
  PV(context,'name',"GaugePV",'pvname',"SI0C:SYS1:MLOO:A0953",'monitor',true,'guihan',app.Gauge);
  PV(context,'name',"NinetyDegreeGaugePV",'pvname',"SI0C:SYS1:MLOO:A0953",'monitor',true,'guihan',app.NinetyDegreegaugeGauge);
  PV(context,'name',"SemicircularGaugePV",'pvname',"SI0C:SYS1:MLOO:A0953",'monitor',true,'guihan',app.SemiCircularGauge);
  PV(context,'name',"StateButtonPV",'pvname',"SI0C:SYS1:MLOO:A0956",'monitor',true,'guihan',app.StateButton) ] ;
pset(pvlist,'debug',0) ; % Set debug level to 0 to enable read/write operations (make PV objects live)
% start timer which keeps local values of PV data updated and updates GUI field. async=true option uses asnchronous get methods (non blocking)
run(pvlist,true,0.02); % (async, polltime) - set polling time to a value (s) less than the fastest rate at which you expect PV values to be changing
```

Future Project Idea:
Add PV UI Components to App Designer

The component appears in the **Component Library** of the app, under the category specified in the dialog box.



Drag a slider-spinner component onto the canvas. You can set public properties of the component using the **Component Browser**, ar



- Matlab has custom App Design UI component functionality
- Combine with PV class development to provide “EDM-style” drag-and-drop PV-aware UI components to App Designer
- Would provide similar look-and-feel to EDM to graphically design, test and deploy CA-aware GUI’s