Report **Subject** **Control of CCD Cameras**	Reference No : TARLA-REP-004 Date : 04.03.2020 Page : 19 Author : Ali Can Canbay Email : accanbay@tarla.org.tr Version : 01 Distribution : TARLA

Project:

TARLA

Abstract

Many CCD cameras are used at different points on the TARLA beam line. To control these cameras, Experimental Physics and Industrial Control System (EPICS) IOCs must be installed and their interfaces must be designed in Control System Studio (CSS). This document describes these setups and user interface for the use of Baumer VLG-20M CCD cameras.

Table of Contents

1. Baumer VLG-20M CCD Cameras

Baumer VLG-20M CCD cameras are used in the TARLA beam line. The features of these cameras are given in table 1.1.

Table 1.1 Features of Baumer VLG-20M CCD Cameras [1]

Sensor	Sony ICX274
Resolution	1624×1228 px
Exposure time	0,004 ... 60000 ms
Pixes size	4.4 × 4.4 µm
Shutter type	Global Shutter
Sensor type	1/1.8″ CCD
Image formats, Interface, Frame rate max.	Full Frame, 1624×1228 px 27 fps
Pixel formats	Mono 8 Mono 12 Mono 12 Packed
Analog controls	Gain (0 ... 29 dB) Offset (0 ... 1023 LSB 14 Bit)
Data interface	Gigabit Ethernet, Transfer rate 1000 Mbits/sec, Fast Ethernet, Transfer Rate 100 Mbits/sec, Connector: SACC-CI-M12FS-8CON-L180-10G
Process interface / Power supply	SACC-CI-M12MS-8CON-SH TOR 32 / 8 pins

2. EPICS areaDetector

TARLA control system uses Experimental Physics and Industrial Control System (EPICS) [2] and EPICS areaDetector module is used for camera systems.

The EPICS areaDetector module provides a general-purpose interface for area (2-D) detectors in EPICS. It is intended to be used with a wide variety of detectors and cameras, ranging from high frame rate CCD and CMOS cameras, pixel-array detectors such as the Pilatus, and large format detectors like the Perkin Elmer flat panels. [3]

The top-level repository for the EPICS areaDetector software is the *areaDetector* repository. This repository contains mostly documentation, configuration files, and a top-level Makefile to build the entire areaDetector package.

The areaDetector code is contained in submodules under this module. In our installations three of these are “core” submodules:

1. ADSupport. This contains the source code for support libraries (TIFF, JPEG, HDF5, XML2, etc.). This is required for Windows and vxWorks, and can optionally be used on Linux and Darwin.

2. ADCore. This contains the base classes, plugins, and documentation.

3. ADViewers. This contains viewers for displaying areaDetector images in ImageJ and IDL.

4. aravisGigE. This provides the connection between the areadetector and gige vision cameras.

2.1 Installation of areaDetector

To install areaDetector, EPICS and required linux packages must be installed first. Our installation used EPICS version 3.15 (TARLA-REP-003 can be examined for installation information). For installing required Linux packages for Centos7, you can use the commands below

```
yum install -y epel-release
```

```
yum install -y re2c libtiff zlib libjpeg-turbo-devel libxml2-devel glib2-devel libXext-devel libusbx-devel glibmm24-devel gstreamer-plugins-base-devel gstreamer1-plugins-base-devel python-gobject gobject-introspection gtk-doc gtk3-devel libnotify-devel GraphicsMagick
```

The following EPICS Support modules must also be installed before starting the installation.

sequencer
ipac
asyn
autosave
busy
sscan
calc
iocStats

The versions of the modules installed in the TARLA control systems are given in the Table 2.1.

Table 2.1 EPICS Modules and Versions of TARLA Control Systems

asyn	4-32
autosave	R5-7-1
busy	R1-6-1
calc	R3-6-1
ipac	2.14
seq (Sequencer)	2.2.5
areaDetector	R3-3-1
ADCore	R3-3-1
ADSupport	R1-4
aravisGigE	R3-0

For installation, the areaDetector and related submodules are moved to the directory to be installed as follows (In EPICS_ROOT/support directory).

areaDetector/ADCore
areaDetector/ADSupport
areaDetector/ADViewers
areaDetector/aravisGigE

Example files are converted into usable files by running the

```
cp EXAMPLE_RELEASE.local          RELEASE.local
cp EXAMPLE_RELEASE_LIBS.local     RELEASE_LIBS.local
cp EXAMPLE_RELEASE_PRODS.local    RELEASE_PRODS.local
cp EXAMPLE_CONFIG_SITE.local      CONFIG_SITE.local
```

commands in the areaDetector/configure folder.

In **RELEASE.local** file, comment in all rows other than fallowing rows by #.

```
ADSUPPORT=$(AREA_DETECTOR)/ADSupport
ARAVISGIGE=$(AREA_DETECTOR)/aravisGigE
```

In **RELEASE_LIBS.local** and **RELEASE_PRODS.local** file, fix module paths to true ones.

In CONFIG_SITE.local file, change WITH_PVA YES to NO and add the fallowing lines for fixing glib2 errors.

```
GLIB_INCLUDE = /usr/include/glib-2.0 /usr/lib64/glib-2.0/include
glib-2.0_DIR = /usr/lib64
```

Then go to areaDetector/aravisGigE directory and run **install.sh** script.

After all this is done you can go back to areaDetector directory and run make command for complate installation.

2.2 Adding New Camera to aravisGigE

For adding new cameras you must fallow steps below:

1. First go to aravisGigE directory and run command below to see connected cameras.

   ```
   ./bin/linux-x86_64/arv-tool-0.6
   ```

   In my case it returns with “Baumer Optronic-0402910514”

2. Create XML file for your genicam camera

   ```
   ./bin/linux-x86_64/arv-tool-0.6 -n "Baumer Optronic-0402910514" genicam > "Baumer_VLG20M.xml"
   ```

   The name of XML file can be anything.

3. Create database files for IOC

   ```
   ./aravisGigEApp/src/makeDbAndEdl.py Baumer_VLG20M.xml Baumer_VLG20M
   ```

4. Create ADL files (for medm)

   ```
   ./aravisGigEApp/src/makeAdl.py Baumer_VLG20M.xml Baumer_VLG20M
   ```

5. Then you can go to areaDetector/aravisGigE/iocs/aravisGigEIOC/iocBoot/iocAravisGigE directory and stange st.cmd file for your camera.

After all this done you have to run commands below to rename some files to run camera IOC.

```
mv ${EPICS_ROOT}/support/areaDetector/ADCore/iocBoot/EXAMPLE_commonPlugins.cmd ${EPICS_ROOT}/support/areaDetector/ADCore/iocBoot/commonPlugins.cmd
```

```
mv ${EPICS_ROOT}/support/areaDetector/ADCore/iocBoot/EXAMPLE_commonPlugin_settings.req ${EPICS_ROOT}/support/areaDetector/ADCore/iocBoot/commonPlugin_settings.req
```

2.3 Custom Database and Scripts for TARLA CCD Cameras

The overall workflow required for CCD camera script is shown in Figure 2.2.

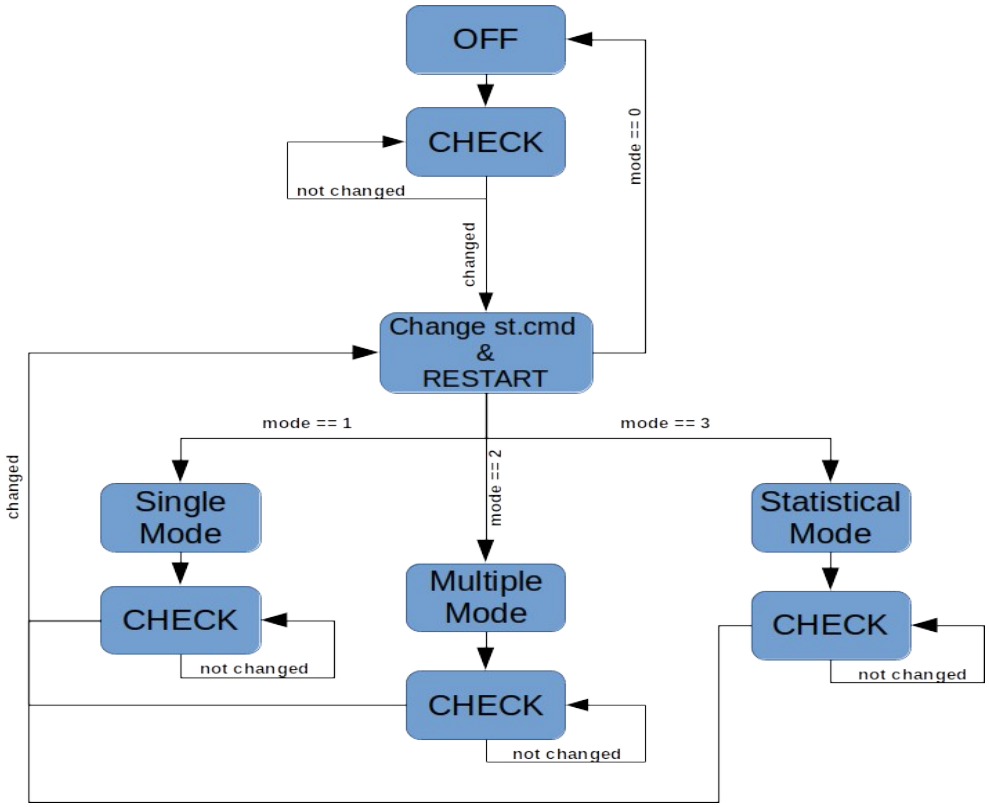

Figure 2.1 General workflow of TARLA CCD camera script

According to this workflow, new databases are needed to instantly check the camera states and the mode used. The contents of the database file created by assuming that up to 3 cameras will work simultaneously in the multiple mode are given below.

```
record(ai, "BL:DCC:RESTART"){
        field(DESC,"IF EPICS on restart state or not")
        field(VAL,"0")
}
record(stringin, "BL:DCC:ViewerMode"){
        field(DESC,"Set Viewer Mode")
        field(VAL,"None")
}
record(stringin, "BL:DCC:CamState1"){
        field(DESC,"Status of Camera")
        field(VAL,"OFF")
}
record(stringin, "BL:DCC:CamState2"){
        field(DESC,"Status of Camera2")
        field(VAL,"OFF")
}
record(stringin, "BL:DCC:CamState3"){
        field(DESC,"Status of Camera3")
        field(VAL,"OFF")
}
record(ai, "BL:DCC:CamSelector1"){
        field(DESC,"Set Camera1")
        field(VAL,"0")
}
record(ai, "BL:DCC:CamSelector2"){
        field(DESC,"Set Camera2")
        field(VAL,"0")
}
record(ai, "BL:DCC:CamSelector3"){
        field(DESC,"Set Camera3")
        field(VAL,"0")
}
record(ai, "BL:DCC:ViewerModeA"){
        field(DESC,"Analog Mode")
        field(VAL,"99999")
}
record(ai, "BL:DCC:CamState1A"){
        field(DESC,"Status Analog Mode")
}
record(ai, "BL:DCC:CamState2A"){
        field(DESC,"Status Analog Mode")
}
record(ai, "BL:DCC:CamState3A"){
        field(DESC,"Status Analog Mode")
}
```

The RESTART option is required to restart IOC in case of mode change (explained in Section 2.2); The ViewerMode controls the current mode of the camera; CamState# shows state (connected or disconnected) of camera; CemSelector# determines which camera's information is sent to that number (#) for single and multiple modes; VieweModeA contains ViewerMode information as analog value (0 for OFF, 1 for Single Mode, 2 for Multiple Mode and 3 for Statistic Mode. Since this PV is necessary for button styles in CSS, the default value is chosen as a large value) and

finnally CamState#A contains CamState# information as analog value (0 for disconnected and 1 for connected). BL:DCC: at the beginning of PVs is the part of TARLA PV naming standard (BL means “Beam Line” and DCC means “Diagnostic CCD Camera”).

After the database was created, the script for the workflow given in Figure 2.1 was written according to this database and its own database of aravisGigE. Workflow of Single Mode, Multiple Mode and Statistical Mode is shown respectively in Figure 2.2, Figure 2.3 and Figure 2.4.

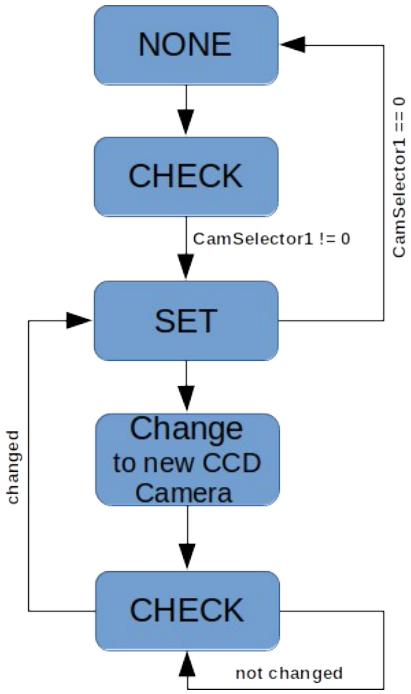

Figure 2.2 Workflow of Single Mode

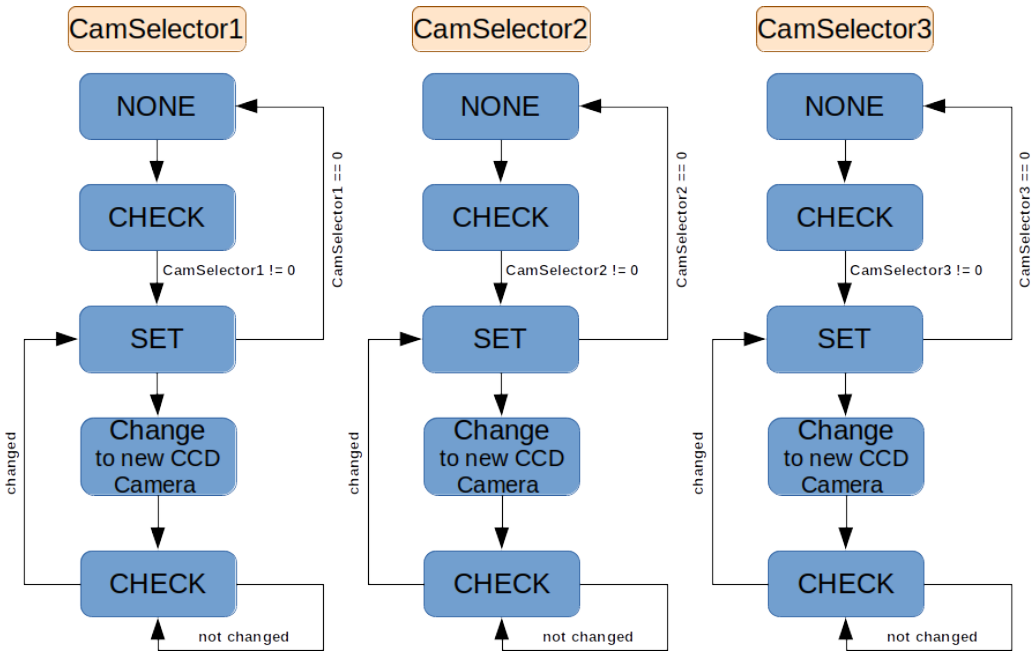

Figure 2.3 Workflow of Multiple Mode

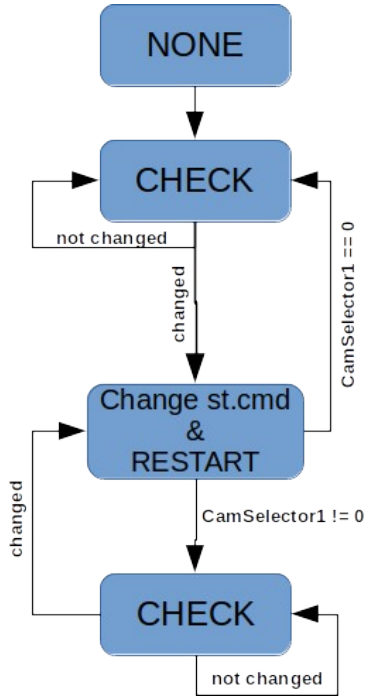

Figure 2.4 Workflow of Statistic Mode

A script with these flow algorithms can work alone, but in some cases EPIC also needs to be checked separately. Another script created for CCD IOCs is the scrip needed to restart EPICS. To solve this problem, an additional script that controls EPICS was written according to the flow diagram in Figure 2.5. Looking at this new algorithm, it seems that ending IOC in main script is enough to restart EPICS.

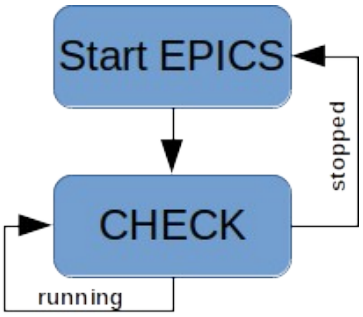

Figure 2.5 Workflow for the EPICS Controling Scripts

This mode is available to quickly access the camera and view the beam, so only the incoming image is important. In this mode it may also be necessary to access multiple cameras simultaneously. For this reason, a spesific st.cmd file must be created. Below is the content of the st.cmd file created for Multiple Mode (This file can also be used for Single Mode as it will use CamSelector1).

st_Image.cmd :

```
< envPaths
errlogInit(20000)

dbLoadDatabase("$(TOP)/dbd/aravisGigEApp.dbd")
aravisGigEApp_registerRecordDeviceDriver(pdbbase)
epicsEnvSet("QSIZE",  "20")
epicsEnvSet("XSIZE",  "1624")
epicsEnvSet("YSIZE",  "1228")
epicsEnvSet("NCHANS", "2048")
epicsEnvSet("CBUFFS", "500")

epicsEnvSet("EPICS_DB_INCLUDE_PATH", "$(ADCORE)/db")
dbLoadRecords("$(ARAVISGIGE)/iocs/aravisGigEIOC/iocBoot/iocAravisGigE/Cam_Checker.db")

aravisCameraConfig("ARV1", "Baumer Optronic-0329812713")
asynSetTraceMask("ARV1",0,0x21)
NDStdArraysConfigure("Image1", 5, 0, "ARV1", 0, 0)

aravisCameraConfig("ARV2", "Baumer Optronic-0402910514")
asynSetTraceMask("ARV2",0,0x21)
NDStdArraysConfigure("Image2", 5, 0, "ARV2", 0, 0)

...
########
########

dbLoadTemplate "camera.substitutions"
iocInit()
```

Here, “Baumer Optronic-##########” written places, connected cameras seen by the method specified in 2.1 are shown. 2 cameras have been added for demonstration. More cameras can be added in a similar way.

In Statistics Mode it is necessary to access all the features of the camera. Therefore, a new st.cmd file and substitutions file are needed.

camera.substitutions :

```
file /home/cam/epicsv3/support/areaDetector/aravisGigE/db/Baumer_VLG20M.template {
        pattern { P         R         PORT    ADDR  TIMEOUT }
                { BL:DCC:CCD1:        cam1:   ARV1   0       1 }
                { BL:DCC:CCD2:        cam1:   ARV2   0       1 }

                • • •
}
file /home/cam/epicsv3/support/areaDetector/aravisGigE/db/aravisCamera.template {
        pattern { P         R         PORT    ADDR  TIMEOUT }
                { BL:DCC:CCD1:        cam1:   ARV1   0       1 }
                { BL:DCC:CCD2:        cam1:   ARV2   0       1 }

                • • •
}
file /home/cam/epicsv3/support/areaDetector/ADCore/db/NDStdArrays.template {
        pattern { P         R         PORT    NDARRAY_PORT        TYPE    FTVL    NELEMENTS }
                { BL:DCC:CCD1:        image1: Image1  ARV1    Int16   SHORT 1994272 }
                { BL:DCC:CCD2:        image1: Image2  ARV2    Int16   SHORT 1994272 }

                • • •
}
```

st_Statistic.cmd :

```
< envPaths
errlogInit(20000)
dbLoadDatabase("$(TOP)/dbd/aravisGigEApp.dbd")
aravisGigEApp_registerRecordDeviceDriver(pdbbase)
epicsEnvSet("PREFIX", "BL:DCC:")
epicsEnvSet("PORT",   "ARV1")
epicsEnvSet("QSIZE",  "20")
epicsEnvSet("XSIZE",  "1624")
epicsEnvSet("YSIZE",  "1228")
epicsEnvSet("NCHANS", "2048")
epicsEnvSet("CBUFFS", "500")
epicsEnvSet("EPICS_DB_INCLUDE_PATH", "$(ADCORE)/db")

dbLoadRecords("$(ARAVISGIGE)/iocs/aravisGigEIOC/iocBoot/iocAravisGigE/Cam_Checker.db")
aravisCameraConfig("$(PORT)", "Baumer CAMERA_VARIABLE")
dbLoadRecords("$(ARAVISGIGE)/db/Baumer_VLG20M.template","P=$(PREFIX),R=cam1:,PORT=$
(PORT),ADDR=0,TIMEOUT=1")

asynSetTraceMask("$(PORT)",0,0x21)
dbLoadRecords("$(ARAVISGIGE)/db/aravisCamera.template",          "P=$(PREFIX),          R=cam1:,          PORT=$
(PORT),ADDR=0,TIMEOUT=1")

NDStdArraysConfigure("Image1", 5, 0, "$(PORT)", 0, 0)
dbLoadRecords("$(ADCORE)/db/NDStdArrays.template",   "P=$(PREFIX),   R=image1:,   PORT=Image1,   ADDR=0,
TIMEOUT=1, NDARRAY_PORT=$(PORT), TYPE=Int16, FTVL=SHORT, NELEMENTS=1994272")

< $(ADCORE)/iocBoot/commonPlugins.cmd
set_requestfile_path("$(ADPILATUS)/prosilicaApp/Db")
iocInit()
```

When no mode is selected (NONE for us), you can run an empty st.cmd as below to reduce network traffic.

st_None.cmd :

```
< envPaths
errlogInit(20000)
dbLoadDatabase("$(TOP)/dbd/aravisGigEApp.dbd")
aravisGigEApp_registerRecordDeviceDriver(pdbbase)
dbLoadRecords("$(ARAVISGIGE)/iocs/aravisGigEIOC/iocBoot/iocAravisGigE/Cam_Checker.db")
iocInit()
```

Finally, in order to separate the background and the beam when the bundle arrives, new PVs must be created in addition to the PVs at ADCore. For this, the following lines should be added to the **$(ADCORE)/db/NDStdArrays.template** file.

```
record(acalcout,"$(P)$(R)SSArray"){
	field(NELM,"1994272")
	field(NUSE,"1994272")
	field(SCAN,"I/O Intr")
	field(INAA,"$(P)$(R)ArrayData")
	field(CALC,"AA")
}
record(acalcout,"$(P)$(R)ArrayDataNoBg"){
	field(NELM,"1994272")
	field(NUSE,"1994272")
	field(SCAN,"Passive")
	field(INAA,"$(P)$(R)ArrayData")
	field(INBB,"$(P)$(R)SSArray.AVAL")
	field(CALC,"AA - BB")
}
```

SSArray receives a copy of the frame from the camera as soon as it is run. ArrayDataNoBg instantly extracts the background data stored in SSArray from the camera image and broadcast the new image without background.

Another file is required to record the camera names and to test whether the related cameras are connected by serial numbers. The file required for this to be done for 2 cameras is below. Other cameras can be added in a similar way.

camera_list :

```
OFF,OFF,NONE
CCD1,0329812713
CCD2,0402910514
```

...

Whether the cameras are connected can be determined by comparing the output of the command showing the connected cameras given in section 2.2 with the serial numbers in this file.

After these processes, the IOC is started by running the scripts created. PVs that are sufficient to test whether the camera is working are included in the Table 2.2 (these are sufficient for image

acquisition only). However, it is necessary to make a correction on the computer before starting the test process. The maximum array size that the computer can handle should be determined by the camera and added to the end of the bashrc file as follows (for Baumer VLG-20M, this value is 1624×1228 = 1994277 and the value to be written must be larger than this).

```
export EPICS_CA_MAX_ARRAY_BYTES=31908352
```

If you are using two ethernet cards, you should add the following lines to bashrc so that you can broadcast PVs to the network (in some cases).

```
export EPICS_CA_AUTO_ADDR_LIST=NO
export EPICS_CA_ADDR_LIST=127.0.0.1
```

Table 2.2 PVs are sufficient to capture the camera image

$(Prefix)cam1:Acquire	Enable/Disable image acquisition from CCD camera
$(Prefix)image1:ArrayCallbacks	Image data as an array
$(Prefix)EnableArrayCallbacks	Enabling/Disabling to export the image as an array

$(Prefix) in Table 2.2 is BL:DCC: for Statistic Mode and BL:DCC:CCD# for other modes.

3. User Interfaces

3.1 Motif Editor and Display Manager

Motif Editor and Display Manager (medm) is a Motif graphical user interface for designing and implementing control screens, called displays, that consist of a collection of graphical objects that display and/or change the values of EPICS process variables [4]. areaDetector and its modules (including aravisGigE) contain medm files in itself.

For installing medm, EPICS extensions [5] module must be installed. After installing EPICS extensions, medm files are added to the src folder and medm is installed by running make command in EPICS extensions. After these operations, medm can be run in everywhere by adding to the path (the following lines are added to bashrc).

```
alias medm=${EPICS_ROOT}/extensions/bin/linux-x86_64/medm
```

In aravisGigE files of medm (files with .adl extensions) are produced as specified in section 2.2 and are in **$(aravisGigE)/aravisGigEApp/op/adl/** . The expert interfaces of the cameras can be accessed by editing cameraTop.adl here. However, these files created by aravisGigE must run files in other locations within them. Therefore the lines below should be added to bashrc.

```
export AREA_DETECTOR=${EPICS_ROOT}/support/areaDetector
export EPICS_DISPLAY_PATH=${AREA_DETECTOR}/ADCore/ADApp/op/adl:${EPICS_ROOT}/support/calc/calcApp/op/adl:${EPICS_ROOT}/support/iocStats/op/adl/:${EPICS_ROOT}/support/sscan/sscanApp/op/adl/:${EPICS_ROOT}/support/asyn/opi/medm/:${EPICS_ROOT}/support/autosave/asApp/op/adl/:${EPICS_ROOT}/support/busy/busyApp/op/adl/
```

After these adjustments, you can access expert mode with medm and make the necessary settings. The use of medm is as follows. Files (file.adl in the example below) can be opened in executable mode by typing “x” or in edit mode by typing “e” instead of “option” via the terminal.

```
medm -option file.adl
```

medm only provides the interface required for adjustments, imaging is not done here. The main user interface for CCD Cameras is given in Figure 3.1.

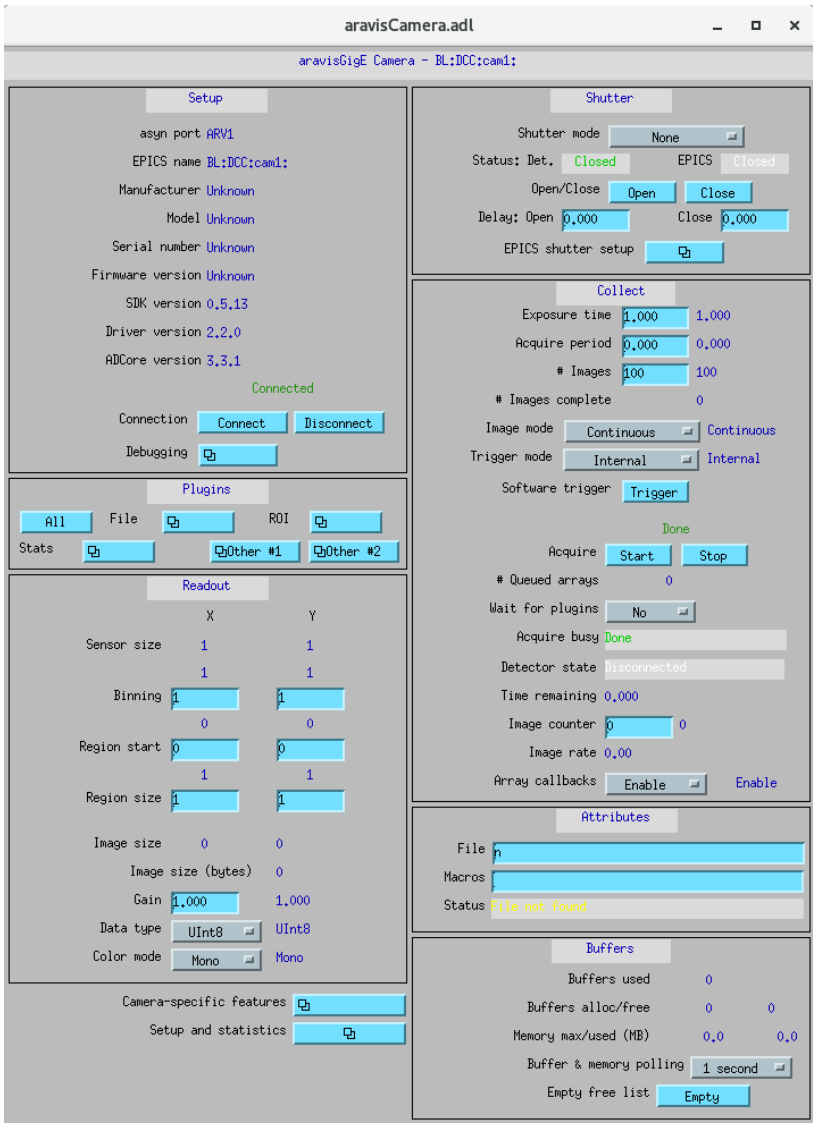

Figure 3.1 The main user interface for CCD Cameras on medm

3.2 ImageJ

ImageJ is a public domain Java image processing program [6]. After BL:DCC:cam1:Acquire and BL:DCC:EnableArrayCallbacks are activated, the image can be imported with ImageJ. For this, ImageJ files (java based) in ADViewer are used.

ImageJ runs and opens EPICS_AD_Controller.java from ImageJ plugins in ADViewer (automatically added to ImageJ plugins after first run). In the window that opens, $(Prefix)image1 is written in PVPrefix section ($(Prefix) must be written as mentioned in section 2.3.) and click START. Then the display screen of the camera will appear. ImageJ's Gaussian_Profiler.java plugin is used for gaussian graphics (this is also in ADViewer). You can see a ImageJ's interface screen in Figure 3.2.

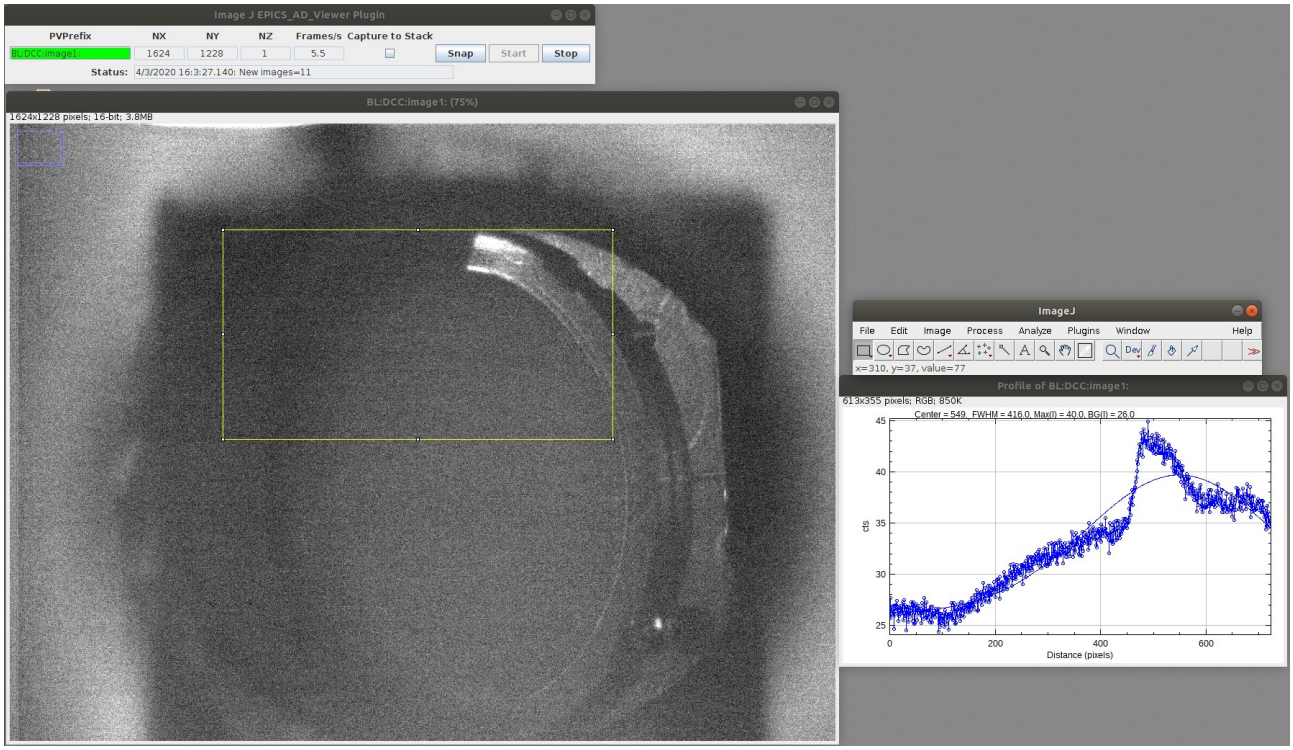

Figure 3.2 ImageJ's interface screen

3.3 Control System Studio

Control System Studio (CSS) [6] is used for TARLA control system user interfaces. CSS works with OPI files, and aravisGigE's OPI files can be created by converting existing ADL files (as in medm, it should be noted that all necessary OPI files in EPICS support modules should also be used here). The main user interface OPI, converted from ADL file, is given in Figure 3.3.

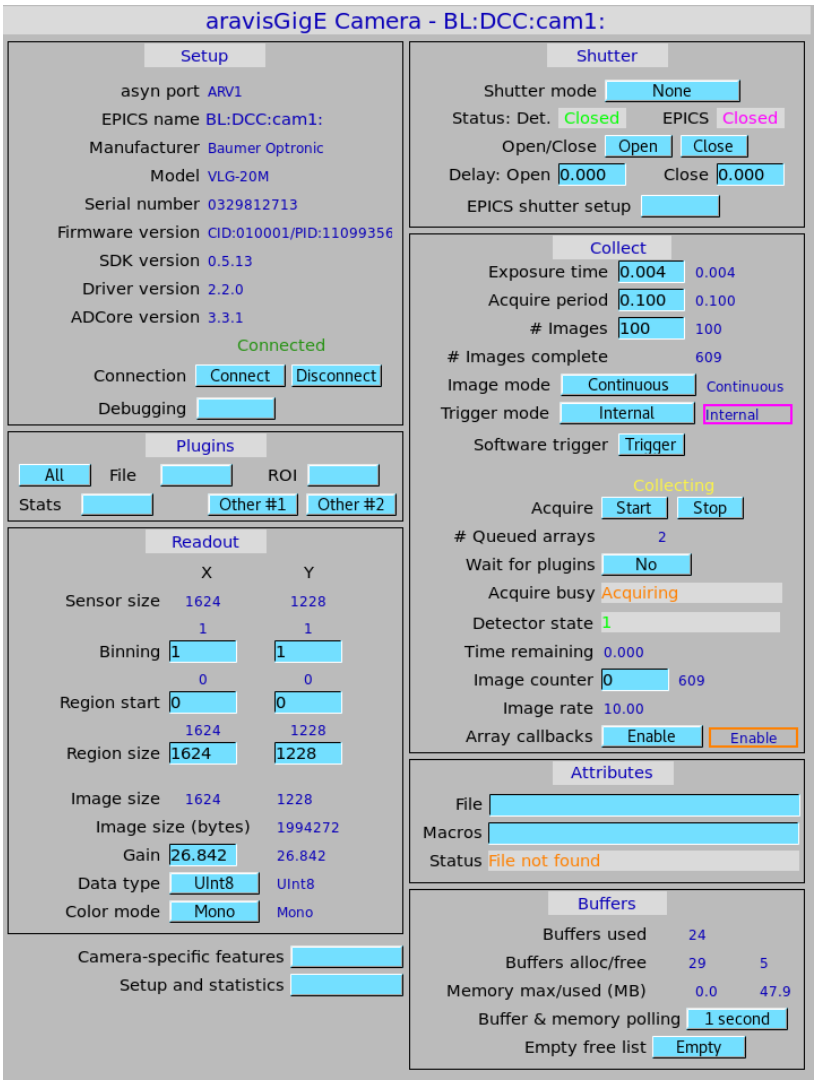

Figure 3.3 The main user interface for CCD Cameras on CSS

Apart from these OPIs, we also need new and particular OPIs for the TARLA control system. These new OPIs are made for operators to access the cameras more easily and control them quickly. OPIs prepared for Single Mode, Multiple Mode and Statistical Mode are given in Figure 3.3, Figure 3.4 and 3.5, respectively. Expert mode, which appears in Figure 3.6, opens the screen in Figure 3.1.

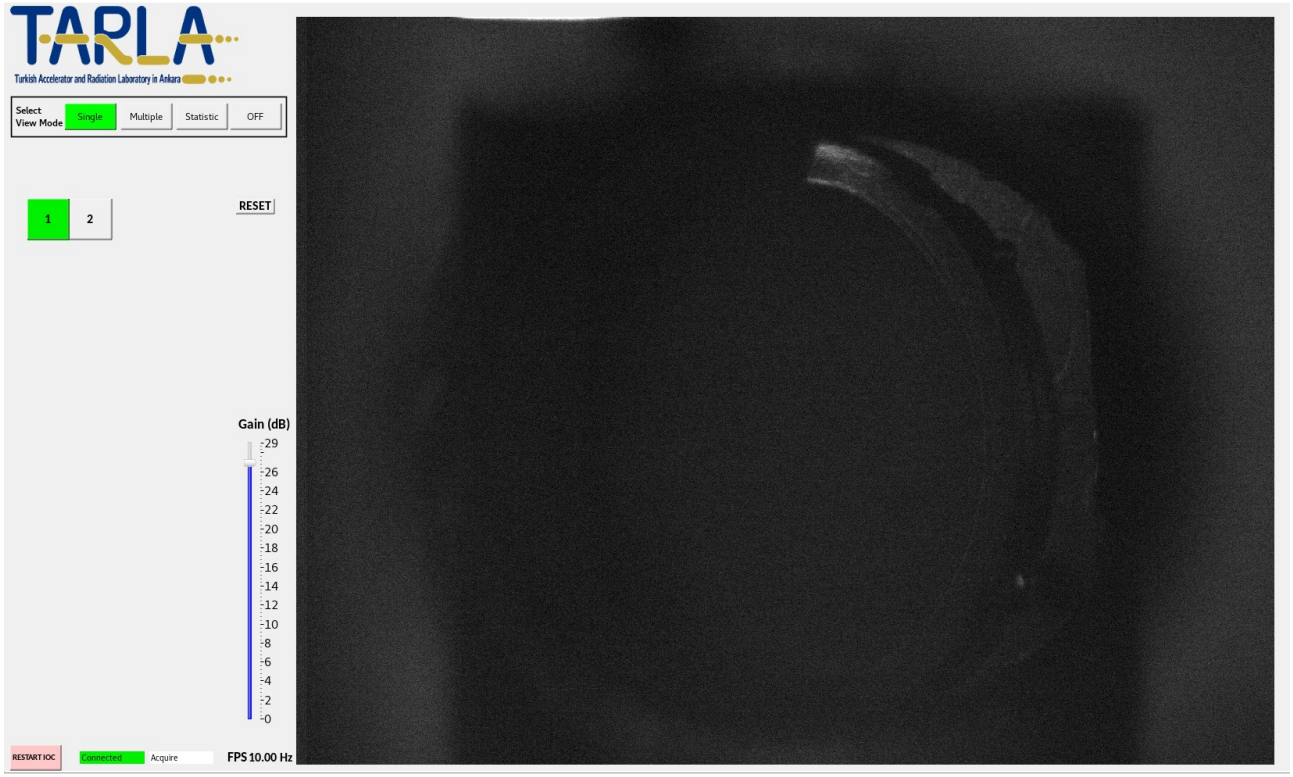

Figure 3.4 Single Mode user interface

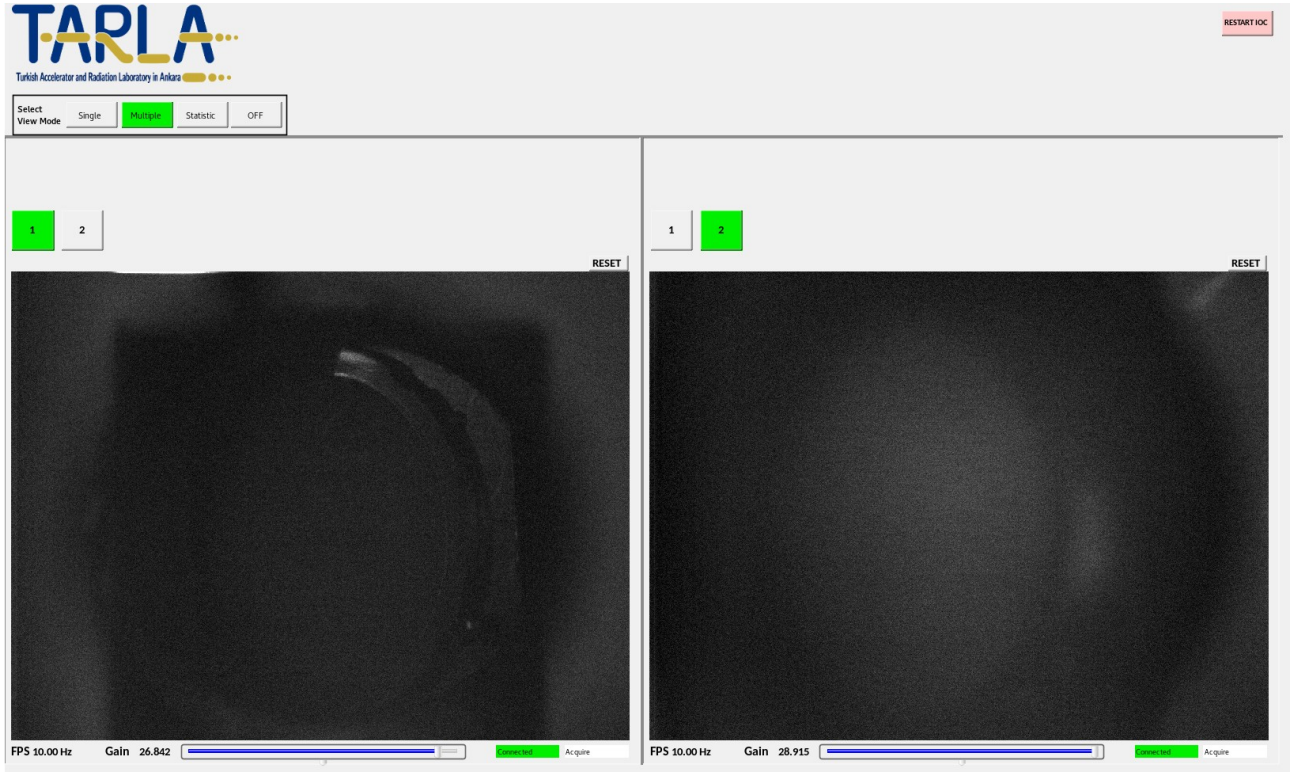

Figure 3.5 Multiple Mode user interface

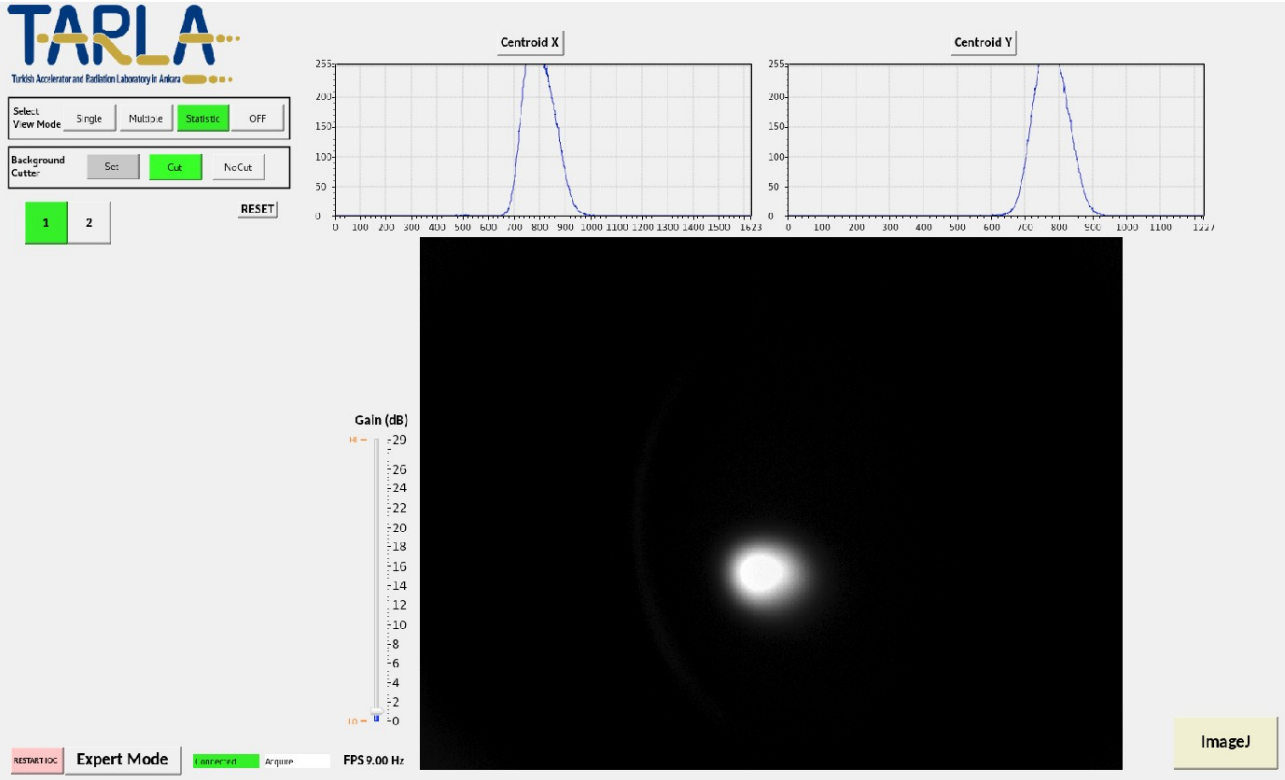

Figure 3.6 Statistic Mode user interface

4. References

1. Baumer VLG-20M Features, https://www.baumer.com/medias/__secure__/Baumer_VLG_20M_I_DS_EN.pdf?mediaPK=8798840651806
2. Experimental Physics and Industrial Control System, https://epics.anl.gov/index.php
3. areaDetector Overview, https://cars9.uchicago.edu/software/epics/areaDetectorDoc.html#Overview
4. Motif Editor and Display Manager, https://epics.anl.gov/extensions/medm/index.php
5. EPICS extensions, https://epics.anl.gov/extensions/index.php
6. Control System Studio, http://controlsystemstudio.org/